Universitat Autònoma
de Barcelona

SISTEMA DE RECONEIXEMENT DE VEU PER A DOMÒTICA

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per
Marta Bocos Hernández
i dirigit per
Javier Serrano Garcia

Escola Universitària d'Informàtica
Sabadell, setembre de 2009

El/la sotasignant, Javier Serrano Garcia,
professor/a de l'Escola Universitària d'Informàtica de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció
per en Marta Bocos Hernández.

I per a que consti firma la present.
Sabadell, setembre de 2009.

.....
Signat: Javier Serrano Garcia.

RESUMEN

En este proyecto se discute y se prueba la viabilidad de un sistema reconocedor de voz, así como la integración de éste en una aplicación gráfica que simula una vivienda domótica. Se hace uso de tecnologías de reconocimiento del habla para llevar a cabo la primera parte del proyecto, aplicándolo al reconocimiento de un conjunto de frases relacionadas con las tareas que se realizan habitualmente en hogar. En cambio, para la creación de la interfaz gráfica se utilizan tecnologías gráficas tridimensionales para dar más realismo al simulador. A lo largo del presente proyecto se estudia el funcionamiento de una herramienta de código abierto de reconocimiento del habla llamada HTK, la cual permite generar los modelos acústicos que empleará el reconocedor. También se presentan otras herramientas utilizadas durante la creación de la aplicación creada, tales como Julius que permite la integración del reconocedor de voz obtenido con HTK en una aplicación.

Índice

Capítulo 1: Introducción 5

1. Presentación. 5

2. Motivación. 5

3. Objetivos. 6

4. Estudio de viabilidad 7

4.1 Introducción. 7

4.2 Estado del arte. 7

4.3 Objetivos. 7

4.4 Sistema a realizar. 8

4.5 Planificación. 10

5. Estructura de la memoria. 10

Capítulo 2: Reconocimineto automático del habla 12

1. Reconocimiento del habla. 12

2. Planteamiento del problema. 14

3. Elementos básicos de un reconocedor. 16

3.1 Modelos acústicos. 16

3.2 Modelos ocultos de Markov. 17

3.3 Modelo de lenguaje o gramática. 25

3.4 Diccionario de pronunciación. 25

4. Herramienta empleada. 25

Capítulo 3: La domótica 26

1. Introducción. 26

2. Beneficios de la domótica. 27

3. Aplicaciones. 28

4. La interfaz de voz..... 30

5. Especificaciones de un sistema real. 30

Capítulo 4: Análisis de especificaciones 32

1. Especificaciones del proyecto. 32

1.1 Requerimientos funcionales para el reconocedor de voz. 32

1.2 Especificaciones para el simulador de domótica……..33

1.3 Requerimientos no funcionales. 34

Capítulo 5: Diseño 36

1. Estructura de la aplicación completa 37

1.1 Ficheros adicionales. 39

1.2 Librerías. 40

2. Creación, entrenamiento y prueba de los modelos acústicos. 42

2.1 Preparación de los datos. 42

2.2 Creación de los modelos de monophones. 44

2.3 Creación de los modelos de triphones con estados ligados. 48

2.4 Adaptación de los modelos. 50

2.5 Evaluación del reconocedor. 52

3. Desarrollo de la aplicación. 53

3.1 Adaptación de los modelos acústicos para su uso con Julius. 53

3.2 Creación de la aplicación de reconocimiento de voz con Julius. 54

3.3 Creación de la interfaz gráfica. 56

4. Conexiones entre hilos. 62

5. Control de errores. 62

6. Juego de pruebas. 64

Capítulo 6: Conclusiones 66

1. Conclusiones. 66

2. Posibles mejoras 67

Bibliografía 68

Índice de figuras

Figura 2.1: Esquema general de un sistema de reconocimiento de voz. 15

Figura 2.2: Extracción de características. 16

Figura 2.3: Mel- frequency cepstral coefficients. 17

Figura 2.4: Ejemplo de HMM..... 19

Figura 5.1: Diagrama de la aplicación completa. 37

Figura 5.2: Diagrama de la aplicación reconocedor de voz..... 37

Figura 5.3: Diagrama de la aplicación gráfica, simulador de domótica. 38

Figura 5.4: Mapa de la vivienda visualizada en el simulador..... 57

Capítulo 1

INTRODUCCIÓN

1. Presentación.

Este proyecto esta orientado a desarrollar un reconocedor de voz aplicado a la domótica con fin de que sea capaz de interpretar algunas órdenes sencillas y mostrarlas en forma de animación gráfica en un simulador de domótica. Para ello se hará uso de tecnologías de reconocimiento del habla de código abierto con el fin de automatizar algunas de las tareas más habituales que se llevan a cabo en una vivienda. Además, se presentan las posibilidades de estas tecnologías en el ámbito de las viviendas inteligentes y se discute la viabilidad de un sistema de estas características.

2. Motivación.

Hoy más que nunca está a la orden del día tanto los sistemas domóticos como los de reconocimiento del habla e incluso la combinación de ambos que hace más fácil y cómodo el uso de los controles de la vivienda. De ahí surge la idea para llevar a cabo este proyecto que propone una aproximación a un sistema reconocedor de voz orientado a la domótica que además incorpora un simulador de domótica que recibe y ejecutas las órdenes.

3. Objetivos.

Los objetivos de este proyecto son realizar un sistema de reconocimiento de voz para domótica, de forma que éste sea capaz de distinguir diferentes frases relacionadas con las tareas del hogar y también otras cotidianas que se llevan acabo en una vivienda.

Se dispone de un software que permite crear los modelos acústicos que una vez entrenados son capaces de identificar los rasgos fonéticos y de entonación de cada persona. Además, actualmente se ha empezado a desarrollar un sistema de reconocimiento de voz que interpreta algunas frases mediante la identificación de palabras completas. Con la realización de este proyecto también se pretende mejorar i ampliar el sistema de reconocimiento de voz ya existente de forma que éste sea capaz de realizar más acciones con un margen de error mucho menor que el actual mediante identificación de fonemas y conjunto de algunos de ellos.

Debido a la complejidad y cantidad de recursos necesarios para implementar un sistema de reconocimiento de voz en una vivienda, éste no se llevará acabo en un sistema real. Sin embargo, para demostrar su funcionamiento y eficacia, se desarrollará un simulador virtual. Dicho simulador tiene la interfaz gráfica de la planta de una casa en la cual se visualizan en forma de animación todas las ordenes dadas por el usuario, que previamente han sido analizadas y reconocidas como frases válidas por el sistema de reconocimiento de voz. En caso de pronunciar una frase que no consta en la gramática del sistema de reconocimiento de voz, también notificará de forma visual al usuario de lo ocurrido. Finalmente, se desarrollará la comunicación entre los dos procesos para conseguir una mayor independencia del reconocedor de voz, y así poder reaprovecharlo para futuros proyectos.

4. Estudio de viabilidad.

4.1 Introducción.

El objetivo de este proyecto es diseñar y desarrollar una aplicación de reconocimiento de voz para domótica. Con la implementación de este software se pretende automatizar algunas de las tareas cotidianas más habituales en una casa.

4.2 Estado del arte.

Hoy en día ya existen sistemas de reconocimiento del habla e incluso viviendas domóticas con la posibilidad de incorporar estos sistemas [PRO2009 y ALH2009]. El principal inconveniente es el elevado coste de implementación de un sistema completo de este tipo e incluso en ciertos casos el reconocedor de voz es limitado en cuanto a la eficacia y robustez ante la gran variedad de usuarios y acciones a realizar.

4.3 Objetivos.

Se dispone de un software de creación de modelos acústicos que una vez entrenados son capaces de identificar los rasgos fonéticos y de entonación de cada persona. Además, actualmente se ha empezado a desarrollar un sistema de reconocimiento de voz que interprete algunas frases mediante la identificación de palabras completas.

Los objetivos de este proyecto son mejorar i ampliar el sistema de reconocimiento de voz ya existente, de forma que éste sea capaz de realizar más acciones con un margen de error mucho menor que el actual. También debe desarrollarse una aplicación gráfica que reciba las órdenes ya identificadas y muestre una animación de la orden mencionada por el usuario, ya que se trata de simulador de domótica, y no se va a implementar en un sistema real.

El abanico de usuarios finales de la aplicación es muy amplio, puede ser utilizado por cualquier persona siempre y cuando sepa hablar español, independientemente de la edad, condición social o conocimientos específicos. Sin embargo, es limitado, ya que antes de poder utilizarlo, el usuario debe entrenar al programa de reconocimiento de voz para que éste pueda funcionar correctamente. Si no

es así, el reconocedor puede fallar con mayor frecuencia al identificar correctamente las acciones.

Para obtener toda la información necesaria para el desarrollo del nuevo sistema, las fuentes a las que se recurrirán son:

- El programa de reconocimiento de voz HTK.
- Gramáticas y Lenguajes Independientes del Contexto.
- Software y API de desarrollo Julius.
- Librerías de desarrollo de aplicaciones.
- Manuales y librerías de animaciones y gráficos tridimensionales por ordenador.

4.4 Sistema a realizar.

Aplicación informática de reconocimiento de voz para domótica. El proyecto consta de dos partes bien diferenciadas: el sistema de reconocimiento de voz y la interfaz gráfica. La aplicación a desarrollar debe ser capaz de reconocer un amplio repertorio de frases en español relacionadas con las tareas que se llevan acabo en una vivienda y ejecutarlas en forma de animación en la pantalla.

Con el objetivo de mejorar el sistema de reconocimiento de voz hay que separar en fonemas los modelos acústicos con los cuales va a trabajar y así reducir los posibles errores a la hora de identificar las palabras. Por otro lado, para incrementar el repertorio de frases que el sistema de voz puede identificar, sólo se tendrá que ampliar la gramática que define el lenguaje independiente del contexto, la cual incluye todas las frases a ser interpretadas, siempre teniendo en cuenta que si aparecen nuevos fonemas también habrá que generar y entrenar su correspondiente modelo acústico. Sin embargo, la interfaz gráfica y las animaciones deben desarrollarse por completo, además de la unión de comunicación entre ambas partes del proyecto.

El sistema a realizar seguirá un modelo de desarrollo evolutivo, puesto que se pueden desarrollar las dos partes del proyecto simultáneamente por separado y luego crear el nexo de unión.

Riesgos asociados al proyecto:

- Problemas con el sistema de reconocimiento de voz, derivados del ruido ambiental o introducido por el micrófono.
- Problemas con el sistema de reconocimiento de voz, debidos a los diferentes dialectos del idioma o distintas entonaciones de la persona según su estado de ánimo.
- No lograr un margen de error inferior al actual.

Recursos necesarios para el desarrollo de la aplicación:

- Un ordenador personal con sistema operativo Ubuntu o similar.
- Micrófono, periférico para el ordenador.
- Aplicación HTK.
- Software de programación gráfica.
- API de desarrollo Julius entre otras.

Análisis coste-beneficio:

- Estudio del estado actual. (30 horas)
- Grabación de las frases de muestra para creación y entrenamiento de los modelos acústicos. (4 horas)
- Desarrollo y mejora de los modelos acústicos. (20 horas)
- Creación de la aplicación de reconocimiento de voz. (20 horas)
- Creación de la interfaz gráfica del simular de domótica. (100 horas)
- Creación de las animaciones. (8 horas)
- Desarrollo del nexo entre ambas partes. (2 horas)
- Pruebas de la aplicación completa. (5 horas)
- Documentación y memoria (60 horas)

TOTAL: 249 horas x 40€

9960 €

Alternativas:

Si no es posible reducir el margen de error, una alternativa sería mantener el margen actual y dedicar el proyecto ampliar el número de ordenes que éste es capaz de interpretar y a la creación del simulador de domótica.

4.5 Planificación.

La creación de la memoria y toda la documentación correspondiente se va desarrollando simultáneamente al proyecto en si, el cual se desglosa en tareas a continuación.

a. Estudio del estado actual del sistema de reconocimiento de voz: Recogida de información sobre sistemas de reconocimiento de voz, lenguajes independientes del contexto y gramáticas. Estudio de la aplicación para reconocimiento de voz, HTK (30 horas).
b. Grabar las muestras para la creación de los modelos acústicos para el reconocedor de voz (4 horas).
c. Desarrollar y mejorar los modelos acústicos para el sistema de reconocedor de voz y ampliar el lenguaje que éste debe distinguir para que sea capaz de identificar las características fonéticas del usuario (20 horas).
d. Realizar test de eficacia y comprobar que realmente se ha conseguido el objetivo (20 horas).
e. Desarrollar la aplicación de reconocimiento de voz (50 horas).
f. Diseñar y desarrollar la interfaz gráfica e integrar las animaciones 3D. (58 horas)
g. Desarrollo del nexo de comunicación entre ambas partes (2 horas).
h. Realizar las pruebas necesarias para comprobar el funcionamiento de todo el sistema (5 horas).
i. Implementación del software.

5. Estructura de la memoria.

El capítulo que prosigue a la introducción está dedicado a los fundamentos del reconocimiento del habla, donde se habla de los elementos básicos de un reconocedor de voz y concretamente los empleados en este proyecto, ya que se trata de un tema muy extenso y con varios métodos para abordarlo. Le sigue un tercer capítulo donde se comenta de forma resumida que es la domótica, que beneficios aporta y el uso de sistemas reconocedores de voz en el ámbito de los edificios inteligentes. El resto de la

memoria incluye un capítulo de análisis de especificaciones donde se exponen los requerimientos funcionales y no funcionales del proyecto y los de un sistema real. El capítulo de diseño cuenta el proceso seguido para llevar a cabo la creación de los modelos acústicos y del simulador de domótica. Terminamos con el capítulo 6, dedicado a las conclusiones obtenidas y a las posibles mejoras futuras que se podrían aplicar al proyecto.

Capítulo 2

RECONOCIMIENTO AUTOMÁTICO DEL HABLA

1. Reconocimiento del habla.

El reconocimiento automático del habla (*ASR, Automatic Speech Recogntion*) es el proceso por el cual un modelo de cómputo es capaz de traducir fielmente los sonidos asociados a una unidad de discurso y codificarlos en alguna forma de secuencia simbólica (habitualmente texto) que representa total o parcialmente la carga conceptual del mensaje hablado.

Cualquier sistema de reconocimiento automático del habla parte de la hipótesis de que la señal acústica contiene toda la información necesaria para reconocer el discurso hablado.

El reconocimiento automático del lenguaje hablado no es una tarea trivial, resulta extremadamente difícil de realizar a pesar del conocimiento que hay actualmente de él. El objetivo de las tecnologías ASR es abordar el problema mediante sistemas que relacionen una señal acústica con una cadena de palabras.

Hasta la fecha todos los sistemas imponen restricciones que permiten abordar con éxito el reconocimiento del habla en un contexto lingüístico restringido, acorde con la tecnología existente. Según [CAS87], a pesar de que los resultados con los sistemas actuales en condiciones de gramática restringida y medio de laboratorio son buenos, las prestaciones decaen fuertemente en condiciones reales. Hoy en día el trabajo se centra principalmente en sistemas capaces de tener en cuenta modelos del lenguaje, integrando entonación y lenguaje natural y capaces de funcionar con robustez frente a variaciones en los hablantes y condiciones de comunicación adversas.

El reconocimiento del habla consiste básicamente en un proceso de clasificación de patrones que podemos dividir en tres fases: adquisición de datos, preproceso de datos y clasificación por decisión.

En la fase de adquisición de datos, los datos analógicos del espacio físico se obtienen a través de algún dispositivo, en este caso un micrófono, y se convierten a un formato digital apropiado para su tratamiento. En esta etapa, las variables físicas se convierten a un conjunto de datos medibles. En el caso del reconocimiento del habla, estos datos medibles son las amplitudes de un sonido a lo largo del tiempo. Estos datos se utilizan como entrada de la fase de preproceso de datos. En esta segunda fase, tratamos los datos de entrada para obtener un conjunto discreto de características distintivas mediante un cambio de espacio (vectorial) de representación, con el que se trata de obtener aquella representación que mejor caracterice los datos de entrada. La tercera fase es el clasificador, formado por un conjunto de funciones de decisión que nos permite clasificar el objeto introducido a partir del conjunto de características distintivas. La clasificación consiste en asignar una de las clases preestablecidas al conjunto de características evaluando el modelo acústico correspondiente. Cada clase se representa del mejor modo posible mediante uno o varios modelos acústicos. En este caso cada una de las clases es un fonema.

