VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

POSGRADO DE CIENCIAS E INGENIERIA DE LA COMPUTACION

"DETECCIÓN DE FUENTE DE SONIDO Y RECONOCIMIENTO DE VOZ CON REDES NEURONALES PARA ROBOT DE SERVICIO"

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERIA (COMPUTACION)

PRESENTA:

ISRAEL BAHENA CASALES

TUTOR:

DR. JESÚS SAVAGE CARMONA

FACULTAD DE INGENIERIA

MÉXICO, ENERO, 2017

Índice General

Agradecimientos

Se agradece a la DGAPA-UNAM por el apoyo proporcionado para la realización de esta tesis a través del proyecto PAPIIT IG100915 "Desarrollo de técnicas de la robótica aplicadas a las artes escénicas y visuales"

Al programa de estudios de maestría de Conacyt 2014.

Capítulo 1

Presentación

1.1 Introducción

En la actualidad, dentro del ámbito tecnológico, más concretamente en las interfaces hombre-máquina se busca llegar a un nivel de interacción lo más natural y cercano a lo que está acostumbrado el ser humano. Un ejemplo de desarrollo de este tipo de interacción está presente en los robots dedicados al servicio humano para realizar tareas del tipo domésticas, los cuales se encuentran en constante evolución de partes mecánicas y físicas así como de sus algoritmos de ejecución para lograr dicho nivel de interacción.

Un ejemplo de una interfaz hombre-máquina es el robot Justina del departamento de Bio-robótica de la Facultad de Ingeniería (UNAM), sometida a pruebas de laboratorio y competencias organizadas por comités nacionales e internacionales para exponer sus avances tecnológicos. El robot interpreta señales espaciales y temporales tales como imágenes, sonido etc., adquirida por sus sensores para ser analizadas por su unidad de procesamiento y de esta forma con los resultados se pueda hacer una planeación de las acciones a realizar, tales como responder una pregunta, tomar un objeto, moverse a un destino, etc.

La Fig 1.1 presenta el diagrama organizacional del robot Justina (denominado Virbot), el cual se divide en 4 grandes módulos: un módulo de entrada, uno de planeación, otro de representación del conocimiento y finalmente uno de ejecución.

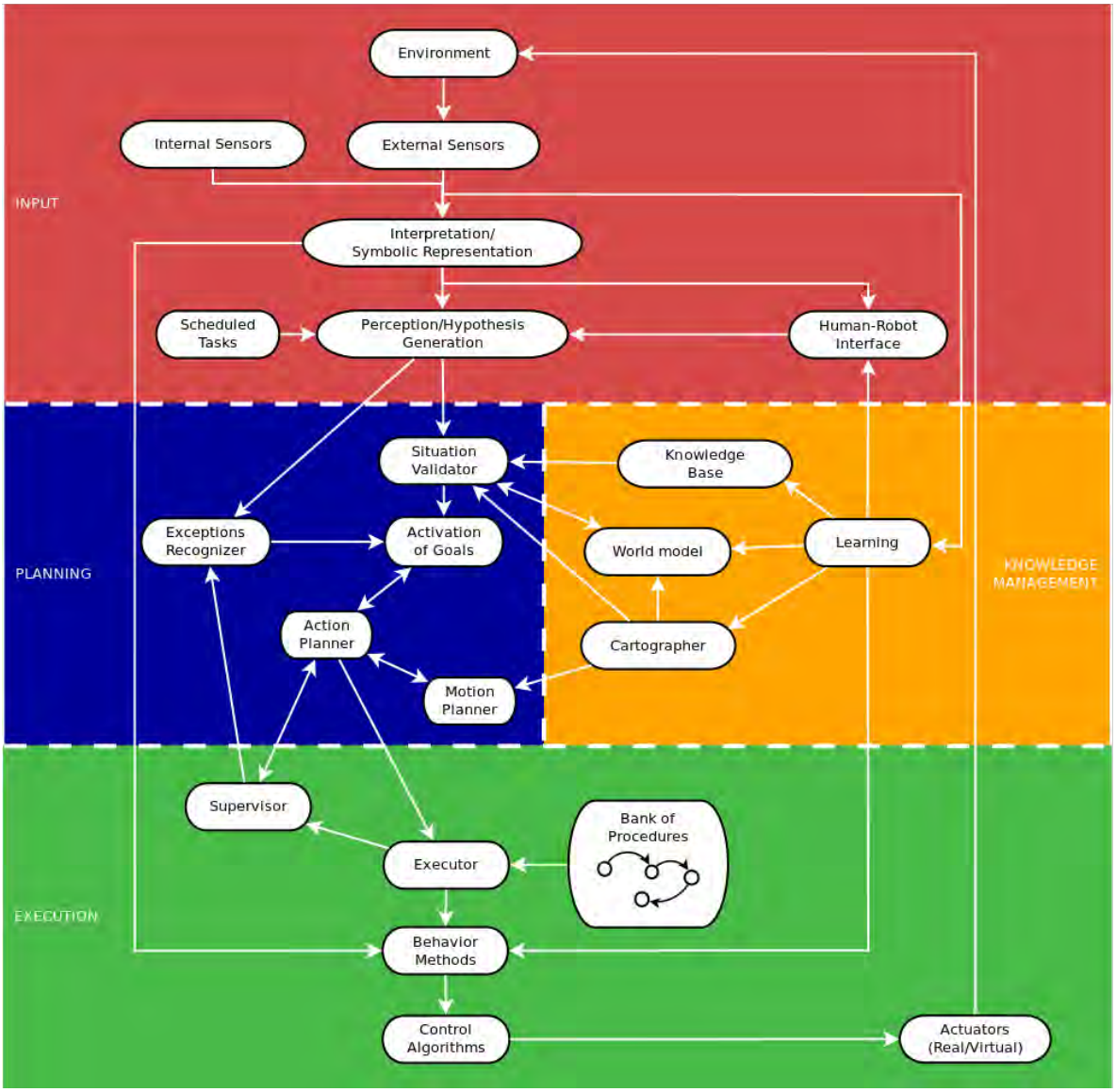

Figura 1.1 Diagrama de los sistemas del Robot Justina (VirBot versión 2016).

En el bloque de entrada se encuentra la interfaz hombre-máquina del robot Justina, el cual a su vez cuenta con un módulo de audio, cuyo objetivo es tener una representación simbólica de los comandos de voz. Este sistema, en el primer trimestre del 2016 tiene un micrófono frontal donde se capta la señal de voz y es alimentada al reconocedor de voz de Microsoft[1], sin embargo presentan los siguientes problemas:

1.- Escuchar a cualquier tipo de persona incluyendo a aquellos que no se dirijan a él.

2.- El volumen de voz, ya que si un usuario se encuentra en otra posición que no sea frontal al robot tendrá dificultades para ser escuchado y el reconocimiento falla.

[1] Speech SDK propiedad de Micrsoft, https://www.microsoft.com/en-us/download/details.aspx?id=10121

En Fig 1.2 se muestra el bloque de interés (módulo de audio) del robot Justina.

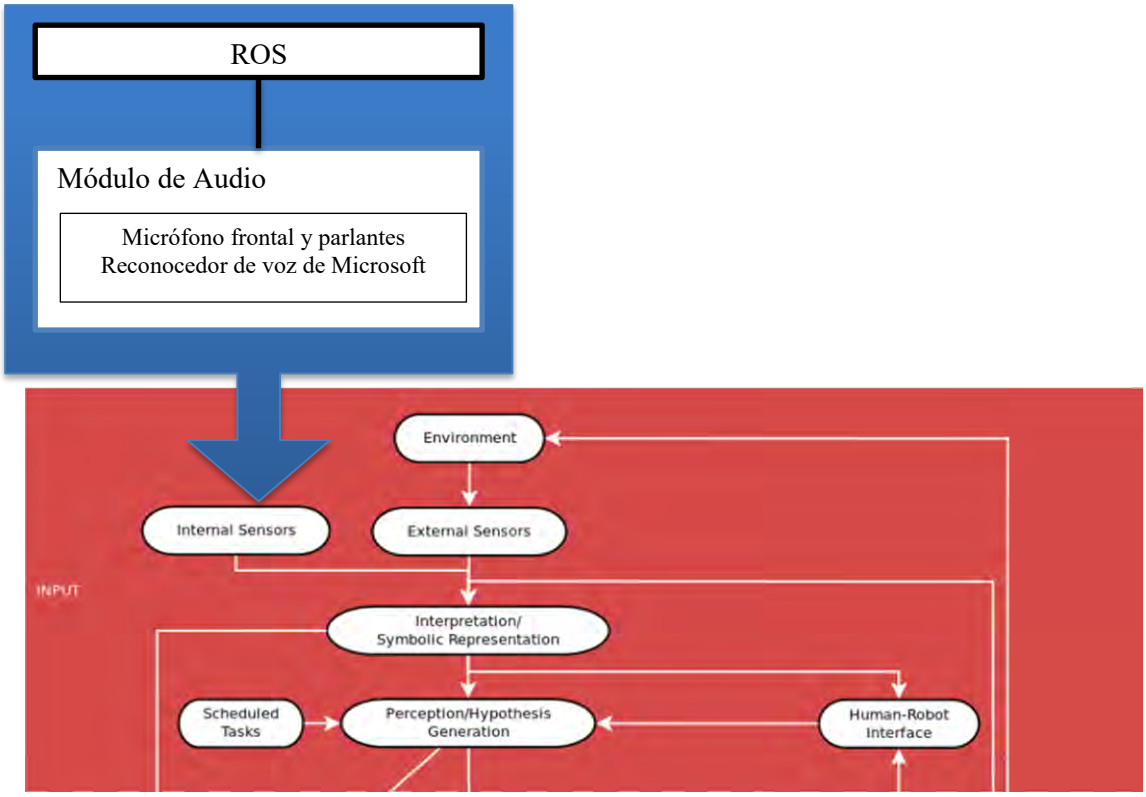

Figura 1.2 Diagrama de los sistemas de entrada del Robot Justina, se muestra parte de los sensores de entrada actuales (VirBot versión 2016).

El presente trabajo muestra la búsqueda e implementación de una versión actualizada del módulo de audio del robot Justina, para tratar de solventar sus deficiencias. Para esto se propone: por una parte tener un sistema basado en el uso de una "*red neuronal recurrente*", para aprovechar el potencial de este tipo de redes neuronales que popularmente ha tenido buenos resultados en el tratamiento de señales como lo es la voz. Se propone una arquitectura propia y dedicada al reconocimiento de palabras claves, hecha a la medida para ejecutarse con el menor número de recursos posibles y ayudar a identificar cuando una persona se dirige al robot, sin embargo existe el riesgo de no alcanzar al estado del arte. Por otra parte implementar algoritmos ya conocidos (convencionales) para la *"detección de la fuente de sonido"*, pero con una ligera modificación para así darle al robot la información con la que pueda ubicar la fuente de sonido en un ángulo de 0° a 360°, de esta forma solucionar el problema cuando el usuario no está frente al robot.

Ambos desarrollos parecieran distantes, sin embargo no es así, pues se requiere la integración de ambos junto con el ya utilizado reconocedor de voz de Microsoft para actualizar el módulo de audio del robot y así Justina tenga un mejor desempeño al momento de interactuar con el usuario. Con estos conceptos presentes, la tesis tiene como objetivo hacer un aporte tecnológico necesario al robot.

1.2 Objetivos

Este trabajo se enfoca en el desarrollo de una red neuronal recurrente para el reconocimiento de palabras claves. También se busca la implementación de un sistema de detección de audio convencional, de esta forma el robot voltee para que el usuario hable de frente al micrófono y así mejorar la tasa de reconocimiento del reconocedor de Microsoft. Los objetivos generales que guiarán el desarrollo de la tesis son:

- Establecer una arquitectura de red neuronal recurrente y determinar datos de entrada a la red así como el acondicionamiento a la señal de voz.
- Proponer sistema de detección de usuario, usando sonido, el cual se encuentre entre 0° a 360°.
- Implementación, integración y análisis de resultados de los módulos desarrollados.

1.3 Estructura del documento

La tesis está organizada por capítulos, en los cuales se describe el proceso llevado a cabo para el desarrollo del aporte tecnológico hecho al robot, a continuación una breve descripción del contenido en cada capítulo.

En el *Capítulo 2* se proporciona los antecedentes necesarios para introducirse al contexto del documento, también se aborda de manera breve el estado del arte con la finalidad de estar familiarizados con algunos conceptos y metodologías.

En el *Capítulo 3* se analiza la información existente para la selección e implementación de una de las arquitecturas de red neuronal recurrente (RNN por sus siglas en ingles *recurrent neural network*). Se describe el proceso que se lleva a cabo para probar una red neuronal, preparar sus entradas, así como la propuesta de una red para el reconocimiento de palabras claves.

En el *Capítulo 4* se presenta el desarrollo e implementación de un algoritmo para la detección de la fuente de sonido con una ligera modificación para obtener un ángulo entre 0° a 360°.

En el *Capítulo 5* se presentan las pruebas, comparaciones y resultados de los desarrollos realizados en el capítulo 3 y capitulo 4.

En el *Capítulo 6 s*e presentan las conclusiones de la tesis y se propone trabajo a futuro.

Capítulo 2

Estado del arte.

El reconocimiento del habla natural por un sistema como una máquina, es un problema que ha experimentado una extensa e intensa evolución para lograr obtener los sistemas de reconocimiento de voz que existen actualmente. Dado que el objetivo de la tesis es desarrollar un sistema de reconocimiento de palabras claves a continuación se explora en una sección la evolución de sistemas que han sido implementados en máquinas. Mencionando metodologías convencionales y tendencias de los nuevos sistemas, que a pesar del gran avance que tienen, siguen sin igualar las capacidades del reconocimiento de un ser humano. En otra sección se mencionan las aplicaciones en las que se ha originado la metodología de detección de dirección de arribo DOA (direction of arrival).

2.1 Reconocimiento de voz.

A continuación se aborda las técnicas usadas en los sistemas de reconocimiento de voz, divididos en dos grandes grupos: técnicas convencionales (los cuales fueron desarrollados desde la década de los 40) y los que son basados en Aprendizaje profundo (desarrollados al día de hoy).

2.1.1 Técnicas convencionales en el reconocimiento de voz.

Las técnicas convencionales son ideadas por los humanos, esto significa, que son la interpretación o modelación de un sistema físico por medio de una ecuación matemática propuesta por un humano. Por otra parte, un sistema basado en aprendizaje profundo implementa un modelo de red el cual durante su entrenamiento se encarga de la interpretación y abstracción de la información.

Técnicas basadas en modelos de la fonética acústicos.

En los años 40 y 50 se desarrollaron los primeros sistemas de reconocimiento de líneas mono-locutor como el de la Fig 2.1, basados en medidas espectrales, obtenidas del modelo del tracto vocal generado para cada digito. Para los años 60 comenzaron a existir sistemas electrónicos, con hardware especifico, lo que supuso un impulso en el tratamiento y captura de señales. También en estos años se propuso que cada segmento de la voz puede ser representado por un modelo acústico con determinadas propiedades, estas propiedades se podían extraer ya sea en el dominio del tiempo o la frecuencia para formar el modelo. A su vez el modelo era promediado y actualizado con múltiples generaciones de del mismo segmento de voz [6]. Para ejecutar el reconocimiento se

alineaban temporalmente (técnica DTW, Dynamic Time Warping) los modelos almacenados de las palabras con la señal entrante hasta que el modelo convergiera con la señal a reconocer.

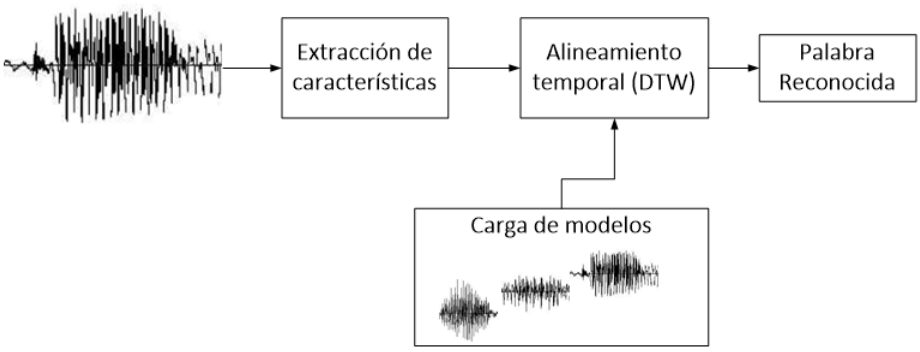

Figura 2.1 Sistema de reconocimiento basado en el análisis de segmentos de voz representados como un modelo acústico fonético.

Técnicas basadas en: LPC, cepstral, ajuste de patrones, predicción probabilística.

En los años 70 se contaba con un conocimiento más profundo sobre las señales, lo que llevo a desarrollar un estándar como el mostrado en la Fig 2.2, para el reconocimiento de palabras y frases. Este sistema está compuesto de una etapa de captura de datos, los cuales pasan a un bloque de pre-procesamiento para preparar a la señal para una etapa de procesamiento que es conocida como extracción de características, es aquí donde entran los modelos de interpretación de la señal. Finalmente se tiene una etapa de clasificación donde los resultados del procesamiento de la señales de prueba se ajustan a una familia pre-entrenada por el sistema, tales como cuantizacion vectorial o incluso una red neuronal artificial (RNA).

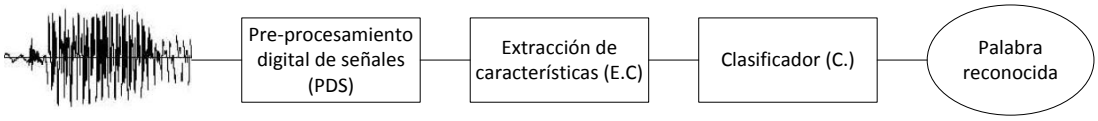

Figura 2.2 Sistema estándar en el reconocimiento de voz.

