Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo y evaluación de métodos para reducir la reverberación en señales de voz

Development and evaluation of methods to reduce reverberation in speech signals

Autor/es

Enol Ayo Sando

Director/es

Alfonso Ortega Giménez

Titulación del autor

Ingeniería de Tecnologías y Sistemas de Telecomunicación

Escuela de Ingeniería y Arquitectura

2022

Resumen

La reverberación es el proceso físico que sucede cuando se producen reflexiones del frente de ondas, las cuales llegan al oyente de forma retardada, pudiendo degradar la calidad e inteligibilidad de una señal de voz. En los últimos años, la investigación sobre el procesamiento de señales de voz reverberante ha experimentado un progreso significativo, y estas técnicas están listas para ser incorporadas a nuestras aplicaciones cotidianas. En este TFG se planteará el estudio, desarrollo y evaluación de algunos algoritmos prácticos que puedan reducir el efecto perjudicial de la reverberación, partiendo de algoritmos clásicos e incorporando técnicas más modernas que incluyen el uso de redes neuronales.

Tabla de contenidos:

1. Introducción

1.1. Motivación y alcance

La reverberación se produce cuando, además del frente de ondas principal que sale de la fuente del sonido, llegan otros frentes con diferentes retardos, y que pueden estar más o menos atenuados. Que el objetivo de este trabajo sea eliminarla o mitigarla no significa que siempre sea indeseable. Por ejemplo, se puede aplicar la reverberación para simular la acústica de distintos tipos de salas y también se utiliza en la producción musical. No obstante, es totalmente indeseable en comunicaciones por voz, ya que su existencia produce varios inconvenientes, como son el efecto de que el oyente está lejos del micrófono y dificultar la comprensión del habla[1]. También es un fenómeno indesado en los sistemas de reconocimiento automático del habla (RAH o en inglés, ASR, Automatic Speech Recognition) o de reconocimiento automático del hablante (ASV, Automatic Speaker Verification) ya que constituye una fuente de degradación de prestaciones respecto a los que estos sistemas presentan cuando la señal de entrada es limpia, libre de ruido y reverberación.

Entre las aplicaciones de los algoritmos de desreverberación se encuentran su aplicación a la telefonía portátil. Si bien el efecto no es muy perjudicial en una sala normal, se ha probado que se producen degradaciones en la calidad del sonido en salas como escaleras o pasillos [2]. Otra aplicación interesante es su uso en el procesado de audio binaural. Con este procesado se trata de producir una sensación de audición realista mediante el uso de auriculares, en la cual se pueden percibir las direcciones de las fuentes de cada sonido. Con los algoritmos de desreverberación se consiguen mejorar las prestaciones de este tipo de procesado [3], ya que a la fuente original del sonido se le añaden otras con distintas direcciones debido a las reflexiones. Por último, podemos mencionar también su aplicación en algoritmos de reconocimiento del habla (ASR), donde la aplicación de estos algoritmos a la señal que queremos reconocer junto con un entrenamiento del modelo mediante señales a las que se le añade reverberación (lo que en redes neuronales se conoce como aumento de datos), logra mejorar los resultados.

Dada la degradación que la reverberación introduce tanto en la inteligibilidad del lenguaje como en los sistemas de reconocimiento automático del habla, resulta conveniente abordar la problemática y estudiar las técnicas que reduzcan en medida de lo posible sus efectos

1.2. Contexto

En los últimos años se han llevado a cabo grandes avances en el campo del realce del habla. Esto ha sido posible gracias a la evolución de herramientas como la acústica de salas, el filtrado óptimo, *machine learning*, modelado del habla, etc.

Existen tres principales aproximaciones al problema de la desreverberación[1]. Por un lado, tenemos el *beamforming* mediante el uso de arrays de micrófonos, con el que se intenta compensar los distintos tiempos de llegada de cada reflexión añadiendo distintos retardos a cada uno de ellos, de forma que se refuerza el frente principal que viene de la fuente y se atenúan en

la medida de lo posible las reflexiones. Si bien puede resultar una aproximación interesante al problema, tiene el inconveniente de que se necesitan varios micrófonos, por lo que no es una solución simple y económica.

Por otro lado, existen las aproximaciones del *Speech Enhancement*, las cuales utilizan principios de técnicas para la reducción del ruido y no hacen suposiciones sobre las propiedades de las salas, sino que tratan de eliminar la reverberación tardía en el dominio espectral. La ventaja de estos métodos es que son eficientes computacionalmente y que se pueden llevar a cabo con un solo micrófono, aunque por otro lado sus prestaciones están limitadas, comparando con otras aproximaciones[4]

Finalmente está la aproximación en la que está basado este trabajo, que se conoce como *Blind deconvolution* o deconvolución ciega. El objetivo es intentar estimar la respuesta impulsional de la sala y proceder a convolucionar la señal de voz con la respuesta impulsional inversa. Estas técnicas son las que más se han estado investigando en los últimos años y pueden llevarse a cabo de forma eficiente con el uso de un micrófono o mejorando las prestaciones aumentando el número de éstos.

1.3. Objetivos

El objetivo de este trabajo es la evaluación de los métodos existentes mediante las métricas propuestas en el Reverb Challenge [5], que fue una campaña de evaluación de distintos algoritmos de desreverberación que tuvo lugar en 2014. Probaremos diferentes implementaciones del algoritmo WPE (*Weighted Prediction Error*)[6], desarrollado por la *Nippon Telegraph and Telephone Corporation* en el año 2010. Partiremos desde las más clásicas e incorporando posteriormente herramientas como el *Deep Learning* para tratar de mejorar las prestaciones. También comprobaremos como influye el número de canales de audio considerados y el tipo de sala en el que se han llevado a cabo las grabaciones, entre otras evaluaciones.

2. Método de realce o desreverberación

La primera implementación que vamos a evaluar es la conocida como NDLP (*Variance-Normalized Delayed Linear Prediction*)[7], aunque es más conocida como WPE (*Weighted Prediction Error*), y es la implementación original que podemos encontrar en [6]. Como mencionamos anteriormente, nos centramos en los algoritmos que buscan estimar la respuesta impulsional de las salas. A estas funciones se les conoce como RTFs (*Room Transfer Functions*), y una vez estimadas se procede a filtrar la señal de voz con la RTF inversa. Es importante destacar que para garantizar la existencia de las RTFs el número de micrófonos debe ser mayor o al menos igual que el número de fuentes de sonido[7].

Este algoritmo tiene como desafíos distinguir las características propias del tracto vocal de las que son provocadas por la sala, regular el sistema para controlar el ruido, ya que los algoritmos de desreverberación tienden a amplificarlo, y tratar de llevar a cabo la estimación de la RTF con unos pocos segundos de observación de la señal, con el fin de que sea eficiente computacionalmente.

Como veremos, esta aproximación está basada en la predicción lineal multicanal y se diferencia de otros algoritmos anteriores en que tiene en cuenta que las propiedades estadísticas del habla no se mantienen constantes en el tiempo.

2.1. Definición de NDLP

En primer lugar, modelamos la señal captada por el micrófono como:

$$x_t^{(m)} = \sum_{k=0}^{L_h-1} h_k^{(m)} s_{t-k} + b_t^{(m)} \qquad (1)$$

x_t representa la señal captada por el micrófono *m*, la cual se obtiene mediante la convolución de la señal de voz pura s_t con la respuesta impulsional que modela el canal *h*. Por último, b_t representa el ruido captado por el micrófono.

Suponiendo que hay dos canales por simplicidad, el esquema que representa el sistema que buscamos es el siguiente:

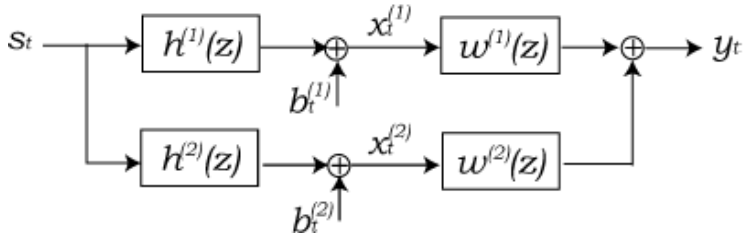

Figura 1: Esquema del sistema NDLP. Fuente: [7]

El objetivo es encontrar los filtros de desreverberación $w^{(m)}$. Para ello primero tratamos de modelar la señal de forma que diferenciemos que parte de la señal nos interesa y cual

consideramos reverberación. Hay que tener en cuenta que en los desarrollos que vienen a continuación vamos a obviar el efecto del ruido por la complejidad a la que nos llevaría.

$$\hat{d}_t^{(m)} = x_t^{(m)} - (\hat{\bar{c}}^{(m)})^T \bar{x}_{t-D} \tag{2}$$

En la ecuación (2), $\widehat{d_t}$ representa la estimación de la señal deseada, mientras que $\hat{\bar{c}}$ son los coeficientes de regresión que hemos de estimar también. *D* representa el *delay* o retardo de predicción, con el cual diferenciamos la señal en dos partes: por un lado, la parte anterior al retardo se corresponde con la señal directa y la reverberación temprana, mientras que la parte posterior al retardo es la reverberación tardía.

La solución propuesta para el problema consiste en maximizar la función de verosimilitud:

$$\begin{aligned} &\mathcal{L}(\theta) = \log p(\bar{x}_{\langle \tau, -\infty \rangle}; \theta) \\ &\theta = \left\{\bar{c}^{(1)}, \bar{c}^{(2)}\right\} \end{aligned} \tag{3}$$

Dicho de otra forma, debemos buscar los coeficientes de regresión $\bar{c}^{(1)}$ y $\bar{c}^{(2)}$ que maximizan el logaritmo de la función de verosimilitud de $\bar{x}$. El logaritmo aparece ya que es habitual en este método de optimización, puesto que facilita las operaciones y da lugar al mismo resultado ya que se trata de una función monótona creciente. Tras unas cuantas manipulaciones matemáticas llegamos a la siguiente aproximación:

$$\begin{aligned} &\mathcal{L}(\theta) \approx \sum_{t=1}^{\tau} \log p(\, x_t^{(1)} | \bar{x}_{t-D}; \theta) \\ &= \sum_{t=1}^{\tau} \log p(\, d_t^{(1)} = x_t^{(1)} - \left(\bar{c}_t^{(1)}\right)^T \bar{x}_{t-D}; \theta) \end{aligned} \tag{4}$$

Como vemos en la ecuación (4), solo se aparece $\bar{c}^{(1)}$, ya que al haber despreciado el ruido podemos considerar que es estadísticamente independiente de $\bar{c}^{(2)}$.