Sin embargo, se tiene la dificultad añadida de la variabilidad de la entrada, debido a la gran variedad de hablantes (hombres, mujeres, etc.), entonación y timbre de cada hablante, las variantes de vocabulario y forma de expresión (por la influencia dialectal) y la posibilidad de que un mismo hablante utilice una entonación, vocabulario o forma de expresión diferente según la ocasión y el estado de ánimo. Es posible subsanar este problema disponiendo de un cuerpo de entrenamiento suficientemente grande y añadiendo una fase de reentrenamiento del sistema.

Concretamente, un sistema de reconocimiento del habla queda determinado por los siguientes parámetros:

- La elección de la unidad de reconocimiento. Delimitando la naturaleza de los fragmentos de discurso a reconocer. Las unidades de reconocimiento pueden ser:

microfonemas, pseudofonemas, fonemas, difonemas, trifonemas, sílabas, palabras o incluso frases completas.

- El tamaño del vocabulario. Cuanto mayor es el vocabulario, mayor deberá ser el tiempo invertido en la búsqueda del candidato idóneo.
- Restricción de la tarea. Un sistema que responda al lenguaje natural o a un conjunto restringido de frases y palabras.
- Los patrones de referencia son sensibles a las condiciones de adquisición donde se introduce ruido auditivo (ruido de fondo), y ruido generado como consecuencia de los medios de registro y transmisión.

Existen varias estrategias fundamentales para abordar el reconocimiento del habla. Actualmente, la mayoría de los reconocedores en funcionamiento se basan en técnicas estadísticas que requieren menos memoria física y tienen un mejor tiempo de respuesta. Sin embargo, necesitan de una fase de entrenamiento que es mucho más lenta y costosa pero que se realiza una sola vez, por lo que merece la pena realizarlo.

Para la realización de este proyecto se ha empleado el método probabilístico basado en la utilización de modelos ocultos de Markov (HMM, *Hidden Markov Model*) teniendo en cuenta que se busca un reconocedor dependiente del hablante, de habla continua, con un pequeño vocabulario y en un entorno ruidoso. Los modelos ocultos de Markov serán explicados detalladamente más adelante.

2. Planteamiento del problema.

Matemáticamente, el objetivo del reconocimiento del habla puede ser formulado de la siguiente manera [RAB93]: Dada una observación acústica $X = x_1, x_2, ..., x_n$, hay que encontrar la correspondiente secuencia de palabras $\hat{W} = w1, w2, ..., w_m$ que maximice la probabilidad *a posteriori* $P(X \mid W)$. Empleando el *teorema de Bayes* podemos expresarlo como:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W \mid X) = \underset{W}{\operatorname{argmax}} \frac{P(X \mid W)P(W)}{P(X)}$$

Puesto que $P(X)$ es la probabilidad *a priori* de la observación acústica X, éste es un valor fijo y no influye en el cálculo. Por lo tanto podemos rescribir la ecuación anterior de la siguiente manera:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(X \mid W)P(W)$$

La probabilidad $P(W)$ es la probabilidad *a priori* de observar W independientemente de la señal acústica observada y es conocido como *modelo de lenguaje*. $P(X \mid W)$ es la probabilidad de observar la señal acústica X dada una secuencia de palabras específica W, y es determinada por el modelo acústico.

En el proceso de reconocimiento del habla, se postula una secuencia W de palabras y se determinan sus probabilidades mediante el modelo de lenguaje. Cada palabra se convierte entonces en una secuencia de fonemas usando un diccionario de pronunciación, también conocido como *léxico* y, para cada fonema, hay un modelo estadístico llamado *modelo oculto de Markov* (HMM, *Hidden Markov Model*). La secuencia de HMMs necesaria para representar la pronunciación es concatenada formando un único modelo compuesto y se calcula la probabilidad $P(X \mid W)$ de generación de este modelo de observación X. Este proceso se repite para toda la secuencia de palabras y se selecciona la secuencia más probable. Ésta será la salida del reconocedor.

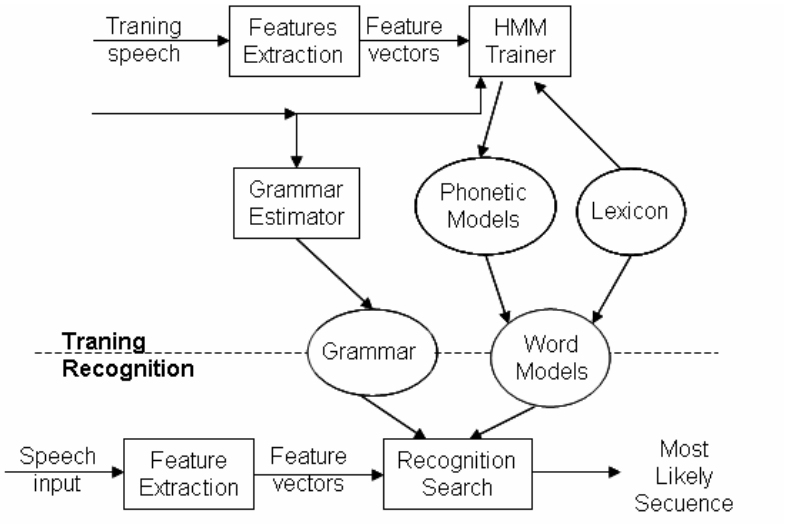

Figura 2.1: Esquema general de un sistema de reconocimiento de voz.

3. Elementos básicos de un reconocedor.

3.1 Modelos acústicos.

Los modelos acústicos se pueden ver como filtros que contienen la variabilidad acústica de una lengua. A partir de estos modelos y de una señal de entrada se obtiene la hipótesis de lo que el hablante ha dicho.

Los modelos acústicos toman las propiedades acústicas de la señal de entrada, obtienen un conjunto de vectores de características que después compararán con un conjunto de patrones que representan los símbolos de un alfabeto fonético y devuelve los símbolos que más se parecen. En esto se basa el proceso matemático probabilístico llamado Modelo Oculto de Markov (HMM). Este tipo de modelo es el que utiliza la herramienta de trabajo para este proyecto.

El análisis acústico es el proceso de extracción de un vector de características de la señal acústica de entrada con el fin de aplicar la teoría del reconocimiento de patrones. Un vector de características es básicamente una representación paramétrica de la señal acústica. Contiene la información más importante y almacenada de forma compacta. La mayoría de los sistemas de reconocimiento del habla preprocesan la señal para reducir el ruido y la correlación, y así extraer un buen conjunto de vectores de características. En la figura *2.2* se ilustra el proceso de extracción del vector de características:

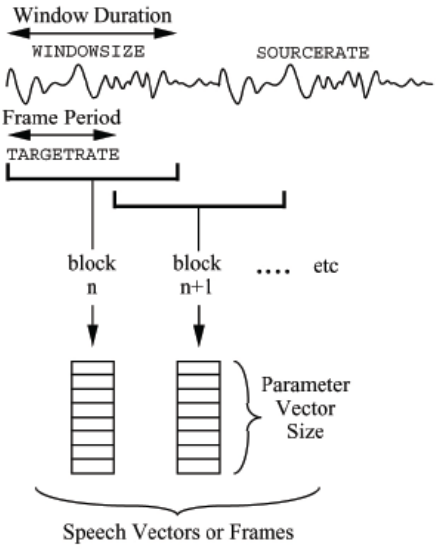

Figura 2.2: Extracción de características.

Existen diferentes técnicas de filtrado, dos de las más frecuentes son: LPC (*Linear Predictive Coding*) y MFCC (*Mel Frequency Ceptral Coefficients*). Para la realización de este proyecto se ha empleado la técnica MFCC.

Mediante esta técnica se calculan una serie de coeficientes llamados MFCC's siguiendo una serie de pasos que comprenden la división de la señal en *frames* de un determinado número de milisegundos, obtener el espectro de amplitud mediante la transformada de *Fourier*, convertir al llamado espectro de Mel, calcular los logaritmos del espectro y por último la aplicación de la transformada discreta del coseno. Es común obtener también los vectores de características de Mel derivados para contener el análisis diferencial de primer y segundo orden, además de la diferencia de coeficientes estáticos; tal y como se ilustra en la figura 2.3.

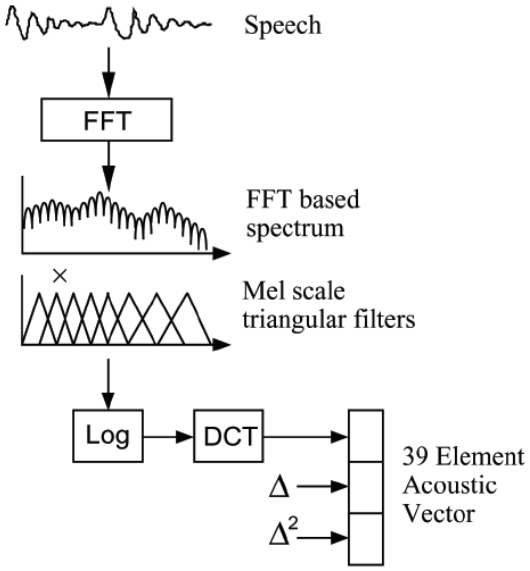

Figura 2.3: Mel- frequency cepstral coefficients

3.2 Modelos ocultos de Markov.

Un HMM es un proceso estocástico donde el cambio de estado sólo depende del estado actual. Consta de una máquina con un número finito de estados y un conjunto de funciones aleatorias, cada una de las cuales está asociada a un estado. Asociado a cada transición entre estados se produce un vector de observaciones o parámetros (correspondiente a un punto del espacio n-dimensional).

En un instante de tiempo se asume que el proceso está en un estado determinado y que genera una observación de un símbolo mediante la función de distribución de probabilidad que depende únicamente del estado actual. Al instante siguiente, se cambia de estado siguiendo una matriz de probabilidades de transición entre estados que dependen del estado anterior, produciendo una nueva observación mediante la función aleatoria correspondiente. Puede ser que el proceso permanezca en el estado anterior una vez efectuada la transición. Las funciones de distribución de probabilidad se fijan para cada estado. Tendremos, por consiguiente, tantas distribuciones de probabilidad de observación que representan variables aleatorias o procesos estocásticos como estados tenga el proceso. Una vez que la transición ha sido efectuada se hace una nueva observación.

Se llaman modelos ocultos debido a que tienen asociado un proceso no observable directamente correspondiente a las transiciones entre los diferentes estados internos, y sólo es observable la salida de las funciones aleatorias asociadas a cada estado, es decir, los vectores de parámetros o secuencia de señales que emiten dichos estados y que forman el patrón a identificar; permaneciendo así el modelo inaccesible.

Formalmente, un modelo oculto de Markov se describe como [JEL98]:

- $S = \{s_1, \cdots, s_N\}$ el conjunto de estados, que representa el espacio de estados, s_t denota al estado en el instante t.

- $O = \{o_1, o_2, ..., o_T\}$ el alfabeto de observación de salida. Estos símbolos corresponden a la salida física del sistema que se está modelando.

- $X = X_1, X_2, ..., X_t, ...$ la salida observada de un HMM.

- $\Pi = \{\pi_1, \cdots, \pi_N\}$ el conjunto de probabilidades iniciales para cada estado, asigna a cada estado una probabilidad inicial tal que $\pi_i = P(s_0 = i)$, con $1 \le i \le N$ y $\sum_{i=0}^{N} \pi_i = 1$.

- $A = \{a_{ij}\}$ la matriz de probabilidades de transición, donde a_{ij} es la probabilidad de pasar del estado i al estado j independientemente del tiempo $a_{ij} = P(s_t = j \mid s_{t-1} = i)$, con $i, j = 1, \ldots, N$; $a_{ij} \geq 0$ y $\sum_{j=1}^{N} a_{ij} = 1$.

- $B = \{b_i(k)\}$ la matriz de probabilidades de salida para cada estado, donde $b_i(k)$ es la probabilidad de que se genere símbolo o_k estando en el estado i independientemente del tiempo. $b_i(k)$ es diferente para cada estado y símbolo. $b_i(k) = P(X_t = o_k \mid s_t = i)$ con $j = 1, \ldots, N$; $k = 1, \ldots, M$; $b_i(k) \geq 0$ y $\sum_{k=1}^{M} b_i(k) = 1$.

  Es la distribución más importante ya que hace referencia directa a los símbolos observados.

Para completar la especificación del HMM, falta definir los parámetros N, M y T que son, respectivamente, el número de estados, el número de observaciones y la longitud del alfabeto de salida. El HMM completo se define como: λ=(A, B, Π).

En la siguiente figura muestra un ejemplo de HMM:

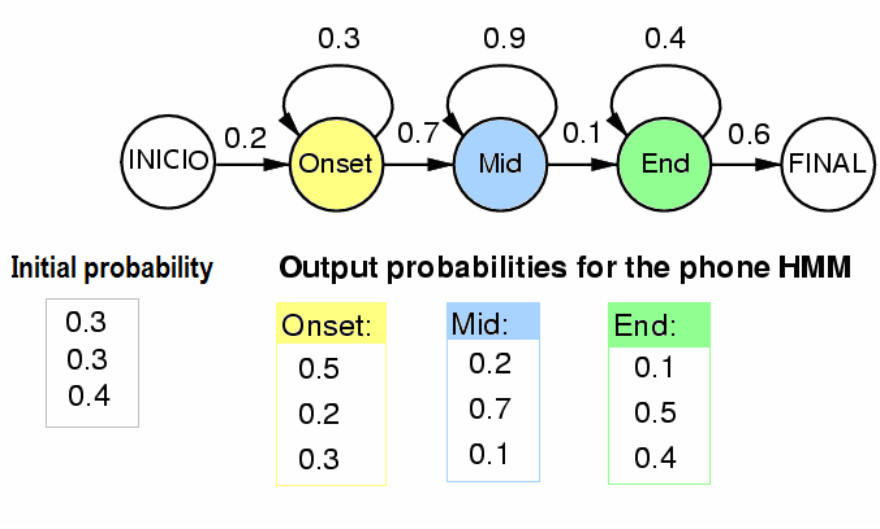

Figura 2.4: Ejemplo de HMM

El modelo descrito anteriormente es un HMM de primer orden en el cual cada estado produce un solo símbolo y el conjunto de observaciones coincide con el conjunto de estados y que asume:

- La suposición de Markov: $P(s_t \mid s_1^{t-1}) = P(s_t \mid s_{t-1})$, siendo s_1^{t-1} la secuencia de estados $s_1, s_2, ..., s_{t-1}$.
- La suposición de independencia de la salida: $P(X_t \mid X_1^{t-1}, s_1^t) = P(X_t \mid s_t)$, es decir, la probabilidad de emitir un símbolo de salida en el instante t sólo depende del estado s_t y es independiente de observaciones pasadas.

Una vez obtenida la definición del HMM, dados un modelo λ y una secuencia de observaciones $X = (X_1, X_2, ..., X_t)$, existen tres problemas básicos que hay que resolver:

1. El problema de la evaluación. ¿cuál es la probabilidad $P(X \mid \lambda)$ de que el modelo λ genere la observación X?
2. El problema de la decodificación. ¿cuál es la secuencia de estados $S = (s_1, s_2, ..., s_t)$ más probable que produce la observación X?
3. El problema del aprendizaje: ¿Cómo se pueden ajustar los parámetros del modelo λ para maximizar la probabilidad conjunta $\prod P(X \mid \lambda)$?

3.2.1 Evaluación de un HMM.

El primer problema que se plantea es obtener la probabilidad de que aparezca una serie de observaciones para un modelo dado. El problema puede verse también como la evaluación del modelo, dada una observación concreta. Elegir un modelo de entre varios posibles que se ajuste mejor a una observación dada.