Pre-procesamiento digital de voz

En este punto de la historia ya se contaba con un estudio de lo que es el tratamiento de la señal de voz, este conjunto de técnicas no hacían más que preparar la señal de voz para cualquier sistema que procesara estos datos, coloquialmente se conoce como acondicionar los datos de entrada al sistema de procesamiento de voz. Un ejemplo de esta etapa es la Fig 2.3, donde: primero se usa una

herramienta lógica para digitalizar la señal de voz, comúnmente se usa una frecuencia de muestreo de 16 khz porque se trabaja con voz humana. Después se hace uso de un filtro de preénfasis que realza las componentes de mayor frecuencia de la voz además de que los segmentos de voz denominados sordos contiene información de consonantes los cuales se encuentran muy atenuados, por lo cual se ven beneficiadas de éste filtro. Otra herramienta es la partición de bloques seguida de un ventaneo para evitar distorsiones de la información al aplicar una transformación [7].

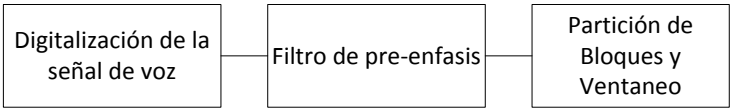

Figura 2.3 Etapas de Pre-procesamiento de un sistema convencional de procesamiento de voz.

Extracción de Características

Una vez con la señal preparada, se intentan reconocer ciertos patrones sin la necesidad de identificar las características explícitas de cada segmento de voz. Estos métodos como se mencionó anteriormente surgen como interpretación del ser humano para poder generar la señal de voz, siendo el modelo del tracto vocal el más utilizado.

El tracto vocal como se explica en [30][31][32] se modela como un filtro con coeficientes variables en el tiempo en función de la palabra a pronunciar con dos posibles señales de entrada para su excitación, ya sea un tren de pulsos para las señales sonoras o ruido aleatorio para las señales no sonoras.

- **Análisis de predicción lineal LPC**

Para el proceso de reconocimiento de voz se usan mecanismos que ayuden a representar la señal para generar una descripción representativa con la cual se pueda trabajar, una de ellas es el análisis de predicción lineal. En la predicción lineal Markhoul [8] (ecuación 2.1) se asume que cada muestra de la señal de voz que es digitalizada s(n) se puede aproximar como una composición lineal de un número de sus muestras pasadas s(n-k), se trata de un filtro FIR (filtro de respuesta de impulso infinita)

$$s(n) = \sum_{k=1}^{p} a_k \cdot s(n-k) + G \cdot u(n) \quad \ldots (2.1)$$

Donde *u(n)* es la entrada del filtro el cual será un ruido aleatorio para producir señales no sonoras o un tren de pulsos para producir señales sonoras, de esta manera se representa el tracto bucal que fue el modelo con el cual se plateó el problema.

Para obtener la función de transferencia (ecuación 2.2) se obtiene la transformada de Fourier:

$$S(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k \cdot z^{-k}} \dots (2.2)$$

Donde G será la ganancia del filtro.

Así el problema radica en determinar los coeficientes de predicción a_k denominados los LPC de la señal de voz, pues serán los coeficientes de predicción los parámetros a usar para describir a la señal. Una forma de obtenerlos es mediante la minimización del error cuadrático medio.

La distancia de Itakura-Saito (ecuación 2.3) es una medida espectral [9] que permite comparar la señal con sus coeficientes LPC, de esta forma se cuenta con una medida al momento de comparar los LPC del patrón con las señales a reconocer.

$$d_{ita-sai} = \sum_{n=0}^{p-k} a_n \cdot a_{n+k} \dots (2.3)$$

- **Análisis cepstral**

Además de los descriptores LPC se idearon otros que mostraron mejores resultados como lo son los denominados coeficientes (modelo en la ecuación 2.7) cepstrales Tokhura [10], Juang et al. [11], los cuales se obtienen partiendo de la misma idea del modelado del tracto vocal pero una etapa antes esto es modelando la excitación de los pulmones y las cuerdas vocales como dos señales:

$$v(t) = h(t) * x(t) \qquad (* \rightarrow operador\ convolucion) \quad \dots (2.4)$$

Por lo cual se tiene la siguiente propiedad en el dominio de Fourier:

$$V(\omega) = H(\omega) \cdot X(\omega) \dots (2.5)$$

Al aplicar el operador logaritmo en ambos lados de la ecuación 2.5 se puede convertir la multiplicación de logaritmos como la suma de los mismos (ecuación 2.6). De esta forma lo que se logra es una separación de la señal de excitación a la señal correspondiente al tracto vocal, tratando de emular al proceso que realiza el oído humano.

$$\log\big(V(\omega)\big) = \log\big(H(\omega)\big) + \log\big(X(\omega)\big) \dots (2.6)$$

A la transformada inversa de Fourier de ésta señal se le denomina cepstrum (ecuación 2.7)

$$\text{cepstrum}\big(\text{v(t)}\big) = \sum_{n=-\infty}^{\infty} C_n e^{-i\omega t} \dots (2.7)$$

De esta señal se pueden obtener los coeficientes C_n que corresponden a los coeficientes cepstrales como se explica en [25], los cuales serán utilizados para describir la señal.

Al igual que en los LPC, existe una medida para comparar dos vectores cuya representación es Cepstral, la cual como bien se recuerda, se obtiene de una separación en el dominio de Fourier de las señales, por tanto la comparación sigue siendo espectral en la ecuación 2.8:

$$d_{cep} = \sum_{n=0}^{p-k} (C_a(n) \cdot C_b(n))^2 \dots (2.8)$$

Clasificador de palabras aisladas.

Una vez que la señal está caracterizada se usa un sistema de clasificación, este sistema tiene una etapa de entrenamiento donde se usan muestras para almacenar los modelos o patrones característicos de la señal. Como ejemplos se tienen a cuantizacion vectorial y K-medias.

- **Cuantización vectorial**

El método consiste en una separación de datos en nubes de todos los patrones en un espacio definido de forma iterativa usando una distancia de medida entre datos patrón, con el fin de generar centroides para esos conjuntos de puntos y de esta forma se tienen representantes para cada grupo de patrones [22]. Esto reduce en gran medida la cantidad de información almacenada para clasificar.

Una vez generado los centroides, estos se colocan en bloques en la etapa de procesamiento como en la Figura 2.2, para calcular la distancia acumulada de la señal de prueba con cada centroide que representa cada palabra aprendida en el entrenamiento (es la distancia acumulada ya que se procesa cada ventana de la palabra aislada). Una vez procesada todos los bloques de la señal, aquella suma acumulada que sea la mínima, será la palabra identificada [32].

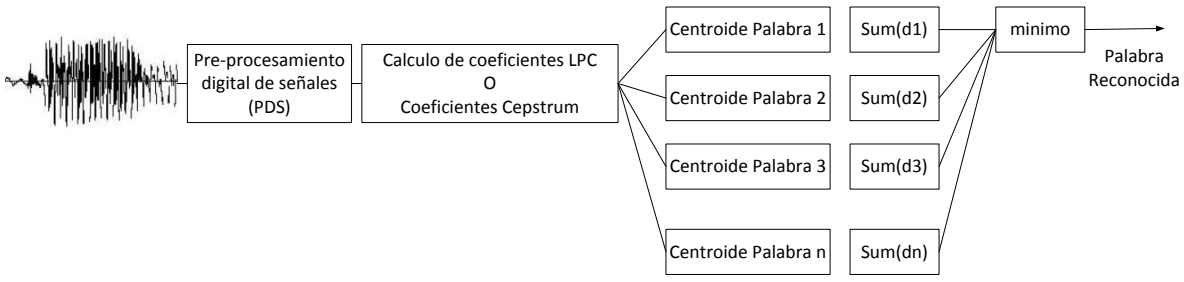

Figura 2.3 Sistema de reconocimiento de palabras aisladas, digitalización de la señal, procesamiento y acondicionamiento de la misma, etapa de codificación y descripción, distancias con centroides de patrones en previo entrenamiento y finalmente selección de la palabra.

Clasificador con predicción probabilística.

A pesar de que los clasificadores convencionales presentaban resultados buenos, no tenían una taza de reconocimiento similar al del ser humano, por lo tanto se buscó un mejor desempeño para sistemas de clasificación, ya sea modificando el método o más comúnmente colocando una etapa de predicción. Una técnica muy recurrente es el uso de modelos ocultos de Markov.

- **Modelos ocultos de Markov**

HMM (hidden Markov Model), modelos de markov ocultos es un modelo probabilístico, la cual fue propuesta y estudiada entre los años 60 y 70 por Baum y colegas [26][27][28].

Estos modelos matemáticos están compuestos de dos variables aleatorias: el estado S y las observaciones O (Figura 2.3). El objetivo es determinar un modelo de probabilidad correspondiente a las observaciones de entrada ya que se asume que los estados están ocultos, estos modelos pueden representar segmentos de voz (palabras, oraciones) y un estado de este modelo puede ser un elemento estructural (una silaba, un fonema) [11].

Se puede analizar a un modelo λ como un conjunto de estados los cuales están conectados por probabilidades de transición entre estados y con una función de probabilidad de observación en cada estado. De este modelo se generan dos matrices: la primera de probabilidad de transiciones entre estados A y la segunda de probabilidad de una observación en cada estado B.

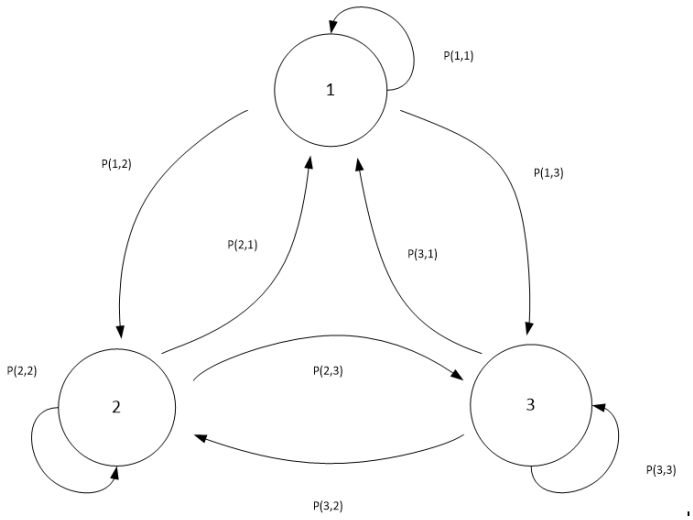

Figura 2.4 Modelo de Markov Oculto, las probabilidades de transición P y los estados ocultos.

Dado un modelo $\lambda(\pi, A, B)$ y una secuencia de observaciones O = (o_1,o_2,…o_n) se debe decidir la secuencia de estados más probable Q=(q_1,q_2,..q_n) al calcular la probabilidad de cada secuencia posible y seleccionar la de mayor probabilidad para lo cual se utiliza el algoritmo de Viterbi [12]. El algoritmo de Viterbi calcula de forma recursiva las rutas parciales, descarta los de menor probabilidad para al final tener la ruta más probable.

Los modelos de Markov se pueden utilizar para corregir el error de estimación de la etapa de cuantizacion vectorial en el reconocimiento de palabras aisladas, esto es, que el modelo reciba las observaciones de los fonemas y que calcule el modelo más probable que corresponde a la palabra, estos modelos y su aplicación en el reconocimiento de voz fueron estudiados previamente por Rabiner [11].

- **Reconocimiento de palabras clave en señales de voz**

En vez de usar la cuantizacion vectorial para identificar una palabra, lo que se hace es entregar el índice del centroide como una observación para los modelos de markov (un ejemplo de este sistema se presenta en la Figura 2.4), de esta manera con viterbi se calcula el modelo más probable de entre todos los posibles y por ende se tiene la palabra clave más probable dentro de una trama de voz. Cuando el vocabulario resulta ser muy amplio y se desea agregar información de tipo gramatical o información semántica para mejorar la identificación y corrección de palabras se utilizan técnicas de inteligencia artificial como las descritas en Carmona [12].

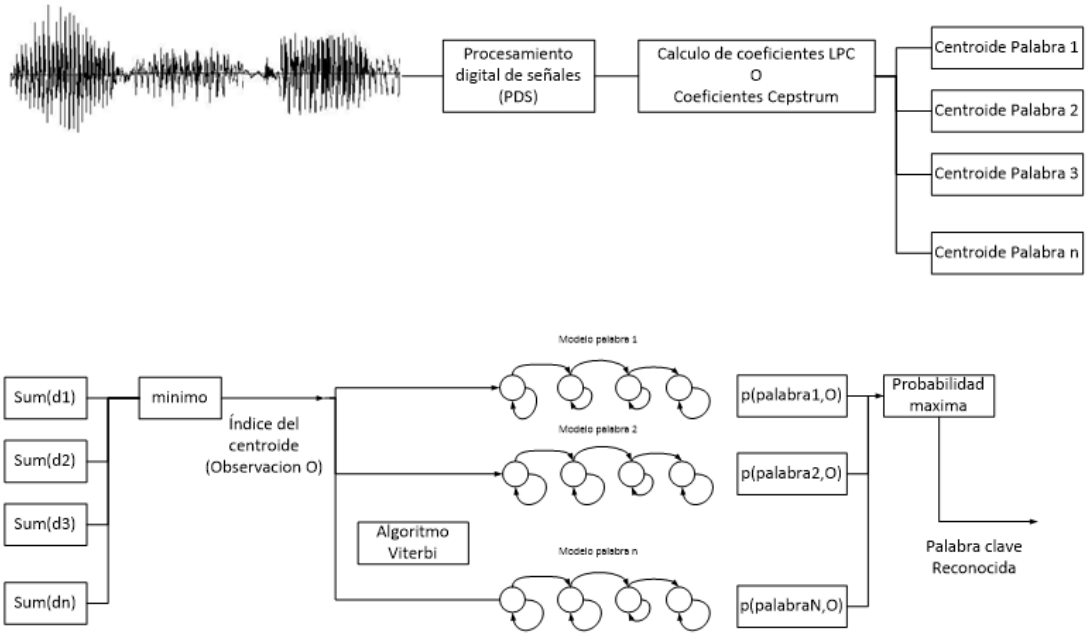

Figura 2.5 Sistema convencional de reconocimiento de palabras claves en una trama de voz, de izquierda a derecha el flujo es el mismo al de la figura 2.2 con la adición de HMM para determinar la palabra clave.

Clasificador con redes neuronales artificiales

La popularización de las redes neuronales artificiales (RNA) en los años 80 y 90 llevó a la comunidad científica a buscar su implementación en muchos problemas, de entre ellos el reconocimiento de voz. Hasta este momento las redes multicapas para la clasificación de patrones

resultan ser las más eficaces y caracterizadas. Usando el método de aprendizaje backpropagation se podía obtener una red robusta y entre más patrones de entrada tuviera era mejor para el aprendizaje de la red, por esta razón se intentó reemplazar elementos dentro de las técnicas convencionales de reconocimiento con redes neuronales dando resultados bastante aceptables [18]. Un ejemplo de la implementación de estas redes multicapas es el remplazo de la cuantizacion vectorial por un bloque de RNA para clasificación de los coeficientes LPC o cepstral como se observa en la Fig 2.5.

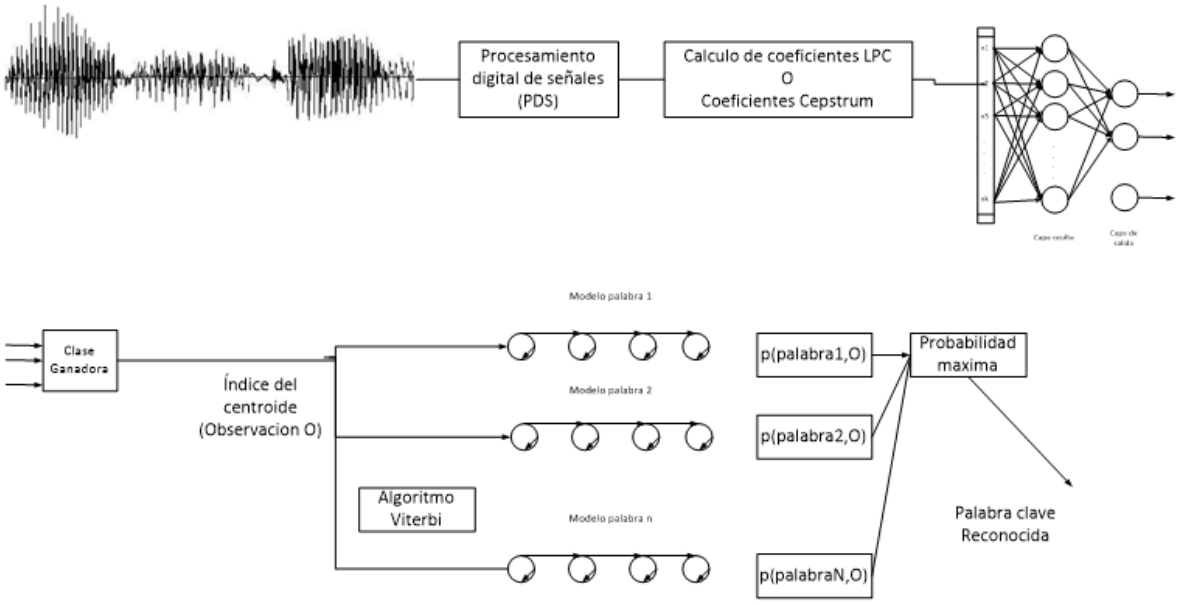

Figura 2.6 Sistema de reconocimiento de palabras clave en una trama de voz utilizando un clasificador por redes neuronales multicapa de los descriptores LPC o Cepstral y a la salida de la red neuronal se selecciona la clase ganadora al cual pertenece el fonema o silaba siendo la observación para HMM.

2.1.2 Técnicas de reconocimiento de voz basadas en aprendizaje profundo.

Con el tiempo la ambición de remplazar módulos del sistema de reconocimiento de voz en una metodología convencional creció, sin embargo el tipo de redes neuronales convencionales ya no eran suficientes para llevar a cabo dicha tarea. Como resultado, han adquirido un mayor interés de estudio para poder llevar a cabo su aplicación dentro de estas áreas [35][34].