El siguiente paso es determinar un modelo estadístico para la señal. Tomamos fragmentos de la señal de longitud L_f y asumimos que consiste en un proceso cuasiestacionario y que sólo existe correlación entre muestras en un pequeño intervalo de tiempo. Cada trocito de longitud L_f se considera que es un proceso gaussiano de media cero (ecuación (5)). Como es bien sabido, una combinación lineal de variables con distribución gaussiana da lugar a otra distribución del mismo tipo, por lo que d_t, que es la variable que aparece en la función de densidad de probabilidad que queremos maximizar también lo es:

$$p\left(d_t^{(1)}\right) = \mathcal{N}\left(d_t^{(1)}; 0, \sigma_t^2\right) \tag{5}$$

El problema ahora es determinar la varianza de cada fragmento de señal. Para tratar de aproximarnos a las propiedades "reales" del habla consideramos que la varianza puede cambiar en cada fragmento, aunque es constante dentro de cada uno. Siendo realistas, la anterior

aproximación difiere notablemente de las propiedades estadísticas del habla, pero es una aproximación con la cual es sencillo de trabajar y que da buenos resultados. La expresión para estimarla la podemos ver en la ecuación (6).

$$\sigma_t^2 \approx \frac{1}{L_f} \sum_{t'=t-\frac{L_f}{2}+1}^{t+\frac{L_f}{2}} \left| d_{t'}^{(1)} \right|^2 \tag{6}$$

Finalmente, nos queda la siguiente expresión como función de verosimilitud:

$$\begin{aligned} &\mathcal{L}(\theta') = \sum_{t=1}^{\tau} \log \mathcal{N}\left(d_t^{(1)} = x_t^{(1)} - \left(\bar{c}_t^{(1)}\right)^T \bar{x}_{t-D}; 0, \sigma_t^2\right) \\ &= -\frac{1}{2} \sum_{t=1}^{\tau} \frac{\left| x_t^{(1)} - \left(\bar{c}_t^{(1)}\right)^T \bar{x}_{t-D} \right|^2}{\sigma_t^2} \\ &-\frac{1}{2} \sum_{t=1}^{\tau} log\ \sigma_t^2 + const. \\ &\theta' = \left\{\bar{c}^{(1)}, \bar{\sigma}^2\right\} \\ &\bar{\sigma}^2 = \{\sigma_1^2, \sigma_2^2, \dots\} \end{aligned} \tag{7}$$

Dada la complejidad de encontrar una solución exacta para la ecuación (7) (tenemos tanto dependencia de la varianza como de los coeficientes de regresión), debemos encontrar una forma alternativa de maximizarla. Esto se puede llevar a cabo de forma iterativa alternando actualizaciones de $\bar{c}^{(1)}$ y $\bar{\sigma}^2$. En la primera iteración, calcularemos directamente la varianza de la señal reverberante (ecuación (8)), ya que todavía no hemos podido estimar la señal deseada. Se elige un valor mínimo ϵ para evitar la división por cero.

$$\hat{\sigma}_t^2 = max\left\{ \frac{1}{L_f} \sum_{t'=t-\frac{L_f}{2}+1}^{t+\frac{L_f}{2}} \left| x_{t'}^{(1)} \right|^2, \epsilon \right\} \tag{8}$$

Posteriormente estimaremos los coeficientes de regresión mediante el uso del filtrado óptimo (ecuaciones (9), (10), y (11)) y estimaremos la señal deseada con estos coeficientes (ecuación (2)). En la siguiente iteración ya podemos calcular la varianza de la señal deseada (ecuación (12)) y volvemos a estimar los nuevos coeficientes del filtro. Este procedimiento se repite sucesivamente hasta que obtenemos la señal sin reverberación.

$$\hat{\bar{c}}^{(1)} = \hat{\Phi}^{+} \hat{\phi} \tag{9}$$

$$\hat{\Phi} = \sum_{t=1}^{\tau} \frac{\bar{x}_{t-D}\bar{x}_{t-D}^{T}}{\hat{\sigma}_t^2} \tag{10}$$

$$\hat{\phi} = \sum_{t=1}^{\tau} \frac{\bar{x}_{t-D}x_t^{(1)}}{\hat{\sigma}_t^2} \tag{11}$$

$$\hat{\sigma}_t^2 = max\left\{\frac{1}{L_f}\sum_{t'=t-\frac{L_f}{2}+1}^{t+\frac{L_f}{2}} \left|d_{t'}^{(1)}\right|^2, \epsilon\right\} \tag{12}$$

Viendo el resultado, es destacable que la solución sólo depende de las propiedades estadísticas del habla, la reverberación que se produzca antes del retardo, y de la varianza estimada. Además, con observaciones suficientemente largas, la señal deseada solo está compuesta por la señal directa y esa reverberación, con solo una parte de ella distorsionada, mientras que la reverberación que llegara después del retardo es eliminada completamente. La principal aportación de este método es la normalización con la varianza, que logra mejorar las prestaciones siempre y cuando se ésta se haya estimado correctamente.

Hasta ahora, hemos revisado sin muchos detalles la implementación en el dominio del tiempo con el fin de ver el trasfondo matemático de este método. No obstante, existe otra implementación en la cual se descompone la señal por bandas de frecuencia mediante la STFT (*Short Time Fourier Transform)* o transformada de Fourier localizada y se procesa cada una de ellas por separado, para posteriormente sintetizarlas en una sola señal.

Con esta implementación se consigue por un lado mejorar el coste computacional del algoritmo, ya que se reduce el tamaño de los vectores y matrices que aparecen en la resolución del problema, y por otro lado se consiguen mejores resultados comparando con otros métodos cuando la observación de la señal es pequeña.

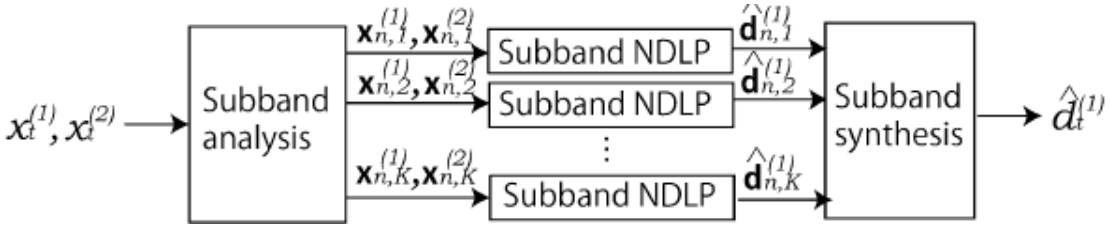

Figura 2: Esquema de NDLP por bandas de frecuencia. Fuente: [7]

2.2. DNN-WPE

Como su nombre indica, el último de los métodos que vamos a ver también está basado en el algoritmo WPE y está desarrollado en [8]. El prefijo DNN significa *Deep Neural Network*, ya que se utiliza una red neuronal con el fin de mejorar las estimaciones que vamos a ver.

Como mencionamos al explicar el algoritmo WPE, se han de estimar tanto la varianza como los coeficientes de la regresión lineal iterativamente para mejorar la desreverberación con cada iteración. En el WPE convencional, en la primera iteración se obtiene la varianza calculándola directamente a partir de la señal observada, para posteriormente estimar los coeficientes de regresión. También debemos recordar que el algoritmo WPE se podía llevar a cabo por bandas de frecuencia transformando la señal previamente mediante la STFT. En DNN-WPE se hace de esta manera, por lo que, en lugar de estimar la varianza, se estima el espectro de potencia en el dominio de la frecuencia. Esto es debido a que, para una distribución gaussiana de media nula, la transformación de la varianza a este dominio se corresponde precisamente con el espectro de potencias.

En la estimación del espectro es donde se hace uso de la red neuronal. El primer paso es entrenar la red dándole como entrada el set de señales de entrenamiento del Reverb Challenge (Señales con reverberación simulada y ruidosas) y las correspondientes señales deseadas. Las señales deseadas están disponibles debido a que la reverberación es simulada convolucionandolas con la respuesta impulsional de distintas salas. La red trata de estimar el espectro de la señal deseada a partir de la señal reverberante y se obtiene el error cuadrático medio de su resultado con el deseado. De esta forma, el error se va reduciendo conforme la red aprende hasta que se considere que cumple los objetivos.

Para ser más precisos, sea $F[\,]$ la función de la red neuronal que estima el espectro y, $\lambda(n)$ el espectro de la señal para cada fragmento de tiempo n, el entrenamiento se realiza de la siguiente forma:

$$\log\left(\lambda(n)\right) = F[\log(|x(n)|^2)] \tag{13}$$

Una vez llevado a cabo el entrenamiento, utilizaremos la función de la red $F[\,]$ para estimar el espectro y calculamos los coeficientes del filtro $\hat{g}(k)$en el dominio de la STFT mediante el uso del filtro óptimo similar al de Wiener, sin olvidarnos de la normalización con la varianza/espectro:

$$\hat{g}(k) = R^{-1}r \tag{14}$$

Donde R es la matriz de autocorrelación y r el vector de autocorrelación.

Cabe destacar que, al haber estimado el espectro mediante la red, ya no es necesario el proceso de iteraciones que se llevaba a cabo en NDLP, por lo que cabría esperar que la desreberveración sea llevada a cabo más rápidamente.

3. Marco experimental

3.1. Trabajo y conocimientos previos

El objetivo de este primer apartado es explicar el trabajo previo que tuve realizar antes de estar en la disposición de evaluar los algoritmos mencionados, así como mencionar los conocimientos necesarios para llevar a cabo dicha tarea, algunos de los cuales ya poseía, pero no la mayoría.

En primer lugar, hay que comentar que, dada la naturaleza del trabajo, el cual requiere del procesamiento de una base de datos compuesta por cientos de señales, era inviable realizarlo en un ordenador convencional debido a los larguísimos tiempos de ejecución que supondría. Por tanto, el trabajo se realizó en un *cluster* de computación de la Universidad, al cual me conectaba remótamente con el protocolo SSH desde mi ordenador con el programa *Visual Studio Code*. Esto implicaba que el primer reto era familiarizarse con el control mediante los comandos de Bash[9]. Si bien ya conocía algunos comandos, llevó algún tiempo el conocer todos los necesarios y poder trabajar con soltura en ese entorno.

Pese a la gran capacidad de computación del *cluster*, al tratarse de un recurso compartido con otros usuarios, se requería el uso de un planificador de tareas que está instalado en el servidor, el cual se conoce como *Condor*[10]. Al planificador se le envían las tareas que quieres ejecutar, éste las pone en cola y las va lanzando en paralelo en las distintas máquinas que componen el *cluster.* Como cada tarea que lanzaba consistía en procesar la base de datos completa con cada algoritmo, haciendo uso de *Condor* pude dividir cada tarea en varias subtareas, donde en cada una de ellas se procesaba una lista de señales. Con este procedimiento se conseguía poner en paralelo las subtareas y lograr tiempos de ejecución más cortos. Si bien el uso del planificador fue una gran ayuda, ya que no tenía que esperar tantas horas para llegar a resultados, también fue un desafío, ya que las tareas se lanzaban mediante un *script* de *Shell* y yo nunca había trabajado con ese lenguaje.

La parte más importante del trabajo consistió en adaptar los códigos de cada algoritmo para poder trabajar con la estructura de archivos y directorios de la base de datos, además de corregir alguna línea de código que daba lugar a errores. En caso del NDLP, utilizamos una implementación en el dominio tiempo-frecuencia encriptada para MATLAB denominada wpe.p, desarrollada por los creadores del método[6]. Al tratarse de un código de MATLAB no tuve ningún problema al tener amplia experiencia en ello. El trabajo que realicé consistió en un script de MATLAB que cargaba las señales necesarias (según el número de canales requerido, y utilizando las listas de señales del Reverb Challenge) y las procesaba con wpe.p, para posteriormente guardar la señal de salida en el directorio adecuado, ya que para que funcionara correctamente el script con el que se obtienen las métricas se necesita que las señales estén organizadas de una forma particular. En este caso no hubo problemas y el código funcionaba al primer intento.