El método más directo para calcular la probabilidad $P(X \mid \lambda)$ es combinar las probabilidades de todas las posibles secuencias de estados. En cada instante $t = 1, 2, ..., T$ se tienen N posibles estados alcanzables, por lo tanto N^T operaciones, por lo que es computacionalmente no realizable. Sin embargo, existe un algoritmo más sencillo y eficiente llamado algoritmo de avance (*forward algorithm*). La idea básica de este algoritmo es calcular la probabilidad de la secuencia de observación parcial $X' = (X'_1, X'_2, ..., X'_t)$, de esta manera el número de operaciones se reduce a N^2T.

3.2.2 Decodificación de un HMM.

Para muchas aplicaciones basadas en HMMs, incluido el reconocimiento automático del habla, encontrar el mejor camino es esencial. Dicho camino es la secuencia de estados que tiene mayor probabilidad de ser reconocida dada una secuencia de observaciones. Un algoritmo eficiente para encontrar la mejor secuencia de estados es el algoritmo de Viterbi.

El algoritmo de Viterbi es un método de programación dinámica y que obtiene la probabilidad del mejor camino, es decir, dado un modelo de Markov y una secuencia de símbolos, el algoritmo de Viterbi encuentra la secuencia de estados del modelo de Markov que con mayor verosimilitud ha podido producir dicha secuencia. El algoritmo de Viterbi es una simplificación del algoritmo de avance en donde sólo se considera la probabilidad máxima acumulada entre de los distintos caminos óptimos que llevan a un mismo estado, es decir, sólo se considera la secuencia de estados que resulta de maximizar las expresiones del algoritmo de avance. En general, la secuencia es globalmente óptima una vez garantizada la optimación local. Aún así, la probabilidad es todavía incompleta porque no consideramos más que las mejores transiciones en cada estado y tiempo.

A cada posible secuencia de estados le corresponde un camino, y la longitud total del camino representa (salvo la probabilidad inicial) la inversa de la probabilidad de que la secuencia a reconocer haya sido producida por la secuencia de estados. Así pues, el camino de mínima longitud nos proporcionará la secuencia buscada de estados de máxima probabilidad.

3.2.3 Estimación de los parámetros de un HMM, aprendizaje.

La estimación exacta de los parámetros del modelo es el problema más difícil, y uno de los primeros problemas del entrenamiento de HMM es conseguir una buena estimación inicial de los parámetros.

La estimación de los parámetros consiste en una expresión recurrente de los parámetros de definición de λ, más la actualización que pretende la adaptación a los patrones de entrenamiento. Hay que replantear la reescritura de las ecuaciones para cada uno de los parámetros. Se inventa una nueva cantidad, la probabilidad de transiciones

del estado S_1 al estado S_2 con la observación x_t. A partir de estas ecuaciones se escriben las fórmulas de reestimación de los parámetros de un modelo λ, dado el número de estados, una descripción de los observables, y una estimación aceptable de las probabilidades. Así obtenemos un nuevo modelo $\overline{\lambda}$. Para este nuevo modelo se puede demostrar que o bien $\overline{\lambda} = \lambda$, en cuyo caso estamos en un punto crítico donde todas las fórmulas de reestimación son exactamente correctas, o bien el nuevo modelo se ajusta mejor, de modo que $P(X|\overline{\lambda}) = P(X|\lambda)$. Además se puede demostrar que las cantidades cumplen las restricciones de probabilidad.

El problema se puede resolver mediante el algoritmo de *Baum-Welch* (también conocido como algoritmo de avance y retroceso, *forward and backward algorithms*), basado en el algoritmo de maximización de la expectativa (*Expectation Maximization*). El algoritmo *Baum-Welch* permite entrenar las probabilidades de transición entre los estados y las probabilidades que emite cada estado de manera iterativa.

3.2.4 Alineamiento.

En el caso concreto de reconocimiento de palabras conectadas es necesario obtener la secuencia de modelos que maximice la probabilidad de aparición de una cierta secuencia de palabras. Se realiza el cálculo de la probabilidad de emisión, para cada HMM, partiendo de un mismo tiempo dentro de la secuencia de observación. La cadena de modelos se obtiene realizando un rastreo retrospectivo de la cadena con mayor probabilidad de emisión.

3.2.5 Modelos con transiciones nulas y estados ligados.

Las transiciones nulas permiten obviar estados de un HMM. No producen salida y se realizan en un tiempo nulo. Permiten utilizar un mismo conjunto de entrenamiento para un mismo HMM en el cual se dispone de muestras en las que un determinado fragmento sonoro puede no aparecer como consecuencia de diferentes pronunciaciones. Por tanto se convierte en un entrenamiento unificado de dos HMM que de otro modo habría que entrenar de forma paralela.

En cuanto a los estados ligados, en éstos se permite establecer una equivalencia entre parámetros de HMM en diferentes estados. Como ventaja, se ofrece una

estimación más simple, una reducción de parámetros del modelo y una disminución en el número necesario de muestras, ya que se utilizan partes ya entrenadas de un HMM para ser ligadas a otros HMM en los que posiblemente se necesiten muchas más muestras.

3.2.6 Conectividad de los estados.

En principio, los procesos Markovianos consideran una matriz de probabilidades de transición sin restricciones, es decir, cualquier estado se puede alcanzar desde cualquier otro. En el caso de los HMM reales, al entrenar un modelo con una serie de muestras de voz, es el proceso de adaptación el que pone de manifiesto la escasa probabilidad de determinadas transiciones. En la práctica este tipo de modelo no se utiliza más que en ocasiones especiales.

Uno de los modelos más utilizados es el que se denomina modelo de izquierda-derecha, es el actual responsable de la más exitosa tecnología en muchos reconocedores de habla y otras aplicaciones de procesamiento de voz. Su éxito se debe a su rigurosa formulación matemática para determinar de manera óptima los parámetros de un modelo dado. En él se introduce una relación de orden parcial entre los estados en función de su precedencia temporal de modo que se prohíben las transiciones hacia atrás y aquellas que introduzcan una compresión excesiva de la escala temporal. La ventaja de este modelo es el ahorro de tiempo en la etapa de entrenamiento y en la etapa de obtención de probabilidades de observación.

3.2.7 Aplicación de los HMM en reconocimiento de voz.

Para aplicar la teoría de los HMM´s en reconocimiento de voz, se representa cada palabra del vocabulario del reconocedor como un modelo generativo (que se calculará en la fase de entrenamiento) y posteriormente, se calcula la probabilidad de que la palabra a reconocer haya sido producida por algunos de los modelos del reconocedor. Para ello, se asume que durante la pronunciación de una palabra el aparato fonador puede adoptar sólo un número finito de configuraciones articulatorias o estados, y que desde cada uno de esos estados se producen uno o varios vectores de observación cuyas características espectrales dependerán del estado en el que se hayan generado. Vista la generación de la palabra, las características espectrales de cada fragmento de señal dependen del estado activo en cada instante, y las características del espectro de la

señal durante la pronunciación de una palabra dependen de la función de transición entre estados.

Además existe una gran variedad de tipos de ruido que afectan el ambiente acústico de un sistema reconocedor de voz, por lo que siempre es bueno tener un conocimiento *a priori* de la naturaleza del ruido. Es importante crear, además de los modelos de las unidades acústicas, uno o más **modelos de relleno** entre los cuales el más significativo es el silencio; sin embargo, para evitar confusión entre los sonidos sordos (principalmente los fricativos) y segmentos sin habla contaminados con ruido, sólo se usan segmentos de más de 10 milisegundos para entrenar el modelo del silencio. Así, dado que los sonidos fricativos u otros fonemas sordos no duran más de 10 milisegundos, el sistema rara vez los confundirá con intervalos de ruido puro.

Entre las principales ventajas que trae consigo la aplicación de HMMs al problema de reconocimiento de palabras conectadas, está la técnica que incluye un mecanismo intrínseco de separación de palabras basado en una adecuada modelización del silencio, lo cual elimina la necesidad de realizar una detección de bordes previa.

Por último hay que mencionar que existe otra alternativa a los modelos acústicos de fonemas (que suelen usar HMM de 3 o 5 estados para cada unidad fonética), que es la de los modelos acústicos de palabras. En vez de usar HMM de fonemas e ir encadenándolos para formar palabras, se puede construir un HMM con más estados (el número de estados elegido dependerá de la longitud de la palabra). Esta alternativa tiene ventajas e inconvenientes con respecto a los modelos de fonemas. La ventaja principal es que la tarea del entrenamiento se simplifica bastante, ya que sólo hay que entrenar las palabras necesarias, mientras que con los modelos de fonemas hay que entrenarlos todos, con vistas de que en el futuro se puedan reconocer nuevas palabra. Pero esta alternativa tiene un gran inconveniente, si se quieren añadir nuevas palabras a reconocer, hay que volver a entrenar el sistema con las nuevas palabras. Esto no pasa con los modelos de fonemas que se entrenan una sola vez.

3.3 Modelo de lenguaje o gramática.

Adivinar la siguiente palabra, o lo que se llama *predicción de la palabra,* es una subtarea esencial en el reconocimiento de voz. Puesto que la identificación de una palabra es difícil debido a que la entrada es ruidosa o ambigua, observando la palabra o palabras previas, nos puede dar una pista importante sobre cual puede ser la palabra que intentamos reconocer. Para este proyecto se ha empleado una gramática restringida. Estas gramáticas se definen por medio de maquinas de estados finitos y expresiones regulares. Este modelo es usado para sistemas de reconocimiento de voz sencillos, en los cuales los enunciados de entrada son limitados, es decir, al usuario sólo se le permite decir aquellos enunciados que están explícitamente cubiertos por la gramática. Dado que a nosotros no nos interesa reconocer solamente algunas frases necesarias para las tareas cotidianas en una vivienda, optamos por usar este tipo de modelo.

3.4 Diccionario de pronunciación.

Una parte muy importante en el reconocimiento de voz es el diccionario de pronunciación. En él se especifica la secuencia de sonidos (representados mediante un conjunto de símbolos) que componen una palabra. Los símbolos pueden ser definidos específicamente para la tarea de reconocimiento, o bien, obtenidos de un alfabeto fonético.

4. Herramienta empleada.

Para el presente proyecto se ha empleado la herramienta HTK, una aplicación de código abierto desarrollada en el Departamento de Ingeniería de la Universidad de Cambridge. HTK esta orientado al diseño, entrenamiento y evaluación de HMMs basado en la tecnología izquierda-derecha Con esta herramienta es posible diseñar HMMs en los que la probabilidad de emisión de símbolos se obtenga por mezclas de múltiples funciones de densidad de probabilidad continúa.

Capítulo 3

LA DOMÓTICA

1. Introducción.

La domótica es la instalación e integración de varias redes y dispositivos electrónicos en el hogar, que permiten la automatización de actividades cotidianas y el control local o remoto de la vivienda o del edificio. La domótica no son servicios o productos aislados, sino simplemente la implementación e integración de todos los aparatos del hogar.

La domótica se aplica a la ciencia y a los elementos desarrollados por ella que proporcionan algún nivel de automatización o automatismo dentro de la casa; pudiendo ser desde un simple temporizador para encender y apagar una luz o aparato a una hora determinada, hasta los más complejos sistemas capaces de interactuar con cualquier elemento eléctrico de la casa.

La vivienda domótica es, por lo tanto, aquella que integra una serie de automatismos en materia de electricidad, electrónica, robótica, informática y telecomunicaciones, con el objetivo de asegurar al usuario un aumento del confort, de la seguridad, del ahorro energético, de las facilidades de comunicación, y de las posibilidades de entretenimiento. La domótica, pues, busca la integración de todos los aparatos del hogar, de manera que todo funcione en perfecta armonía, con la máxima utilidad y con la mínima intervención por parte del usuario.

En este contexto se suele utilizar también mucho el concepto de hogar inteligente el cual era empleado antes de que naciese el de domótica. El término inteligente se utiliza en ámbitos informáticos para distinguir aquellos terminales con

capacidad autónoma de procesamiento de datos. La domótica también se suele asociar actualmente, sobre todo en ámbitos de telecomunicaciones, al denominado hogar digital u hogar conectado.

En realidad no existen acusadas diferencias entre una vivienda tradicional y otra con equipamiento domótico. Se trata de la misma vivienda, con equipamiento semejante, y con el mismo diseño arquitectónico. La diferencia sólo estriba en la incorporación de una mínima tecnología, que permita gestionar de forma más eficiente e integrar los distintos equipos e instalaciones domésticas que conforman la vivienda.

2. Beneficios de la domótica.

El principal problema que se encontraba para la introducción de la domótica era que muy pocas personas estaban dispuestas a pagar los costes adicionales que implica construir una vivienda inteligente, pero el actual descenso de los precios ha hecho de la vivienda domótica un sueño asequible. Es necesario agregar fuerzas y ofrecer al usuario soluciones integradas que le faciliten a vida, pues en otro caso este mercado nunca acabará de explotar.

La domótica proporciona un sinfín de beneficios para el usuario del edificio, el principal beneficiado por la incorporación de sistemas inteligentes, englobados en un incremento de la seguridad, de la comodidad, del ahorro energético y de las posibilidades de entretenimiento y comunicación, todos ellos personalizados para cada tipo de usuario e inalcanzables mediante los edificios tradicionales. Además ofrece nuevas oportunidades de negocio a todos los actores consolidados y abre también las puertas a nuevas empresas que buscan posicionarse en este atractivo mercado.

Pero para que la proliferación de edificios inteligentes sea una realidad son necesarios ciertos cambios en los agentes tradicionales de la construcción y equipamiento del hogar, así como la incorporación de nuevos actores que permitan la gestión integrada se ese hogar conectado.

El usuario desconoce estos beneficios y piensa que la tecnología es muy cara y compleja, aunque el coste de está es un porcentaje muy pequeño para el incremento de la calidad de vida que le reporta a su usuario.

3. Aplicaciones.

La incorporación e integración de redes y dispositivos en la vivienda domótica posibilitan una cantidad ilimitada de nuevas aplicaciones y servicios en el hogar, consiguiendo así un mayor nivel de confort, se aumenta la seguridad, se reduce el consumo energético, se incrementan las posibilidades de ocio, etc. En definitiva, se produce un incremento de la calidad de vida de sus habitantes. Las principales aplicaciones de los edificios inteligentes para sus usuarios son:

Comodidad y confort:

- Control centralizado del hogar accesible a través del teléfono móvil o Internet, teniendo la posibilidad de controlar remotamente el hogar.
- Control de los dispositivos eléctricos y electrónicos del hogar desde cualquier sitio.
- Teléfono manos libres.
- Teletrabajo, telecompra, telebanca, , ,…
- Automatización de funciones y programación el sistema para que realice automáticamente las tareas.
- Integrar el portero automático con el teléfono fijo o móvil.
- Lectura remota de contadores.
- Electrodomésticos y aparatos electrónicos inteligentes.

Seguridad

- Detector de intrusión, simulador de presencia.
- Simular que la vivienda está ocupada.
- Detección de incendios, de escapes de gas y agua, cortes de electricidad
- Configuración de procedimientos de avisos en caso de intrusión o avería.
- Teleasistencia, teleseguridad y video vigilancia.

- Instalación de cámaras y micrófonos para ver y escuchar la que pasa, con posibilidad de grabación.
- Control de acceso a la vivienda.
- Puertas acorazadas y cajas fuertes.
- Sensores de alarma y botones de pánico.
- Seguridad de la comunicaciones, seguridad de los datos informáticos, cortafuegos, evitar que intrusos accedan a la red interna de nuestro hogar.

