Grado en Tecnologías de Telecomunicación

Escuela Politécnica Superior de Linares

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DESARROLLO DE UN SISTEMA DE RENDERIZADO AUDIOVISUAL DE LA ESCENA SONORA BASADO EN SEPARACIÓN DE FUENTES

Alumno: **Daniel Martos Pancorbo**

Tutor: Julio José Carabias Orti
Depto.: Ingeniería de Telecomunicación

Tutor: Pedro Vera Candeas
Depto.: Ingeniería de Telecomunicación

Septiembre, 2019

RESUMEN

En una actualidad, en la que la realidad virtual está en plena expansión, surge la idea y motivación de este proyecto, en el que se busca fusionar el mundo de los videojuegos con el de la música.

Se centrará el desarrollo de este proyecto en la realización de una escena de realidad virtual en la que el usuario pueda revivir un concierto o evento musical, interaccionando con las fuentes que se encuentren en él.

Para la realización de la experiencia virtual se usará unas gafas de realidad virtual como soporte visual y se utilizarán los principios de la síntesis binaural, para obtener la sensación de audio 3D.

Otro de los aspectos del proyecto a desarrollar, es la interacción del usuario con las fuentes sonoras, para adquirir un mayor realismo de la escena. Para ello, se utilizarán técnicas de separación de fuentes basadas en Non-Negative Matrix Factorization (NMF), muy útiles en muchas aplicaciones de procesado de señal musical, para así, obtener y crear con fuentes aisladas esa sensación de realismo de la escena virtual.

Por último, se integrarán los resultados obtenidos del procesamiento de señal, en la aplicación donde se encuentre la escena virtual, obteniendo así, por tanto, el objetivo final de este proyecto.

AGRADECIMIENTOS

Agradecer a todas las personas que me han acompañado durante estos últimos 4 años de carrera y me han ayudado a poder terminar, con este proyecto, esta etapa tan increíble.

A mis padres y hermana, a mis abuelos, a Rocio, a mis amigos, a todos y cada uno de los compañeros que me llevo de estos 4 años, por crear tan buenos momentos y anécdotas. Gracias.

Agradecer, por último, también a mis tutores de proyecto por darme la oportunidad de realizarlo con ellos y por ayudarme a sacar el máximo partido de mí.

ÍNDICE DE FIGURAS Y TABLAS

Figura 1: Estructura del proyecto 12

Figura 2: Proceso de mezcla 13

Tabla 1: Tipos de modelos de mezcla 14

Figura 3: Procesos de mezcla según número de micrófonos y fuentes……...13

Figura 4: Matriz de ganancias para el caso de la trompeta 18

Figura 5: Espectro de la nota 22 de la trompeta 19

Figura 6: Espectro de la nota 32 de la trompeta 19

Figura 7: Excitaciones del modelo glotal 24

Figura 8: Filtros de fonema 25

Figura 9: Diagrama de flujo del proceso de separación de fuentes 33

Figura 10: Resultados para la parte vocal 36

Figura 11: Resultados para la parte de acompañamiento 37

Figura 12: Resultados para la parte de percusión 38

Figura 13: Resultados para la parte del bajo 39

Figura 14: Resultados para la parte del residuo armónico 40

Figura 15: Efecto phantom estéreo 43

Figura 16: Técnica de grabación Blumlein 45

Figura 17: Otros métodos de grabación estereofónica (X/Y, A/B y Mid-Side) 46

Figura 18: Método VBAP 47

Figura 19: Cortina acústica 47

Figura 20: Aplicación del principio de Huygens 48

Figura 21: Formato básico de Ambisonics 49

Figura 22: Ambisonics de primer orden 49

Figura 23: Principio de síntesis binaural Figura 24: Dummy head 50

Figura 25: Sistemas anáglifos 52

Figura 26: Sistemas con gafas polarizadas 52
Figura 27: Pantalla de visionado para VR 53
Figura 28: Interfaz gráfica de Adobe Fuse 57
Figura 29: Interfaz gráfica de Blender 58
Figura 30: Plataforma de Mixamo 58
Figura 31: Ajuste de atenuación de la fuente 59
Figura 32: Modelo de atenuación de la fuente sonora 60
Figura 33: Iluminación de la escena virtual 60
Figura 34: Creación de proyecto en Unreal Engine 63
Figura 35: First Person Character en la escena virtual 64
Figura 36: Entorno gráfico de programación Unreal Engine 64
Figura 37: Evento del “First Person Character” 65
Figura 38: Ajustes del "First Person Character" 65
Figura 39: Plugins a instalar en Unreal Engine 66
Figura 40: Ajustes del sistema operativo del dispositivo móvil 67
Figura 41: Ajustes del proyecto para VR 67
Figura 42: Lanzamiento e instalación de la aplicación en el dispositivo móvil 68
Figura 43: Esquema del sistema VR 69

ÍNDICE

RESUMEN 3
AGRADECIMIENTOS 5
ÍNDICE DE FIGURAS Y TABLAS 7
ÍNDICE 9
1. INTRODUCCIÓN 11
1.1. MOTIVACIÓN Y OBJETIVOS 11
1.2. ORGANIZACIÓN 11
2. SEPARACIÓN DE FUENTES 13
2.1. MODELOS DE MEZCLA 14
2.2. MODELO BÁSICO NMF 15
2.3. APRENDIZAJE DE FUNCIONES BASE PARA INSTRUMENTOS MUSICALES 17
2.4. MODELO DE SEPARACIÓN ARMÓNICO-PERCUSIVA 19
2.4.1. COSTE PERCUSIVO 20
2.4.2. COSTE ARMÓNICO 21
2.4.3. ALGORITMO ARMÓNICO-PERCUSIVO NMF 22
2.5. MODELO FILTRO-FUENTE PARA VOZ CANTADA 23
2.6. FASE DE UNMIXING (Filtro de Wiener) 26
2.7. MEDIDAS DE EVALUACIÓN 27
2.8. BASES DE DATOS MUSICALES PARA SEPARACIÓN DE FUENTES 30
2.8.1. RWC Musical Instrument Sound Database 31
2.8.2. OBTENCIÓN DE SEÑALES MUSICALES UTILIZADAS EN LA DEMO 31
2.9. MODELO DE SEPARACIÓN VOZ/ARMÓNICO/PERCUSIVA PROPUESTO 32
2.10. RESULTADOS OBTENIDOS 35
3. AUDIO 3D 43
3.1. INTRODUCCIÓN 43
3.2. EFECTO PHANTOM 43
3.3. TÉCNICAS DE AUDIO ESPACIAL 47
3.3.1. WAVE FIELD SYNTHESIS 47

3.3.2. AMBISONICS 48
3.4. SÍNTESIS BINAURAL 50
4. REALIDAD VIRTUAL 51
4.1. INTRODUCCIÓN 51
4.2. UNREAL ENGINE 53
4.2.1. APLICACIONES DE UNREAL ENGINE 53
4.2.2. UNREAL ENGINE VS UNITY 54
4.3. DESARROLLO DE LA ESCENA VIRTUAL 55
1. DISEÑO DE LA ESCENA VIRTUAL 55
2. RECREACIÓN DE LA ESCENA VIRTUAL E IMPORTACIÓN DE MODELOS 3D 55
3. IMPORTACIÓN DE PERSONAJES Y ANIMACIONES A LA ESCENA VIRTUAL 56
4. MODELO DE ATENUACIÓN DE LAS FUENTES 58
5. ILUMINACIÓN DE LA ESCENA VIRTUAL 60
5. APLICACIÓN ANDROID PARA REALIDAD VIRTUAL 63
5.1. PROYECTO UNREAL EN PRIMERA PERSONA 63
5.2. ENCAPSULACIÓN DE LA APLICACIÓN 66
5.3. INTEGRACIÓN DE LA APLICACIÓN EN UN SISTEMA DE AUDIO Y VIDEO 68
5.4. LIMITACIONES Y FUTURAS MEJORAS 69
6. CONCLUSIONES 71
7. BIBLIOGRAFÍA 73

1. INTRODUCCIÓN

En la actualidad, el continuo uso de los dispositivos móviles, nos ha llevado a cambiar nuestra forma de vida, de tal manera que, para hacer cualquier actividad (ya sea de ocio o profesional) no sea una excusa el desplazamiento o la falta de tiempo, sino que, en unos segundos, tengamos en nuestra mano la solución para realizar todo aquello que antes no podíamos hacer.

De esta manera, si cualquier persona pudiera tener la posibilidad de consultar su cuenta bancaria sin tener que ir al cajero de un banco o pudiera ver una película sin tener que ir al cine, ¿por qué no podría caber la posibilidad de poder disfrutar de un concierto en vivo con la percepción y la sensación de vivirlo en ese mismo instante?

Por ello, con la ayuda de la separación de fuentes y la realidad virtual, se diseñará una aplicación que nos permita revivir eventos musicales que por ciertas circunstancias ya no se podrán volver a experimentar y de esta manera se podrá disfrutar de ellos en el momento que se quiera, manteniendo, en la medida de lo posible, el realismo sonoro que se tendría si se estuviera viviendo en vivo en ese mismo instante.

1.1. MOTIVACIÓN Y OBJETIVOS

La motivación principal para este proyecto, es la realización de una aplicación de realidad virtual para teléfonos Android, en la que el usuario pueda revivir un evento musical (por ejemplo, un concierto) interaccionando con la escena y disfrutando del realismo y la sensación sonora que proporciona tanto la separación de fuentes como la distribución de dichas fuentes por la escena.

Si se tuviera, un sistema con el que se pudiera realizar el proceso de separación de fuentes, se podría construir y recrear una escena acústica virtual a partir de una sola señal monoaural.

Por lo tanto, el objetivo del presente proyecto, es implementar un sistema de separación de fuentes sonoras para señales monoaurales y poder renderizarlas mediante la creación de un sistema de realidad virtual para audio y video.

Dicha separación de fuentes, es realizada de manera separada a la aplicación móvil, debido a la pequeña capacidad de la que disponen dichos terminales a la hora de realizar procesado en tiempo real. La separación consiste en un algoritmo que, partiendo de una sola pista de audio, es capaz de extraer las distintas fuentes armónicas de cada uno de los instrumentos que suenan en ella.

1.2. ORGANIZACIÓN

El presente proyecto se dividirá en 4 secciones principalmente:

Parte 1: Se indicará la introducción, así como el objeto del proyecto y el estado del arte.

Parte 2: Se describirán todos los conceptos previos y el proceso relativo a la separación de fuentes.

Parte 3: Se explicarán las técnicas y sistemas de Audio 3D que se han desarrollado a lo largo de los tiempos.

Parte 4: Se desarrollará el proceso que se ha llevado a cabo para la construcción de la escena virtual, así como un resumen del estado del arte relativo a la realidad virtual.

Parte 5: Se indicarán los resultados alcanzados y las conclusiones obtenidas, así como futuras mejoras para el desarrollo de la aplicación.

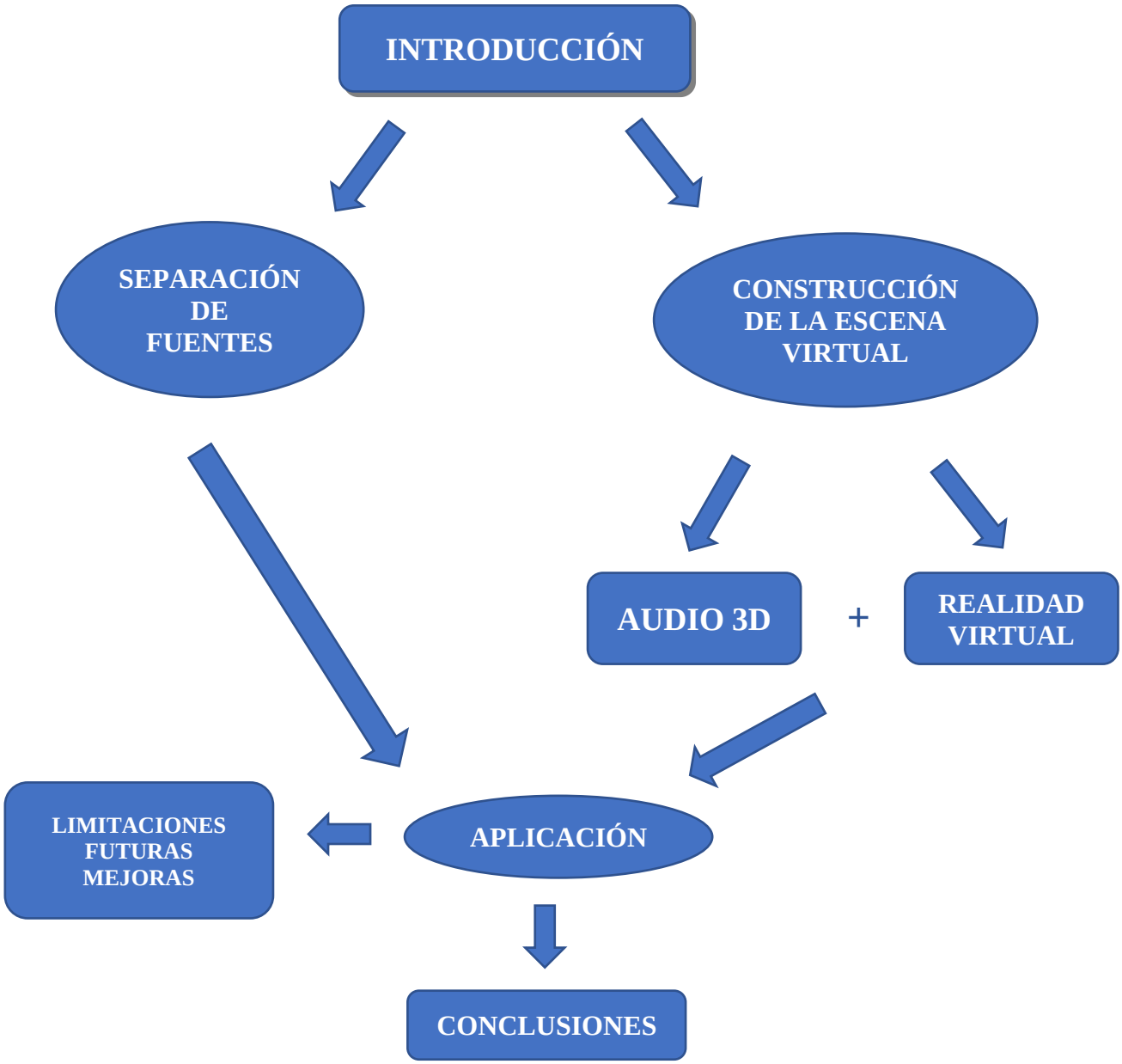


Figura 1: Estructura del proyecto

2. SEPARACIÓN DE FUENTES

En este capítulo, se van a introducir los conceptos básicos y necesarios para el entendimiento del proceso de separación de fuentes, así como los sistemas y modelos que se utilizan para su correcta ejecución. En primer lugar, se expondrán los diferentes modelos de mezcla de audio que existen, a continuación, se introducirá el proceso de aprendizaje de los modelos de instrumento, así como los modelos armónico-percusivo, filtro-fuente para voz y filtro de Wiener, usados en este proceso. Por último, se explicarán las medidas de evaluación y las bases de datos musicales utilizadas, así como el desarrollo y descripción del proceso de separación de fuentes, y la exposición de los distintos resultados obtenidos.

El concepto de separación de fuentes es un concepto aplicado a una gran variedad de ámbitos, como puede ser el video o la biomedicina, y no solo al audio como muchos piensan. En este caso, se centrará el estudio en la separación de fuentes en el campo del audio, ya que dicha separación puede ser generada por la necesidad, entre otras muchas razones, de eliminar la parte vocal de una canción para un karaoke, minimizar el ruido que se encuentra presente en grabaciones o extraer una determinada información de una fuente sonora a estudiar.

Para todos los casos, se parte de una premisa, la cual consiste en que cualquier audio que se vaya a estudiar tiene que contener la mezcla de dos o más elementos mezclados y, dicha mezcla, coincidirá en un punto del espacio y en un instante determinado, cumpliendo el teorema de la superposición, cuya expresión matemática es:

$$(v_i[n])_{1\leq l\leq L} = f((x_m[n])_{1\leq m\leq M}) \qquad (2.1)$$

siendo $v_i[n]$ la suma de las distintas señales procedentes de los diferentes instrumentos y recogidas por un micrófono, L la cantidad de micrófonos y $x_m[n]$ el sonido que emite cada instrumento, siendo así M el número total de instrumentos que se tienen.

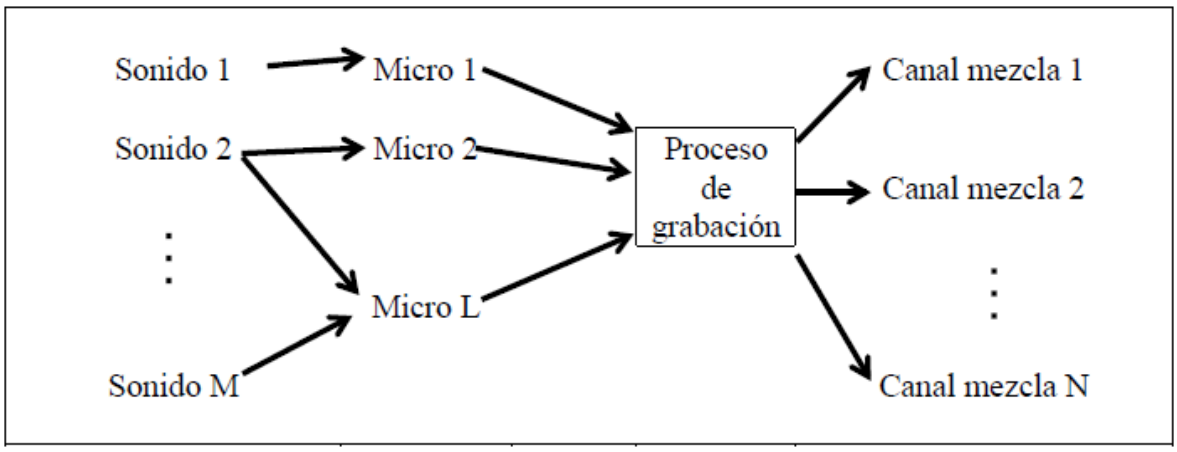


Figura 2: Proceso de mezcla

2.1. MODELOS DE MEZCLA

Antes de que se produzca el proceso de separación de fuentes, tiene lugar otro proceso denominado proceso de mezcla, en el cual, dos o más señales son recogidas y añadidas en una grabación dando lugar a una nueva señal. Existen bastantes modelos de mezcla, los cuales se clasifican atendiendo a una serie de criterios, como el número de micrófonos y fuentes, el entorno en el que nos encontramos, o el aspecto creativo y estético de la mezcla.

En este caso, se estudiará el modelo de mezcla según el entorno en el que se vaya a realizar la mezcla [Bernalte, 14], de esta manera se tendrán:

- Entornos en los que el camino que recorre el sonido desde la fuente al micrófono no es el camino directo, sino que se produce un efecto de reverberación. En este caso se utilizan **mezclas convolutivas**.
- Entornos en los que no se producen reverberaciones y el sonido sigue el camino directo desde la fuente al micrófono. En este caso se utilizan **mezclas no convolutivas o instantáneas**, que consisten en la combinación lineal de las diferentes fuentes junto con una ganancia "a" propia para cada fuente.

    - Dentro de las mezclas instantáneas, se pueden encontrar las mezclas anecoicas o retrasadas, que se caracterizan porque además de las diferentes ganancias se añaden a su vez, retrasos de retransmisión entre las fuentes y los micrófonos.

A continuación, se adjunta una tabla donde se describen, en primer lugar, la expresión matemática que describe cada modelo de mezcla, en segundo lugar, la matriz de mezcla, y por último una representación gráfica del proceso de mezcla [Sarmiento, 17].

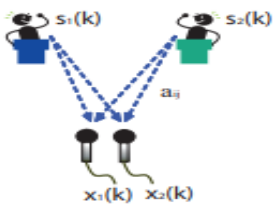
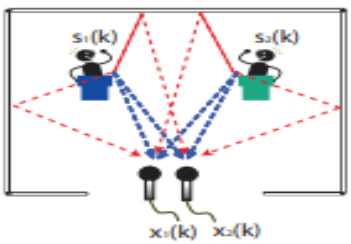
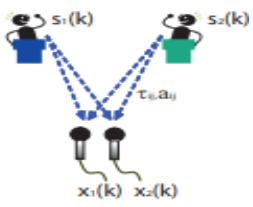
MEZCLA INSTANTANEA	MEZCLA CONVOLUTIVA	MEZCLA ANECOICA
$x_i(k) = \sum_{j=1}^{N} a_{ij} s_j(k)$	$x_i(k) = \sum_{j=1}^{N} \sum_{l=0}^{L-1} a_{ij}(l) s_j(k-l)$	$x_i(k) = \sum_{j=1}^{N} a_{ij} s_j(k-\tau_{ij})$
A= $\begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{L1} & \cdots & a_{LN} \end{bmatrix}$	$A = \begin{bmatrix} \sum_{k=1}^{K} a_{11k} x_n(t-\delta_{11k}) & \cdots & \sum_{k=1}^{K} a_{1Nk} x_n(t-\delta_{1Nk}) \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{K} a_{L1k} x_n(t-\delta_{L1k}) & \cdots & \sum_{k=1}^{K} a_{LNk} x_n(t-\delta_{LNk}) \end{bmatrix}$	$A = \begin{bmatrix} a_{11}\delta(t-\delta_{11}) & \cdots & a_{1N}\delta(t-\delta_{1N}) \\ \vdots & \ddots & \vdots \\ a_{L1}\delta(t-\delta_{L1}) & \cdots & a_{LN}\delta(t-\delta_{LN}) \end{bmatrix}$
		

Tabla 1: Tipos de modelos de mezcla

Además, los modelos de mezcla pueden ser variantes o invariantes en el tiempo, con presencia o no de ruido aditivo o sin él, y dependiendo del número de fuentes y micrófonos, podemos distinguir entre procesos de mezcla: sobredeterminado (más micrófonos que fuentes), determinado (igual número de micrófonos y fuentes) o indeterminado (más fuentes que micrófonos).