Posteriormente, probamos otra implementación de WPE para *Python* llamada NARA-WPE[11]. Si bien nunca habíamos trabajado en *Python* a lo largo del grado, unos meses antes de comenzar este trabajo hice un curso *online* de dicho lenguaje, puesto que tenía pensado utilizarlo en el TFG por la importancia que tiene en el contexto actual. Para este algoritmo tuve que escribir un código equivalente al necesario en NDLP, pero en *Python.*

Por último, en DNN-WPE el trabajo que tuve que llevar a cabo fue algo distinto que en los casos anteriores. Primero hubo que entrenar la red con un código al que hubo que hacerle algunas correcciones para que no produjera errores. En este método, el código que había ya estaba preparado para leer las señales correctamente, pero organizaba las señales de salida de una forma que no se podían calcular las métricas. Por ello, hice un código en *Python* que las reorganizaba en directorios de la forma que necesitábamos. Además, hubo que corregir algunas líneas del código, ya que sólo funcionaba en principio para 8 canales.

Una vez hecho todo el trabajo anterior, solo restaba procesar la base de datos y hacer pruebas con una serie de parámetros para ver como influían en el resultado.

3.2. *Set* de datos

La base de datos de señales que vamos a utilizar para la evaluación de los algoritmos está basada en el recopilatorio WSJCAM0[12], y se puede obtener en la página web del Reverb Challenge[5]. La base de datos está compuesta por tres *sets* [13]: uno de señales de desarrollo, uno de señales de evaluación, y uno de señales de entrenamiento, y se dispone de 8 canales para cada una de las señales, ya que fueron grabadas utilizando un *array* de 8 micrófonos. Los que vamos a utilizar son los de desarrollo, para obtener resultados con las métricas que veremos más adelante, y el de entrenamiento, que lo utilizaremos para entrenar la red neuronal en el último de los métodos.

Señales de desarrollo

El *set* de desarrollo está compuesto tanto por señales reales como simuladas, aunque las señales reales no las vamos a utilizar ya que para el uso de la mayoría de las métricas necesitamos tanto las señales limpias como las reverberantes. Las señales simuladas se obtienen a partir de la convolución de las señales del WSJCAM0 con las respuestas impulsionales RIRs de distintas habitaciones, para posteriormente añadir una cantidad de ruido que da lugar a una relación señal-ruido de 20 dB. Para que las señales simuladas sean coherentes, la obtención de las RIRs se ha de hacer con una fuente en la misma posición del hablante y en la misma habitación.

Las señales de este *set* vienen organizadas según distintas condiciones de grabación. La primera de ellas es la distancia: Por un lado, están las señales etiquetadas como *near*, en las cuales la distancia entre los micrófonos y el hablante es de 50 cm. Por otro lado, en las señales etiquetadas como *far*, esta distancia es de 200 cm.

La otra condición es el tipo de habitación, pues disponemos de 3; pequeña, grande y mediana, con tiempos de reverberación (RT60) distintos. Este valor nos indica el tiempo que ha de transcurrir para que el nivel de presión sonora decaiga 60 dB [14]. Las señales de cada tipo de habitación están etiquetadas como *Room1*, *Room2*, y *Room3*, y cada una de ellas tiene un RT60 de 300 ms, 600 ms y 700 ms respectivamente. En la siguiente tabla podemos ver las propiedades de las señales según las condiciones anteriores, donde podemos ver también el parámetro D_{50}, el cual representa el porcentaje de la energía que representa la señal directa y de las reflexiones que lleguen antes de 50 ms con respecto a la total [13].

	RT60	**D_{50}** ***(near)***	**D_{50}** ***(far)***
Room 1	300 ms	99 %	98 %
Room 2	600 ms	95 %	79 %
Room 3	700 ms	97 %	81 %
Tabla 1 : Características de las salas			

Señales de entrenamiento

El *set* de señales de entrenamiento está compuesto por señales con reverberación simulada con las mismas condiciones de ruido que las anteriores y por sus correspondientes señales limpias. La diferencia es que para estas señales se dispone de RIRs de 24 habitaciones diferentes, cuyos tiempos RT60 varían entre 200 ms y 800 ms. Con esto se logra una mayor variedad de escenarios para que nuestra red aprenda.

3.3. Métricas de evaluación

Entre las métricas que se utilizan para evaluar los algoritmos de desreverberación podemos distinguir entre subjetivas y objetivas. Las medidas subjetivas se basan en la percepción humana, por lo que tienen los inconvenientes de necesitar oyentes para la evaluación, con el correspondiente gasto de tiempo. Por otro lado, es difícil garantizar una cierta coherencia entre todas las evaluaciones. En cambio, las métricas objetivas son rápidas de llevar a cabo y la correlación entre las medidas es mucho más alta[15], por lo que vamos a utilizar únicamente este tipo de métricas, las cuales presentamos a continuación.

En primer lugar, calcularemos los valores de las métricas comparando las señales limpias con las señales reverberantes, para después comprobar cómo cambian los valores al comparar las señales limpias con las realzadas.

Todos los códigos que vamos a utilizar para calcular las métricas han sido obtenidos en la página web del Reverb Challenge y vienen implementados para MATLAB.

Cepstral Distance (CD)

El *Cepstrum* de una señal se define como la transformada de Fourier inversa del logaritmo del espectro de esa señal. Pese a lo arbitrario que podría parecer, esta señal nos proporciona información que es útil para el análisis de señales reverberantes, como por ejemplo determinar el tiempo de llegada del frente principal de una señal de voz, así como el de las sucesivas reflexiones[16].

Por tanto, podemos definir la distancia cepstral como una estimación de la distancia espectral logarítmica entre dos espectros, la cual es una buena métrica de la distorsión que ha sufrido una señal de voz. La fórmula para calcular este valor es la siguiente [17]:

$$CD = \frac{10}{\log 10}\sqrt{2\sum_{i=1}^{p}\{Cx(i) - Cy(i)\}^2} \quad (15)$$

Donde Cx y Cy representan los coeficientes de cepstrum de las señales limpias y reverberantes respectivamente.

Como podemos observar en la fórmula, los valores estarán comprendidos entre 0 y 10 en una escala de decibelios. El objetivo será obtener el valor más pequeño posible, ya que eso implica que ambas representaciones cepstrales tienen un mayor parecido.

Log Likelihood Ratio (LLR)

También conocida como distancia de Itakura, esta métrica es habitualmente usada para medir la calidad de las codificaciones digitales. Está definida como[18]:

$$LLR = log\left[\frac{a_x R_y a_x^T}{a_y R_y a_y^T}\right] \quad (16)$$

a_x representa el vector de coeficientes LPC de la señal limpia, mientras que a_y representa los de la señal reverberante. R_y representa la matriz de autocorrelación de la señal reverberante. Los coeficientes LPC se corresponden como los coeficientes del predictor lineal que obtendríamos de cada una de las señales.

Para el cálculo del valor final de esta métrica, es decir, obtener la media del resultado obtenido con todas las señales, sólo se tendrá en cuenta el 95% de los valores más pequeños y se limitaran los resultados a el rango entre cero y dos.

Al igual que la métrica anterior, esta también está en escala logarítmica y nos interesa obtener el valor más pequeño posible, lo cual implica una menor distorsión de la señal comparándola con la señal limpia.

Frequency-Weighted Segmental SNR

La expresión para obtener esta métrica es la siguiente[15]:

$$fwSNRseg = \frac{10}{M}\sum_{j=1}^{M-1}\frac{\sum_{j=1}^{K} W(j,m)\log_{10}\dfrac{|X(j,m)|^2}{\left(|X(j,m)| - \left|\hat{X}(j,m)\right|\right)^2}}{\sum_{j=1}^{K} W(j,m)} \qquad (17)$$

K representa el número de bandas de frecuencia que estamos considerando. Se consideran 25 bandas, ya que se estima que ese es el número de bandas críticas del oído. El oído humano, dentro de cada una de estas bandas frecuenciales, no puede distinguir un tono de otro, sino que realiza una integración de ellos. En cambio, sí somos capaces de distinguir dos tonos que están en diferentes bandas críticas.

Por su parte, M representa el número de muestras, $W(j,m)$ la función de ponderación para la banda de frecuencias j y la muestra m. $|X(j,m)|$ representa el espectro de la señal limpia enventanado con una ventana gaussiana, mientras que $\left|\hat{X}(j,m)\right|$ representa el espectro de la señal realzada enventanado de la misma forma.

La función de ponderación $W(j,m)$ está compuesta por el espectro de la señal limpia ponderado y elevado a una potencia, tal que:

$$W(j,m) = |X(j,m)|^{\gamma} \qquad (18)$$

El valor de γ que se utiliza en el código que utilizaremos es de 0.2, que es el que da lugar a la máxima correlación según se ha comprobado en [15].

Speech-to-Reververation Modulation Energy Ratio (SRMR)

A diferencia de las métricas anteriores, el SRMR [19] no necesita la señal limpia como referencia para calcularse. Esta medida se obtiene mediante el análisis espectral de las envolventes de modulación de señal de voz, con los pasos que podemos ver en la siguiente figura.

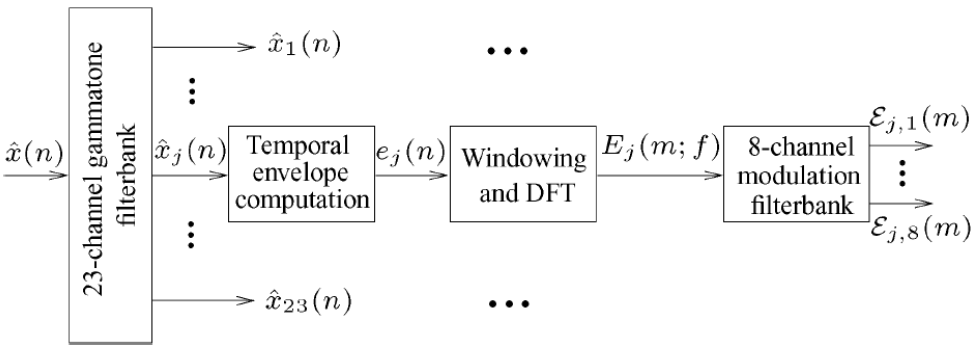

Figura 3: Esquema de cálculo del SRMR. Fuente: [19]

El primer paso de este proceso es el filtrado de la señal de voz $\hat{x}(n)$ mediante un banco de 23 filtros. Como explicamos en la métrica anterior, esto viene de la existencia de las bandas críticas en el oído, aunque en este caso se han considerado 23 en lugar de 25.