La investigación sobre redes creció [13][14], siempre apoyadas por el crecimiento en el poder de computo de estas últimas dos décadas pues permitían pruebas a gran escala y en menor tiempo a lo conocido hasta entonces. Este tipo de desarrollos apoyo a un nuevo término en las redes neuronales el cual era “profundo”. Una red profunda se diferenciaba de las convencionales en que la red ya no solo se encargaba de pequeños problema como clasificación, sino que remplazaban sistemas complejos como descripción y clasificación juntas en una sola red. Caracterizadas por su gran número de capas y neuronas, capaces de decodificar a un nivel exhaustivo las características de

la señal, tener una mayor extracción e interpretación de la información, básicamente la red lo hace todo. Cabe aclarar que este tipo de redes son voraces por lo cual su efectividad se ve relacionada en gran medida por la cantidad de patrones alimentados es por ello que durante su entrenamiento de hace uso de enormes bases de datos de aprendizaje., pero siempre apoyadas por el poder de cómputo que permiten su entrenamiento con algoritmos modernos (los cuales son modificaciones al Backpropagation [19]).

Reconocimiento de voz con redes neuronales recurrentes.

La tendencia en la actualidad como lo muestra la Fig 2.8, es remplazar la mayor parte de los componentes dentro del reconocimiento de voz e ir probando la funcionalidad del sistema hasta tener en su totalidad un sistema de reconocimiento que utilice redes neuronales profundas.

Varias de estas redes que se usan para el reconocimiento de señales temporales, como lo es la voz, se denominan recurrentes (RNN por sus siglas en inglés, recurrent neural network), ya que en sus resultados influye la información pasada de la red, lo que se traduce en memoria. Su eficiencia es mejor a la de una red meramente reactiva, para este tipo de señales, por lo cual tienen un gran impulso para su implementación en los sistemas actuales como la Fig 2.7 [33].

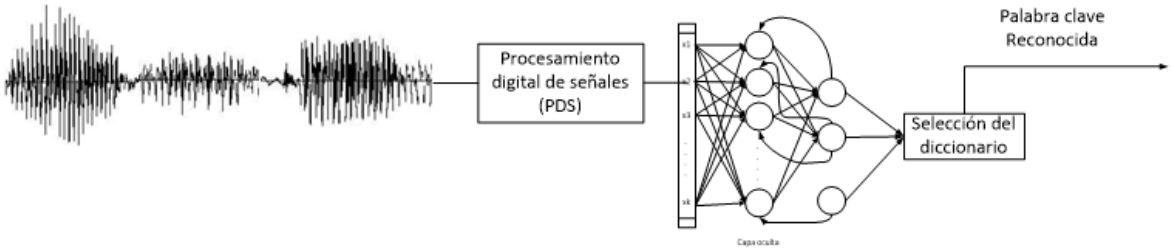

Figura 2.8 Tendencia de sistema de reconocimiento de voz realizada por una red neuronal profunda con determinada topología.

2.2 Detección de la dirección de origen en una señal

El objetivo en la detección de la dirección de arribo (direction of arrival, DOA) es para determinar con precisión el número de fuentes que producen las formas de onda y sus lugares correspondientes. Históricamente, las técnicas de DOA han encontrado aplicación en radar, sonar, la vigilancia electrónica, campos de exploración sísmica y antenas [38].

Como mencionan Rascón y Meza [39] surgen aplicaciones en el ámbito de la robótica, en 1989 el robot Squirt usaba algoritmos de DOA para localizar una fuente de sonido en un ambiente oscuro. Para 1993 surge la primera generación de robots [40][41] que están integrados por un sistema completo de audio (separación de fuentes y reconocimiento de voz) que se enfrentan a escenarios

más elaborados, como lo son fuentes de audio móviles, localización activa, interacción hombre-máquina más dinámica[2].
A inicios del año 2000 los robots usaban un sistema de dos micrófonos, inspirados por el modelo natural del ser humano, continúan diciendo Rascón y Meza [39]. Posteriormente a pesar de la popularidad que consiguió, su debilidad fue el intentar imitar la naturaleza humana al usar dos sensores pues había interés en elevar el desempeño del sistemas de audio localización, por lo cual este modelo se abandonó y se propuso un incremento en el número de sensores, este echo abrió las puertas a técnicas de DOA más sofisticadas como MUSIC y algoritmos Beamforming que se basan en el uso de datos de varios sensores[3].

Existen varios tipos de técnicas de para calcular DOA, como lo son basadas en el ángulo de recepción (Angle of Arribal, AoA), diferencia en el tiempo de recepción (Time Difference of Arribal, TDOA) y diferencia en el tiempo de recepción (Frecuence Difference of Arribal, FDOA) [42][43]. La tesis usa una técnica basada en TDOA, pues en las últimas dos décadas son de mayor recurrencia en la literatura, sistemas de entretenimiento como Kinect[4] de Microsoft y sistemas robóticos de servicio.

[2] Cita textual a Rascón y Meza, Localization of Sound Sources in Robotics: A Review, Mexico.
[3] Cita textual a Rascón y Meza, Localization of Sound Sources in Robotics: A Review, Mexico.
[4] Información obtenida del SDK de Kinect for Xbox One de Microsoft 2016

Capítulo 3

Redes Neuronales Recurrentes para reconocimiento de palabras clave.

Los primeros intentos de introducción de redes neuronales para resolver el problema del reconocimiento de voz fue el remplazar partes de las metodologías convencionales, como los clasificadores por modelos comunes de redes neuronales. A partir de la década de los 70 se ha venido demostrando el potencial del uso de redes neuronales artificiales para problemas de clasificación [13]. El modelo más utilizado y popularizado en la época de los 80 fue una arquitectura de varias capas con un número de neuronas en cada una de estas capas del tipo perceptronica ha demostrado ser suficientes para el problema de clasificación de numerosos problemas de señales.

3.1 Redes neuronales artificiales.

El modelo de la Fig 3.1 es de las neuronas artificiales, las cuales buscan emular de una manera simplificada y básica la función sináptica de las neuronas biológicas[14]. Tienen receptores que serán las entradas a la neurona multiplicadas o ponderadas por pesos que asignan determinada importancia a cada entrada de la red, posteriormente se suman todas las entradas y se usa una función de activación para generar la salida de la red, hay modelos que adicionalmente cuentan con un sesgo que es una medida para corrección de referencia.

Denominadas *feed-forward* porque siempre el flujo de la información es hacia adelante.

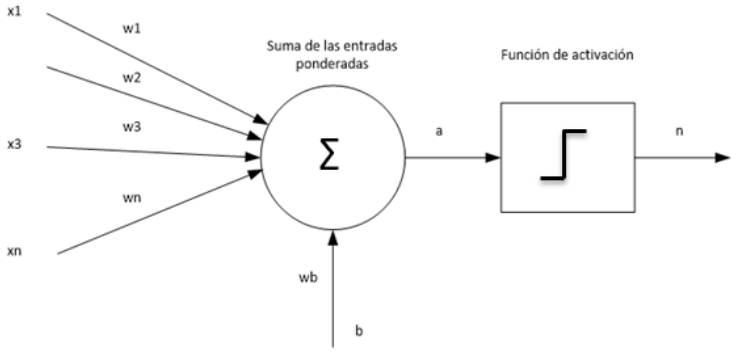

Figura 3.1 Neurona Artificial.

La interpretación matemática de la neurona artificial es:

$$n = f(a) \quad donde \quad a = \sum_{k=1}^{n} (w_k \cdot x_k) + (w_b \cdot b)$$

Redes neuronales artificiales hacia adelante (feed-forward)

Al conectar varias unidades de estas neuronas artificiales se obtiene una red neuronal artificial capaz de resolver problemas de clasificación, pues al trabajar conjuntamente, la red se ve obligada a interactuar con otras neuronas y decodificar la información de entrada para generar la salida deseada. En una red neuronal, las neuronas que reciben la información se les denomina capa de entrada, las neuronas que se encuentran en la etapa final de la topología se denomina capa de salida, y todas aquellas que se encuentran entre estas dos capas se denomina capas ocultas y como su nombre lo indica, casi siempre se ocultan del diseño ilustrativo de la arquitectura (Fig 3.2).

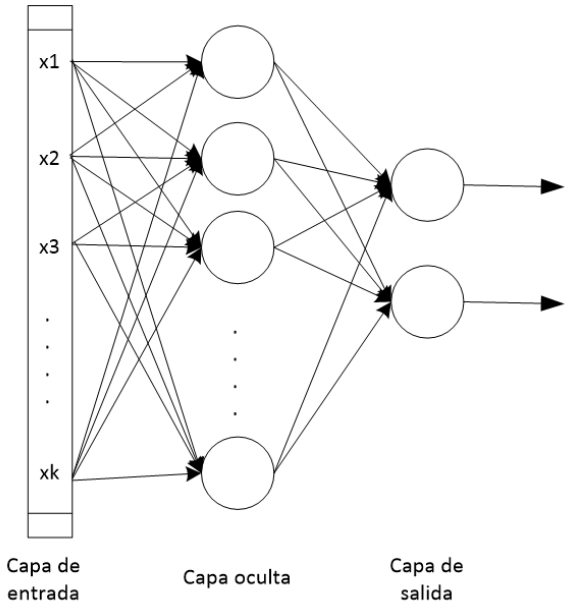

Figura 3.2 Topología de una Red Neuronal artificial multicapa

Existen múltiples funciones de activación, desde aquellas que solo tienen como salida el rango binario (Harlim), o su versión simétrica con valores negativos (Hardlims), hasta las que tienen rango decimal como la Sigmoide.

También existen distintos tipos de neuronas artificiales feed-forward, en la literatura se manejan muchos términos para su clasificación, y el ensamble de estas neuronas forma distintas topologías, cada topología resuelve más eficazmente un tipo de problema. Desde clasificación en clases con una red multicapa como el de la Figura 2.6, autoencoders que son arquitecturas con pocas neuronas en sus capas ocultas que sirven para reproducir la señal de entrada (permiten reconstruir información con ruido o perdidas), redes convolucionales para análisis espaciales en las muestras de entrada, ya que identifican rasgos o pequeñas descripciones en zonas de la señal de entrada. Se han encontrado muchas topologías y todo depende del problema a resolver.

Algoritmo BackPropagation

Existen varios métodos de entrenamiento para las redes neuronales como el algoritmo LMS o Hebbiano, etc., pero de estos métodos el que más sobresale es el denominado BackPropagation cuya premisa es propagar el error de la salida a las capas anteriores.

En las últimas décadas fue muy usado para entrenamiento con redes multicapa, con la desventaja de que requiere muchas iteraciones y que también el error se va diluyendo de forma no gradual a lo largo de las capas donde se propaga, sin embargo es robusto y tiene una alta capacidad de generalización para diferentes topologías.

Pasos de algoritmo [15][16][17]:

1. Inicialización de pesos w de la red neuronal, pueden ser valores fijo o aleatorios
2. Mientras el error sea mayor a un umbral repetir pasos del 3 al 6.
3. Se presenta a la red un patrón de aprendizaje como entrada [x1, x2, x3, … xk] y se establece la salida deseada [n1, n2, n3, … nk].
4. Se calculan las entradas de cada capa de la red reuronal como evolución de las capas anteriores. "p" como índice de capa.

   $N_{pj}^{h} = \sum_{i=1}^{m} w_{ji}^{h} x_{pi} + b_{i}^{h}$ al aplicar la función de activación $y_{pj} = f_{j}^{h}(N_{pj}^{h})$

5. Se determina el error para todas las neuronas

   Salida Deseada – Salida obtenida $e = (d_{pk} - y_{pk})$

   cálculo de la delta (producto del error con la derivada de la función de activación con respecto a los pesos de la red) $\delta_{pk}^{0} = e * f_{k}^{0}(N_{pk}^{0})$

6. Actualización de los pesos empleando recursivamente el modelo de gradiente descendente

$$W_{kj}^{h}(t+1) = W_{ji}^{h}(t) + \Delta w_{ji}^{h}(t+1)$$

$$\Delta w_{ji}^{h}(t+1) = \min \delta_{pk}^{0} \, x_{pi}$$

7. Se cumple la condición de paro.

3.2 El paso a redes recurrentes

Al probar con distintas conexiones entre neuronas se descubrió un nuevo tipo de red artificial la cual incorpora el concepto de temporalidad con una recurrencia de su salida a su entrada como lo muestra la Figura 3.6, esta característica permite aprender secuencias.

Actualmente existen investigaciones sobre aplicaciones de estos modelos para el reconocimiento de voz como el propuesto por Alex Abdel y Geoffrey en [15], el cual presenta resultados favorables en

su problema. Fue aquí cuando se dio el auge en la implementación de redes recurrentes (las cuales son profundas) sobre todo en voz, pues cada quien proponía y probaba su propia arquitectura [18][19].

3.3 Arquitecturas comunes de RNN.

Puesto que el punto de partida es proponer una red recurrente propia, se indagara en los modelos más comunes de redes desarrolladas, con la finalidad de aprovechar las características ya publicadas de los mismos. Los siguientes modelos son los más comunes pero no únicos.

Red neuronal completamente recurrente (FRNN)

La Fig 3.3 muestra a la red neuronal completamente recurrente (Fully recurrent neuronal network por sus siglas en inglés FRNN) consta de dos capas, una capa de entrada de las unidades lineales y una capa de salida. La capa de entrada está completamente conectada a la capa de salida por los pesos ajustables. Además, la FRNN tiene conexiones de retardo de unidad que retroalimentan las activaciones de las unidades de salida a las unidades de la capa de entrada. Las unidades de salida por tanto, tienen un cierto conocimiento de sus activaciones anteriores, lo que permite que puedan desempeñar aprendizaje que se extiende en el tiempo. FRNNs logran su tarea de aprendizaje mediante la asignación de secuencias de entrada junto con el retraso de sus unidades, a otro conjunto de secuencias de salida. Debido a la naturaleza de la retroalimentación alrededor de las unidades de salida, estas unidades pueden continuar almacenando información del ciclo en la red a través de varios pasos de tiempo, y de este modo descubrir representaciones abstractas en el tiempo [16].

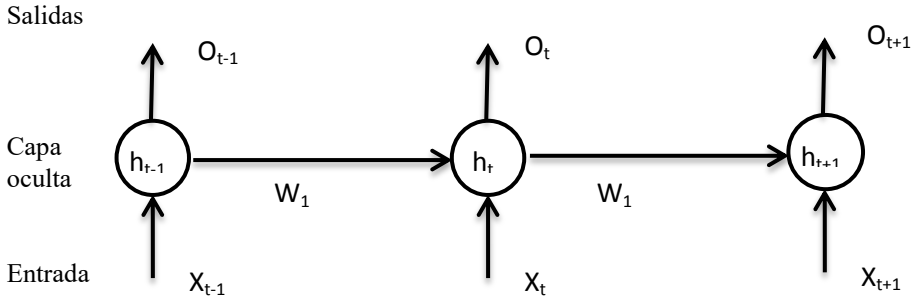

Figura 3.3 Arquitectura FRNN

Bidireccional RNN (BRNN)

La red bidireccional (Fig 3.4) es un tipo de arquitectura que a diferencia de las RNN comunes no solo usa el contexto pasado, sino usa el contexto futuro. Se aplica al reconocimiento de voz donde los enunciados se trascriben directamente.

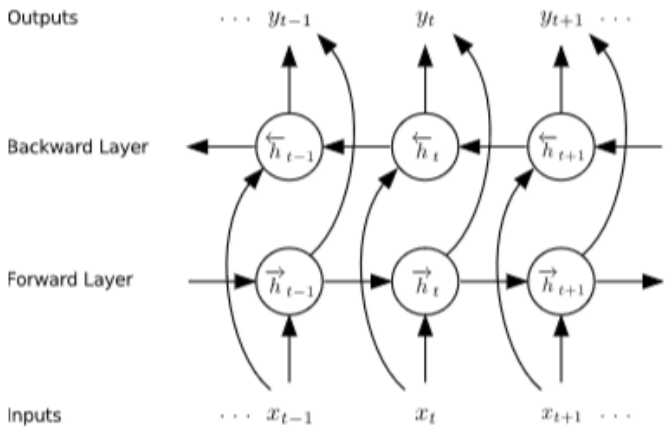

Figura 3.4 Arquitectura BRNN [5]

Larga memoria a corto plazo (LSTM)

Desarrollado por Hochreiter y Schmidhuber en 1997. Su estudio surge como solución a un problema con las RNN, el cual es, que a pesar de que las redes recurrentes pueden almacenar secuencias relativamente largas no funciona para almacenar los contextos en un periodo largo de tiempo, por ejemplo para el caso de voz no pueden reconocer el contexto al principio de una frase larga, sin embargo el LSTM (Long short term memory expuesto en la Fig 3.5) funciona incluso cuando hay grandes retrasos y puede manejar señales que tienen una mezcla de componentes de baja y alta frecuencia, esto por medio de activaciones de módulos donde se guarda la información muy atrasada para ser usada cuando se requiera.

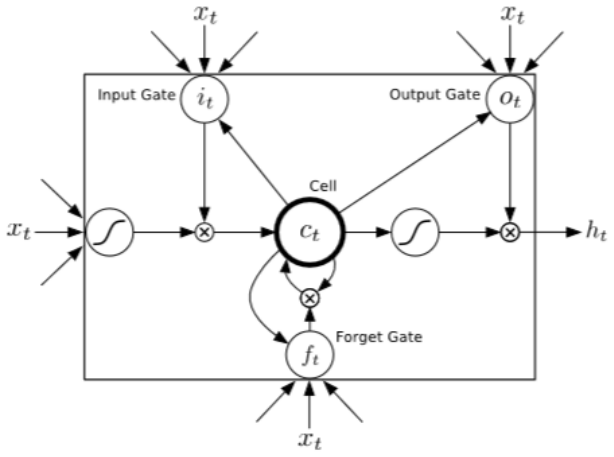

Figura 3.5 Arquitectura LSTM[6]

[5] Imagen de arquitectura BRNN tomada del trabajo de investigación de Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton [15].