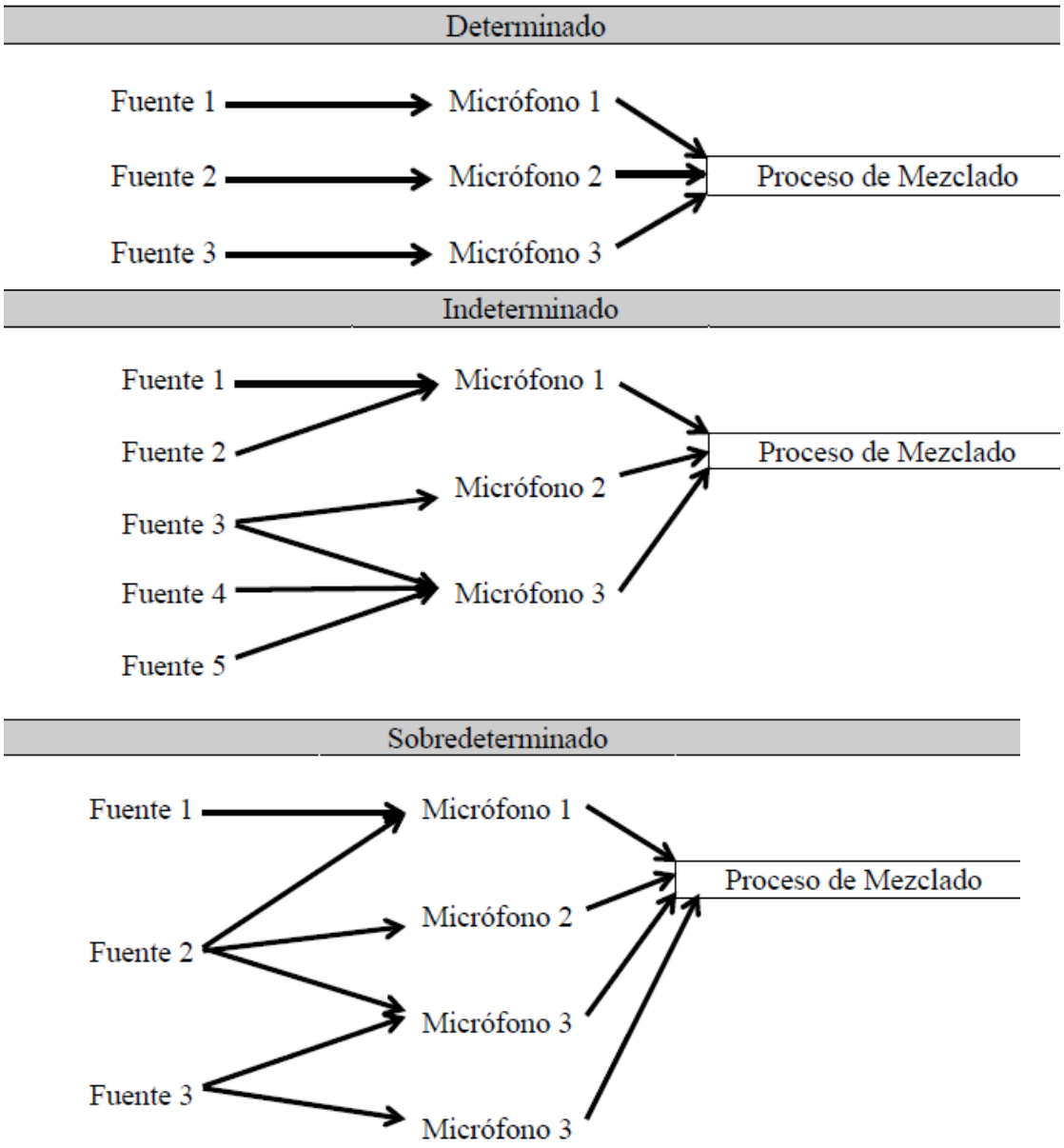


Figura 3: Procesos de mezcla según número de micrófonos y fuentes

2.2. MODELO BÁSICO NMF

El método de descomposición *Non-Negative Matrix Factorization* (NMF) [Rodriguez, 14] se encarga de representar datos en dos dimensiones como una combinación lineal de una serie de elementos básicos representativos. NMF es el problema de encontrar una factorización de la forma:

$$X \approx BG = \hat{X} \tag{2.1}$$

siendo X una matriz de dimensiones FxT y, B y G matrices de dimensiones FxN y NxT, respectivamente, conteniendo cada una de ellas valores no negativos. Normalmente, N toma valores pequeños, de manera que las dimensiones de las matrices a tratar son reducidas.

Este método de descomposición ha sido aplicado a numerosas aplicaciones como el aprendizaje de patrones de rostros o la transcripción automática de música polifónica.

Cuando este método, es utilizado en aplicaciones destinadas a la música, el espectro de potencia o amplitud, se correspondería con la matriz X (representación tiempo-frecuencia), en la cual se encuadran los *bins* frecuenciales *f* de cada trama temporal *t*.

El objetivo de NMF consiste en obtener las matrices no negativas B y G, con el menor coste posible, es decir:

$$(B, G) = \arg \min_{B,G>0} D(X|BG) \tag{2.2}$$

siendo $D(X|BG)$ una función de coste que se define de la siguiente manera:

$$D(X|BG) = \mathrm{D}(\mathrm{X}|\hat{X}) = \sum_{f=1}^{F} \sum_{t=1}^{T} d(X(f,t)|\hat{X}(f,t)) \tag{2.3}$$

donde $d(x|\hat{x})$ es una función de coste escalar. A continuación, se muestran algunos ejemplos de funciones de coste que son bastante conocidas:

- Distancia Euclídea: $d_{EUC}(x|y) = \frac{1}{2}(x-y)^2$
- Divergencia Kullback-Leibler (KL): $d_{KL}(x|y) = x \log \frac{x}{y} - x + y$

Estas funciones son positivas y toman valor cero si $x = y$.

Las reglas de actualización de NMF se caracterizan por su simplicidad y han provocado que NMF sea uno de los métodos de descomposición más utilizados en las aplicaciones anteriormente mencionadas.

En el modelo básico NMF, la única restricción, por tanto, es la no negatividad de los elementos que contienen cada una de las matrices que participan en este método. El resto de propiedades son solo efectos adicionales y propios de él mismo. Este algoritmo es capaz de obtener información sobre la señal de entrada, obtener datos interpretables, e incluso desarrollar una separación, pero todo esto no lleva a que se pueda tomar como una herramienta mágica. Los autores de otras variantes a este método, comparan la falta de control en la descomposición, introduciendo unas ciertas restricciones que, como consecuencia, dan lugar a variantes de este modelo NMF básico. Algunos ejemplos de restricciones pueden ser la dispersión, la localización espacial o incluso la continuidad temporal.

Una variante típica al modelo básico NMF, consiste en agregar un término de penalización o regulación a la función de coste y minimizarla, pero esta variante produce algunos inconvenientes como, por ejemplo: se tiene que establecer algún criterio que

permita cuantificar la propiedad a limitar o penalizar y, además, tampoco se asegura la convergencia para el esquema evolutivo del modelo. A todo esto, hay que sumar que el término de penalización, se debe de obtener de forma empírica, lo que complica mucho el proceso. Por ello, muchos de los autores han aplicado la restricción de no negatividad a otros esquemas.

En los esquemas NMF también es necesario disponer de una ordenación de las bases, de manera que facilite la práctica de este método. Aplicando una restricción de armonicidad a cada una de las bases, se puede relacionar cada pitch con cada una de ellas, dando un papel más coherente tanto a la base como a las ganancias que le correspondan.

2.3. APRENDIZAJE DE FUNCIONES BASE PARA INSTRUMENTOS MUSICALES

Cuando se dispone de datos de entrenamiento apropiados, es posible realizar un aprendizaje de patrones espectrales limitando la reconstrucción del espectrograma individual de cada fuente a la estimación de sus ganancias. Se ha demostrado [Carabias, 13] que este enfoque proporciona buenos resultados cuando las condiciones de la escena musical no difieren mucho entre el entrenamiento y los datos de prueba.

En primer lugar, las amplitudes de cada nota y de cada instrumento musical, $a_{j,n}(h)$, se aprenden de antemano utilizando la base de datos musical *Real World Computing (RWC*), que almacena datos de entrenamiento de instrumentos solistas que tocan notas aisladas y, en la cual, se profundizará en el apartado 2.8 de esta memoria. La transcripción de los datos de entrenamiento se representará mediante $R_{j,n}(t)$, como una matriz tiempo-frecuencia para cada instrumento *j*, representando en la dimensión de la frecuencia, la escala de la interfaz digital del instrumento musical (*MIDI*) y, en la dimensión del tiempo, las tramas. En la etapa de entrenamiento, las ganancias se inicializan con $R_{j,n}(t)$, conocida de antemano por la base de datos. Por ello, las ganancias se ponen a uno, para cada tono, en el periodo de tiempo en el que el instrumento esté activo generando dicho tono, mientras que el resto de ganancias se pondrá a cero. Las ganancias inicializadas a cero permanecen en cero debido a las reglas de actualización multiplicativas y, por lo tanto, se restringe la dispersión de los valores aprendidos, obteniéndose un único patrón espectral para la nota entrenada. Esta propiedad es conocida en el argot de procesado de señal como *sparsity*.

Las reglas de actualización multiplicativas que minimizan la *β-divergencia* para los parámetros del modelo se calculan de la siguiente manera:

$$a_{j,n}(h) \leftarrow a_{j,n}(h) \frac{\sum_{f,t} x(f,t)\hat{x}(f,t)^{\beta-2} g_{j,n}(t) G(f - hf_o(n))}{\sum_{f,t} \hat{x}(f,t)^{\beta-1} g_{j,n}(t) G(f - hf_o(n))} \tag{2.4}$$

$$g_{j,n}(t) \leftarrow g_{j,n}(t) \frac{\sum_{f,m} x(f,t)\hat{x}(f,t)^{\beta-2} a_{j,n}(t) G(f - hf_o(n))}{\sum_{f,m} \hat{x}(f,t)^{\beta-1} a_{j,n}(t) G(f - hf_o(n))} \tag{2.5}$$

El proceso de entrenamiento se puede resumir en el siguiente algoritmo:

1. Calcular $X(f,t)$ a partir de la interpretación solista de cada instrumento en la base de datos de entrenamiento.
2. Inicializar las ganancias $g_{j,n}(t)$ con la transcripción $R_{j,n}(t)$ y $a_{j,n}(h)$ con valores aleatorios positivos.
3. Actualización de las ganancias usando la ecuación 2.4.
4. Actualización de las bases usando la ecuación 2.5.
5. Repetir los pasos del 2 al 3 hasta que el algoritmo converja o bien hasta que se realicen el número máximo de iteraciones (en nuestro caso, 200 iteraciones)
6. Calcular las funciones base $b_{j,n}(f)$ para cada instrumento usando la ecuación:

$$b_{j,n} = \sum_{h=1}^{H} a_{j,n}(h)\, G(f - hf_o(n)) \tag{2.6}$$

El algoritmo de entrenamiento calcula las funciones base requeridas en la etapa de factorización para cada instrumento. Estas funciones dependientes del instrumento son conocidas y se mantienen fijas, por lo que la factorización de nuevas señales del mismo instrumento se puede reducir a la estimación de las ganancias $g_{j,n}(t)$.

A continuación, se representa la matriz de ganancias, para el caso de la trompeta, la cual presenta 37 notas en la escala. Además, se representan los espectros de las notas 22 y 32, donde se pueden apreciar los correspondientes armónicos de cada una de las notas:

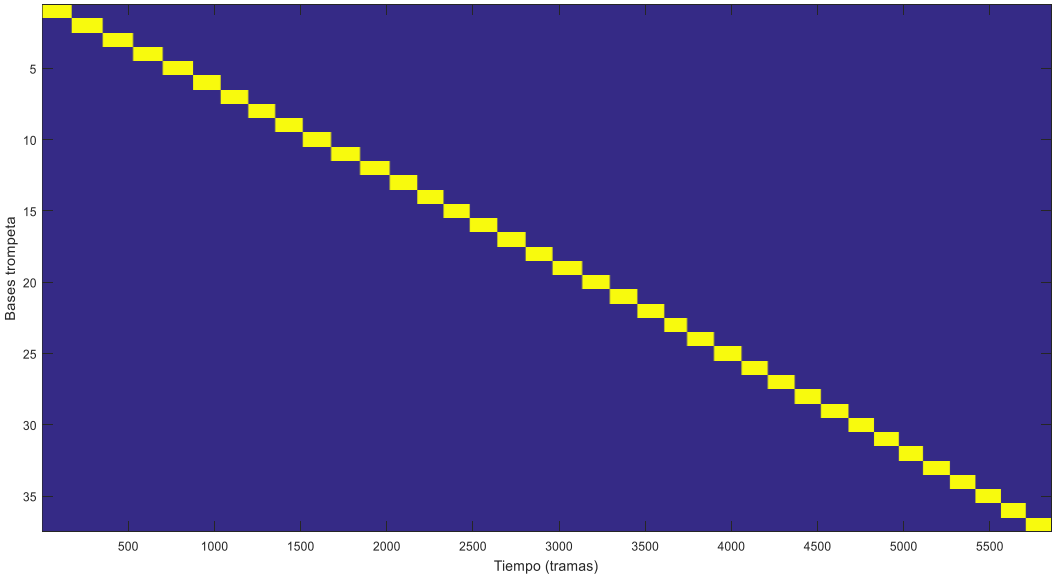


Figura 4: Matriz de ganancias para el caso de la trompeta

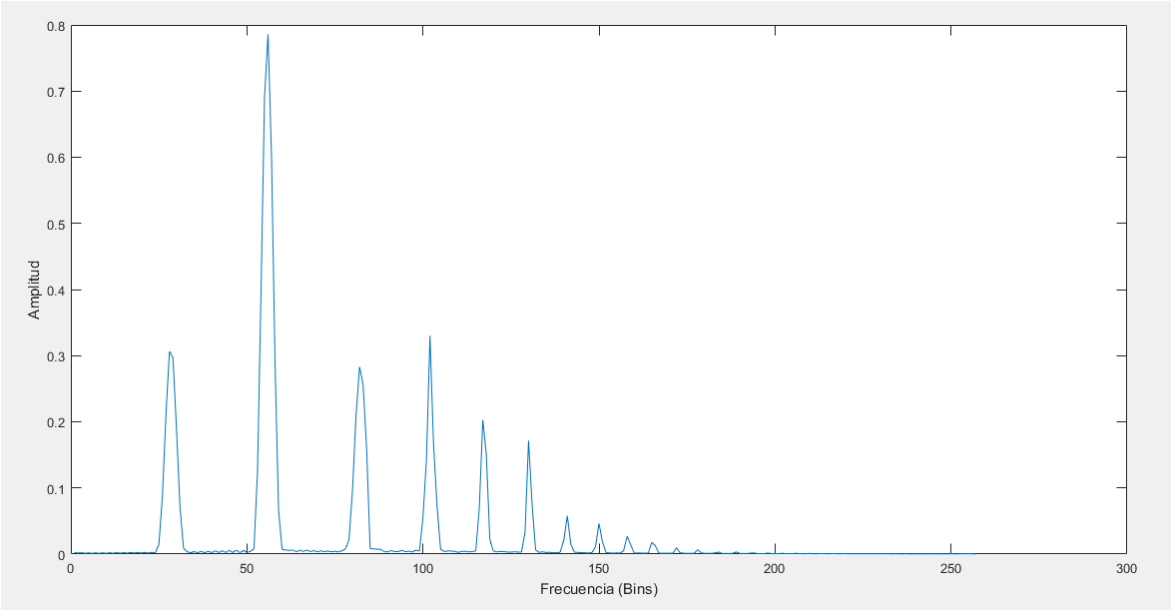


Figura 5: Espectro de la nota 22 de la trompeta

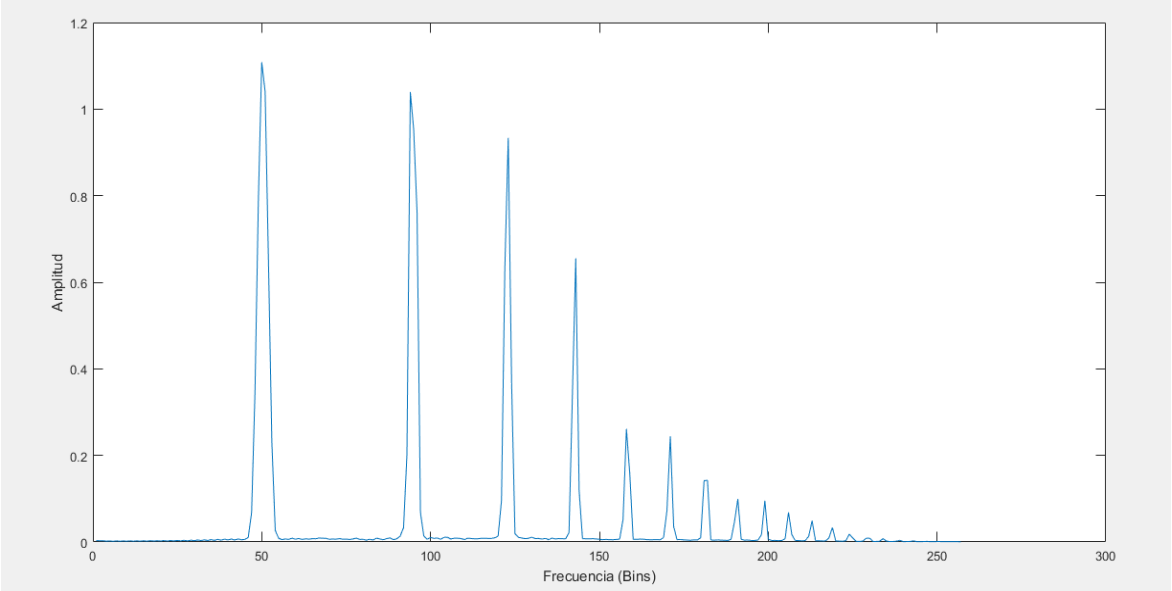


Figura 6: Espectro de la nota 32 de la trompeta

2.4. MODELO DE SEPARACIÓN ARMÓNICO-PERCUSIVA

Para superar el problema inicial del enfoque estándar que se le da a NMF, se procede [Cañadas, 14] a distinguir las bases percusivas y armónicas en el proceso de factorización. Para conseguir este objetivo, se define una función que descomponga el espectrograma de la mezcla $X_{n\beta}$ en dos espectrogramas separados: X_p (espectrograma percusivo) y X_H (espectrograma armónico). Cada uno de ellos presentas características espectro-

temporales para los sonidos armónicos y percusivos. Esta factorización se puede formular de la siguiente manera:

$$X_{n\beta} = X_p + X_H = W_{P_{F,Rp}} W_{P_{Rp,T}} + W_{H_{F,Rh}} W_{H_{Rh,T}} \quad (2.7)$$

donde cada una de las matrices presenta valores no negativos, *Rp* representa el número de bases percusivas y *Rh* el número de bases armónicas.

El proceso de descomposición adapta el concepto de "*Anisotropy"* expuesto en [Cañadas,14], el cual se utiliza para estimar las matrices W_P, H_P, W_H y H_H, minimizando la función global de coste que depende del coste de la β-divergencia, del coste percusivo y del coste armónico.

Los espectrogramas X_P y X_H se construyen para minimizar el coste de la β-divergencia en comparación con el espectrograma normalizado de entrada $X_{n\beta}$. La distancia Euclídea (β=2), la divergencia de Kullback-Leibler (KL) (β=1) y la divergencia de Ikatura-Saito (IS) (β=0), son definidos como funciones del parámetro β, como se muestra a continuación:

$$d_\beta(x|y) = \begin{cases} \frac{1}{\beta(\beta-1)} \sum_{f=1}^{F} \sum_{t=1}^{T} (x_{f,t}^{\beta} + (\beta-1) y_{f,t}^{\beta} - \beta x_{f,t} y_{f,t}^{\beta-1}) & \beta \epsilon (0,1) \cup (1,2] \\ \sum_{f=1}^{F} \sum_{t=1}^{T} x_{f,t} \log \frac{x_{f,t}}{y_{f,t}} - x_{f,t} + y_{f,t} & \beta = 1 \\ \sum_{f=1}^{F} \sum_{t=1}^{T} \frac{x_{f,t}}{y_{f,t}} - \log \frac{x_{f,t}}{y_{f,t}} - 1 & \beta = 2 \end{cases}$$

siendo $x = X_{n\beta}$ e $y = X_P + X_H$.

Esta factorización solo considera el coste por β-divergencia y no puede determinar si una cierta base pertenece a una señal percusiva o armónica. Por ello, para solucionar este problema, el proceso de factorización se modifica incluyendo características espectro-temporales específicas de los sonidos armónicos y percusivos. Por lo tanto, a la función de coste total, se añaden otras dos funciones de coste como son la percusiva y la armónica.

2.4.1. COSTE PERCUSIVO

El coste percusivo se utiliza para calcular el coste de modelos de sonidos percusivos que se caracterizan por ser "*smoothness*" en frecuencia y "*sparseness*" en el tiempo. Estos dos términos se traducirán, de aquí en adelante, como suavidad y dispersión o escasez, respectivamente. Los sonidos percusivos suelen representarse como sonidos de banda ancha que se confinan en un periodo muy corto del tiempo. La suavidad espectral (SSM), la cual se asocia a la matriz *Wp*, se define de la siguiente manera:

$$SSM = \frac{T}{R_p} \sum_{r_p}^{R_p} \frac{1}{\sigma w_{P_{r_p}}{}^2} \sum_{f=2}^{F} (W_{P_{f-1,r_p}} - W_{P_{f,r_p}})^2 \quad (2.8)$$

Se atribuye un alto coste cuando se producen grandes cambios en frecuencia entre las bases $W_{P_{f,r_p}}$ y $W_{P_{f-1,r_p}}$, en *bins* adyacentes de frecuencia. Cabe destacar que se utiliza la normalización, ya que es conveniente que la función de coste proporcione un valor real independientemente de la señal normalizada. En este caso, las bases W_P están normalizadas por el término σw_P, cuyo valor es calculado como $\sigma w_{P_{r_P}} = \sqrt{\frac{1}{F}\sum_{f=1}^{F} W_{P_{f,r_p}}^2}$. De manera que cada coste armónico o percusivo tenga el mismo peso en la función global de coste, este será normalizado. En este caso, el valor SSM es normalizado por un valor de $\frac{T}{R_p}$. Así, se pueden evitar grandes problemas que pueden surgir, fruto de un incremento del número de tramas, del número de *bins* de frecuencia o incluso del número de componentes considerado.

Otra restricción del coste percusivo es la asociada a la distribución temporal de las activaciones. El concepto de escasez temporal (TSP) es definido siguiendo la siguiente ecuación:

$$\left[\frac{\partial \mathrm{SSM}}{\partial W_P}\right]_{f,r_p}^{+} = \frac{4FW_{P_{f,r_p}}}{\sum_{j=1}^{F} W_{P_{j,r_p}}^2} \tag{2.9}$$

Pero a diferencia de SSM, esta restricción es aplicada a la matriz Hp. Para este concepto, un alto coste se traduce en ganancias no negativas asumiendo que los sonidos percusivos pueden ser representados como transitorios cuya energía, está concentrada en cortos periodos de tiempo.

$$TSP = \frac{F}{R_p}\sum_{r_p=1}^{R_p}\sum_{t=1}^{T}\left|\frac{H_{P_{r_p,t}}}{\sigma H_{P_{r_p}}}\right| \tag{2.10}$$

De igual manera, que se hizo para *Wp*, las activaciones de *Hp* se normalizarán por un valor igual a $\sigma H_{P_{r_P}} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} H_{P_{r_p,t}}^2}$. Además, TSP se normaliza multiplicando por un factor de $\frac{F}{R_p}$ para que tenga el mismo peso en el coste percusivo.

2.4.2. COSTE ARMÓNICO

Por otro lado, el coste armónico es aplicado para el modelo de sonidos armónicos, caracterizándose por presentar una suavidad temporal y ser disperso en frecuencia. Los sonidos armónicos son considerados sonidos estables porque presentan poca variación de amplitud en el tiempo con mucha de su energía concentrada en picos espectrales. La suavidad temporal (TSM) está asociada con la matriz de activaciones H_H y es utilizada para asignar un alto coste, cuando se producen grandes cambios entre dos ganancias de tramas adyacentes.

$$TSM = \frac{F}{R_h}\sum_{r_h=1}^{R_h}\frac{1}{\sigma H_{H_{r_h}}{}^2}\sum_{t=2}^{T}(H_{H_{r_h,t-1}} - H_{H_{r_h,t}})^2 \tag{2.11}$$

Para este coste armónico se vuelve a utilizar la normalización por la misma razón que se describió en el coste percusivo. Por ello, la matriz H_H de ganancias se normalizará con el valor $\sigma H_{H_{r_h}} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} H_{H_{r_h,t}}^2}$, y el coste armónico TSM se normalizará con un factor de $\frac{F}{R_h}$.

En el caso de la dispersión espectral, se aplicará a la matriz W_H y, se definirá de la siguiente manera:

$$SSP = \frac{T}{R_h}\sum_{r_h=1}^{R_h}\sum_{f=1}^{F}\left|\frac{W_{H_{f,r_h}}}{\sigma W_{H_{r_h}}}\right| \tag{2.12}$$

De igual manera que se ha realizado la normalización de H_H, la matriz de bases W_H se normalizará mediante un valor igual a $\sigma W_{H_{r_h}} = \sqrt{\frac{1}{F}\sum_{f=1}^{F} W_{H_{f,r_h}}^2}$, y el coste armónico SSP será normalizado con un factor de $\frac{T}{R_h}$.

