INSTITUTO TECNOLÓGICO DE MEXICALI

Metodología para la Evaluación del Desempeño de Filtros Digitales de Señales

Presenta

Steven Delgadillo Quezada

Tesis

Para obtener el grado de

Maestro en Ingeniería Electrónica

Director de tesis
Dr. Arnoldo Díaz Ramírez

Co-Director de tesis
Dr. Víctor Hugo Díaz Ramírez

Mexicali, B. C. **Diciembre 2011**

Metodología para la Evaluación del Desempeño de Filtros Digitales de Señales.

Agradecimientos

Quiero expresar mi agradecimiento a mi familia.También a kary por los corajes que hizo a la hora de ayudarme. Además a mi asesor Arnoldo Diaz por apoyo en este trabajo y demás personas he instituciones que participaron en este trabajo gracias.

Resumen

La degradación de las señales digitales de voz son causadas por modificaciones indeseables de las señales durante el proceso de grabación, almacenamiento y/o transmisión. Para mejorar la calidad de las señales capturadas por un sistema digital y que han sido degradadas, se han propuesto una gran cantidad de filtros digitales. Una alternativa para evaluar el desempeño de un filtro, consiste en implementarlo en algún dispositivo y llevar a cabo diversas pruebas. Sin embargo, mucho de los filtros propuestos utilizan una gran cantidad de parámetros, por lo que su evaluación puede ser un proceso tedioso y que consuma demasiado tiempo. En esta tesis se propone una metodología para la evaluación de filtros digitales y la elección de sus parámetros. Esta metodología se basa en las métricas de calidad del error cuadrático promedio y el error absoluto promedio. Además se propone una modificación a un algoritmo localmente adaptativo publicado recientemente y se utiliza la metodología propuesta para evaluar ambas versiones. La metodología permite obtener los parámetros de los algoritmos y compararlos para determinar cuales proporcionan un mejor rendimiento. Los algoritmos se implementaron y fueron evaluados utilizando los parámetros obtenidos utilizando la metodología propuesta, para comprobar su efectividad.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El mundo que nos rodea esta caracterizado por contar con un gran número de señales analógicas que varían entre un valor mínimo y otro máximo, pasando por valores infinitos. El vídeo y el audio son dos ejemplos importantes de este tipo de señales. Para poder utilizarlas de mejor manera, es habitual tratarlas utilizando convertidores analógicos-digitales, obteniendo así una señal digital que puede ser procesada mas fácilmente. Sin embargo, la conversión de una señal analógica a su equivalente digital puede provocar pérdidas de información. Por tal motivo, el correcto tratamiento de dichas señales tiene una gran importancia para el control de procesos y dispositivos.

La degradación de las señales digitales de voz son causadas por modificaciones indeseables de las señales durante el proceso de grabación, almacenamiento y/o transmisión [13]. Para mejorar la calidad de las señales capturadas por un sistema digital y que han sido degradadas, se han propuesto una gran cantidad de filtros digitales. Un filtro es un dispositivo en forma de hardware o software para a) eliminar el ruido de los datos de un sistema, b) extraer información de acuerdo a algunas características del sistema; c) predecir el comportamiento del sistema a analizar; y d) reconstruir el comportamiento a analizar [29, 23]. Para evaluar el desempeño de estos filtros, diferentes métricas han sido propuestas [6]. Es común que un filtro o algoritmo para el procesamiento de señales digitales se implemente como un módulo de software en computadoras o sistemas inmersos.

En los últimos años, los avances de las tecnologías de la información han permitido el desarrollo de aplicaciones muy especializadas y complejas. La gran capacidad de integración electrónica que existe hoy en día, ha motivado el desarrollo de sistemas digitales de procesamiento compactos, de bajo costo y de bajo consumo de energía. Como resultado, las aplicaciones actuales en donde se involucran señales de voz requieren de sistemas de procesamiento robustos, eficientes en su implementación y compactos [19].

La diferencia principal entre las estrategias de procesamiento existentes, radica en la gran variedad de tipo de perturbaciones que afectan a las señales. Algunos ejemplos típicos de estas perturbaciones son ruido aditivo del sensor, ruido acústico del ambiente, ruido electrónico, ruido tipo impulsivo, entre otras

[22]. Además de una buena calidad en la señal, se necesitan algoritmos eficientes y compactos que restauren correctamente la señal de voz.

Existen varios métodos para el tratamiento de la señal y eliminación del ruido. Como ejemplo de algunos de ellos se tienen los filtros Wiener [38] y Kalman [18], que son útiles para reducir el ruido aditivo. Otro enfoque consiste en el uso de filtros lineales localmente adaptativos, que permiten que el filtro lineal tenga cierta capacidad de adaptación, como en los filtros Wiener. Sin embargo, cuando la señal de voz esta perturbada por varios tipos de ruido, estos filtros no son muy eficientes. En contraste, existen filtros digitales no lineales los cuales se caracterizan por procesar las señales sin destruir los detalles finos [5]. Ejemplos de estos filtros son los filtros recortados, los filtros de orden prioritario y los filtros de mediana estadística, por mencionar algunos [26].

Las técnicas no lineales tienen un enfoque poco robusto, ya que solo pueden utilizarse para perturbaciones cuya distribución estadística es conocida. Una evolución de los filtros no lineales esta representada por los filtros de orden localmente adaptativo. Este tipo de filtros explotan las características estadísticas locales de las señales y estiman en cada paso el valor correspondiente de la señal libre de ruido. Para tal fin, utilizan técnicas de estadística robusta, tales como la creación de vecindarios EV, vecindarios KNV y vecindarios ER [22].

El desempeño de los diferentes métodos existentes para el procesamiento de voz puede medirse en términos de diferentes criterios de calidad. Los criterios de calidad pueden clasificarse en dos categorías principales: métricas cualitativas y cuantitativas. Las métricas cualitativas se basan en el uso del criterio subjetivo de los seres humanos. Por otro lado, las métricas cuantitativas, basan sus resultados en términos de magnitudes escalares claramente medibles y reproducibles.

1.1. Motivación

Existen muchos filtros como los son los Wiener, Kalman, Wiener adaptativo, filtros recortados, los filtros de orden prioritario, los filtros localmente adaptativos y cuanto mas complejos sean computacionalmente tendrán un mayor coste además de que cuentan con muchos parámetros y es difícil elegir los mas adecuados si no se eligen los mejores puede parecer que el filtro no esta funcionando correctamente. La elección de los mejores parámetros nos puede asegurar que los filtros trabajaran de una forma mas adecuada y verificar su eficiencia. Esto se hará en base a una metodología para medir el error del algoritmo. Además poder aplicarlo a dispositivos inmersos como lo es el DsPIC para confirmar que son los mejores parámetros los que se han utilizado.

1.2. Objetivo general

Se propone una metodología para la evaluación de filtros digitales y la elección de sus parámetros. Esta metodología se basa en las métricas de calidad del error cuadrático promedio y el error absoluto promedio. Además, en esta

tesis se propone una modificación a un algoritmo localmente adaptativo publicado recientemente [31] y se utiliza la metodología propuesta para evaluarlo. La metodología permite obtener los parámetros para el algoritmo, así como comparar ambas versiones de los algoritmos para determinar cual proporciona un mejor rendimiento. Los algoritmos se implementaron y fueron evaluados utilizando los parámetros obtenidos utilizando la metodología propuesta, para comprobar su efectividad.

1.3. Objetivos específicos

- Estudio del arte de procesamiento de señales.
- Estudio de filtros localmente adaptativos.
- Desarrollo de una metodología para la implementación de un filtro localmente adaptativo.
- Implementación de un filtro localmente adaptativo en lenguaje C.
- Evaluación del filtro localmente adaptativo implementado.

1.4. Organización del documento

El presente trabajo de tesis esta organizada de la siguiente manera, en el capitulo 2 se muestra el procesamiento de señales empezando por definir una señal digital, ya teniendo esta señal puede ser afectada por señales no deseadas así que se mencionan formas de tratar dicha señal aquí se explican diferentes técnicas que existen para el diseño de filtros digitales, como lo son los filtros pasa bajas,pasa altas,FIR,IIR,Wiener,entre otros. Además se explica la teoría de los estimadores robustos, también algunos filtros no lineales como el recortado o los de orden prioritario y filtros L. Por último los métodos adaptativos como lo son el vecindario KNV, ER y EV.

En el capitulo 3 se presentan lo que es una metodología así como sus divisiones en cuantitativa y cualitativa en la primera se describen las métricas de calidad que existen como lo son la relación señal ruido, los coeficientes de correlación, el error cuadrático promedio y el error absoluto promedio. Al final de este capitulo se da una descripción de la metodología que se propone.

En el capitulo 4 contiene una breve historia de lo que son los DsPIC de como se diseñaron para el procesamiento de señales. Los diferentes DsPIC que existen y sus características mas sobresalientes, las herramientas de desarrollo con las que trabaja el DsPIC (MPLAB) y una descripción del DsPIC starter kit que es dispositivo en el cual se utilizo el filtro.

En el capitulo 5 se explica el filtro utilizado en el DsPIC y se da un ejemplo de como funciona el filtro.

En el capitulo 6 se describe un ejemplo de la aplicación de la metodología, se muestra que rango de valores se usaron para la ventana deslizante S y los

vecindarios EV, así como el valor de L. Además se observan las gráficas y las tablas obtenidas con dichos valores, así como los dos valores mas significativos para ser utilizados en el DsPIC.

Finalmente en el capitulo 7 se muestran las conclusiones obtenidas al realizar esta investigación y el trabajo futuro a realizar.

Capítulo 2

Procesamiento de Señales

Las señales eléctricas son tensiones o corrientes que contienen información. Además de las señales eléctricas existen otras, de naturaleza magnética, hidráulica, neumática, luminosa, etc.

Las señales pueden ser generadas de forma natural o artificial. Algunos ejemplos de señales naturales son la radiación electromagnética de una estrella, la altura de la marea y la velocidad del viento. Algunos ejemplos de señales artificiales son la emisión de un canal de televisión, las ondas emitidas y recibidas por radares, teléfonos celulares, sonares, etc.

Las señales se representan matemáticamente como funciones de una o mas variables independientes. La variable independiente mas común es el tiempo y algunas señales que dependen de el son la voz, una onda de radio, etc. Otras señales, tales como las imágenes son funciones de dos variables independientes, ya que contienen información de brillo o de color en función de las coordenadas X e Y de un plano.

Procesamiento de señales en tiempo discreto se refiere al procesamiento de señales discretas en el tiempo o en el espacio. Esto implica que solo se conoce el valor de la señal en instantes o en puntos específicos. Sin embargo, la amplitud de la señal es continua, es decir, puede tomar infinitos valores diferentes.

Procesamiento digital de señales añade a la característica anterior la de manejar la amplitud en forma discreta, la cual es una condición necesaria para que la señal pueda ser procesada en un computador digital. La amplitud de la señal solo puede tener un numero finito de valores diferentes.

La discretización en el tiempo es la diferencia mas importante entre el procesamiento digital y el procesamiento analógico. La discretización en el tiempo modifica las formulas de las transformadas, convolución, correlación, etc., el cual se origina cuando la tasa de muestreo es insuficiente, generando una perdida irrecuperable de la información contenida en la señal.

La discretización en la amplitud puede ser casi imperceptible, como cuando se efectúa los cálculos en punto flotante con doble precisión (alrededor de 15 decimales) en un lenguaje de programación de alto nivel, o notoria, si se cuantiza la señal con pocos bits. La discretización en la amplitud puede provocar algunos

efectos indeseados, tales como:

- Si proviene del CAD de la señal, es equivalente a sumarle un cierto tipo de ruido, el cual se denomina *ruido de cuantización*.
- Si afecta a los cálculos y es significativa, puede producir errores importantes, e incluso inestabilidad en algunos sistemas.

El Procesamiento Análogo de las Señales (ASP) es generalmente mas simple que el procesamiento digital el cual requiere típicamente de un filtro análogo, un Conversor Análogo Digital (CAD), un procesador DSP, Con conversor Digital Análogo (CDA) y un filtro análogo para suavizar la salida.

Algunas de las desventajas del procesamiento digital con respecto al análogo son:

- Mayor limitación en frecuencias altas, ya que normalmente se requieren CAD capaces de tomar muestras a una tasa varias veces mayor que la frecuencia de la señal análoga y procesadores capaces de efectuar muchas operaciones por cada muestra recibida.
- El diseño es generalmente mas complejo, ya que incluye aspectos de hardware y de software.
- El rango dinámico en la amplitud (razón entre la señales mas grande y la mas pequeña puede ser procesada) es muy limitado, debido a la discretización en la amplitud. Sin embargo, la disponibilidad actual del CAD de alta resolución (18 a 24 bits) y de procesadores capaces de efectuar cálculos en punto flotante con gran numero de decimales, puede eliminar esta desventaja en muchas casos.
- El rango dinámico en la frecuencia también es muy limitado.

Sin embargo, el procesamiento análogo es incapaz de realizar muchas funciones que el procesamiento digital si puede realizar. Algunas de las ventajas del procesamiento digital con respecto al análogo son:

- El envejecimiento de los componentes y las derivas térmicas no afectan al resultado del proceso.
- Todos los dispositivos fabricados se comportan en forma idéntica, ya que la tolerancia de los componentes no influye en el procesamiento.
- Se puede reconfigurar un dispositivo modificando los valores de algunos coeficientes; no es necesario ajustar potenciómetros, o remplazar componentes.
- El procesamiento análogo de señales de muy baja frecuencia se dificulta debido al requerimiento de capacitores de gran capacidad y muy baja corriente de fuga. En el caso del procesamiento digital no existen limitaciones; se puede procesar señales con periodos de horas (tales como las mareas) e incluso años (manchas solares).
- El procesamiento digital es capas de realizar tareas muy complejas.

2.1. Procesamiento digital de señales

2.1.1. Historia del procesamiento digital de señales

El mundo que nos rodea esta caracterizado por contar con un gran numero de señales analógicas o continuas que varían entre un valor mínimo y otro máximo pasando por infinitos valores. El correcto tratamiento de estas señales tiene una gran importancia para el control de procesos y dispositivos. Hasta mediados del siglo XX se utilizaron recursos y técnicas analógicas cuyo esquema se muestra en 2.1. En este esquema el sensor de entrada es el encargado de medir el valor instantáneo de la señal a procesar, como podría ser la señal de voz. El sistema de procesamiento analógico recibe dicha señal y la manipula para transformarla en otra del mismo tipo que se aplica a un actuador.

Figura 2.1: Técnicas y dispositivos analógicos

Desde que en 1971 Intel comercializó el primer microprocesador, las posibilidades y ventajas del procesamiento digital se incrementaron de forma exponencial. Un microprocesador es la parte mas compleja e importante de un computador digital. En nuestros días existen una gran variedad de circuitos integrados con millones de transistores.

Dado que el procesamiento digital de señal solo requiere efectuar ciertos cálculos a partir de los datos disponibles y que en muchas ocasiones dichos cálculos pueden ser efectuados en forma manual. Uno de los primeros avances en DSP fue el articulo *Certain topic in Telegraph Transmition Theory*, publicado por Harry Nyquist en 1928, en el cual se presento el efecto producido en el espectro de frecuencia de una señal análoga al ser discretizada en el tiempo y se planteo que, para preservar la información original, la tasa de muestreo debía ser mayor que el doble de la máxima componente de frecuencia contenida en la señal análoga.

Posteriormente, en 1949, Claude Shannon en *Comunication in the Presence of Noise*, donde demostró que es posible reconstruir perfectamente un señal análoga a partir de sus muestras, si se dispone de un filtro pasabajas análogo ideal. (si bien no es posible fabricar un filtro ideal, es posible aproximarse bastante).

El procesamiento digital de una señal requiere de la realización de un gran numero de cálculos, haciendo invisible si no se dispone de una maquina cal-

culadora de gran velocidad o de un computador. Este problema dificulto el avance en el área de DSP hasta los años 60 y 70, época en la cual progreso rápidamente, gracias a la disponibilidad de grandes computadores, algunos de los tópicos abordados fueron:

- Diseño e implementación de filtros digitales.
- Intensión y optimización del algoritmo de la FFT.
- Compresión de voz.
- Procesamiento de imágenes.
- Sismología.

En esa época las aplicaciones de DSP al procesamiento de las señales en tiempo real eran muy limitadas tales como radares, cancelación de ecos, módems, etc. Los procesadores DSP se construían con centenares de circuitos integrados TTL, tenían un costo prohibitivo y eran muy complejos. Por lo tanto, la mayoría de los trabajos consistían en desarrollar y ensayar algoritmos en los grandes computadores que poseían las universidades, empresas y otras instituciones.

La capacidad de efectuar multiplicaciones en forma rápida es el requerimiento mas importante para poder realizar algoritmos de DSP en tiempo real. Las CPU de esa época no eran capaces de multiplicar en forma directa, si no que lo hacían en base a sumas y desplazamientos. La multiplicación en hardware requería en una gran área en la pastilla de silicio. A principio de los años 80 se logro reducir el tamaño de los transistores lo suficiente como para poder fabricar un procesador DSP capaz de multiplicar 2 números en 1 ciclo de maquina (800nseg). Con el progreso de la tecnología de integración, este tiempo ha disminuido actualmente a unos pocos nseg. El desarrollo de CAD y CAD cada vez mas rápido, de mayor resolución.

Los avances electrónicos en microelectrónica, así como el desarrollo de métodos matemáticos eficientes facilito el análisis y tratamiento de las señales continuas. Así, la serie básica de Fourier es una extraordinaria herramienta para el procesamiento de señales periódicas, mientras que la transformada de Fourier lo es para las señales aperiódicas. Con la Transformada Discreta de Fourier (DFT) y la Transformada Rápida de Fourier (FFT) se consiguió reducir drásticamente el numero de multiplicaciones necesarias en los cálculos, lo que unido a la aplicación de la transformada de Laplace y especialmente su evolución a la transformada Z constituyo excelentes métodos matemáticos muy propicios para su resolución mediante computadores digitales.

El procesamiento digital de señales se ha consolidado como una línea de investigación importante dentro de las ciencias de la tecnología y la ingeniería. El análisis, extracción de características y el reconocimiento de patrones en señales 1D, 2D y 3D, como audio, imágenes y vídeo, son temas de gran interés por su gran impacto social. En sus inicios, el procesamiento de señales rindió frutos en aplicaciones analógicas basadas en sistemas de componentes electrónicos pasivos. Sin embargo, la falta de flexibilidad del hardware limitó en gran

medida el desarrollo de aplicaciones compactas, robustas y programables. Con la llegada de las computadoras electrónicas digitales, estos problemas se redujeron drásticamente. No obstante, la limitada capacidad de procesamiento de estos equipos, trajo consigo nuevos problemas que afectaron directamente la velocidad de procesamiento de las aplicaciones. Posteriormente, con el surgimiento de nuevos campos del conocimiento como las ciencias de la computación y la electrónica digital, el procesamiento digital de señales ha tenido un desarrollo muy significativo; hoy en día existen algoritmos que tienen un excelente desempeño, para el procesamiento de información orientados a una gran gama de aplicaciones que involucran señales multidimensionales y multicomponente. Además, la técnicas actuales pueden implementarse de forma eficiente en poderosos equipos digitales de alta integración electrónica, conocidos como sistemas inmersos [12].

2.1.2. Muestreo de las señales analógicas

La señal analógica se muestre para ir pasando las muestras a un Conversor Analógico Digital (CAD) que las convierte en un formato digital para introducirlas al procesador. Una vez obtenidos los resultados digitales, nuevamente se convierten los mismos en una señal analógica con un Convertidor Digital Analógico (CDA) de salida que puede transformarse de nuevo en una magnitud física por medio de un actuador.

La señal analógica generada por el sensor se muestrea mediante un circuito de Captura y Mantenimiento o Sample-and-Hold (S&H) que va tomando cada cierto tiempo (t) una muestra que carga y mantiene su valor en el condensador mientras el CAD realiza su conversión a digital. El tiempo entre cada muestra se denomina periodo de muestreo (Tm = 1/ Frecuencia de muestreo). Un inconveniente del proceso de muestreo de la señal analógica es la perdida de información entre muestras.

La frecuencia de muestreo es el numero de muestras que se realizan por segundo. Cuanto mayor sea esta frecuencia, mas parecido sera el resultado obtenido a la señal original. Para representar con precisión una señal analógica, el teorema de muestreo (Nyquist-Shannon) establece que la frecuencia de muestreo debe ser igual o mayor que dos veces la máxima componente de frecuencia de la señal original (ancho de banda de la señal de muestreo) $F_{muestreo} \geq 2 \cdot F_{max}$.

Tanto el CAD como el procesador digital deben disponer del tiempo suficiente entre cada dos muestras para llevar a cabo las labores que tienen asignadas. Los procesadores especializados en trabajar en el campo de las señales deben disponer de un conjunto de instrucciones muy potentes y rápidas, siendo la multiplicación la operación mas importante que debe soportar de forma óptima, ya que en caso contrario, supondría un verdadero cuello de botella en el calculo en la mayoría de los algoritmos.