Lo siguiente es calcular la envolvente temporal $e_j(n)$ para cada banda de frecuencia j, haciendo uso de la transformada de Hilbert de la siguiente manera:

$$e_j(n) = \sqrt{\hat{x}(n)^2 + H\{\hat{x}(n)\}^2} \tag{19}$$

En el siguiente paso, se enventana cada envolvente con una ventana de Hamming de 256 ms y desplazamientos de 32ms en m fragmentos y se calcula la energía espectral de modulación de cada uno:

$$E_j(m; f) = |F\{ej(m; n)\}|^2 \tag{20}$$

Finalmente, agrupamos la función anterior en 8 partes por el índice de frecuencia, y en cada grupo promediamos para todos los fragmentos m. Así obtenemos lo que se denomina espectrograma de modulación $\bar{\varepsilon}_{j,k}$, representando j las 23 bandas frecuenciales del primer filtro, y k los 8 grupos que hicimos con los índices de frecuencia f. Si promediamos para las 23 bandas de frecuencia, obtenemos un valor $\bar{\varepsilon}_k$ que es el que finalmente utilizaremos para calcular el SRMR:

$$SRMR = \frac{\sum_{k=1}^{4} \bar{\varepsilon}_k}{\sum_{k=5}^{K} \bar{\varepsilon}_k} \tag{21}$$

El significado de la fórmula anterior es que los espectrogramas de modulación de una señal limpia contienen mayor cantidad de energía en las frecuencias bajas, mientras que en una señal reverberante la energía esta más distribuida a lo largo de todas las frecuencias. El valor de K puede variar según cómo sea la señal que vamos a evaluar, teniendo en cuenta que el valor máximo es 8.

4. Resultados

En este capítulo procederemos a evaluar con las métricas los tres algoritmos de que disponemos. El primero de los métodos es NDLP, el cual explicamos en el capítulo 2. Existe una implementación encriptada de MATLAB de este método llamada wpe.p, la cual está implementada en el dominio tiempo frecuencia al hacer uso de la STFT.

El segundo de los algoritmos que vamos a evaluar se denomina NARA-WPE y lo podemos encontrar en [11]. Su principio teórico está basado en el método anterior, que como comentamos, también es conocido como WPE. Por tanto, en este caso no vamos a entrar en una explicación teórica. Sin embargo, hay una diferencia con el método anterior, la cual consiste en que es posible estimar la varianza eligiendo el número de muestras que queremos tener en cuenta y realizando un suavizado. Con esta variación se pretende eliminar el proceso de iteraciones y estimar la varianza directamente de la señal. Por último, hay que mencionar que la implementación que hemos revisado es la que aparece en el punto 4.1 de [11].

Finalmente, el último método es el denominado DNN-WPE, que también explicamos en el capítulo 2 y que hace uso de una red neuronal para estimar la varianza de la señal que queremos realzar.

El primero de los experimentos que vamos a realizar consistirá en comparar los resultados de los tres algoritmos utilizando las métricas descritas en el capítulo anterior, con el fin de evaluar el nivel de los resultados ofrecidos por cada uno y ver si hay alguno que destaca sobre el resto.

Para ello procesaremos todas las señales de desarrollo de la base de datos teniendo en cuenta 1, 2, 4 y 8 canales. Además, obtendremos los resultados considerando todas las señales, pero también los resultados para cada uno de los seis escenarios que disponemos (3 salas diferentes, fuente cerca y lejos del micrófono).

En los experimentos que realizaremos sólo vamos a considerar 3 parámetros que se pueden modificar en todos los métodos: número de coeficientes del filtro, retardo de predicción, y número máximo de iteraciones. Para este primer experimento, los valores de los parámetros serán 6, 3 y 3 respectivamente. El número de coeficientes lo cambiamos a ese valor para que los resultados se obtengan más rápido y para que haya igualdad de condiciones, ya que cada método por defecto tenía un valor distinto. En cuanto al retardo de predicción y el número de iteraciones, ambos estaban por defecto con el valor que vamos a utilizar. No obstante, hay que recordar que en DNN-WPE no hay iteraciones, sino que se estima el espectro directamente con la red.

4.1. Experimento 1: Comparación de los 3 métodos

En este experimento analizaremos los resultados de cada método para cada uno de los 6 escenarios que disponemos:

	Habitación	**Distancia al micrófono**
Escenario 1	*Room1*	*Near*
Escenario 2	*Room1*	Far
Escenario 3	*Room2*	Near
Escenario 4	*Room2*	Far
Escenario 5	*Room3*	Near
Escenario 6	*Room3*	Far

Tabla 2. Escenarios disponibles

Para analizar visualmente mejor los resultados, para cada métrica utilizaremos la misma escala en todos los escenarios, salvo para los resultados globales, y además añadiremos el resultado de la señal original, es decir, la señal reverberante.

Análisis de los resultados para escenarios 1 y 2

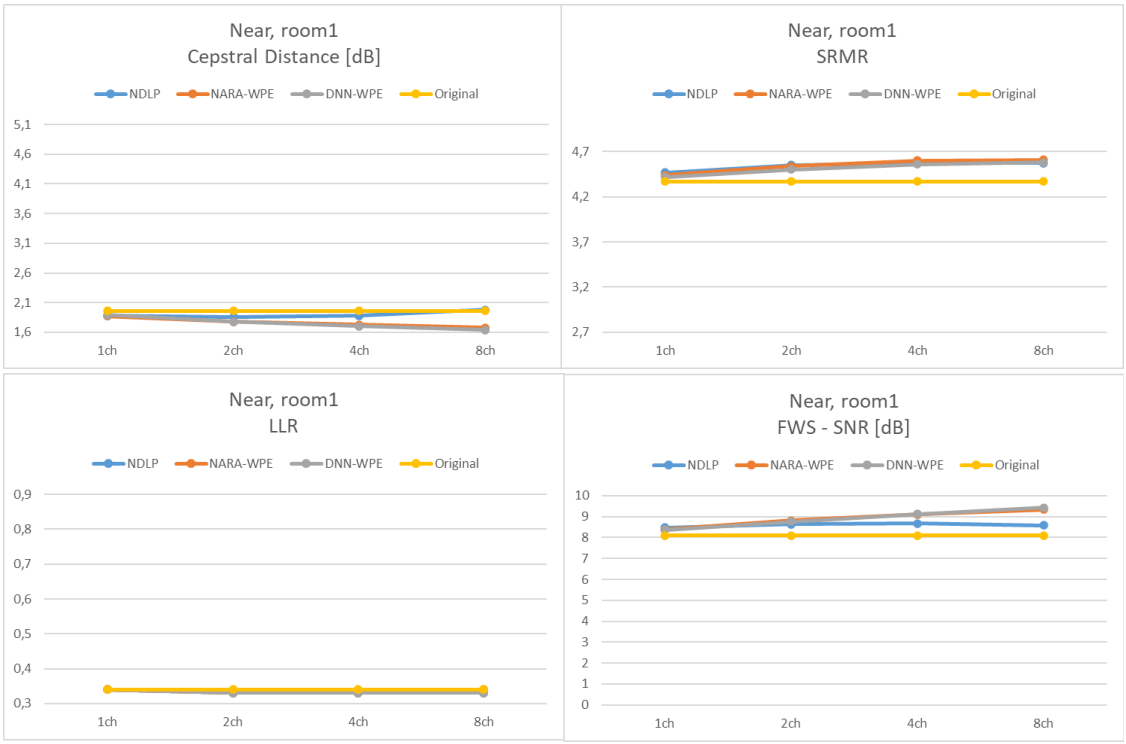

Figura 4: Resultados escenario 1

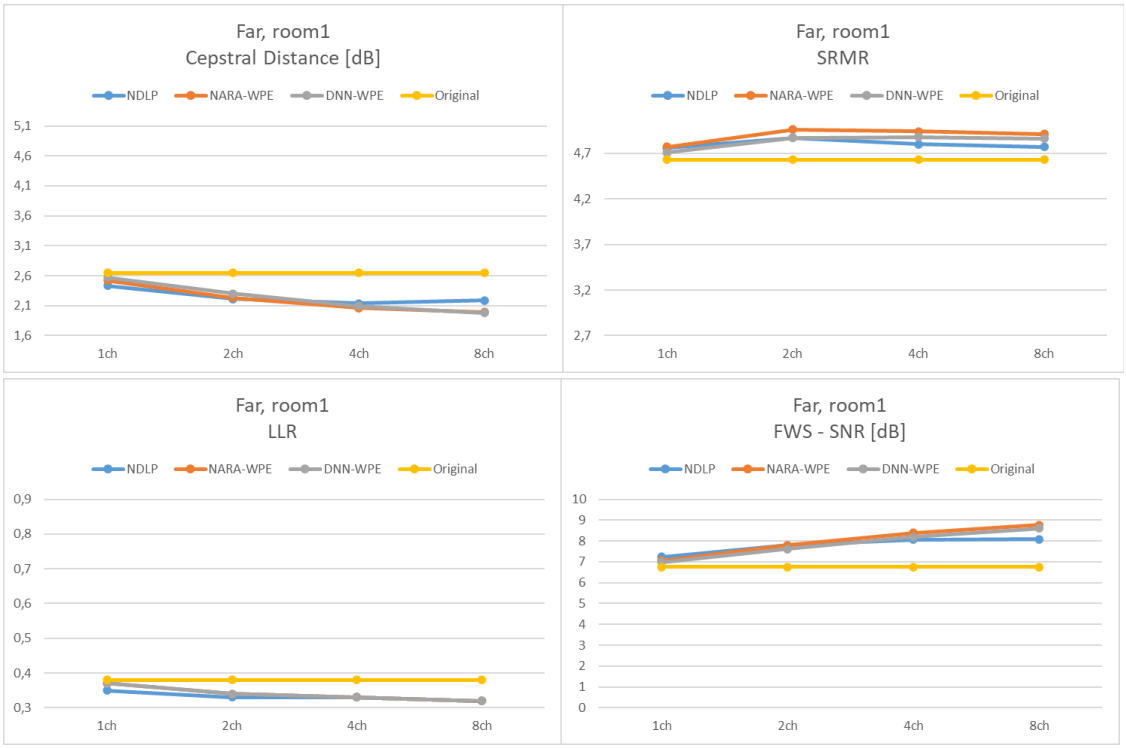

Figura 5: Resultados escenario 2

Si observamos los resultados para la habitación 1 (habitación pequeña) vemos que se logra mejorar los resultados ligeramente tanto para el caso con el micrófono *near* como *far*. Sin embargo, es destacable que en el caso *far*, se parte de un resultado peor y la mejora obtenida es algo mayor, aunque quedándonos por debajo del caso *near*.

Además, es destacable que el LLR para el caso *near* no tiene apenas mejora, mientras que en el caso *far* sí que hay mejora, pero cuando se utilizan muchos canales tiende al mismo valor, por lo que parece haber un límite algo superior al 0,3. También me parece destacable que mientras que para el caso *near*, el resultado de todas las métricas es mejor que en el caso *far*, salvo el SRMR, que es mejor en el caso *far*. Esto resulta algo desconcertante y podría hacernos desconfiar de la validez de la métrica, aunque de todas formas sí que mejora al aumentar el número de canales.

Por último, es destacable en estos dos escenarios que los algoritmos ofrecen resultados similares para cualquier número de canales, salvo NDLP, que parece que pierde efectividad al pasar de 4 a 8 canales salvo en el LLR. Por su parte, DNN-WPE obtiene prácticamente los mismos resultados que NARA-WPE, lo cual es bastante impactante ya que éste último tiene que estimar los coeficientes del filtro 3 veces.