[6] Imagen de arquitectura LSTM tomada del trabajo de investigación de Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton [15].

3.4 La paquetería Theano

Theano[7] es una biblioteca de Python que permite definir, optimizar y evaluar expresiones matemáticas que involucran matrices multidimensionales de manera eficiente. Theano ha sido capaz de impulsar a gran escala investigaciones científicas computacionalmente intensivas desde el año 2007. Durante la última década se han implementado métodos de aprendizaje profundo así como funciones para trabajar con redes neuronales, por lo cual se ha convertido en una herramienta usada en las investigaciones con redes neuronales artificiales.

En cuanto a redes neuronales recurrentes, Theano ya cuenta con bibliotecas optimizadas con arquitecturas de redes conocidas sin embargo no se usarán ya que se requiere crear una arquitectura propia con conexiones específicas. La definición de dicha arquitectura tendrá que ser declarada en lenguaje de alto nivel (Python) puesto que no existe una versión aún para lenguajes como C o C++ disponible al público en general. Por tanto la ejecución del algoritmo puede no ser óptima ya que no existe una gran manipulación de recursos de optimización como por ejemplo gestión de la memoria.

El código de implementación en Theano se presenta en el apéndice de la tesis, cabe aclarar que cuando se trata de aprendizaje y entrenamiento con redes neuronales es muy recomendable contar con una tarjeta de video (GPU) para reducir tiempos de ejecución.

La máquina en la cual se ejecutarán todas las pruebas cuenta con las siguientes características: Equipo de cómputo portátil con procesador Intel® Core i5 tercera generación con reloj a 2.5 Mhz, memoria ram 6 GB, tarjeta de video Nvidia GeForce GT 640M LE con reloj 500 Mhz.

3.6 Concepto de RNN.

El núcleo de las RNNs tienen una característica muy simple que las separa de las demás redes: los contenidos de manera crucial del vector de salida se ven afectados no sólo por la entrada, sino también por toda la historia de entradas que se ha alimentado en el pasado por medio de una retroalimentación de sus valores de salida (retraso a la salida).

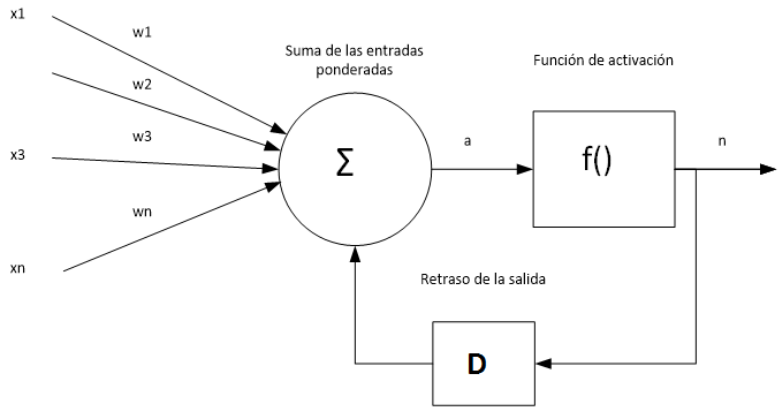

Figura 3.6 Unidad básica de la red neuronal recurrente, neurona feed-back.

[7] Theano® tecnología tomada de http://deeplearning.net/software/theano/

Programar dicho núcleo en theano resulta relativamente sencillo: se multiplica el vector de entrada *x* por la matriz de pesos *W_x*, *h* es el vector oculto que tiene la historia de la salida de la neurona que también se multiplica por su matriz de pesos correspondiente *W_h*, se suman ambas señales y se someten a una función de activación *Tanh* (una función no lineal que obliga a los valores a estar en un rango de [-1 1])[8].

```
h = np.tanh(np.dot(W_h, h) + np.dot(W_x, x))
```

Posteriormente el vector adquirido, se multiplica por una matriz de pesos *W_hn* , la cual es la salida de la neurona *n* para ese momento, pero también se retorna la información para retroalimentar el pasado de la neurona al método en la siguiente ejecución (será la nueva h). *np.dot* es la operación entre matrices.

```
n = np.dot(W_hn, h)
return n
```

Los resultados de las pruebas de concepto se presentan en el capítulo 5.

Antes de dar el siguiente paso el cual es incrementar las capas y alimentar las redes con datos de voz, se requiere acondicionar las señales de voz que servirán como alimentación para este entrenamiento.

3.7 Acondicionamiento señal de voz

Para acondicionar la señal de voz, la cual será la entrada a la red, se usará la metodología convencional que es más que recurrente en el estado del arte de sistemas que trabajan con voz, practicante son un estándar en el manejo de voz.

Señal de voz

La señal de voz es el objetivo a procesar y se puede interpretar como un conjunto de señales sonoras las cuales oscilan en determinada frecuencia. Son el resultado de una combinación de una señal de alimentación generada por los pulmones y modificada por el tracto bucal. Esta señal es digitalizada por un micrófono el cual muestrea la señal a determinada frecuencia para finalmente tener como resultado una señal discreta digital en el tiempo.

[8] Cita textual tomada de WILDML RECURRENT NEURAL NETWORKS TUTORIAL, PART 1 – INTRODUCTION TO RNNS, http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

Transformada de fourier de término reducido STFT

La transformada de Fourier es la representación más recurrente con la cual se comienza a trabajar los datos de voz para su procesamiento. Pero existe una modificación a la técnica la cual nos proporciona más información y es la transformada de Fourier de término reducido (*short-term Fourier transform*, STFT) usada para determinar el contenido en frecuencia sinusoidal y de fase en secciones locales de una señal así como sus cambios con respecto al tiempo.

La información a ser transformada podra ser dividida en pedazos o tramas (que usualmente se traslapan unos con otros, para reducir irregularidades en la frontera). Cada pedazo forma una transformación de Fourier, y el resultado se agrega a una matriz, que almacena magnitud y fase para cada punto en tiempo y frecuencia. Esto se puede expresar en la ecuación 3.1.

$$STFT\{x[n]\} = X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \quad \ldots (3.1)$$

Donde, *x[n]* es la señal y *w[n]* es la ventana. En este caso *m* es discreta y ω es continua, pero en la mayoría de aplicaciones típicas la STFT se hace usando la Transformada Rápida de Fourier, así ambas variables son discretas y cuantizadas. De nuevo, el índice de tiempo discreto *m* es normalmente considerado como un tiempo "lento" y usualmente no se expresa con tan alta resolución como con el tiempo *n*. El cuadrado de la STFT se le conoce como espectrograma (ecuación 3.2):

$$espectograma\{x(n)\} = |STFT|^2 \quad \ldots . (3.2)$$

Extracción de información de la señal de voz para RNN

Se usa la tarjeta integrada de audio de la computadora para hacer la adquisición de las señales de voz con una resolución del ADC de 16 bits (la máxima otorgada por la tarjeta) y para el muestreo se usa una frecuencia de 16 kHz (dato establecido por convención). En la figura 3.6 se muestra la digitalización de la letra “i”.

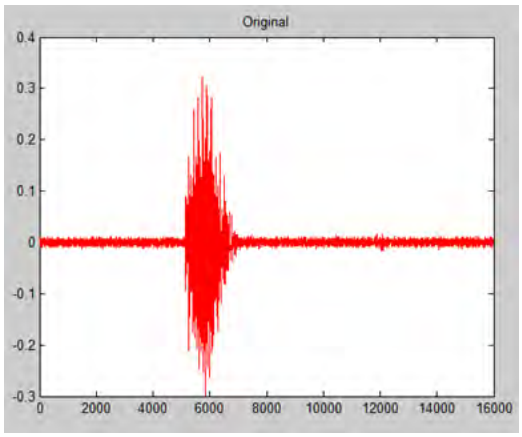

Figura 3.6 Digitalización de una letra

También se usa un filtro de pre-énfasis cuya tarea será amplificar las frecuencias altas. En la Fig 3.7 se observa el resultado de aplicar pre-enfasis a la letra "i" (filtro paso altas) cuya transformada z es:

$$H(z) = 1 - 0.95z^{-1}$$

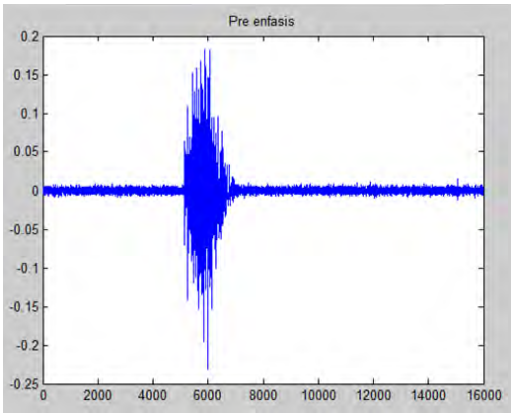

Figura 3.7 Pre-énfasis

STFT con ventanas de Hamming

Se usan ventanas de 512 con corrimientos de 256, a su vez estas ventanas serán tratadas con subventanas de Hamming de 256 muestras. Posteriormente se obtiene la trasformada de Fourier para dicha subventana y así sucesivamente para crear el espectrograma de la señal.

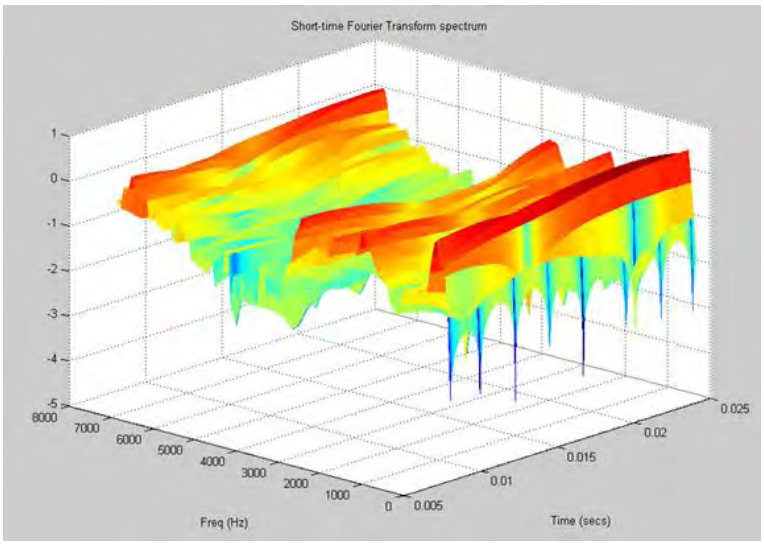

Figura 3.8 STFT de la palabra

Para finalmente tener su representación como imagen de la señal mostrada en la Fig 3.9, cuyos valores de magnitud de intensidad serán la entrada a la red neuronal.

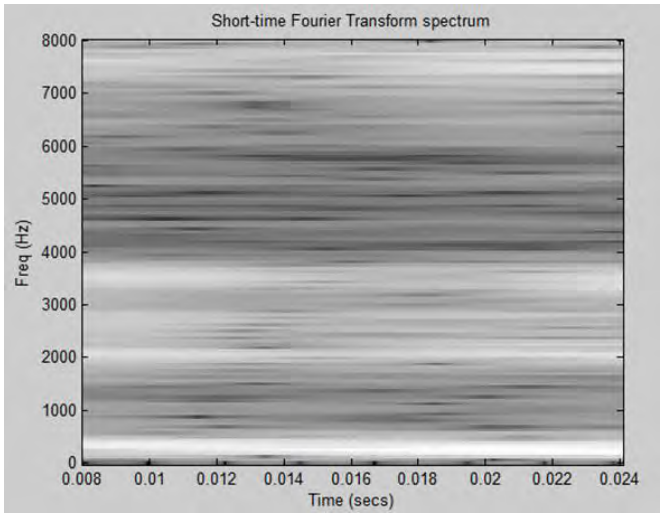

Figura 3.9 imagen de la STFT

Toda esta etapa de preparación de la señal de voz y su alimentación a la red neuronal se puede apreciar en la Fig 3.10. Cada bloque de la señal (512 datos) es pre-procesada, después se aplica transformada de Fourier no sin antes hacer un ventaneo de la señal, lo anterior se mapea a un instante de tiempo en dominio de las frecuencias el cual es un vector columna de (256 datos). Este

vector es la entrada a la red, por tanto la red avanza a lo largo del tiempo alimentándose de cada columna de la STFT.

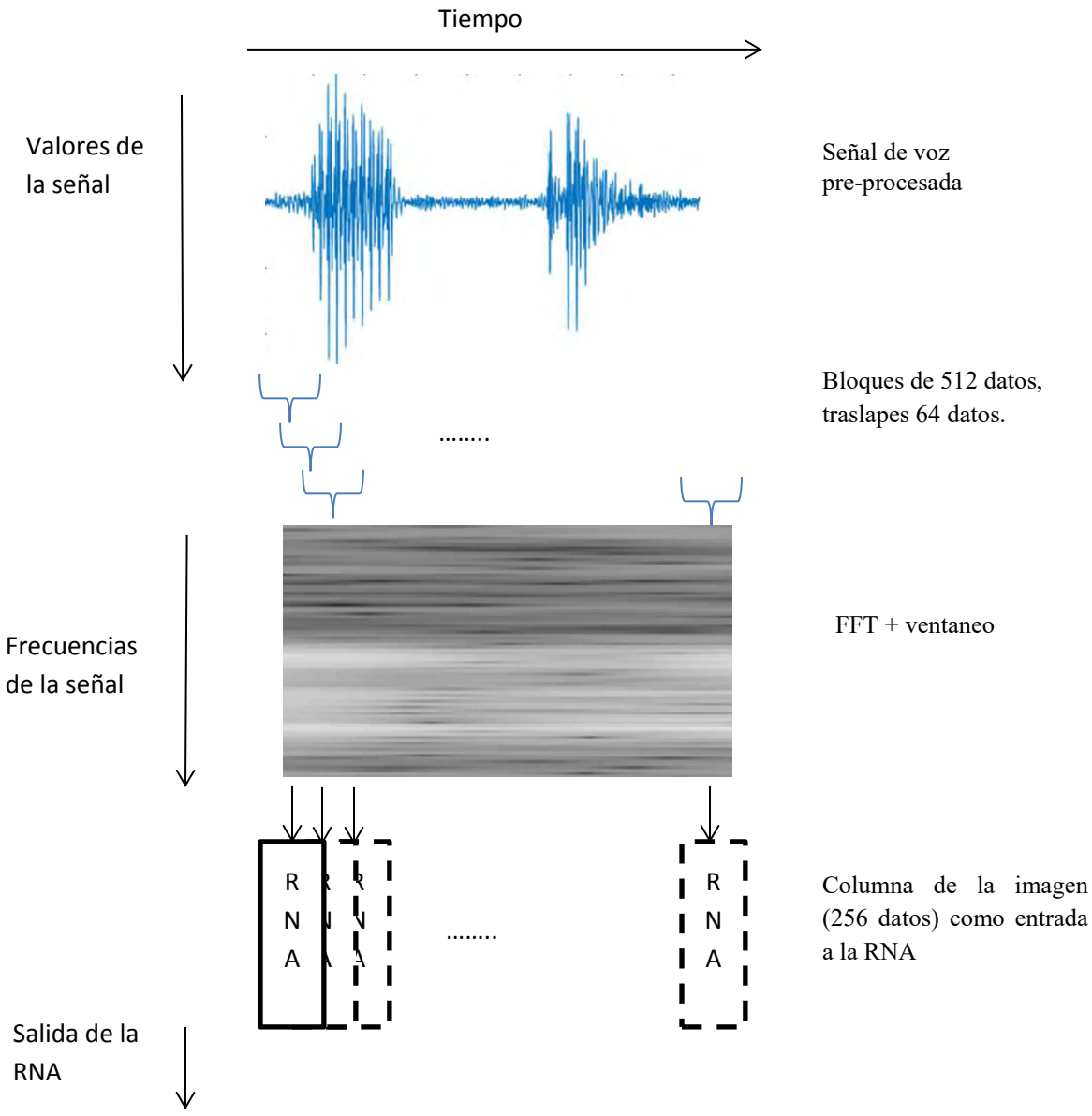

Figura 3.10 Metodología para alimentar la señal de voz a la red neuronal

3.8 Redes Recurrentes en el reconocimiento de palabras aisladas

Con la señal acondicionada se procede a programar una arquitectura común de redes recurrentes: red completamente recurrente que recibe como entrada X_t la STFT de la señal de voz (Figura 3.11), para ser multiplicada por una matriz de pesos *W1*. Posteriormente con una función de activación *tanh* se genera la salida $h1_t$ en la primera capa, la cual es recurrente. En la segunda capa se repite lo mismo, se comparten las salidas de la capa 1 a las entradas de la capa 2 (la relación es la misma en toda la red, todos contra todos) por medio de una matriz *W2* para generar con una función de activación *tanh* la salida $h2_t$. Finalmente se llega a la capa de salida o clasificación Y_t, esto se hace multiplicando la salida anterior $h2_t$ por una matriz *W3*. La red servirá como base de modificación para generar la arquitectura propia y también como red de comparación en la etapa de resultados.

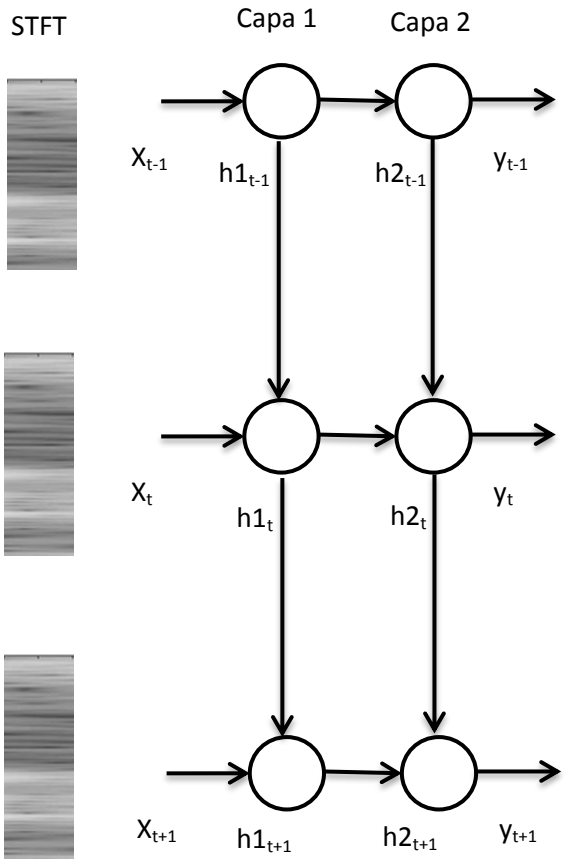

Figura 3.11 Despliegue en el tiempo de la red neuronal recurrente la cual recibe como entrada la STFT de la señal de voz.