A continuación se citan las ventajas mas relevantes:

- La información digital se almacena y se transfiere con seguridad y rapidez

posibilitando la realización de cálculos matemáticos complejos con gran exactitud y mínimo tiempo.

- La tolerancia de los componentes activos y pasivos que conforman los circuitos electrónicos analógicos unida a la variedad de sus valores y su comportamiento ante las condiciones ambientales externas y su edad, supone una grave falta de precisión que afecta al funcionamiento general de dichos circuitos. Estas alteraciones contrastan con la exactitud de los resultados obtenidos en los procesadores digitales que dependen exclusivamente del programa ejecutado.
- La modificación de un parámetro o una operación en un circuito electrónico analógico supone el rediseño completo del mismo, mientras que esta misma situación se resuelve rápidamente en el procesamiento digital alterando el programa y manteniendo intacto el equipo físico.
- El menor coste y volumen que caracterizan los sistemas digitales son otro factor clave en su utilización.

Un claro ejemplo de procesamiento digital de señales en nuestro entorno es el tratamiento de las señales de sonido captadas por un micrófono, el cual se ilustra en la figura 2.2 en la cual el micrófono transforma una señal analógica como la voz. A dicha señal se le aplica un CAD, cuya salida se trata en un procesador digital de señales según el algoritmo implementado en el programa con el fin de eliminar ruido de la señal obtenida del micrófono. El resultado de este procesamiento se aplica un CDA para obtener de nuevo una señal analógica que se reproduce en un altavoz.

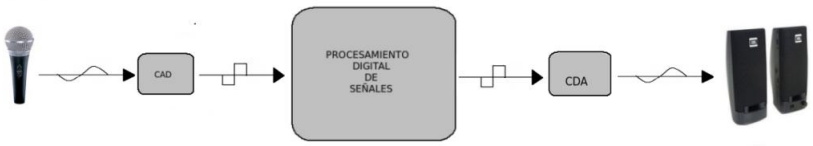

Figura 2.2: Procesamiento Digital de Señales

2.2. Métodos para el procesamiento de señales

2.2.1. Filtro pasa altas

Los filtros pasa altas se forman cuando la salida de un circuito RC se toma de la resistencia como se muestra en la figura 2.3 atenúan el voltaje de salida de todas las frecuencias que están por debajo de la frecuencia de corte f_c,

para frecuencias superiores a la de corte, la magnitud del voltaje de salida es constante.

La función de transferencia es

$$TF(\omega) = \frac{V_{sal}}{V_{ent}} = \frac{R}{R + \frac{1}{j\omega C}} \tag{2.1}$$

cuando multiplicamos todo por $j\omega C$ nos queda la función de transferencia como sigue:

$$TF(\omega) = \frac{j\omega RC}{1 + j\omega RC} \tag{2.2}$$

para el filtro pasa altas la frecuencia de corte es

$$\omega_c = \frac{1}{\tau} = \frac{1}{RC} \tag{2.3}$$

En los filtros pasa altas se diseñan para dejar pasar las frecuencias superiores a la frecuencia de corte ω_c. También es posible formar un filtro pasa altas cuando la salida de un circuito RL se toma desde la bobina. Aunque veremos solo este caso mas básico.

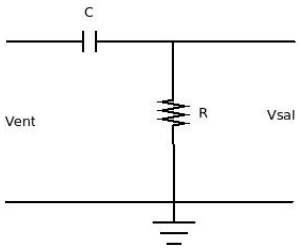

Figura 2.3: Filtro pasa altas RC

2.2.2. Filtro pasa bajas

Permite el paso de señales de baja frecuencia desde la entrada hasta la salida, mientras que atenúa las señales de alta frecuencia.

Filtro pasa bajas RC a bajas frecuencias el capacitor tiene una reactancia (oposición ofrecida al paso de la corriente alterna) muy grande, en consecuencia, se comporta en esencia como un circuito abierto, que resulta en un voltaje en el capacitor V_{sal}, que es prácticamente igual al voltaje aplicado V_{ent}. A altas frecuencias, el capacitor tiene una reactancia muy pequeña la cual prácticamente pone en corto circuito las terminales de salida. Por lo tanto, el voltaje en la salida se aproxima a cero conforme la frecuencia se incrementa.

El circuito de la figura 2.4 se analiza con facilidad al aplicar la regla del divisor de voltaje.

$$V_{sal} = \frac{Z_c}{R + Z_c} V_{ent} \tag{2.4}$$

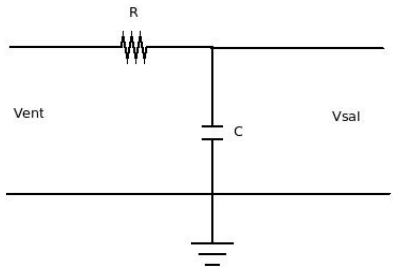

Figura 2.4: Filtro pasa bajas RC

La función de transferencia del circuito se expresa como

$$TF(\omega) = \frac{1}{1 + j\omega RC} \tag{2.5}$$

Se define la frecuencia de corte ω_c, como aquella en la cual la potencia de salida es igual a la mitad de la potencia de salida máxima. Esta frecuencia ocurre cuando el voltaje de salida tiene una amplitud de 0.7071 del voltaje de entrada. Para el circuito RC, la frecuencia de corte ocurre en

$$\omega_c = \frac{1}{\tau} = \frac{1}{RC} \tag{2.6}$$

la función de transferencia se escribe ahora como

$$TF(\omega) = \frac{1}{1 + j\frac{\omega}{\omega_c}} \tag{2.7}$$

Filtro pasa bajas RL este se construye con un resistor y un inductor como se ilustra en la figura 2.5. De manera similar a la que se uso en el filtro pasa bajas RC, se puede escribir la función de transferencia como sigue:

$$TF(\omega) = \frac{R}{R + Z_L} = \frac{R}{R + j\omega L} \tag{2.8}$$

ahora al dividir el numerador y el denominador por R, se tiene la función de transferencia expresada como

$$TF(\omega) = \frac{1}{1 + j\omega\frac{L}{R}} \tag{2.9}$$

y como la frecuencia de corte se encuentra que es:

$$\omega_c = \frac{1}{\tau} = \frac{R}{L} \tag{2.10}$$

y así

$$TF(\omega) = \frac{1}{1 + j\frac{\omega}{\omega_c}} \tag{2.11}$$

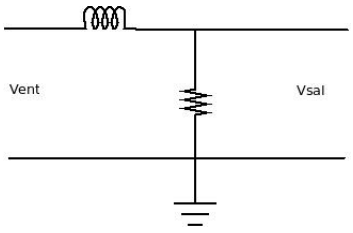

Figura 2.5: Filtro pasa bajas RL

2.2.3. Filtro pasa banda

Un filtro pasa banda permite pasar frecuencias dentro de cierto intervalo desde la entrada del circuito hasta la salida. Todas las frecuencias que caigan fuera del intervalo deseado serán atenuadas y no aparecerán con potencia apreciable en la salida. Este circuito se construye con facilidad con un filtro pasa bajas en cascada con un pasa altas, como se observa en la figura 2.6.

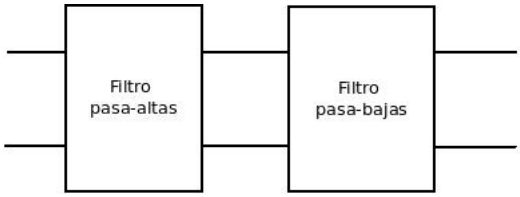

Figura 2.6: Diagrama de bloques de un filtro pasa banda

Aunque los bloques pasa bajas y pasa altas pueden consistir de varios combinaciones de elementos, una posible es construir la red del filtro completo a partir de resistores y capacitores como se muestra en la figura 2.7.

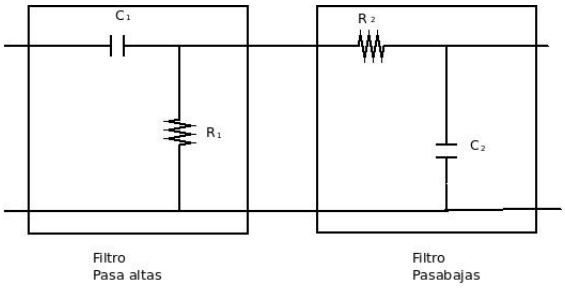

Figura 2.7: Filtro pasa banda

El ancho de banda del filtro pasa banda que resulta sera aproximadamente igual a la diferencia entre las dos frecuencias de corte, es decir

$$AB \cong \omega_2 - \omega_1 \tag{2.12}$$

2.2.4. Filtro rechaza banda

El filtro que evita el paso de una banda de frecuencia entre dos valores designados (ω_1 y ω_2) se conoce variablemente como filtro rechaza banda, para banda o muesca. Un filtro rechaza banda se forma cuando la salida del circuito resonante en serie RLC se toma de la combinación en serie LC como se muestra en la figura 2.8.

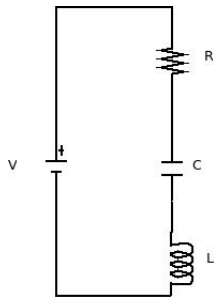

Figura 2.8: Filtro rechaza banda RCL

La función de transferencia es

$$TF(\omega) = \frac{V_{sal}}{V_{ent}} = \frac{j(\omega L - 1/\omega C)}{R + j(\omega L - 1/\omega C)} \tag{2.13}$$

En este caso, la frecuencia central esta dada por:

$$\omega_0 = \frac{1}{\sqrt{LC}} \tag{2.14}$$

Aquí ω_0 recibe el nombre de frecuencia de rechazo, en tanto que el ancho de banda correspondiente ($B = \omega_2 - \omega_1$) se conoce como el ancho de banda de rechazo. Por lo tanto, el filtro rechaza banda se diseña para detener o eliminar todas las frecuencias dentro de una banda de frecuencias, $\omega_1 < \omega < \omega_2$.

2.2.5. Filtros FIR (Finite Impulse Response)

Los filtros FIR son siempre estables (la energía de su respuesta impulsional esta acotada), por lo que los polos de todos los filtros FIR causales deberán estar siempre en el interior de la circunferencia de radio unidad. Además, como la ecuación en diferencias solo contiene dinámica de la entrada (no tienen memoria de salida), su función de transferencia, en principio, solo tendrá términos en el numerador. Supongamos como ejemplo, el sistema:

$$Y[n] = a\,x[n] + b\,x[n-1] + c\,[n-2] \tag{2.15}$$

cuya función de transferencia es:

$$H(z) = \frac{Y(z)}{X(z)} = a + bz^{-1} + cz^{-2} \tag{2.16}$$

si ahora multiplicamos z^2, tenemos:

$$H(z) = \frac{az^2 + bz + c}{z^2} \tag{2.17}$$

de donde vemos que el filtro tiene tantos polos como indica el orden de la ecuación y todos ellos esta situados en el origen ($z = 0$). En la figura 2.9 podemos ver el diagrama de polos y ceros de la ecuación, donde podemos ver que la ubicación de todos los polos en el origen es la característica de los filtros FIR. Además presenta siempre igual número de polos que de ceros, aunque esta característica no es exclusiva de este tipo de filtros.

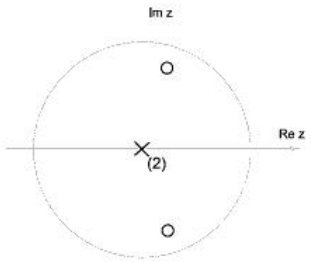

Figura 2.9: Diagrama de polos y ceros

2.2.6. Filtros IIR (Infinite Impulse Response)

Este es un tipo de filtro digital que si su entrada es impulso (una delta de Kronecker) la salida sera un número ilimitado de términos no nulos, es decir, que nunca volverá a un estado de reposo. Para obtener la salida se emplean valores de la entrada actual y anteriores además, valores de salida anteriores que son almacenados en memoria y realimentados a la entrada. También se llaman filtros digitales recursivos. Su expresión en el dominio discreto es:

$$y[n] = \sum_{i=0}^{P} b_i \, x[n-i] - \sum_{j=1}^{Q} a_j \, y[n-j] \tag{2.18}$$

El orden del filtro esta dado por el máximo entre P y Q. Una de las posibles escrituras para un filtro IIR es la siguiente

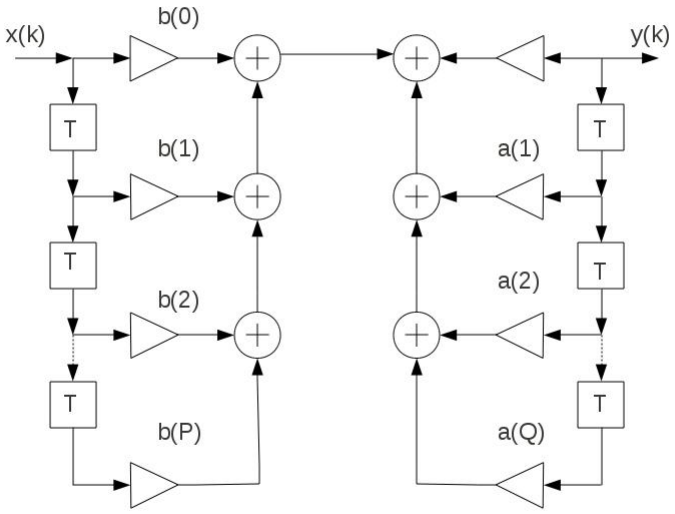

Figura 2.10: Diagrama del filtro IIR

En la figura 2.10 se puede ver como la salida $y(k)$ es introducida de nuevo en el sistema. La transformada Z del mismo es:

$$H(z) = \frac{\sum_{i=0}^{P} b_i z^{-i}}{\sum_{j=0}^{Q} a_j z^{-j}} \tag{2.19}$$

Vemos que ahora tenemos un denominador, es decir, ceros además de polos, que son los causantes de las posibles inestabilidades que pueden comprometer la estabilidad y causalidad del sistema. Las ventajas de los filtros IIR respecto de los FIR es que pueden conseguir una misma respuesta empleado un número de coeficientes en el filtro mucho menor, requiriendo un menor tiempo de cálculo.

2.2.7. Método de substracción espectral

El método de substracción espectral es una aproximación no paramétrica que se basa en la estimación del espectro del ruido [7]. Este método considera que la estimación del espectro de frecuencias del ruido puede estimarse durante los periodos de silencio del locutor. Sea x(n) la señal observada, dada por:

$$F[N] = x[n] + v[n] \tag{2.20}$$

donde $x[n]$ es la señal libre de ruido y $v[n]$ es la función de ruido con distribución Gaussiana. En este caso, se asume que la señal limpia y la función de ruido no están correlacionadas. Sean $F(\omega)$, $X(\omega)$ y $V(\omega)$, la transformada de Fourier de $f[n]$, la transformada de Fourier de $x[n]$ y la densidad espectral del ruido $v[n]$. La señal libre de ruido puede estimarse como la transformada inversa de Fourier de la señal $\hat{X}(\omega)$, la cual puede aproximarse como

$$\hat{X}(\omega) = (|X(\omega)| - \left|\hat{V}(\omega)\right|)exp^{j\angle F(\omega)} \tag{2.21}$$

donde $|X(\omega)|$ es la magnitud del espectro de la señal observada, $\left|\hat{V}(\omega)\right|$ es la magnitud de la densidad espectral de potencia del ruido estimado y $\angle F(\omega)$ es la fase del espectro de la señal observada. En la figura 2.11 se observa el diagrama a bloques del método substracción espectral.

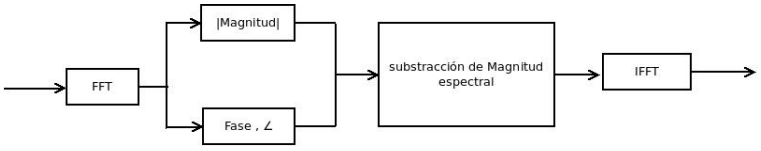

Figura 2.11: Diagrama a bloques del método substracción espectral

El método de substracción de magnitud espectral, es un método simple y eficiente en su implementación. Sin embargo, es común que se presenten efectos secundarios no deseados como ruido musical [7]. El ruido musical es un sonido artificial producido por el algoritmo de procesamiento. El cual consiste en un sonido similar al que se produce al golpear una lata metálica.

2.2.8. Filtro Wiener

El objetivo del filtro Wiener es estimar la señal libre de ruido $x[n]$ a partir de la señal observada $f[n]$. Aquí $x[n]$ es la señal libre de ruido o señal deseada y $v[n]$ es una función de ruido gaussiano donde $E\{v[n]\} = \mu_v = 0$. Se considera que la señal deseada y el ruido son estadísticamente independientes entre si. Sea $e[n] = x[n] - \hat{x}[n]$, la función de error entre la señal deseada $x[n]$ y la señal estimada $\hat{x}[n]$. Aquí, se asume que el filtro $h[n]$ es lineal e invariante al desplazamiento, como a continuación:

$$\hat{X}[n] = \sum_{k=0}^{N-1} f[k]h[n-k] \tag{2.22}$$

La similitud entre la señal de voz de referencia y la señal restaurada se puede medir usando el error cuadrático promedio (MSE); en este caso esta dado por $MSE = E\{[x[n] - \hat{x}[n]]\}^2$, es decir,

$$MSE = E\{[x[n]^2 - 2x[n]\hat{x}[n] + \hat{x}[n]^2]\} \tag{2.23}$$

o también

$$MSE = E\{[x[n]^2 - 2x[n]\sum_{k=0}^{N-1} f[k]h[n-k] + \sum_{l=0}^{N-1}\sum_{k=0}^{N-1} f[k]f[l]h[n-k]h[n-l]]\} \tag{2.24}$$

sustituyendo el modelo de la señal de la ecuación 2.22 en la ecuación 2.24, se obtiene

$$MSE = E\{x[n]^2 - 2\sum_{k=0}^{N-1} x[n]\left(x[n]+v[n]\right)h[n-k] + \sum_{k=0}^{N-1}\sum_{k=l}^{N-1}\left(x[n]+v\left(n\right)\right)\left(x[l]+v[l]\right)h[n-k][n-l]\} \tag{2.25}$$

Después de aplicar el teorema de Perseval's a la ecuación 2.25, el MSE puede expresarse como en el dominio Fourier como a continuación:

$$MSE = \mu_x^2\delta[\omega] - \frac{2}{N}\sum_{\omega=0}^{N-1} H^*[\omega]S_{xx}[\omega] + \frac{1}{N}\sum_{\omega=0}^{N-1} H[\omega]H^*[\omega]S_{nn}[\omega] \tag{2.26}$$

Notemos que las funciones S_{xx}y S_{nn}son positivas, por lo tanto, el valor mínimo de la ecuación 2.26 se obtiene aplicando $\frac{d}{dx}\{MSE = 0\}$, es decir

$$-2S_{xx}[\omega] + H[\omega]S_{xx}[\omega] + H[\omega]S_{nn}[\omega] = 0 \tag{2.27}$$

Finalmente, el filtro de Wiener óptimo, puede expresarse como

$$H(\omega) = \frac{S_{xx}}{S_{xx}[\omega] + S_{nn}[\omega]} \tag{2.28}$$

2.2.9. Filtrado de Wiener Adaptativo

Es un filtro lineal variante en el tiempo, cuyas variaciones corresponden con las características estadísticas locales de la señal de voz. En la figura se muestra el diagrama a bloques para la estimación de la señal de voz.

En este enfoque el valor esperado m_x y la varianza σ_x^2 son medidos localmente recorriendo toda la extensión de la señal durante el proceso de estimación. En este filtro se asume que la función $v[n]$ es ruido blanco con promedio igual a cero y varianza σ_v^2. Observemos que para una ventana deslizante suficientemente pequeña, la densidad espectral de potencia de la función de ruido puede aproximarse como constante

$$P_v(\omega) = \sigma_v^2 \tag{2.29}$$

Así mismo la señal de voz $x(n)$ puede ser considerada como estadísticamente estacionaria en pequeños tramos. En consecuencia, la señal de voz $x(n)$ en un segmento local puede ser modelada como

$$x(n) = m_x + \sigma_x w(n) \tag{2.30}$$

donde m_x y σ_x son el promedio local y la desviación estándar de $x(n)$, respectivamente. La función $w(n)$ representa ruido blanco con varianza igual a la unidad. En cada ventana local la señal de voz, el filtro Wiener puede aproximarse a como a continuación [26].

$$H(\omega) = \frac{P_s(\omega)}{p_s(\omega) + P_v(\omega)} = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_v^2} \tag{2.31}$$

Notemos como la función del filtro depende de los valores estadísticos locales, los cuales deben de ser estimados en cada posición de la ventana deslizante. La ecuación 2.31 se puede representar en el dominio del tiempo como

$$h(n) = \frac{\sigma_s^2}{\sigma_s^2 + \sigma_v^2} \cdot \delta(n) \tag{2.32}$$

por lo tanto, la señal estimada en un segmento local esta dado por

$$\hat{s}(n) = m_x + \frac{\sigma_s^2}{\sigma_s^2 + \sigma_v^2}(x(n) - m_x) \tag{2.33}$$

para estimar los parámetros estadísticos locales de la señal de voz, es necesario calcular la varianza de la señal σ_s^2. Si $\sigma_x^2 = \sigma_s^2 + \sigma_n^2$ entonces $\sigma_2^2(n)$ puede estimarse a partir de $x(n)$ de la siguiente manera

$$\sigma_s^2 = \begin{cases} \hat{\sigma}_x^2(n) - \hat{\sigma}_v^2, & si\, \hat{\sigma}_x^2(n) > \hat{\sigma}_v^2 \\ 0, & si\, otros\, casos \end{cases} \tag{2.34}$$

2.2.10. Predicción Lineal

Consiste en dos etapas principales

1. Se localizan las muestras afectadas por el ruido tipo impulsivo y
2. Las muestras corruptas son removidas y después son remplazadas por valores nuevos obtenidos mediante diferentes técnicas de interpolación [15].