Ocio y comunicaciones

- Internet con conexión permanente de banda ancha desde cualquier punto.
- Recursos informáticos, digitales y acceso a Internet compartido por varios dispositivos
- Visión de canales de televisión en cualquier habitación.
- Televisión digital interactiva, video bajo demanda, cine en casa, …
- Videojuegos en red, videoconferencia, voz sobre IP, …

Ahorro energético

- Programación del encendido y apagado de todo tipo de aparatos (calderas, aire acondicionado, toldos, luces, etc.), según las condiciones ambientales.
- Programación y zonificación de la temperatura, regulación automática de la intensidad luminosa según el nivel de luz natural, …
- Acomodación a los planes de tarifas reducidas.
- Automatización de funciones y control de operaciones realizado por el sistema centralizado.
- Incorporación de electrodomésticos de última generación.
- Contadores electrónicos.

Todos estos ejemplos de cómo la domótica puede mejorar la comodidad en el hogar son claramente trasladables a otros edificios, incluso en otro tipo de edificios determinados a un mayor número de personas (hospital, colegio, universidad, hotel, oficina, residencia de la tercera edad, aeropuerto, supermercados, etc.), las soluciones a introducir pueden ser aún más avanzadas. El conjunto de todas estas aplicaciones tiene el objetivo básico de incrementar la calidad de vida de los usuarios, independientemente del tipo concreto de edificio considerado.

4. La interfaz de voz.

Es fundamental, para que el usuario pueda apreciar estas aplicaciones, que la tecnología necesaria para hacerlas realidad, le sea totalmente transparente. El usuario no quiere tener que estar consultando continuamente vastos y aburridos manuales, prefiere sistemas intuitivos y sencillos. Es más, la verdadera inteligencia de los dispositivos inteligentes debe residir en su capacidad de liberar al hombre de un esfuerzo continuo de observación y corrección. Ahí es donde entra en juego la interfaz de voz.

Los sistemas de control centralizado suelen integrar una interfaz de voz, que permite al usuario conocer o programar el estado del edificio en cualquier momento y desde cualquier lugar mediante teléfono fijo o móvil.

La utilización de un reconocedor del habla dentro de un contexto permite que el sistema sea capaz de entender lo que ha dicho el usuario llevando acabo las acciones indicadas con una comunicación más intuitiva y natural. El reconocedor de voz, además de mejorar a comodidad para los usuarios normales que pueden conectarse de forma remota, también puede ayudar a personas con deficiencias visuales. Asimismo, supone una mejora de la seguridad por la posibilidad de determinar si una persona es quien dice ser a través del análisis de su voz.

5. Especificaciones de un sistema real.

Si se tratara de implementar el reconocedor de voz en un sistema físico real, ya sea una maqueta o una casa, los requerimientos funcionales no variarían demasiado con respecto a las de este proyecto; ya que la parte del reconocedor de voz y la aplicación interna serían exactamente las mismas.

Así pues, en el supuesto caso de implementar el sistema de reconocimiento de voz en una casa real, la aplicación en vez de enviar el valor único de la frase reconocida a una interfaz gráfica, lo enviaría a un PLC (***P****rogrammable* ***L****ogic* ***C****ontroller* o Controlador lógico programable) que activará y/o desactivará los actuadores

correspondientes tales como: interruptores de la luz, electrodomésticos o motores de las puertas, ventanas y persianas.

En cuanto a los requerimientos no funcionales de un sistema físico real, serían muy extensos, puesto que se necesita gran variedad de elementos electrónicos anexados a puertas, ventanas y electrodomésticos entre otros; así como sensores de presencia, temperatura, etc. un PLC y el software para programarlo además de todo el cableado necesario y sus protocolos de transmisión.

Capítulo 4

ANÁLISIS DE ESPECIFICACIONES

1. Especificaciones del proyecto.

En primer lugar hay que remarcar el hecho de que el reconocedor de voz no se va a implementar en un sistema físico real, sino que se va a desarrollar un simulador de una vivienda domótica, ya que esto modifica un poco las especificaciones del proyecto. Por lo tanto el proyecto se divide en dos partes, el reconocedor de voz y el simulador. A su vez, el reconocedor también se divide en la creación y entrenamiento de modelos acústicos y decodificación.

1.1 Requerimientos funcionales para el reconocedor de voz.

A continuación se detallan los requerimientos funcionales del proyecto teniendo en cuenta sólo el reconocedor de voz. En el siguiente apartado se tratarán los requerimientos relacionados con el simulador.

1.1.1 Modelos acústicos.

En cuanto al reconocedor de voz, se trata de crear unos modelos acústicos basados en modelos ocultos de Markov, los cuales han sido explicados con detalle en el capítulo 2.

Para la creación de los modelos acústicos son necesarios algunos elementos tales como un diccionario con todas las palabras admitidas por el sistema y su correspondiente pronunciación, una gramática restringida la cual solamente permite un conjunto finito de frases y que precisan todas las que el sistema ha de reconocer. Dicha gramática está definida por un lenguaje independiente del contexto. Junto a todo esto también son imprescindibles un número elevado de archivos de audio, grabados con los

usuarios finales de la aplicación, cuyo formato es mono de 16KHz y 16 bits por muestra. Es recomendable que haya suficientes muestras de cada una de las palabras admitidas para un mejor rendimiento del reconocedor de voz.

Llegados a este punto ya se tiene todo lo necesario para la realización de modelos acústicos para las unidades fonéticas. El primer paso para la creación de los modelos acústicos se trata de delimitar y etiquetar cada uno de los fonemas que componen la señal acústica necesarios para identificar todas las posibles palabras que acepta nuestro reconocedor. Se pueden etiquetar tantos fragmentos de onda como se desee, habiendo como mínimo tres ejemplos de cada uno. Cuantos más se etiqueten, más fácil será para el reconocedor identificar cada fonema.

1.1.2 Requerimientos funcionales para la decodificación.

Una vez obtenidos, el reconocedor de voz ha de ser capaz de analizar en tiempo real una entrada de audio, compararla con los modelos acústicos obtenidos y tener una tasa de acierto superior al 95%. Para comprobar su rendimiento, se utilizan nuevas entradas de audio válidas que no se hayan empleado en el proceso de creación y entrenamiento de los modelos acústicos, con el mismo formato que las usadas anteriormente, y se comprueba que más del 95% de ellas hayan sido reconocidas con éxito. En caso de no ser así hay que mejorar los resultados obtenidos, existen varios métodos para llevarlo a cabo; se pueden grabar más archivos de audio, etiquetar más fragmentos de onda y/o reentrenar los modelos acústicos hasta lograr los objetivos.

1.2 Especificaciones para el simulador de domótica.

Una vez logrado un reconocedor de voz robusto para nuestras necesidades, se ha de desarrollar una aplicación gráfica que simule un edificio domótico y en la cual se visualicen de forma clara e inmediata las diferentes acciones mencionadas por el usuario. Se ha de crear una aplicación que muestra la planta de una vivienda en 3D en la que la aplicación navegara por su interior y con todos los elementos necesarios para ver que realmente se ejecuta la orden pronunciada.

Como se ha expuesto anteriormente, la aplicación incorpora el reconocedor de voz desarrollado para analizar la orden dicha por el usuario. Pero para poder utilizarlo, antes hay que adaptarlo según el formato de la librería empleada.

Esta aplicación, en primer lugar, recibe la entrada de audio (frase que pronuncia el usuario) a través de un micrófono previamente configurado. Mediante el reconocedor de voz desarrollado, se analiza la señal acústica para identificar la frase formulada por el usuario.

En caso de ser reconocida y ésta coincide con una de las acciones permitidas, se analiza en profundidad y se le asigna un valor único en toda la aplicación. Este valor es enviado a la interfaz gráfica que lo relaciona con una de las animaciones que posee en su base de datos. Para reproducir la animación correspondiente a la frase dicha, se posiciona la cámara y el punto de vista en la habitación adecuada y con una orientación apropiada para ver la reproducción gráfica de la orden mencionada por el usuario.

En caso contrario, cuando una de las frases no ha sido reconocida o bien no corresponde a ninguna acción válida, se muestra un mensaje en la pantalla para advertir al usuario de que no se ha podido reconocer la oración formulada.

También existe la posibilidad de que el reconocedor de voz reconozca una frase diferente a la pronunciada por el usuario. En tal caso la aplicación realiza la orden interpretada por el reconocedor. Para corregir el fallo, el usuario debe formular otra acción para corregir la anterior.

1.3 Requerimientos no funcionales.

Para este proyecto se usa la herramienta de software libre HTK que permite la creación y entrenamiento de los modelos acústicos. También trae integradas aplicaciones: para grabar los archivos de audio necesarios, para la decodificación y reconocimiento de frases tanto desde archivos de sonido como en tiempo real y para comprobar el rendimiento de los modelos acústicos.

Asimismo, la herramienta de desarrollo *Julius*, también de software libre, permite la integración de los modelos acústicos creados con HTK en nuestras aplicaciones e incorpora aplicaciones para realizar esta adaptación. Con apoyo de la API de *Julius*, la cual esta desarrollada en C, se crea nuestro propio reconocedor de voz, o decodificador, y se incorpora en el simulador.

El uso de esta API, implica que la aplicación debe desarrollarse con un lenguaje de programación que admita anexar librerías hechas en C, si no es en toda su totalidad, como mínimo el software de reconocimiento de voz.

Capítulo 5

DISEÑO

Primero recordar que el objetivo de este proyecto es desarrollar un reconocedor de órdenes sencillas para una aplicación de domótica. Algunos ejemplos de órdenes son:

- Abre la puerta de la cocina.
- Cierra la ventana de la habitación.
- Enciende la luz.
- Apaga la calefacción.

Todo el proyecto realizado puede ser dividido en cuatro bloques bien diferenciados:

1. Creación, entrenamiento y prueba de los modelos acústicos. En este bloque se aprendió a crear y entrenar modelos acústicos con HTK. Se obtuvieron unos modelos para los fonemas que componen nuestras frases y se observaron las limitaciones que había para conseguir unos modelos robustos.
2. Creación de la aplicación reconocedora de voz utilizando los modelos acústicos elaborados. En este bloque se creó una aplicación que recibe una entrada de audio por el micrófono y la analiza para identificar la frase mencionada por el usuario. Se empleó el software de desarrollo *Julius* para integrar en la aplicación los modelos acústicos obtenidos anteriormente.
3. Creación de la aplicación gráfica. En este bloque se creó una aplicación gráfica que simula una vivienda domótica donde se reproducen las órdenes enviadas por el reconocedor y que previamente han sido admitidas.
4. Por último, se desarrolló el nexo de comunicación entre ambas partes de la aplicación.

1. Estructura de la aplicación completa

La aplicación desarrollada se estructura en cuatro partes bien diferenciadas, los modelos acústicos, el reconocedor de voz, la aplicación gráfica que simula una vivienda domótica y la comunicación entre ambos procesos de la aplicación. El siguiente diagrama ilustra dicha estructura y a continuación se detalla la estructura del reconocedor de voz y el simulador de domótica.

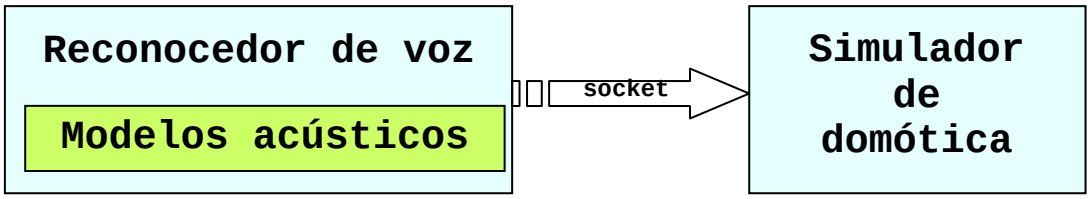

Figura 5.1: Diagrama de la aplicación completa.

Para el reconocedor de voz se emplea un único fichero tal y como muestra el diagrama siguiente.

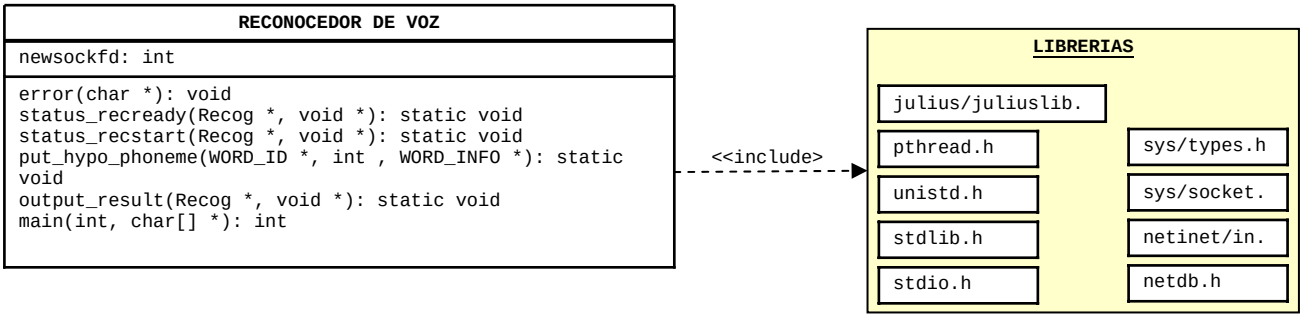

Figura 5.2: Diagrama de la aplicación reconocedor de voz.

Sin embargo, la parte del simulador de voz incluye varios ficheros, dos de ellos dedicados exclusivamente para cargar las texturas que utilizadas para la estructura de la casa. El resto de acciones tales como traslados de la cámara o visualizado de objetos tridimensionales, están incluidas en el fichero principal. El la figura 4.2 muestra la estructura de la aplicación gráfica, incluyendo variables, funciones, librerías y estructuras.

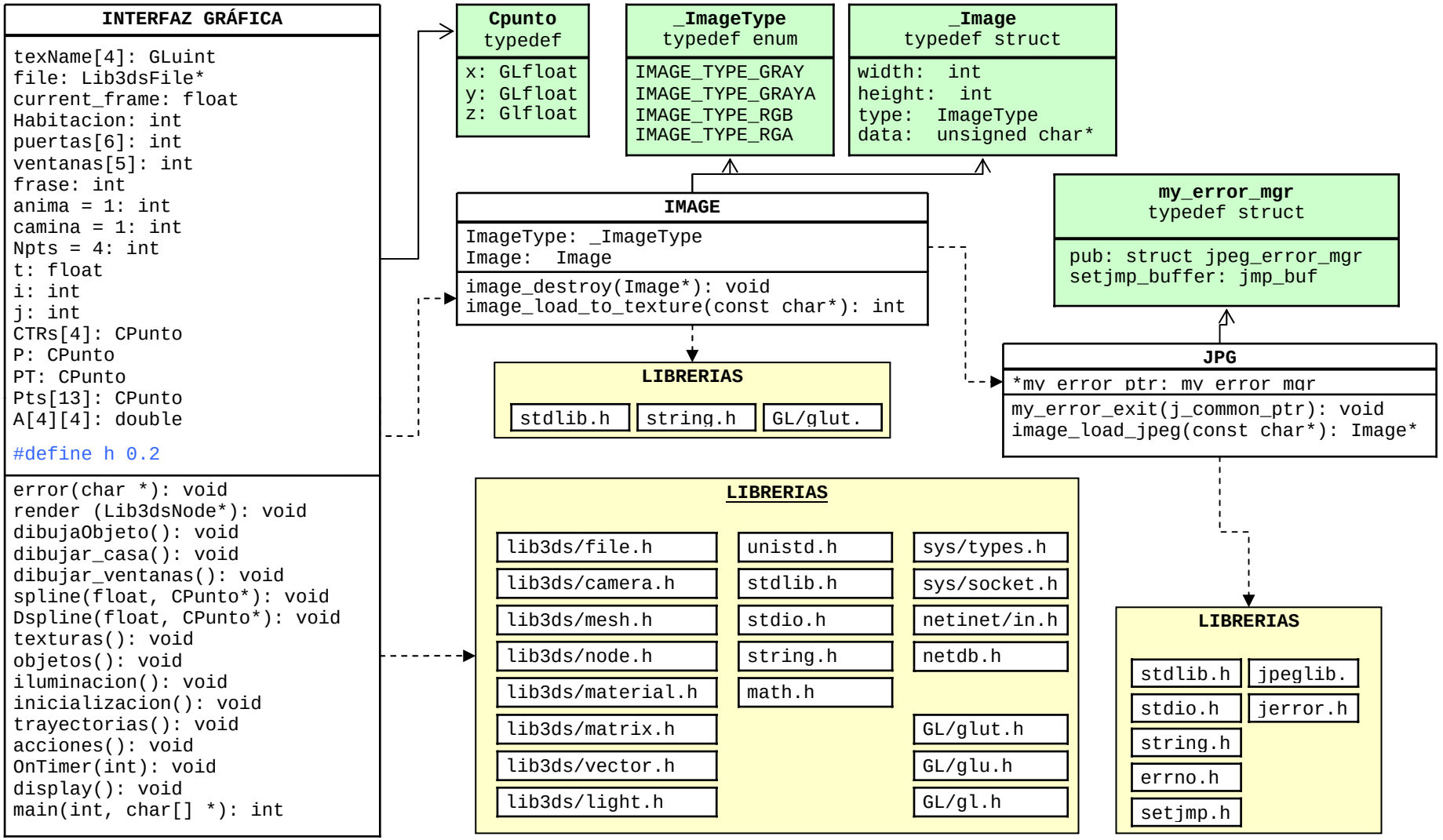

Figura 5.3: Diagrama de la aplicación gráfica, simulador de domótica.