2.4.3. ALGORITMO ARMÓNICO-PERCUSIVO NMF

Por lo tanto, la función global de coste, una vez se ha incluido tanto la *β-divergencia* como los costes armónico y percusivo, se define de la siguiente manera:

$$D = d_\beta\left(X_{n\beta}\middle|(X_P + X_H)\right) + K_{SSM}SSM + K_{TSP}TSP + K_{TSM}TSM + K_{SSP}SSP \tag{2.13}$$

Teniendo en cuenta una serie de estudios preliminares, se llega a la conclusión de que $K_{SSM} = K_{TSM}$, y que, $K_{TSP} = K_{SSP}$, representando cada *K* el peso de cada penalización, ya que para valores diferentes entre sí no se aprecian diferencias significativas. Para comprobar si la suavidad y la dispersión afectan al proceso de separación, se definirán 2 parámetros:

- $K_{SP} = K_{TSP} = K_{SSP}$, que representará los términos de suavidad (*smoothness*).
- $K_{SM} = K_{TSM} = K_{SSM}$, que representará los términos de dispersión (*sparseness*).

Se utilizarán las conocidas como reglas de actualización multiplicativas, en las cuales D no aumenta y, además, se garantiza la no negatividad de las bases y las ganancias. Estas reglas son implementadas mediante un algoritmo de descenso por gradiente, eligiendo apropiadamente el tamaño del paso, y siendo estimadas para cada parámetro escalar Z expresando las derivadas parciales de la función objetivo como una división de dos términos positivos:

$$Z = Z \odot \frac{\left[\frac{\partial \mathrm{D}}{\partial \mathrm{Z}}\right]^{-}}{\left[\frac{\partial \mathrm{D}}{\partial \mathrm{Z}}\right]^{+}} \tag{2.14}$$

siendo ⊙ un producto elemento a elemento.

Si se sustituye las bases percusivas W_P y las ganancias percusivas H_P en la anterior ecuación, se obtendrán las reglas de actualización multiplicativas para los sonidos percusivos:

$$W_P = W_P \odot \frac{\left[\frac{\partial d_\beta}{\partial W_P}\right]^- + K_{SSM}\left[\frac{\partial \mathrm{SSM}}{\partial W_P}\right]^-}{\left[\frac{\partial d_\beta}{\partial W_P}\right]^+ + K_{SSM}\left[\frac{\partial \mathrm{SSM}}{\partial W_P}\right]^+} \tag{2.15}$$

$$H_P = H_P \odot \frac{\left[\frac{\partial d_\beta}{\partial H_P}\right]^- + K_{TSP}\left[\frac{\partial \mathrm{TSP}}{\partial H_P}\right]^-}{\left[\frac{\partial d_\beta}{\partial H_P}\right]^+ + K_{TSP}\left[\frac{\partial \mathrm{TSP}}{\partial H_P}\right]^+} \tag{2.16}$$

Y, por otro lado, si se sustituyen las bases armónicas W_H y ganancias H_H, en la ecuación 2.14, se obtendrán las reglas de actualización multiplicativas para los sonidos armónicos:

$$W_H = W_H \odot \frac{\left[\frac{\partial d_\beta}{\partial W_H}\right]^- + K_{SSP}\left[\frac{\partial \mathrm{SSP}}{\partial W_H}\right]^-}{\left[\frac{\partial d_\beta}{\partial W_H}\right]^+ + K_{SSP}\left[\frac{\partial \mathrm{SSP}}{\partial W_H}\right]^+} \tag{2.17}$$

$$H_H = H_H \odot \frac{\left[\frac{\partial d_\beta}{\partial H_H}\right]^- + K_{TSM}\left[\frac{\partial \mathrm{TSM}}{\partial H_H}\right]^-}{\left[\frac{\partial d_\beta}{\partial H_H}\right]^+ + K_{TSM}\left[\frac{\partial \mathrm{TSM}}{\partial H_H}\right]^+} \tag{2.18}$$

Al ir actualizando, de forma iterativa las matrices de bases y activaciones $W_P, H_P, W_H\ y\ H_H$, hasta un número máximo de iteraciones, este sistema es capaz de distinguir las bases que pertenecen a sonidos percusivos de las bases que pertenecen a sonidos armónicos.

2.5. MODELO FILTRO-FUENTE PARA VOZ CANTADA

En [Cabañas, 16], se propone un modelo filtro-fuente para obtener y representar la señal de interés que, en este caso, se trata de la señal de voz cantada y, además, discriminarla del resto de la mezcla original. Este modelo define que cada vector espectral del habla s_t, se descompone en una parte de excitación s_t^{Fo} multiplicada por una parte de filtro s_t^{Φ}, las cuales están, respectivamente, compuestas por una combinación lineal de P bases de excitación w_P^{Fo} y E bases de filtro w_e^{Φ}, quedando formulado de la siguiente manera:

$$s_t = s_t^{\Phi} \bullet s_t^{Fo} = \left(\sum_{e=1}^{E} h_{et}^{\Phi}\, w_e^{\Phi}\right) \bullet \left(\sum_{p=1}^{P} h_{pt}^{Fo}\, w_p^{Fo}\right) \tag{2.19}$$

siendo h_{et}^{Φ} y h_{pt}^{Fo} ganancias no negativas y $\bullet$ el producto de Hadamard.

Las bases de excitación w_p^{Fo}, representan una colección de sonidos discretos, a partir de los cuales, una señal puede ser construida. Además, estas bases de excitación pueden ser moduladas por una combinación de bases w_p^{Fo}. Este modelo, está diseñado para representar vocales, por lo que es conveniente que cada vector w_p^{Fo}, represente la señal glotal correspondiente a una frecuencia fundamental. Las bases w_p^{Fo} son generadas a través de un modelo fuente glotal (KLGLOTT88), dando lugar a un diccionario fijo donde se relacionan las excitaciones con el *pitch*. Si se dispone de un número suficiente de excitaciones P, se puede conseguir una cuadrícula completa de valores de *pitch*, de manera que se cubra todo el rango de tonos del hablante con una suficiente resolución. Por otro lado, las bases correspondientes a los filtros w_e^{Φ}, deben de ser capaces de representar correctamente la suave envolvente de la señal. Todas estas bases son generadas a partir de una serie de funciones suaves, dando lugar a un diccionario fijo de componentes suaves, las cuales se pueden combinar para representar cualquier envolvente suave.

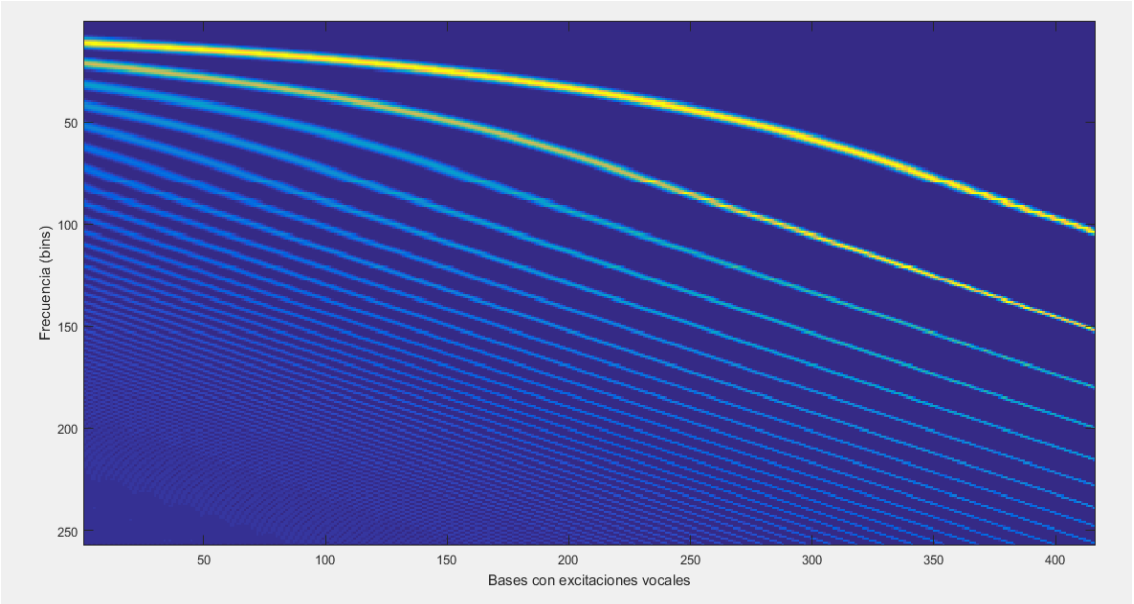


Figura 7: Excitaciones del modelo glotal

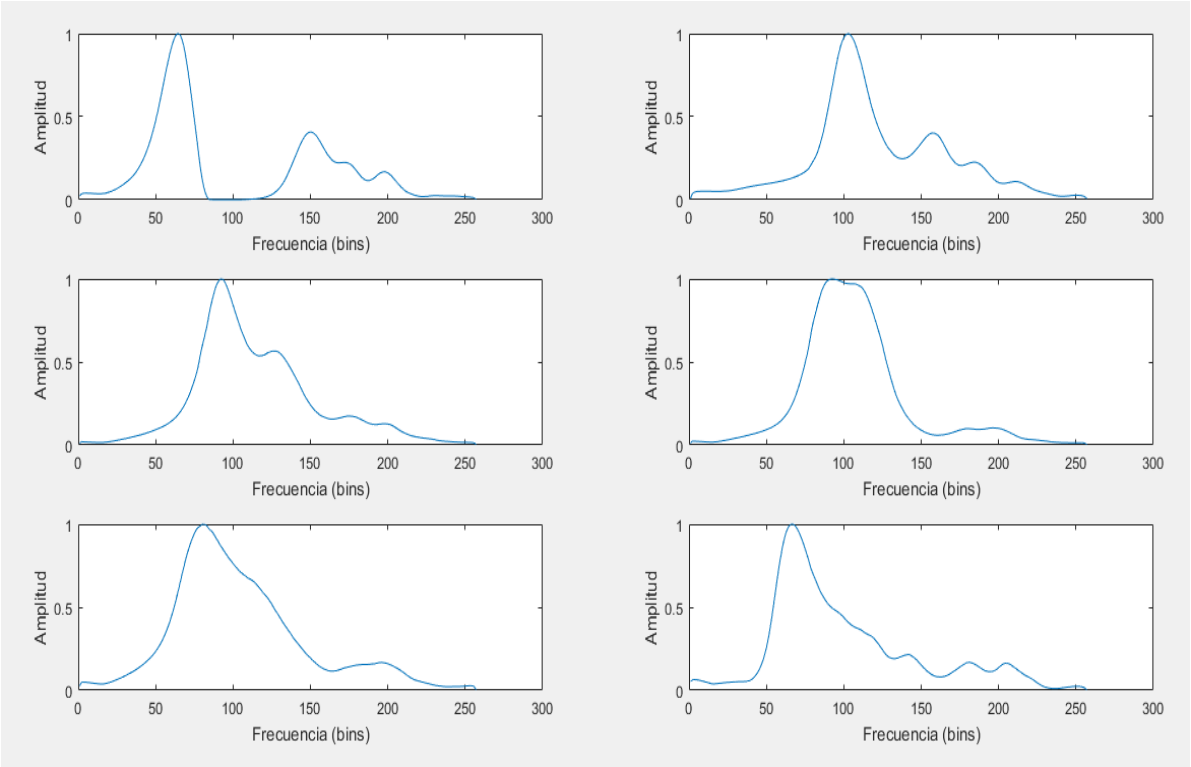


Figura 8: Filtros de fonema

Las P bases de excitaciones serán agrupadas en una matriz $W^{Fo} = [w_1^{Fo}, \dots, w_P^{Fo}]$, y E bases de filtro serán agrupadas en otra matriz $W^{\Phi} = [w_1^{\Phi}, \dots, w_E^{\Phi}]$. Por lo tanto, el modelo se formulará de la siguiente manera:

$$S = (W^{\Phi} H^{\Phi}) \bullet (W^{Fo} H^{Fo}) \tag{2.20}$$

Las matrices de ganancia $[H^{\Phi}]_{et} = h_{et}^{\Phi}$ y $[H^{Fo}]_{pt} = h_{pt}^{Fo}$, proporcionan la descomposición de la fuente de voz en los dos diccionarios anteriormente mencionados: W^{Fo} y W^{Φ}. Las amplitudes contenidas en las matrices de ganancia son estimadas a partir de la señal de entrada, mientras que los diccionarios permanecerán fijos y no variarán.

Este modelo presenta una serie de ventajas:

- Describe un modelo característico del habla y discrimina significativamente los sonidos ruidosos típicos, los cuales no siguen esta estructura.
- Mejora una descripción más estructurada del habla, ya que la información del tono y el timbre se encuentra individualmente representada, permitiendo el uso de un número razonable de bases.

En muchos casos, este modelo es extendido al habla sin voz (es decir, a la pronunciación sin la vibración de las cuerdas vocales), pero este hecho, hace al modelo más sensible a la hora de capturar interferencias.

El modelo propuesto de señal puede ser visto como una extensión del marco genérico descrito en [Durrieu, 10], con algunas modificaciones apropiadas para el problema de la separación entre la voz y el ruido. El espectrograma de entrada representado por la matriz X y de dimensiones FxT, es modelado como una suma instantánea de 4 partes, de la siguiente manera:

$$X \approx \hat{X} = (W^{\Phi} Z^{\Phi} H^{\Phi}) \bullet (W^{Fo} Z^{Fo} H^{Fo}) + W^{N} Z^{N} H^{N} + W^{I} Z^{I} H^{I} + W^{H} Z^{H} H^{H} \quad (2.21)$$

El primer término representa la voz por medio del modelo filtro-fuente descrito anteriormente. Los siguientes tres términos representan el ruido de fondo como una suma de tres tipos diferentes de ruido, expresados cada uno por una matriz diferente factorizada en un conjunto de funciones base y activaciones. El modelo de señal será definido como $\hat{X}$.

El algoritmo de estimación de parámetros se basa en la técnica de gradiente multiplicativo. Las matrices del modelo cuyo contenido se desconoce, son, en primer lugar, inicializadas con números aleatorios positivos y, a continuación, estos valores se van actualizando en cada iteración con reglas de actualización multiplicativas.

En cada iteración, una vez que la matriz ha sido actualizada con su correspondiente regla, se recalcula el modelo $\hat{X}$ antes de seguir actualizando las siguientes matrices, repitiendo el proceso descrito hasta que se alcancen un cierto número de iteraciones máximas.

Las ganancias H^{Fo} y H^{Φ} describen la descomposición de la parte del habla que solo queda restringida por el error de reconstrucción. Como consecuencia, el problema de estimación de H^{Fo} y H^{Φ} consiste en la reducción de la β-divergencia entre X y el modelo $\hat{X}$, lo cual se consigue, mediante las siguientes reglas de actualización:

$$H^{F_0} \leftarrow H^{F_0} \bullet \frac{(W^{Fo} Z^{Fo})^T ((W^{\Phi} Z^{\Phi} H^{\Phi}) \bullet \hat{X}^{\beta-2} \bullet X)}{(W^{Fo} Z^{Fo})^T ((W^{\Phi} Z^{\Phi} H^{\Phi}) \bullet \hat{X}^{\beta-1})} \quad (2.22)$$

$$H^{\Phi} \leftarrow H^{\Phi} \bullet \frac{(W^{\Phi} Z^{\Phi})^T ((W^{Fo} Z^{Fo} H^{Fo}) \bullet \hat{X}^{\beta-2} \bullet X)}{(W^{\Phi} Z^{\Phi})^T ((W^{Fo} Z^{Fo} H^{Fo}) \bullet \hat{X}^{\beta-1})} \quad (2.23)$$

siendo $\bullet$ el producto de Hadamard, $(\cdot)^T$ es el operador de transposición y todas las fracciones y exponenciales se aplican elemento a elemento.

2.6. FASE DE UNMIXING (Filtro de Wiener)

Es la última fase del proceso de separación de fuentes y en ella se parte de las matrices que se han aprendido de la síntesis. En esta parte, se utilizará el filtrado de Wiener [Rodriguez, 14] que consiste en un enmascaramiento tiempo-frecuencia y un filtrado de la mezcla según los patrones que se han aprendido inicialmente dando lugar a una separación de fuentes, objeto de este apartado.

A continuación, se describirá y detallará las directrices que sigue este filtrado para conseguir dicho resultado:

Para comenzar, se identificará la densidad espectral de potencia de la fuente *j* en el bin de la Transformada de Fourier (t, f) con el término $|S_j(t, f)|^2$, siendo j=1, …, J, por lo que, cada una de las fuentes separadas idealmente S_j podrá ser estimada a partir de

$x(t)$ haciendo uso del filtrado Wiener generalizado en tiempo-frecuencia sobre la transformada en el dominio de Short-Time Fourier Transform (STFT).

El filtro de Wiener $\alpha_j(t,f)$ de la fuente j' se corresponde con la contribución de la energía relativa aportada por cada fuente a la energía total de la señal mezclada $x(t)$ y se define como:

$$\alpha_j(t,f) = \frac{|S_j{}'(t,f)|^2}{\sum j|S_j(t,f)|^2} \tag{2.24}$$

En resumen, la suma de cada uno de los espectrogramas de potencia de las fuentes estimadas $|S_j{}'(t,f)|^2$ da lugar, de forma aproximada, al espectrograma de potencia de la señal mezclada $|X(t,f)|^2$, por lo que el espectrograma de potencia se puede estimar como:

$$|\hat{S}_{j'}(t,f)|^2 = \alpha_{j'}(t,f) \bullet |X(t,f)|^2 \tag{2.25}$$

Por último, calculando la inversa de la STFT del espectrograma estimado $|\hat{S}_{j'}(t,f)|^2$ podremos obtener la fuente de la señal estimada $S_j(t)$.

2.7. MEDIDAS DE EVALUACIÓN

En cualquier campo de investigación y desarrollo, se necesitan obtener una serie de medidas para poder ser evaluadas y obtener los resultados propios de la calidad relativa al método que se está estudiando. En el campo que se está estudiando en este proyecto, la separación de fuentes, se han utilizado numerosos métodos de evaluación, aunque solo algunos han sido reconocidos por la comunidad científica. En este capítulo, se describirán los métodos usados para evaluar el sistema de separación de fuentes de este proyecto.

Como se expone en [Rodriguez, 14], Lambert y Schobben y Torkkola, fueron pioneros en proporcionar un método de evaluación genérico para el proceso de separación de fuentes BSS (Blind Source Separation) que permitiera comparar resultados de diversos sistemas de mejora y separación de voz. Posteriormente, se han ido desarrollando otros métodos específicos para cada proyecto, cuyas evaluaciones se basaban en medidas indirectas de los resultados o test subjetivos, aunque usando estas medidas, no se podía obtener una comparativa real de los resultados que se habían obtenido para diferentes métodos de separación.

En primer lugar, Schobben y Torkkola, proponen un conjunto de datos junto con un método de evaluación de la separación de fuentes cuyo fin es, que se empleen en la comparación de algoritmos. Se presentan 2 casos: el primero, consiste en la separación controlada de un conjunto de señales sintéticas, lo cual permite conocer los límites del algoritmo al poder elegir las señales que se van a evaluar; mientras que el segundo, utiliza un conjunto de señales reales con sonidos aislados, los cuales pueden conseguirse en la red ya que están disponibles para toda la comunidad científica. Para definir la dificultad del problema, estos autores definen unos parámetros de manera que se conozca el alcance de la dificultad del problema, objeto de estudio. Como nota, hay que remarcar, qué durante la grabación de las fuentes individuales, se produce a su vez la mezcla de las

demás fuentes. A continuación, se definirán diferentes desarrollos donde $v_{o,s_j}(t)$ representa la mezcla $v_o(t)$ en la grabación de la fuente sonora $s_j(t)$ y, $\hat{s}_j(t)$ representa la fuente separada estimada. La primera de ellas se corresponde con la medida de distorsión, que indica cómo de distorsionada está la fuente con respecto a la fuente real:

$$D_j^{distortion} = 10\log_{10}\left(\frac{\sum_t \left| v_{j,x_j}(t) - \frac{v_{j,x_j}(t)}{\hat{s}_j(t)} \right|^2}{\sum_t \left| v_{j,x_j}(t) \right|^2}\right) \tag{2.26}$$

siendo j el índice que indica la fuente j-sima.

La segunda de las medidas, indica la cantidad de señal que se ha conseguido separar, formulándose así:

$$D_j^{separation} = 10\log_{10}\left(\frac{\sum_t \left| \hat{s}_{j,x_j}(t) \right|^2}{\sum_t \left| \sum_{i \neq q} \hat{s}_{q,x_i}(t) \right|^2}\right) \tag{2.27}$$

siendo $\hat{x}_{q,x_i}$, la q-sima salida del sistema de mezcla en cascada cuando sólo está activa x_i.

Por otra parte, *Vincent et al*., propone dos grupos de medidas de evaluación para BSS. Uno de ellos de denomina *Audio Quality Oriented (AQO)*, y su objetivo consiste en obtener una buena relación señal/ruido (SNR) y reducir, en la medida de lo posible, los artefactos que se encuentren en las señales separadas, mientras que el otro grupo se denomina *Significance Oriented (SO)*, y su objetivo trata de centrarse en la obtención de las características de la escena de audio a través de la estimación de las fuentes y/o los parámetros del proceso de mezcla.

Por ello, este autor, propone dividir el término que resulta de la diferencia entre la señal estimada por el sistema $\hat{s}_j(t)$ y la señal real $s_j(t)$ en 3 términos que se corresponderán con 3 tipos de error:

$$\hat{s}_j(t) - s_j(t) = e_j^{target}(t) + e_j^{interf}(t) + e_j^{artif}(t) \tag{2.28}$$

siendo $e_j^{target}(t)$ el error de objetivo, $e_j^{interf}(t)$ el error por interferencia del resto de fuentes sobre la fuente y $e_j^{artif}(t)$ el error de artefactos generados por el algoritmo de separación.

La distorsión total que hay entre la fuente estimada y la real, se define como:

$$D_j^{total} = \frac{\sum_t \left| \hat{s}_j(t) - s_j(t) \right|^2}{\sum_t \left| s_j(t) \right|^2} \tag{2.29}$$

Y las distorsiones relativas a cada uno de los términos de error, se definen como:

$$D_j^{target} = \frac{\sum_t \left| e_j^{target}(t) \right|^2}{\sum_t \left| s_j(t) \right|^2} \tag{2.30}$$

$$D_j^{interf} = \frac{\sum_t \left| e_j^{interf}(t) \right|^2}{\sum_t \left| s_j(t) + e_j^{target}(t) \right|^2} \tag{2.31}$$

$$D_j^{artif} = \frac{\sum_t \left| e_j^{artif}(t) \right|^2}{\sum_t \left| s_j(t) + e_j^{target}(t) + e_j^{interf}(t) \right|^2} \tag{2.32}$$

Una vez obtenidas las expresiones de estas distorsiones, se pueden definir los siguientes términos: *Signal to Distortion Ratio (SDR), source Image to Spatial Radio (ISR), Signal to Interference Ratio (SIR)* y *Signal to Artifact Radio (SAR).*

$$SDR_j = 10 \log_{10} \left(\frac{1}{D_j^{total}} \right) \tag{2.33}$$

$$ISR_j = 10 \log_{10} \left(\frac{1}{D_j^{target}} \right) \tag{2.34}$$

$$SIR_j = 10 \log_{10} \left(\frac{1}{D_j^{interf}} \right) \tag{2.35}$$

$$SAR_j = 10 \log_{10} \left(\frac{1}{D_j^{artif}} \right) \tag{2.36}$$

Estas medidas presentan un inconveniente, ya que no representan la calidad perceptual de la separación de fuentes, y no tienen en cuenta fenómenos como el enmascaramiento espectral o la intensidad de percepción del sonido por parte del oído humano. Las nuevas medidas que se han desarrollado intentan solventar este problema, realizando un protocolo de pruebas subjetivas para la evaluación de la calidad de percepción en separación de fuentes de audio. Estas medidas tratan de evaluar los términos de error en un conjunto de bandas, las cuales están relacionadas con las particularidades de la percepción del sonido. Estas, se definen de la siguiente manera:

- *Overall Perceptual Score (OPS)*
- *Target-related Perceptual Score (TPS)*
- *Interference-related Perceptual Score (IPS)*
- *Artifacts-related Perceptual Score (APS)*

En su conjunto componen el paquete *Perceptual Evaluation of Audio Source Separation (PEASS).*

Las medidas perceptuales se implementan en dos fases: la primera, emplea el modelo perceptual PEMO-Q, con el que se obtiene una medida de similaridad perceptual,

Perceptual Similarity Measure (PSM); mientras que la segunda combina las cuatro medidas con un mapeado no lineal, adaptándose a una escala de medidas subjetivas.