En la primera etapa las muestras con ruido impulsivo (corruptas) son detectadas y removidas mediante el uso de estrategias de predicción. En la segunda etapa, diferentes técnicas de interpolación pueden ser utilizadas para estimar las muestras faltantes o muestras perdidas de la señal proveniente de la etapa 1. En la figura 2.12 se puede observar un modelo iterativo donde se estiman las muestras vecinas. Algunas de las aplicaciones del método de interpolación son conversión de una señal de tiempo discreto a tiempo continuo, baja tasa de

bits de codificación de la señal de voz, el muestreo de una señal para mejorar la representación gráfica, eliminación del ruido tipo impulsivo,etc[37].

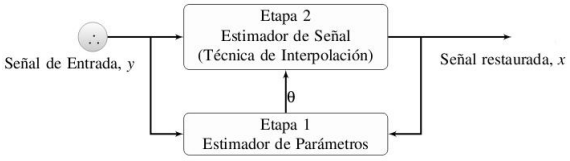

Figura 2.12: Diagrama de flujo de la técnica de interpolación

La idea central de la interpolación por splines cúbicos es construir polinomios para interpolar los datos. Se puede usar segmentos de polinomios entre pares coordenados de datos y unir cada uno de ellos adecuadamente para ajustar lo datos. Así pues, se puede decir de manera informal, que una función spline esta formada por varios polinomios, cada uno definido en un intervalo los cuales se une entre si bajo ciertas condiciones de continuidad. Para un conjunto numeroso de puntos no es muy útil calcular el polinomio interpolante que pasa por estos puntos pues este tiende a tener grandes oscilaciones. En este caso es mas aconsejable hacer una interpolación secuencial de bajo grado sobre subconjuntos pequeños en relación al total de puntos.

La estrategia de interpolación a trozos mas popular se realiza mediante polinomios de tercer grado llamados trazadores cúbicos [8], los cuales pueden definirse como

$$p(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 \tag{2.35}$$

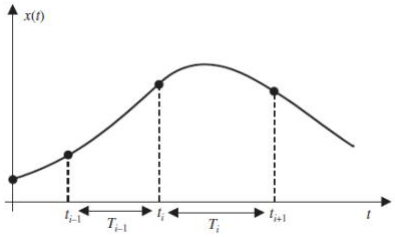

Figura 2.13: Interpolación spline de tercer grado

Un polinomio de tercer orden tiene cuatro coeficientes y necesita cuatro condiciones para la determinación de un sistema único de coeficientes. Para cada intervalo, dos condiciones son fijadas por las muestras en los puntos finales del intervalo. Dos condiciones mas deben satisfacerse por la primera derivada del polinomio a través de cada uno de los dos puntos finales. Consideremos un intervalo $t_i \leq t \leq t_{i+1}$ de tamaño $T_i = t_{1+1}$ como se muestra en la figura 2.13 , cuando $\tau = t - t_i$ el polinomio cubico puede expresarse como

$$p(\tau) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2.36}$$

Así en $\tau = 0$, obtenemos el coeficiente a_0 de la siguiente manera

$$a_0 = p(\tau = 0) = x(t_i) \tag{2.37}$$

aplicando la segunda derivada a $p(\tau)$, se obtiene

$$p"(\tau) = 2a_2 + 6a_3\tau \tag{2.38}$$

Si evaluamos la segunda derivada cuando $\tau = 0$, obtenemos el coeficiente a_2 de la siguiente manera

$$a_2 = \frac{p"(\tau = 0)}{2} = \frac{p"_i}{2} \tag{2.39}$$

de forma similar se evalúa la segunda derivada en el punto $t_i + 1$ (es decir, $\tau = T_i$) para obtener el coeficiente a_3

$$a_3 = \frac{p"_{i+1} - p"_i}{6T_i} \tag{2.40}$$

El coeficiente a_1, se obtiene evaluando $p(\tau)$ cuando $\tau = T_i$, de la siguiente manera

$$p(\tau = T_i) = a_0 + a_1 T_i + a_2 T_i^2 + a_3 T_i^3 = x(t_i + 1) \tag{2.41}$$

observemos que si se sustituyen los coeficientes a_0, a_2 y a_3 de las ecuaciones 2.37,2.39 y 2.40

$$a_1 = \frac{x(t_i + 1) - x(t_i)}{T_i} = \frac{p"_{i+1} + 2p"_i}{6} T_i \tag{2.42}$$

El polinomio cubico se puede expresar como

$$p(\tau) = x(t_i) + \left[\frac{x(x(t_i + 1) - x(t_i))}{T_i} - \frac{p"_{i+1} + 2p"_i}{6} T_i \right] \tau + \frac{p"_i}{2}\tau^2 + \frac{p"_{i+1} - p"_i}{6T_i}\tau^3 \tag{2.43}$$

Para determinar los coeficientes de la ecuación 2.43 es necesario evaluar las funciones $p"_i$ y $p"_{i+1}$, los cuales se obtienen de las restricciones de las primeras derivadas en los puntos finales de cada intervalo donde es continua. En la ecuación 2.43, la primera derivada de $p(\tau)$ es evaluada en los puntos finales t_i y t_{i+1}, como a continuación

$$p'_i = p'(\tau = 0) = -\frac{T_i}{6}[p"_{i+1} + 2p"_i] + \frac{1}{T_i}[x(t_{i+1}) - x(t_i)], \tag{2.44}$$

$$p'_{i+1} = p'(\tau = T_i) = \frac{T_i}{6}[2p"_{i+1} + p"_i] + \frac{1}{T_i}[x(t_{i+1}) - x(t_i)] \tag{2.45}$$

para calcular el intervalo $t_{i-1} \leq t \leq t_i$, se obtiene la primera derivada cuando $\tau = t_i$. De esta forma se obtiene

$$p'_i = p'(\tau = t_i) = \frac{T_{i-1}}{6}[2p"_i + p"_{i-1}] + \frac{1}{t_{i-1}}[x(t_i) - x(t_{i-1})] \tag{2.46}$$

2.2.11. Banco de filtros

Son un conjunto de filtros interconectados entre si para separar las señales en sub-bandas de frecuencia. Los bancos de filtros son una herramienta importante en aplicaciones tales como analizadores de espectros, sistemas telefónicos privados, comprensión de señales, separación de señales sub-bandas, implementación de multiplexores, entre otras.

Figura 2.14: Banco de filtros de dos canales

La figura 2.14 muestra la estructura básica de un banco de filtro. En esta figura, H_0 y H_1 representa un filtro pasa bajas y un pasa altas, respectivamente, en el banco de análisis. Así mismo, F_0 y F_1 representan a un filtro pasa bajas y un pasa altas, respectivamente, en el banco de síntesis.

Los filtros H_0 y H_1, están diseñados para separar la señal de entrada en diferentes bandas de frecuencias, proporcionando sub-señales cuya distribución de frecuencias ocupan solo una porción de la banda de frecuencias de la señal de origen. Los filtros H_0 y H_1 poseen una frecuencia de corte dentro del rango $\left[-\frac{\pi}{M}, \frac{\pi}{M}\right]$ para eliminar o reducir el traslape producido por el sub-muestreo. Observemos que las diferentes sub-señales producidas a la salida de los filtros H_0 y H_1 puede ser convenientemente procesadas mediante diferentes algoritmos de codificación, modulación, comprensión, entre otras, dependiendo del tipo de aplicación.

Una vez que las diferentes sub-señales han sido procesadas, cada una de ellas entra a la etapa de síntesis, la cual consiste de sobre muestreadores que insertan ceros en las muestras recibidas para obtener nuevamente la longitud inicial de la señal. A continuación, el conjunto de sub-señales es filtrada por el banco de filtros de salida que contiene la tarea de eliminar los efectos indeseables producidos por el muestreo de un rango de $|w| \leq \pi$. Finalmente, se combinan las sub-bandas de frecuencias para obtener la secuencia original con un atraso, de $(O_r - 1)/2$ muestras, donde O_r es el orden del filtro.

La selección del tipo de banco de filtros tiene una influencia muy significativa en el desempeño del sistema de procesamiento, en términos de calidad de la señal procesada, complejidad computacional del sistema y tiempo de retraso de la señal de salida respecto a la entrada. Una opción muy común, es la

construcción de banco de filtros en base al uso de la transformada discreta de Fourier de tiempo corto (STDFT). Este tipo de banco de filtros son computacionalmente eficientes sin embargo, producen una descomposición de amplitud uniforme en las diferente bandas de frecuencias que no es muy adecuada para el sistema auditivo humano. Esto se debe a que el oído humano tiene una respuesta en frecuencia que decae de manera gradual conforme la frecuencia de la señal aumenta [14]. Para compensar estos efectos, una alternativa importante es construir banco de filtros mediante la transformada Wavelet [32]. Son ondas localizadas en un tiempo determinado y con una frecuencia en particular, que a diferencia de las funciones base de algunas transformadas ortogonales como la Transformada de Fourier que oscilan todo el tiempo, los wavelets convergen a cero después de un periodo de tiempo. De esta forma, al variar los parámetros de tiempo y frecuencia de la ondicula, es posible hacer diferentes representaciones de la señal de entrada en diferentes sub-señales a diferente tiempo-frecuencia.

De manera general la transformada Wavelet puede expresarse como a continuación

$$x(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \cdot \psi^k \left(\frac{t-\tau}{s} \right) dt \tag{2.47}$$

donde $x(t)$ es la señal de entrada, ψ es la función Wavelet madre, τ es el parámetro de translación que identifica la localización de la función que se desplaza a través de la señal s es un parámetro de escala que corresponde a la información de frecuencia.

2.3. Técnicas de estimación robusta

Algunas de las estrategias de estimación robusta mas importantes que han sido utilizadas para la construcción de filtros digitales son los estimadores M,L y R [5], los cuales se describen enseguida.

Estimador M es una generalización del estimador de Máxima verosimilitud se obtiene al minimizar una función característica que depende de la función de densidad de probabilidad conjunta. Sea "f" una función real, el estimador M se define como

$$\hat{\emptyset} = arg\ max \left\{ \sum_{i=1}^{N} p(X_i, \emptyset) \right\} \tag{2.48}$$

donde $p(X_i, \emptyset)$ es la función M construida comúnmente de la siguiente manera:

- si $p(X_i, \emptyset) = (X_i, \emptyset)^2$ entonces la estimación es $\hat{\emptyset} = MEAN \{X_i \mid i = 1, 2, ..., N\}$
- si $p(X_i, \emptyset) = |X - \emptyset|$ entonces la estimación es $\hat{\emptyset} = MEDIAN \{X_i \mid i = 1, 2, ..., N\}$

Una de las propiedades mas importantes del estimador M es que puede controlar la función p mientras mantiene la naturaleza del estimador de máxima verosimilitud, del cual proviene su nombre.

Estimador L su nombre proviene de "combinación lineal de estadística de orden" [24]. En estadística de orden, las variables de la muestra aleatoria $\{X_1, X_2, .., X_N\}$ son ordenadas de forma ascendente respecto a su magnitud. De esta forma, diferentes estimaciones pueden realizarse a partir de la muestra ordenadas $\{X_1^\circ, X_2^\circ, ..., X_N^\circ\}$, donde

$$X_{(1)}^\circ \leq X_{(2)}^\circ \leq ... \leq X_{(N)}^\circ \tag{2.49}$$

donde cada elemento $X_{(i)}^\circ$ se conoce como la i-ésima estadística de orden [16]. Los estimadores tipo L son una combinación lineal de las estadísticas de orden de una muestra aleatoria y se expresan de la siguiente forma

$$\hat{\emptyset} = \sum_{i=1}^{N} a_i X_{(i)}^\circ \tag{2.50}$$

Aquí X_i° es el i-ésimo valor ordenado de las muestras $\{X_i \mid i = 1, 2, ..., N\}$ y $\{a_1 \mid i = 1, 2, ..., N\}$ son coeficientes constantes. Un aspecto clave en el diseño de filtros basados en estimadores es la selección de los coeficientes a_i. Pitas, I. y A. Venetsanopoulus proponen obtenerse matemáticamente en [16].

$$a_i = \frac{\frac{\int^{1/n}(i-1)}{n} h(\lambda) d\lambda}{\int_0^1 h(\lambda) d\lambda} \tag{2.51}$$

donde $h(\lambda)$ es una función real definida dentro del rango [0,1] y satisface la restricción.

Estimador R proviene de Rango de una variable X_i" y se denota por $R(X_i)$. Sea X una muestra aleatoria ordenada de forma ascendente $X = (X_1^\circ, X_2^\circ, ..., X_N^\circ)$. Aun mas, observemos que podemos formar la secuencia $X_{(j)} + X_{(k)/2}$ con $j \leq k \leq N$ donde a cada elemento de la secuencia se le asigna el siguiente coeficiente de peso [24].

$$w_{jk} = \frac{d_{N-k+j}}{\sum_{i=1}^{N} i d_i} \tag{2.52}$$

El estimador R se define como la mediana de la secuencia que asigno el peso w_{jk} a cada uno de los promedios $(X_{(j)} + X_{(k)})/2$.

2.4. Filtros no lineales

Son filtros no lineales aquellos aquellos que no cumplen el principio de superposición. Si bien los filtros lineales presentan la ventaja de que existen herramientas matemáticas que facilitan y simplifican su estudio y diseño, existen multitud de aplicaciones en las que se obtienen mejores resultados utilizando filtros no lineales, esto es debido a la naturaleza no lineal de un buen numero de fenómenos como, por ejemplo, no linealidades debidas al sistema que genera la señal y/o el ruido, no linealidades del canal de transmisión, etc.

Para comprobar si un sistema cumple el principio de superposición, habrá que comprobar si cumple las siguientes dos propiedades:

$$f(x) = y \rightarrow f(\alpha x) = \alpha y \tag{2.53}$$

$$f(x_1 + x_2) = f(x_1) + f(x_2) \tag{2.54}$$

La primera 2.53 establece que, conocida la salida a una determinada entrada, si la entrada se escala por un factor multiplicativo, la salida queda afectada por ese mismo factor. Un ejemplo de función de transferencia que no cumple esta propiedad es la función cuadrática.

La segunda 2.54 propiedad establece que, conocidas las salidas de un determinado sistema a dos entradas por separado, resulta inmediato conocer la salida a una entrada igual a la suma de las dos señales anteriores. Si el sistema es lineal, su valor es igual a la suma de las salidas de cada una de las entradas por separado.

2.4.1. Filtros de estadísticos ordenados

En este tipo de filtros la salida se obtiene después de realizar una ordenación sobre la muestra de entrada al filtro. Si las muestras de entrada son:

$$X_1, X_2, ..., X_{2N+1} \tag{2.55}$$

y las muestras ordenadas:

$$X_{[1]}, X_{[2]}, ..., X_{[2N+1]} \tag{2.56}$$

Se pueden obtener filtros estadísticos ordenados recursivos utilizando muestras ya filtradas previamente. Entre las diferentes ordenaciones destacan:
s

1. ***Ordenación global:***
   La regla de ordenación es independiente de la información contenida en la ventana del filtro.

   a) Ordenación Temporal:
   No se realiza ninguna ordenación de la información contenida en el filtro.

   b) Ordenación Natural o inversa:
   Se ordenan las muestras según el criterio $X_{[i]} < X_{[i+1]}$ o el criterio inverso $X_{[i]} > X_{[i+1]}$.

   c) Ordenación Absoluta:
   Idéntico al anterior, pero realizando el valor absoluto de las muestras de entrada antes de ordenarlas.

d) Ordenación de Nivel:
Generalización del caso anterior, ordenando los datos según su distancia respecto a una referencia o nivel arbitrario (Antes de realizar el valor absoluto, se resta la referencia a todos los números).

2. ***Ordenación Local***:
La regla de ordenación depende de los datos presentes en la ventana.

a) Ordenación de Media:
Particularización de la ordenación de nivel, donde el nivel se toma igual a la media de los datos contenidos en la ventana.

b) Ordenación de la Mediana:
Igual que el caso anterior, pero sustituyendo la media por la mediana.

c) Ordenación de Variación:
Las muestras se ordenan según su derivada unilateral, bilateral, diferencia respecto a la salida anterior del filtro.

2.4.2. Filtros recortados

La idea principal de los filtros recortados es excluir los valores extremadamente pequeños y extremadamente grandes de la muestra de la señal de entrada antes antes de calcular la estimación de salida [5]. Es fácil darse cuenta que un filtro recortado puede obtenerse fácilmente a partir del estimador L. Consideremos los siguientes coeficientes a_i

$$a_i = \begin{cases} \frac{1}{[(1-2\alpha)N]}, & si\, \alpha N \leq i \leq N - \alpha N \\ 0, & en\, caso\, contrario. \end{cases} \tag{2.57}$$

donde $N = 2k$. La estimación L resultante es conocida como promedio de orden [5]. Si en la ecuación 2.57 $\alpha = \frac{j}{N}$ donde $0 \leq j \leq \frac{N}{2}$ es entero, se obtiene el estimador conocido como promedio α recortado [2]. El estimador de promedio recortado puede expresarse como a continuación:

$$\hat{\emptyset} = \frac{1}{((1-2\alpha)\, N)} \sum_{i=\alpha N}^{N-\alpha N} X\,(i) \tag{2.58}$$

Este estimador descarta los αN valores mas pequeños y los αN valores mas grandes antes de efectuar la operación promedio. En su formula mas general, el filtro α recortado descarta los r_1 valores mas pequeños y los r_2 valores mas grandes, pueden expresarse como sigue:

$$recortado(X_1, X_2, ..., X_N; r_1, r_2) = \frac{1}{N - r_1 - r_2} \sum_{i-r_1+1}^{N-r_2} X(i) \tag{2.59}$$

Finalmente observamos que si se utilizan los coeficientes:

$$a_i = \begin{cases} \frac{1}{2}, & si\, i = k \ o \ i = k+1 \\ 0, & en\, caso\, contrario \end{cases} \tag{2.60}$$

donde $N = 2k$, se obtiene el estimador conocido como promedio de orden [2].

2.4.3. Filtros de orden prioritario

También conocido como filtro de rango es aquel que tiene como salida una función sobre las estadísticas de orden de un conjunto de variables aleatorias. Básicamente, un filtro de rango puede expresarse por [21].

$$Filtro\, de\, rango(X_1, X_2, ..., X_n) = f(X_{(1)}, X_{(2)}, ..., X_{(n)}) \tag{2.61}$$

otro filtro de rango importante es el filtro de estadística de orden ponderado. La salida de este filtro con entrada $(X_1, X_2, ..., X_n)$ y pesos $a = (a_1, a_2, ..., a_n)$, se expresa como [5]

$$F(X_1, X_2, ..., X_n; a, i) = i-esima\, estadistica\, de\, orden\, de\, (a_1 \Diamond X_1, a_2 \Diamond X_2, ..., a_n \Diamond X_m) \tag{2.62}$$

2.4.4. Filtro L

Son aquellos que aplican el estimador L a una pequeña ventana deslizante que recorre la señal. Básicamente, un filtro L puede verse como un compromiso entre un filtro no lineal y un filtro lineal. Cada punto a la salida del filtro se obtiene como na suma ponderada de valores de los datos ordenados dentro de una ventana deslizante. El conjunto de peso determina las características del filtro. El conjunto de peso se denota como $a = (a_1, a_2, ..., a_N)$ y sus vectores de entrada $x = (x_1, x_2, ..., x_N)$. La salida del filtro esta dado por [5]

$$L(X_1, X_2, ..., X_N; a) = \sum_{i=1}^{N} a_i X(i) \tag{2.63}$$

A partir de la ecuación 2.63, si $a_i = \frac{1}{N}$; $i = 1, 2, ..., N$, se obtiene

$$\hat{\emptyset} = \frac{1}{N} \sum_{i=1}^{N} X_{(i)} \tag{2.64}$$

que representa la media aritmética de las muestras $X_i s$. Ahora si los coeficientes a_i son

$$a_1 \begin{cases} 1; & si\, i = k = 1 \\ 0; & en\, caso\, contrario \end{cases} \tag{2.65}$$

donde $N = 2k + 1$, se obtiene

$$\hat{\emptyset} = median\{X_1, X_2, ..., X_N\} \tag{2.66}$$

y representa la media de la muestra.