Hasta este punto se ha probado el concepto de redes recurrentes, se ha acondicionado las señales de entrada y se ha probado el modelo más común de red recurrente (Resultados en el capítulo 5), el siguiente paso es la propuesta de una arquitectura dedicada para el reconocimiento de palabras claves.

3.8 Diseño de una arquitectura de red recurrente para el reconocimiento de palabras claves

La propuesta de una arquitectura podría ser aleatoria incluso usando códigos genéticos, mutando a una red común, sin embargo lo que se intenta es trasladar una solución existente en microprocesadores al ámbito de redes neuronales.

Propuesta de red neuronal.

Las señales de voz contienen datos temporales, que son secuencias que dependen de sus datos pasados para formar un fonema o silaba, etc., se podría hacer una analogía a los estados de una *máquina de estados*. En las *máquinas de estado,* el estado siguiente está determinado por una combinación del estado pasado y sus entradas, se podría asemejar esta restricción a las de las señales de voz. Siguiendo la analogía de las *máquinas de estado*, existe una solución implementada con hardware, siendo un tipo de arquitectura para direccionar la memoria dentro de un microprocesador como el de la Figura 3.12. Dicha arquitectura fue creada y probada para la ejecución de máquinas de estados y forma parte del estado del arte del diseño de microprocesadores [20].

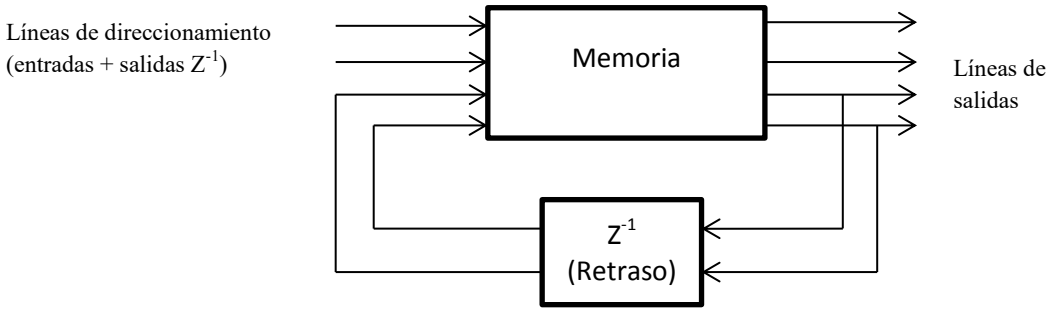

Figura 3.12 Direccionamiento de memoria usando entradas al sistema y parte de las salidas del estado presente

De igual forma se propone una arquitectura análoga a los componentes de Figura 3.12, pero con sus equivalentes en redes neuronales. Esta arquitectura propuesta en la Fig 3.13, será el sistema a

entrenar e implementar. Recibe como vector de entrada X_t a la STFT de la señal de voz, la cual se concatena con parte de las salidas de la penúltima capa de un instante pasado $h2_{t-1,}$ y se multiplican por una matriz de pesos *W1* para generar la salida $h1_t$. Estas salidas se comparte nuevamente con todas las neuronas en la capa 2 pero no sin antes multiplicarse por una matriz de pesos *W2* para generar la salida $h2_t$ para finalmente pasar por otra multiplicación por una matriz *W3* y generar la capa de salida Y_t que es la capa clasificadora. Para emular la unidad de memoria se hace una recurrencia entre neuronas de la primera capa.

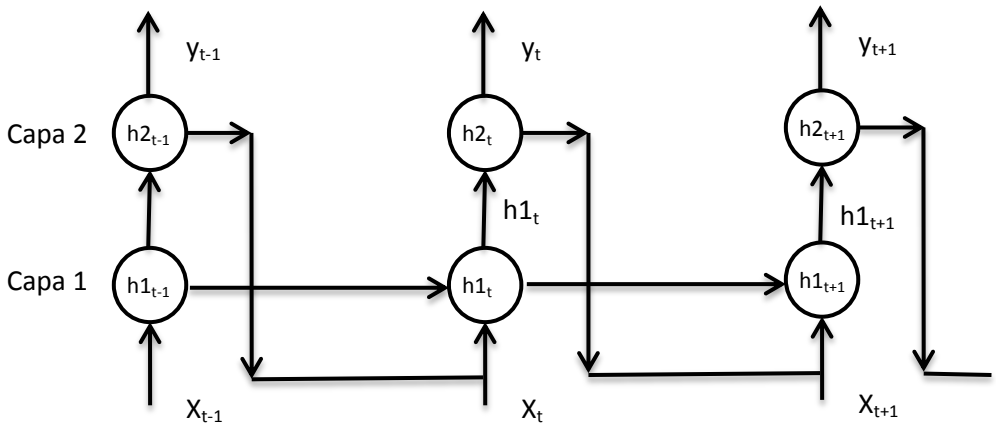

Figura 3.13 Modelo propuesto de red recurrente análogo a una arquitectura de máquina de estados desplegado en el tiempo.

Con la implementación del sistema de palabras claves en el módulo de audio del robot Justina, se podría ahorrar recursos y batería, pues el sistema puede ejecutarse en un procesador de bajo consumo todo el tiempo, sin restricciones de un sistema operativo en específico. De esta forma servirá como apoyo para saber cuándo el usuario llama al robot y entonces interactuar con el usuario usando el reconocedor de voz de Microsoft.

Sin embargo es necesario proporcionarle al robot un nivel de interacción mayor con el usuario, pues el humano frecuentemente reacciona a un estímulo auditivo girando y observando a la dirección de la fuente de sonido, de igual forma el robot requiere poder estimar la dirección de arribo del parlante que detecta y poder reaccionar. Esta implementación se desarrollara en el siguiente capítulo.

Capítulo 4

Dirección de arribo de la señales usando arreglo de sensores

4.1 Arreglo de sensores lineales

Para estudiar la estimación en la dirección de arribo de una señal sobre un arreglo de sensores, se parte de la teoría general de arreglo de antenas, donde se tiene en vez de una onda electromagnética una onda mecánica de presión y como receptores en vez de tener antenas se tienen micrófonos. La dinámica del sistema es el mismo ya que se percibe la misma señal en todos los sensores del arreglo pero con un desfase en el tiempo.

4.2 Modelo de arreglo de micrófonos lineales

Se parte de un arreglo de dos sensores (mic1 y mic2) y una fuente de sonido , El sistema de referencia se encuentra en el centro del arreglo de micrófonos.

Modelo Campo Cercano

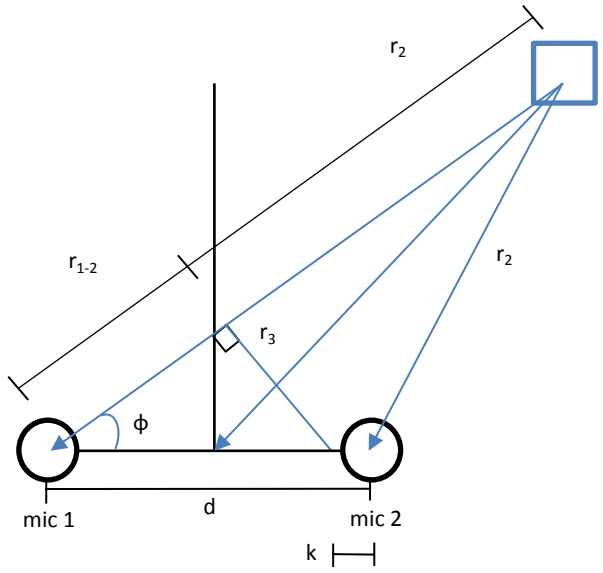

Fig 4.1 Modelo campo cercano del sonido[9]

[9] Figura de modelo tomado del material didáctico de la materia "Audición Robótica" en posgrado de Ciencias e Ingeniería de la computación por Dr. Caleb Rascón UNAM

Como se puede observar en la Fig 3.6, para un instante de tiempo de captura de la fuente de sonido, al usar geometría euclidiana se construye un triángulo rectángulo. Se parte del supuesto que se conoce el desfase de la señal, esto, por medio del tiempo que tarda la señal en llegar al micrófono 1 ya que el micrófono 2 es el micrófono de referencia. Por lo tanto conocemos indirectamente esta delta $r_{1\text{-}2}$, lo que nos interesa conocer es el ángulo ϕ, d es la distancia física entre los micrófonos, todas las r´s son desconocidas además existe una distancia k entre los micrófonos la cual es desconocida y no se puede determinar por el modelo, en conclusión el modelo considerado tiene muchas incógnitas para poder ser resuelto con la información disponible.

Modelo Campo Lejano

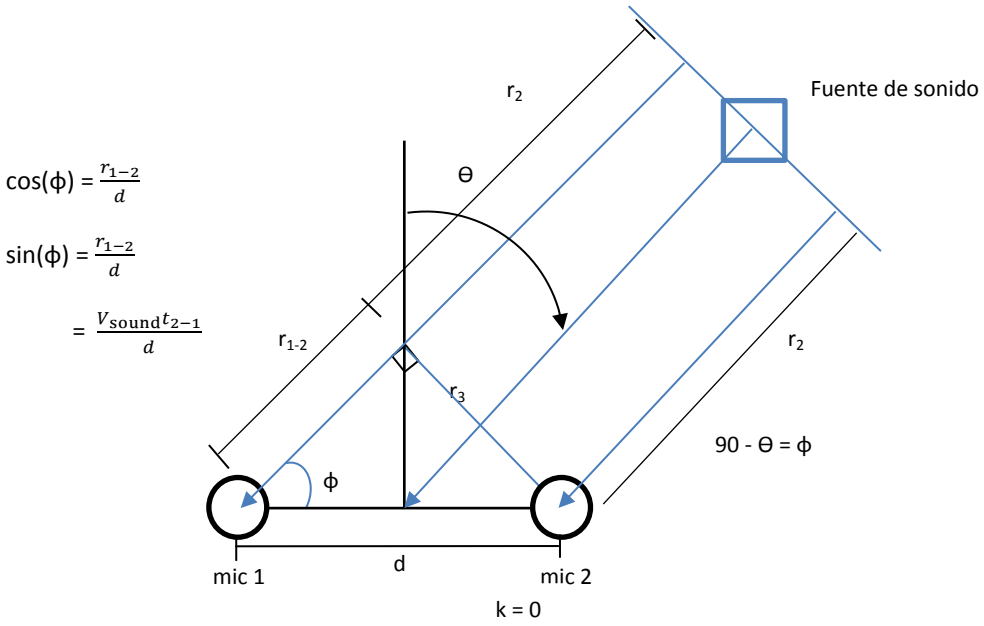

Fig 4.2 Modelo campo cercano del sonido[1]

Si se observa a la fuente de sonido como una fuente muy lejana[10], esto es $d<<D$ (donde D es la distancia del centro del arreglo de micrófonos a la fuente de sonido y d distancia entre micrófonos), se observaría que las señales emitidas se convierten en ondas planares por tanto estas ondas serían perpendiculares a una recta que cruza por la fuente de sonido, esto permite una corrección en el triángulo rectángulo dejando a la distancia d como la distancia única entre la señal captada por el

[10] Esta distancia varía dependiendo el autor, mientras para algunos es 10 veces más, para otros es 20 veces más la distancia entre micrófonos.

micrófono 1 y 2. El ángulo ϕ puede ser determinado a partir de la relaciones del triángulo rectángulo, más concretamente sobre su cateto (ecuación 4.1).

$$\cos(\phi) = \frac{r_{1-2}}{d} \quad \ldots (4.1)$$

De esta forma se tiene expresada la distancia $r_{1\text{-}2.}$ Si se observa, esta distancia es la diferencia de distancia que tarda en llegar del micrófono 2 al micrófono 1, adicionalmente si se parte del echo que se conoce el tiempo de desfase entre señales y se conoce la velocidad del sonido, se remplazan los datos conocidos en ecuación 4.1 y se tiene la ecuación 4.2.

$$\cos(\phi) = \frac{V_{sonido} t_{1-2}}{d} \quad \ldots (4.2)$$

Finalmente si el sistema está referido con respecto al centro del arreglo de micrófonos, el ángulo a determinar es Θ. De la geometría del modelo se tiene que 90-Θ=ϕ , por propiedades de los ángulos dobles se tiene la ecuación 4.3:

$$\sin(\theta) = \frac{V_{sonido} t_{1-2}}{d} \ldots (4.3)$$

Desfase en la señal

Digitalmente se tiene la siguiente captura en cada micrófono

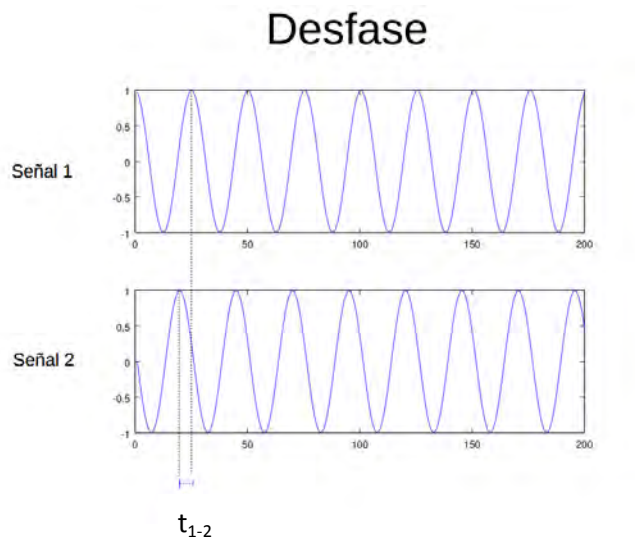

Fig 4.3 Desfase de una misma señal en diferentes sensores[11]

[11] Figura tomado del material didáctico de la materia "Audición Robótica" en el posgrado de Ciencias e Ingeniería de la computación por Dr. Caleb Rascón UNAM.

Para tener todos los datos disponibles ahora se debe determinar el tiempo de desfase de la señal con algún algoritmo tal como correlación-cruzada, algún tipo de formación de emisión (beamforming), MUSIC, etc. No existe un método absoluto para la detección del desfase en la señal y por ende de la dirección de arribo, pues cada método tiene ventajas y desventajas por tanto queda a criterio del diseñador la selección del algoritmo.

4.3 Concepto de formación de emisión (Beamforming)

En un arreglo de sensores, generalmente la información capturada se coloca en una matriz de información, esta información contiene la propagación espacial desde una dirección determinada. Posteriormente esta información se procesa, lo que se pretende es combinan las señales de todos los sensores con coeficientes para estimar la dirección desde donde se transmiten los datos. A esta operación se le conoce como formación de emisión *Beamforming* [21]. Mediante el cálculo de una suma debidamente ponderada de las señales de los sensores individuales como un filtro finito de respuesta de impulso (FIR) para generar una salida (a una frecuencia de interés) que es la suma ponderada de muestras en el tiempo, podría verse como un filtro que atenúa las otras direcciones [22].

Este concepto ha dado lugar a muchos algoritmos que se basan en la definición, usan una matriz de datos la cual se pondera con coeficientes fijos o adaptativos, siendo este concepto otro punto de ramificación de algoritmos Beamforming pues existen los convencionales (delay and sum, Mínimum Variance Distiortionless Response, etc) y los adaptativos (adaptive beamforming) [23].

Un algoritmo *beamforming* convencional es el Minimimum Variance Distiortionless Response (MVDR) el cual fue seleccionado para su implementación por sus características (análisis de la señal en la frecuencia, desempeño con respecto al número de micrófonos, robustez ante las interferencias conocidas como ruido, y complejidad para su tiempo de ejecución).

4.4 Mínimum Variance Distiortionless Response (MVDR)

Se asume que hay una o varias señales de interés (SOI) tal que:

$$A=\begin{bmatrix} e^{-i2\pi fT_{1:1}} & e^{-i2\pi fT_{1:2}} & \cdots & e^{-i2\pi fT_{1:M}} \\ e^{-i2\pi fT_{2:1}} & e^{-i2\pi fT_{2:2}} & \cdots & e^{-i2\pi fT_{2:M}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi fT_{D:1}} & e^{-i2\pi fT_{D:2}} & \cdots & e^{-i2\pi fT_{D:M}} \end{bmatrix} \qquad S=\begin{bmatrix} s_1(1) & s_1(2) & \cdots & s_1(N) \\ s_2(1) & s_2(2) & \cdots & s_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ s_M(1) & s_M(2) & \cdots & s_M(N) \end{bmatrix}$$

$$X=S\,A$$

[12]

[12] Imagen tomado del material didáctico de la materia "Audición Robótica" en posgrado de Ciencias e Ingeniería de la computación por Dr. Caleb Rascón UNAM

X: señales capturadas por los micrófonos, se tiene un renglón por cada uno.
S: matriz de contiene las señales de origen
N: tamaño de la señal (número de muestras de forma discreta)
$T_{d:m}$: es el retraso en la señal s_m en el micrófono d
A: matriz que contiene los vectores de dirección

El objetivo es usar una matriz de modificación denominada *steering vector* W a la señales de entrada X para estimar las señales de origen S^.

$$X=\begin{bmatrix} x_1(f_1) & x_1(f_2) & \cdots & x_1(f_N) \\ x_2(f_1) & x_2(f_2) & \cdots & x_2(f_N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(f_1) & x_M(f_2) & \cdots & x_M(f_N) \end{bmatrix} \quad W=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-i2\pi f_1 T_1} & e^{-i2\pi f_2 T_1} & \cdots & e^{-i2\pi f_N T_1} \\ e^{-i2\pi f_1 T_2} & e^{-i2\pi f_2 T_2} & \cdots & e^{-i2\pi f_N T_2} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f_1 T_M} & e^{-i2\pi f_2 T_M} & \cdots & e^{-i2\pi f_N T_M} \end{bmatrix}$$

13

$$\hat{S}=W\,X$$

X: matriz de las señales capturadas transformadas con FFT
W: steering vector
T_m: es el desfase apropiado para cada micrófono m
f_n: la frecuencia n

El método intenta minimizar la energía de las interferencias por medio de filtros direccionales por cada frecuencia. Esta minimización la provee por medio del cálculo de un *steering vector* A diferente al steering vector W pero basado en W [24].