De todo lo dicho anteriormente, podemos obtener como conclusiones que todos los algoritmos consiguen mejorar en las condiciones de poca reverberación de una habitación pequeña, aunque en mayor medida si la distancia al micrófono es mayor. Por otro lado, parece que NDLP funciona peor con estas condiciones que el resto de los métodos.

Análisis de los resultados para escenarios 3 y 4

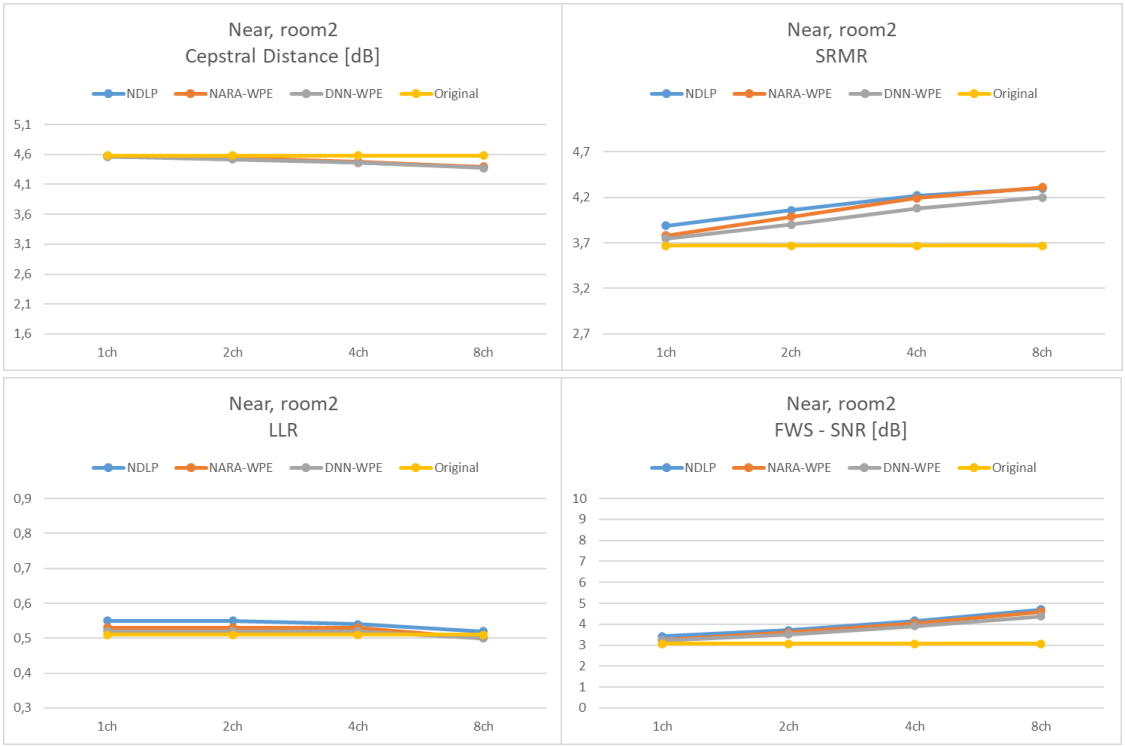

Figura 6: Resultados escenario 3

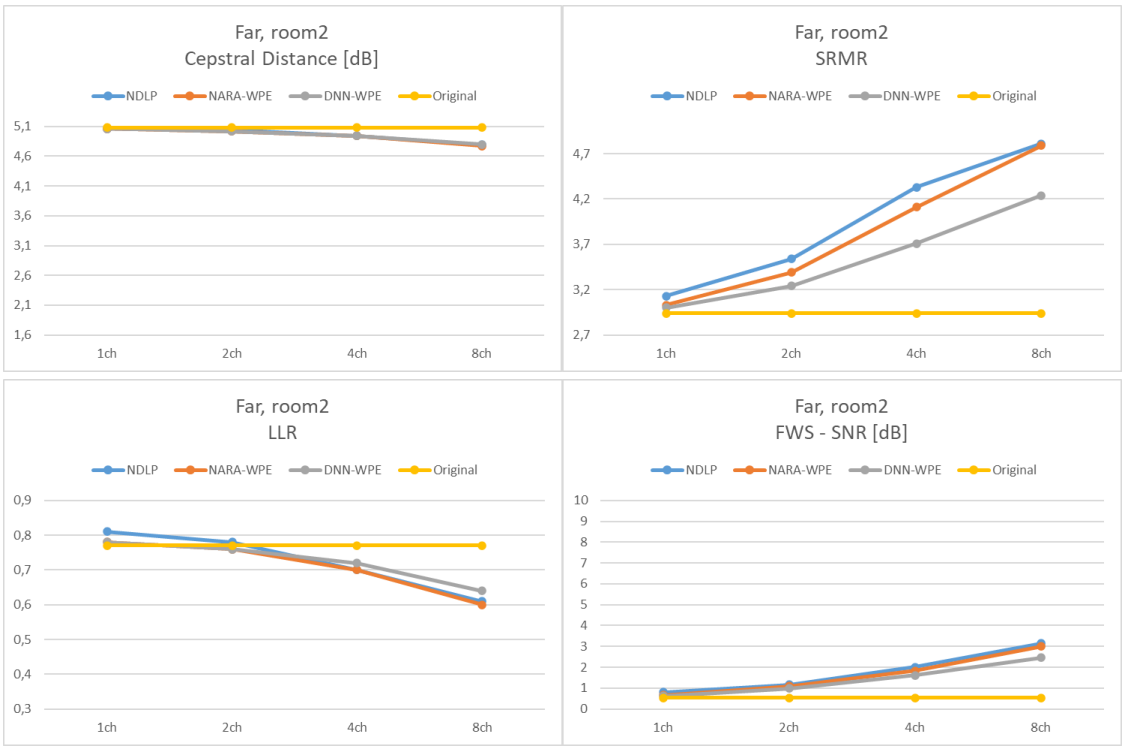

Figura 7: Resultados escenario 4

En cuanto a la habitación 2 (habitación mediana), me parece destacable que se consigue una mejora pequeña comparada con los dos casos anteriores, especialmente en el caso *near*. De nuevo, se repite el hecho de que en el caso *far* la mejora es mayor, especialmente en el LLR y el SRMR.

A diferencia de los escenarios 1 y 2, aquí el método que da peores resultados es DNN-WPE, aunque está bastante cerca. Parece que NDLP funciona mejor cuando la cantidad de reverberación es superior y consigue unos resultados ligeramente superiores a NARA-WPE.

También en estos escenarios se repite que el SRMR da lugar a un mejor resultado en el caso *far*, aunque el resto de las métricas tengan mejores valores.

Análisis de los resultados para escenarios 5 y 6

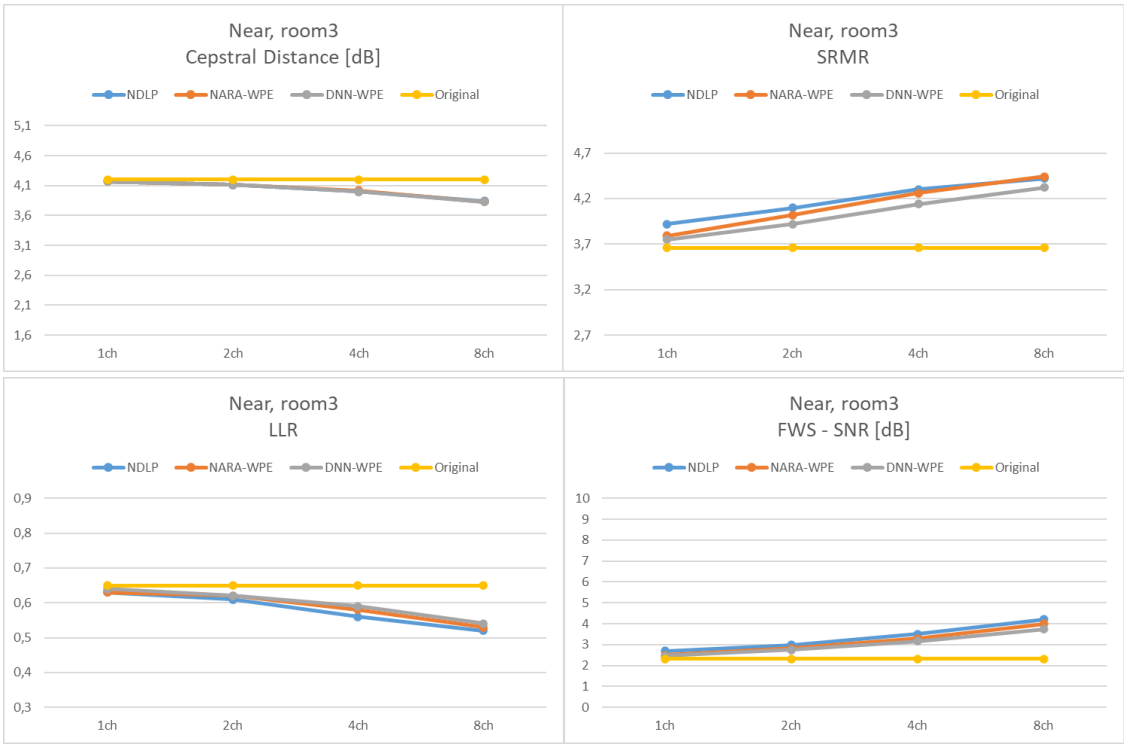

Figura 8: Resultados escenario 5

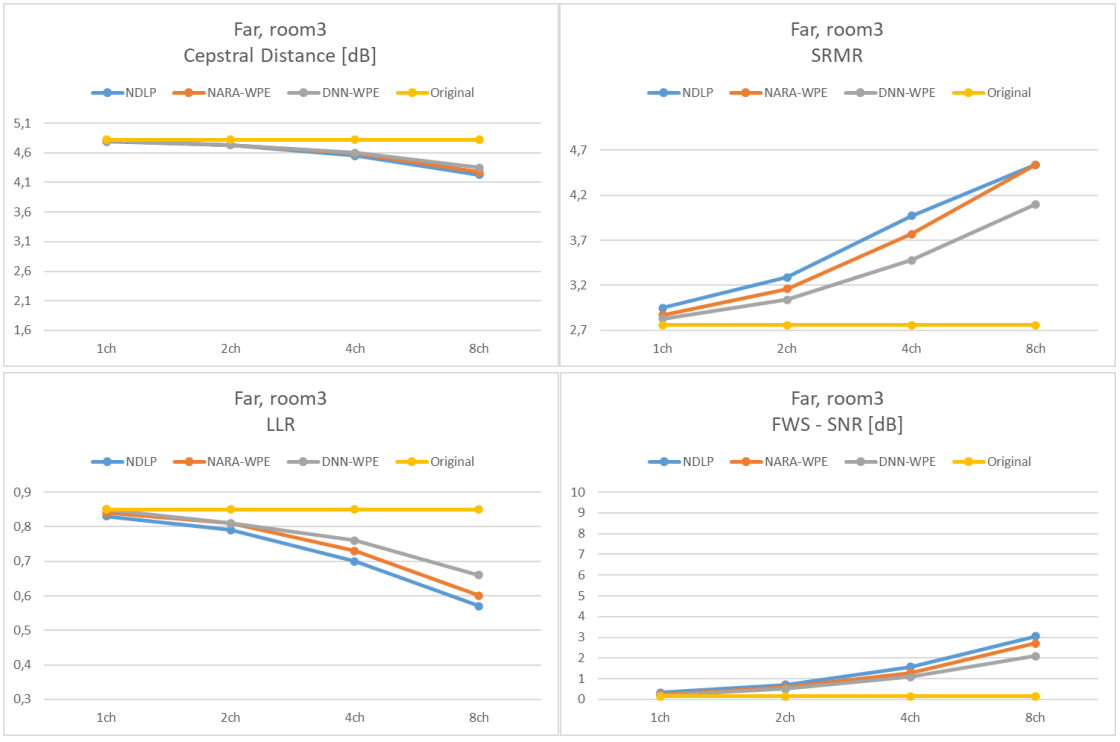

Figura 9: Resultados escenario 6

Finalmente, de los resultados de la habitación 3 (habitación grande), podemos concluir que es el escenario en el que se produce una mayor mejora de las métricas. Además, vuelve a ocurrir que la mejora es superior en el caso *far*. Sin embargo, en estos dos últimos escenarios, a diferencia de en los cuatro anteriores, el SRMR no es mejor en el caso *far*, siguiendo la misma tendencia que el resto de las métricas.