2.5. Filtros en imágenes digitales

El reconocimiento de objetos en imágenes digitales, consiste en reconocer y estimar la posición exacta de un conjunto de objetos de interés a partir de una escena observada. Los sistemas de reconocimiento de objetos más populares realizan tres tareas principales [28] segmentación de objetos, reconocimiento y localización de los objetos reconocidos en la escena. Estos sistemas comúnmente presentan buenos resultados; sin embargo, cuando los objetos están inmersos en un fondo complicado, las técnicas de segmentación no funcionan correctamente. En consecuencia, las etapas posteriores de reconocimiento y localización podrían arrojaran resultados pobres. En contraste, existen filtros de correlación para el reconocimiento de objetos que poseen un buen fundamento matemático, son invariantes al desplazamiento y pueden implementarse de forma eficiente en procesadores digitales o en sistemas híbridos opto-digitales a gran velocidad [33]. Básicamente hay dos diferentes enfoques para diseñar filtros de correlación invariantes a distorsiones como diseño analítico y diseño compuesto. El diseño analítico [39] consiste en derivar la respuesta al impulso del filtro como una expresión de forma cerrada, mediante la optimización de algún criterio estadístico. En el enfoque compuesto [33, 1], los filtros están construidos como una combinación de imágenes de entrenamiento, las cuales deben ser suficientemente descriptivas y representativas del objeto y sus distorsiones esperadas. No nos enfocaremos mas en tratamiento de imágenes por no ser la idea a tratar en la tesis.

2.6. Métodos Adaptativos

Diversos métodos adaptativos para la solución de importantes problemas ligados al procesamiento digital de señales han sido propuestos e investigados [40]. Algunos de los filtros adaptativos mas populares están basados en estimadores no-lineales de estadística de orden prioritario [16, 9, 11, 30]. Además, existen filtros no-lineales muy conocidos como los filtros de mediana estadística [4], filtros de mediana multinivel y multi-etapa [3, 25, 10], filtros de pila, filtros de media truncada en alfa [16], filtros de estadística de orden [16, 30], filtros morfológicos [34, 35], entre otros. Estos filtros han demostrado ser muy efectivos en la eliminación de ruido aditivo e impulsivo y en la restauración imágenes. La principal característica de estos filtros, es que son espacialmente no-homogéneos y sus parámetros se adaptan en función de la información espacial local de las señales como estadística robusta e información espectral, obtenida mediante transformadas de tiempo-corto [27, 17]. Los filtros localmente adaptativos, comúnmente

se diseñan resolviendo un problema de estimación estadística, cuyo objetivo es la optimización de métricas de desempeño y cuyos sus parámetros provienen de los modelos matemáticos de la señal y de las características estadísticas de las funciones de ruido. Los estimadores resultantes pueden procesar las señales sin destruir los detalles finos, como lo hacen los filtros lineales convencionales. No obstante, es común que los métodos adaptativos presenten una complejidad computacional mas alta que los métodos basados en filtrado lineal. Algunos esfuerzos importantes se han realizado para tratar de reducir la complejidad computacional de estos métodos, empleando algoritmos recursivos y transformadas de tiempo-corto [20, 36]. Sin embargo, es necesario desarrollar más investigación en la realización de nuevas estrategias de procesamiento local adaptativo, que sean computacionalmente eficientes. Un aspecto importante es proponer nuevas arquitecturas de procesamiento de información en paralelo y tiempo-real, que exploten al máximo las capacidades de los equipos digitales actuales de arquitectura libre, como lo son los arreglos de compuertas lógicas programables FPGA [12].

La reducción de ruido aditivo y/o ruido tipo impulsivo utilizando técnicas localmente adaptativas, se obtiene aplicando los filtros no lineales a una señal de entrada de forma convencional [22]. Sin embargo, los valores de los elementos de la ventana deslizante son funciones de las relaciones espaciales entre los mismos elementos. A estas estructuras de datos se le conoce como vecindades adaptativas. Las vecindades adaptativas se calculan utilizando las estadísticas de orden prioritario sobre su renglón variacional. El renglón variacional consiste en ordenar los elementos de la ventana deslizante en forma ascendente respecto a su magnitud.

En este caso, la complejidad computacional del calculo de las vecindades adaptativas depende de la complejidad computacional del calculo de los histogramas locales [22].

El renglón variacional se denota como una secuencia unidimensional $\{V(r)\}$ de k elementos los cuales son ordenados de forma ascendente con respecto a sus valores son $\{V(r) \leq V(r+1); r = 1, 2, ..., k\}$. A $V(r)$ y $r(V)$ se les llama la r-ésima estadística de orden y el rango de valor V, respectivamente. Ambas cantidades se pueden obtener del histograma local $h(q), q = 0, 1, ..., Q-1$, niveles de cuantización, de la distribución de la señal dentro de la ventana deslizante, es decir,

$$r(V) = \sum_{q=0}^{V} h(q) \tag{2.67}$$

Para formar una vecindad adaptativa, es necesario crear un vecindario adaptativo al rededor del elemento central en el renglón variacional. En la actualidad, existen diversos criterios para determinar los valores de los vecindarios en una ventana deslizante. Tres de estos criterios para la reducción de ruido aditivo y/o tipo impulsivo, se describen a continuación.

2.6.1. Vecindario EV

Desvía el valores del elemento central v_c, una cantidad $\pm\varepsilon_v$ respecto al valor de magnitud v_t [22], como a continuación:

$$\varepsilon_v(v_t) = \{v_t : v_c - \varepsilon_v \leq v_t \leq v_c + \varepsilon_v\} \tag{2.68}$$

En la figura 2.15 podemos observar los elementos que son considerados dentro del vecindario adaptativo.

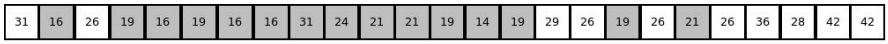

Figura 2.15: Ejemplo vecindario EV

Cuando $EV(19) = \{14 \leq v_c \leq 24\} = \{14, 16, 16, 16, 16, 19, 19, 19, 19, 19, 21, 21, 21, 24\}$.

2.6.2. Vecindario KNV

Se define como un subconjunto de la ventana deslizante formado por un numero especifico de k elementos, cuyos valores son los mas cercanos al valor de la magnitud del elemento central v_c. Se expresa como en [22]:

$$KNV(v_t) = \left\{ v(r) : \sum_{r=p}^{p+k-1} |v_c - v(r) = MIN_p| \right\} \tag{2.69}$$

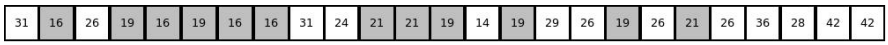

Figura 2.16: Ejemplo vecindario KNV

En la figura 2.16 se consideran los elementos que se consideran dentro del criterio KNV.
$KNV(19) = \{16, 16, 16, 16, 19, 19, 19, 19, 19, 21, 21, 21\}$.

2.6.3. Vecindario ER

Es el conjunto de elementos cuyos rangos definidos en el renglón variacional, se desvía una cantidad predeterminada $\pm\varepsilon_R$ del rango del elemento central pueden expresarse como en [22]:

$$\varepsilon_R(v_t) = \{v_c : r(v_c) - \varepsilon_R \leq r(v_c) \leq r(v_c) + \varepsilon_R\} \tag{2.70}$$

Aquí $\{v_t\}$ es un subconjunto de muestras, (t) son la posición de los elementos contenidos en el vecindario adaptativo, v_c es el valor del elemento ubicado en la posición (c), $r(v_t)$ es el rango que ocupa un elemento con un valor de la magnitud igual a v_c.

En la figura 2.17 se muestra un ejemplo del vecindario cuando el valor de $\varepsilon_R = 3$

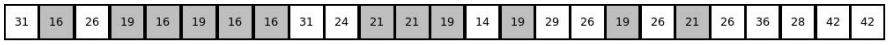

Figura 2.17: ejemplo vecindario ER

Los valores obtenidos para el vecindario ER son:
$ER(19) = \{5 \leq r(19) = 8 \leq 11\} = \{16, 16, 16, 16, 19, 19, 19, 19, 19, 21, 21, 21\}$

Capítulo 3

Metodología

Una metodología es aquella guía que se sigue a fin realizar las acciones propias de una investigación. En términos más sencillos se trata de la guía que nos va indicando qué hacer y cómo actuar cuando se quiere obtener algún tipo de investigación. Es posible definir una metodología como aquel enfoque que permite observar un problema de una forma total, sistemática, disciplinada y con cierta disciplina.

Al intentar comprender la definición de lo que es una metodología, resulta de suma importancia tener en cuenta que una metodología no es lo mismo que la técnica de investigación. Las técnicas son parte de una metodología y se define como aquellos procedimientos que se utilizan para llevar a cabo la metodología, por lo tanto, como es posible intuir, es uno de los muchos elementos que incluye.

En el contexto de la investigación son muchas las metodologías que es posible seguir, sin embargo, existen 2 grandes grupos que incluyen a otras más específicas. Se trata de la metodología de investigación cuantitativa y la cualitativa.

La metodología cuantitativa es aquella que permite la obtención de información a partir de la cuantificación de los datos sobre variables, mientras que la metodología cualitativa, evitando la cuantificación de los datos, produce registros narrativos de los fenómenos investigados. En este tipo de metodología los datos se obtienen por medio de la observación y las entrevistas, entre otros. Como vemos, la diferencia más importante entre la metodología cuantitativa y la cualitativa radica en que la primera logra sus conclusiones a través de la correlación entre variables cuantificadas y así poder realizar generalizaciones y producir datos objetivos, mientras que la segunda estudia la relación entre las variables obtenidas a partir de la observación en contextos estructurales y situacionales.

El desempeño de los diferentes métodos existentes para el procesamiento de voz puede medirse en términos de diferentes criterios de calidad [6]. Los criterios de calidad pueden clasificarse en dos categorías principales son las métricas cualitativas y cuantitativas. Las métricas cualitativas se basan en el uso del criterio subjetivo de los seres humanos. Por otro lado, las métricas cuantitativas, basan sus resultados en términos de magnitudes escalares claramente medibles y

reproducibles. Se pudiera pensar que el enfoque formal que poseen las métricas cuantitativas las hace mucho mejores respecto a las métricas cualitativas para caracterizar el desempeños de los algoritmos de procesamiento. Esto puede ser cierto para ciertos tipos de problemas, por ejemplo, en la supresión de ruido. Sin embargo, para otros problemas típicos como el medir la inteligibilidad de las señales existen métricas cualitativas usadas ampliamente y que en algunos casos puede brindar mejores resultados que las métricas cuantitativas.

Las señales de voz son funciones altamente variantes en tiempo. Esto quiere decir que sus características estadísticas y su distribución espectral varían considerablemente con el paso del tiempo. Esto hace que los métodos de procesamiento globales (invariantes en el tiempo) produzcan resultados pobres. Para solucionar este problema, es deseable diseñar métodos de filtrado que tengan la capacidad de adaptarse a:

1. Las características locales propias de la señal.

2. Las características de las funciones de ruido que corrompen las señales.

El algoritmo propuesto por Sandoval et al en [31] trata de cumplir con ambos objetivos. Por una parte, el uso de estadísticas de orden (renglón variacional) sirve principalmente para aislar los efectos del ruido impulsivo y también para concentrar la acción de procesamiento (estimación) sobre los elementos de las señales mas cercanos a su valor esperado real. Por otra parte, el uso de vecindarios adaptativos locales (criterio EV) sirve principalmente para evitar los efectos de inercia espacial que se producen en las regiones de cambios abruptos en las señales altamente variantes en el tiempo. Los resultados obtenidos hasta el momento con el algoritmo demuestran que el método cumple parcialmente con el objetivo 1 y 2. Las gráficas comparativas entre la señal de entrada y la señal procesada, demuestran que el algoritmo elimina considerablemente los efectos de ruido y se adapta bien a los cambios abruptos que presenta la señal. Por esto las métricas de calidad objetivas como la relación señal a ruido (SNR) y el error cuadrático promedio (MSE) entregan buenos resultados. Sin embargo, cuando se usa el criterio auditivo subjetivo en la señal procesada pueden escucharse algunos efectos no deseables en la región de transición entre cambios abruptos de la señal. Estos efectos se deben principalmente a que los parámetros S, EV y L son fijos a lo largo de las acciones del procesamiento del algoritmo. Bajo esta idea, para un conjunto fijo de parámetros {S,EV,L} el algoritmo se adapta mejor (se cumple mejor los objetivos 1 y 2) a unas regiones de la señal que en otras. Estas regiones deben cambiar si los parámetros varían. La idea es realizar un buen número de experimentos sobre una señal controlada, variando los parámetros mencionados, con la idea de poder establecer relación entre los parámetros y la calidad local de la señal. Así se podrían identificar patrones de comportamiento que ayuden a diseñar estrategias para la selección dinámica de los parámetros durante la acción del procesamiento.

3.1. Relación señal ruido

También conocido con las siglas SNR por su nombre en ingles (Signal to Noise Ratio). Se define como el margen que hay entre la potencia de la señal y la potencia del ruido que la corrompe. Es quizás la medida mas popular para medir la supresión de ruido de un algoritmo.

$$SNR = \frac{P_{señal}(w)}{P_{ruido}(w)} \tag{3.1}$$

donde $P_{señal}$ es la densidad espectral de potencia de la señal de voz y P_{ruido} es la función de ruido respectivamente. Esta relación se calcula en el dominio de la frecuencia.

Al ser las señales de ruido señales aleatorias superpuestas a la señal de medida, tomar un valor promedio de la señal sobre un periodo de tiempo puede utilizarse para reducir el efecto de ruido y mejorar la relación señal ruido.

3.2. Coeficiente de correlación

El coeficiente de correlación es una medida de asociación entre variables. La correlación indica la fuerza y la dirección de una relación lineal entre dos variables aleatorias. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra; si tenemos dos variables (X y Y) existe correlación si el aumentar los valores de X lo hacen también los de Y y viceversa. El coeficiente de correlación entre las variables X y Y, esta dado por

$$p_{x,y} = \frac{cov(x,y)}{\sigma_x \sigma_y} = \frac{E((X-\mu x)(Y-\mu y))}{\sigma_x \sigma_y} \tag{3.2}$$

donde $cov(x,y)$ es la covarianza entre las variables aleatorias X, Y y σ_x,σ_y es la desviación estándar de X e Y. En la actualidad, existe un gran numero de trabajos donde se proponen nuevos métodos de diseño de filtros digitales para la mejora de la voz [6].

Con rangos desde -1 hasta 1 y medidas de correlación estática entre los valores predichos y los valores actuales en la muestra de prueba un valor de 0 indica que no hay correlación. Una correlación negativa indica correlación inversa, pero no debería ocurrir en para modelos de predicción.

3.3. Error cuadrático promedio (ECP)

Del ingles Mean Squared Error (MSE), aveces es necesario usar un estimador. En tales casos, el error cuadrático promedio del estimador puede ser importante. El error cuadrático promedio de un estimador $\hat{\Theta}$ es el cuadrado de la diferencia entre $\hat{\Theta}$ y θ el cual esta definido como a continuación

$$ECP(\hat{\Theta}) = E(\hat{\Theta} - \theta)^2 \tag{3.3}$$

El cual puede ser reescrito como sigue

$$ECP(\hat{\Theta}) = E[\hat{\Theta} - E(\hat{\Theta})]^2 + [\theta - E(\hat{\Theta})]^2 = V(\hat{\Theta}) + (bias)^2 \tag{3.4}$$

Esto es, el error cuadrático promedio de $\hat{\Theta}$ es igual a la varianza del estimador mas el cuadrado de bias. Si $\hat{\Theta}$ es un estimador imparcial de θ, el error cuadrático promedio de $\hat{\Theta}$ es igual a la varianza de $\hat{\Theta}$.

El error cuadrático promedio es un importante criterio para comparar dos estimadores. Sea $\hat{\Theta}_1$ y $\hat{\Theta}_2$ los dos estimadores del parámetro θ y dejar que $ECP(\hat{\Theta}_1)$ y $ECP(\hat{\Theta}_2)$ sean los errores cuadráticos promedio de $\hat{\Theta}_1$ y $\hat{\Theta}_2$. Cuando la eficiencia relativa de $\hat{\Theta}_2$ a $\hat{\Theta}_1$ es definido como:

$$\frac{ECP(\hat{\Theta}_1)}{ECP(\hat{\Theta}_2)} \tag{3.5}$$

Si esta eficiencia relativa es menor que 1 podemos concluir que $\hat{\Theta}_1$ es un estimador mas eficiente de θ que $\hat{\Theta}_2$, en el sentido que que tiene un error cuadrático promedio menor. Para nuestro caso las estimadores serán como señales de voz. Sea $S_1(t)$ y $S_2(t)$ dos vectores de datos, los cuales representan a un segmento de esta señal de voz libre de ruido, a un segmento de voz que a sido procesada a través de algún procesamiento digital. El error cuadrático promedio de la señal procesada $S_2(t)$ respecto a $S_1(t)$, esta dado por

$$ECP(S_2(t)) = E\left\{(S_2(t) - S_1(t))^2\right\} \tag{3.6}$$

donde $E\{\bullet\}$ es el valor esperado.

Aun mas si $S_1[n]$ y $S_2[n]$ son señales discretas y utilizamos la estimación de Máxima Verosimilitud para el valor esperado, entonces la ecuación 3.6 puede ser reescrita como a continuación:

$$ECP = \frac{\sum_{n=1}^{N} |S_2[n] - S_1[n]|^2}{\sum_{n=1}^{N} S_1^2[n]} \tag{3.7}$$

3.4. Error Absoluto Promedio

Es una cantidad utilizada para medir que tan cercano son las predicciones de los valores estimados por el filtro. Es el promedio de la magnitud de los valores absolutos individuales. Estas métricas de evaluación no tienen un rango fijo como los coeficientes de correlación pero esta orientado a los valores que serán predichos.

El error absoluto promedio (MAE) esta definido como:

$$EAP(S_2(t)) = E\{|(S_2(t) - S_1(t))|\} \tag{3.8}$$

observemos que para señales discretas $S_1[n]$ y $S_2[n]$ utilizando el estimador de máxima verosimilitud para el valor esperado, la ecuación 3.8 puede reescribirse como

$$EAP = \frac{\sum_{n=1}^{N} |S_2[n] - S_1[n]|}{\sum_{n=1}^{N} |S_1[n]|} \tag{3.9}$$

3.5. Descripción de la metodología

Para nuestro caso específico utilizaremos dos métricas de calidad, sea $Y_{clean}(x)$ y $Y_{proc}(x)$, la señal libre de ruido y la señal procesada respectivamente. La primera métrica es el error cuadrático promedio MSE es una medida global del error total entre dos vectores de datos. De manera formal el MSE puede describirse como

$$MSE = \frac{1}{N} \sum_{x=1}^{N} [Y_{clean}(x) - Y_{proc}(x)]^2 \tag{3.10}$$

En algunas ocasiones, el MSE puede fallar en registrar el error entre valores pequeños. Una alternativa para evitar este problema es la segunda métrica a utilizar la cual es el error absoluto promedio MAE, dado por:

$$MAE = \frac{1}{N} \sum_{x=1}^{N} |Y_{clean}(x) - Y_{proc}(x)| \tag{3.11}$$

En este caso, tanto el MSE como el MAE proporcionan un valor escalar como resultado. Este valor indica el valor promedio del error total entre las dos señales. Sin embargo no es posible detectar los cambios en el error con el paso del tiempo. Las versiones locales del MSE y del MAE pueden escribirse como:

$$MSE(k) = \frac{1}{S} \sum_{x=k-\frac{(S-1)}{2}}^{x=k+\frac{(S-1)}{2}} [Y_{clean}(x) - Y_{proc}(x)]^2 \tag{3.12}$$

donde $k = 1, 2, 3, ..., N$. También,

$$MAE(k) = \frac{1}{S} \sum_{x=k-\frac{(S-1)}{2}}^{x=k+\frac{(S-1)}{2}} |Y_{clean}(x) - Y_{proc}(x)| \tag{3.13}$$

podemos observar como las versiones locales del MSE y MAE utilizan ambas un vector de datos del mismo tamaño y proporcionan el error local entre las señales involucradas, en un pequeño vecindario alrededor de k.

La metodología consiste en la utilización de las métricas MSE y MAE considerando todos los valores posibles de los parámetros del filtro. Para esto, se asigna inicialmente un valor a alguno de los parámetros y se varían los valores de los demás, dentro de un rango predefinido. Debido a que la cantidad de datos

a evaluar puede ser muy grande, se propone que este proceso se automatice. Con los resultados obtenidos se generan gráficas que permitan visualizar de manera mas sencilla, el desempeño del filtro en función de los diferentes valores de los parámetros. Después del análisis de los resultados, puede probarse el filtro en algún dispositivo especifico, utilizando los valores de los parámetros que mejores resultados proporcionaron.

La metodología propuesta reduce significativamente el rango de valores a evaluar y el tiempo para llevar a cabo las pruebas y permite comparar la eficiencia entre diferentes algoritmos o filtros.

Capítulo 4

DsPIC

En el mundo que nos rodea esta caracterizado por contar con un gran numero de señales analógicas o continuas que varían entre un valor mínimo y otro máximo pasando por infinitos valores. La temperatura, la luz, la presión atmosférica, la humedad relativa, entre otras.

Los DSP (Procesadores Digitales de Señales) un circuito integrado que contiene un procesador digital y un conjunto de recursos complementarios capaces de manejar digitalmente las señales analógicas del mundo real como los sonidos y las imágenes.