4.5 Especificaciones de la implementación.

La implementación está pensada para el robot de servicio Justina (perteneciente al departamento de Bio-Robótica FI-UNAM en su versión de actualización de hardware y software 2016) basándose en las condiciones de respuesta impuestas por el Rulebook de la Robocup® 2016. Las del Rulebook condiciones exponen que el robot debe girar entre 0 y 360° por lo cual se debe proponer un sistema para resolver este problema.

La implementación correrá en una tarjeta Raspberry Pi 2® para no consumir recursos de la computadora principal en el robot de servicio pues ya tiene bastantes algoritmos complejos ejecutándose.

[13] Imagen tomado del material didáctico de la materia "Audición Robótica" en posgrado de Ciencias e Ingeniería de la computación por Dr. Caleb Rascón UNAM

El modulo deberá ser compatible con ROS (Robot Operating System), que es un software de herramientas y librerías para la comunicación de módulos así como intercambio de información entre los mismos para aplicaciones robóticas. Deberá ejecutarse en tiempo real desde la perspectiva de un ser humano (1 a 2 segundos de respuesta).

Para la implementación del algoritmo se usa el lenguaje Python junto con la librería pyaudio, la ventana de datos para procesar será de un tamaño de 3072 el cual se lanza en hilos pues el recurso donde se ejecutara el modulo será en un tarjeta raspberrypi 2.

Se usan tarjetas de audio usb con un costo de $25 mexicanos (precio 2016) con frecuencia de muestreo 44100 hz, 1 canal, y velocidad de transmisión de datos USB 2.0.

Fig 4.4 Tarjeta de audio USB implementada (3D Sound).

Los micrófonos usados son de condensador, extraídos de auriculares comerciales (de una misma marca) atractivos por su tamaño y precio ($45 MXN en el 2016). Son muy populares por su sensibilidad y relación costo/producción. El micrófono requiere energización externa para funcionar y su construcción como encapsulado entorpece la captura de audio ya que si la parte frontal del sensor no apunta a la fuente sonido, esta se deteriorara un poco en la captura. A un así con estas dificultades, su calidad de captura es suficiente para desarrollar el módulo de detección de fuente de sonido.

Fig 4.5 Micrófono de condensador, muy común en los auriculares comerciales.

Propuesta para detectar ángulos entre 0 y 360°.

El modelo de la Fig 4.6 muestra el arreglo de micrófonos lineales con dos fuentes (original y espejo).

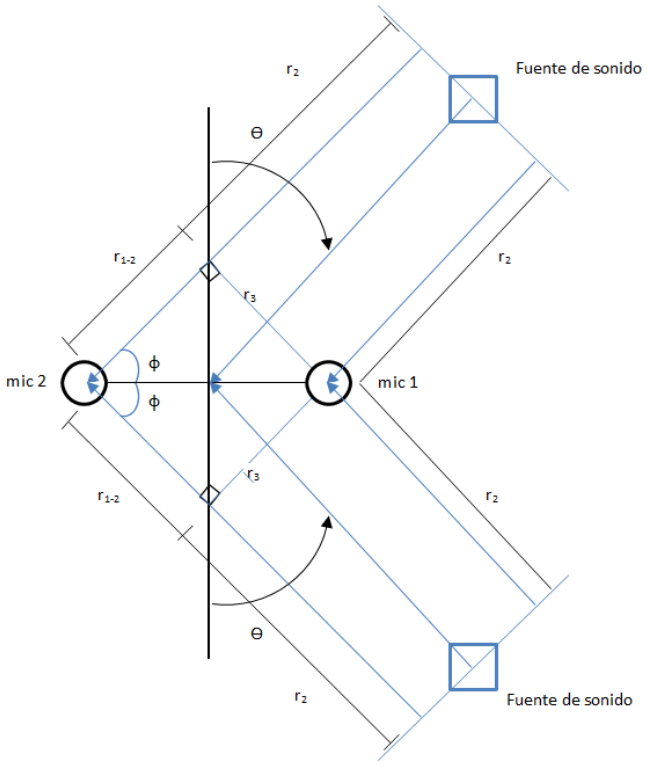

Fig 4.6 Posibles soluciones para un arreglo lineal

Se puede observar que existen dos soluciones que cumplen con la ecuación 4.4.

$$\cos(\phi) = \frac{V_{sonido} t_{1-2}}{d} \quad \ldots (4.4)$$

Por lo tanto el sistema puede identificar falsas fuentes de sonido (pues pueden ser el complemento en espejo de la fuente verdadera de sonido). Entonces se necesita una tercera referencia para poder resolver el problema de la detección de arribo del sonido pues el robot necesita de esta resolución.

Existen varias soluciones al problema, la que se propone a continuación es pensando en los recursos del robot. Se propone dos sistemas con 2 sensores (uno de ellos compartidos), por lo tanto, se utiliza una inferencia entre dos sistemas de micrófonos siendo dicho modelo a implementar el de la figura 4.7:

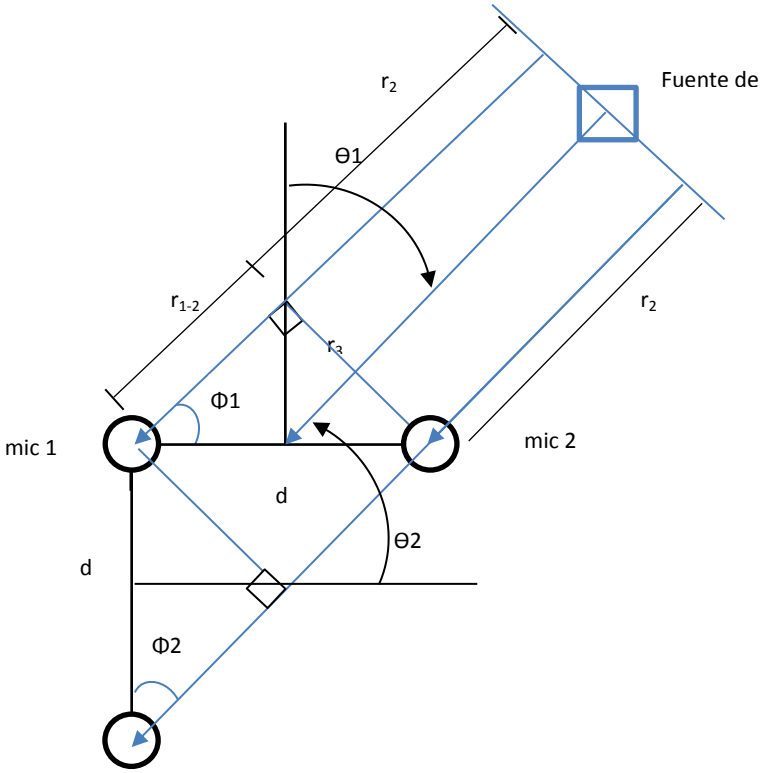

Fig 4.7 Arreglo de micrófonos propuesto.

El micrófono 1 y 2 forman el primer sistema, mientras que el sistema 1 y 3 será el discriminante para determinar el ángulo de solución (recordar que el modelo se sigue basándose en campo lejano).

Ejemplo: el sistema 1 y 2 devuelven como solución 45° y por ende el complemento en espejo 315°, y el segundo sistema 1 y 3 devuelve 150° y por ende el espejo 330°, dado que la orientación del sistema 2 es con dirección hacia arriba (con respecto a la figura 4.6) el ángulo solución es 45° entregado del sistema 1.

Una vez que el sistema obtenga la dirección de arribo de la fuente de audio y comparta dicha información, el planeador del robot solo tendrá que hacer girar la estructura del robot para hacer que entre en funcionamiento el reconocedor de voz de Microsoft y así tratar de solventar el problema cuando un usuario no es escuchado por el micrófono frontal del robot a la vez de proporcionarle un nivel de interacción mayor con el usuario.

Capítulo 5

Pruebas y Resultados.

A continuación se detallan los resultados comentados durante el capítulo 3 y 4.

5.1 Prueba de concepto de RNN.

La premisa de las redes recurrentes es que funcionan para aprender secuencias, por tanto se procede a hacer pruebas de escritorio con el núcleo de las redes recurrentes, la neurona recurrente que se observa en la Fig 3.6, la cual se motera a patrones de prueba distintos a los de entrenamiento con el fin de observar su respuesta.

Modelos de entrenamiento.

Se presentan los modelos con los cuales se entrena a la neurona, que son secuencias de 3 estados, las entradas y salidas son números decimales.

El primer modelo es la secuencia es 0,10,10 y se clasificara como la secuencia de salida 0,0,1. Mientras tanto el modelo 2 es la secuencia 0,0,30 y se clasificara como la secuencia de salida 0,0,0.

	Modelo 1		Modelo 2	
Estado de la secuencia	Entrada	Salida	Entrada	Salida
1	0	0	0	0
2	10	0	0	0
3	10	1	30	0

Modelos de prueba.

Se agregan valores aleatorios al primer modelo de entrenamiento para generar modelos de prueba, de esta forma se puede observar la respuesta de la neurona recurrente a las secuencias de entrada con las cuales no fue entrada.

	Modelo de prueba 1		Modelo de prueba 2		Modelo de prueba 3	
Estado de la secuencia	Entrada	Salida	Entrada	Salida	Entrada	Salida
1	0	0.01	0	0.01	0	0.01
2	15	0.09	3	0.09	15	0.09
3	3	9.7	20	9.7	15	9.9
Modelo de clasificación	Modelo 1		Modelo 1		Modelo 1	

Modelo de prueba 4		Modelo de prueba 5	
Entrada	Salida	Entrada	Salida
0	0.000	0	0.000
1	0.000	2	0.000
15	0.003	30	0.003
Modelo 2		Modelo 2	

Al inspeccionando los modelos de alimentación (Modelos de prueba) a la red se observa que la clasificación echa al modelo de prueba 1, modelo de prueba 3, modelo de prueba 4 y modelo de prueba 5 es correcta ya que los valores son muy cercanos a los modelos de clasificación que arroja la red (Modelos de clasificación 1 y 2) usando la distancia de Hamming para comparar patrones de prueba con entrenamiento, sin embargo para el modelo de prueba 2 su clasificación es incorrecta pues la secuencia debería corresponder al modelo 1 de entrenamiento.

Reforzando entrenamiento.

Para reforzar el entrenamiento a la red se toma el conjunto de prueba de interés (Modelo de prueba 2), para corregir la respuesta de la red se agrega este como modelo de entrenamiento y se re-entrena por completo la red. A esta forma de entrenamiento se denomina entrenamiento supervisado ya que alguien ajeno a la red le indica la corrección de clasificación de los patrones o en este caso modelos de entrada.

	Modelo de prueba 1		**Modelo de prueba 2**		Modelo de prueba 3	
Estado de la secuencia	Entrada	Salida	**Entrada**	**Salida**	Entrada	Salida
1	0	0.01	**0**	**0.00**	0	0.01
2	15	0.09	**3**	**0.03**	15	0.09
3	3	9.5	**20**	**0.08**	15	9.7
Modelo de clasificación	Modelo entrenamiento 1		**Modelo entrenamiento 2**		Modelo entrenamiento 1	

Modelo de prueba 4		Modelo de prueba 5	
Entrada	Salida	Entrada	Salida
0	0.000	0	0.000
1	0.000	2	0.000
15	0.003	30	0.003
Modelo entrenamiento 2		Modelo entrenamiento 2	

Se puede apreciar que la corrección esta hecha, sin embargo se comprueba que las redes neuronales dependen mucho de su base de entrenamiento para clasificar correctamente por lo cual en las pruebas con voz se requerirá de varias pruebas de entrenamiento para una misma palabra.

5.2 Pruebas reconocimiento de RNN para palabras aisladas.

Ahora se programa una red completamente recurrente el modelo correspondiente es el de la Fig 5.3, esta red por tanto, presenta recurrencia en sus neuronas de cada capa y se comparte información de todas contra todas, así pues, de un corpus de 16 palabras para la palabra aislada *gato* (Fig 5.2), se utilizara 10 de estos archivos de audio aplicando la técnica de procesamiento a la señal de entrada y alimentando a la red recurrente para su entrenamiento. Para crear la imagen a partir del espectro de Fourier se usaran bloques de 64 muestras y alimentando 8 de estos bloques procesados. El estándar marca que debe existir una neurona de salida por cada familia o clase que se clasifica sin embargo solo para efectos de pruebas se manejara como codificación binario solo para observar la reacción de la red.

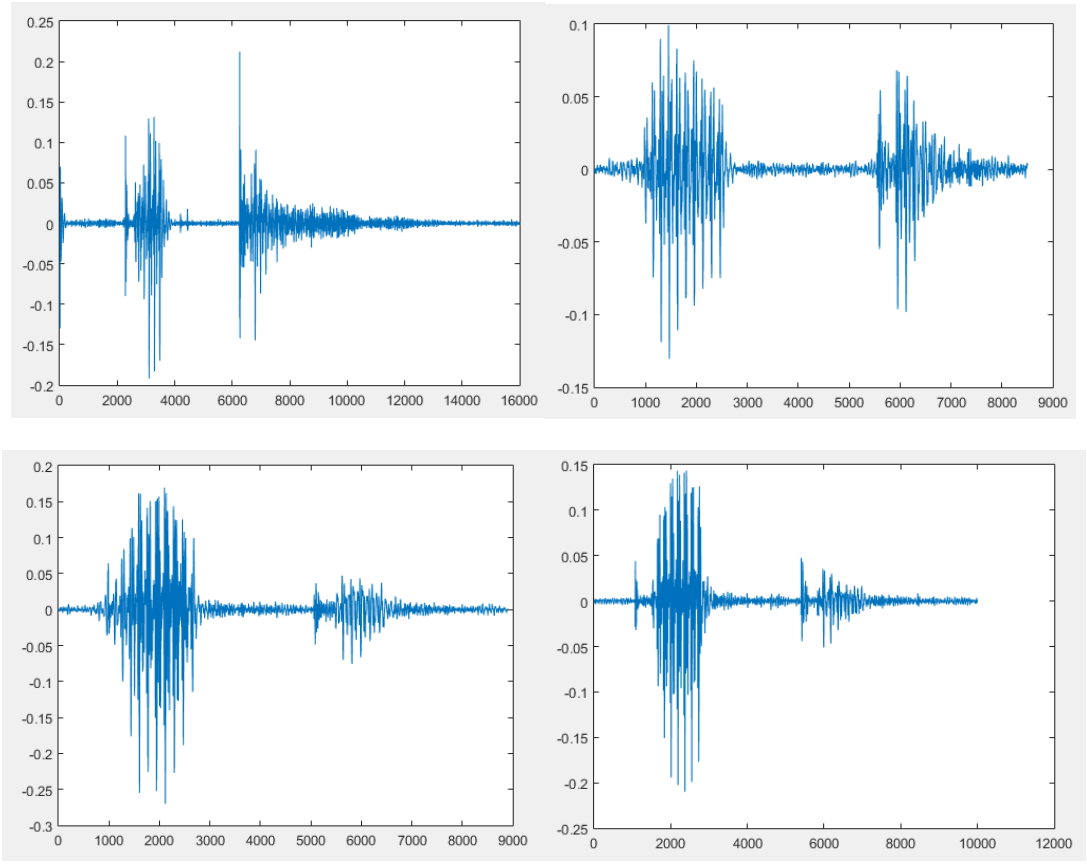

Figura 5.2 Muestras de una misma palabra "gato"

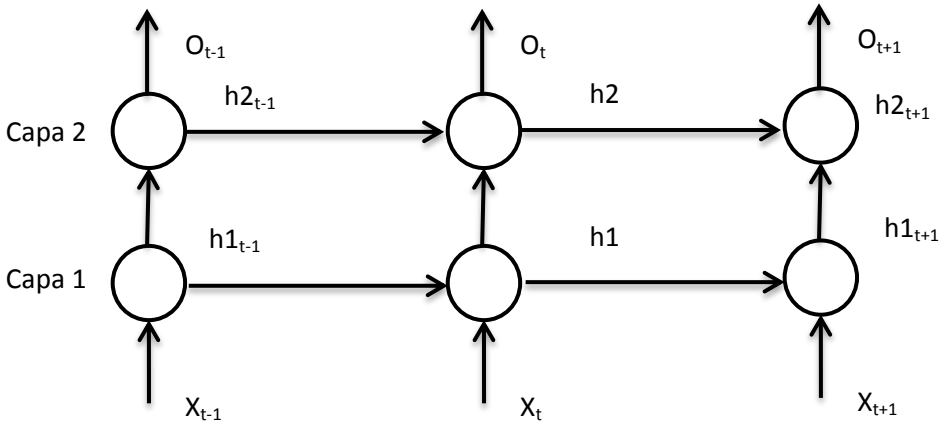

Figura 5.3 Despliegue en el tiempo de la red neuronal recurrente la cual recibe como entrada la STFT de la señal de voz.

Se hacen pruebas con la arquitectura de la Fig 5.3, con un número definido de neuronas como se muestran en la siguiente tabla.

Neuronas de entrada	512	STFT de la señal de voz como entrada a la red
Neuronas en la 1ra capa	60	Neuronas recurrentes en capa intermedia
Neuronas en la 2da capa	10	Neuronas recurrentes en capa intermedia
Neuronas de salida	3	Neuronas de salida, clasificadoras.