2.8. BASES DE DATOS MUSICALES PARA SEPARACIÓN DE FUENTES

A la hora de evaluar un sistema de separación de fuentes, no existe una única base de datos estándar que se aplique a todos los casos, sino que se utilizan un conjunto de bases de datos respaldadas por la comunidad científica y caracterizadas por su rigurosidad.

Estas bases de datos musicales pueden ser clasificadas en dos grandes grupos:

- **Bases de datos de sonidos sintéticos**

En estas bases de datos, las notas son generadas por un sintetizador de ficheros simbólicos MIDI (sintetizadores capaces de generar sonidos muy similares a los de un instrumento real).

- **Bases de datos de sonidos reales**

Estas bases de datos consisten en señales grabadas por músicos profesionales, que incorporan muchos aspectos como: la interpretación del músico, las características acústicas de la sala e incluso las características del propio instrumento.

Se suelen utilizar, sobre todo, las bases de datos de sonidos reales debido a que dichos sonidos son más cercanos al escenario real y a que, en separación de fuentes, los sistemas trabajan con modelos de instrumento, los cuales se ajustarán más a los instrumentos reales.

A su vez, las bases de datos de sonidos reales se pueden dividir en dos grupos más:

- **Bases de datos de sonidos individuales**

Consisten en un conjunto de notas interpretadas de manera aislada por cada instrumento, músico o estilo. Dichas notas se pueden encontrar en orden aleatorio o siguiendo algún criterio como, puede ser, el número de instrumentos o el número máximo de notas concurrentes, de forma que puedan obtenerse diferentes mezclas polifónicas. La finalidad de estas bases de datos es realizar entrenamiento de modelos de instrumentos o construir señales con unas características específicas.

Algunos ejemplos de este tipo de bases de datos son: *Musical Instrument Sound Data-base de Real World Computing (RWC), University of Iowa Musical Instrument Samples* y *McGill University's Master Samples.*

- **Bases de datos compuestas de fragmentos de composiciones musicales**

Son bases de datos muy usadas para la evaluación de sistemas de separación de fuentes en distintas escenas con distintas interpretaciones (distintos instrumentos o distintas características de la sala). Además, estas bases permiten que se pueda estudiar y comparar resultados que se han obtenido con otros métodos, por el hecho solo de utilizar la misma base de datos y el mismo conjunto de ficheros.

Algunos ejemplos de este tipo de bases de datos son: *Classical Music Database and the Jazz Music Database de Real World Computing (RWC)* y *The Bach Chorals Dataset*.

A continuación, se expondrán las características y especificaciones de la base de datos utilizada en el proceso de separación de fuentes que se ha llevado a cabo en este proyecto.

2.8.1. RWC Musical Instrument Sound Database

Consiste en una base de datos que, como se ha descrito antes, forma parte del grupo de bases de datos de sonidos reales. Está formada por 50 instrumentos musicales y se caracteriza por ofrecer un amplio abanico de sonidos:

- **Variaciones**: Cada variación se caracteriza por ser interpretada con un instrumento de diferente fabricante, intérprete o músico.
- **Técnica interpretativa**: No se produce en general, pero si en algunos instrumentos. Se incluyen grabaciones con diferentes técnicas interpretativas como pueden ser, vibrato, staccato, etc…
- **Pitch**: El músico toca todo el rango de notas que es capaz de generar con el instrumento, manteniendo intervalos de un semitono. Esto se realiza de manera individual para cada estilo interpretativo.
- **Matiz dinámico:** Se realizan grabaciones con cada nivel dinámico (*forte, mezzo y piano),* intentando abarcar como se ha mencionado antes, todo el rango de notas que el instrumento es capaz de generar.

Todos los sonidos que forman parte de la base de datos, están grabados con 16 bits a 44100 Hz en 3544 ficheros de audio monoaural, lo que se traduce, en términos de almacenamiento, en un total de 29,1 Gbytes. Cada fichero contiene en su interior una secuencia de sonidos individuales, los cuales se encuentran ordenados de forma ascendente según el pitch, obteniendo así todo el rango posible de notas que puede generar el instrumento.

2.8.2. OBTENCIÓN DE SEÑALES MUSICALES UTILIZADAS EN LA DEMO

Como paso previo, a la creación del sistema de separación de fuentes, se han escogido las señales de audio que se utilizarán como señales de entrada, para dicho sistema.

Para la elección de las señales de audio de entrada al sistema se han seguido algunos criterios como son:

- La señal debe contener sonidos armónicos, en la medida de lo posible, evitando siempre los sonidos sintéticos.
- Deben de detectarse fácilmente los instrumentos que intervienen en la mezcla, para saber que modelos de instrumentos son necesarios utilizar.
- Deben de estar muy bien definidas e identificadas las partes de la canción donde existe o no, voz cantada.

Por ello, atendiendo a los anteriores criterios, se utilizará como señal de entrada al sistema de separación de fuentes, la canción "*Fly me to the moon*" interpretada por Frank Sinatra. Esta canción no fue originalmente compuesta por este artista, sino por *Kaye Ballard*, pero su versión, en 1964, fue la que acabó dando la fama y la popularidad a este tema convirtiéndose en la versión que mucha gente identifica de esta.

Esta versión se caracteriza por ser interpretada por muchos instrumentos armónicos como son trompetas, saxofones o incluso violines, además, presenta unas partes muy bien definidas, donde se intercalan partes musicales con partes de voz solista, cada una de ellas con bastante separación entre sí.

Las señales de audio de "*Fly to the moon*" de Frank Sinatra, se han obtenido a través de la plataforma de *Youtube*, siendo descargadas y convertidas a formato WAV para poder ser tratadas por el algoritmo de separación de fuentes. Se han intentado adquirir de alguna base de datos de mayor calidad que tratara y almacenara canciones de Frank Sinatra, pero tras realizar una exhaustiva búsqueda no se ha encontrado los resultados oportunos.

2.9. MODELO DE SEPARACIÓN VOZ/ARMÓNICO/PERCUSIVA PROPUESTO

En este apartado, se procederá a explicar y describir el método llevado a cabo para la separación voz/armónico/percusiva de la señal de entrada al sistema. Para comenzar, se presenta un diagrama de flujo para entender más fácilmente el procedimiento seguido:

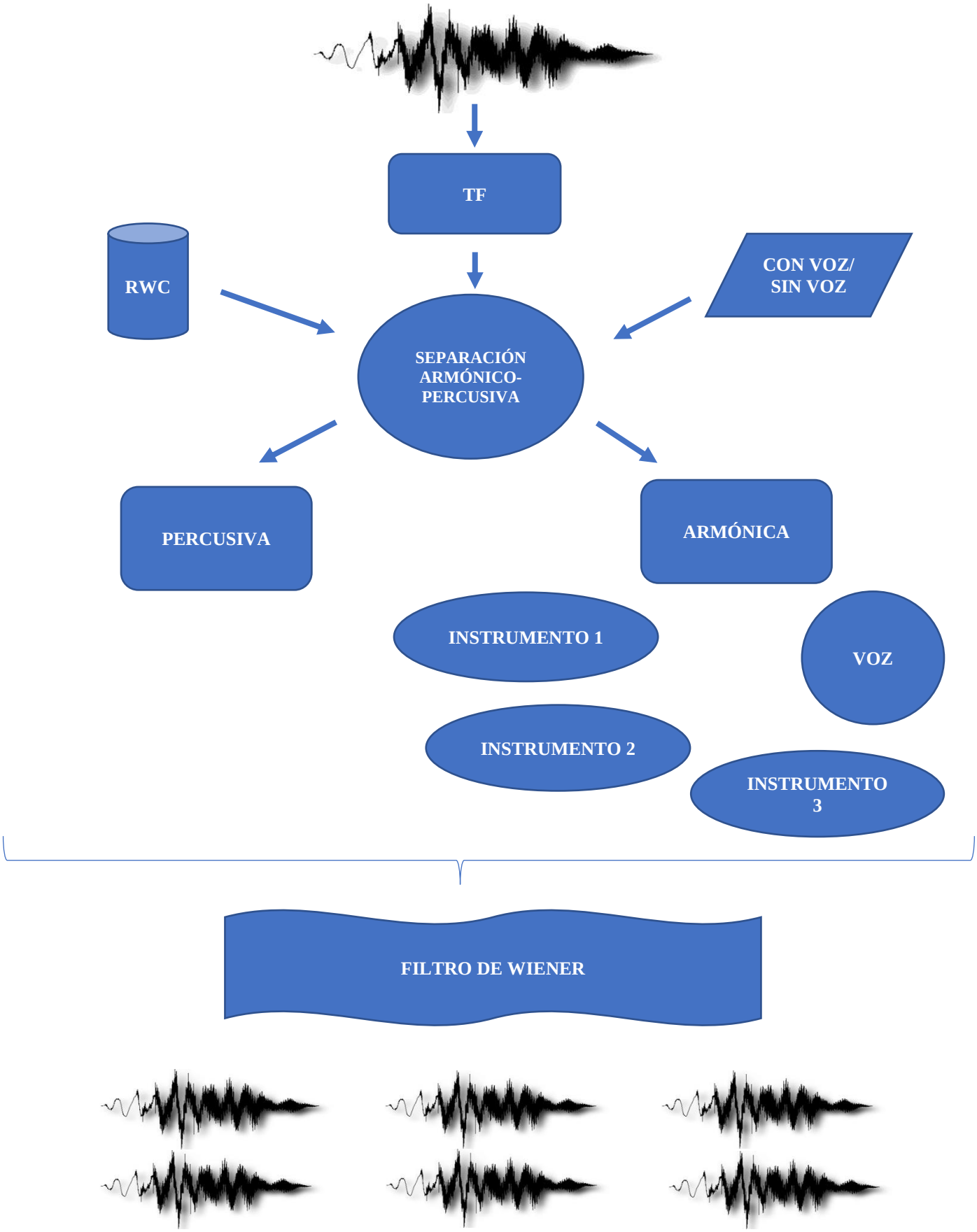


Figura 9: Diagrama de flujo del proceso de separación de fuentes

En primer lugar, este modelo pide al usuario que seleccione la señal de entrada que será objeto de análisis para este sistema y que, debe estar siempre, en formato WAV. A continuación, el sistema realiza un diezmado de la señal de entrada pasando de una frecuencia de muestreo de 44100 Hz (frecuencia a la que el sistema lee la señal) a una de 16000 Hz, para que la señal pueda ser tratada por el sistema. Además, se convierte la señal estéreo a una señal monoaural.

En segundo lugar, se obtienen los parámetros de análisis (longitud de las tramas, salto de trama, tipo de ventana, etc…) a partir de la frecuencia de muestreo que se introduzca en la función como parámetro de entrada.

En tercer lugar, se calcula el espectrograma de la señal mediante un "modelo rápido suma" el cual, discretiza el espectrograma de la señal de entrada a escala MIDI sumando los bins de cada intervalo MIDI según el número de intervalos por semitono que se escoja (en nuestro caso 4). A continuación, se normaliza el espectrograma con un valor de β=1, ideal para la separación de voz.

En cuarto lugar, se han obtenido y entrenado las bases armónicas de los distintos instrumentos que intervienen en la escena. En este caso, se han obtenido las bases armónicas del bajo, trompeta y saxofón. Para cada uno de ellos, se ha procedido al aprendizaje de los modelos de instrumento (funciones base), cuyo proceso se detalla en el apartado 2.3 de este capítulo, obteniendo las matrices de ganancias y funciones base de cada instrumento. Una vez obtenidas estas matrices, son usadas para el entrenamiento de estas bases, consiguiendo finalmente las bases ya entrenadas de cada uno de los instrumentos.

En nuestro caso, cabe mencionar, que se ha usado como paso previo al entrenamiento de las bases, un segmentador, el cuál consiste en que a través de un fichero TXT se diferencian las partes en las que solo hay música de las que hay voz cantada, de manera que, se obtienen resultados mucho más satisfactorios y eficientes ya que en las partes donde solo hay música no hace falta entrar en el entrenamiento para la parte vocal.

Siguiendo con el procedimiento que se describía en quinto lugar, las funciones base ya entrenadas de cada instrumento, son concatenadas en una matriz "Wh_i" que representará la matriz de bases armónicas.

En sexto lugar, se procederá a realizar el entrenamiento armónico-percusivo con la misma función con la que se entrenaron las bases armónicas. En este caso, se utilizará como parámetro de entrada las bases armónicas que se han concatenado dando lugar a la matriz "Wh_i" y, además, se establecerá un número de bases percusivas que, se ha establecido como 100 para nuestro caso. Otros parámetros de entrada que se utilizan en este caso son: el espectrograma de la señal de entrada que corresponde solo a la parte instrumental "Xmi" y el resto de parámetros, como el número de filtros vocales o las bases de las excitaciones vocales permanecerán a 0, ya que en este punto solo se va a realizar la separación armónico-percusiva.

De las dos partes que se obtienen, armónica y percusiva, nos centraremos en la parte armónica, de la cual obtendremos las fuentes sonoras de cada uno de los instrumentos y la parte vocal de la canción que se está estudiando. Para la obtención de la parte vocal se procederá al entrenamiento de los filtros y excitaciones vocales, pasando como

parámetros de entrada fijos, las matrices donde se encuentran las bases y activaciones tanto armónicas como percusivas, obtenidas en el paso anterior.

Por último, se calcularán las máscaras y el espectrograma inverso de cada una de las fuentes sonoras para, a través del filtro de Wiener, obtener una señal aislada de cada fuente y, de esta manera, poder reconstruir la escena virtual con un mayor realismo.

2.10. RESULTADOS OBTENIDOS

Para la evaluación del sistema de separación de fuentes se va a utilizar la base de datos [DSD100, 19], la cual está formada por un conjunto de 100 canciones de diferentes estilos. Cada una de estas canciones lleva asociada una serie de 4 pistas aisladas (bajo, percusión, voz y otros, refiriéndose esta última al resto de la mezcla que no se incluye en las 3 pistas anteriores).

[DSD100, 19] contiene dos carpetas, una se utilizada para entrenamiento y se denomina "Dev", mientras que la otra se utiliza para testear y se denomina "Test". Cada una de estas carpetas cuenta con 50 canciones y, cada archivo de audio, contiene una mezcla que se corresponde con la suma de todas las señales, siendo todas ellas, señales estereofónicas y codificadas a 44100 Hz. Todos los datos relativos a esta base de datos se derivan de la biblioteca de descargas multipista gratuita de *'Mixing Secrets'*.

De esta fuente [DSD100, 19], también se obtiene el algoritmo que permitirá la evaluación del sistema de separación de fuentes, así como la posterior obtención de los correspondientes resultados que se exponen a continuación.

Los resultados se representarán en diferentes histogramas, que se corresponderán con cada una de los archivos de audio extraídos de la mezcla original, es decir, parte vocal, parte percusiva, bajo, resto (residuo de la parte armónica) y acompañamiento (mezcla original sin la voz). Dentro de cada uno de los histogramas se encuentran los diferentes métodos que se han usado para la evaluación de esta base de datos, representando así los resultados que se han obtenido para cada método, así como para medida de evaluación (SDR, ISR, SIR y SAR). Cada uno de estos métodos, no es desarrollado en la memoria de este proyecto, ya que no es objeto del mismo, solo se explicará el método propuesto en el presente proyecto. Para más información sobre los métodos comparados el lector puede consultar [SISEC,17]. Al final de cada histograma, se representa este método propuesto para, de esta manera, comparar los resultados obtenidos y evaluar si son mejores o peores con respecto a los obtenidos por otros métodos.

A continuación, se pueden visualizar los diferentes histogramas con los resultados para cada archivo de audio:

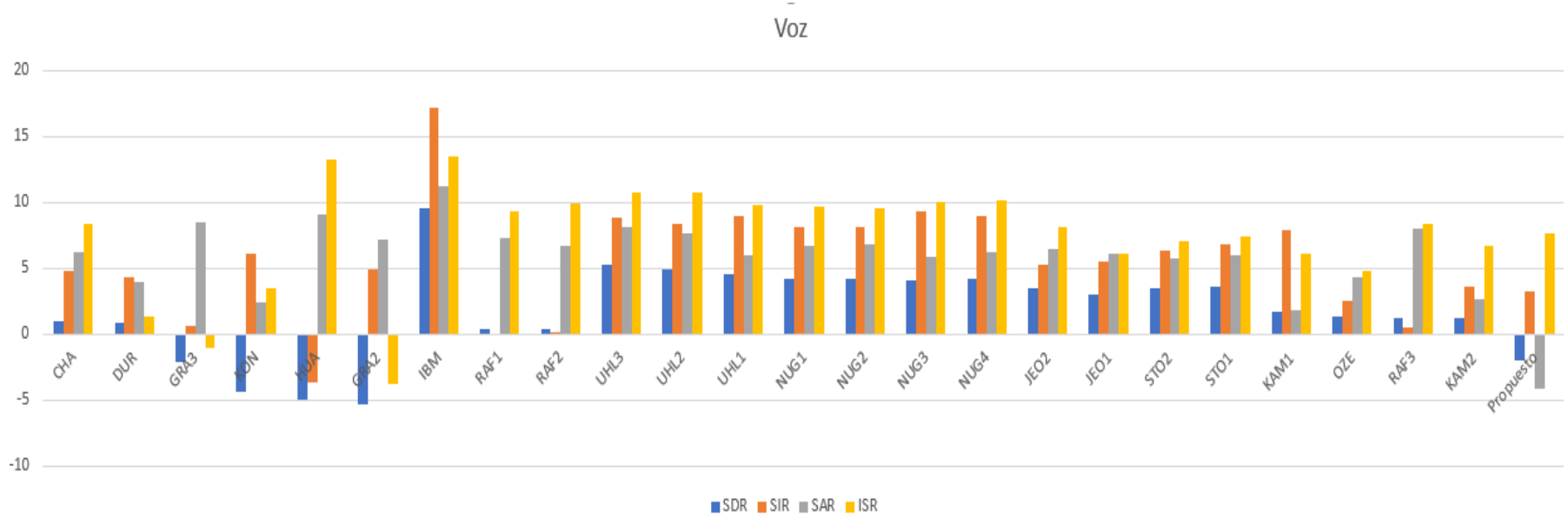


Figura 10: Resultados para la parte vocal

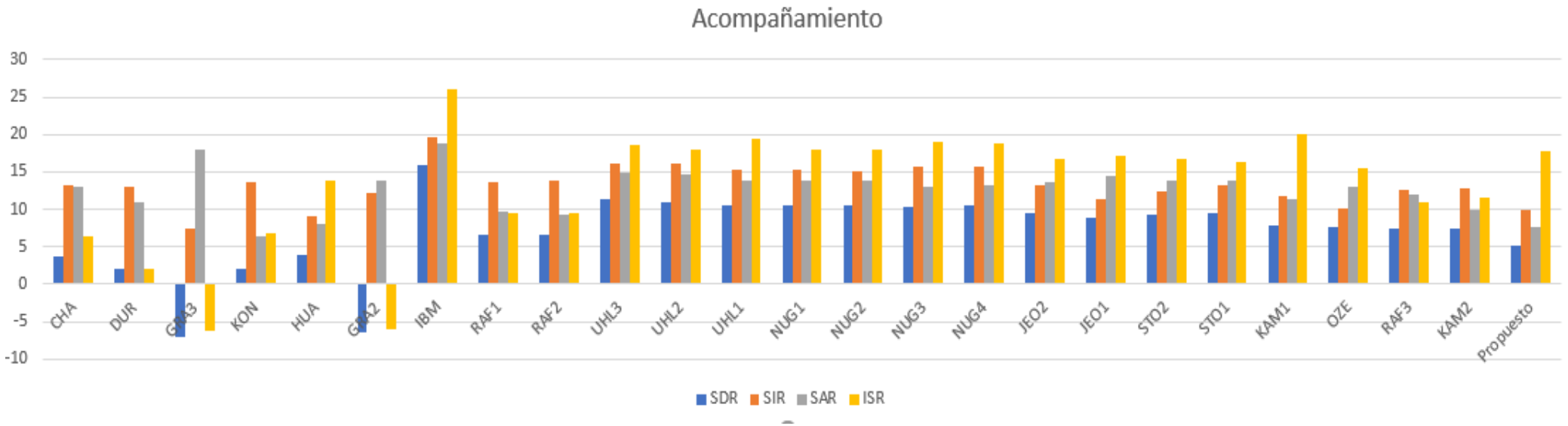


Figura 11: Resultados para la parte de acompañamiento

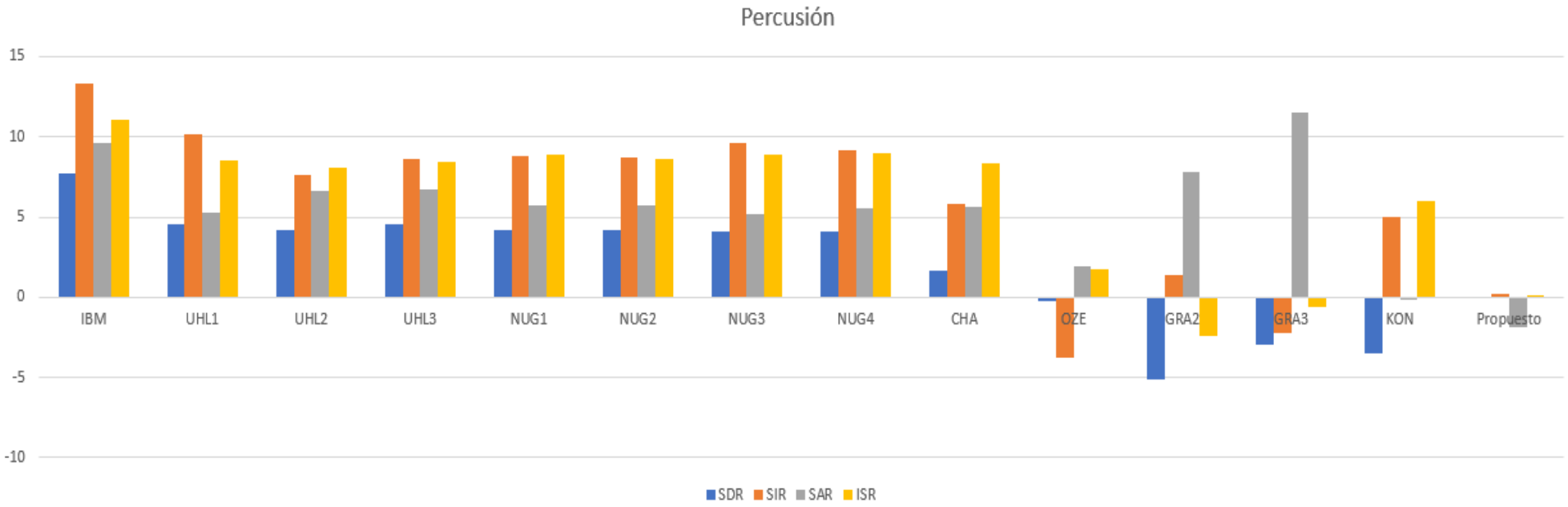


Figura 12: Resultados para la parte de percusión

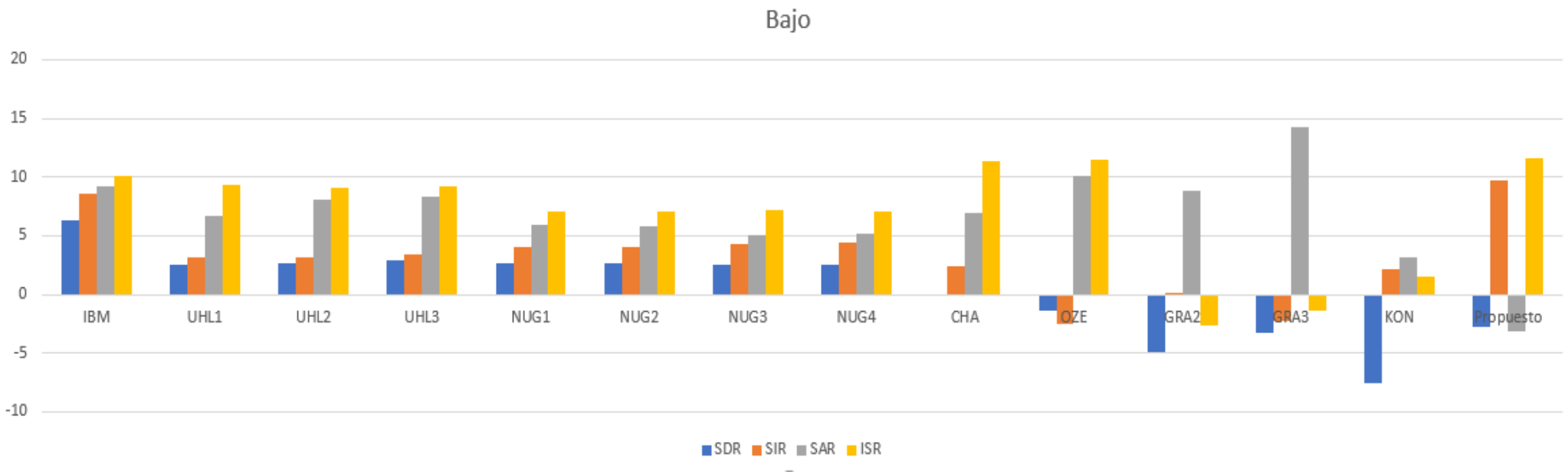


Figura 13: Resultados para la parte del bajo

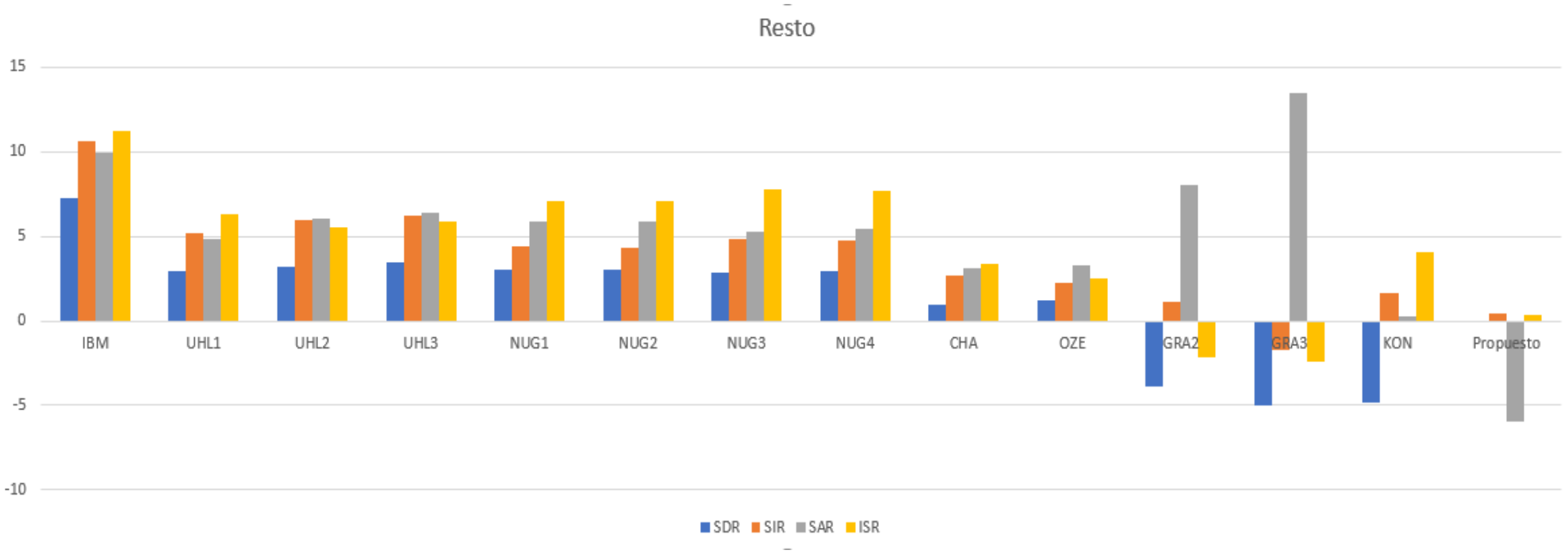


Figura 14: Resultados para la parte del residuo armónico

La base de datos analizada y evaluada en este apartado se caracteriza por ser muy novedosa y los métodos con los que se obtienen un mejor funcionamiento, están basados en técnicas de *Deep Learning*, haciendo un uso extensivo del conjunto de entrenamiento de esta base de datos. Por otro lado, el método propuesto en este caso, se caracteriza por ser más genérico para cualquier tipo de señales de mezcla, no solo para las señales de esta base de datos en particular.

Para la parte vocal, se han usado un conjunto de filtros fuente pre-aprendidos de una base de datos de voz según [Cabañas,16], los cuales parece que no se ajustan adecuadamente al caso de voz cantada en las señales analizadas, dando lugar a unos valores bajos en términos de SAR y SDR, y haciendo que la reconstrucción de la parte vocal no sea todo lo buena que debería llegar a ser.

Con respecto a la parte del acompañamiento, aunque se ha partido de patrones de notas individuales aprendidos de la base de datos de RWC, se han obtenido una serie de resultados que son bastantes competitivos respecto al resto de métodos evaluados.

Los resultados obtenidos para la parte percusiva han sido poco satisfactorios con respecto a la mayoría de métodos evaluados ya que los patrones percusivos no se ajustan adecuadamente con los de la base de datos en cuestión, aunque en la reconstrucción de la parte percusiva se consiguen mejores resultados que en la reconstrucción de la parte del bajo o la parte vocal.

Como ocurre en la parte vocal, los patrones que se han aprendido para la parte del bajo no parecen ajustarse bien con los de la base de datos analizada en esta sección. Por lo tanto, se obtienen también unos valores bajos en términos de SDR y SAR, dando lugar a una mala reconstrucción de la señal.

Por último, la parte armónica de la señal tampoco obtiene buenos resultados debido a que los patrones aprendidos tampoco se ajustan adecuadamente a los de la base de datos analizada, lo que origina una mala reconstrucción de dicha parte.

Cabe mencionar en este apartado que, como mejora del método propuesto, se ha utilizado un archivo *txt* para diferenciar las partes de la canción donde aparece la voz y donde no. De manera que, el algoritmo de separación solo entre a analizar la parte vocal en las partes donde esta aparezca y no en toda la canción. De esta manera es posible ajustar las bases aprendidas y obtener una mejora substancial de los resultados de separación.

A pesar de esto, los resultados obtenidos para el método propuesto son bastantes bajos con respecto al resto de métodos evaluados, pero ha de tenerse en cuenta que el escenario de aplicación de nuestro sistema de separación de fuentes es un *remixing* acústico, de manera que, a pesar de que el resultado de la separación individual de cada fuente no sea óptimo, una vez se reconstruya la mezcla se obtendrá un resultado global cuya calidad final estará más asociada a los valores de SIR que al resto de métricas (SDR,SAR,ISR).

Además, cabe destacar que la mayor parte de los métodos expuestos en esta sección están optimizados para esta base de datos, mientras que el método propuesto no está ajustado a los parámetros de dicha base de datos.

3. AUDIO 3D

3.1. INTRODUCCIÓN

El audio 3D es aquel sistema de sonido que pretende ubicar en un determinado punto del espacio, a través de cualquier dispositivo de reproducción de sonidos, una fuente sonora de manera que el oyente experimente la sensación de estar en un entorno real, sin llegar a estarlo.

En 1997, Blauert consiguió desarrollar esta tecnología después de realizar ciertos estudios y utilizar el hecho de que el oído humano funcionaba como un filtro de señales.

En este capítulo, se describirán las diferentes técnicas que se usan o se han usado para recrear sensaciones sonoras de localización de un objeto virtual, como pueden ser: el efecto phantom, las técnicas de audio espacial o la síntesis binaural.

En particular, el estudio se centrará sobre todo en la síntesis binaural ya que será la que se use en el desarrollo de la aplicación móvil.

3.2. EFECTO PHANTOM

Nació del estudio de Lord Rayleigh en 1907 y consiste en la localización de una fuente sonora por parte del cerebro debido a la diferencia de nivel (ILD) o tiempo (ITD) con la que llega la fuente a cada oído. Este efecto solo es válido en el plano horizontal, ya que en el plano vertical se produce de manera diferente, y solo se producirá en reproducciones estéreo con fuentes coherentes (fuentes relacionadas entre sí).

De esta manera, el cerebro, pondera los valores que recibe en cada oído, situando la fuente entre alguna de las distintas posiciones que hay entre los dos altavoces de manera que, el oyente escuche el sonido como si viniera desde una única posición o fuente. A continuación, se muestra un ejemplo:

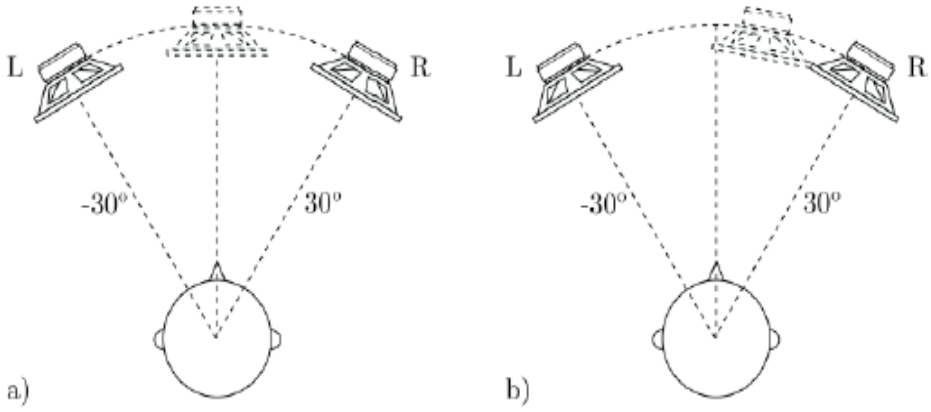


Figura 15: Efecto phantom estéreo

Se puede apreciar en la anterior imagen, que se tienen dos casos: en el primero de ellos, la fuente virtual se encuentra centrada entre los dos altavoces debido a que el sonido producido por los dos altavoces llega a ambos oídos con el mismo nivel y transcurrido, aproximadamente, el mismo tiempo; mientras que en el segundo, se encuentra la fuente virtual algo desplazada hacia la derecha debido a que el altavoz derecho, en este caso, está emitiendo el sonido o bien en menor tiempo o bien con mayor nivel respecto al altavoz que se encuentra en la parte izquierda.

Los sistemas que hacen uso de este efecto para generar la sensación descrita anteriormente, son el sistema estéreo y el sistema Surround.

El sistema estéreo consiste en dos canales de reproducción, donde la creación de fuentes virtuales solamente se produce en el espacio que separa ambos altavoces. Para que se produzca esta sensación sonora el ángulo máximo que se recomienda es de 60º.

El efecto phantom se consigue utilizando el *panning* de amplitud, que se describe mediante la ley de la tangente, mostrada a continuación:

$$\frac{\tan \Theta_p}{\tan \Theta_p} = \frac{g_L - g_R}{g_L + g_R} \tag{3.1}$$

Además de esta ecuación, es necesaria alguna otra ecuación que nos permita resolver la incógnita de qué ganancias se deben aplicar al sonido de cada altavoz para poder producir dicha fuente virtual.

Otra de las condiciones que se imponen para la fuente virtual es que su amplitud debe ser constante, por lo que su volumen se corresponderá con la distancia euclídea de las ganancias de cada uno de los altavoces para esa fuente, describiéndose de la siguiente manera:

$$\sqrt[p]{\sum_{n=1}^{N} g_n^p} = 1 \tag{3.2}$$

siendo el parámetro p el que indicará las características acústicas de la sala, donde un valor de 1 se corresponderá con una sala en la que no existe reverberación y un valor de 2 con una sala reverberante.

Hay otros métodos [Senderos, 18] para conseguir este efecto como pueden ser los formatos de grabación microfónica que se usan para esta función. Alan Blumlein, concibió en 1931, una nueva forma de grabación con la que se consiguieron grandes resultados en lo que se refiere a la espacialización de fuentes en una escena virtual. Lo hizo utilizando un par coincidente de micrófonos figura de 8 o bidireccionales, dejando un ángulo de 90 grados entre ellos, de la siguiente manera:

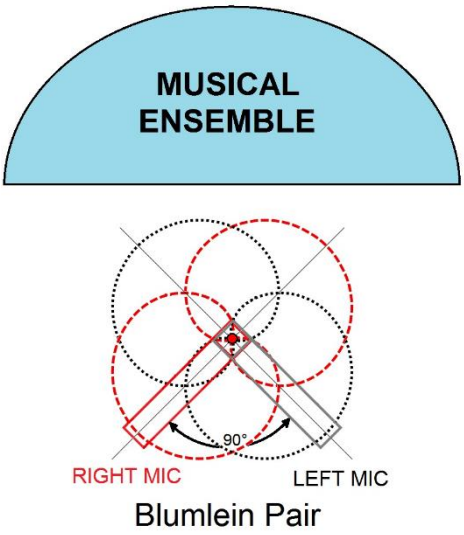


Figura 16: Técnica de grabación Blumlein

Posteriormente, con el tiempo fueron apareciendo nuevos sistemas de grabación estéreo como fueron:

- **A/B**

Consiste en el uso de dos micrófonos idénticos separados entre sí por una determinada distancia y apuntando hacia la fuente sonora. El diagrama polar de estos micrófonos suele ser omnidireccional por su respuesta en frecuencia extendida. Además, en esta técnica los elementos que se encuentran fuera del centro son difíciles de localizar.

- **X/Y**

Consiste en el uso de dos micrófonos cardioides cuyas cápsulas se disponen lo más cerca posible entre sí [Belchior, 19], con un ángulo aproximado de 90 grados, aunque dependiendo de la situación pueda variar. Las cápsulas de los micrófonos se sitúan en el mismo punto, una en lo alto de la otra, para cancelar así las fases. Esta técnica se suele usar cuando la fuente se encuentra cerca.

- **Mid-Side (M-S)**

Consiste en el uso de un micrófono cardioide apuntando hacia el frente de la fuente y otro en figura en ocho apuntando hacia los laterales de la sala, estando uno del otro lo más cercano posible. Esta técnica es efectiva si el sonido proviene del centro de la fuente, pero no lo es tanto, cuando se desea captar un sonido de grandes grupos o con un ancho mayor. Además, no presenta problemas de fase y tiene gran compatibilidad mono.

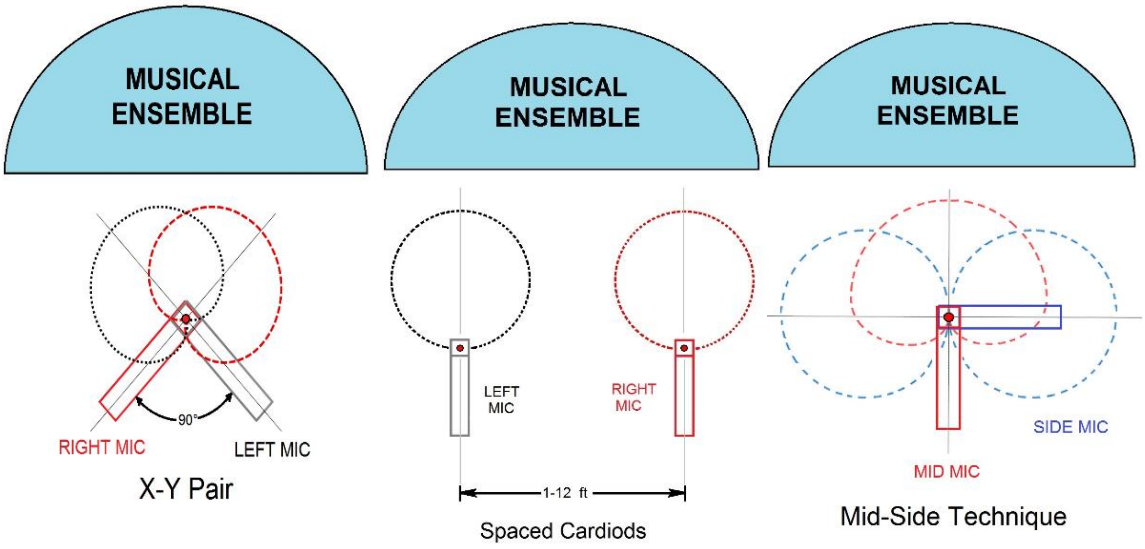


Figura 17: Otros métodos de grabación estereofónica (X/Y, A/B y Mid-Side)

Con los avances que dieron lugar a estas tecnologías [Owsinski, 17], surgió también otra novedad: el incremento de canales de reproducción, de manera que se generase una sensación todavía más realista al oyente. Un ejemplo de ello fueron los sistemas 5.1 o 10.2, con los que se consiguió emular la presencia de fuentes sonoras alrededor del oyente en el plano donde se encuentran los altavoces.

Siguiendo esta línea, se pueden encontrar sistemas que han usado la triangulación entre altavoces para simular efectos en tres dimensiones.

El sistema *Vector Base Amplitude Panning* (VBAP) [Gutiérrez, 15] fue desarrollado por V. Pulkki [Pulkki, 97] y consiste en un método de reconstrucción de fuentes virtuales que emplea únicamente el *panning* en amplitud en 3 dimensiones. Su funcionamiento es sencillo y trata de rodear el escenario con una superficie ficticia que se deberá triangular colocando un altavoz en cada vértice. Para lograr la síntesis de una imagen *phantom*, primero habrá que ver en qué triangulo se sitúa la fuente para posteriormente calcular la ganancia que se debería aplicar a cada uno de los altavoces que forman el triángulo. La ganancia de cada altavoz se obtiene expresando la posición de la imagen *phantom* como una combinación lineal de las posiciones de cada uno de los tres altavoces, formando dichas posiciones una base.

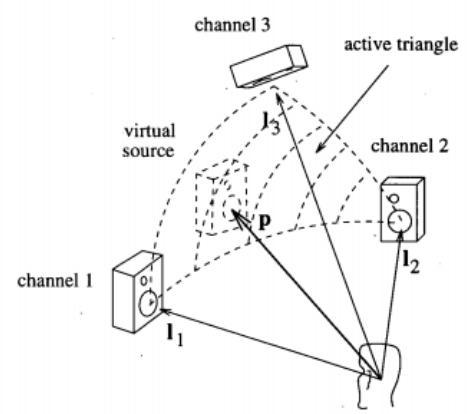


Figura 18: Método VBAP

Los sistemas que se han descrito anteriormente tienen una cierta limitación, y es que la posición del oyente debe ser fija, ya que en el momento en el que se desplaza, el efecto no es percibido con tanta claridad. A continuación, se describirán técnicas que no presentan esta limitación y donde el oyente si podrá desplazarse.

3.3. TÉCNICAS DE AUDIO ESPACIAL

3.3.1. WAVE FIELD SYNTHESIS

Esta técnica [Gutiérrez, 13] se basa, principalmente, en el concepto de "cortina acústica" el cual consiste en que si delante del escenario se colocara una cortina como si fuera un telón, en el que hubiera una infinidad de micrófonos, se podría registrar el sonido que se propaga desde el escenario al patio de butacas. Sí, además, colocáramos otra cortina con tantos altavoces como micrófonos se han colocado en la cortina anterior, se podría enviar el sonido que registra cada micrófono a su altavoz correspondiente, reconstruyendo de esta manera al otro lado de la segunda cortina la escena original.

Este concepto fue descrito por Berkhout haciendo uso de la teoría de ondas a finales de los 80, comenzando a emplearse el término *Wave Field Synthesis* a principios de los 90.

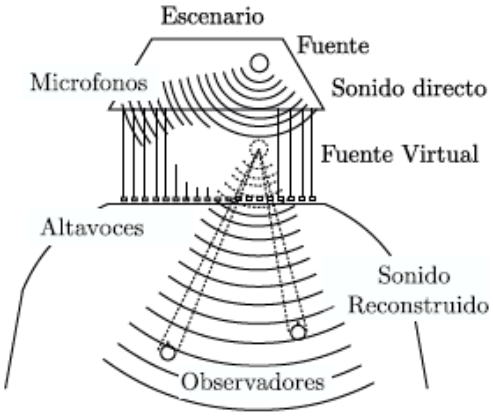


Figura 19: Cortina acústica

Esta técnica se fundamenta en el principio de Huygens, el cual establece que un frente de ondas radiado por una fuente sonora se comporta como un conjunto infinito de fuentes puntuales que se distribuyen por todo el frente de ondas, denominándose fuentes secundarias, y pudiendo ser descrita la radiación del campo sonoro a través de dichas fuentes. Para recrear las fuentes secundarias, se tendrían que colocar altavoces en las mismas posiciones donde se encuentras las fuentes puntuales, pero debido a la forma curva del frente de ondas y a que además dicho frente varía al moverse la fuente, se opta mejor por colocar los altavoces en una línea recta y compensar su movimiento mediante la aplicación de una ganancia o retardo.

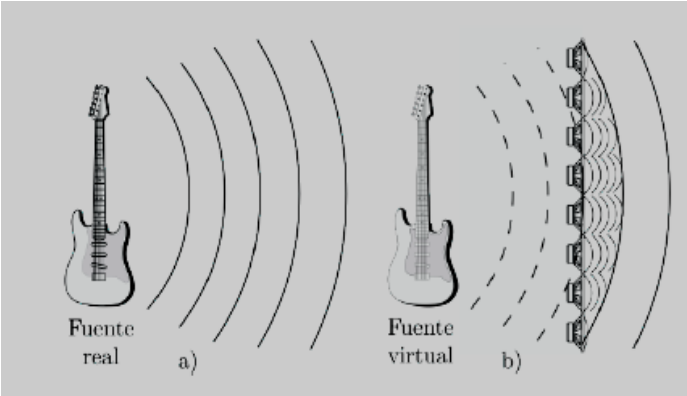


Figura 20: Aplicación del principio de Huygens
a) Campo sonoro emitido por una fuente real
b) Campo sonoro sintetizado mediante WFS

El inconveniente de esta técnica es la cantidad de altavoces que se necesitan y el coste computacional que conlleva generar la señal necesaria para cada altavoz.

3.3.2. AMBISONICS

Esta técnica [Perales, 15], basada en los principios estudiados por la Psicoacústica, trata de reproducir las características direccionales de un campo sonoro más allá de las que se pueden obtener por el uso de diferencias de amplitud o fase de una grabación estereofónica convencional. Por ello, la ventaja que presenta esta técnica es la capacidad de reproducción de un sonido inmersivo independientemente del número y la posición de los altavoces y del oyente, además Ambisonics proporciona el área de escucha más grande entre todos los sistemas de audio envolvente.