En cuanto a la comparación entre los métodos, NDLP destaca sobre NARA-WPE más que en el caso de la habitación 2, especialmente cuando la voz está alejada del micrófono. De nuevo DNN-WPE obtiene resultados un poco peores, aunque en el caso *near* son bastante similares.

Conclusiones del resultado por escenarios

Tras haber analizado lo más destacable de los resultados para los seis escenarios que disponemos, podemos obtener las siguientes conclusiones:

Para la habitación 1, NDLP consigue peores resultados que NARA-WPE y DNN-WPE, que están ambos bastante a la par. En todos los casos, se consigue una mejora pequeña de las métricas.

En el caso de la habitación 2, NDLP y NARA-WPE consiguen resultados un poco mejores que DNN-WPE, mientras que la mejora es más pequeña que en el caso de la habitación 1.

En los resultados de la habitación 3 es donde se logra la mayor mejora. NDLP es el que consigue los mejores resultados, seguido de cerca por NARA-WPE, y DNN-WPE algo más atrás.

Las métricas CD, LLR, y FWseg-SNR son consistentes con sus resultados en todos los escenarios, mientras que el SRMR no parece seguir su tendencia, ya que en los primeros cuatro escenarios da resultados mejores cuando el resto obtiene mejores y viceversa. Por tanto, podríamos considerar que este valor es válido para comparar resultados dentro de un mismo escenario, pero no para compararlos cuando son distintos.

Resultado global

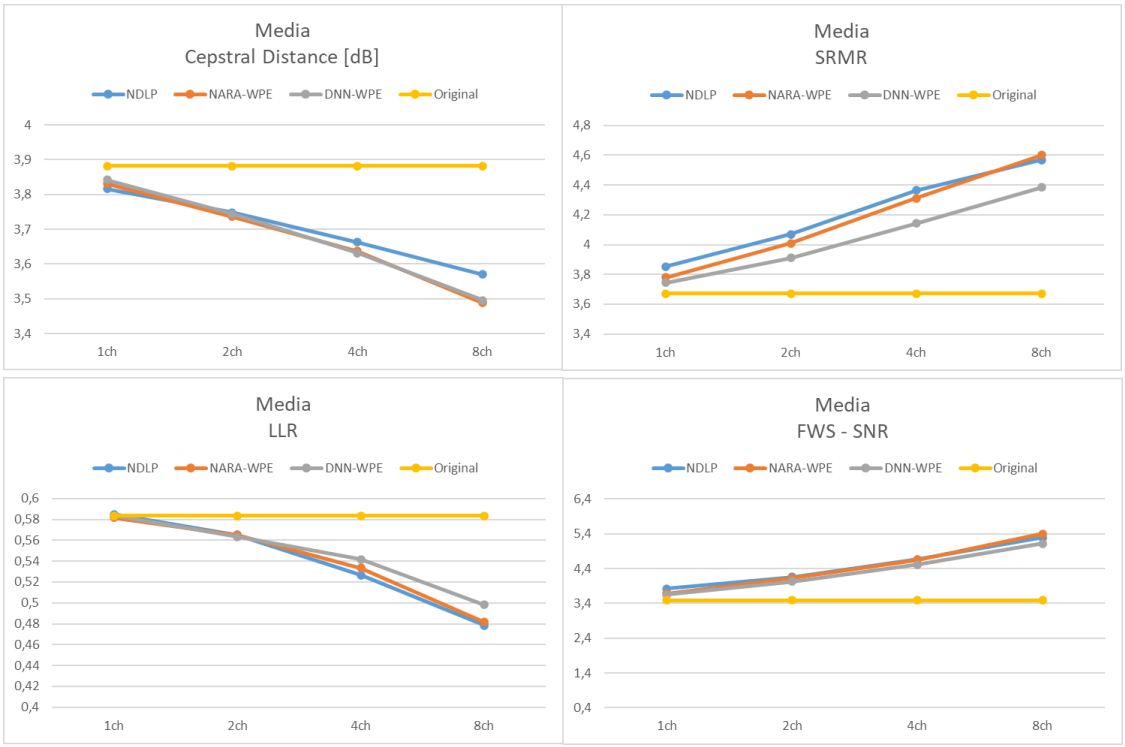

Figura 10: Resultados globales

Si consideramos los resultados para el conjunto de todo el *set* de señales, llegamos a la conclusión de que NDLP y NARA-WPE logran unos resultados muy parecidos, lo cual tiene sentido puesto que son distintas implementaciones del mismo algoritmo. No obstante, me parece destacable que la CD de NDLP es la peor de todas, aunque en el resto este a la par con NARA-WPE.

Por otro lado, DNN-WPE logra unos resultados un poco peores en general, aunque la CD la resultante es prácticamente igual que para NARA-WPE, lo cual es bastante destacable.

En cuanto a la evolución de las métricas según el número de canales, podemos observar que si solamente utilizamos uno la mejora es bastante pequeña. Sin embargo, hay un gran salto al pasar a dos canales, el cual no es un escenario excesivamente optimista ya que los micrófonos estéreo no son raros. Si aumentamos el número de canales a cuatro u ocho vemos cómo los resultados mejoran aún más, siendo el salto de calidad cada vez superior al pasar a un número superior de micrófonos. No obstante, hay que mencionar que escenarios de 4 y 8 micrófonos son bastante menos habituales. También es destacable que, en cada métrica, se mantiene la posición de cada uno de los métodos en cuanto a prestaciones para cualquier número de canales, salvo la CD para NDLP.

Por todo ello, los métodos más reseñables me parecen NARA-WPE y DNN-WPE, ya que uno obtiene los mejores resultados en todo, pero el otro obtiene la CD y la SNR prácticamente igual sin llevar a cabo 3 iteraciones. Aunque los resultados de NDLP también son buenos, el hecho de que es mucho más lento que NARA-WPE le hace quedarse por detrás. Si bien, el hecho de que estemos ejecutando los algoritmos en un servidor compartido hace que un análisis de los tiempos de ejecución estricto sea inviable, sí que puedo decir que NDLP suele tardar alrededor

de 20 segundos por señal si consideramos 8 canales, mientras que NARA-WPE y DNN-WPE suelen tardar 5 segundos aproximadamente.

Una forma más visual de ver el efecto de los algoritmos de desreverberación es el espectrograma. Mientras que en una señal con reverberación vemos como la energía de cada componente frecuencial se mantiene más en el tiempo, en una señal que carece de ella lo que deberíamos observar es un pulso a lo largo del eje de frecuencias y de una duración pequeña.

Si observamos la siguiente serie de figuras, que corresponden al espectrograma de una señal del escenario 4 (habitación mediana y fuente lejos del micrófono, que es el escenario más reverberante), vemos precisamente cómo en la señal sin realzar cada pulso se extiende a lo largo del tiempo. Conforme va aumentando el número de canales considerados, vemos que los espectrogramas tienden a tener pulsos correspondientes a cada frente de ondas directas, siendo especialmente notable en los casos de cuatro y ocho canales.

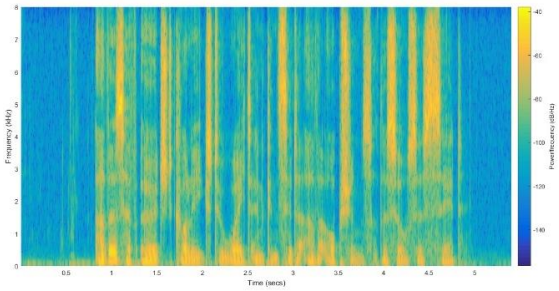

Figura 11: Espectrograma de la señal limpia

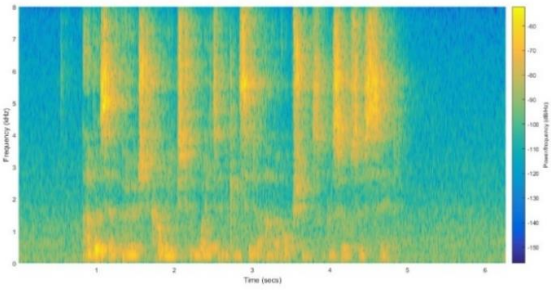

Figura 12: Espectrograma de la señal reverberante

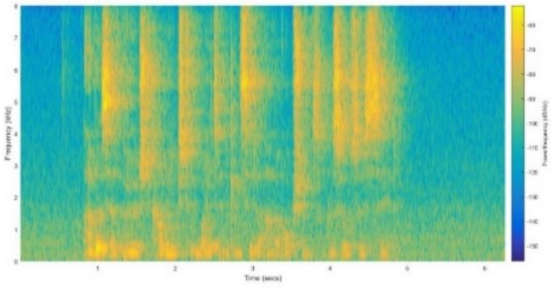

Figura 13: Espectrograma de la señal realzada (1ch)

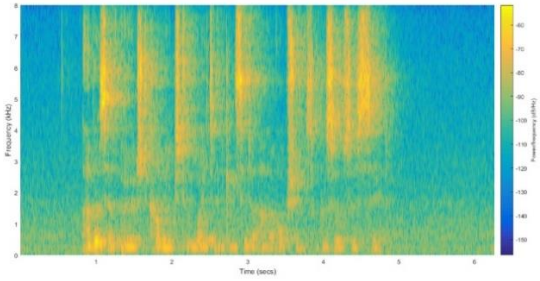

Figura 14: Espectrograma de la señal realzada (2ch)

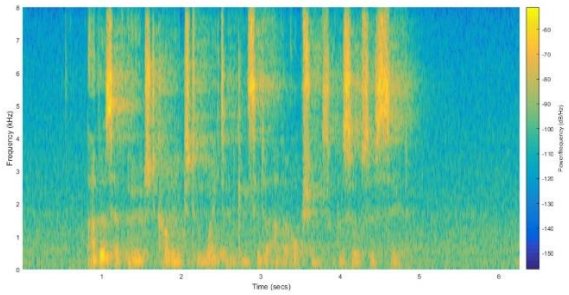

Figura 15: Espectrograma de la señal realzada (4ch)

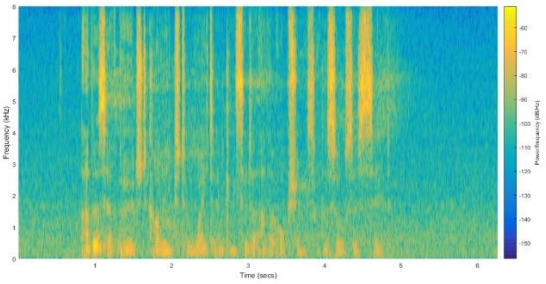

Figura 16: Espectrograma de la señal realzada (8ch)

4.2. Experimento 2: Comparación de NARA-WPE con suavizado vs DNN-WPE

Como mencionamos en el inicio de este capítulo, en NARA-WPE es posible estimar la varianza/espectro de potencia teniendo en cuenta un número mayor de muestras y realizando un suavizado. Según [11], con esta variación se logran buenos resultados utilizando la varianza de la señal observada. En otras palabras, ya no tenemos que realizar el proceso de iteraciones para estimar la varianza de la señal deseada, y de esta forma se obtienen los coeficientes del filtro en un solo paso.