A continuación se muestran las salidas en la última capa (neuronas clasificadoras) para la muestra de entrenamiento *gato* (Fig 5.2):

Neurona clasificadora 1=0.000034 Neurona clasificadora 2=0.34553
Neurona clasificadora 3=0.45563

No se observan resultados muy atractivos, a pesar de ingresar los datos desfasados, el promedio de resultados no varía. El entrenamiento converge sin embargo se requiere una mejor caracterización. Por tanto se proceden a hacer pruebas con arquitectura Fig 5.3, pero con el siguiente número de neuronal en sus capas ocultas:

Neuronas de entrada	512	STFT de la señal de voz como entrada a la red
Neuronas en la 1ra capa	90	Neuronas recurrentes en capa intermedia
Neuronas en la 2da capa	30	Neuronas recurrentes en capa intermedia
Neuronas de salida	3	Neuronas de salida, clasificadoras.

Las salidas en la última capa (neuronas clasificadoras) para la muestra de entrenamiento “gato” con las modificaciones a la arquitectura son las siguientes:

Neurona clasificadora 1=0.000001 Neurona clasificadora 2=0.7453
Neurona clasificadora 3=0.8275

Se observa que al aumentar las neuronas recurrentes, no solo se aumenta la memoria en la red sino que también aumenta la caracterización de la misma, de esta forma se obtienen mejores resultados.

5.3 Características en pruebas de RNN

De las pruebas en el software con redes recurrentes, de entre ellas las mostradas anteriormente, se obtienen las siguientes características, las cuales permitirán entender y modificar a la red para tratar de resolver el problema de reconocimiento de palabras claves.

- Aumento de unidades de recurrencia en capas. – Permite el aprendizaje de un mayor número de modelos, se podría traducir como unidades de memoria.

- Aumento de capas.- Permite una mejor abstracción de la información con cada capa de aumento e interpretación por parte de la red, pero no existe una métrica para indicar cuantas capas se requieran.

- Aumento de unidades de neuronas en capas. – Aumenta la descripción e interacción entre información aumenta el reconocimiento, sin embargo no existe un factor determinante que indique cuanto se deba aumentar dependiendo el problema.

5.4 Reconocimiento de palabras claves con RNN

Como se mencionó durante el desarrollo del reconocedor de palabras claves, la arquitectura de redes neuronales, la cual es de interés probar e integra, es aquella cuyo modelo está basado en máquinas de estados denominada Arquitectura C. La comparación inmediata se llevara a cabo con el modelo más común de red recurrente con una variación en su número de neuronas que dan como resultado la Arquitectura A y Arquitectura B.

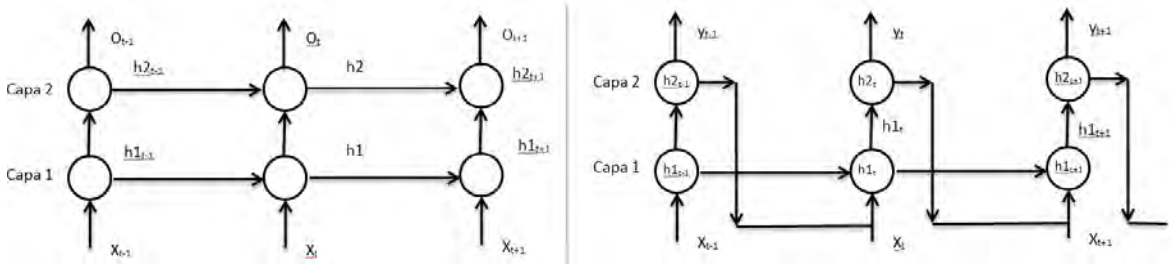

Fig 5.1 Arquitectura de las redes recurrentes utilizadas, a la izquierda el modelo convencional de redes recurrentes (A y B) y a la derecha la propuesta de red basada en máquina de estados (C).

Corpus de entrenamiento.

El corpus de entrenamiento usado consta de 10 palabras (del uno al diez) de las cuales se tienen 10 repeticiones, de estas diez repeticiones se hacen desfasamientos de 10 datos 50 veces con el fin de contemplar un mayor número de ventaneos posibles al momento de tratar la señal capturada y además mejorar el reconocimiento. Lo anterior nos da un total de 5000 muestras de entrenamiento.

El número de neuronas que contiene cada capa de la red se encuentra en la siguiente tabla.

	Capa de entrada	Capa 1	Capa 2	Capa de salida
Arquitectura A	256	400	400	10
Arquitectura B	256	600	300	10
Arquitectura C	406	500	200	10

Si bien como se observa en la Fig 5.2, la arquitectura propuesta (C) se encuentra con un desempeño similar al de una arquitectura común (B) pero existen otros datos que resultan interesantes como lo es el número de iteraciones que lleva entrenar la red neuronal, pues es menor, además al revisar la tabla se observa que el número de neuronas que conforman la arquitectura es menor.

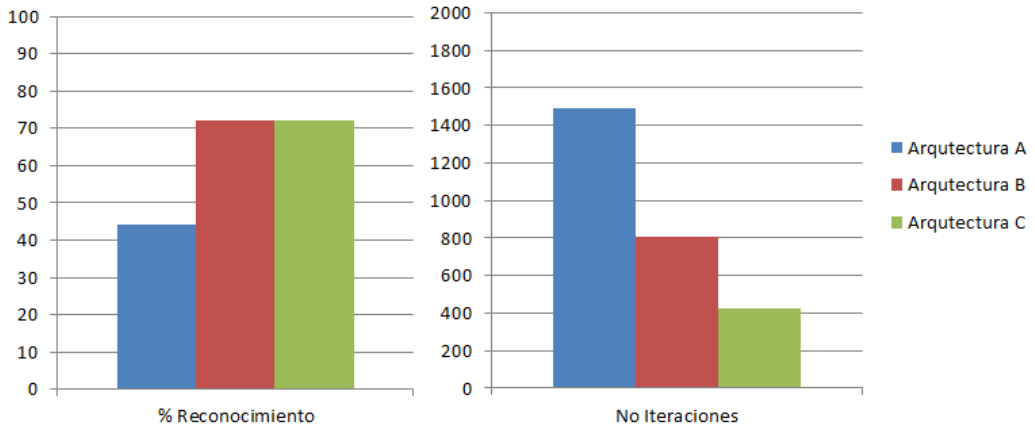

Fig 5.2 Comparación de desempeño y numero de iteraciones de entrenamiento para las arquitecturas A B y C.

Ahora bien, al revisar el estado del arte, arquitecturas como el LSTM cuentan con una eficiencia de reconocimiento por arriba del 80% y sistemas híbridos como el reconocedor de voz de Microsoft con una eficiencia del 95%.

Solo a modo de comparación se presentan a continuación matrices de confusión entre el sistema desarrollado basado en redes neuronales y el reconocedor de voz de Microsoft, con la finalidad ver el desempeño de la red con un sistema de la actualidad.

Matriz de confusión RNN

	Uno	Dos	Tres	Cuatro	Cinco	Seis	Siete	Ocho	Nueve	Diez
Uno	2	2	0	0	0	1	0	0	0	0
Dos	0	3	1	0	0	0	0	1	0	0
Tres	0	0	4	0	0	1	0	0	0	0
Cuatro	0	2	0	3	0	0	0	0	0	0
Cinco	0	0	1	0	3	0	0	1	0	0
Seis	0	0	0	0	0	4	1	0	0	0
Siete	0	0	1	0	0	1	3	0	0	0
Ocho	0	0	0	0	0	1	0	3	0	1
Nueve	0	0	0	0	0	0	1	0	4	0
Diez	0	0	0	0	0	0	0	1	0	4

Matriz de confusión Reconocedor de voz Microsoft (SDK)

	Uno	Dos	Tres	Cuatro	Cinco	Seis	Siete	Ocho	Nueve	Diez
Uno	5	0	0	0	0	0	0	0	0	0
Dos	0	5	0	0	0	0	0	0	0	0
Tres	0	0	5	0	0	0	0	0	0	0
Cuatro	0	0	0	5	0	0	0	0	0	0
Cinco	0	0	0	0	5	0	0	0	0	0
Seis	0	0	0	0	0	5	0	0	0	0
Siete	0	0	0	0	0	0	4	0	1	0
Ocho	0	0	0	0	0	0	0	5	0	0
Nueve	0	0	0	0	0	0	0	0	5	0
Diez	0	0	0	0	0	0	0	0	0	5

Por lo tanto se puede determinar que la arquitectura aún no alcanza el estado del arte sin embargo como sucede en el ámbito de las redes neuronales, no significa que no tenga futuro. Se propone un aumento en el corpus de entrenamiento y en el poder de cómputo para acelerar las pruebas.

Se debe recordar que uno de los objetivos es que este módulo se pueda integrar al robot y tener como resultado en una actualización al sistema como el de la Fig. 5.3.

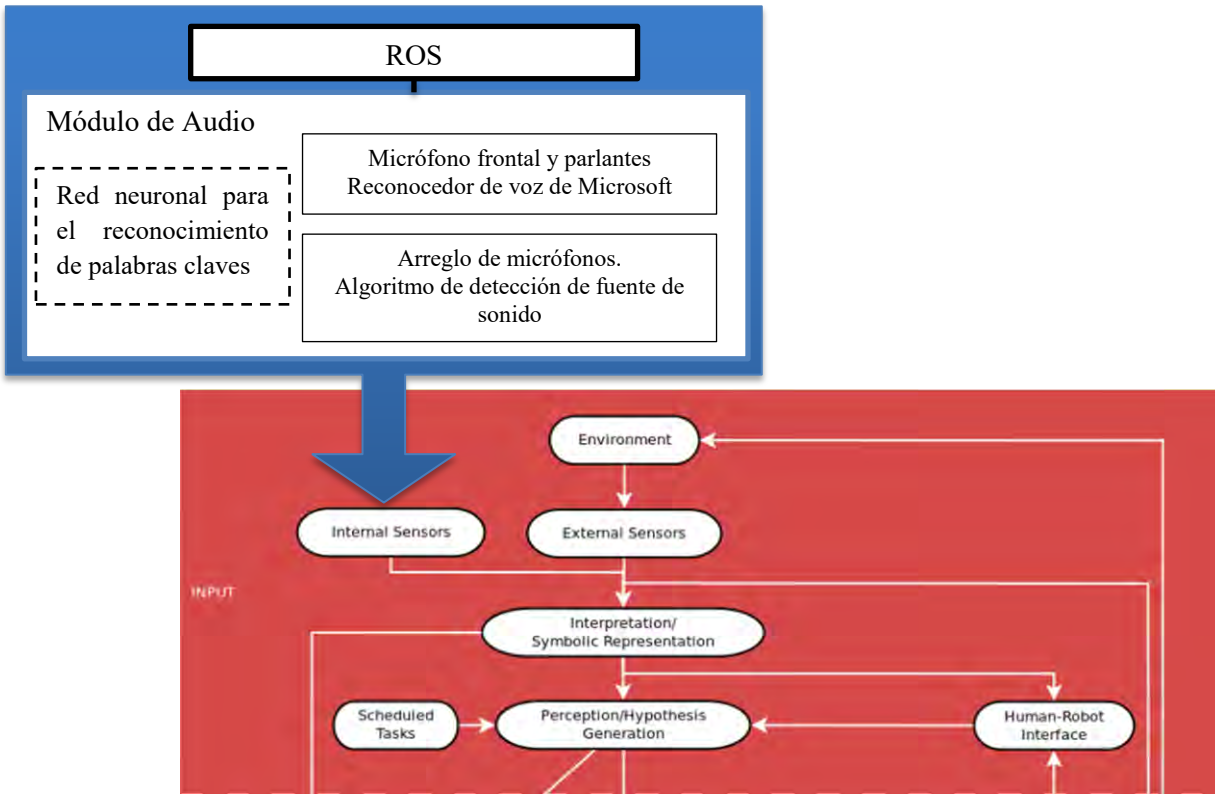

Fig. 5.3 Idealización del sistema actualizado con el módulo de reconocimiento de palabras claves.

5.5 Detección de fuente de sonido para robot de servicio.

Para demostrar la eficiencia de sistema, además de presentar los resultados de ensayos de laboratorio también se incluyen aquellos obtenidos del comité de competencia Robocup® 2016. El sistema presenta los siguientes resultados de eficiencia en la detección de la fuente de sonido bajo las siguientes condiciones del ambiente.

Condiciones del ambiente	**Porcentaje de eficiencia**
Espacio sin paredes (mayor 5 m redonda) con una sola fuente de sonido	85 %
Espacio sin paredes (mayor 5 m redonda) con una fuente de sonido y ruido	80 %
Espacio con paredes, una fuente de sonido y ruidos	55%
Robocup 2016 (espacio con paredes y ruido ambiental muy alto)	50%

El ruido es considerado como cualquier fuente de sonido ajeno a la dirección de interés (ruido ambiental, otro parlante, etc.), el hecho de existir las paredes agrega ecos o reverberaciones como interferencias. La fuente de sonido estaba ubicada en un rango de 1 a 3 metros. El módulo se implementa en el robot Justina corriendo bajo un módulo de ROS el cual publica en un tópico la

dirección detectada, de esta forma el planeador del robot podrá hacer uso de los motores para girar la base del robot y entonces usar el reconocedor de voz de Microsoft para interactuar con el usuario Figura 5.4.

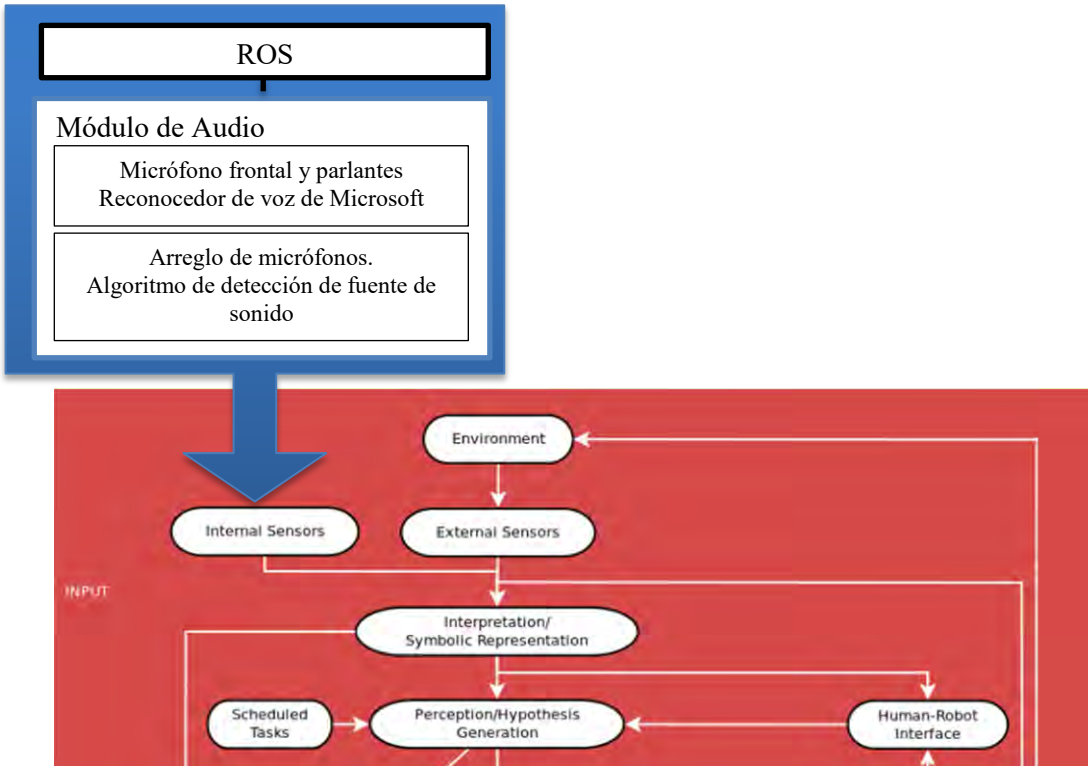

Fig. 5.4 Actualización del robot con el módulo de detección de fuente de sonido (sistema actualmente en 2017).

El robot implementa el reconocimiento de voz de Microsoft el cual tiene un reconocimiento del 90%, sin embargo esto sucede cuando se le habla de frente, pues el robot solo cuenta con un micrófono, este escenario se observa en le Fig. 5.5.

Figura 5.5 En el caso de la izquierda la eficiencia del reconocimiento es 90% pues el robot mira directamente al usuario mientras que en la derecha la eficiencia oscila entre 50% a 70% dependiendo que tan alto hable el usuario.

Debido a que el reconocedor de voz (SDK de Microsoft) se instala en el sistema operativo Windows y configura el micrófono predeterminado para la captura de audio, como consecuencia de esta instalación, aún no se pueden alimentar la señal de varios micrófonos al reconocedor de voz, por el momento. Entonces se implementa el sistema de detección de la dirección de la fuente de sonido para elevar la eficiencia del sistema de reconocimiento.

Se utiliza ROS como intermediario para el intercambio de información entre módulos del robot. El módulo de detección de dirección de sonido detecta una fuente de sonido y publica el ángulo donde se encuentra dicha fuente con respecto al sistema de referencia del robot. Acto seguido el módulo de planeación que rige las acciones a realizar del robot gira en la dirección detectada e inicia el reconocimiento de voz como se observa en la Fig 5.6.

La eficiencia del reconocimiento ya no se ve entorpecido por la dirección de la fuente de sonido y de nuevo se cuenta con eficiencia del 90% para el reconocer de voz de Microsoft.

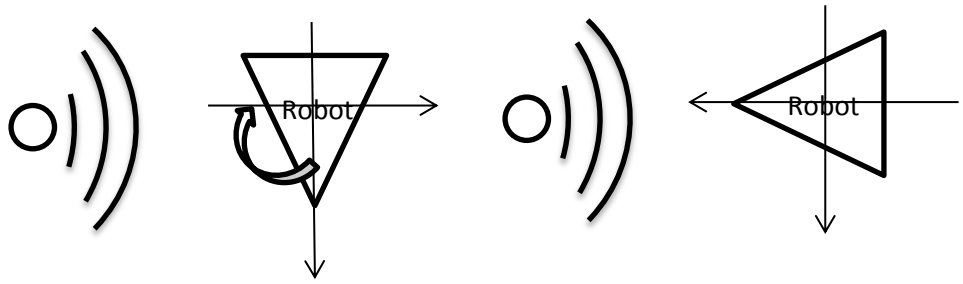

Figura 5.6 El sistema ahora detecta la fuente de sonido y hace que el robot gire para tener una eficiencia mayor al 90% y de esta manera interactuar con el usuario.