A principios de la década de los 80 del siglo pasado ya se comercializaban varios modelos de DSP, siendo los pioneros Texas Instruments, NEC e Intel, en la actualidad Texas Instruments mantiene el liderazgo en el mercado de DSP, fabricando modelos cada vez mas potentes para acoplarse a las aplicaciones mas complejas y cambiantes.

4.1. Microcontroladores y Procesadores Digitales de Señales

Los microcontroladores clásicos, denominados de forma resumida MCU son circuitos integrados que contienen un procesador digital completo junto a diversos periféricos auxiliares que facilitan el desarrollo de las aplicaciones a las que se dedican. Su parecido con los DSP es muy grande, pero las diferencias que los distinguen hacen que sus campos de aplicaciones sean diferentes.

Se puede definir un procesador digital de señales o DSP como un procesador monochip diseñado para resolver un conjunto de operaciones matemáticas sobre una señal continua o analógica expresada digitalmente. De forma resumida puede decirse que los DSP son microcontroladores dotados de los recursos físicos y lógicos necesarios para poder soportar las aplicaciones especificas del procesamiento digital de señales.

Las instrucciones aritméticas complejas de los MCU se ejecutan en varios ciclos, mientras que las de los DSP solo precisan de uno.

En el momento que la expansión de las aplicaciones de los MCU ha alcanzado cotas extraordinarias y se fabrican un sin fin de productos y sistemas que llevan embebidos uno o varios, el nivel científico y los avances tecnológicos ha logrado fabricar DSP que añaden al MCU los recursos para soportar el procesamiento digital de las señales y cubrir así casi todo el campo de posibilidades y aplicaciones que envuelve.

4.2. DSP y DSC

Las aplicaciones modernas mezclan las funciones típicas MCU con las de procesamiento digital de señales (DSP). Esta situación ha impulsado a microchip a fabricar un circuito híbrido MCU/DSP, cuyo manejo es similar a los clásicos microcontroladores pero que incluye las principales prestaciones de los DSP. Así ha nacido el controlador digital de señales abreviado DSC (Digital Signal Controller), que reúne las características de un microcontrolador PIC de 16 bit y las de un DSP de gama baja. En base ha este controlador de señales, Microchip consigue reunir lo mejor de ambos mundos.

La primera generación de DSC, denominada dsPICC30F, constituyo un intento por parte de Microchip de facilitar el acercamiento al mundo del procesamiento digital de señales a sus usuarios de MCU de 8 y 16 bits. Posteriormente la segunda generación de DSC, llamada dsPIC33F, ha potenciado las capacidades, el numero de periféricos y el rendimiento, permitiendo acceder a campos de aplicación mas complejos.

A continuación se describen algunas aplicaciones concretas en las que se pueden encontrar DSP como dispositivos imprescindibles de las mismas.

En medicina son aparatos como los equipos destinados a la monitorización de pacientes, MNR, escáner, electrocardiogramas, electroencefalogramas y tomografías, han logrado que sus imágenes sean mostradas con mucho mas detalle que las técnicas convencionales gracias a los DSP, permitiendo un diagnostico y tratamiento significativamente mejor.

En industria el DSP ha revolucionado áreas como la exploración petrolera, minera, submarina y espacial, el control de motores y la robótica.

Productos de consumo como lo son los televisores, radios, reproductores de musica digital, reproductores de CD, órganos, lavadoras y equipos de aire acondicionado.

Control de motores este es un campo al que los fabricantes de DSP se están dedicando ampliamente ya que al estar presentes en casi todos los procesos industriales el desarrollo de módulos de control de potencia es una inversión segura. De este modo se emplean DSP en sistemas como controladores de motores, inversores de potencia, controladores de posición , impresoras y fotocopiadoras, compresores de alta potencia, etc.

Automoción es la introducción de DSP en esta área ha puesto un incremento de las prestaciones de los automóviles. Se aplica en los activadores de airbag, sistemas de chequeo, control de velocidad de crucero, suspensión activa, control de motor, ordenador de bordo, ABS y control de la combustión.

En tipo militar son el sonar, el radar, el piloto automático y el guiado automático de misiles son algunos de los ejemplos de la utilización en el ámbito militar.

En telecomunicaciones el DSP ha revolucionado la industria de las telecomunicaciones e muchas áreas como la generación y detección de tonos, filtrado para eliminar el ruido de la linea eléctrica, multiplexado, compresión, control de eco,etc. y en productos como son los teléfonos móviles e inalámbricos, módems, emisoras, videoconferencias, GPS, PDA, cámaras digitales, fax, RDSI, redes de sensores, receptores DRM y en-criptadores de datos entre muchos otros.

En las imágenes y sonido el DSP puede realizar muchas funciones importantes durante el procesamiento de sonido y las imágenes. Permite añadir ecos, soporta la visión artística, el reconocimiento de patrones y la generación de audio, la cancelación de ruido, la cancelación de eco, el cifrado y la síntesis de voz.

4.3. Controladores Digitales de Señales

4.3.1. Concepto de DSC

Microchip ha unido la potencia y posibilidades de sus microcontroladores de 16bits (MCU) con las prestaciones mas interesantes de los DSP para fabricar un nuevo circuito integrado denominado DSC, que intenta ser una respuesta eficaz a las necesidades de las modernas aplicaciones que combinan las funciones típicas de los microcontroladores con las del procesamiento digital de señales de los DSP.

Dada la similitud del DSC con los MCU en cuanto arquitectura y repositorio de instrucciones, los usuarios de las familias de microcontroladores PIC no encuentran dificultades para introducirse en el campo del procesamiento digital de señales.

Estos dispositivos se caracterizan por alcanzar un rendimiento de 40 MIPS e integrar memoria FLASH de alta calidad junto a novedosos recursos hardware, apoyándose en herramientas de desarrollo muy fácil de manejar y manteniendo la compatibilidad de los diversos modelos con encapsulado de diferente patillaje. Los DSC se comercializan en diferentes dispositivos agrupados en dos familias:

1. Familia dsPIC30F
2. Familia dsPIC33F

Nos enfocaremos mas en la segunda familia por ser mas nuevo y fue el que se decidió a utilizar.

4.3.2. Características generales de los dsPIC33F

El voltaje de alimentación admite un rango comprendido entre 2 y 3.6 VDC. El rango de temperaturas es idéntico a los dsPIC30F. El rendimiento máximo alcanza los 40 MIPS cuando el voltaje de alimentación tiene un valor de 3.3 VDC.

Los aspectos mas destacables que incorporan los dsPIC33F son la ampliación en el numero de patas de E/S, la capacidad de la memoria FLASH que alcanza 256KB, se mantiene los periféricos, la disponibilidad del doble de interrupciones, SRAM de hasta 30KB. no dispone de memoria EEPROM.

Tienen un nuevo modo de bajo consumo DOZE y un nuevo Controlador DMA de 8 canales, que sirve para no utilizar a la CPU en la transferencia de datos entre periféricos y memoria. En el modo DOZE todos los osciladores funcionan, pero a muy poca frecuencia, reduciendo de esta forma el consumo.

RECURSOS	RANGO DE VALORES
Memoria de Programa FLASH	Hasta 256KB
Memoria de Datos RAM	Hasta 30KB
Memoria de Datos EEPROM	No Disponible
Temporizadores de 16bits	Hasta 9
Modulo de Captura	Hasta 8 entradas
Modulo Comparador/PWM	Hasta 8 salidas
Conversor A/D de 10bits	2.2Mbps, hasta 32 canales
Conversor A/D de 12bits	1Mbps, hasta 32 canales
UART	Hasta 2 módulos
SPI (8-16bits)	Hasta 2 módulos
IC	Hasta 2 módulos
QEI	1
Interfaz CODEC	1
CAN	Hasta 2 módulos

Cuadro 4.1: Tabla que presenta las características mas importantes de la familia dsPIC33F.

4.3.2.1. Modelos de la familia dsPIC33F

Microchip ha clasificado a los 27 modelos de la familia dsPIC33F que actualmente fabrica y comercializa en dos categorías atendiendo a su aplicación mas apropiada.

1. Dispositivos dsPIC33F de propósito general.
2. Dispositivos dsPIC33F para el control de motores y sistemas de alimentación.

4.3.2.2. Dispositivos dsPIC33F de propósito general

Los dsPIC33F de propósito general son ideales para una amplia gama de avanzadas de MCU de 16 bits embebidos y en concreto, las variantes con interfaces CODEC son idóneas para aplicaciones de tratamiento de la voz y audio. Este grupo esta formado por 15 modelos diferentes dsPIC33F como se muestra en la tabla 4.2.

Modelo	Patitas	FLASH (KB)	SRAM (KB)	A/D 10bit 500kbps	UART	CAN	IC	Patitas E/S max	Cod. Encaps.
j64GP206	64	64	8	1ADC,18ch,1S/H	2	-	1	53	PT
j64GP306	64	64	16	1ADC,18ch,1S/H	2	-	2	53	PT
j64GP706	64	64	16	2ADC,18ch,2S/H	2	2	2	53	PT
j128GP206	64	128	8	1ADC,18ch,1S/H	2	-	2	53	PT
j128GP306	64	128	16	1ADC,18ch,1S/H	2	-	2	53	PT
j128GP706	64	128	16	2ADC,18ch,2S/H	2	2	2	53	PT
j256GP506	64	256	16	1ADC,18ch,1S/H	2	1	2	53	PT
j64GP708	80	64	16	2ADC,18ch,2S/H	2	2	2	69	PT
j128GP708	80	128	16	2ADC,18ch,2S/H	2	2	2	69	PT
j64GP310	100	64	16	1ADC,18ch,1S/H	2	-	2	85	PT,PF
j64GP710	100	64	16	2ADC,18ch,2S/H	2	2	2	85	PT,PF
j128GP310	100	128	16	1ADC,18ch,1S/H	2	-	2	85	PT,PF
j128GP710	100	128	16	2ADC,18ch,2S/H	2	2	2	85	PT,PF
j256GP510	100	256	16	1ADC,18ch,1S/H	2	1	2	85	PT,PF
j256GP710	100	256	30	2ADC,18ch,2S/H	2	2	2	85	PT,PF

Cuadro 4.2: Principales características de los modelos que componen los dsPIC33F de propósito general

dsPIC30F	dsPIC33F
26 módulos disponibles	27 módulos disponibles
-	Controlador de DMA (8 canales)
5 temporizadores	9 temporizadores
mod. trabajo baja energía:IDLE,SLEEP	mod. trabajo baja energía: IDLE,SLEEP,DOZE
Alimentación de 2 a 5.5v	Alimentación de 2 a 3.6v
Rendimiento: 30MIPS a 4.5 o 5.5v	Rendimiento: 40MIPS a 3.3v
Pocos vectores de interrupción(62)	Muchos vectores de interrupción (118)
Memoria EEPROM	-
Memoria FLASH de 144KB	Memoria FLASH de 256KB
Memoria SRAM de 8KB	Memoria SRAM de 20KB
Abundantes periféricos	Mas periféricos

Cuadro 4.3: Diferencias destacables entre las familias dsPIC30F y dsPIC3F

4.3.2.3. Dispositivos dsPIC33F de control de motores y sistemas de alimentación

la categoría de control de motores ofrece una gran variedad de aplicaciones relacionadas con el gobierno de motores, como los de inducción, de fase simple o trifásicos y los de corriente continua. También son muy apropiados para la gestión de los sistemas de alimentación ininterrumpibles, conversores, módulos para la corrección del factor de potencia, telecomunicaciones u otros equipamiento industrial.

4.3.2.4. Diferencias entre dsPIC30F y dsPIC33F

A pesar de ser muy similares, existen algunas diferencias apreciables entre ambos DSC. El rango de valores soportado por cada uno es diferente, así como el voltaje óptimo para su mejor rendimiento. Los dsPIC30F tienen como tensión nominal 5VDC y los dsPIC33F 3.3VDC. También difieren en las patas de E/S y la memoria FLASH que alcanza 144KB en los 30F y 256KB en los 33F. Los 33F carecen de EEPROM, pero su SRAM puede llegar a 30KB, mientras que en los 30F no superan los 8KB. Finalmente, los dsPIC33F disponen del doble de interrupciones y un Controlador de DMA. En la tabla 4.3 se representan las principales diferencias existentes entre las familias del dsPIC30F y dsPIC33F:

4.3.3. Herramientas de desarrollo software

Para mantener la continuidad a los diseñadores con PIC MCU y facilitar su transición a los DSC, Microchip proporciona un paquete de herramientas software para estos últimos muy similar a las utilizadas con los microcontroladores clásicos.

Todo el software se cimenta en el conocido entorno integral MPLAB IDE, que contiene un compilador de C, MPLAB C30 y el inicializador Gráfico de Pro-

gramas MPLAB VDI. La mayor parte de estas herramientas pueden recogerse libremente desde el sitio de Microchip en internet.

El MPLAB IDE gestiona todas las etapas del diseño software de un proyecto y se adapta a las herramientas hardware de Microchip y otras segundas fuentes. Corre en un sistema operativo Windows 7, Vista, XP, 2000, ME, 98SE y NT.

Para el desarrollo y análisis de algoritmos DSP se dispone de la herramienta dsPICworks que entre sus posibilidades es capaz de diseñar filtros, generar señales, proporcionar operaciones para procesar señales aritméticas y digitales, tiene capacidad de mostrar y cuantificar elementos de una, dos y tres dimensiones.

Una extensa oferta de librerías,están a disposición de los usuarios para facilitar el diseño de proyectos, tales como los que soportan algoritmos DSP, de reconocimiento de voz, driver para periféricos, CAN, protocolos TCP/IP, MÓDEM embebidos,etc.

4.3.4. Herramientas de desarrollo hardware

La arquitectura abierta de los productos de Microchip y su generosa política comercial han conseguido inundar el mercado de interesantes herramientas hardware y software de segundas fuentes que optimizan el desarrollo de las aplicaciones con los productos de este fabricante.

Para agilizar el entrenamiento y aprendizaje de los usuarios en el manejo de los dsPIC, Microchip ha desarrollado un conjunto de tarjetas especificas para desarrollar proyectos en áreas como el control de motores, la conectividad.etc.

4.3.5. Aplicaciones recomendadas

El fabricante ha clasificado en 6 áreas las posibles aplicaciones de los DSC.

- Control de motores.

Bombas industriales, lavadoras, aspiradoras, equipos de gimnasia, tensores de cinturones de seguridad, calefacción, ventilación, aire acondicionado, refrigeración, herramientas eléctricas, apertura de puertas, control de estabilidad.

- Control de sensores

Sensores de torsión, de presión, de vibraciones, de golpes, de lluvia, de fallos en la red eléctrica, de rotura de cristales, sensores químicos y de gases, detección PIR avanzada 2-D.

- Automoción

Control de potencia, control de estabilidad y balanceo, caja de cambios y embrague electrónico, Dirección hidráulica asistida eléctricamente, cancelación de ruido en el habitáculo, airbag, control de ignición, sensor de presencia de ocupantes, control del combustible.

- Conectividad en Internet

Monitorización remota, contadores automáticos de agua y gas, Indeterminación médica remota, sistemas de seguridad, maquinas expendedoras, diagnostico industrial remoto.

- Audio

Reconocimiento de voz, altavoces telefónicos, redes distribuidas de megafonía, efectos especiales en instrumentos musicales, cancelación de ruido, reproductor de sonido digital, auriculares para la eliminación de ruidos, puertos de comunicación para manos libres, micrófono activado por voz.

- Gestión y monitorización de sistemas de alimentación

convertidores AC-DC, convertidores DC-AC, fuentes de alimentación ininterrumpibles(UPS), inversores, vehículos eléctricos, corrección del factor de potencia, monitorización de la alimentación en servidores, gestión de la alimentación y ahorro de consumo, detectores de fallos.

4.4. dsPIC Starter kit

En primer lugar contara un disco de instalación y la tarjeta starter kit que se puede observar en la figura 4.1, además de un cable USB mini-B. Este cable es la interfaz entre Microchip MPLAB IDE y provee la energía para la tarjeta starter kit.

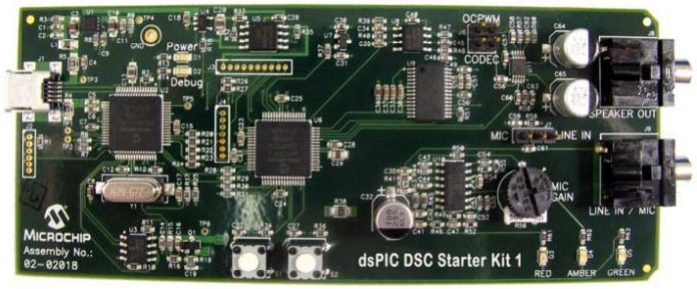

Figura 4.1: Tarjeta dsPIC starter kit

La tarjeta tiene tres secciones como se puede ver en la figura 4.2, la primera parte es la sección de debug y proporción que trae conexión al puerto USB en la PC. La segunda parte es la sección de dsPIC DSC y por ultimo la sección de audio la cuál contienen los circuitos que so requeridos para la captura de la señal de audio

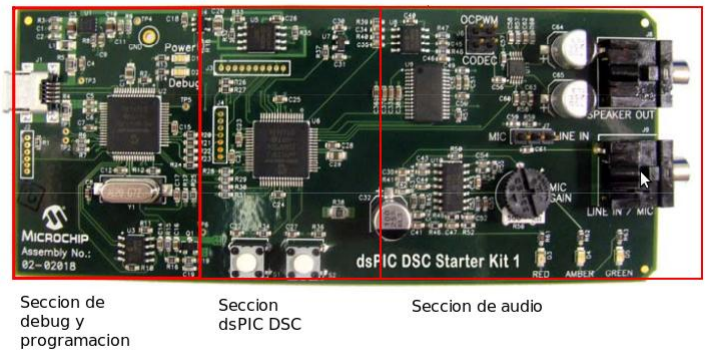

Figura 4.2: Secciones principales del dsPIC

La tarjeta viene equipada con:

1. Un dsPIC33FJ256GP506 con DSC de 16 bits además de 256KBytes de memoria flash, 16KBytes de RAM. Este dispositivo sirve como el dispositivo computacional de la tarjeta.
2. Un PIC18F67J50 microcontrolador que sirve como debuger y programador del dsPIC, también provee una máxima velocidad para la interfaz USB requerida para conectar la tarjeta a la computadora.
3. Un regulador de 3.3v que alimenta a la tarjeta starter kit vía USB.
4. Memoria flash serie de 4MBits para guardar el audio y otros datos.
5. Un Codec de audio de alta fidelidad. Este Codec es capaz de procesar arriba de 32 bits de datos de audio en una frecuencia de muestreo de arriba de 40khz.
6. Filtros análogos para la implementación de reconocimiento de bajo-costo.
7. Un circuito de pre-amplificación y un filtro anti-aliasing (alias es efecto que causa que las señales continuas distintas se tornen indistinguibles cuando se muestrean digitalmente) la cual provee el limite de banda limitando la señal micrófono antes de ser procesada por el ADC de 12 bits.
8. Un amplificador de Audífonos con 100 mwatts provee la amplificación requerida de la señal de audio para manejo de los audífonos. El volumen del amplificador puede ser controlado por la tarjeta dsPIC.
9. Tres indicadores LED y dos switch que están disponibles.
10. Una conexión USB entre la tarjeta y la PC.
11. Un conector para los audífonos.

12. Una conexión para un micrófono o una señal de audio externa.

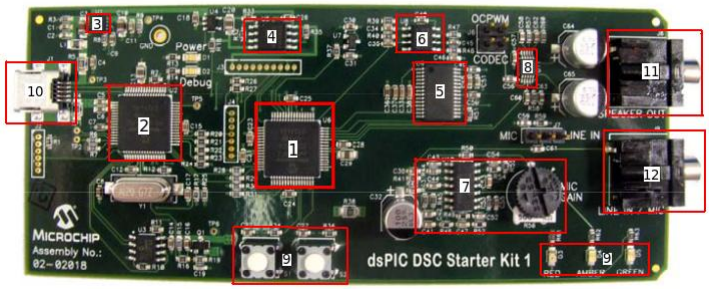

Figura 4.3: Partes del dsPIC starter kit

En la figura 4.4 se muestra la conexión de la tarjeta se muestra en la figura, en la computadora al hacer la instalación se instala el compilador MPLAB IDE. La conexión USB provee la comunicación y alimentación hacia la tarjeta con la PC.