Teniendo en cuenta lo anterior, en este experimento vamos a comprobar cuál es la diferencia de resultados entre NARA-WPE con el suavizado y DNN-WPE, ya que ambos obtienen la varianza directamente de la señal observada. De este modo podemos evaluar hasta qué punto sale a cuenta emplear una red neuronal con estos métodos.

Como ya no estamos evaluando los algoritmos por escenarios, únicamente mostraremos los resultados de cada métrica considerando todo el *set* de señales de desarrollo, teniendo en cuenta el procesado con 1, 2, 4 y 8 canales. Por otro lado, vamos a prescindir del SRMR, ya que como vimos en el experimento 1, sus resultados no son muy consistentes.

NARA-WPE	**CD [dB]**	**LLR**	**FWS-SNR [dB]**
1ch	3.81	0.58	3.77
2ch	3.74	0.57	4.08
4ch	3.65	0.55	4.48
8ch	3.52	0.51	5.02
DNN-WPE			
1ch	3.84	0.58	3.65
2ch	3.74	0.56	4.02
4ch	3.63	0.54	4.51
8ch	3.49	0.5	5.12

Tabla 3: Resultados experimento 2

En general, los resultados muestran que ambos métodos tienen prestaciones similares. Si consideramos un micrófono, NARA-WPE es el que obtiene los mejores resultados. Cuando aumentamos el número de micrófonos a dos o cuatro los resultados son bastante similares. No es hasta que consideramos los ocho canales disponibles que DNN-WPE tiene resultados superiores.

Sin embargo, considero que la mejora que se obtiene al utilizar la red neuronal no es lo suficientemente grande (desde el punto de vista de nuestras métricas) como para que merezca la

pena su uso. Aunque es cierto que sólo hay que entrenar la red una vez, este es un proceso que me ha llevado más de un día, mientras que en NARA-WPE el hecho de aplicar el suavizado no parece ralentizar los tiempos de ejecución comparado con el NARA-WPE estándar.

4.3. Experimento 3: Evaluación de la influencia del retardo de predicción

En este último experimento vamos a evaluar cómo influye el valor del retardo o *delay* de predicción en los resultados de las métricas. Para ello vamos a emplear como método de desreverberación NARA-WPE considerando 8 canales. Los parámetros serán los mismos que en el Experimento 1; salvo el *delay*, puesto que iremos variando su valor desde uno hasta seis para ver cómo evolucionan los resultados.

El retardo de predicción lo podemos ver en la ecuación (2, y con él distinguimos la reverberación temprana de la tardía para posteriormente eliminar la tardía. Ya que disponemos de señales correspondientes a salas con distintos niveles de reverberación, podemos ver cómo influye el valor del *delay* en cada una de estas. Al igual que en el experimento anterior, no vamos a considerar el SRMR.

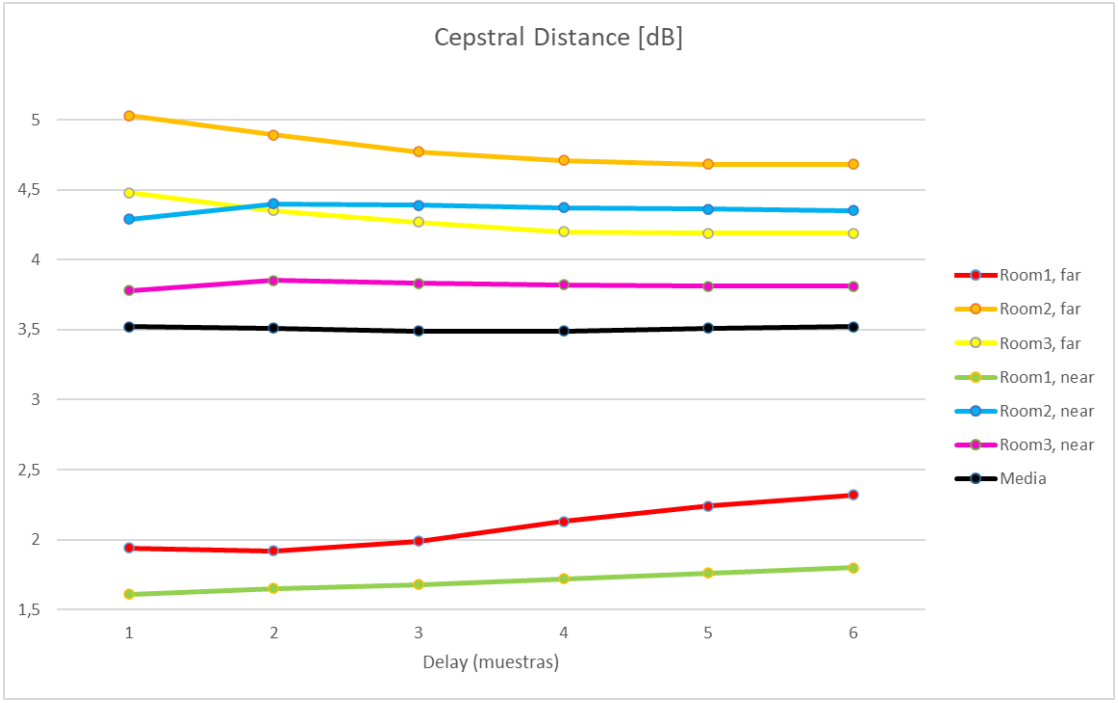

Figura 17: Resultados Cepstral Distance Experimento 3

Como se puede observar en la Figura 17, los resultados con la CD muestran una tendencia bastante clara. Por un lado, en todos los escenarios en los que la fuente de sonido está cerca del micrófono, se consigue el mejor resultado (recordamos que en esta métrica es el más bajo) con un valor de retardo de 1. Por otro lado, en el escenario 2 (*room 1, far*), se consigue el mejor resultado con un valor de 2. Esto lo podemos interpretar como que cuando el D_{50} presenta un muy cercano al 100% se consiguen mejores resultados con valores de retardo más pequeños. Aun así, me parece reseñable que en los escenarios 1 y 2 (*room 1*), al ir aumentando el retardo,

después del mejor valor cada vez se obtienen peores, mientras que en los escenarios 3 y 5, los valores empeoran al principio para posteriormente volver a empeorar.

En cuanto a los escenarios 4 y 6, donde el valor D_{50} es considerablemente menor, los mejores resultados se obtienen para ambos casos con valores de retardo de 5 y 6 (mismo resultado con ambos retardos). En cuanto a la evolución, hay una clara tendencia de mejorar los resultados al ir aumentando el retardo, aunque cada vez en menor cantidad, hasta llegar a un valor de retardo de 5, donde la mejora se detiene.

Finalmente, si observamos el resultado del valor medio, éste se mantiene aproximadamente constante en torno a 3,5 dB. Al tener un *set* de señales muy amplio, las mejoras de unos escenarios se compensan con los empeoramientos en otros.

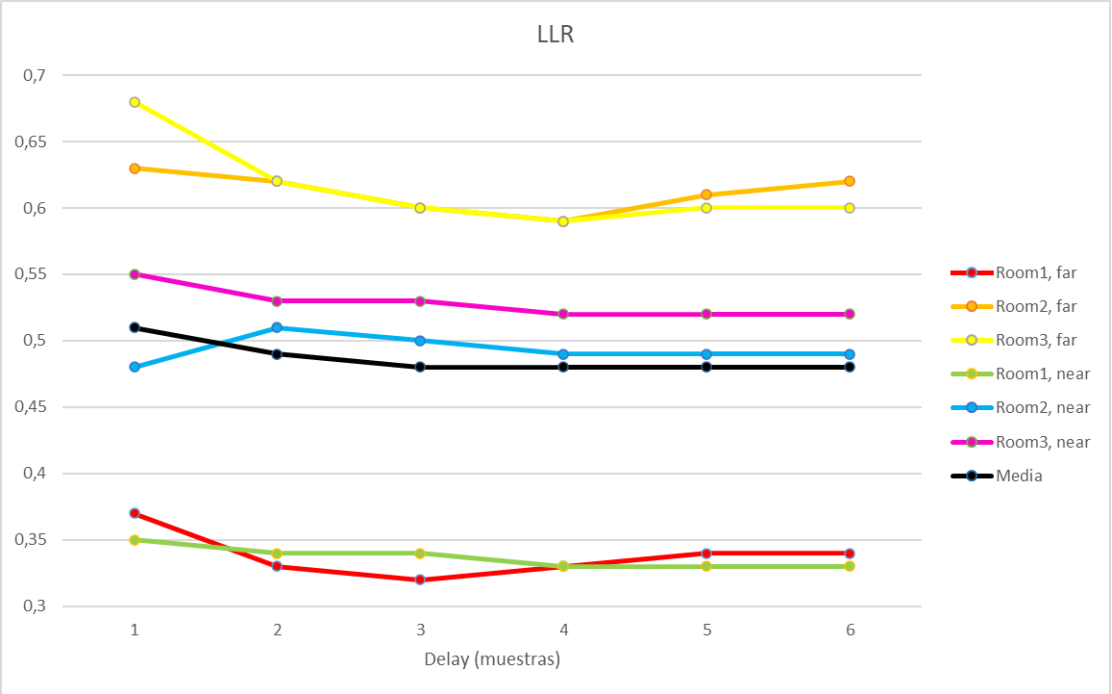

Figura 18: Resultados LLR Experimento 3

Los resultados del LLR (Figura 18) no nos llevan a la misma conclusión que los de la CD. La diferencia es que ahora, todos los casos *far* consiguen el mejor resultado para un valor de *delay* de 3 o 4, mientras que si aumentamos más el retardo éstos se vuelven peores.