1.1 Ficheros adicionales.

La aplicación desarrollada necesita información externa, no incluida en el código, para funcionar y toda la que requiere se encuentra almacenada en ficheros de texto y ficheros binarios en lugar de bases de datos. Los ficheros empleados son los siguientes: fichero de configuración de Julius, modelos acústicos y un listado de éstos, gramática, diccionario, texturas e imágenes. En caso que la aplicación trabaje con grabaciones de voz en vez de un realizar un reconocimiento en vivo, directamente des del micrófono, también serán necesarios dichos archivos de audio, un listados de éstos y las transcripciones.

El fichero de configuración de Julius empleado para que la aplicación de reconocimiento de voz trabaje con entradas de audio desde un dispositivo de captura de sonido, especifica a Julius cuales son los ficheros que contienen la gramática, el diccionario y los modelos acústicos, entre otros parámetros como por ejemplo el uso del modelo de pausa corta.

Los modelos acústicos utilizados por la aplicación y el listado de éstos, son exactamente los que se han generado con HTK en la primera parte de este proyecto. Hay que tener en cuenta que Julius necesita que el fichero que contiene los modelos acústicos no sea binario, sino que esté escrito en texto plano, por tanto habrá que generarlo de nuevo teniendo esto en cuenta. En cuanto a la gramática y el diccionario que definen el lenguaje admitido por el reconocedor, también podemos reaprovechar los que se generaron con HTK pero adaptándolos al formato de *Julius*.

Los ficheros de texturas empleados son imágenes cuadradas en formato *jpg*. Sin embargo las imágenes tridimensionales que se visualizan en el simulador son objetos en formato *3ds* que a su vez también pueden incluir su propia textura en ficheros *jpg* o mapa de bits.

En caso de trabajar con ficheros de audio almacenados en disco, de igual modo que HTK, se necesita tener las transcripciones de cada uno de los archivos en un fichero de texto y otro con las rutas a cada uno de éstos archivos. Las grabaciones tienen que estar grabadas con formato mono de 16KHz y 16bits por muestra, igual que las utilizadas para la creación y entrenamiento de los modelos acústicos.

1.2 Librerías.

Las principales librerías empleadas para el desarrollo de la aplicación, además de otras librerías estándar de Linux y C, son: Julius, OpenGL + GLUT y Lib3ds.

Julius es una herramienta de código libre, con un alto rendimiento, utilizada para reconocimiento de voz en discurso continuo, que permite utilizar los modelos acústicos entrenados con HTK. Para ello incorpora herramientas, también de código libre, para adaptar los modelos acústicos generados y entrenados con HTK al formato que necesita. Puede realizar, casi en tiempo real, la decodificación de un discurso de habla continua en la mayoría de los ordenadores actuales utilizando muy poca cantidad de memoria ya que incluye las principales técnicas de búsqueda y es independiente de la estructura de los modelos acústicos. Además, Julius posee su propia API de desarrollo, que nos permite construir aplicaciones basadas en reconocimiento de voz para aplicaciones diversas. Con Julius se puede realizar el reconocimiento de archivos de audio, una entrada de micrófono, una entrada de red y definir distintas características de configuración mediante un archivo de parámetros. La biblioteca, llamada *JuliusLib*, está completamente escrita en ANSI-C portable para lograr más independencia de hardware. Otras de las principales características de esta librería son:

- Altamente configurable una variedad diversa de parámetros de búsqueda.
- Reconocimiento sobre la marcha para entradas desde micrófono y desde la red.
- Permite el uso del modelo de pausa corta.

OpenGL (*Open Graphics Library*) es una biblioteca de código abierto para desarrollo de aplicaciones interactivas y que requieran de computación visual de alta calidad y rendimiento en 2D y 3D, como por ejemplo animaciones y simulaciones. Incorpora un amplio conjunto de prestaciones como el mapeo de texturas, efectos especiales, y otras potentes funciones de visualización. OpenGL es la única librería verdaderamente abierta, de proveedor neutral, multilenguaje y multiplataforma totalmente portable, por lo que se puede garantizar una amplia distribución de las aplicaciones desarrolladas con ésta en gran variedad de sistemas. Aunque oculta la complejidad de la interfaz presentando al programador una única API uniforme, su diseño de bajo nivel requiere que los programadores conozcan el funcionamiento de la

pipeline gráfica; a cambio ofrece la libertad para implementar algoritmos gráficos. Otras de las principales características de esta librería son:

- Fácil de usar, bien estructurada y con un diseño intuitivo y comandos lógicos.
- Opera con imágenes así como primitivas geométricas (punto, línea o polígono).
- Usa el sistema de ventanas.

GLUT es una API simple, fácil y pequeña para escribir programas en openGL generando un sistema sencillo de ventanas independientes e interacción por medio de teclado y ratón, ya que no es un juego de herramientas con funciones completas ni para aplicaciones que requieras una interfaz de usuario sofisticada. GLUT no es de código abierto ni está en el dominio público pero es de libre distribución sin derechos de licencia y portable a casi todas las implementaciones de OpenGL y plataformas. La biblioteca GLUT es compatible con las siguientes funcionalidades:

- Varias ventanas de visualización simultáneas.
- Varios dispositivos de entrada.
- Rutina inactiva y temporizadores.
- Menús emergentes.
- Generación de sólidos y objetos de alambre.
- *Callbacks* de eventos.

Lib3ds es una biblioteca de desarrollo de software de código abierto para la gestión de archivos 3d-Studio, o dicho de otro modo, archivos con extensión *.3ds*. Se trata de una alternativa de software libre a *Autodesk 3DS File Toolkit*. La biblioteca está completamente escrita en ANSI-C portable para lograr más independencia de hardware. Algunas de las principales características de esta librería son:

- Cargar y guardar ficheros 3ds: Ajustes de la atmósfera, el fondo y el *viewport*; mapa de la configuración de jerarquía, sombras, materiales, cámaras, luces, mallas y fotogramas clave entre otros parámetros.
- Fácil manipulación de las estructuras de datos.
- Se integra completamente con OpenGL.

2. Creación, entrenamiento y prueba de los modelos acústicos.

La primera fase del proyecto consistió en la creación de unos modelos acústicos para los fonemas que componen nuestras frases. Para ello se utilizaron las herramientas para entrenamiento de modelos acústicos que incorpora HTK. Dichas herramientas necesitan que se les proporcionen grabaciones de audio de las frases, la lista de fonemas, la gramática, el diccionario de pronunciación y el fichero de transcripciones de las grabaciones de audio.

2.1 Preparación de los datos.

La primera fase consiste en la preparación de todos los datos necesarios para empezar con el entrenamiento y evaluación del sistema reconocedor.

2.1.1 Gramática de trabajo.

Para que se puedan reconocer todas las órdenes, es necesario especificarlas de alguna manera. Para ello se utiliza el lenguaje de definición de gramáticas de HTK donde | corresponde a las distintas alternativas y < > indican que se pueden producir repeticiones. También se añaden unas etiquetas de INICIO y FINAL a las que se atribuirá silencio y además se pueden reunir varias palabras en una sola para mayor comodidad, ya que algunas secuencias de palabras siempre tendrán el mismo orden.

```
$sala = COMEDOR | HABITACION | COCINA | ESTUDIO | SALADEESTAR | SALON;
$temperatura = VEINTE |  EINTIUNO | VEINTIDOS | VEINTITRES | VEINTICUATRO | VEINTICINCO;
$invento = LUZ | LUCES | CALEFACCION;
( INICIO (
  ( (ABRE | CIERRA) (LA-S) (VENTANA | PUERTA) (DEL-A) $sala ) |
  ( (ENCIENDE | APAGA) (LA-S) $invento ) |
  ( (ENCIENDE | APAGA) (EL) (AIREACONDICIONADO) ) |
  ( (FIJALATEMPERATURAA) $temperatura ) )
FINAL )
```

Aún así esta gramática es tan solo una representación dirigida al usuario para una mayor facilidad a la hora de su construcción. HTK necesita una red de palabras definidas utilizando un nivel más bajo de notación llamado SLF (*Standard Lattice Format*), donde cada palabra y cada transición palabra-palabra esté incluida explícitamente. Esta conversión puede hacerse de forma automática mediante la herramienta *HParse* que proporciona HTK.

2.1.2 Diccionario.

El siguiente paso corresponde a la construcción del diccionario. Este proyecto se corresponde a una aplicación de voz donde los modelos acústicos modelan fonemas, por tanto en el diccionario se tienen transcripciones en unidades fonéticas de cada una de las palabras que detecta el reconocedor. Estas transcripciones de cada palabra en unidades fonéticas se pueden realizar con símbolos arbitrarios.

Al final de cada transcripción se añade el símbolo sp (*short pause*), necesario para pasos posteriores. El diccionario también debe incluir las etiquetas INICIO y FINAL, que sirven para indicar el silencio que hay al principio y al final de cada frase, y hay que asegurarse que las palabras aparecen en orden alfabético. Estas son algunas de las palabras del diccionario empleado en este proyecto:

```
ABRE                a b r e sp
CIERRA              T j e rr a sp
COCINA              k o T i n a sp
ENCIENDE            e n T i e n d e sp
FIJALATEMPERATURA   f i x a l a t e m p e r a t u r a sp
FINAL []            sil
HABITACION          a b i t a T i o n sp
INICIO []           sil
LA-S                l a sp
LA-S                l a s sp
SALADEESTAR         s a l a d e e s t a r sp
SALON               s a l o n
VEITIUNO            b e j t i u n o sp
```

2.1.3 Grabación de los Datos de Voz.

HTK proporciona una herramienta gráfica llamada *HSLab* para la grabación de los archivos de audio necesarios para el entrenamiento y la evaluación de los modelos acústicos. Puesto que hubo problemas al emplear dicha aplicación porque introducía mucho ruido a la entrada de audio, se optó por la opción de grabar los ficheros de audio con otra herramienta de captura de sonido, siempre teniendo en cuenta el formato *wave* mono de 16KHz y 16 bits por muestra.

Con la herramienta *HSGen* se genera un fichero donde se indica el contenido que debe tener cada archivo de audio. Hay que tener en cuenta que cuantas más grabaciones se utilicen, mejor entrenado estará el sistema. Se realizaron 150 grabaciones, pronunciadas por un único hablante. A priori se sabe que son pocas grabaciones y hablantes, pero bastan para aprender los mecanismos de creación de modelos acústicos.

2.2 Creación de los modelos de monophones.

Los modelos de monophones, son aquellos modelos acústicos que se corresponden con fonemas independientes

2.2.1 Inicialización de los modelos acústicos.

La inicialización de los modelos acústicos se hace de forma individual para cada uno de los fonemas omitiendo el modelo sp el cual se genera posteriormente.

Lo primero que se hace para la creación de los modelos de monophones es el denominado proceso de etiquetaje de los archivos de audio, el cual consiste en delimitar los diferentes fragmentos de onda pertenecientes a cada uno de los fonemas que componen la señal de audio. Este proceso se realiza fácilmente con la herramienta *HSLab* mencionada anteriormente. Hay que tener en cuenta que para la inicialización de los modelos acústicos no se emplean todas las grabaciones realizadas anteriormente, sino sólo las grabaciones que se han etiquetado. Posteriormente, estos modelos inicializados se irán reestimando utilizando la totalidad de datos de audio.

Una vez etiquetados cada uno de los archivos escogidos, se crea un archivo con el mismo nombre del archivo de audio pero con extensión *.lab*, que contiene cada una de las etiquetas de los modelos que almacena dicho archivo, asociados a unos valores numéricos correspondientes a los valores de inicio y final. Se pueden crear tantos archivos *.lab* como se desee, habiendo como mínimo tres ejemplos de cada uno de los modelos acústicos que a entrenar. Además, no es necesario etiquetar todos los fonemas que aparecen en la señal de audio, pudiendo etiquetar tan solo aquellos que nos interesan. A continuación se muestra un ejemplo de señal etiquetada.

```
47500        9253750      sil
9448125      10731250     a
12470625     12979375     p
12981250     13727500     a
13763750     14081875     g
14077500     14431875     a
14428125     15516250     l
15515000     16013750     a
16010625     16412500     l
16411875     17419375     u
17414375     17675000     T
```

Una vez completada esta fase, se crea un fichero prototipo para cada uno de los modelos acústicos (excluyendo el modelo sp) aunque todos los modelos tengan la misma estructura. Cada uno de los prototipos tiene cinco estados de los cuales el primer y último estado deben omitirse. A cada estado se le asignan los valores de las medias y las varianzas, a 0.0 y 1.0 respectivamente, formados por vectores de tamaño 39. También se define una matriz de dimensión 5x5 de transiciones entre estados (debido a que estos modelos se componen de 5 estados). Los valores que contengan los estados inicialmente no son importantes ya que el objetivo es definir la topología. Finalmente, hay que especificar el nombre de cada uno de los modelos acústicos en la opción *h* del prototipo. El prototipo tiene el siguiente aspecto:

```
~o <VecSize> 39 <MFCC_0_D_A>
     ~h "proto"
     <BeginHMM>
      <NumStates> 5
      <State> 2
        <Mean> 39
           0.0 0.0 0.0 ...
        <Variance> 39
           1.0 1.0 1.0 ...
      <State> 3
        <Mean> 39
           0.0 0.0 0.0 ...
        <Variance> 39
           1.0 1.0 1.0 ...
      <State> 4
        <Mean> 39
           0.0 0.0 0.0 ...
        <Variance> 39
           1.0 1.0 1.0 ...
      <TransP> 5
       0.0 1.0 0.0 0.0 0.0
       0.0 0.6 0.4 0.0 0.0
       0.0 0.0 0.6 0.4 0.0
       0.0 0.0 0.0 0.7 0.3
       0.0 0.0 0.0 0.0 0.0
     <EndHMM>
```

Antes de poder ejecutar la herramienta *HInit* de HTK para la inicialización de cada uno de los modelos acústicos, hay que obtener la conversión de los archivos de audio en sus correspondientes vectores de características. Este proceso se realiza con la herramienta *Hcopy*. A su vez, ésta necesita un archivo cuyo contenido esta formado por las rutas a los archivos de audio y su correspondiente archivo de vectores de características, otro con las rutas a los archivos de salida especificados en el anterior y un fichero de configuración donde se especifican los parámetros de configuración para llevar a cabo la codificación. El contenido del fichero de configuración necesario este proyecto es el siguiente:

```
#Parámetros de codificación
TARGETKIND = MFCC_0_D_A
TARGETRATE= 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
#Si las grabaciones son wave
SOURCEKIND = WAVEFORM
SOURCEFORMAT = WAV
ZMEANSOURCE = TRUE
```

2.2.2 Entrenamiento integrado.

Una vez hemos realizados con éxito los pasos anteriores, ya se dispone de todos los elementos necesarios para poder realizar el entrenamiento del sistema reconocedor de voz. La inicialización con *HInit* sólo sirvió para realizar una primera aproximación de los modelos acústicos, en cambio, la herramienta *HERest* permite proceder al entrenamiento de los mismos. Con *HERest* se pueden realizar tantas reestimaciones de los modelos acústicos como se crea conveniente e ir almacenándolas de manera independiente.