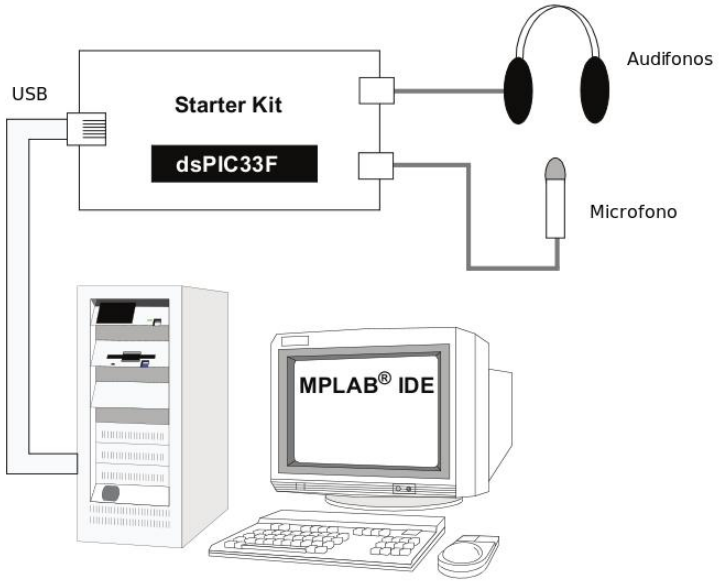

Figura 4.4: Conexión con la PC del dsPIC

4.4.1. Driver WM8510 Codec

El driver WM8510 Codec configura el Codec de audio WM8510 y provee una interfaz para leer y escribir datos de audio para el Codec. Este driver es imple-

mentado en WM8510CodecDrv.c y la interfaz esta definida en WM8510CodecDrv.h el driver usa el modulo DCI en el modulo del dispositivo dsPIC33F para procesar datos y el modulo I^2C^{TM}como un bus de control de Codec.

4.4.2. G.711 Codificador y Decodificador

El G.711 codificador y decodificador implementa el algoritmo ITU-T G.711 de compresión de voz. Este algoritmo es un ejemplo de codificador de forma de onda y provee una compresión de 2:1. El algoritmo es implementado en G711.s y su interfaz esta definida por G711.h.

La primera generación de DSC, denominada dsPIC30F, constituye un intento por parte de microchip de facilitar el acercamiento al mundo del procesamiento digital de señales a sus usuarios de MCU de 8 y 16 bits. Posteriormente la segunda generación de DSC, llamada dsPIC33F, ha potenciado las capacidades, el número de periféricos y el rendimiento, permitiendo acceder a campos de aplicaciones mas complejas. Los dsPIC33F de propósito general son ideales para una amplia gama de avanzadas de MCU de 16 bits embebidos y en concreto, las variantes con interfaces CODEC son idóneos para aplicaciones de tratamiento de voz y audio.

Aquí se muestra una figura 4.5 de la señal de voz antes de ser filtrada por el algoritmo, a través de las herramientas del dsPIC starter kit.

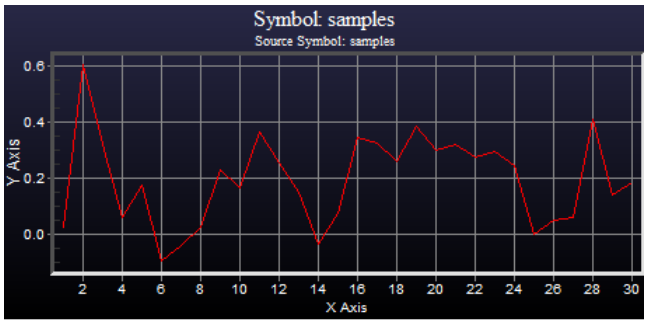

Figura 4.5: Señal de entrada

La otra figura 4.6 es después de ser aplicado el algoritmo en el dispositivo y se observa como la señal se reducen los valores espurios.

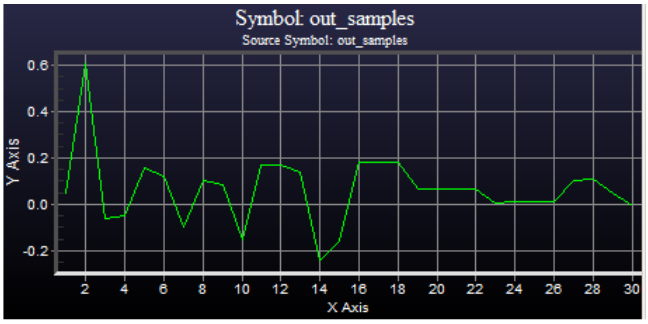

Figura 4.6: señal de salida del filtro

Capítulo 5

Filtro utilizado

El algoritmo localmente adaptativo para la mejora de la señal de voz propuesto en [31] es el siguiente:

1. Capturar la señal de voz, se obtiene un vector $f[t]$ de tamaño N.

2. Iniciar el ciclo de procesamiento, $k = 1$.

3. Crear ventana deslizante mv_k con S elementos, alrededor del elemento central $f[k]$, los valores de t deben satisfacer que:

$$mv_k = \left\{ f[t] \; k - \frac{S-1}{2} \leq t \leq k + \frac{S-1}{2} \right\}$$

   (S impar).

4. Crear renglón variacional V_{rk} y establecer el valor de r_k (rango del valor $f[k]$). El renglón variacional se obtiene de manera ascendente los elementos de la ventana deslizante mv_k.Crear vecindario adaptativo Amv_k utilizando el criterio EV, a partir del renglón variacional. En este punto el algoritmo necesita identificar dos posibles casos:

   a) el elemento central de la ventana deslizante $Amv_k(r)$ es un elemento solo afectado por ruido aditivo.

   b) El elemento $Amv_k(r)$ es un elemento afectado por el ruido impulsivo.

5. Para cada uno de estos casos, el algoritmo realiza una acción diferente. Por tal motivo, es necesario un procedimiento para reconocer los casos (a) y (b). Notemos que la ventana adaptativa Amv_k se construye en relación al valor del elemento central de la ventana deslizante $mv_k((S-1)/2)$. Por lo tanto, si este valor central es una muestra impulsiva, entonces el tamaño de la ventana adaptativa resultante sera pequeño, impulsivas dentro de la ventana deslizante es minoría. Para identificar este efecto, definimos un parámetro $L = pS$ como un valor de umbral para detectar los casos (a)

y (b). Aquí, p es la probabilidad de ocurrencia del ruido impulsivo y S es el tamaño de la ventana deslizante mv_k. De esta manera, es fácil darse cuenta que cuando $(S_A < L) = falso$, entonces el caso mas probable es (a). De otra manera cuando $(S_A < L) = verdadero$, entonces el caso mas probable es (b),

6. Determinar el tamaño del vecindario adaptativo S_A.

7. Si $S_A \leq L$ ir a 11, en caso contrario, significa que el elemento central $f[k]$ es espurio (corrupto por ruido impulsivo) por lo que hay que remplazar su valor a partir de un nuevo vecindario adaptativo.

8. La nueva ventana deslizante $mv_k^{'}$ se obtiene como resultado del complemento de la intersección entre los conjuntos mv_k y Amv_k.

9. Definir un nuevo renglón variacional $V_{rk}^{'}$ y establecer el nuevo valor de $r_k^{'}$. Si el tamaño de $V_{rk}^{'}$ es impar, $f[k]$ sera igual al elemento central de $V_{rk}^{'}$; de otra manera, sera igual a la media aritmética entre los dos valores centrales de $V_{rk}^{'}$.

10. Crear vecindario adaptativo Amv_k utilizando el criterio EV.

11. El valor inferido para $f\,[k]$= promedio ponderado de los elementos de Amv_k (En la versión 3 del filtro es el valor promedio de los elementos deAmv_k).

12. $k = k + 1$.

13. Si $k \prec N$, ir al paso 3; de otra forma, terminar.

Con este algoritmo se intenta corregir la señal afectada y eliminar de ella las variaciones debidas a la mezcla de ruidos aditivos al sensor y ruido tipo impulsivo. La diferencia entre el filtro propuesto en [31] y el presentado en esta sección consiste en la utilización del promedio ponderado para inferir el valor de la señal resultante (paso 11). En lo que resta del artículo, al filtro propuesto en [31] se le denominará Algoritmo Versión 1, mientras que al filtro presentado esta sección se le nombrará Algoritmo Versión 3.

5.1. Ejemplo del algoritmo

Veremos los casos que pueden ocurrir en el algoritmo y su forma de comportarse ante tales situaciones.

Parámetros del algoritmo:

- S=7
- EV=2
- L=floor(S/6)=2

Para el caso 1:

1. Vector resultante de tamaño S: $mv_k = \{3, 0, 3, 1, -1, 2, 4\}$
2. Vector variacional $vr_k = \{-1, 0, 1, 2, 3, 3, 4\}$
3. Vecindario adaptativo $Amv_k = \{-1, 0, 1, 2, 3, 3\}$ utilizando el criterio EV
4. Tamaño de vecindario adaptativo $S_A = 6$
5. Debido a que S_A es menor que L, Y_k = 1,3333 (Media aritmética de los valores del vecindario adaptativo).

Para el caso 2:

1. Vector resultante de tamaño S: $mv_k = \{3, 0, 3, 9, 8, 2, 4\}$
2. Vector variacional $vr_k = \{0, 2, 3, 3, 4, 8, 9\}$
3. Vecindario adaptativo $Amv_k = \{8, 9\}$ utilizando el criterio EV
4. Tamaño del vecindario adaptativo $S_A = 2$
5. Debido a que S_A es mayor que L, se considera que el valor $f[k] = \{9\}$ es espurio (corrupto)
6. Nueva ventana deslizante $mv_k^{'} = \{3, 0, 3, 2, 4\}$
7. Nuevo vector variacional $vr_k^{'} = \{0, 2, 3, 3, 4\}$
8. Nuevo vecindario adaptativo $Amv_k^{'} = \{2, 3, 3, 4\}$ utilizando el criterio EV
9. $Y_k = 3$ (Media aritmética de los valores del vecindario adaptativo).

En esta tesis se usaron dos diferentes algoritmos en el primero se usa el promedio aritmético llamado V1 y otro que calculaba el promedio ponderado con el nombre de V3.

Capítulo 6

Ejemplo de la aplicación de la metodología

Como se explico en el capitulo 3 para determinar los valores de las dos versiones de los algoritmos que proporcionan un mejor rendimiento y para comparar el desempeño de ambos filtros, se utilizo la metodología propuesta.

Por tal motivo, se aplicaron los filtros a una misma señal corrompida con ruido gaussiano, variando en cada ecuación los valores de los parámetros {S, EV, L}. Para la señal generada se calcularon los valores de MSE y MAE, comparando la señal filtrada con la señal original (libre de ruido). Los valores utilizados para S para los experimentos fueron valores nones los cuales fueron los siguientes:

3,9,15,21,27,33,39,45,51,57,63,69,75,81,87,93,99.

Los valores utilizados para los vecindarios EV en los experimentos fueron:

$$0{,}5 \cdot std(Y_{noisy}), 0{,}6 \cdot std(Y_{noisy}), ..., 3 \cdot std(Y_{noisy})$$

Estos van desde el 0.5 hasta el 3 aumentando en 0.1 en cada iteración. Mientras que el valor de L dependiente del valor S y el cual esta dado por:

$$floor(\frac{S}{6})$$

En esta sección se muestran los gráficas y las tablas obtenidas, primero las gráficas aplicando los valores que se utilizaron para las pruebas.

En las figuras 6.1,6.2,6.3,6.4se muestran las gráficas obtenidas de la versión 1 del filtro cuando el valor de EV fue fijo y se fue variando el tamaño de la ventana deslizante S, como puede observarse, el valor de MAE es menor cuando el valor de S se encuentra entre 27 y 45.

En las figuras 6.5,6.6,6.7,6.8 se observan las gráficas de la versión 3 del filtro cuando el valor de EV es fijo y se fue variando el tamaño de S, como se observa, el valor de MAE es menor cuando para los valores de S están en los rangos de [15,21] en 1.1,1.2 de EV se tiene un error de MAE bajo, así como para 0.9 de EV cuando S es 63. y para S=51 cuando el valor de EV es [0.9,1.4-1.8]

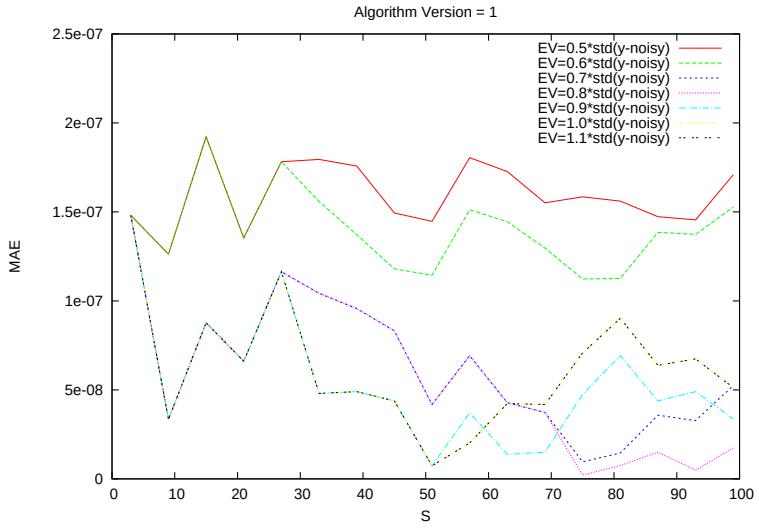

Figura 6.1: MAE EV fijo para la versión 1(a)

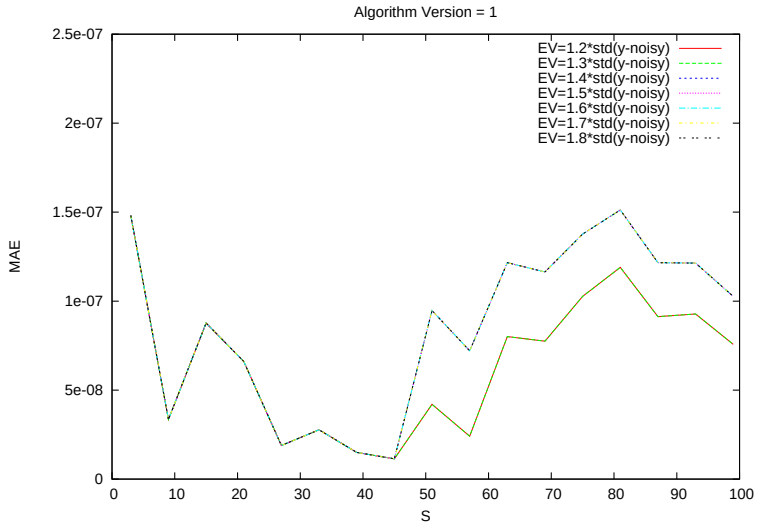

Figura 6.2: MAE EV fijo para la versión 1(b)

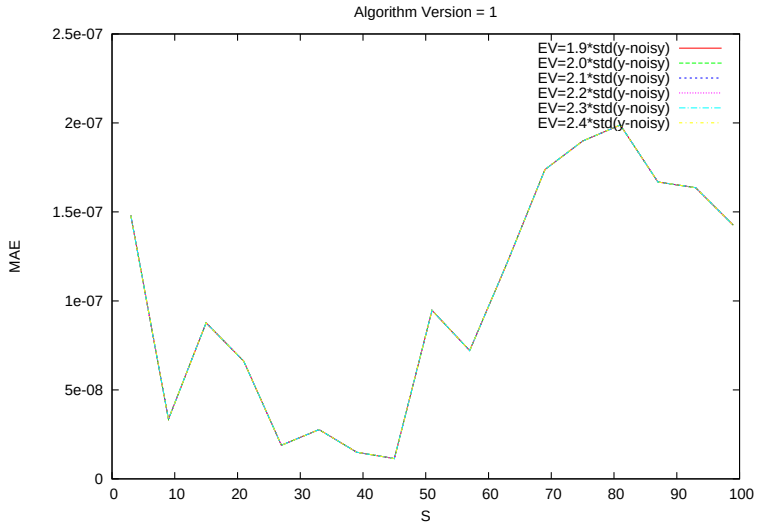

Figura 6.3: MAE EV fijo para la versión 1(c)

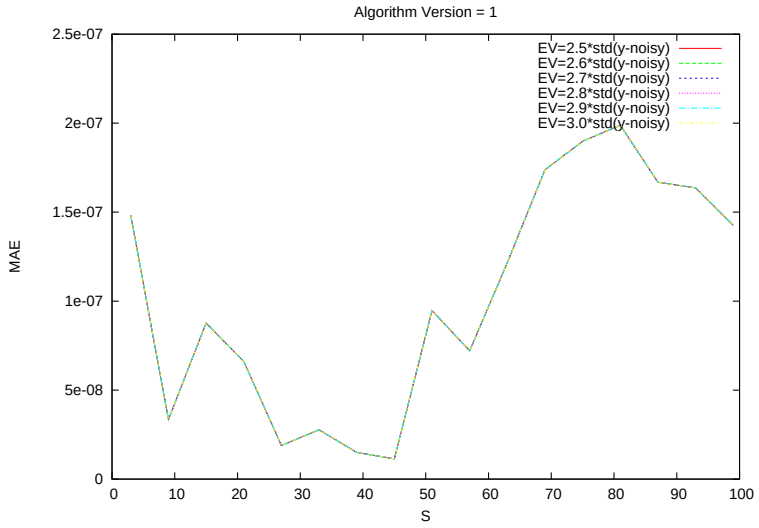

Figura 6.4: MAE EV fijo para la versión 1(d)

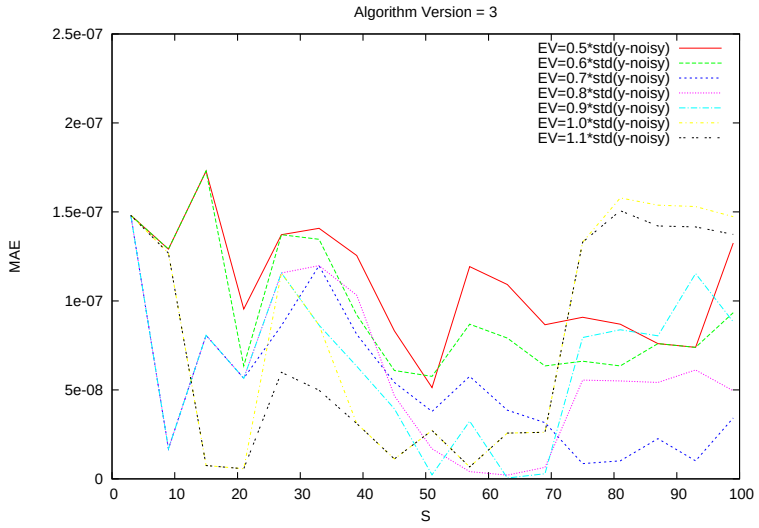

Figura 6.5: MAE EV fijo para la versión 3(a)

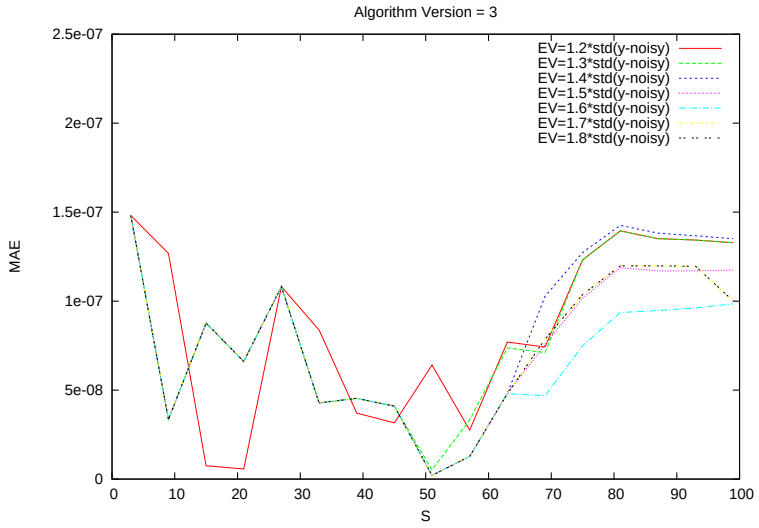

Figura 6.6: MAE EV fijo para la versión 3(b)

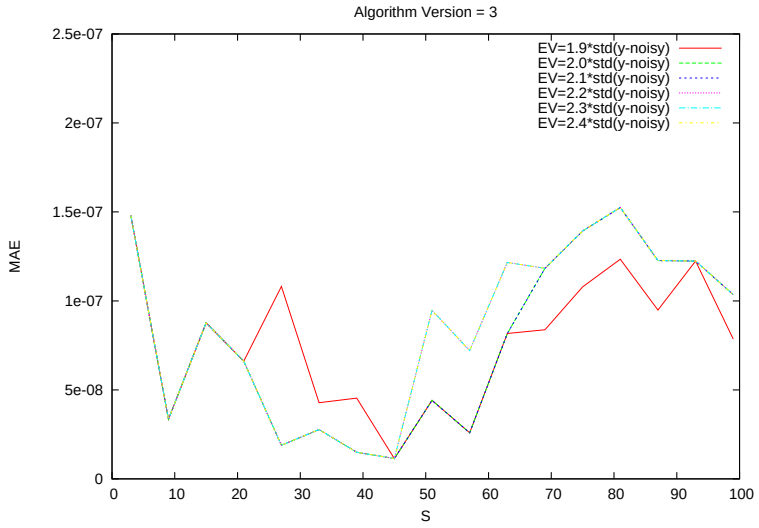

Figura 6.7: MAE EV fijo para la versión 3(c)

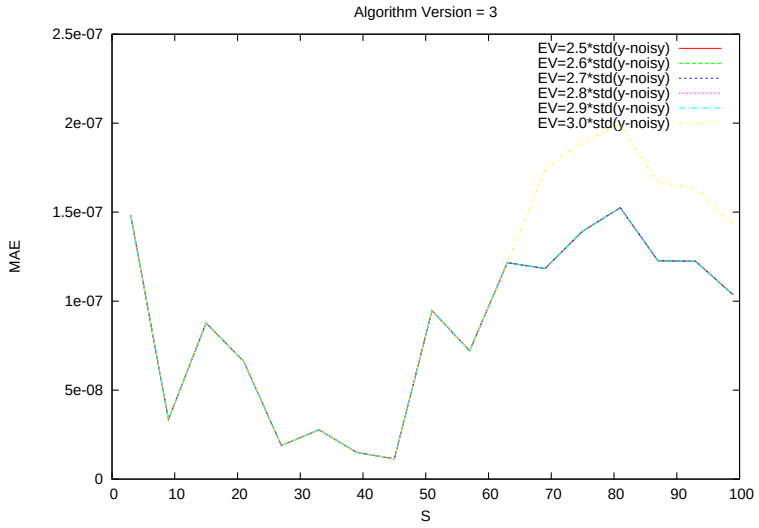

Figura 6.8: MAE EV fijo para la versión 3(d)

La Tabla 6.1de la versión 1 del filtro [31]con la métrica MAE muestran como para valores de EV[0.8] de S =81 y 93 mantiene un error bajo, así como para los valores de S= 75 también se obtuvo menor error, el menor de la tabla.