En cambio, los casos *near* alcanzan el mejor resultado a partir de un valor de retardo de 4, para después mantenerse constante si lo seguimos aumentando. Sólo hay una excepción, que es para la habitación 2, donde se logra el mejor resultado con un retardo de 1.

En cuanto al resultado de la media, se alcanza el tope a partir de un retardo de 3, y posteriormente se mantiene constante.

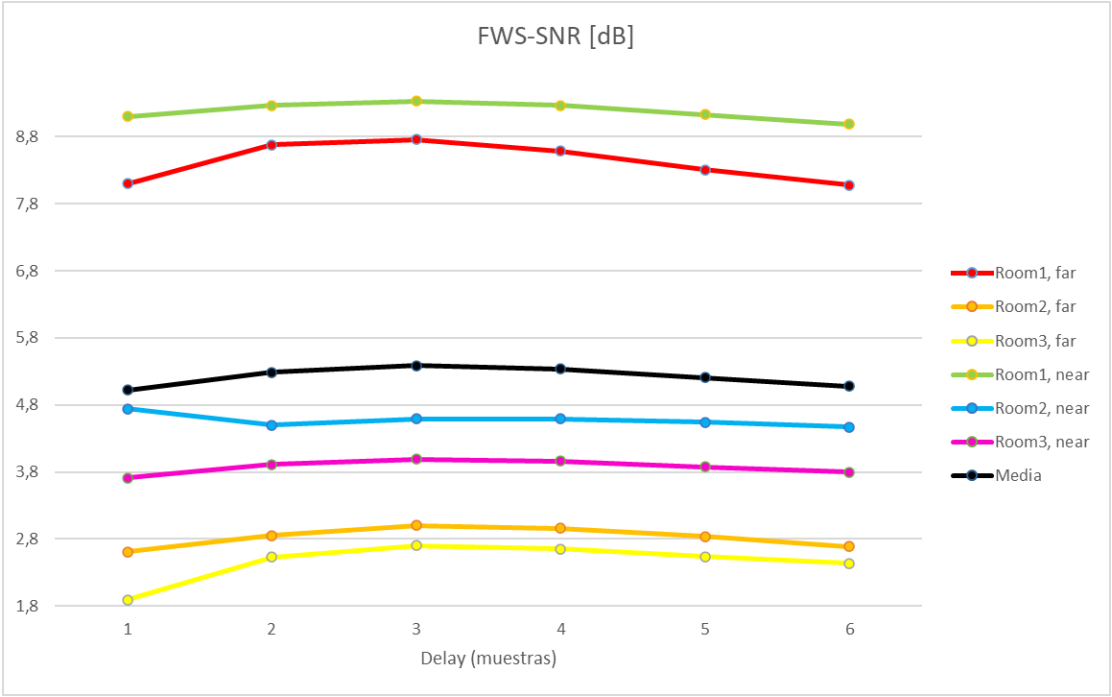

Figura 19: Resultados FWS-SNR Experimento 3

Para terminar, en la Figura 19 tenemos los resultados de la SNR ponderada, donde todos los escenarios salvo 1 tienen el mejor resultado cuando el retardo tiene valor 3. La excepción es de nuevo el caso *room2, near*, que vuelve a obtener el mejor resultado con un retardo de 1.

Teniendo en cuenta que, el valor medio obtenido en la CD se mantiene constante, que para el LLR se mantiene constante a partir de un retardo de 3, y que en la SNR el mejor resultado es también para 3, en este experimento podemos concluir que el mejor valor de retardo es 3, que por otro lado es el que venía por defecto. También hemos comprobado que, desde el punto de vista de la CD, que es la métrica que está más relacionada con la reverberación, se puede ajustar el valor del *delay* a un valor mayor cuando las condiciones de reverberación sean superiores. El resto de las métricas no nos indican esto, pero éstas están también relacionadas con la calidad general de la señal, lo cual podría explicar la diferencia de resultados.

5. Conclusiones

Como mencionamos, el objetivo de este trabajo era evaluar algunos de los algoritmos de desreverberación que se han desarrollado en los últimos años. Para ello comenzamos evaluando el algoritmo NDLP o WPE en MATLAB, el cual podríamos considerar como base de este trabajo, para posteriormente evaluar implementaciones alternativas del mismo. Con NARA-WPE desponíamos de una implementación similar para *Python*, mientras que con DNN-WPE incorporábamos nuevas técnicas como el uso de redes neuronales.

De todo el trabajo realizado, se desprenden las conclusiones que se describen a continuación. En primer lugar, en el experimento 1 vimos que los métodos que obtenían los mejores resultados eran NARA-WPE y NDLP, aunque los elevados tiempos de ejecución de este último nos llevaron a su descarte. Tras esto nos planteamos si era posible que NARA-WPE pudiera lograr también buenos resultados si eliminábamos el proceso de tres iteraciones. En el experimento 2 vimos que la respuesta era afirmativa: con el suavizado del espectro, NARA-WPE lograba unos resultados ligeramente peores que los de DNN-WPE. Por tanto, concluimos que el uso de la red neuronal para estimar el espectro de la señal no da lugar a resultados tan buenos como esperábamos. Una vez concluido que NARA-WPE era el mejor algoritmo, tanto para el procesado de forma iterativa como de forma directa, en el experimento 3 tratamos de ver la influencia del *delay* de predicción en los resultados. Si bien concluimos que valores de *delay* pequeños funcionan mejor para situaciones con poca reverberación, y que valores grandes lo hacen para entornos más reverberantes, no llegamos a una conclusión de qué valor en concreto elegir en cada caso.

Por otro lado, también llegamos a una serie de conclusiones sobre la calidad general de los algoritmos y las métricas. Los resultados en general son buenos, es decir, si comparamos una escucha de la señal reverberante con una de la misma señal procesada, la mejora es notable, especialmente con auriculares. Se nota claramente que la señal es más inteligible, especialmente si utilizamos 8 canales, donde diría que la mejora es muy buena. Sin embargo, me parece que, si utilizamos 1 o 2 canales, la mejora no es tan notable. Esto supone un problema, puesto que lo más habitual es encontrar micrófonos de uno o dos canales. En cuanto a las métricas, concluimos en el experimento 1 que el SRMR no ofrece resultados consistentes, por lo que posteriormente dejamos de emplearlo.

Finalmente, es importante destacar que todo el procesamiento de señales realizado no ha sido en tiempo real, por lo que resultaría interesante evaluar cómo varían los resultados si se utiliza ese tipo de procesamiento, puesto que es el necesario para las aplicaciones que se podrían beneficiar de estos métodos. Por otro lado, también sería interesante establecer un modelo estadístico más próximo a la realidad de las señales de voz. El modelo gaussiano con varianza no constante vemos que ofrece buenos resultados para lo simple que es, pero quizá con otro modelo se lograrían resultados muy buenos reduciendo el número de canales. No obstante, sería necesario verificar si el resultado es lo suficientemente bueno como para aumentar la complejidad de los métodos tan considerablemente.

Referencias

[1] Naylor P and Gaubitch N, *Speech Dereverberation*. London: Springer London, 2010. doi: 10.1007/978-1-84996-056-4.

[2] H. Krüger *et al.*, "Do We Need Dereverberation for Hand-Held Telephony?," 2010. [Online]. Available: www.ind.rwth-aachen.dewww.infineon.com

[3] M. Jeub, M. Schafer, T. Esch, and P. Vary, "Model-based dereverberation preserving binaural cues," *IEEE Trans Audio Speech Lang Process*, vol. 18, no. 7, pp. 1732–1745, 2010, doi: 10.1109/TASL.2010.2052156.

[4] Takuya Yoshioka, Hirokazu Kameoka, Tomohiro Nakatani, and Hiroshi G. Okuno, *STATISTICAL MODELS FOR SPEECH DEREVERBERATION*. IEEE, 2009.

[5] "The REVERB challenge - Evaluating de-reverberation and ASR techniques in reverberant environments." https://reverb2014.dereverberation.com/index.html (accessed May 07, 2022).

[6] "WPE speech dereverberation." https://www.rd.ntt/cs/team_project/media/signal/wpe/index.html (accessed Aug. 28, 2022).

[7] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B. H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE Trans Audio Speech Lang Process*, vol. 18, no. 7, pp. 1717–1731, 2010, doi: 10.1109/TASL.2010.2052251.

[8] K. Kinoshita, M. Delcroix, H. Kwon, T. Mori, and T. Nakatani, "Neural network-based spectrum estimation for online WPE dereverberation," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017, vol. 2017-August, pp. 384–388. doi: 10.21437/Interspeech.2017-733.

[9] "An A-Z Index of the Linux command line: bash + utilities." https://ss64.com/bash/ (accessed Aug. 29, 2022).

[10] "HTCondor Overview." https://htcondor.org/htcondor/overview/ (accessed Aug. 29, 2022).

[11] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach, "NARA-WPE: A Python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing."

[12] T. ROBINSON, J. FRANSEN, D. PYE, J. FOOTE, and S. RENALS, "WSJCAM0 - A BRITISH ENGLISH SPEECH CORPUS FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION," in *1995 INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING - CONFERENCE PROCEEDINGS, VOLS 1-5*, 1995, pp. 81–84.

[13] K. Kinoshita *et al.*, "A summary of the REVERB challenge: state-of-the-art and remaining challenges in reverberant speech processing research," *EURASIP J Adv Signal Process*, vol. 2016, no. 1, pp. 1–19, Dec. 2016, doi: 10.1186/S13634-016-0306-6.

[14] “MEDICIÓN DE TIEMPO DE REVERBERACIÓN RT60.” https://dewesoft.com/es/aplicaciones/acustica/tiempo-de-reverberacion (accessed Jul. 28, 2022).

[15] Y. Hu and P. C. Loizou, “Evaluation of objective quality measures for speech enhancement,” *IEEE Trans Audio Speech Lang Process*, vol. 16, no. 1, pp. 229–238, Jan. 2008, doi: 10.1109/TASL.2007.911054.

[16] D. G. CHILDERS, D. P. SKINNER, and R. C. KEMERAIT, “CEPSTRUM - GUIDE TO PROCESSING,” *PROCEEDINGS OF THE IEEE*, vol. 65, no. 10, pp. 1428–1443, 1977, doi: 10.1109/PROC.1977.10747.

[17] N. KITAWAKI, H. NAGABUCHI, and K. ITOH, “OBJECTIVE QUALITY EVALUATION FOR LOW-BIT-RATE SPEECH CODING SYSTEMS,” *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 6, no. 2, pp. 242–248, Feb. 1988, doi: 10.1109/49.601.

[18] R. E. CROCHIERE, J. M. TRIBOLET, and L. R. RABINER, “AN INTERPRETATION OF THE LOG LIKELIHOOD RATIO AS A MEASURE OF WAVEFORM CODER PERFORMANCE,” *IEEE Trans Acoust*, vol. 28, no. 3, pp. 318–323, 1980, doi: 10.1109/TASSP.1980.1163417.

[19] T. H. Falk, C. Zheng, and W. Y. Chan, “A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech,” *IEEE Trans Audio Speech Lang Process*, vol. 18, no. 7, pp. 1766–1774, 2010, doi: 10.1109/TASL.2010.2052247.