De esta forma el modulo será capaz de detectar la dirección de la fuente de sonido, se sacrifica precisión y resolución en pro de los recursos consumidos (circuito embebido de bajo consumo energético).

Capítulo 6

Conclusiones y Trabajo a futuro.

Ahora que el robot implementa el reconocimiento de voz de Microsoft y la detección de la dirección de arribo de la fuente de sonido el sistema trabaja mucho mejor a cuando solo se tenía el reconocedor de voz de Microsoft, además de que el robot presenta desde el punto de vista humano mayor inteligencia pues tiene una mayor interacción con el usuario.

El sistema es robusto al ruido sin embargo cuando se encuentra entre espacios cerrados pequeños es cuando su eficiencia decae demasiado. Pero esto no se debe a que el sistema no trabaje adecuadamente sino a una mala suposición de las condiciones de operación pues se evaluó como principal interferencia del sistema el ruido ajeno a la fuente de sonido de interés. En cierto modo existe este tipo de ruido y la técnica Formación de emisión (Beamforming) es robusta a este tipo de interferencia, sin embargo su deficiencia se presenta cuando existe reverberación, estas interferencias son generadas incluso por la fuente de interés cuando el sonido rebota en paredes por lo tanto se sugiere pasar del método Beamforming y optar por un algoritmo GSC (Generalized Sidelobe Canceller) pues a pesar de que sea un método con valores de entrada experimentales para la disminución de interferencias, es este mismo bloqueo de interferencias el que incluyendo reverberación, lo que vuelve a este método idóneo para la tarea.

El aumento de micrófonos no se sugiere pues el tiempo de procesamiento aumentaría enormemente, tampoco el aumento en un arreglo circular de micrófonos para discretizar el espacio de detección de sonido pues sería una solución poco precisa y muy descuidada en el uso de recursos (un sensor para cada grado de dirección).

El módulo de reconocimiento de palabras claves queda en una primera etapa de desarrollo con una arquitectura propia inspirada en el modelo de máquinas de estado el cual hasta el momento es prometedora, se ha demostrado que el sistema si bien no es mejor puede ser competente además de quedar las puertas abiertas para futuras etapas de desarrollo con esta arquitectura. El sistema de detección de palabras claves aún no está implementado en el robot, pero cuando se logre terminar su desarrollo será un sistema que podrá ejecutarse todo el tiempo sin el mayor uso de recursos por su simplicidad de operaciones y porque no está limitada por un sistema operativo, para el reconocimiento de palabras claves.

Finalmente el objetivo general se ha cumplido al hacer un aporte tecnológico al robot Justina, no en su totalidad, pues un módulo aún requiere otra etapa de desarrollo.

Apéndice

Código RNN

Código hecho en Python 2.7 y el framework theano para el módulo de entrenamiento de Redes Recurrentes

```
import numpy as np
import theano
import theano.tensor as T

dtype=theano.config.floatX

import SamplesTraining
n_r   =  150
n_in   =  8*64/2
n_hid  =  500
n_hid1 =  200
n_out  =  10

#declaracion de matriz
v = T.matrix(dtype=dtype)

def rescale_weights(values, factor=1.):
   factor = np.cast[dtype](factor)
   _,svs,_ = np.linalg.svd(values)
   #svs[0] is the largest singular value
   values = values / svs[0]
   return values

def sample_weights(sizeX, sizeY):
   values = np.ndarray([sizeX, sizeY], dtype=dtype)
   for dx in xrange(sizeX):
      vals = np.random.uniform(low=-1., high=1.,  size=(sizeY,))
      #vals_norm = np.sqrt((vals**2).sum())
      #vals = vals / vals_norm
      values[dx,:] = vals
   _,svs,_ = np.linalg.svd(values)
   #svs[0] is the largest singular value
   values = values / svs[0]
   return values

# parameters of the rnn as shared variables
def get_parameter(n_in, n_out, n_hid, n_hid1, n_r):
   # retroalimetacion
   x_r = theano.shared(np.zeros(n_r, dtype=dtype))
   x_in = theano.shared(np.zeros(n_in + n_r, dtype=dtype))
   W_ho_r = theano.shared(sample_weights(n_hid1, n_r))
   b_o_r = theano.shared(np.zeros(n_r, dtype=dtype))
   # capa entrada
   b_h = theano.shared(np.zeros(n_hid, dtype=dtype))
   h0 = theano.shared(np.zeros(n_r, dtype=dtype))
   W_ih = theano.shared(sample_weights(n_in + n_r, n_hid))
   W_hh = theano.shared(sample_weights(n_hid, n_hid))
   # capa oculta 1
```

```
    b_h1 = theano.shared(np.zeros(n_hid1, dtype=dtype))
    h1 = theano.shared(np.zeros(n_hid1, dtype=dtype))
    W_ih1 = theano.shared(sample_weights(n_hid, n_hid1))
    W_hh1 = theano.shared(sample_weights(n_hid1, n_hid1))
    # capa salida
    W_ho = theano.shared(sample_weights(n_hid1, n_out))
    b_o = theano.shared(np.zeros(n_out, dtype=dtype))
    print "initial moder:"
    print W_ih.get_value(), b_h.get_value()
    return W_ih, b_h, W_ho, b_o, h0, b_h1, h1, W_ih1, W_hh1, x_r, x_in, W_ho_r, b_o_r

W_ih, b_h, W_ho, b_o, h0, b_h1, h1, W_ih1, W_hh1, x_r, x_in, W_ho_r, b_o_r = get_parameter(n_in, n_out, n_hid,
n_hid1, n_r)
params = [W_ih, b_h, W_ho, b_o, h0, b_h1, h1, W_ih1, W_hh1]


# we could use the fact that maximaly two outputs are on,
# but for simplicity we assume independent outputs:
def logistic_function(x):
    return 1./(1 + T.exp(-x))

# sequences: x_t
# prior results: h_tm1
# non-sequences: W_ih, W_hh, W_ho, b_h
def one_step(x_t, x_r, h_t1m1, W_ih, b_h, W_ho, b_o, W_ih1, W_hh1, b_h1, x_in, W_ho_r, b_o_r):

    x_in = T.concatenate([x_t, x_r], axis=0)
    h_t = T.tanh(theano.dot(x_in, W_ih) + b_h)
    h_t1 = T.tanh(theano.dot(h_t, W_ih1) + theano.dot(h_t1m1, W_hh1) + b_h1)
    y_t = theano.dot(h_t1, W_ho) + b_o
    y_t = logistic_function(y_t)
    y_r = T.tanh(theano.dot(h_t1, W_ho_r) + b_o_r)
    y_r = logistic_function(y_r)

    return [y_r, h_t1, y_t]

# hidden and outputs of the entire sequence
[h_vals, _, o_vals], _ = theano.scan(fn=one_step,
                        sequences = dict(input=v, taps=[0]),
                        outputs_info = [h0, h1, None], # corresponds to return type of fn
                        non_sequences = [W_ih, b_h, W_ho, b_o, W_ih1, W_hh1, b_h1, x_in, W_ho_r, b_o_r] )


# target values
target = T.matrix(dtype=dtype)



# learning rate
lr = np.cast[dtype](0.2)
learning_rate = theano.shared(lr)


cost = -T.mean(target * T.log(o_vals) + (1.- target) * T.log(1. - o_vals))



def get_train_functions(cost, v, target):
    gparams = []
```

```
    for param in params:
        gparam = T.grad(cost, param)
        gparams.append(gparam)

    updates=[]
    for param, gparam in zip(params, gparams):
        updates.append((param, param - gparam * learning_rate))
    learn_rnn_fn = theano.function(inputs = [v, target],
                                   outputs = cost,
                                   updates = updates)
    return learn_rnn_fn



learn_rnn_fn = get_train_functions(cost, v, target)




import SamplesTraining
train_data = SamplesTraining.get_n_samples()


def train_rnn(train_data, nb_epochs=600):
  train_errors = np.ndarray(nb_epochs)
  for x in range(nb_epochs):
    print "Epoca "+str(x)
    error = 0.
    for j in range(len(train_data)):
        index = np.random.randint(0, len(train_data))
        i, o = train_data[index]
        train_cost = learn_rnn_fn(i, o)
        error += train_cost
    train_errors[x] = error
    print "Error "+str(error)
    if(error<0.005):
        break
  return train_errors

nb_epochs=100
train_errors = train_rnn(train_data, nb_epochs)

#matplotlib inline
import matplotlib.pyplot as plt
def plot_learning_curve(train_errors):
    plt.plot(np.arange(nb_epochs), train_errors, 'b-')
    plt.xlabel('epochs')
    plt.ylabel('error')
    plt.show()

plot_learning_curve(train_errors)



predictions = theano.function(inputs = [v], outputs = o_vals)


inp, outp = SamplesTraining.get_one_samples("uno8.wav",1,8,clase=[0.,0.,0.,0.,0.,0.,0.,0.,1.])
inp = np.array(inp,dtype=dtype)
```

```
pre = predictions(inp)
for p, o in zip(pre, outp):
   print p # prediction
   print o # target
   print
```

Código MVDR

Código hecho en Python 2.7 que ejecuta el algoritmo para detectar la dirección de arribo de la fuente de sonido

```
import numpy as np
import matplotlib.pyplot as plt

def beamForming(nphones, \
   sound_speed, \
   spacing, \
   look_dirs, \
   samples, \
   phone_data):
   bf_data = np.zeros((samples, len(look_dirs)))
   time_delays = np.matrix( (spacing/sound_speed))
   fft_freqs = np.matrix(np.linspace( 0, sampling_rate, samples, endpoint=False)).transpose()
   print fft_freqs.shape
   print time_delays.shape
   for ind, direction in enumerate(look_dirs):
      spacial_filt = 1.0/nphones*np.exp(-2j*np.pi*fft_freqs*time_delays*np.cos(direction))


      for k in range(0,samples/2):
         R= (np.fft.fft(phone_data,samples,0)[k]) * (np.fft.fft(phone_data,samples,0)[k][np.newaxis].T)
         for m in range (0,nphones):
            R[m][m]=1.001*R[m][m]
         R_inv=(np.linalg.inv(R))/900
         w_a=spacial_filt[k]
         spacial_filt[k]=(R_inv*w_a[np.newaxis].T )* (1/(w_a*R_inv*w_a[np.newaxis].T))[np.newaxis]


      bf_data[:,ind] = np.sum(np.fft.irfft( np.fft.fft(phone_data,samples,0)*np.array(spacial_filt), \
      samples, 0), 1)

   return bf_data
```

Bibliografía

[1] Montero Mata, Jordi (2009). Design and implementation of a system of voice control. Master Thesis, National Autonomous University of Mexico UNAM.

[2] Schaffer, J. D. (1992). Combinations of Genetic Algorithms and Neural Networks: A survey of the state of the art. COGANN-92, IEEE, 1-37.

[3] McDonnall, J. R. (1994). Neural network construction using evolutionary search. Conference On Evolutionary Programming, 9-16.

[4] MacLeod, C (1999). The Synthesis of Artificial Neural Networks using Single String Evolutionary Techniques, PhD Thesis, The Robert Gordon University, Aberdeen UK.

[5] Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), Advances in Neural Information Processing Systems (Vol. 7,). Cambridge MA, USA: MIT Press

[6] Lawrence R. Rabiner y Ronald W. Schafer Digital Precessing of Speech Signals Prentice Hall, 1978.

[7] Jhon R. Deller, Jhon H.L. Hansen, y Jhon G. PRoakis. Discrete-Time Processing of Speech Signals. IEEE Press Editorial, 1993.

[8] John Makhoul. Linear prediction: A tutorial review, 1976

[9] Lawrence R. Rabiner y Biing-Hwang Juang, Fundamentals of Speech Recognition. Prentice Hall International Inc., 1993.

[10] Stuart P. Lloyd. Least squares quantization in pem´s. IEEE Trans. On Comunication, 1957.

[11] Lawrence R. Rabiner A tutorial on HIdden Markov Models and slected applications in speech recognition, 1989.

[12] Jesus Savage Carmona. A Hybrid System with Symbolic AI and Statistical Methods for Speech Recognition. Tesis de Doctorado, Universidad de Washintongton, 1995.

[13] H.A. Bourlard and N. Morgan, Connnectionist Speech Recognition: A Hybrid Approach, Kluwer Academic Publishers, 1994.

[14] Qifeng Zhu, Barry Chen, Nelson Morgan, and Andreas Stolcke, Tandem connectionist feature extraction for conversational speech recognition, Conference on Machine Learning for Multimodal Interaction, Berlin, Heidelberg, 2005, MLMI'04, pp. 223– 231, Springer-Verlag.

[15] Alex Graves, Abdel-rahman Mohamed and Geoffrey Hinton, Speech Recognition with Deep Recurrent Neuronal Networks, University of Toronoto, 2014.

[16] P. Maya , E. Zhoub,∗ , C.W. Lee b, Learning in fully recurrent neural networks by approaching tangent planes to constraint surfaces, 2012

[17] Mike Schuster and Kuldip K. Paliwa, Bidirectional Recurrent Neural Networks, 1997, IEEE

[18] A. Mohamed, G.E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," Audio, Speech, and Language Processing, IEEE Transactions on, vol. 20, no. 1, pp. 14 –22, jan. 2012.

[19] G. Hinton, Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," Signal Processing Magazine, IEEE, vol. 29, no. 6, pp. 82 –97, nov. 2012.

[20] Savage Carmona Jesús, Chávez Rodríguez Norma Elva, Gabriel Vázquez, Diseño de microprocesadores, Maquinas de estado y su construcción, pp. 22-35 Facultad de Ingeniería UNAM 2015.

[21] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," IEEE ASSP Magazine, pp. 4-24, April. 1988

[22] C. Farsakh and J. A. Nossek, "Spatial covariance based downlink beamforming in an SDMA mobile radio system," IEEE Trans. Commun., vol. 46, pp. 1497-1506, Nov. 1998.

[23] P. S. Naidu, Sensor Array Signal Processing, 2nd Ed., CRC, 2009

[24] Benesty J., Chen J. and Huang Y., Microphone Array Signal Processing, springer 2008.

[25] Xuedong Huang, Acero A, y Hon H. Spoken Language Processing-A Guide to theory, Algorithms, and System Development, Prentice Hall, 2001.

[26] Leonard E. Baum y J. A. Eagon An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for acology. Bulletin American Meteorological Society 1967.

[27] Leonard E. Baum and Ted Petrie. Statictical inference for porbabilistic funtions of markov chain. The Annals of Mathematical Statistics, 1966.

[28] Leonard E. Baum , Ted Petrie, George Soules and Norman Weiss. A maximization technique occurring in the statistical, analysis of probabilistic funtions of markov chains. The Annals of Mathematical Statistics, 1970.

[29] Li Deng, Fellow, Dong Yu, and Alex Acero, "Structured Speech Modeling", IEEE Transactions on audio, speech, and language processing, Vol. 14, No. 5, september 2006

[30] J. L. Flanagan, Speech analysis, synthesis, and perception, Berlin: Springer-Verlag, 2nd edition, 1972.

[31] R. McGowan and S. Cushing, "Vocal tract normalization for midsagittal articulatory recovery with analysis-by-synthesis," J. Acoust. Soc. Am., vol. 106, Issue 2, pp.1090–1105, August 1999.

[32] Dr. H. B. Kekre, Ms. Vaishali Kulkarni, "Speaker Identification by using Vector Quantization", International Journal of Engineering Science and Technology, Vol. 2(5), 2010, 1325-1331.

[33] Graves Alex, Supervised Sequence Labelling with Recurrent Reunal Networks, Springer 2012

[34] Li Deng and Dong Yu, Deep Learning Methods and Applications, NOW the essence of knowledge, Volume 7, 2006

[35] Dong Yu and Li Deng, Automatic Specch Recognition, Springer, 2015.

[36] Mohamed, A., Dahl, G., and Hinton, G. "Deep belief networks for phone recognition," in Proc. NIPS Workshop Deep Learning for Speech Recognition and Related Applications, 2009.

[37] Bengio, Y. Artificial Neural Networks and Their Application to Speech and Sequence Recognition, Ph.D. Thesis, McGill University, Montreal, Canada, 1991

[38] R Schmidt, Multiple emitter location and signal parameter estimation. IEEE Trans. Antennas Propagation. 34(3), 276–280 (1986).

[39] Caleb Rascón y Iván Meza, Localization of Sound Sources in Robotics: A Review, Instituto de Investigaciones en Matematicas Aplicadas y en Sistemas, Universidad Nacional Autonoma de Mexico, Mexico.

[40] Y. Matsusaka, T. Tojo, S. Kubota, K. Furukawa, D. Tamiya, K. Hayata, Y. Nakano, T. Kobayashi, Multi-person conversation via multi-modal interface - a robot who communicate with multi-user, in: Proceedings of European Conference on Speech Communication and Technology (EUROSPEECH), Vol. 99, 1999, pp. 1723–1726.

[41] S. Hashimoto, S. Narita, H. Kasahara, A. Takanishi, S. Sugano, K. Shirai, T. Kobayashi, H. Takanobu, T. Kurata, K. Fujiwara, et al., Humanoid robot - development of an information assistant robot Hadaly, in: Proceedings of 6th IEEE International Workshop on Robot and Human Communication (RO-MAN), IEEE, 1997, pp. 106–111.

[42] P. Pertil¨a. Acoustic Source Localization in a Room Environment and at Moderate Distances - [PhD Thesis]. Tampere University of Technology, 2009

[43] D. R. Griffin. Listening in the dark: the acoustic orientation of bats and men. Yale University Press, 1958.