Se crea un fichero llamado *hmacs* en el cual se añaden las primeras estimaciones de los modelos acústicos, obtenidas anteriormente con *HInit*.

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "a"
...
~h "b"
...
~h "B"
...
...
```

Seguidamente se procede a realizar las transcripciones de los archivos de audio a nivel de palabras y a nivel de fonemas. Para ello se utiliza un tipo de fichero llamado MLF (*Maste Label File*). Utilizando este formato, no es necesario tener un fichero de transcripción por cada archivo de audio, sino que se pueden juntar todas las transcripciones en un solo fichero.

HTK incluye la herramienta prompts2mlf que permite realizar una primera transcripción a nivel de palabra generando un fichero MLF. Las transcripciones a nivel de fonemas, necesarias para pasos posteriores, se hacen con la herramienta *HLEd*. Concretamente se crean dos ficheros MLF con transcripciones a nivel de fonemas: uno sin incluir el fonema sp y otro incluyéndolo (éste último es necesario más adelante cuando se inserte el modelo de pausa corta). *HLEd* precisa unos ficheros que contienen una serie de instrucciones para expandir las palabras a nivel de fonemas utilizando el diccionario, insertar el modelo sil al principio y final de cada frase y eliminar o mantener todas las ocurrencias del símbolo sp, que estaban incluidas en el diccionario de pronunciación al final de cada una de las palabras.

También se crean dos archivos que contienen la lista de todos los modelos acústicos. El primero de ellos no incluye el modelo sp, en cambio el otro si lo incorpora.

Finalmente es necesario obtener todos los vectores de características de todas las grabaciones de audio y especificar la lista de modelos acústicos para proceder a la reestimación con *HERest*. El proceso es el mismo que el realizado para la inicialización de los modelos, pero ahora hay que considerar todas las grabaciones.

Llegados a este punto ya se tiene todo lo necesario para empezar el entrenamiento de los modelos acústicos con *HERest*. Si anteriormente *HInit* exigía que los modelos fuesen inicializados de forma individual, *HERest* permite realizar el entrenamiento de forma conjunta. Para este proyecto, se empiezan haciendo hasta 6 reestimaciones de los modelos.

2.2.3 Inserción del modelo de pausa corta.

Hasta ahora se han creado HMMs de cinco estados con topología *left-to-right* para cada fonema y para el modelo de silencio sil. En este paso se añaden transiciones entre los estados 2-4 y 4-2 en el modelo de silencio sil y se genera un nuevo modelo de un único estado para pausas cortas, llamado sp.

Para la creación del modelo sp se copia el archivo *hmacs* de la última reestimación, éste se edita y se añade el nuevo modelo sp de tres estados, cuyo estado

central es el estado central del modelo sil. Después se ejecuta *HHEd* para añadir las transiciones adicionales y ligar el estado de sp con el estado central de sil.

Una vez ya se han creado el nuevo modelo y las nuevas transiciones, se procede a reestimar dos veces más los parámetros de los modelos acústicos mediante *HERest*.

2.2.4 Realineación de los datos de entrenamiento.

Observando el diccionario, se puede apreciar que hay palabras que tienen más de una pronunciación como es el caso de la palabra LA-S, que tiene dos posibles transcripciones. Se puede hacer un realineamiento de los datos de entrenamiento usando los modelos obtenidos hasta ahora y así crear nuevas transcripciones.

Esto se realiza con la herramienta *HVite* y se obtiene una nueva transcripción a nivel de fonemas en un fichero distinto. Ahora el reconocedor tendrá en cuenta todas las posibles pronunciaciones de las palabras del diccionario.

Para terminar con el proceso de creación de modelos de monophones se reestiman sus parámetros dos veces más mediante *HERest*.

2.3 Creación de los modelos de triphones con estados ligados.

Hasta ahora se han creado modelos acústicos que se corresponden con fonemas (llamados modelos de monophones). Estos modelos son independientes del contexto, es decir, no tienen en cuenta qué fonema ocurrió antes de uno dado ni cual vendrá después de éste. En esta fase se construyen modelos de triphones, conjuntos de tres fonemas, con el fin de aumentar la eficacia del reconocedor de voz.

2.3.1 Creación de triphones a partir de monophones.

Se pueden realizar triphones dependientes del contexto copiando los monophones y utilizar nuevas transcripciones con triphones para reestimar los parámetros de los modelos. Con *HLEd* obtendremos estas nuevas transcripciones. Tras la ejecución de *HLEd* se crean dos ficheros, uno contiene la lista de triphones generados y el otro las transcripciones a nivel de triphones.

También se requiere el fichero *mktri.hed*, que se puede generar con el script *maketrihed*, y que tiene la siguiente apariencia, y luego se procede a la copia de los modelos con *HHEd* almacenando los modelos editados en formato binario con el fin de mejorar la eficiencia.

```
CL triphones1
TI T_a {(*-a+*,a+*,*-a).transP}
TI T_b {(*-b+*,b+*,*-b).transP}
TI T_B {(*-B+*,B+*,*-B).transP}
TI T_d {(*-d+*,d+*,*-d).transP}
TI T_e {(*-e+*,e+*,*-e).transP}
TI T _f{{*-f+*,f+*,*-f}.transP}
TI T _g{{*-g+*,g+*,*-g}.transP}
...
```

Con estas instrucciones se han generado unos modelos de triphones a partir de los modelos de monophones. Se vuelve a reestimar estos modelos con *HERest* dos veces más.

2.3.2 Creación de triphones con estados ligados.

Momentáneamente se han creado modelos de triphones que comparten matrices de transición. El objetivo ahora es que también puedan compartir estados, y así poder realizar unas estimaciones más robustas de los parámetros. Esto se hace creando los llamados estados ligados. Una manera de crearlos es mediante los árboles de decisión.

Se genera parte del fichero que contiene el árbol de decisión con *mkclsscript*, el resto debe ser escrito manualmente, que utiliza *HHEd* para crear los estados ligados. Con esto se obtiene un fichero con la lista de todos los modelos acústicos resultantes, y los modelos de triphones con estados ligados y ya sólo falta realizar dos reestimaciones más con *HERest* para completar el entrenamiento de los modelos acústicos. Aquí se presenta un fragmento del árbol de decisión:

```
 RO 100.0 stats
TR 0
QS "L_a" {a-*"}
QS "R_a" {*+a"}
QS "L_b" {b-*"}
...
QS "L_T" {-*"}
QS "L_u" {-*"}
QS "R_u" {*+"}
QS "L_x" {-*"}
QS "R_x" {*+"}
TR 2
```

```
TB 350.0 "ST_a_2_" {("a","*-a+*","a+*","*-a").state[2]}
TB 350.0 "ST_b_2_" {("b","*-b+*","b+*","*-b").state[2]}
...
TB 350.0 "ST_u_4_" {("u","*-u+*","u+*","*-u").state[4]}
TB 350.0 "ST_x_4_" {("x","*-x+*","x+*","*-x").state[4]}
TR 1
CO "tiedlist"
ST "trees"
```

2.4 Adaptación de los modelos.

Los pasos anteriores describen las frases necesarias para construir un sencillo sistema de reconocimiento de voz orientado para aplicaciones concretas, como la domótica. Hasta el momento sólo se han utilizado grabaciones de un único hablante, sin embargo, esto puede producir malos resultados si el sistema va a ser empleado por otros hablantes. Se pueden construir sistemas independientes del hablante, pero se requieren grandes cantidades de grabaciones provenientes de muchas personas distintas. La alternativa empleada en este proyecto fue adaptar los modelos actuales a las características de nuevos hablantes, ya que esto no requiere grandes cantidades de datos. Normalmente las técnicas de adaptación se aplican a sistemas robustos e independientes del hablante para conseguir que se comporten mejor ante hablantes particulares.

HTK soporta dos tipos de adaptaciones: la adaptación supervisada, donde se conocen las transcripciones de los datos, y la adaptación no supervisada, donde las transcripciones son hipótesis. La herramienta *HERest*, proporcionada por HTK, permite realizar la adaptación supervisada en modo *offline*, mientras que la herramienta *HVite* realiza la adaptación no supervisada. Para este proyecto se empleó el método de la adaptación supervisada.

2.4.1 Preparación de los datos para la adaptación.

De igual modo que en el desarrollo de un reconocedor corriente, se necesita preparar los datos para la adaptación. Son necesarias grabaciones de los nuevos hablantes, unas para la adaptación y otras para la evaluación del nuevo sistema adaptado. Los datos se adquirieron del mismo modo que la obtención de los datos de entrenamiento y prueba del sistema. Se realizaron 20 grabaciones para el proceso de adaptación y se codificaron los datos igual que se hizo con los de entrenamiento.

Por último se generaron las transcripciones de los datos de adaptación a nivel de fonemas y a nivel de palabra, teniendo en cuenta que se trata de una gramática dependiente del contexto.

2.4.2 Generación de las transformadas MLLR.

HERest da soporte para un conjunto de transformadas lineales y permite realizar la adaptación supervisada en modo offline mediante la aplicación de transformadas lineales de máxima probabilidad.

Se pueden usar *árboles de clases* para especificar dinámicamente el número de transformaciones que hay generar, o bien utilizas un conjunto de clases base. Para este proyecto, con la ayuda de la herramienta *HHed* proporcionada por HTK, se construyó un árbol de clases de regresión utilizando los últimos modelos acústicos reestimados. Además se almacenaron en un conjunto de clases base con los comandos para cargar el fichero estadístico de ocupación de estados *stats* que se generó con la última ejecución de HERest, y construir un árbol de clases de regresión de 32 hojas usando estas estadísticas. También son necesarios los ficheros *config.global* y *config.rc*, con el siguiente contenido respectivamente.

```
HADAPT:TRANSKIND = MLLRMEAN
HADAPT:USEBIAS = TRUE
HADAPT:BASECLASS = global
HADAPT:ADAPTKIND = BASE
HADAPT:KEEPXFORMDISTINCT= TRUE
HADAPT:TRACE = 61
HMODEL:TRACE = 512

HADAPT:TRANSKIND = MLLRMEAN
HADAPT:USEBIAS = TRUE
HADAPT:REGTREE = rtree.tree
HADAPT:ADAPTKIND = BASE
HADAPT:SPLITHRESH = TRUE
HADAPT:KEEPXFORMDISTINCT= TRUE
HADAPT:TRACE = 61
HMODEL:TRACE = 512
```

A continuación se realiza la adaptación en dos fases con HERest. Primero se hace una adaptación global y después otra usando la anterior como transformación de entrada para el conjunto de modelos. Finalmente sólo falta realizar dos reestimaciones más con *HERest* para completar la adaptación de los modelos acústicos.

2.5 Evaluación del reconocedor.

Una vez concluido el proceso de entrenamiento del sistema, el reconocedor se ha completado y ya se puede evaluar su eficacia.

2.5.1 Reconocimiento de los datos de prueba.

Asumiendo que en un archivo se tiene la lista de los ficheros de vectores de características para la prueba del sistema, se procede al reconocimiento de cada uno de ellos con la herramienta *HVite*. El resultado obtenido, almacenado aparte, consta de un fichero con la transcripción de las frases que ha reconocido.

HTK proporciona otra herramienta llamada *HResults*, que genera información y estadísticas sobre la eficacia del reconocedor. Para ello emplea el fichero con la transcripción de los archivos de audio y éste se toma como referencia para compararlo con los resultados logrados con el reconocedor.

Se hicieron pruebas de decodificación con los mismos archivos de audio utilizados para el entrenamiento, obteniendo los siguientes resultados:

```
====================== HTK Results Analysis =======================
SENTENCES: %Correct = 98.67            [H=148,      S=2,      N=150]
WORDS:     %Correct = 99.62, Acc=99.62 [H=525, D=0, S=2, I=0, N=527]
```

2.5.2 Ejecución del reconocedor en vivo.

El reconocedor, además de funcionar en modo *offline*, puede ejecutarse en modo online, tomando como señal de entrada la generada por un dispositivo de entrada como puede ser un micrófono. Este es el modo de ejecución que interesa para este proyecto. Será necesario escribir otro fichero de configuración que contendrá los parámetros del anterior más los siguientes:

```
# Waveform capture
SOURCERATE=625.0
SOURCEKIND=HAUDIO
SOURCEFORMAT=HTK
AUDIOSIG=2
```

Ya sólo basta con ejecutar *HVite* sin especificarle archivos de entrada ni de salida. Antes de comenzar a hablar, hay que pulsar la combinación de teclas ctrl. + c.

Cuando se termine de pronunciar la frase, se vuelve a pulsar la misma combinación. Seguidamente se mostrará por pantalla lo que el sistema ha reconocido.

3. Desarrollo de la aplicación.

La aplicación gráfica creada simula una vivienda domótica que incorpora el reconocedor de voz obtenido para recibir las órdenes del usuario y mostrarlas en forma de animación gráfica por pantalla.

Para el desarrollo de la aplicación se ha tenido en cuenta el uso de herramientas gratuitas, lo que permite grandes ahorros en licencias y por lo tanto abaratamiento durante el desarrollo del producto final en caso de venta o implantación. El conjunto de herramientas de trabajo empleadas en este proyecto se componen de software HTK, API de desarrollo Julius, librería gráfica OpenGL y todo el conjunto de librerías necesarias para la gestión de ficheros, comunicación entre hilos, entre otras. En este proyecto se también se ha empleado la interfaz para desarrollo de aplicaciones llamada *Code::Blocks* ya que facilita la integración de librerías, especialmente las gráficas.

3.1 Adaptación de los modelos acústicos para su uso con Julius.

Antes de poder realizar acciones de reconocimiento con Julius y empezar el desarrollo de una aplicación de reconocimiento de voz, debemos preparar el sistema para ello incluyendo los modelos de lenguaje y modelos acústicos.

Julius necesita que la gramática y el diccionario estén escritos en un formato específico. Para ello, incorpora una serie de herramientas que permiten convertir la gramática y el diccionario escritos con HTK en el formato exigido por Julius. Al finalizar dicha conversión, todos los archivos temporales pueden ser eliminados, conservando únicamente aquellos que contienen la gramática y el diccionario en el formato de Julius.

Para que Julius pueda realizar acciones de reconocimiento, también necesita los ficheros que contienen los modelos acústicos entrenados con HTK y otro con un listado de todos los modelos. A diferencia de los anteriores, éstos no necesitan ser adaptados a otro formato.

De igual modo que HTK, Julius también necesita su propio fichero de configuración para poder realizar las acciones de reconocimiento y que se crea de forma manual. Para comprobar el funcionamiento y rendimiento de los modelos acústicos con Julius, se configura de modo que trabaje desde ficheros de audio tal y como se hizo anteriormente con HTK. Pero para trabajar directamente con la aplicación es necesario especificar el tipo de entrada externa con `mic` en el parámetro `input`. Por lo tanto el fichero de configuración para este proceso tendrá el siguiente contenido:

```
-dfa domotica.dfa
-v domotica.dict
-h hmacs
-hlist hmmlist
-penalty1 5.0
-penalty2 20.0
-iwcd1 max
-gprune safe
-b2 200
-sb 200.0
-iwsp
-iwsppenalty -70.0
-smpFreq 16000
-input mic
-walign
```

Se puede configurar el archivo anterior con muchas más opciones. Todas ellas las podemos encontrar en el manual de Julius.

Una vez finalizados con éxito todos los pasos anteriores, el Julius ya está listo para ejecutarse como herramienta de reconocimiento de voz desde el Terminal. Hay que tener en cuenta que la eficacia del reconocimiento depende en gran medida de la precisión de los modelos acústicos generados.

3.2 Creación de la aplicación de reconocimiento de voz con Julius.

Julius no es solamente una aplicación de reconocimiento de voz, también incluye su propia API de desarrollo programada en C.

Julius nos suministra un código de ejemplo, llamado *julius-simple*, utilizando las llamadas a las funciones de la API que incorpora. Julius-simple tan solo sirve de referencia para crear una aplicación. La aplicación de reconocimiento de voz de este proyecto se ha creado basándose en las llamadas a las funciones de la API que aparecen en el código de dicho ejemplo.

La idea principal de esta aplicación es recibir una entrada de audio y empleando los modelos acústicos y los algoritmos necesarios, programados e incluidos en la API de Julius, identificar la orden a ejecutar.

Antes de nada hay que indicar que tipo de información se suministra a la aplicación, esto se hace mediante la función *j_config_load_file_new()*, a la cual se le proporciona el fichero de configuración de Julius, que posee principalmente los parámetros para especificar si se van a realizar acciones de reconocimiento sobre archivos de audio almacenados en disco, o bien un reconocimiento en vivo a través de un micrófono y todos los ficheros necesarios para llevar a cabo el reconocimiento de las frases. Se hace tal distinción porque la forma de procesar los datos en cada uno de los casos es distinta. En lo referente a archivos de audio, el programa recorre la lista de archivos también nombrada en el fichero de configuración con la bandera *-filelist*.

Inicialmente la aplicación de reconocimiento de voz crea el canal de comunicación con el simulador y lo ejecuta para que éste pueda conectarse a dicho canal. A continuación se crean las instancias para el reconocimiento según sea el modo de ejecución del programa, indicado en el fichero de configuración, y se cargan todos los archivos necesarios para llevar acabo el reconocimiento. Seguidamente, se registran las funciones *callback* necesarias y se inicializa el dispositivo y la entrada de audio que se va a utilizar.

Llegados a este punto la aplicación ya está preparada para empezar el reconocimiento de voz. El procedimiento que sigue a dicho proceso se realiza con las funciones que incluye la API de Julius, así nuestra aplicación sólo ha de esperar en un bucle infinito a que el usuario mencione una acción. Esto es posible dado que julius incorpora un detector de silencio, de modo que el usuario, a diferencia de trabajar con HTK, no ha de hacer nada antes de pronunciar la frase que va a ser reconocida.

Luego, una vez ha sido procesada la información y se han obtenido los datos derivados del proceso de reconocimiento, se almacena un valor único en toda la aplicación que identifica cada una de las frases que admite el reconocedor. Dicho valor será enviado a la aplicación gráfica para reproducir la animación correspondiente.

En una ejecución normal de la aplicación, es decir, con el reconocimiento en vivo desde el micrófono, el programa no finaliza nunca. Si se desea terminar su ejecución el usuario debe forzarla. Sin embargo, cuando la aplicación trabaja desde ficheros de audio, la aplicación se cierra automáticamente al terminar el reconocimiento del último fichero cerrando la comunicación con el simulador y avisándole que también debe finalizar.

Las funciones de *callback* son llamadas automáticamente cuando se produce el evento asociado a ellas. Dichas funciones son: en la espera de una entrada de audio, al iniciar la entrada de audio y cuando ya se tiene el resultado del reconocimiento. La primera de ellas prepara el dispositivo de entrada de la señal acústica. La segunda recoge la señal y la prepara para su reconocimiento. La última de ellas realiza el reconocimiento en si y se comunica con el simulador para enviar el código correspondiente a la frase identificada.

3.3 Creación de la interfaz gráfica.

La interfaz gráfica desarrollada permite al usuario visualizar una vivienda tridimensional donde se reproducen en forma de animación gráfica las órdenes previamente identificadas por el reconocedor de voz.

Para la creación de dicha interfaz se parte del plano de una casa situándola en el cuadrante positivo del eje de coordenadas del mundo virtual generado. Con esto se simplifica la situación de objetos, luces y el punto de vista, eliminando cualquier posición con coordenadas negativas. La imagen 5.1 muestra la planta de la vivienda y las medidas empleadas para este proyecto.

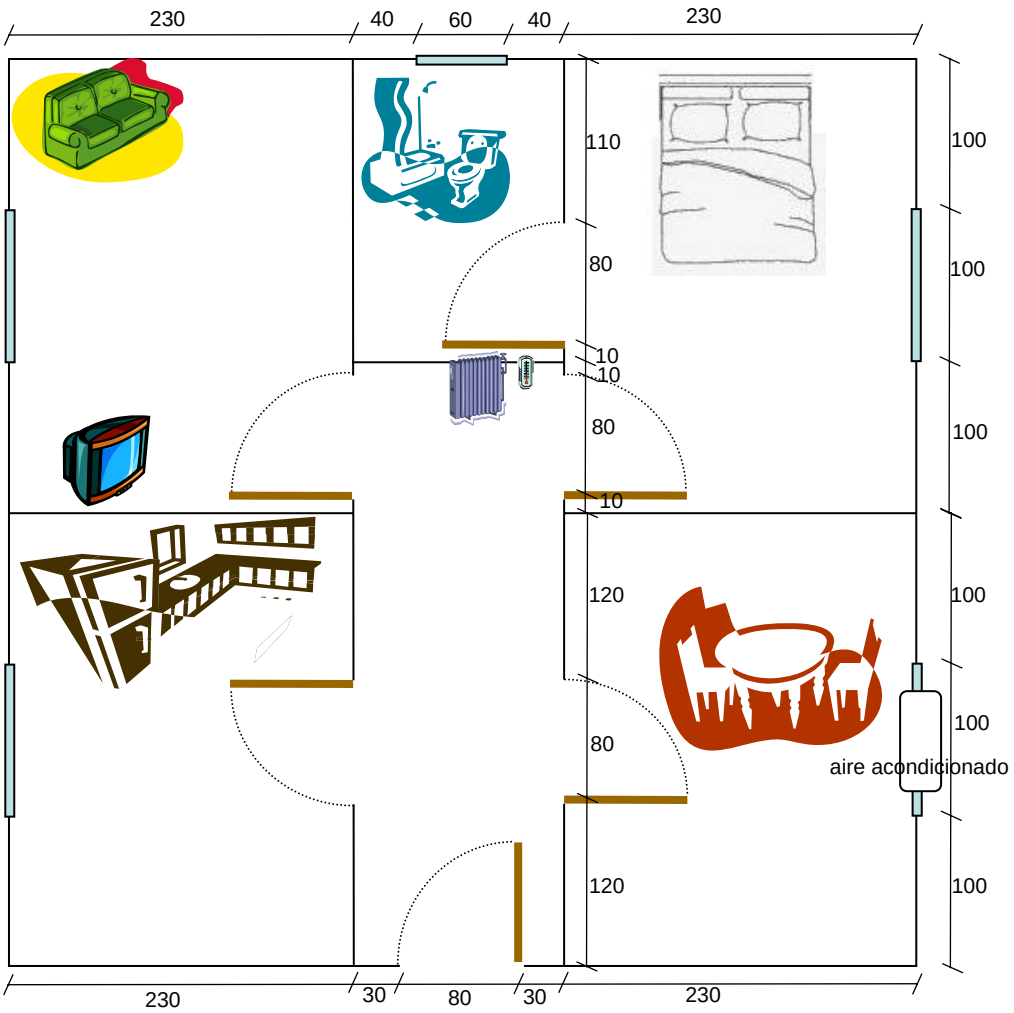

Figura 5.4: Mapa de la vivienda visualizada en el simulador.

Basándose en el plano anterior, la aplicación tiene un procedimiento dedicado para dibujar todo el conjunto de paredes, puertas, ventanas y muebles; asignándoles a todos ellos la textura más adecuada, las cuales han sido cargadas en memoria anteriormente con los parámetros correspondientes. Para un mayor control del dibujo de paredes, puertas y ventanas; éste se traza mediante polígonos simples, especificando a OpenGL las coordenadas de los vértices de dichos polígonos. Al mismo tiempo, en cada vértice se hace referencia a la coordenada de textura correspondiente para así generar un mosaico uniforme de la textura empleada en cada momento.

Para las puertas y ventanas, existe un vector que controla el estado en el que se encuentran, abiertas o cerradas. Inicialmente se encuadran todas cerradas. De igual modo, las luces inicialmente también se encuentran apagadas, excepto la luz ambiental necesaria para visualizar la escena y que no es accesible al usuario. En cambio, los

muebles que se visualizan, en general no sufren ninguna transformación, simplemente sirven para ubicar al usuario dentro de la casa y comprobar que también se ha identificado correctamente la habitación donde el usuario quiere que se realice su acción.

Una vez se visualiza toda la escena, desde el exterior de la casa, la cámara y el campo de vista se desplazan uniformemente hacia el interior de la vivienda y se detienen en el pasillo a la espera de recibir una orden para ser ejecutada.

De igual modo, cuando el simulador recibe una acción para ser ejecutada la cámara se dirige a una ubicación idónea para ver por pantalla la orden mencionada por el usuario. Si la posición y orientación de la cámara ya son buenas, no se realizará ningún desplazamiento. Para el control de la luces no es necesario ningún desplazamiento, puesto que se enciende o apaga el foco de luz de la sala en la que se encuentra el usuario.

Una vez está situado el punto de vista se procede a la reproducción de la correspondiente animación gráfica o cambio en la escena. En este proyecto los únicos objetos que pueden sufrir cambios son, las luces, puertas, ventanas, radiador, aparato de aire acondicionado y termostato. Al finalizar el cambio o animación gráfica, la aplicación se queda en espera de recibir una nueva acción.

En una ejecución normal de la aplicación, es decir, con el reconocimiento en vivo desde el micrófono, el programa no finaliza nunca. Si se desea terminar su ejecución el usuario debe forzarla. Sin embargo, cuando la aplicación trabaja desde ficheros de audio, la aplicación se cierra automáticamente al terminar el reconocimiento del último fichero.

La secuencia de ejecución del simulador cuando trabaja con reconocimiento en vivo consta de un bucle infinito y con una serie de funciones que se ejecutan según un temporizador. El siguiente esquema muestra una secuencia de ejecución simple del programa.

1. Comprobar que la aplicación recibe los parámetros correctos para la conexión con el proceso reconocedor (nombre del terminal y puerto). Sino, detener la aplicación.
2. Crear el socket y conectarlo con el proceso reconocedor. Si se produce algún error, detener la aplicación
3. Creación de la ventana de usuario.
4. Inicializar y cargar los valores predeterminados de la escena tales como iluminación, texturas y objetos tridimensionales.
5. Pintar la escena.
6. Iniciar el temporizador.
7. Cargar el trayecto adecuado para el movimiento inicial de la cámara.
8. Desplazar la cámara siguiendo el trayecto inicial.
9. Esperar la recepción de una nueva acción a reproducir. Si se produce algún error durante la transmisión de dicha información, ignorarlo.
10. Cargar el trayecto adecuado según la acción a reproducir y trasladar la cámara.
11. Reproducir la acción recibida.
12. Activar de nuevo el temporizador.
13. Volver al punto **9**.
99. Cuando finaliza el proceso reconocedor, cerrar socket y finalizar la aplicación.

3.3.1 Imágenes y animaciones.

Como se ha mencionado anteriormente, la interfaz gráfica de usuario consta de un mundo virtual donde se simula una vivienda domótica. Al tratarse de una aplicación gráfica, el usuario ve por pantalla un entorno virtual de una vivienda. Como las posibles acciones que se pueden realizar son recibidas en formato acústico mediante un micrófono, la aplicación no tiene ningún formulario ni menú donde el usuario pudiese interactuar con la aplicación.

De este modo para que la aplicación sea más real, visual, dinámica e intuitiva, se han empleado variedad de imágenes tridimensionales y bidimensionales (o texturas) además de las animaciones gráficas que se reproducen automáticamente cuando el usuario formula una frase.

En cuanto a las imágenes tridimensionales, se han empleado ficheros con extensión *.3ds*, ficheros generados con la aplicación *3D Studio Max* y que representan objetos complejos, con información de los vértices, caras, texturas, materiales, luces, cámaras y punto de vista entre otros. La estructura resumida de estos ficheros es la siguiente:

```
MAIN CHUNK 0x4D4D
   3D EDITOR CHUNK 0x3D3D
      OBJECT BLOCK 0x4000
         TRIANGULAR MESH 0x4100
            VERTICES LIST 0x4110
            FACES DESCRIPTION 0x4120
               FACES MATERIAL 0x4130
            MAPPING COORDINATES LIST 0x4140
```