Fue ideado por un grupo de investigadores, aunque el principal fue Michael Gerzon, y consiste en la grabación por medio de 4 micrófonos, uno omnidireccional que recibe los cambios de presión (W) y tres bidireccionales que recogen el gradiente de presión de cada eje tridimensional (X,Y,Z). Con lo registrado por cada uno de estos canales, podemos obtener una aproximación del campo sonoro envolvente mediante sus armónicos esféricos.

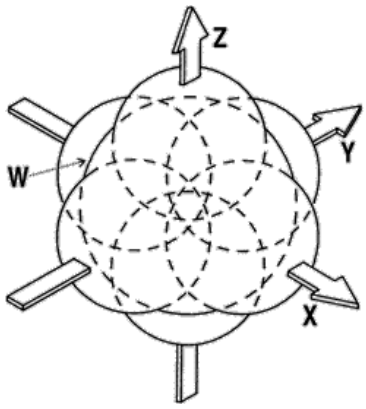


Figura 21: Formato básico de Ambisonics

Las ecuaciones de cada uno de estos patrones se describen, a continuación:

$$W(t) = s(t)/\sqrt{2}$$

$$X(t) = s(t)\cos\Theta\ cos\ \Phi$$

$$Y(t) = s(t)\,\text{sen}\,\Theta\ cos\ \Phi$$

$$Z(t) = s(t)\ sen\ \Phi$$

Siendo Θ el ángulo de la fuente virtual con respecto al eje horizontal y Φ la elevación de la fuente respecto al mismo eje. Se puede observar que el micrófono omnidireccional no depende de ningún ángulo de incidencia mientras que los demás sí.

Cada micrófono guarda en una pista diferente la información recogida y, a continuación, se introduce la localización de cada altavoz y la posición del oyente, para calcular la señal que tendrá que emitir cada altavoz de manera que se adecue la imagen sonora al oyente.

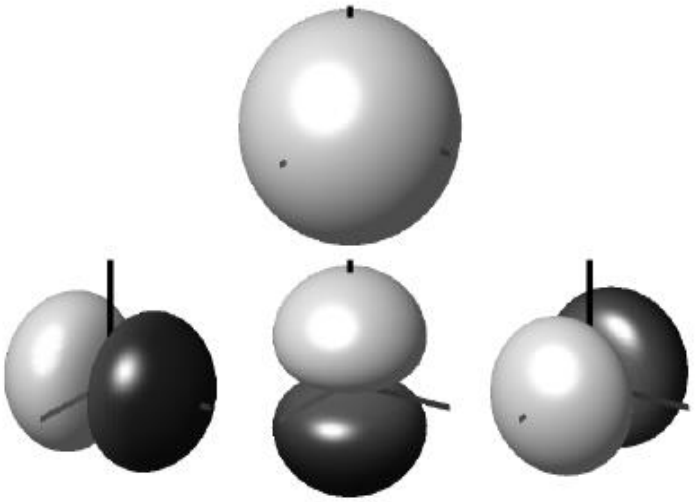


Figura 22: Ambisonics de primer orden

3.4. SÍNTESIS BINAURAL

Esta técnica [García, 17] es la utilizada en el desarrollo de la aplicación, objeto de este proyecto, y consiste en sintetizar, mediante técnicas convolucionales, los sonidos binaurales a partir del sonido monofónico de una fuente sonora puntual en el espacio. Se deben capturar las respuestas al impulso binaurales (Binaural Impulse Response, BIR) del maniquí desde diversas posiciones del espacio y a partir de cada BIR, se realiza la convolución de cada respuesta al impulso con el sonido, de manera independiente para cada oído, obteniendo el sonido que captaría el maniquí si la fuente sonora estuviera en la posición desde la que se realiza la medida del BIR.

La síntesis binaural es capaz de procesar cualquier sonido para que, mediante unos auriculares, se escuche en un punto concreto del espacio. Esta técnica se está empleando actualmente, sobre todo, en desarrollo de videojuegos de realidad virtual.

Otros de los conceptos que se rescatan para entender la síntesis binaural, son los sistemas ILD y ITD. Mediante estudios se ha reconocido que el oído humano por debajo de una determinada frecuencia (en torno a los 2 kHz) tiene una mejor respuesta a la ITD, mientras que para frecuencias mayores responde mejor a los sistemas ILD, debido entre otros factores a la distancia promedio que existe entre los oídos en la cabeza humana. Hay que tener en cuenta que los valores de ITD y ILD, solo son válidos para el plano en el que se encuentran los oídos y no se proporciona ninguna información de elevación.

Por ello, surge el sistema "Respuesta al Impulso Relativa a la Cabeza" (*Head Realted Transfer Function, HRTF*), el cuál sí que aporta toda la información relacionada con la posición de la fuente sonora y la cabeza del oyente junto con su sistema auditivo.

Este sistema utiliza dos técnicas: la primera de ellas, utiliza un maniquí (*dummy head*) para realizar una grabación emulando una cabeza humana, parte del torso y las formas genéricas del oído humano donde se ubican los correspondientes micrófonos, como se puede ver en la siguiente imagen; la segunda, consiste en elegir una posición concreta y filtrar un sonido virtual con los valores HRTF para dicha posición.

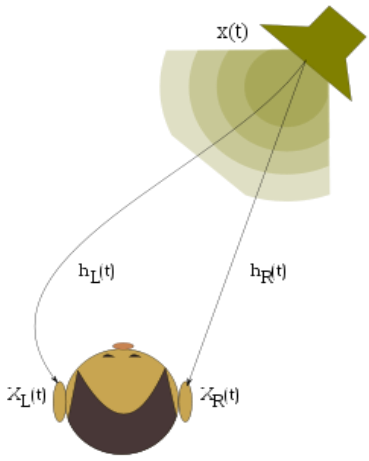


Figura 23: Principio de síntesis binaural

Figura 24: Dummy head

4. REALIDAD VIRTUAL

En este capítulo, se introducirá el concepto de realidad virtual, así como la evolución que ha tenido hasta nuestros días, centrándonos en particular en el sistema de Google Cardboard, el cuál ha sido usado para la elaboración de este proyecto. Se describirá la utilización del software "Unreal Engine" con el que se ha elaborado la interfaz gráfica y la escena virtual del presente proyecto, así como las diferentes plataformas de las que se ha hecho uso para la importación de objetos 3D y personajes de animación.

4.1. INTRODUCCIÓN

Para tomar conciencia de este capítulo, se comenzará dando una definición al término de "Realidad Virtual". Se puede definir [VR, 19] como una simulación interactiva por computador desde el punto de vista del participante, en la cual se sustituye o se aumenta la información sensorial que se recibe.

En esta definición, se puede observar tres elementos básicos que caracterizan la realidad virtual: la simulación interactiva, la interacción implícita y la inmersión sensorial.

Todo sistema de realidad virtual presenta una arquitectura característica que se compone por los siguientes elementos principales:

- Periféricos de entrada (sensores)
- Periféricos de salida (efectores)
- Computador
- Modelo geométrico 3D
- Software de tratamiento de datos de entrada
- Software de simulación física
- Software de simulación sensorial.

A lo largo de la historia, se han usado numerosos sistemas para recrear la escena tridimensional como, por ejemplo, los sistemas anáglifos, que se empezaron a usar en el cine en blanco y negro y, para los cuales, se tenía que grabar con dos cámaras para generar la imagen de cada ojo y pintar después, la película correspondiente a un ojo de azul y la otra de rojo. Cuando el espectador veía la película, utilizaba unas gafas con cada lente de un color, filtrando la imagen correspondiente a cada ojo. A continuación, se muestra un ejemplo de este sistema:

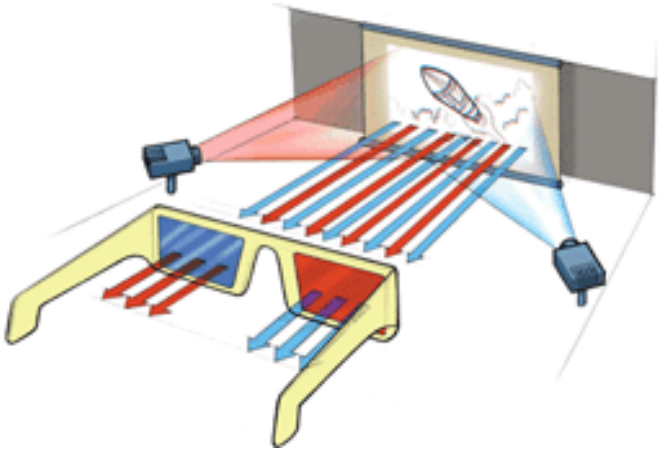


Figura 25: Sistemas anáglifos

Debido a la limitación que existía a la hora de que este sistema solo se podía utilizar para las reproducciones de películas en blanco y negro, se desarrollaron otros sistemas para la recreación de escenas en 3 dimensiones que permitieran la reproducción de escenas en color. Estos sistemas se basaron en la polarización de imágenes haciendo uso de gafas con cristales polarizados, los cuales filtran la imagen que corresponde a cada ojo. Para estos sistemas se vuelven a utilizar dos cámaras sincronizadas para la grabación, mientras que para la reproducción se polariza la imagen en sentido vertical para un ojo y en sentido horizontal para el otro.

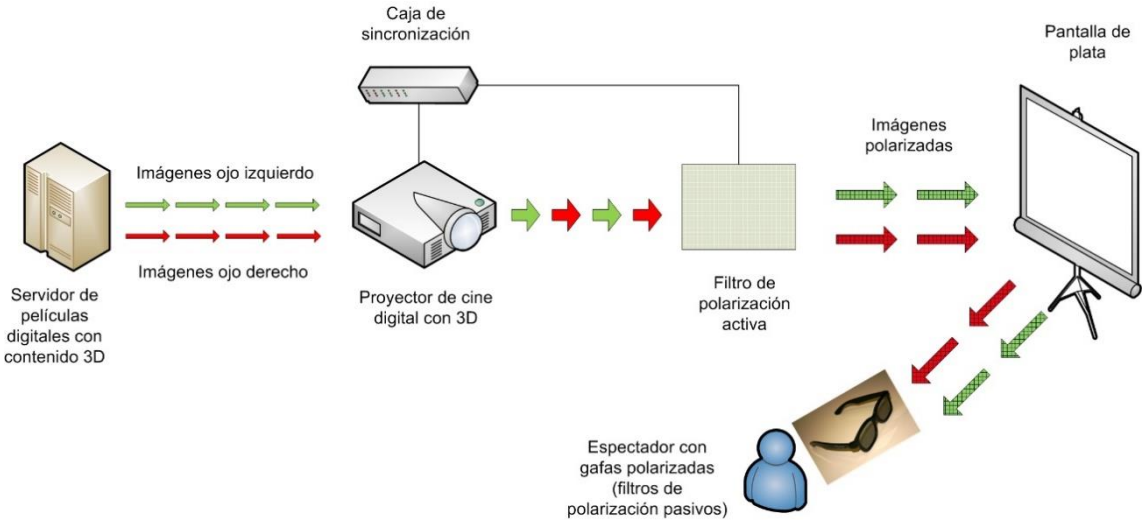


Figura 26: Sistemas con gafas polarizadas

La ventaja que introdujeron estos sistemas fue la mejora de la imagen y la limitación de distorsión, pero también se asumía un mayor rigor de sincronización debido a que si en algún momento, alguna de las dos imágenes no estaba completamente sincronizada se producirían problemas en la reproducción.

En la técnica utilizada en este proyecto para el desarrollo de la aplicación móvil se utiliza este mismo concepto: se dedica una cámara a cada ojo y, la aplicación en tiempo real, recalcula la imagen que irá viendo el usuario conforme se vaya desplazando por la escena

virtual. A continuación, se muestra el ejemplo de nuestra aplicación en el dispositivo móvil:

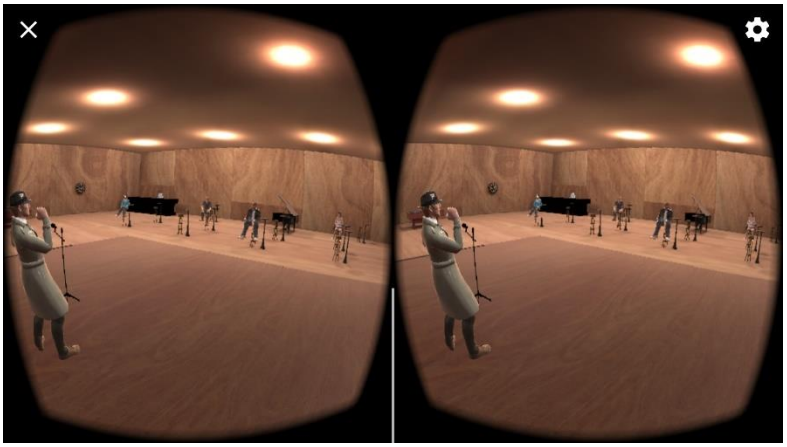


Figura 27: Pantalla de visionado para VR

En efecto, cada imagen presenta una perspectiva y por lo tanto no son totalmente idénticas, es más cada una de las vistas, está pensada para simular la distancia que existe entre nuestros dos ojos y dar la sensación de espacialidad y realismo al usuario.

4.2. UNREAL ENGINE

Ante la necesidad de realizar una escena virtual donde plasmar y ubicar las diferentes fuentes sonoras para que, posteriormente, el usuario pueda moverse e interaccionar con ellas, se realiza un estudio para elegir un desarrollador de videojuegos que permita conseguir los objetivos del presente proyecto.

El motor de videojuegos que se utilizará en este proyecto será “Unreal Engine”. Creado por Epic Games y lanzado por primera vez en 1998, desarrolla principalmente videojuegos de sigilo, lucha, videojuegos de rol multijugador masivo en línea (MMORPG) o juegos de rol (RPG). Utiliza el lenguaje de programación C++ y presenta un alto grado de portabilidad lo que permite su uso a muchos desarrolladores de videojuegos.

La versión utilizada para la realización de este proyecto es la 4.17, aunque su última versión es la 4.21, lanzada en 2018.

4.2.1. APLICACIONES DE UNREAL ENGINE

Se puede pensar que Unreal Engine es solo un motor de videojuegos que se utiliza simplemente en el mundo del ocio y el entretenimiento, pero aquellas personas que crean eso, están muy equivocadas. Este motor es utilizado en muchos otros sectores para diferentes aplicaciones [Unreal, 19], como las que se describen a continuación:

- **NASA**
  La NASA hace uso de este motor gráfico para entrenar a sus futuros astronautas, realizando así una formación más prolongada y barata de estos profesionales.

Actualmente, se les está entrenando con ejercicios de mantenimiento de la Estación Espacial Internacional, de tal manera que, se le sumerge al usuario en un entorno tridimensional haciéndole cumplir objetivos bajo varias restricciones.

Además, Epic Games y la NASA están llevando a cabo el proyecto de "Marte 2030" el cuál consistirá en vivir una experiencia virtual que lleve al usuario al "planeta rojo" sin tener que desplazarse.

– **MCLAREN**

Las aplicaciones de realidad virtual han llegado al sector automovilístico, de tal manera que, la empresa británica ha usado Unreal Engine para diseñar algunos de sus automóviles. Se sabe que McLaren, ha enviado ya a Epic Games, muestras de pintura reales, tela e incluso grabaciones del ruido del motor.

– **BMW**

Otra de las empresas automovilísticas que ha apostado por este motor es BMW, el cual, está diseñando y modelando el interior de sus vehículos para que los usuarios elijan y personalicen su futuro automóvil de forma instantánea.

– **CINE**

Unreal también ha llegado a la industria del cine, un ejemplo de ello fue la película "Invasion de Baobab Studios" desarrollada con este motor gráfico.

– **EDUCACIÓN**

En este ámbito, Unreal también se está haciendo un hueco ya que está ofreciendo a institutos y universidades el uso totalmente gratuito de este motor para poder mostrarlo a los alumnos.

4.2.2. UNREAL ENGINE VS UNITY

Previamente, a la elección del motor gráfico, se ha realizado un estudio comparativo [Unreal, 19] de los dos motores gráficos que, en la actualidad, son los más utilizados y están más desarrollados con respecto a la construcción de una aplicación como la que se pretende realizar en este proyecto. Estos motores son Unreal Engine y Unity.

A continuación, se expondrán las diferentes ventajas e inconvenientes que presentan estos dos motores para justificar la elección escogida en este proyecto

– **VENTAJAS UNREAL ENGINE**

- ✓ Calidad gráfica superior a Unity
- ✓ Permite crear juegos grandes y complejos
- ✓ Es gratuito

– **INCOVENIENTES UNREAL ENGINE**

- × Curva de aprendizaje complicada
- × Comunidad inferior a Unity (menos foros de consulta)

- **VENTAJAS UNITY**

  - ✓ Curva de aprendizaje más sencilla
  - ✓ Comunidad de desarrolladores más amplia
  - ✓ Notable integración multiplataforma

- **INCOVENIENTES UNITY**

  - × Exige mucho rendimiento al PC
  - × Calidad gráfica mucho inferior a Unreal Engine

Después de evaluar los dos motores, se llega a la conclusión de que ambos presentan características muy similares, pero en el presente proyecto, al tener que construir una escena, cuyo objeto es que el usuario experimente una sensación de realismo puro, se ha decidido la elección del motor gráfico Unreal Engine, cuya calidad gráfica es uno de los puntos más fuertes que presenta, aunque disponga de menos tutoriales o foros para la consulta del desarrollo de la aplicación.

4.3. DESARROLLO DE LA ESCENA VIRTUAL

En este apartado, se describirá el proceso cronológico que se ha seguido para desarrollar la escena virtual, desde el diseño hasta la ejecución:

1. DISEÑO DE LA ESCENA VIRTUAL

En primer lugar, al estudiar el objeto y alcance del proyecto, se decidió que la escena virtual que se desarrollaría sería la de un típico pub inglés o americano, donde se encontrara un escenario, y en el cuál, tanto el cantante como su orquesta estuvieran interpretando la pieza musical que fuera objeto del procesado de señal. En él se incorporarían, los elementos y modelos típicos de un pub como pueden ser barras, banquetas, mesas e incluso algún billar.

Como la aplicación se ha orientado a revivir conciertos de artistas que ya no se pueden volver a disfrutar, se optó por la opción de Frank Sinatra (fallecido en 1998) debido a que la música que lo acompaña se caracteriza por ser muy armónica y poco sintética, lo que facilita la separación de fuentes.

2. RECREACIÓN DE LA ESCENA VIRTUAL E IMPORTACIÓN DE MODELOS 3D

Para recrear la escena virtual, en primer lugar, se formó una caja grande, que simularía la estructura del recinto (paredes, techo y suelo) y, a continuación, se formaría el pasillo de entrada al salón donde se sitúa el escenario y las barras.

Una vez, que se decidió la canción que se interpretaría (dados los buenos resultados obtenidos en la parte de procesado de señal) se importaron a la aplicación los diferentes modelos de instrumentos que sonarían en la escena virtual, en la cual, estarían las fuentes sonoras propias de cada instrumento, extraídas de la separación de fuentes. La canción que se interpreta en la aplicación es "*Fly to the moon*" y los instrumentos que aparecen son: la batería, el bajo, la trompeta y el saxofón.

Después, se incorporó el escenario, junto con las barras, taburetes y las mesas altas. Y, por último, se importaron todos los modelos 3D referidos al decorado como pueden ser: el billar, las dianas, el array de altavoces, el micrófono, etc…

Hay que recordar que todos los modelos 3D importados a la escena virtual, además de ser gratuitos, son de formato FBX u OBJ, dado que el desarrollador de videojuegos Unreal Engine solo acepta esos formatos 3D.

Dichos modelos han sido descargados de numerosas fuentes o foros como son *TurboSquid* [Turbosquid, 19], *Free3D* [Free, 19] o *Poliigon* [Poliigon, 19], así como otras fuentes que son nombradas en la bibliografía adjunta.

Otro aspecto que cabe mencionar en esta fase, es el de los materiales, ya que cada modelo 3D se divide en varios elementos, los cuales, se caracterizan por ir asignados a un material determinado que en muchas ocasiones ha tenido que ser cambiado y actualizado por algún otro material, bien porque no era reconocido por el desarrollador de videojuegos o bien por aspectos estéticos de la escena virtual.

3. IMPORTACIÓN DE PERSONAJES Y ANIMACIONES A LA ESCENA VIRTUAL

Una vez, que todos los modelos 3D se han importado a la escena virtual, se pasa a la importación de personajes. Para comenzar, se importan los personajes que interactuarán con los instrumentos, así como el personaje que representará a Frank Sinatra y, a continuación, se importarán los demás personajes secundarios como pueden ser los camareros o el público asistente.

Para importar un personaje, primero hay que empezar por crearlo, y para ello se utiliza el software *Adobe Fuse (software de gráficos en 3D desarrollado por Mixamo),* en él se puede crear cualquier personaje indicando las características que se quieran, como pueden ser la forma y el tamaño de la cabeza, la construcción del torso, el color de piel, el pelo, la ropa o incluso la expresión de la cara de manera que cada personaje refleje una emoción u otra. Aunque en este caso se ha utilizado una versión "Beta", hay otras versiones que complementan todavía más la creación de estos personajes. A continuación, se muestra un ejemplo de la interfaz de este sencillo software.

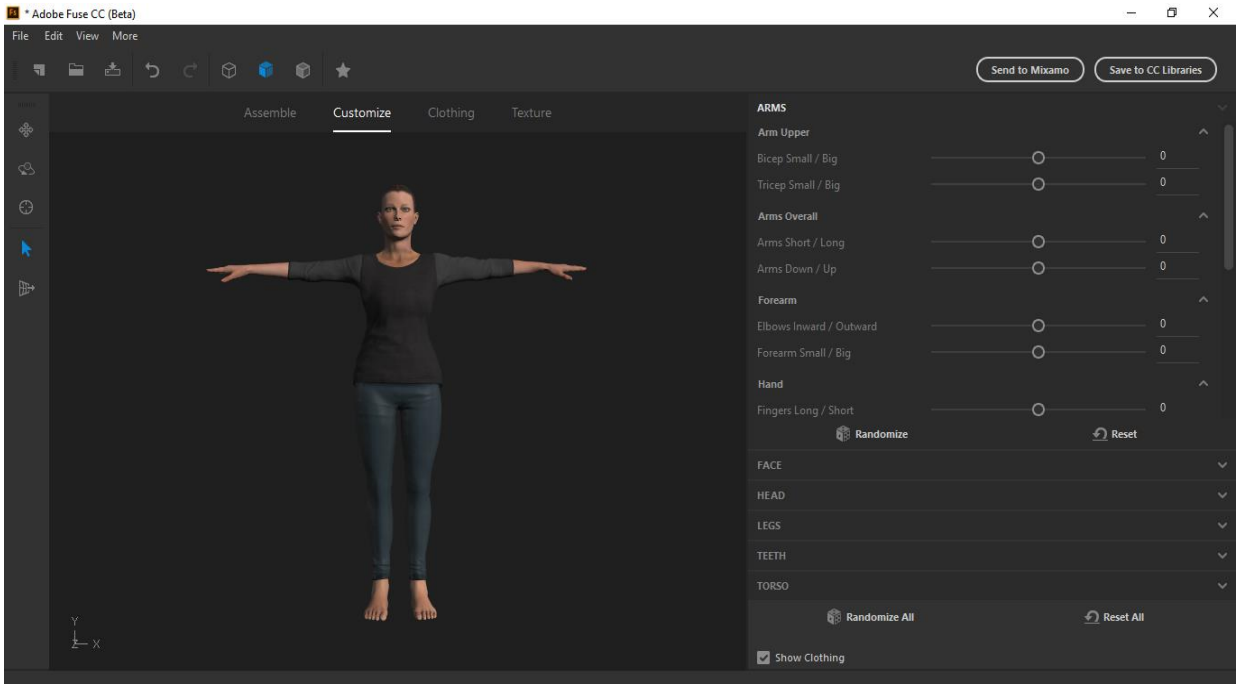


Figura 28: Interfaz gráfica de Adobe Fuse

Además, también podemos configurar la resolución de cada textura, tanto de la piel como de las diferentes prendas de ropa o el pelo; en nuestro caso, hemos configurado la resolución de cada parte al máximo (resolución de la piel 2048x2048 y resolución de las demás partes 1024x1024).