EV	S	MAE
0.7	75	0.0000000096
0.8	75	0.0000000021
0.8	81	0.0000000075
0.8	93	0.0000000049
1.2-3	45	0.0000000114

Table 6.1: MAE del filtro V1

La tabla 6.2obtenidas en la V3 con la métrica MAE para valores pequeños de EV[0.9] los valores de S son los siguientes 51, 63 y 69 en estos se muestra un error bajo. En el caso de S=63 es donde se observa el menor error en la tabla, al igual que en los valores de EV[1.4-1.8] que con valores de S=51 el error que se obtiene es también bajo.

EV	S	MAE
0.7	81	0.0000000101
0.7	93	0.0000000101
0.8	57	0.0000000041
0.8	63	0.0000000021
0.8	69	0.0000000065
0.9	51	0.0000000023
0.9	63	0.0000000005
0.9	69	0.0000000029
1-1.1	45	0.0000000112
1-1.1	57	0.0000000067
1.3	51	0.0000000052
1.4-1.8	51	0.0000000021
2-3	45	0.0000000114

Cuadro 6.2: MAE del filtro V3

En las figuras 6.9,6.10,6.11,6.12se muestran las gráficas de la versión 1 cuando se fue variando el valor de EV y el valor de S se dejo fijo se observa que los mejores resultados pero desde otra perspectiva se observa que para 0.8 del valor de EV y para la S con 81 y 93 tiene un menor error absoluto promedio

En las figuras 6.13,6.14,6.15,6.16se muestra las gráficas de la versión 3 en estas se fijo el valor S y el valor EV se fue variando

Para el caso de las métricas MSE para la versión 1 cuando EV es fijo en las figuras 6.176.186.196.20se observa como los valores varían bastante.

Mientras que para la versión 3 con la métrica MSE cuando EV es fijo se muestra que los valores no siguen un patrón,

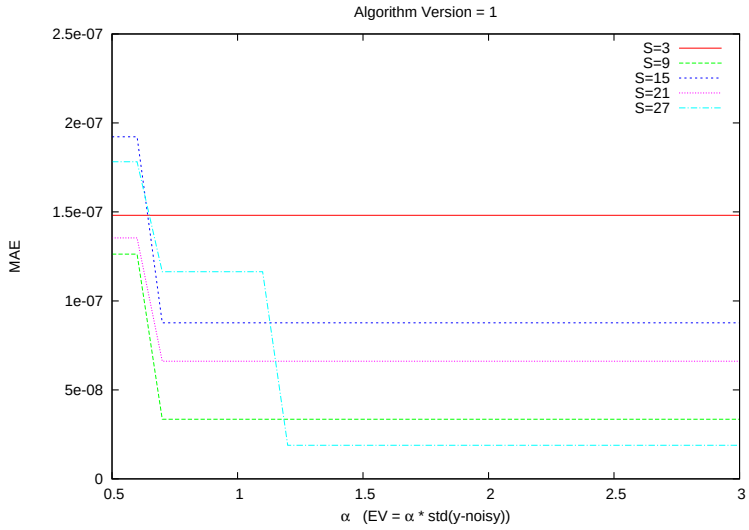

Figura 6.9: MAE S fijo para la versión 1(a)

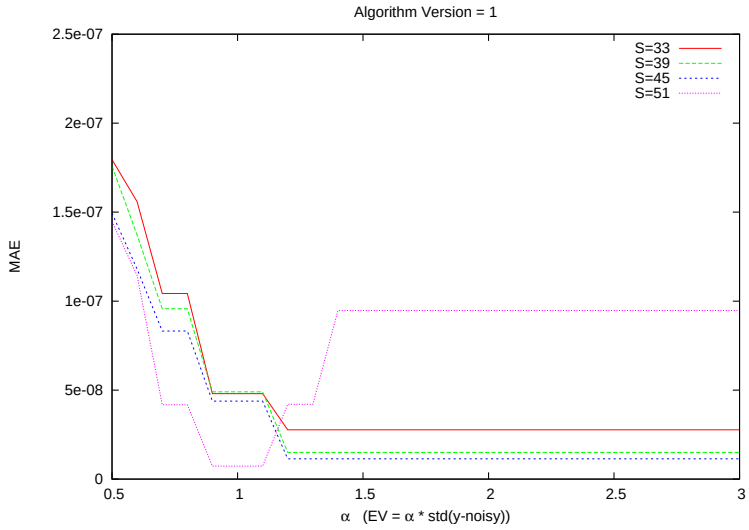

Figura 6.10: MAE S fijo para la versión 1(b)

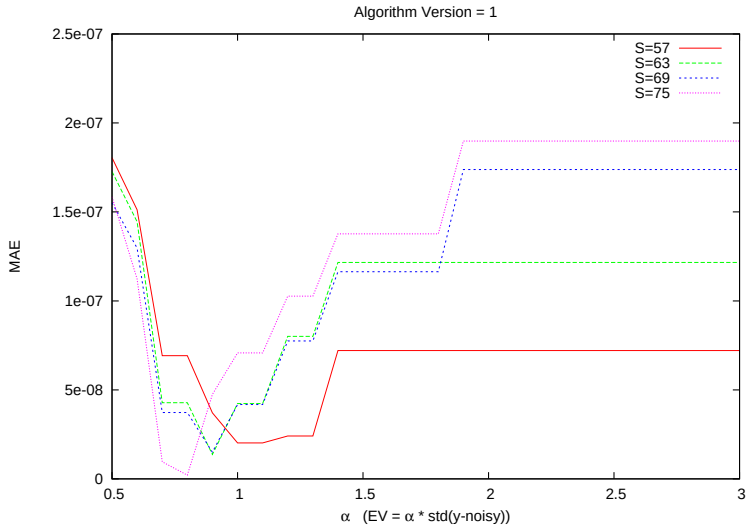

Figura 6.11: MAE S fijo para la versión 1(c)

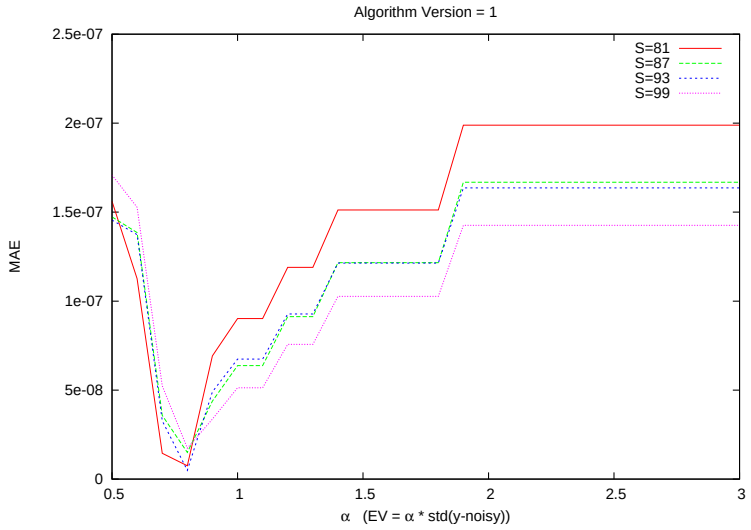

Figura 6.12: MAE S fijo para la versión 1(d)

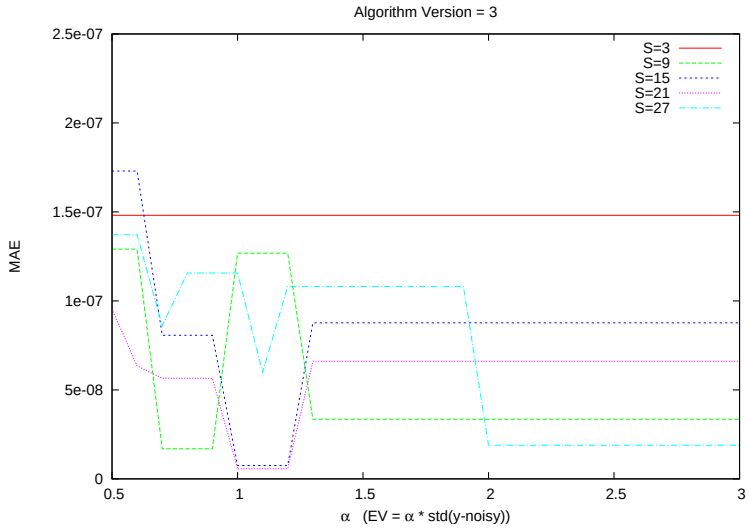

Figura 6.13: MAE S fijo para la versión 3(a)

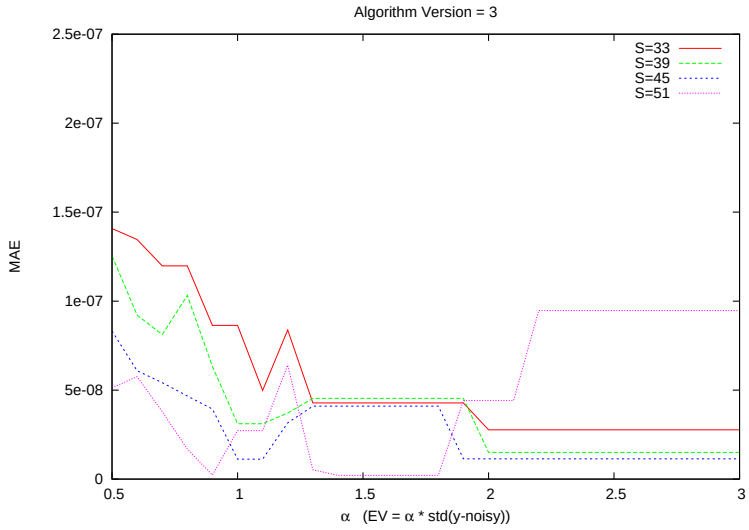

Figura 6.14: MAE S fijo para la versión 3(b)

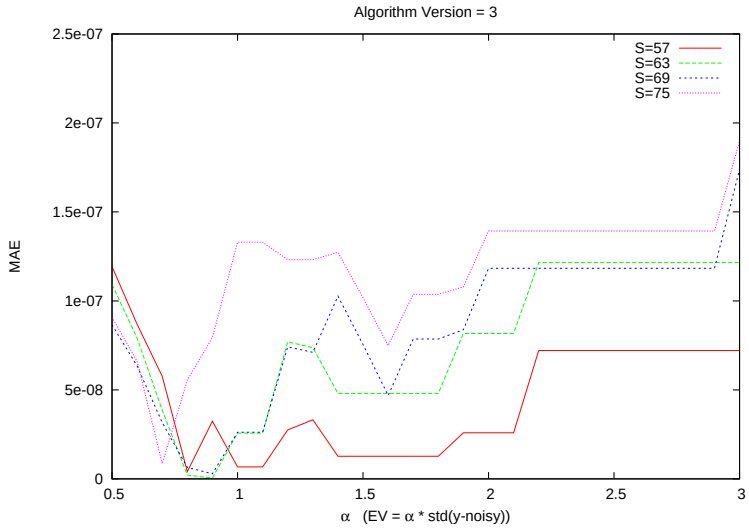

Figura 6.15: MAE S fijo para la versión 3(c)

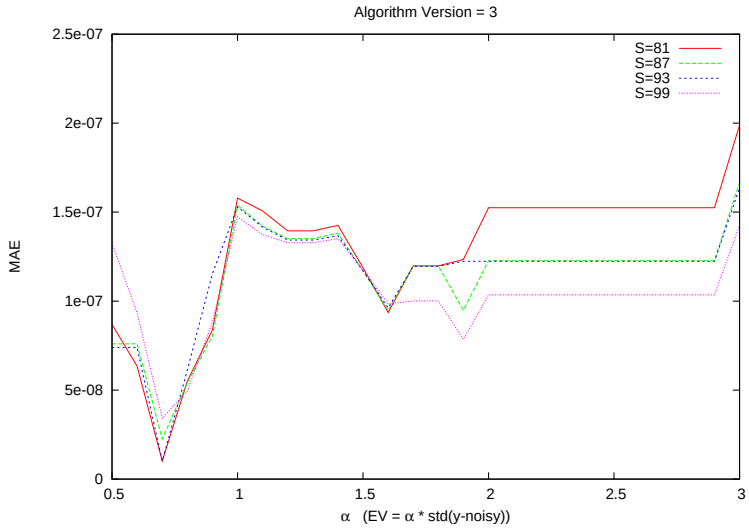

Figura 6.16: MAE S fijo para la versión 3(d)

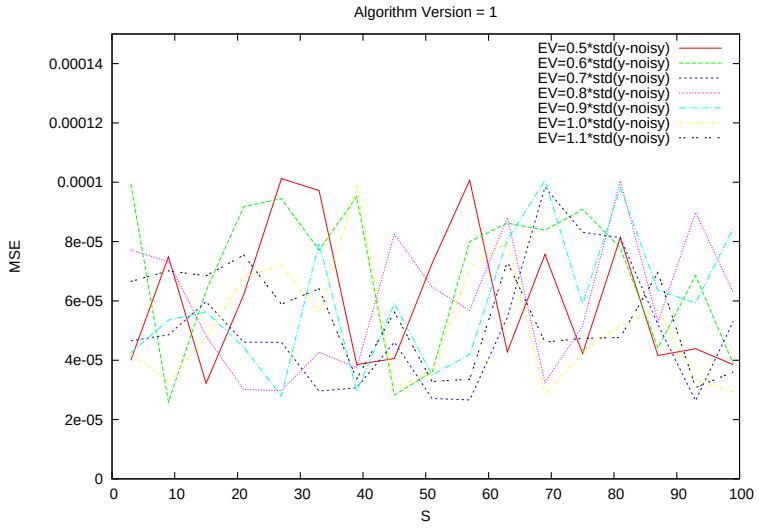

Figure 6.17: MSE EV fijo para la versión 1 (a)

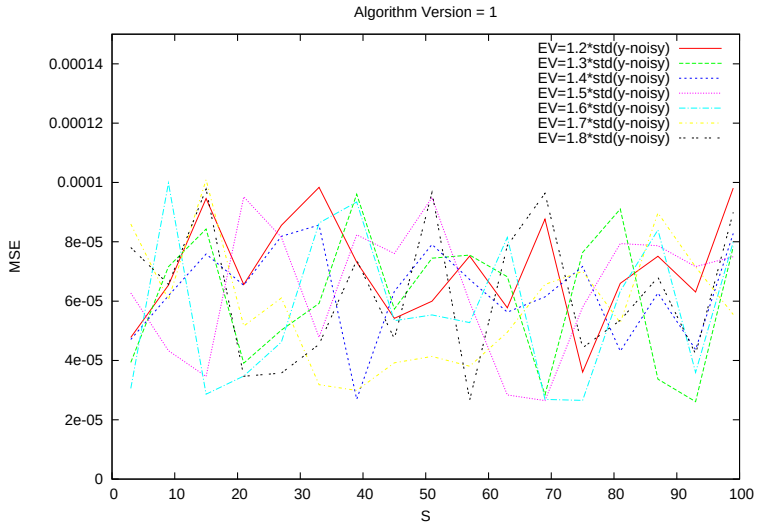

Figura 6.18: MSE EV fijo para la versión 1 (b)

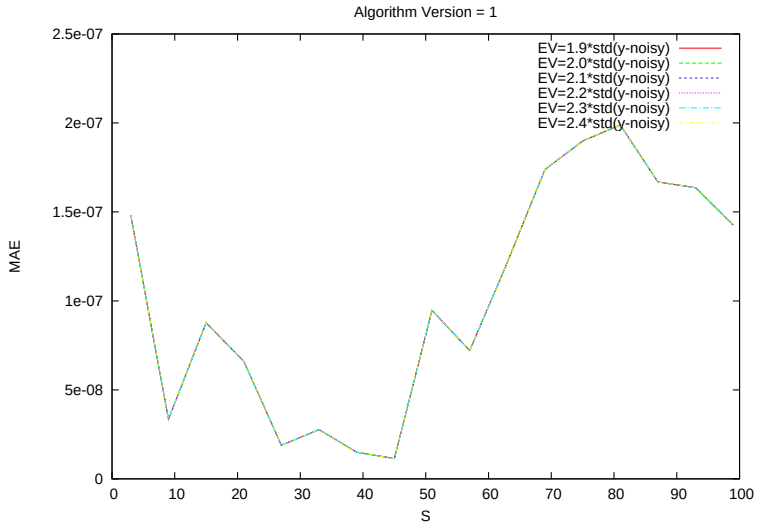

Figura 6.19: MSE EV fijo para la versión 1 (c)

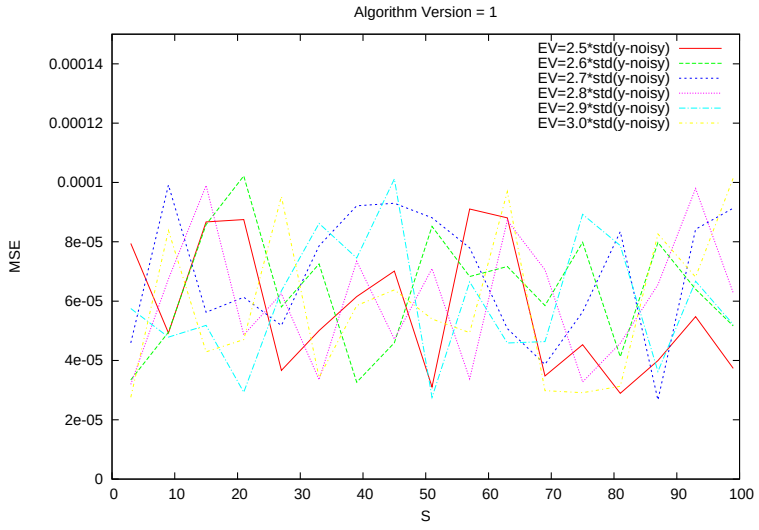

Figura 6.20: MSE EV fijo para la versión 1 (d)

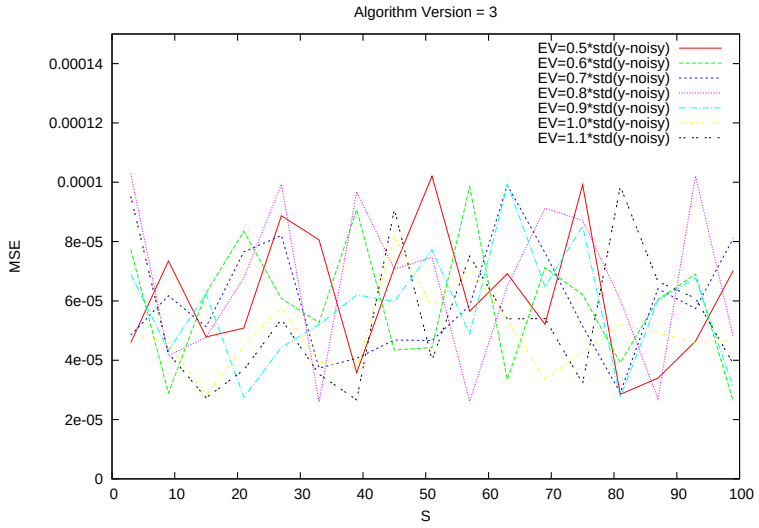

Figura 6.21: MSE EV fijo para la versión 3 (a)