```
         SMOOTHING GROUP LIST 0x4150
      LOCAL COORDINATES SYSTEM 0x4160
   LIGHT 0x4600
      SPOTLIGHT 0x4610
   CAMERA 0x4700
MATERIAL BLOCK 0xAFFF
   MATERIAL NAME 0xA000
   AMBIENT COLOR 0xA010
   DIFFUSE COLOR 0xA020
   SPECULAR COLOR 0xA030
   TEXTURE MAP 1 0xA200
   BUMP MAP 0xA230
   REFLECTION MAP 0xA220
   [SUB CHUNKS FOR EACH MAP]
      MAPPING FILENAME 0xA300
      MAPPING PARAMETERS 0xA351
KEYFRAMER CHUNK 0xB000
   MESH INFORMATION BLOCK 0xB002
   SPOT LIGHT INFORMATION BLOCK 0xB007
   FRAMES (START AND END) 0xB008
      OBJECT NAME 0xB010
      OBJECT PIVOT POINT 0xB013
      POSITION TRACK 0xB020
      ROTATION TRACK 0xB021
      SCALE TRACK 0xB022
      HIERARCHY POSITION 0xB030
```

Dado que estos ficheros tienen una estructura compleja, se ha utilizado la API de desarrollo de la librería *lib3ds*, de código abierto y programada en C, para leer los objetos tridimensionales y presentarlos por pantalla.

A continuación se listan todos los objetos que se visualizan en la aplicación gráfica:

. Mesa de comedor y sillas.

. Cama.

· Cocina.

· Cuarto de baño.

· Sofá.

· Televisor.

· Lámparas.

· Radiador.

· Aparato de aire acondicionado.

· Termostato.

Sin embargo, las distintas acciones que permite la aplicación gráfica se reciben a través del micrófono y una vez identificadas son visualizadas en forma de animación gráfica tridimensional para que el usuario vea que realmente el sistema entendió lo que

él dijo. Las posibles animaciones que se visualizan son apertura y cierre de puestas y ventanas, las cuales se generan en el momento de recibir la orden incrementando o recudiendo el ángulo de apertura progresivamente y empleando las funciones que proporciona OpenGL para aplicar transformaciones geométricas.

Asimismo, antes de reproducir una animación o realizar algún cambio en la escena, la cámara y el punto de vista se trasladan, des del punto donde se había reproducido la acción anterior, hasta la nueva ubicación donde se podrá visualizar correctamente la nueva orden. De igual modo, al iniciar la aplicación también se desplaza la cámara hacia el interior de la vivienda y se queda en el pasillo a la espera de recibir una acción.

El procedimiento que se encarga de los trayectos, para generar los caminos que realiza la cámara, se basa en el cálculo de *splines*. Éste tiene preestablecidos los puntos de control y el parámetro de la curva, solo debe cargarlos en memoria antes de empezar. Finalmente, calculando la derivada de cada una de las posiciones por las que se desplaza la cámara, se obtiene el punto de vista correspondiente en cada instante.

Si la posición y orientación de la cámara ya son buenas, no se realizará ningún desplazamiento. Para el control de la luces no es necesario ningún desplazamiento, puesto que se enciende o apaga el foco de luz de la sala en la que se encuentra el usuario.

Las luces se controlan con las funciones que incorpora OpenGL para ello. Hay que tener en cuenta que los focos de luz son fijos y están definidos en la aplicación con unos parámetros adecuados para una buena visualización. El único cambio que sufrirán las luces durante la ejecución del programa es el hecho de estar activadas o desactivadas según las acciones que emita el usuario.

El radiador y el aparato de aire acondicionado son objetos tridimensionales que no tiene asignadas ninguna textura. Estos cambian su aspecto en función del estado en que se encuentran, ya sea apagado o encendido, y con ayuda de las herramientas que proporciona OpenGL es sencillo hacer que cambien de color para que sea más intuitivo para el usuario. Sin embargo, los cambios en el termostato se realizan cambiando la

textura vinculada al objeto y así ver con mayor claridad que la temperatura fijada es la mencionada por el usuario.

Finalmente, cuando una frase no está incluida en la gramática, el simulador recibe un valor que identifica como tal y el cual muestra un mensaje informativo por pantalla para que el usuario sea consciente de lo ocurrido y pueda repetir la orden o mencionar una frase permitida.

4. Conexiones entre hilos.

La aplicación, como se ha mencionado anteriormente consta de dos partes bien diferenciadas, el reconocedor de voz y el simulador de domótica. Ambas se ejecutan en dos procesos independientes el uno del otro. El primer proceso o proceso padre es el que incorpora el reconocedor de voz, y el proceso hijo es el simulador.

Para interconectar ambos procesos se ha utilizado una conexión por *sockets* donde sólo se envía información en un único sentido, servidor-cliente. Cuando el proceso padre recibe una entrada de audio procede a su interpretación. Tanto si se trata de una frase reconocida como si no, el proceso padre envía un código único para cada situación y frase válida al proceso hijo a través del *socket*. El proceso hijo espera a recibir la orden que le envía el proceso padre a través del *socket*, y ejecuta la animación gráfica correspondiente al valor recibido, sin enviar confirmación ni cualquier otro tipo de mensaje al proceso padre.

5. Control de errores.

Como en toda aplicación informática, ésta también puede generar errores y por lo tanto deben ser tratados como tales y actuar en consecuencia para que no interfieran en la ejecución normal de la aplicación.

Los principales errores que pueden darse en la aplicación sólo pueden ser debidos a no reconocer correctamente una de las acciones mencionadas por el usuario.

Entonces diferenciamos dos posibles casos: que el reconocedor de voz no entienda la frase como una acción válida, y que el reconocedor interprete una frase distinta a la formulada por el usuario.

En el primer caso, el reconocedor de voz no entiende la frase como una acción válida. En este case es el mismo reconocedor de voz el que recoge el error y lo gestiona como si se tratara de una frase correcta. Para que el usuario sea consciente del error producido, la aplicación gráfica, al descifrar el valor recibido, interpreta correctamente el error que se produjo durante el proceso de reconocimiento; y en lugar de ejecutar una animación muestra un mensaje de texto al usuario informándole que la frase mencionada no corresponde con ninguna de las acciones permitidas.

En cambio, la segunda situación es un tanto peculiar, puesto que el reconocedor sí que interpretó la frase, pero en este caso la interpretación no es correcta. Es un error muy difícil de tratar ya que durante una ejecución del reconocedor en vivo no tenemos ningún método para comparar el resultado obtenido por el mismo con el resultado que debiera ser; como sí ocurre en el caso de las grabaciones, donde disponemos de un fichero de texto con las transcripciones de cada una de las grabaciones. Puesto que la aplicación se ejecuta mayoritariamente en vivo, ésta no interpreta la situación como un error y por lo tanto la aplicación gráfica recibe el código de la frase reconocida (aunque no sea la correcta) y ejecuta la animación correspondiente a ésta.

También existen otros tipos de errores que puede detectar la aplicación como el caso de no poder leer un fichero, bien de textura u objeto 3ds, la conexión de los *sockets* y la transmisión de datos a través de ellos. Todos ellos son controlados por la aplicación e informan al usuario en caso de producirse.

La lectura de ficheros, sean del tipo que sean, no suele generar errores a no ser que el fichero esté corrupto, el formato no corresponda al adecuado o no se encuentre en la ruta especificada. Tales errores no pueden ser corregidos desde la aplicación una vez se está ejecutando. Para no interrumpir la aplicación en caso de producirse un error, cuando se trata de un fichero de imagen o textura, directamente se omite dicho fichero y se pasa al siguiente. En cambio, si el error se produce al leer el fichero de configuración

de Julius, se comunica tal error al usuario y la aplicación se detiene puesto que no se podrá llevar a cabo el reconocimiento de las distintas frases.

En cuanto a los *sockets*, si el error se produce durante la conexión, se informa al usuario del problema ocurrido y se detiene la aplicación, ya que no se podrán recibir las acciones a reproducir en la interfaz gráfica. Tal error es solucionado automáticamente por el sistema operativo transcurrido un cierto tiempo, por lo que el usuario podrá reejecutar la aplicación de nuevo y ésta funcionará perfectamente. Sin embargo, cuando el error se produce al escribir datos en el *socket*, se notifica del problema al usuario y la aplicación seguirá su ejecución. Pero, en este caso la interfaz gráfica no recibirá ningún dato y no se ejecutará la orden, por lo que el usuario deberá mencionar la frase de nuevo.

6. Juego de pruebas.

Las primeras pruebas necesarias fueron la evaluación de los modelos acústicos obtenidos y el funcionamiento del reconocedor en vivo. Para una primera evaluación se emplearon nuevas grabaciones de voz de los hablantes con los que fue entrenado el sistema. En cambio, el reconocimiento de voz en vivo fue testeado con cinco nuevos hablantes para comprobar su robustez. Todos pronunciaron una media de 25 frases aceptadas y 5 no reconocidas. Con todos ellos se obtuvieron resultados muy satisfactorios ya que en el 90% de las ocasiones el reconocedor de voz entendía la acción mencionada.

Posteriormente se llevaron a cabo una serie de pruebas relacionadas con la interfaz gráfica para comprobar que todas las acciones reproducidas las realizaba correctamente. Para simplificar un poco la complejidad del simulador, solo se han incorporado al simulador 4 habitaciones de las 6 que admite el reconocedor. Aun así, las dos no representadas en el simulador se han considerado sinónimos de otras que si están. Llegados a este punto se testearon todas las acciones que permite el reconocedor (teniendo en cuanta la gramática del reconocedor y la simplificación hay 30 acciones

distintas), comprobando que realmente se identificaba la frase con la acción reproducida.

Otro juego de pruebas que se hizo relacionado con la aplicación gráfica fue comprobar que todos los trayectos que realizaba la cámara llevaban el punto de vista al punto adecuado y de forma uniforme desde la posición anterior. Igual que ocurría anteriormente con la reproducción de las acciones, el número de posibles trayectos también era muy elevado (del orden de 400) así que también se optó por simplificarlos. Inicialmente se agruparon las acciones complementarias, como por ejemplo apagar y encender las luces, reduciendo a más de la mitad el número de trayectos. Aun así, seguía siendo bastante elevado y se simplificó un poco más aprovechando en un mismo punto de vista el radiador y el termostato, la venta del comedor y el aire acondicionado, y el pasillo inicial y los controles de las puertas. Con esto se consiguió simplificar de aproximadamente 150 trayectorias a tan sólo 31, pero que hubo que comprobar.

Capítulo 6

CONCLUSIONES

1. Conclusiones.

A lo largo de este proyecto se han ido consiguiendo los diferentes objetivos que se habían marcado al principio. También se han adquirido nuevos conocimientos acerca del reconocimiento del habla por computador y su integración con otras aplicaciones, además de aprender otros métodos de programación gráfica.

Después de elegir la herramienta de trabajo HTK, se han a construido y entrenado unos modelos acústicos para las unidades fonéticas que componen nuestra gramática. Ha surgido la dificultad del número de hablantes y de grabaciones necesarias para generar modelos robustos, por lo que finalmente se ha optado por un reconocedor más limitado, es decir, para los usuarios con los que no se han entrenado los modelos acústicos la tasa de aciertos es inferior.

Tras conseguir un reconocedor suficientemente robusto para las necesidades de este proyecto, se prosiguió a la creación de la aplicación incorporando el reconocedor ya creado y desarrollando desde cero toda una interfaz gráfica tridimensional que simula una vivienda domótica. Cabe destacar que la única interacción del usuario con la aplicación es mediante comandos de voz, el resto se gestiona automáticamente desde el propio programa. A continuación se han generado todas las acciones que recibe la aplicación des de el reconocedor.

Finalmente, se han realizado gran variedad de pruebas para comprobar que el funcionamiento de todo el sistema es correcto y no se producen fallos en ningún punto de la ejecución. Desgraciadamente, se ha detectado algún fallo durante el proceso de reconocimiento de las frases; especialmente si el usuario no es uno de los hablantes con

los que se entrenó el sistema. A pesar de ello, se puede afirmar que se han conseguido los objetivos propuestos al principio y que los resultados obtenidos han sido satisfactorios.

2. Posibles mejoras.

Como en la mayoría de proyecto, a éste también se pueden añadir mejoras y ampliaciones. En un sistema domótico real existen muchas más acciones posibles a realizar de las contempladas en este proyecto. Además, puede ser utilizado por cualquier persona aunque el sistema no haya sido entrenado con su voz. Con el fin de aumentar la eficacia y aplicabilidad del sistema, podrían llevarse a cabo varias mejoras.

En cuanto a reconocimiento del habla una de estas mejoras, sería ampliar el lenguaje que acepta el reconocedor de voz y con ello todas las animaciones gráficas y cambios en la escena, además de mejorar la tasa de aciertos para todos los usuarios habituales y esporádicos entrenando los modelos acústicos con grabaciones de muchos más hablantes y más variados. También se podría utilizar otra tecnología distinta de reconocimiento de voz e implementar la identificación del hablante. Con la detección del orador podemos añadir un filtro de manera que si el hablante no es uno de los autorizados no se realice la acción.

Otra mejora para el simulador de domótica sería añadir más efectos a las animaciones que ya hay. Por ejemplo un efecto de onda de aire caliente encima de los radiadores; cintas que se mueven en función de la potencia del aire acondicionado; o cambio gradual de los dígitos del termostato. Incluso se puede mejorar la interacción entre el usuario y la aplicación añadiendo la posibilidad de explorar la casa libremente manejando el ratón o el teclado, y así prescindir de los desplazamientos automáticos de la cámara. Asimismo se podría aumentar el tamaño de la casa y añadirle más habitaciones, e incluso espacios exteriores donde también actuaría la aplicación.

BIBLIOGRAFIA

Alhena. "*Domótica x10 por voz*". Web de la empresa Alhena, distribuidor de productos domóticos. http://www.alhenaing.com/index.php
Último acceso el 11 noviembre de 2009.

Bernal Bermúdez, Jesús; Bobadilla Sancho, Jesús; Gómez Vilda, Pedro. "*Reconocimiento de voz y fonética acústica*". Ra-ma, 2000.

Bristol, Greoff. "*Electronic speech recognition. Techniques, technology and applications*". Collins, 1986. First edition.

Casacuberta Nolla, Francisco; Vidal Ruiz, Enrique. "*Reconocimiento automático del habla*". Marcombo. Boixareu editores, 1987.

Code::Blocks Team. "*Code::Blocks*". Web oficial del programa Code::Blocks.
http://www.codeblocks.org/
Último acceso el 11 noviembre de 2009.

Colás Pasamontes, José. Web del proyecto de un sistema de comprensión del habla.
http://elies.rediris.es/elies12/cap241.htm
Último acceso el 11 noviembre de 2009.

CUED and Entropic. "*HTK3*" Web oficial de HTK. http://htk.eng.cam.ac.uk/
Último acceso el 11 noviembre de 2009.

Hidobro Moya, Jose Manuel; Millán Tejedor, Ramón Jesús. "*Domótica. Edificios inteligentes*". Creaciones copyright, 2004.

Jelinek, Frederick. "*Statistical methods for speech recognition*". Cambridge, Mass. MIT Press, 1998.

Julius development team. "*Open-Source Large Vocabulary CSR Engine Julius*". Web oficial de Julius. http://julius.sourceforge.jp/en_index.php
Último acceso el 11 noviembre de 2009.

Julius development team. "*JuliusLib API* ". Web oficial de la API de Julius. http://www.sp.nitech.ac.jp/~ri/julius-dev/doxygen/julius/4.0/en/group__jfunc.html
Último acceso el 11 noviembre de 2009.

Keller, Eric. "*Fundamentals of speech systhesis and speech recognition. Basic concepts, state-of-the-art and future challenges*". Jhon Wyley & Sons, 1994.

Kyprianidis, Jan Eric. "*Lib3ds 2.0*". Web oficial de la librería lib3ds.
http://www.lib3ds.org/
Último acceso el 11 noviembre de 2009.

Lin, Dekang. "*Intelligent Systems: Introduction to Artificial Intelligence*". Web de un curso de inteligencia artificial, incluye reconocimiento del habla con modelos ocultos de Harkov. http://www.cs.ualberta.ca/~lindek/366/index.htm
Último acceso el 11 noviembre de 2009.

Llamas Bello, César; Cardeñoso Payo, Valentín. "*Reconocimiento automático del habla. Técnicas y aplicaciones.*" Serie: Ciencias nº16. Secretario de publicaciones e intercambio científico. Universidad de Valladolid, 1997. 3ª edición.

Lleida Solano, Eduardo. "*Tecnologías del habla*". Web dedicada a explicar los sistemas tecnológicos relacionados con el habla.
http://physionet.cps.unizar.es/~eduardo/investigacion/voz/voz.html
Último acceso el 11 noviembre de 2009.

M.A Spaans. "*On Developing Acoustic Models Using HTK* ". Delft University of Technology, 2004.

Pérez Pavón, Elia Patricia. "*Construcción de un reconocedor de voz utilizando Sphinx y el corpus DIMEx100*". Universidad Nacional Autónoma de México, Facultad de ingeniería, 2006.

Proinssa S.L.L. "*Domótica para la tercera edad y personas con discapacidad. Control por voz del entorno*". Web de la Proinssa, especializada en sistemas domóticos. http://www.proinssa.com
Último acceso el 11 noviembre de 2009.

Rabiner, Lawrence R. "*Fundamentals of speech recognition*". Prentice Hall, 1993.

Santos-García, Gustavo. "*Inteligencia artificial y matemática aplicada: Reconocimiento automático del habla*" Serie: Ciencias nº19. Secretario de publicaciones e intercambio editorial. Universidad de Valladolid, 2001. 2ª edición.

Vitulli, Damiano. "*Tutorial 4: 3ds loader*". Tutorial sobre la estructura y manejo de ficheros 3ds. http://www.spacesimulator.net/tut4_3dsloader.html
Último acceso el 11 noviembre de 2009.

Firmado**: Héctor Delgado Flores**

Bellaterra, 12 de junio de 2008.

AGRADECIMIENTOS

Este proyecto pone punto y final a mi etapa universitaria. Han sido unos años en los que he disfrutado de esta carrera, de muchas de sus asignaturas y del ambiente de esta universidad. Hoy comienza el recuerdo de unos años en lo que he conocido a muchas personas, con los que sin duda las clases y laboratorios no hubieran sido lo mismo.

Quiero comenzar dando mi más sincero agradecimiento a Javier Serrano por haberme dirigido este proyecto final de carrera, por haber confiado en que podía hacerlo, por todas sus atenciones, por todo el tiempo que me ha dedicado y especialmente por su apoyo.

También me gustaría destacar la paciencia que ha tenido conmigo Enric Martí Y Hector Flores, sufridores constante de mis preguntas para la resolución de los distintos problemas surgidos durante el desarrollo de todo el proyecto.

En general, a todos los profesores de la EUIS que de una u otra manara me han ayudado a realizar este trabajo, que aunque no les mencione de forma explicita, no les puedo negar un sincero agradecimiento.

Finalmente, no puedo dejar de agradecer la comprensión de mis familiares y amigos, puesto que ellos han sido quienes más han tenido que aguantarme en mis malos momentos a lo largo de la elaboración de este proyecto y que me ayudaron a encontrar el camino cuando pensé que estaba perdida.

A todos, muchas gracias.