Una vez que el personaje está creado a gusto del usuario, esta plataforma tiene una opción para mandarlo directamente a la plataforma de *Mixamo*. Conviene antes de seguir, introducir y explicar en qué consiste la plataforma de *Mixamo. Mixamo* es una compañía tecnológica de gráficos en 3D, que desarrolla y vende servicios basados en web para las animaciones de personajes en 3D.

Una vez en *Mixamo* [Mixamo, 19], exportamos el personaje creado en formato *collada (.dae)* para luego importarlo en el software *Blender (software multiplataforma, dedicado al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales).* Este procedimiento, podría acortarse al descargar desde la plataforma de *Mixamo*, el personaje en formato FBX (.fbx) para poder importarlo en Unreal Engine, pero tras realizar varias pruebas, se detectó que ocurría un problema con las pestañas de los personajes que se importaban de esta manera, ya que o bien desaparecían o bien se desformaban al pasarlos a la escena virtual. Tras consultar diferentes fuentes, se optó por la solución de utilizar el software anteriormente mencionado. En Blender, se importa el personaje en formato *collada*, se accede al elemento "Pestañas" del personaje y se procede a eliminar el material predeterminado y crear uno nuevo con el nombre "Pestañas". A continuación, se exporta el personaje en formato FBX y se sube a *Mixamo*, desde donde se vuelve a descargar ahora sí en formato FBX, para después importarlo en Unreal Engine. Este personaje importado no lleva asociado ninguna animación, estas se importarán de manera independiente descargando desde *Mixamo* el esqueleto "sin piel" del personaje con una animación asociada. Se muestran, a continuación, dos figuras donde

se representan las dos plataformas usadas en el desarrollo de los personajes para la escena virtual.

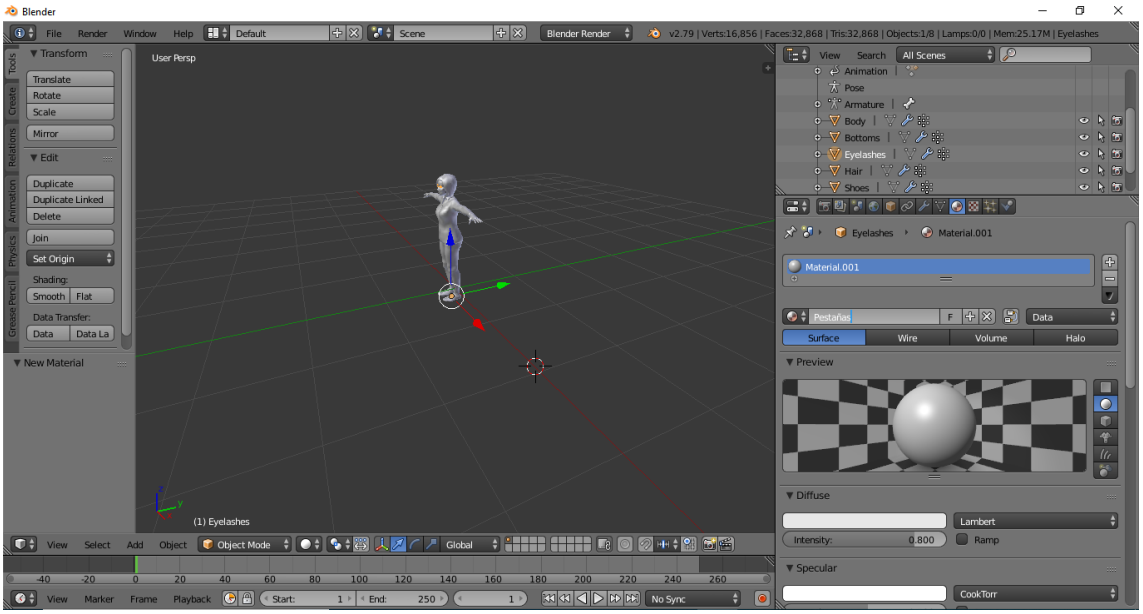


Figura 29: Interfaz gráfica de Blender

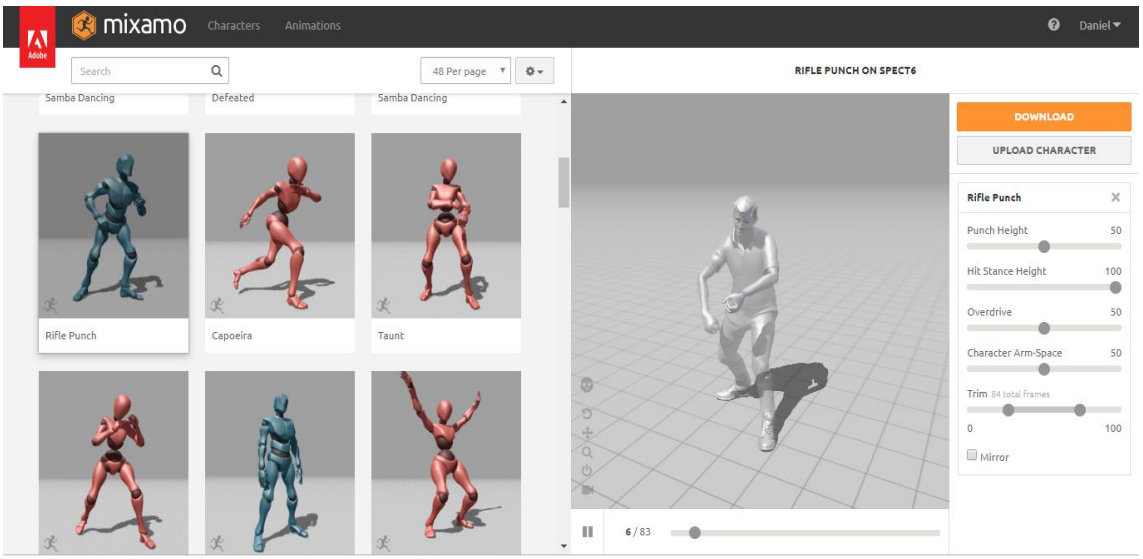


Figura 30: Plataforma de Mixamo

4. MODELO DE ATENUACIÓN DE LAS FUENTES

Dentro del objeto del presente proyecto se encontraba la interacción del usuario con la escena virtual. Bien, en la aplicación móvil, el usuario tiene la posibilidad de moverse por la escena virtual de manera que pueda alejarse o acercarse a las distintas fuentes sonoras, por lo que, si se quiere mantener el realismo y la sensación de estar dentro de la escena, se tendrá que atenuar y amplificar la fuente según la posición en la que se encuentre el usuario. Para ello, Unreal Engine presenta una opción para las distintas fuentes que se

encuentren en la escena virtual, con la cuál, se puede variar la atenuación y percepción de dicha fuente y de esa manera generar diferentes sensaciones sonoras.

En este caso, se ha optado por una atenuación con una distribución natural del sonido debido a los resultados tan realistas que generaba en la aplicación, aunque se puede elegir entre una gran variedad de opciones (logarítmica, logarítmica inversa, lineal, etc…). El sistema de atenuación de Unreal Engine consiste en lo siguiente: existen dos zonas delimitadas por dos esferas (en vez de la forma de esfera también se pueden utilizar otras formas como un cuadrado o una cápsula), dentro de la primera el sonido se mantiene sin atenuación ninguna, es decir, se escucharía como el sonido original; pero, a medida, que salimos de esa primera esfera y nos aproximamos a la siguiente, se produce una atenuación que sigue la distribución anteriormente mencionada. Cuando se llega al límite de la segunda esfera, o incluso, cuando la sobrepasamos, el sonido se mantiene con un nivel igual al valor marcado en la siguiente figura, el cuál puede ser definido con el valor que se crea conveniente. En nuestro caso, se mantendrá con un nivel de -60 dB por lo que prácticamente ni se apreciarán las fuentes sonoras.

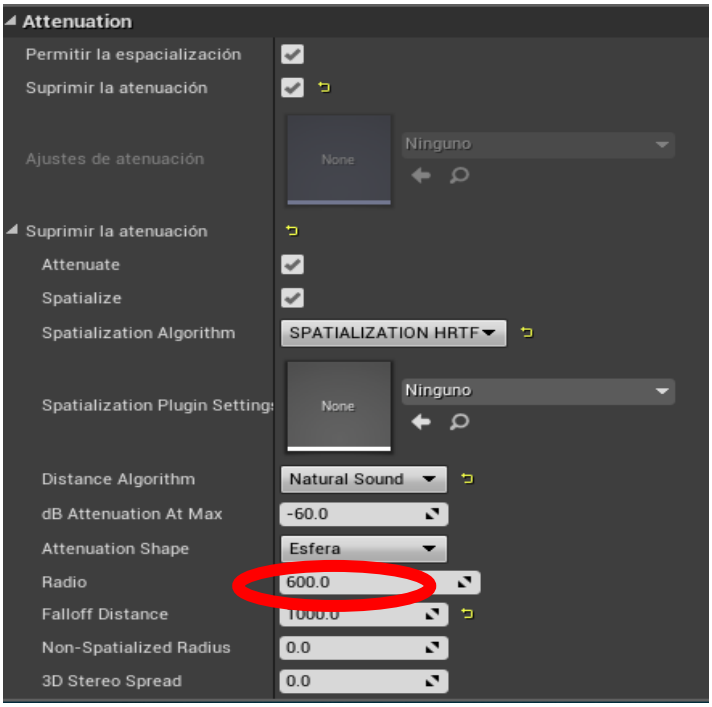


Figura 31: Ajuste de atenuación de la fuente

Figura 32: Modelo de atenuación de la fuente sonora

5. ILUMINACIÓN DE LA ESCENA VIRTUAL

Con la escena virtual ya diseñada y montada en el desarrollador de videojuegos, se procede a dar una iluminación realista y general a la escena. Para ello, se han utilizado varios "*point light*" y se han distribuido de manera uniforme por la escena, como se muestra en la siguiente imagen:

Figura 33: Iluminación de la escena virtual

Cada punto de luz, ubicado en la escena virtual, ha sido configurado con un valor de intensidad máximo y un radio de atenuación medio, de manera que las esferas de atenuación de un punto de luz y otro no interfieran entre sí.

A menudo, cuando se añade algún punto de luz, textura o modelo, o se realiza cualquier cambio en la escena virtual, aparece un error al ejecutar la aplicación que dice: *"Lighting needs to be rebuilt*. Bien, con este error Unreal Engine indica que la iluminación que se está viendo en el desarrollador es ficticia y necesita ser rediseñada. En particular, cada vez que añadimos un nuevo objeto, Unreal Engine necesita recalcular los rayos de luz de la escena que estamos desarrollando. Estos rayos de luz se calculan para ver como indicen sobre las superficies de los objetos, además de saber cómo rebotan o cómo se redibuja la

iluminación adyacente del objeto. Gracias a este hecho, se pueden ver en la escena las sombras bien definidas de cada objeto y sus texturas con un mayor realismo y calidad.

En resumen, cada vez que se añada algo nuevo a la escena se utilizará la opción de "*Rebuild*" ya que lo que ejecuta la aplicación en ese momento no se corresponde con la realidad, se necesita recalcular la luz mediante técnicas de "*raytracing*" o trazado de rayos, para recrear con ecuaciones matemáticas y modelos físicos el propio comportamiento de la luz que se tiene en la vida real

5. APLICACIÓN ANDROID PARA REALIDAD VIRTUAL

En este capítulo, se expondrán las diferentes fases que se han seguido para la obtención de una aplicación Android, en la que el usuario pueda desplazarse libremente por la escena virtual interaccionando con las distintas fuentes sonoras.

5.1. PROYECTO UNREAL EN PRIMERA PERSONA

En la fase de diseño de la aplicación, se estableció que uno de los requisitos importantes para este proyecto sería que el personaje que se moviera e interaccionara con las fuentes, estuviese en primera persona ya que, si no fuera así, la aplicación no sería válida para ejecutarla con realidad virtual.

En Unreal Engine, antes de comenzar a realizar cualquier proyecto, se pueden elegir las opciones básicas que se quiere para el proyecto en cuestión. En este caso, se escogió un proyecto en primera persona, que fuese escalable en 3D o 2D, sin contenido inicial y, por supuesto, que fuese para dispositivos móviles y no para PC.

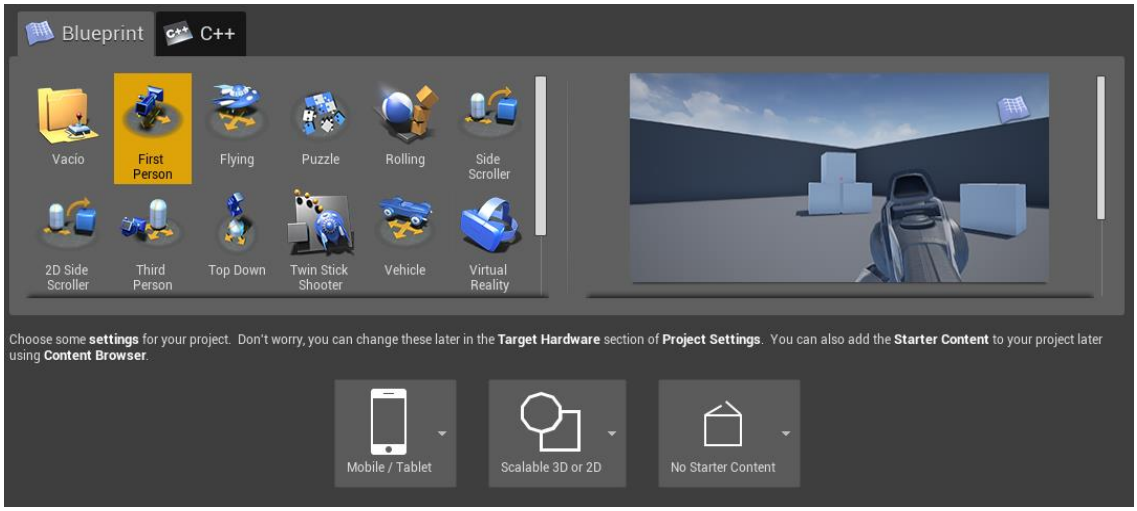


Figura 34: Creación de proyecto en Unreal Engine

El siguiente paso a la creación del proyecto, sería realizar el proceso que con anterioridad se ha descrito en el apartado 4.3 de esta memoria, en el cual se describen las fases de diseño y montaje de la escena virtual, en todos los aspectos, desde la iluminación hasta la importación de personajes y modelos 3D.

Una vez finalizada la escena virtual, se procede a la programación y configuración del "*First Person Character*", el cual va acompañado de una cámara que reproduce todo lo que va recogiendo conforme el personaje se mueve por la escena virtual, de la siguiente manera:

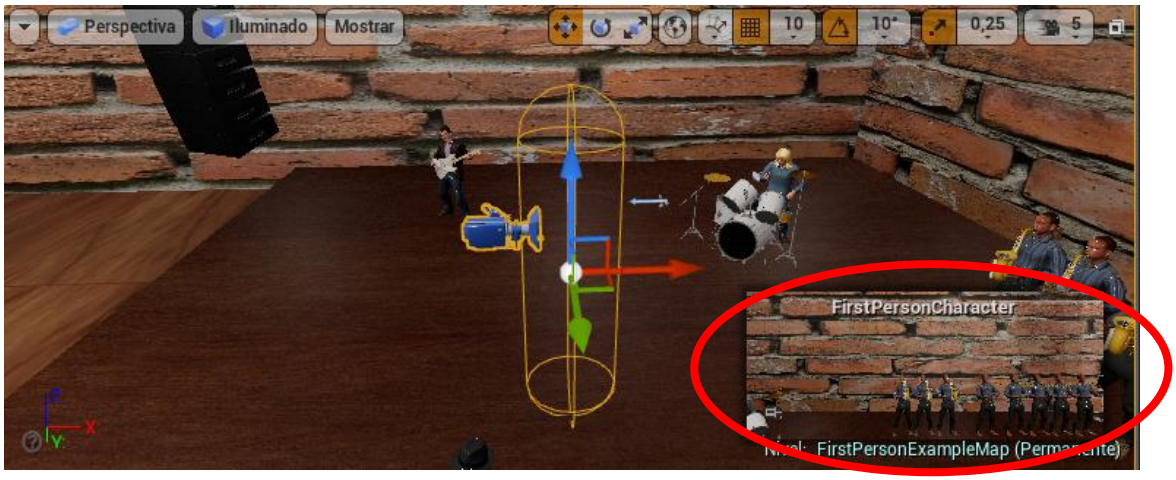


Figura 35: First Person Character en la escena virtual

La interacción con el entorno virtual por parte del "*First Person Character*" (en adelante se nombrará como personaje en primera persona) se realiza mediante un entorno gráfico donde se incorporan los eventos o funciones que el personaje puede o va a realizar. Un ejemplo de evento, sería saltar o disparar, consiste en acciones que se dan en un determinado instante de la ejecución del videojuego. En la siguiente imagen se muestra el entorno gráfico, así como un ejemplo de evento o función.

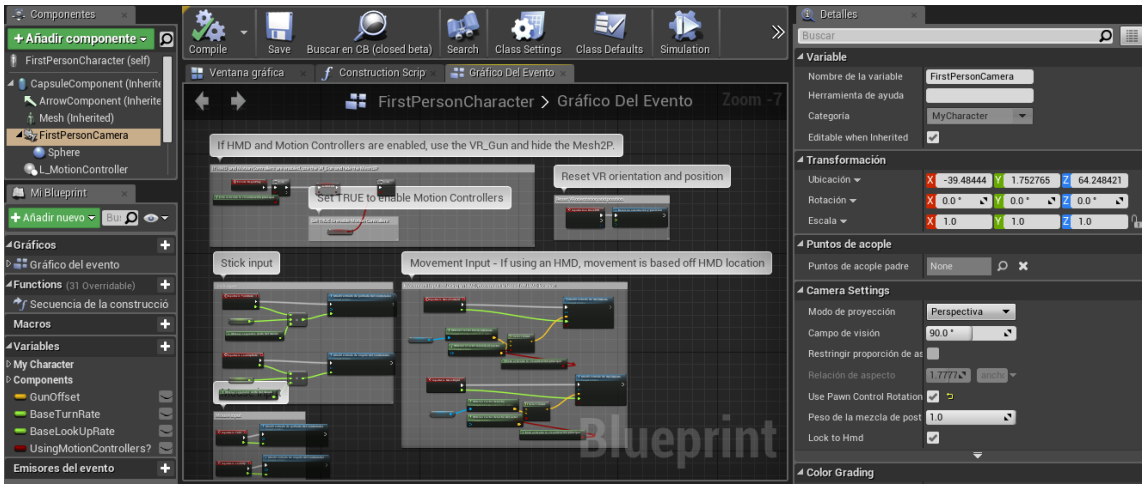


Figura 36: Entorno gráfico de programación Unreal Engine

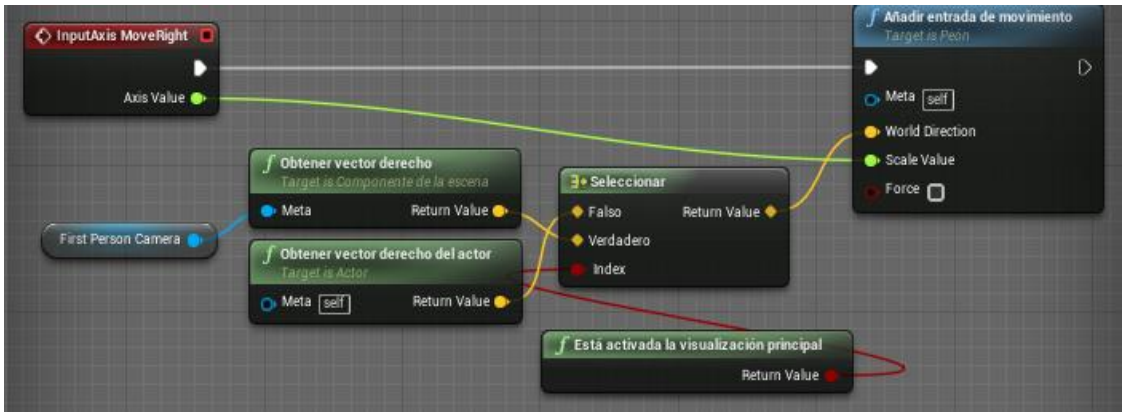


Figura 37: Evento del "First Person Character"

En este evento se describe como el personaje en primera persona se mueve hacia los lados. Dicho evento consiste en que cuando se detecta que el usuario que está controlando la aplicación desea que el personaje vaya a un lado, por ejemplo, a la derecha, el personaje en primera persona mediante la cámara que lleva asociada, obtiene el vector de la derecha y lo manda a la entrada de movimiento para permitir que el personaje se mueva hacia esa dirección. Siempre que se manda una orden de movimiento a un personaje, primero tiene que calcular el vector en esa dirección para luego ejecutarla correctamente.

Otra de las configuraciones que se ha tenido que realizar para el personaje en primera persona, ha sido la regulación tanto de la velocidad a la hora de andar como la altura a la que se encuentra la cámara de este personaje y que, en este caso, simularía lo que serían nuestros ojos y nuestra cabeza cuando se ejecute la aplicación en las gafas VR. Estos dos parámetros pueden ser configurados pinchando sobre la cámara en el entorno gráfico y dirigiéndose a los siguientes apartados:

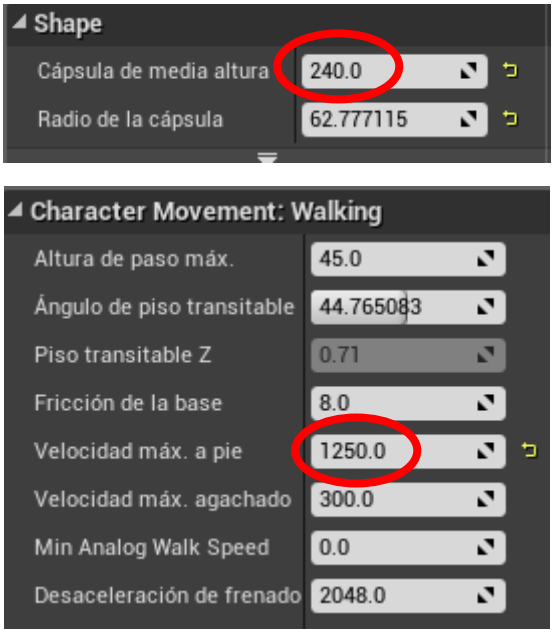


Figura 38: Ajustes del "First Person Character"

La altura media de la cápsula es un parámetro importante de regular ya que tiene que estar situada a la misma altura a la que están las cabezas de los demás personajes de la escena. Así, de esta forma, se puede proporcionar a la aplicación de un mayor realismo.

5.2. ENCAPSULACIÓN DE LA APLICACIÓN

Antes de comenzar con la encapsulación de la aplicación en el motor de videojuegos, tenemos que tener en cuenta algunos aspectos muy importantes relativos al dispositivo móvil donde vayamos a instalar la aplicación. En primer lugar, se tendrán que habilitar las opciones de desarrollador en el dispositivo móvil y, una vez activadas, dentro de estas opciones, se habilitará la opción de "Depuración USB" para que tanto el PC como el motor de videojuegos reconozcan el dispositivo móvil al conectarlo y se pueda lanzar la aplicación a dicho dispositivo.

A continuación, se procederá a instalar y utilizar el software "*NVIDIA CodeWorks for Android*" para instalar la correspondiente versión del SDK de Android, consistente en una serie de herramientas de desarrollador (depurador de código, librerías, etc…) y que es necesario para la creación del APK de la aplicación.