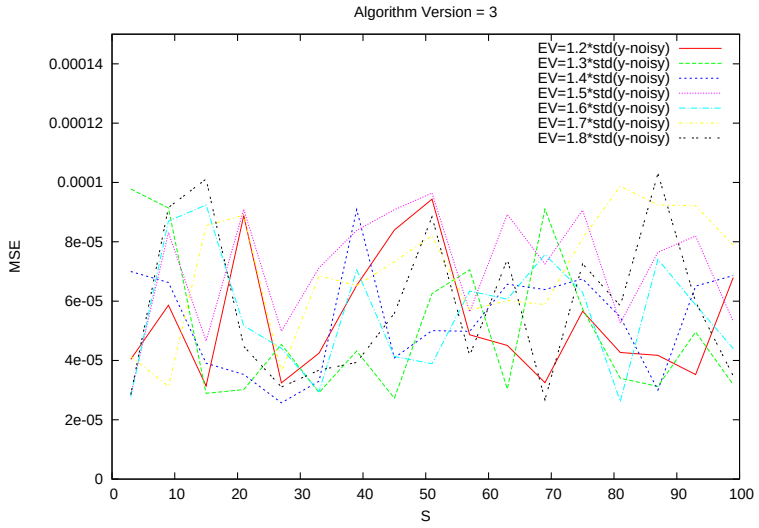

Figura 6.22: MSE EV fijo para la versión 3 (b)

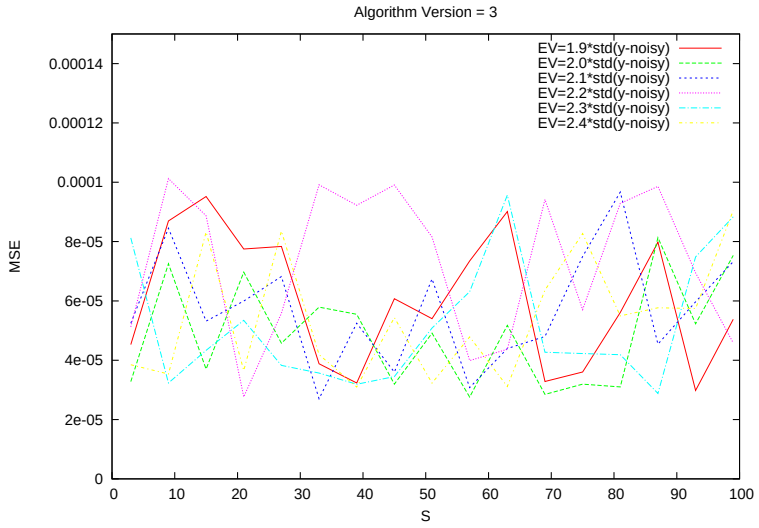

Figura 6.23: MSE EV fijo para la versión 3 (c)

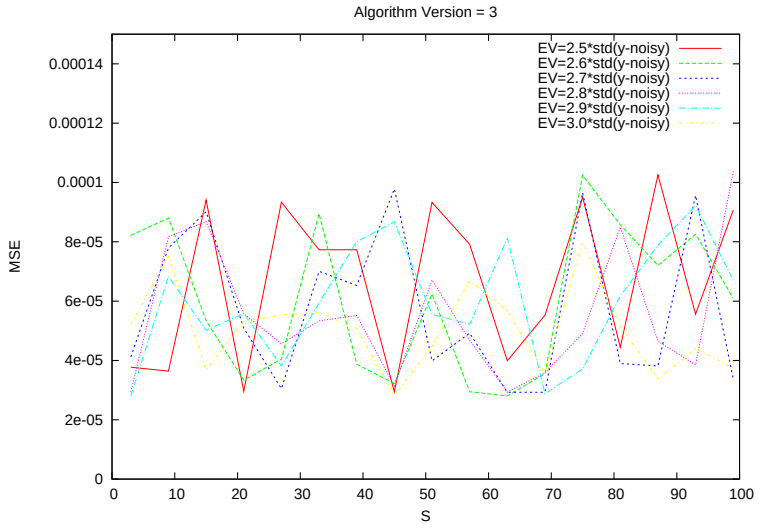

Figura 6.24: MSE EV fijo para la versión 3 (d)

	Algoritmo V1	Algoritmo V2
EV	0.8	0.9
S	75	63

Table 6.5: Tabla con mejores resultados

Para el caso de las métricas MSE en la V1 del filtro [31] se observa en la tabla 6.3que el menor error se encuentra cuando EV[0.6] con S=9.

EV	S	MSE
0.6	9	0.0000259221
0.7	57	0.0000266239
1.3	93	0.0000261015
1.5	69	0.0000264596
1.6	75	0.0000265517
2	3	0.0000263975

Cuadro 6.3: MSE del filtro V1

Mientras que en la V3 del filtro se muestra en la tabla hay dos valores con menor error; cuando EV[0.8] con S=33 es uno de ellos y el menor error se obtiene en EV[1.4] con S=27.

EV	S	MSE
0.6	99	0.0000260631
0.8	33	0.0000258780
1.1	39	0.0000266597
1.4	27	0.0000257027
1.6	81	0.0000261866

Cuadro 6.4: MSE del filtro V3

En la tabla 6.5se muestran con que valores se obtuvieron los mejores resultados para cada algoritmo.

Capítulo 7

Conclusiones y trabajo futuro

Para procesar señales digitales que han sido alteradas por procesos externos una gran cantidad de filtros han sido propuestos. Entre ellos, los filtros localmente adaptativos han mostrado un muy buen desempeño. Sin embargo, este tipo de filtros utilizan una gran cantidad de parámetros, por lo que es importante contar con una metodología que permita identificar sus valores óptimos rápidamente.

Se propone una metodología para evaluar he implementar filtros digitales de señales. Se utilizo un algoritmo localmente adaptativo para demostrar su utilidad. Además, se mostró que esta metodología se puede utilizar para comparar algoritmos mediante las métricas de error utilizadas y determinar cual brinda un mejor desempeño.

Se mostró que esta metodología se puede utilizar para comparar algoritmos mediante las métricas de error utilizadas y determinar cual brinda un mejor desempeño. Se comprobó el uso de la metodología implementando un filtro digital de señales en un dsPIC.

Bibliografía

[1] S. Song S. R. F. Sims A. Mahalanobis, B. V. K. Vijaya Kumar and J. F. Epperson. Unconstrained correlation filters. *Appl. Opt*, 33:3751–3759, 1994.

[2] E. Abreu, M. Lighstone, S. Mitra, and K. Arakawa. A new efficient approach for the removal of impulse noise from corrupted images. *IEEE Transactions on Image Processing*, pages 1012–1025, 1996.

[3] G.R. Arce. Multistage order statistic filters for image sequence processing. *IEEE Trans. Signal Process.*, 39:1146–1162, 1991.

[4] G.R. Arce and M.P. McLoughlin. Theoretical analysis of the max/median filter. *IEEE Trans. Acoust. Speech Signal Process*, 35:60–69, 1987.

[5] P. K. Jaakko Astola. *Fundamental of Nonlinear Digital Filtering.* CRC-Press, 1997.

[6] J. Benesty, S. Makino, and J. Chen. *Speech Enhancement.* springer, 2005.

[7] S. F. Boll. Supression of acoustic noise in speech using spectral substraccion. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 2:113–120, 1979.

[8] R. L. Burden and J. D. Faires. *Numerical Analysis.* Thomson, 2005.

[9] E.D. Dougherty and J.Astola. *Introduction to Nonlinear Image Processing.* Bellingham, WA:SPIE, 1994.

[10] M. Gabbouj E.J. Coyle, J.H. Lin. Optimal stack filtering and the estimation and structural approach to image processing. *IEEE Trans. Acoust. Speech Signal Process.*, 37:2037–2066, 1989.

[11] M. Tian G. R. Arce. Order statistics filter banks. *IEEE Signal Process.*, 5(6):827–837, 1996.

[12] J. Ganssle. *The Art of Designing Embedded Systems.* Newnes, may 2008.

[13] S. J. Godsill and P. J. W. Rayner. *Digital audio restoration a statistical model basad approach.* springer, 1998.

[14] E. Hansler and G. Schmid. *Speech and Audio Processing in Adverse Environments*. springer, 2008.

[15] Hocking. *The analysis of linear models*. Brooks/Cole, 1985.

[16] I. Pitas I. and A.N. Venetsanopoulos. *Nonlinear digital filters. Principles and applications*. Kluwer Academic Publishers, Boston., 1990.

[17] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1988.

[18] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME- Journal of basic Engineering*, 82:35–45, March 1960.

[19] R. Kamal. *Embedded System: Architecture, Programming and Design*. Tata Mcgraw-hill Publishing Company Limited, March 2009.

[20] V. Kober. Fast algorithms for the computation of sliding discrete hartley transforms. *IEEE Transactions on Signal Processing*, 55:6, 2007.

[21] V. Kober, M. Mozerov, and J. Alvarez-Borrego. Nonlinear filters with spatially connected neighborhoods. *Optical Engineering*, pages 971–983, 1986.

[22] V. Kober, M. Mozerov, J. Alvarez-Borrego, and I. A. Ovseyevich. Rank image processing using spatially adaptative neigborhoods. *Pattern Recognition and Image Analysis*, 11:542–552, 2001.

[23] Grewal M. and Andrew A. *Kalmal filtering, theory and practice*. Prentice Hall Infomation and Systems Sciences Series, 1993.

[24] L. A. H. Mateo. Filtros de orden prioritario utilizando vecindades adaptativas. Master's thesis, Centro de Investigacion Cientifica y de Educacion superior de Ensenada (CICESE), 2004.

[25] E.J. Coyle P.D. Wendt and Jr. N.C. Gallagher. Stack filters. *IEEE Trans. Acoust. Speech Signal Process.*, 34:898–911, 1986.

[26] A. Popoulis and S. U. Pillai. *Probability, Random Variables and Stochastic Processes*, volume 4. Mc Graw Hill, 2002.

[27] W. K. Pratt. *Digital Image Processing*. Wiley-interscience, 2007.

[28] P. E. Hart R. O. Duda and D. G. Stork. *Pattern Classification*. Wiley-interscience, 2 edition, 2000.

[29] Haykin S. *Adaptative filter theory*. Prentice Hall Infomation and Systems Sciences Series, 1991.

[30] I. Pitas S. Tsekeridou, C. Kotropoulos. Adaptive order statistic filters for the removal of noise from corrupted images. *Optical Engineering*, 37:2798–2815, 1998.

[31] Yuma Sandoval-Ibarra, Victor H. Diaz-Ramirez, and Juan J. Tapia-Armenta. Algoritmo de orden localmente-adaptativo para la mejora de senales de voz. *Congreso Internacional en Ciencias Computacionales (CiComp)*, 3:185–190, 2010.

[32] G. Strang and T.Nguyen. *Wavelets and Filter Banks*. Wellesley- Cambridge Press, 1997.

[33] A. Mahanalobis V. B. K Viaja-Kumar and R. Juday. *Correlation Pattern Recognition*. Cambridge University Press, 2006.

[34] J. Alvarez-Borrego I.A. Ovseyevich V. Kober, M. Mozerov. Nonlinear image processing with adaptive structural element. *Pattern Recognition and Image Analysis*, 13:476–482, 2003.

[35] J. Alvarez-Borrego I.A. Ovseyevich V. Kober, M. Mozerov. Rank and morphological image processing with adaptive structural element. *Pattern Recognition and Image Analysis*, 13:64–66, 2003.

[36] y I. A. Ovseyevich V. Kober. Fast running algorithm for discrete w transform. *Pattern Recognition and Image Analysis*, 17(3):399–401, 2007.

[37] S. V. Vaseghi. *Advanced digital signal processing and noise reduction*. John Wiley & Sons, 2008.

[38] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. New York: The Technology Press and Wiley, 1950.

[39] V. H. DÃaz-RamÃrez y V. Kober. Target recognition under nonuniform illumination conditions. *Applied Optics.*, 48(7):1408–1418, 2009.

[40] L. Yaroslavsky and M. Eden. *Fundamentas of Digital Optics*. Birkhauser Boston, 2001.

Apéndice:Código de implementación en dsPIC

```
/*****************************************************************************
Librerías del dsPIC
*****************************************************************************/
#include "..\h\p33FJ256GP506.h"
#include "..\h\WM8510CodecDrv.h"
#include "..\h\sask.h"
#include "..\h\SFMDrv.h"
#include "..\h\G711.h"
#include <dsp.h> // librería de funciones DSP
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <float.h>
/*****************************************************************************
Configuración del dsPIC
*****************************************************************************/
_FGS(GWRP_OFF & GCP_OFF);
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & OSCIOFNC_ON & POSCMD_NONE);
_FWDT(FWDTEN_OFF);
/* FRAME_SIZE - Size of each audio frame
* SPEECH_SEGMENT_SIZE - Size of intro speech segment
* WRITE_START_ADDRESS - Serial Flash Memory write address
* */
#define FRAME_SIZE 30
#define SPEECH_SEGMENT_SIZE 98049L
#define WRITE_START_ADDRESS 0x20000
#define s 9
fractional ev = 4700;
/* Allocate memory for buffers and drivers
* codecBuffer - Buffer used by the codec driver
* samples - buffer contained raw audio data
```

```
* encodedSamples - buffer containing G.711 encoded data
* decodedSamples - buffer containing G.711 decoded datat
* flashMemoryBuffer - buffer used by the SFM driver
* */
int codecBuffer [WM8510DRV_DRV_BUFFER_SIZE];
char flashMemoryBuffer [SFMDRV_BUFFER_SIZE];
fractional samples [FRAME_SIZE];
//fractional samples [FRAME_SIZE]= {5, 6, 10, 30, 1, 9, 10, 8, 12, 15};
fractional out_samples [FRAME_SIZE];
fractional slide_window [s];
fractional vsuma [s];
fractional vr [s];
fractional amv [s];
char encodedSamples [FRAME_SIZE];
int decodedSamples [FRAME_SIZE];
void create_sliding_window(int);
long long int create_variatonal_row(fractional);
int create_amv_array(fractional, int);
int create_new_var_row(fractional, int);
fractional set_new_signal_val(int);
/* Instantiate the drivers structures
* and create handles.
* */
WM8510Handle codec;
WM8510Handle * codecHandle = &codec;
SST25VF040BHandle flashMemoryHandle;
/* Addresses
* currentReadAddress - This one tracks the intro message
* currentWriteAddress - This one tracks the writes to flash
* userPlaybackAddress - This one tracks user playback
* address - Used during flash erase
* */
long currentReadAddress;
long currentWriteAddress;
long userPlaybackAddress;
long address;
/* flags
* record - if set means recording
* playback - if set mean playback
* erasedBeforeRecord - means SFM eras complete before record
* */
int record;
int playback;
int erasedBeforeRecord;
int main(void)
{
```

```
    /* Configure Oscillator to operate the device at 40MHz.
    * Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
    * Fosc= 7.37M*40/(2*2)=80Mhz for 7.37M input clock */
    PLLFBD=41; /* M=39 */
    CLKDIVbits.PLLPOST=0; /* N1=2 */
    CLKDIVbits.PLLPRE=0; /* N2=2 */
    OSCTUN=0;
    __builtin_write_OSCCONH(0x01); /* Initiate Clock Switch to FRC with
PLL*/
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0b01); /* Wait for Clock switch to occur */
    while(!OSCCONbits.LOCK);
    /* Intialize the board and the drivers */
    SASKInit();
    WM8510Init(codecHandle,codecBuffer);
    /* Open the flash and unprotect it so that
    * it can be written to.
    * */
    SFMInit(flashMemoryBuffer);
    /* Start Audio input and output function */
    WM8510Start(codecHandle);
    /* Configure codec for 8K operation */
    WM8510SampleRate8KConfig(codecHandle);
    /*********************************************************************
    Ciclo principal
    *********************************************************************/
    /* Main processing loop. Executed for every input and
    * output frame */
    while(1)
    {
    /*Obtain Audio Samples */
    while(WM8510IsReadBusy(codecHandle));
    WM8510Read(codecHandle,samples,FRAME_SIZE);
    int k,rk,amv_size,new_val,new_sz, w,z, suma_media, varianza;
    fractional l= floor(s/120);
    k = 0;
    while(k < FRAME_SIZE){
    create_sliding_window(k);
    rk = create_variatonal_row(samples[k]);
    amv_size = create_amv_array(samples[k], (int) s);
    if (amv_size <= l) { /* f[k] is a corrupted value */
    new_sz = create_new_var_row(new_val, amv_size);
    amv_size = create_amv_array(new_val, new_sz);
    }
    out_samples[k]= set_new_signal_val(amv_size);
    k++;
```

```
}
/* Decode the samples */
G711Ulaw2Lin (encodedSamples,decodedSamples, FRAME_SIZE);
while(WM8510IsWriteBusy(codecHandle));
WM8510Write (codecHandle,out_samples,FRAME_SIZE);
}
}
/*****************************************************************
Crear ventana deslizante alrededor de f[k]
*****************************************************************/
void create_sliding_window( int x)
{
fractional ll, hl; /* low and high sliding window array index limits */
int sw_index, f_index, i; // traia variable l_aux
ll = x - ceil( (s-1)/2 );
hl = x + ceil( (s-1)/2 );
sw_index=floor(s/2);
slide_window[sw_index] = samples[x];
sw_index–;
f_index = x+1;
for (i=x-1; i>=ll; i–) {
if (i < 0) { // use reflected signal if neccesary
slide_window[sw_index] = samples[f_index];
f_index++;
} else {
slide_window[sw_index] = samples[i];
}
sw_index–;
}
f_index = x-1;
sw_index=floor(s/2) + 1;
for (i=x+1; i<=hl; i++) {
if (i>FRAME_SIZE-1){ // use reflected signal if neccesary, le puse el menos
1
slide_window[sw_index] = samples[f_index];
f_index–;
} else {
slide_window[sw_index] = samples[i];
}
sw_index++;
}
}
/*****************************************************************
Creación del vector VR
*****************************************************************/
/*
```

```
Creates variational row; i.e., it sorts the sliding window array
It returns the index in the array of the f[k] value
*/
long long int create_variatonal_row(fractional val)
{
int ind, j;
fractional temp;
long long k_index;
for (ind=0; ind<s; ind++)
vr[ind] = slide_window[ind];
if (val == vr[0])
k_index = 0;
else
k_index = -1;
for (ind=0; ind<s-1; ind++) {
for (j=ind+1; j<s; j++) {
if (vr[j] < vr[ind]) {
temp = vr[ind];
vr[ind] = vr[j];
vr[j] = temp;
if (vr[ind] == val)
k_index = ind;
}
}
}
return k_index;
}
/*************************************************************************
Crear el vecindario EV
*************************************************************************/
/*
Creates amv array, with values >= f[k] - EV and <= f[k] + EV
It reads from sw array of size "sz"
It returns the amv array size
*/
int create_amv_array(fractional val, int sz)
{
int ind, j;
fractional l_limit, h_limit;
l_limit = val - ev; //limite inferior
h_limit = val + ev; //limite superior
ind = 0;
for (j=0; j<sz; j++) {
if ( ( vr[j] >= l_limit) && (vr[j] <= h_limit) ) {
amv[ind] = vr[j];
ind++;
```

```
}
}
return ind;
}
/*************************************************************************
Crear nuevo vector VR
*************************************************************************/
/*
Creates a new variational row, excluding corrupted values
It returns the size of the var row array, whereas "value" is
the "new" value of f[k]'*/

int create_new_var_row(fractional val, int sz)
{
int ind, i, j, is_in_array;
fractional temp;
ind = 0;

for (i=0; i<s; i++) {
   is_in_array = 0;
for (j=0; j<sz; j++) {
   if (slide_window[i] == amv[j]) {
      is_in_array = 1;
      j=s;
   }
}
if (!is_in_array) {
   vr[ind] = slide_window[i];
   ind++;
   }
}
i = ceil (ind/2);
j = ind % 2;
if (j == 0) {
   val = (vr[i-1] + vr[i]) / 2;
} else {
   val = vr[i];
}
for (i=0; i<ind-1; i++) {
   for (j=i+1; j<ind; j++) {
   if (vr[j] < vr[i]) {
      temp = vr[j];
      vr[j] = vr[i];
      vr[i] = temp;
      }
   }
```

```
    }
    for (i=ind; i<s; i++) {
       vr[i] = 0.0;
    }
    return ind;
    }
    /*************************************************************************
    Valor de la señal de salida para la versión 1
    *************************************************************************/
    fractional set_new_signal_val(int sz)
    {
    fractional a;
    int j;
    a=0.0;

    for (j=0; j<sz;j++){
       a +=amv[j];
    }
    if(sz>0){
       return a/sz;
    }else{
       return 0.0;
       }
    }
    /*************************************************************************
    Valor de la señal de salida para la versión 3
    *************************************************************************/
    double set_new_signal_val(int sz)
    {
    double wm, fd, div;
    int j, ind, cl, fa;

    ind = floor( sz / 2 ); // find the index of k
    fd = (double) ev * WMP; // criteria used to calculate "weights" to determine
how to define set elements
    cl = fa = 0; // how many values are close (cl) or far (fa) away from k
    for (j=0; j<sz; j++) {
        if ( ( amv[j] <= amv[ind] + fd) && ( amv[j] >= amv[ind] - fd) ) {
         cl++;
    } else {
       fa++;
       }
    }
    wm = 0.0;
    for (j=0; j<sz; j++) {
    if ( ( amv[j] <= amv[ind] + fd) && ( amv[j] >= amv[ind] - fd) ) {
```

```
    wm += amv[j] * cl;
    div += cl;
  } else {
    wm += amv[j] * fa;
    div += fa; } } return (wm / div);
  }
```