Una vez creado el proyecto, es necesario instalar los *plugins* de "*Google VR*" y "*Google VR Motion Controller",* para que la escena virtual que se realice pueda ser reproducida correctamente en unas gafas de realidad virtual, proyectando en el teléfono las dos cámaras que representarán la visión con cada una de nuestros ojos y, permita mover al personaje en primera persona con un joystick, que veremos con detalle a continuación.

Figura 39: Plugins a instalar en Unreal Engine

Ya instalados los correspondientes *plugins*, se pasará a configurar correctamente los ajustes del proyecto para que se pueda exportar con éxito la aplicación al dispositivo móvil. Es muy importante, tener habilitadas o deshabilitadas las siguientes opciones para el correcto funcionamiento de la aplicación.

- Deshabilitar la opción del móvil HDR (alto rango dinámico, con el que se aplica un algo contraste entre las partes más claras y más oscuras de la imagen)

- Deshabilitar la aparición de joystick virtuales en la pantalla ya que el movimiento del personaje se controlará mediante un joystick externo
- Establecer el mínimo de la versión SDK. En este caso, se ha establecido un valor de 9 (Gingerbread) ya que el dispositivo móvil que se va a utilizar es bastante actual, pero sí el móvil fuera más antiguo, se establecería un valor de 21 (Lollipop).
- Activar la opción de habilitar la pantalla completa inmersiva
- En el apartado de "Build" activaremos la opción del sistema operativo del dispositivo móvil del que dispongamos ya sea de 32 bits o de 64 bits, aunque para que sea más genérico este proyecto, se han habilitado las dos opciones.

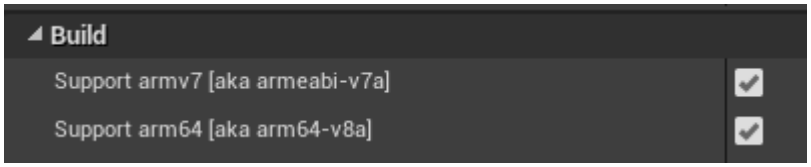


Figura 40: Ajustes del sistema operativo del dispositivo móvil

- Habilitar la opción para la configuración de Google VR en el modo rendimiento sostenido y seleccionar el hardware, según el tipo de gafas que se vayan a usar. En nuestro caso:

Figura 41: Ajustes del proyecto para VR

- Por último, establecer las rutas de los directorios donde se encuentran los archivos destinados a Android SDK, Android NDK, Java y Apache-Ant.

Cuando se termine de completar la configuración de los ajustes del proyecto, se puede proceder a conectar el dispositivo móvil al PC donde se esté desarrollando la aplicación y simplemente, con irse a la pestaña "Launch", aparecerá nuestro dispositivo móvil y se podrá comenzar con la exportación y encapsulación del proyecto en un archivo APK que, posteriormente, se instalará automáticamente en el dispositivo móvil.

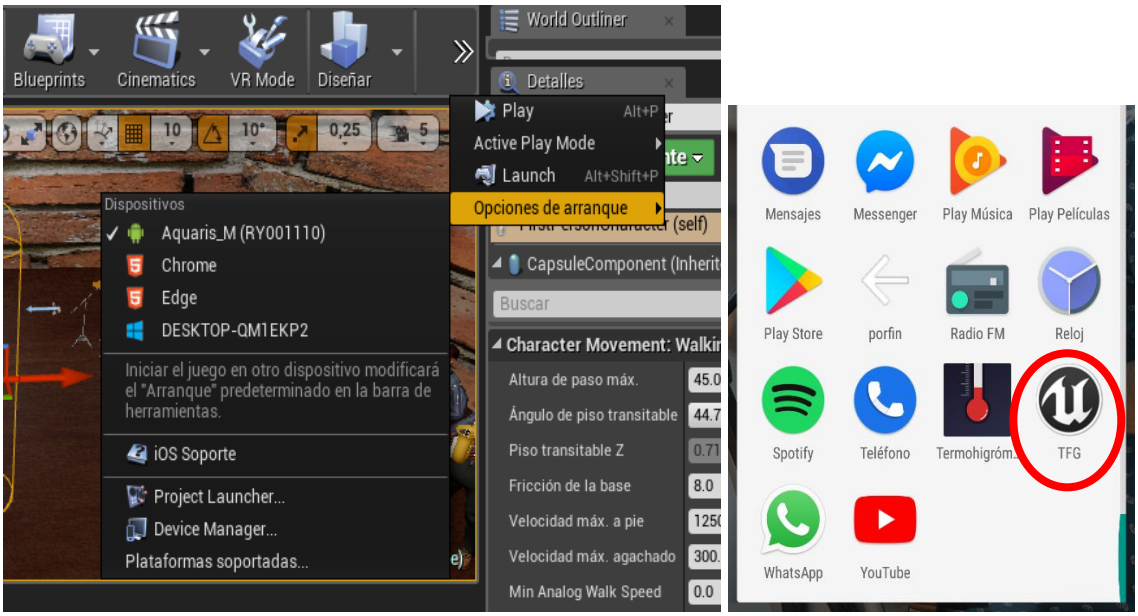


Figura 42: Lanzamiento e instalación de la aplicación en el dispositivo móvil

5.3. INTEGRACIÓN DE LA APLICACIÓN EN UN SISTEMA DE AUDIO Y VIDEO

Ya desarrollada la aplicación móvil, debemos de encajar todos los elementos necesarios para que el proyecto que se está diseñando se convierta en un sistema de realidad virtual en el cuál, se integre la capacidad de reproducir las fuentes sonoras que el personaje percibe cuando se mueve por la escena virtual.

Los elementos que constituirán este sistema serán:

- **Dispositivo móvil**: En este caso, se utilizará un teléfono móvil donde se encontrará la aplicación con la escena virtual y será ejecutada mediante la aplicación "CardBoard", que permite la reproducción de escenas en realidad virtual.
- **Gafas VR**: Se utilizarán unas gafas VR, suministradas por el departamento, que siguen la tecnología y los principios de "Google Cardboard" [Cardboard, 19], y en las cuales existe una ranura donde se encajará el dispositivo móvil para que a través de las lentes de las gafas se aprecie esa sensación de realismo que ofrece la realidad virtual.
- **Auriculares Bluetooth**: Para poder percibir los sonidos que escucha el personaje de la aplicación cuando se mueve por la escena, y para obtener una ausencia total de cables, cuya presencia puede incomodar la utilización de este sistema VR, se utilizan unos auriculares JVC con tecnología Bluetooth 3.0, los cuales se conectarán de forma inalámbrica al teléfono para poder escuchar, como se ha dicho, los sonidos que se produzcan durante la ejecución de la aplicación.

- **Joystick inalámbrico**: Por último, se hará uso de un joystick que, mediante la conexión inalámbrica Bluetooth al dispositivo móvil, permitirá que el personaje en primera persona pueda moverse y, de esa manera, permitir al usuario que ejecute la aplicación, recorrer toda la escena virtual interactuando con las distintas fuentes sonoras de los instrumentos y experimentando la sensación de realismo que se crea al ejecutar la aplicación.

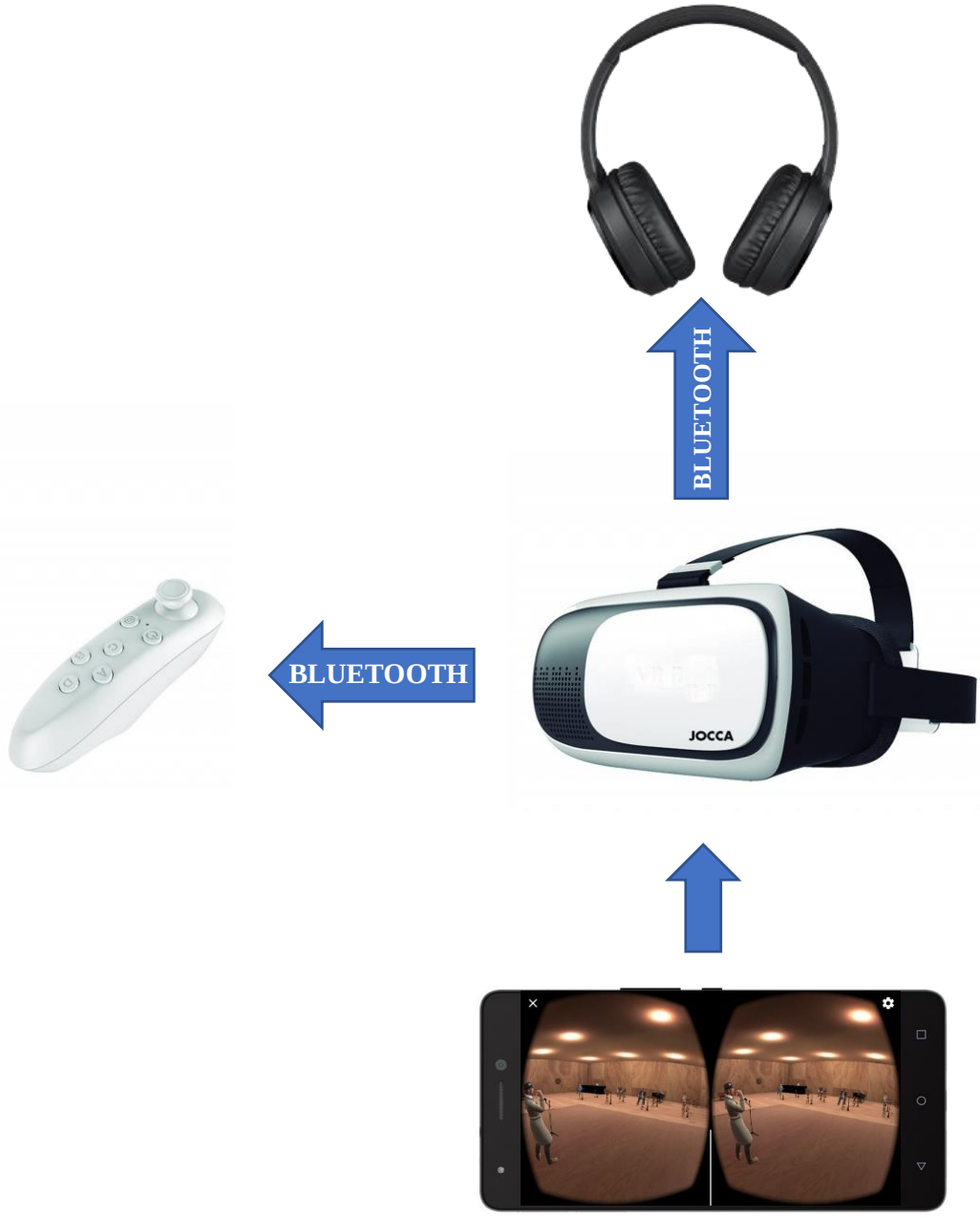


Figura 43: Esquema del sistema VR

5.4. LIMITACIONES Y FUTURAS MEJORAS

A la hora del desarrollo del proyecto, se han encontrado algunos problemas que, como consecuencia, tras no haber encontrado una solución adecuada a dicho problema, se han convertido en limitaciones para el presente proyecto.

En un principio, el diseño de la aplicación estaba pensado para que el usuario que disfrutara de la escena virtual moviéndose por ella, llegará a un punto donde, si bien se acercaba a la fuente sonora (por ejemplo, la batería), esta fuente se amplificase y, si bien se alejaba de ese punto concreto, la fuente se empezase a atenuar. Ese punto del que se habla se corresponde con el límite que establecía la primera esfera del modelo de atenuación de la fuente sonora cuyo diámetro era regulado en el motor gráfico de Unreal Engine. A continuación, se muestra un gráfico para situar el concepto del que se está hablando:

La atenuación de la fuente se llega a producir cuando el usuario sale de esa primera esfera y se aleja de ella, pero lo que no nos permite el motor gráfico, es que se produzca una amplificación de la fuente conforme nos acercamos, en este caso a la batería, sino que el volumen de la fuente se mantiene constante dentro de esta esfera.

Por lo tanto, este hecho constituye una limitación para esta aplicación, aunque se podría resolver, por ejemplo, con la utilización de otro motor de videojuegos como Unity o con una nueva actualización del motor de videojuegos Unreal Engine, en la que se permitiera la opción de amplificar la fuente a medida que el usuario se acerca a ella.

Otra limitación encontrada en el desarrollo de la aplicación ha sido la supresión del efecto de *panning* a la hora de reproducir la aplicación en un dispositivo móvil Android. Este efecto se puede apreciar perfectamente a la hora de reproducir la aplicación en el motor Unreal Engine, pero al pasarla al dispositivo móvil este efecto se pierde y no es apreciable en la reproducción de la escena virtual. Se ha consultado en numerosos foros y se han probado desde diferentes *plugins* de audio 3D hasta diferentes versiones del motor Unreal Engine, pero no se han obtenido resultados satisfactorios.

Como solución a esta limitación se han encontrado algunos *plugins* de audio 3D que se utilizan en Unity y que, según se expone en algunos foros, proporciona a la aplicación Android un efecto de *panning*. Pero debido a que toda la aplicación ya está desarrollada en Unreal Engine y conllevaría demasiado tiempo y esfuerzo rehacer toda la escena virtual en Unity, este problema de la ausencia del efecto de *panning* en la aplicación Android se establece como una limitación más para este proyecto.

6. CONCLUSIONES

Para terminar con la redacción de la memoria de este proyecto, se expondrán las diferentes conclusiones que se han obtenido, una vez el proyecto ha sido finalizado con éxito. Estas conclusiones, bajo el punto de vista del autor, pueden valorarse como muy positivas.

En primer lugar, por los nuevos conocimientos adquiridos en el ámbito del procesado de señal, en especial en el campo de la separación de fuentes sonoras, ya que era un campo donde no había llegado a entrar nunca y que, como aficionado a la música que soy, me ha encantado poder indagar e investigar y, sobre todo, haber podido obtener los resultados tan satisfactorios que se han obtenido. Es una herramienta que está en plena expansión y se debería apoyar cada vez más porque puede ser muy útil, ahora y en el futuro, para muchas aplicaciones en otros muchos sectores que no sean solo el del audio o la música.

En segundo lugar, por los conocimientos adquiridos sobre la historia, evolución y técnicas de realidad virtual, ya que era un ámbito donde tampoco tenía muchos conocimientos ni experiencia y que ha supuesto todo un reto para mí. La investigación por diferentes fuentes, así como los estudios comparativos realizados para encontrar un motor de gráficos que se adaptara lo máximo posible al objeto de este proyecto, me han aportado bastante experiencia en este campo, desconocido hasta ahora para mí.

En tercer lugar, el desarrollo de la escena virtual con el motor de videojuegos Unreal Engine fue otra de las dudas e inquietudes que se plantearon al principio de este proyecto, ante el desconocimiento que tenía de realizar este tipo de desarrollos, pero con el tiempo me he sentido muy satisfecho porque he aprendido mucho sobre el desarrollo de videojuegos y escenas virtuales, así como todos los procedimientos que se deben de seguir para la importación de modelos 3D y personajes o como se debe de programar en este entorno gráfico. Mucho tiempo de aprendizaje invertido, que finalmente, ha visto su fruto.

En cuarto lugar, el proceso de exportación de la aplicación desde el motor gráfico al dispositivo móvil también me ha aportado conocimientos nuevos debido a los numerosos errores y complicaciones que han ido surgiendo durante los primeros lanzamientos de la aplicación al dispositivo Android. Se ha entendido al detalle todo el protocolo y el proceso que se sigue entre el PC y el dispositivo móvil cada vez que éste último se conecta al él, además de investigar y adquirir conocimientos de las opciones que tiene un dispositivo móvil para poder desarrollar una aplicación.

En quinto lugar, se han aprendido nuevos conceptos sobre la plataforma Android como las herramientas de desarrollador (SDK), las características de las recientes versiones del sistema operativo o como se construye el archivo APK para poder ser utilizado en esta plataforma.

En sexto lugar, el haber investigado y estudiado los distintos sistemas de audio 3D así como su evolución e historia, me ha permitido adentrarme, aún más, en el mundo del audio y la música, del cual siempre he sido muy aficionado. En particular, creo que, si de verdad se apostará aún más por el audio y la música y se incentivara más la investigación en este campo, se podrían llegar a conseguir sistemas y técnicas increíbles.

Para finalizar este capítulo, me gustaría destacar que este proyecto ha sido todo un reto para mí desde el principio hasta al final, debido a que me he enfrentado a conceptos y

conocimientos que desconocía completamente hace algunos meses, como el desarrollo de escenas virtuales, creación de aplicaciones móviles, sistemas de realidad virtual o el procesado de señal en términos tan complejos como el que utiliza la separación de fuentes. Pese a las adversidades, puedo decir que estoy muy orgulloso de haber superado todas las dificultades que he encontrado en el camino y de haber realizado un proyecto, que ha englobado dos de las cosas, para mí, esenciales en la vida, los estudios y una gran afición como es la música.

7. BIBLIOGRAFÍA

[Senderos, 18] Los Senderos Studio, LLC, Blanco TX 78606, “Stereo Microphone Techniques”, October 2018

[Owsinski, 17] Bobby Owsinski’s, “The Recording Engineer’s Handbook”, Media Group, Enero 2017.

[Belchior, 19] Eduardo Belchior, “Coincident or Near-Coincident Mic Placement Techniques”, Universidade Federal do Estado do Rio de Janeiro.

[Pulkki, 97] Ville Pulkki, “Virtual Sound Source Positioning Using Vector Base Amplitude Panning”, Helsinki University of Technology, Finland 1997.

[Lee, 01] Lee, D.D., “Algorithms for Non-negative Matrix Factorization” Advances in neural information processing systems, 2001

[Bernalte, 14] Lucas Bernalte, “Separación de mezclas de audio en el dominio tiempo-frecuencia”, Escuela Técnica Superior de Ingeniería de Sevilla, 2014

[Sarmiento, 17] María Sarmiento, “The cocktail-party problem”, Universidad de Sevilla, Febrero 2017

[Gutiérrez, 13] Pablo Gutiérrez, “Espacialización sonora con Wavefield Synthesis y Vector Base Amplitude Panning. Estudio comparativo.”, Universidad Politécnica de Valencia (UPV), Febrero 2013.

[Perales, 15] Victor Perales, “Ambisonics como alternativa de sonido inmersivo”, Noviembre 2015

[García, 17] Javi García, “Medidas de maniquís acústicos binaurales y espacialización de sonidos”, UPV, Grupo de Tratamiento de Audio y Comunicaciones, Junio 2017.

[Unreal, 19] Unreal Engine VS Unity 3D http://www.baboonlab.com/en_US/blog/news-1/post/unreal-engine-4-el-motor-grafico-que-ofrece-realismo-al-maximo-23

[VR, 19] Introducción a la Realidad Virtual, http://www.lsi.upc.edu/~pere/SGI/guions/ArquitecturaRV.pdf

[Free, 19] Free 3D Model https://free3d.com/

[Poliigon, 19] Poliigon Textures and Models 3D https://www.poliigon.com/

[Turbosquid, 19] TurboSquid https://www.turbosquid.com/

[Android, 19] Android Studio https://developer.android.com/studio

[Mixamo, 19] Mixamo Adobe https://www.mixamo.com/#/

[Cardboard, 19] Google Cardboard https://www.google.com/get/cardboard

[Unreal, 19] Unreal Engine Forums https://forums.unrealengine.com/

[Rodríguez, 14] Francisco José Rodriguez, “Tesis Doctoral: Separación de fuentes sonoras en señales musicales”, Escuela Politécnica Superior de Linares, Julio 2014.

[Verdoodt, 15] Diego Verdoodt, “Concierto de música clásica en Realidad Virtual”, Universidad Pompeu Fabra de Barcelona, Junio 2015.

[Cañadas, 14] Francisco Jesús Cañadas-Quesada, Pedro Vera-Candeas, Nicolas Ruiz-Reyes, Julio Carabias-Orti y Pablo Cabañas-Molero, “Percussive/harmonic sound separation by non-negative matrix factorization with smoothness/sparseness constraints”, EURASIP Journal on Audio, Speech, and Music Processing, 2014 - 1, Springer, 2014

[Cabañas, 16] Pablo Cabañas, “Tesis: Classification and separation Techniques based on fundamental frequency for speech enhancement”, Universidad de Jaén, Enero 2016.

[Carabias, 13] Julio J. Carabias-Orti, Máximo Cobos, Pedro Vera-Candeas, Francisco J. Rodrigue-Serrano, “Nonnegative signal factorization with learnt instrument models for sound source separation in close-microphone recordings”, EURASIP Journal on Advances in Signal Processing, December 2013

[Durrieu, 10] JL Durrieu, G. Richard, B. David, C. Févotte, “Source/filter model for unsupervised main melody extraction from polyphonic audio signals”, IEEE Transactions on Audio, Speech, and Language Processing 18 (3), March 2010.

[DSD100, 19], https://sigsep.github.io/datasets/dsd100.html

[SISEC,17], https://www.sisec17.audiolabs-erlangen.de/#/methods

[CHA, 17] P. Chandna, M. Miron, J. Janer, and E. Gomez, “Monoaural audio source separation using deep convolutional neural networks,”, 13th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2017).

[DUR, 11] J.-L. Durrieu, B. David, and G. Richard, “A musically motivated mid-level representation for pitch estimation and musical audio source separation,” IEEE Journal on Selected Topics on Signal Processing, vol. 5, no. 6, pp. 1180–1191, Oct. 2011.

[GRA2, 16] [GRA3, 16] E. Grais, G. Roma, A.J. Simpson, M. Plumbley, “Single-Channel Audio Source Separation Using Deep Neural Network Ensembles. ” Proc. AES 140, 2016, May.

[HUA, 12] P. Huang, S. Chen, P. Smaragdis, and M. Hasegawa-Johnson, “Singing-voice separation from monaural recordings using robust principal component analysis,” in Proc. ICASSP, Mar. 2012, pp. 57–60.

[IBM, 19] Ideal Binary Mask

[JEO1, 19] [JEO2, 19] I.-Y. Jeong and K. Lee, Singing voice separation using RPCA with weighted l1-norm

[KAM1, 15] [KAM2, 15] A. Liutkus, D. FitzGerald, Z. Rafii, and L. Daudet, “Scalable audio separation with light kernel additive modelling,” in Proc. ICASSP, Apr. 2015, pp. 76–80

[KON, 19] Huang, Po Sen, et al. Joint optimization of masks and deep recurrent neural networks for monaural source separation.

[NUG1, 16] [NUG2, 16] [NUG3, 16] [NUG4, 16] A. A. Nugraha, A. Liutkus, and E. Vincent, "Multichannel music separation with deep neural networks," in Proc. 24th European Signal Processing Conference, Budapest, Hungary, Aug. 2016.

[OZE, 12] A. Ozerov, E. Vincent, and F. Bimbot, “A general flexible framework for the handling of prior information in audio source separation,” IEEE Trans. ASLP, vol. 20, no. 4, pp. 1118–1133, Oct. 2012.

[RAF1, 13] "Z. Rafii and B. Pardo, “REpeating Pattern Extraction Technique (REPET)": "A simple method for music/voice separation,” IEEE Trans. ASLP, vol. 21, no. 1, pp. 71–82, January 2013."

[RAF2, 12] A. Liutkus, Z. Rafii, R. Badeau, B. Pardo, and G. Richard, “Adaptive filtering for music/voice separation exploiting the repeating musical structure,” in Proc. ICASSP, Mar. 2012, pp. 53–56.

[RAF3, 12] Z. Rafii and B. Pardo, “Music/voice separation using the similarity matrix,” in Proc. ISMIR, Oct. 2012, pp. 583–588.

[STO1, 16] [STO2, 16] F.-R. Stöter, A. Liutkus, R. Badeau, B. Edler, and P. Magron, “Common Fate Model for Unison source Separation,” in Proc. ICASSP, 2016.

[UHL1, 16] [UHL2, 16] [UHL3,16] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, Y. Mitsufuji, N. Takahashi, Combining DNNs for Enhanced Music Source Separation, to appear.