UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniería Informática

Proyecto Fin de Carrera

Localización estereofónica en robots móviles.

Autor: Luis Vicente Calderita Estévez.

Fdo:

Director: Pablo Bustos García del Castro.

Fdo:

Tribunal Calificador

Presidente:Moreno del Pozo, José
Fdo:

Secretaria: Bachiller Burgos, Pilar
Fdo:

Vocal: Nuñez Trujillo, Pedro
Fdo:

CALIFICACIÓN:
FECHA:

Agradecimientos.

A mi familia por su paciencia y confianza.

A Perico por salvarme tantas veces el culo.

Al equipo de Robolab, y en especial a Pablo Bustos, por demostrarme que al final, lo estudiado, servía para crear algo y no solo para reproducirlo fielmente en un examen.

La vida es un continuo IF
en la que nunca podemos probar
la solución del ELSE.

Luiky.

Índice general

Capítulo 1

Introducción.

El Laboratorio de Robótica y visión artificial de la Universidad de Extremadura (Robolab) lleva varios años dedicado al estudio de la visión y ha realizado trabajos muy interesantes en estos campos. Junto con la movilidad del robot ha conseguido la navegación de manera autónoma esquivando objetos a su paso..., ha conseguido reconocer elementos de su entorno, enfocar, acercarse, seguir trayectorias, en definitiva, a ojos de un observador ajeno, la familia RobEx parece estar dotada de un sentido visual primigenio.

La cuestión de: ¿por qué no oye si ya ve?, invadió mi cabeza al poco tiempo de conocer las funcionalidades anteriores, la percepción del entorno de los humanos y su ubicación relativa en el espacio tiene un alto componente auditivo, si bien el ojo nos puede dar la impresión de ser un sentido dominante e incluso suficiente, el oído nos ayuda en esa localización.

El ideal era conseguir, a largo plazo, una capacidad de atención por parte del robot, conseguir que al llamarlo, por ejemplo, nos mirara. Por supuesto el listón estaba muy alto y no sería alcanzable para un proyecto final de carrera, pero sí se podrían sentar las bases de ese futuro: empezar a oír y empezar a orientarse.

Entonces había que comenzar construyendo las herramientas que nos permitieran obtener el sonido ambiente y luego procesarlo de alguna manera. Como antaño los desarrolladores de Robolab escribieron el código que permitía capturar imágenes y tomar algunas decisiones. Necesitábamos sentar las tecnologías que

permitan la recepción del sonido y su interpretación, centrándose en una primera fase en la localización acústica de una fuente sonora. Este proyecto fin de carrera debe dotar de nuevas funcionalidades a RobEx, no solo para el fin propio, sino para los futuros trabajos relacionados con la capacidad auditiva. Por lo tanto debemos explorar y estudiar el comportamiento humano en la recepción y extracción de información a partir del sonido y adaptarlo a las tecnologías disponibles con el fin de conseguir otro sentido primitivo, el oído.

Disponemos de una base robótica móvil controlada por un portátil, el cual conecta vía TCP/IP con un PC, en general el portátil captura la señales de los dispositivos, láser, cámaras y ahora micrófonos, y los transmite al PC, el cual monitoriza y procesa esa información. El sistema de audición, originalmente, estará compuesto por dos cámaras USB con micrófonos integrados colocados en una torreta estéreo, montada a su vez sobre la base robótica. En la figura se puede ver el robot utilizado.

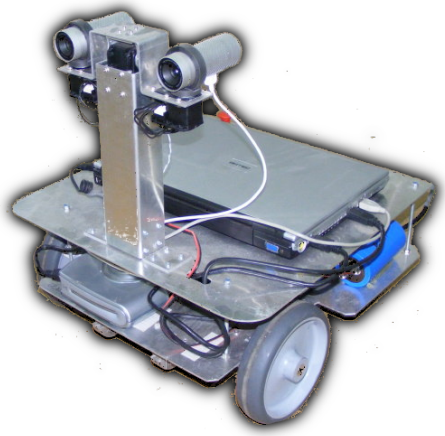

Ilustración 1: RobEx con cámaras USB

Posteriormente el sistema será provisto de un dispositivo digitalizador de audio USB, encargado de transformar la señal proporcionada por dos micrófonos omnidireccionales de mejor calidad.

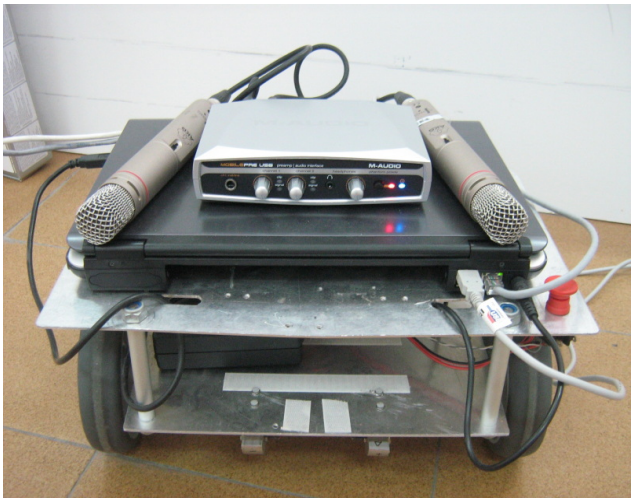

Ilustración 2: Robex equipado con la hardware de digitalización.

A pesar de que la implementación y las pruebas han sido llevadas a cabo sobre el robot RobEx usando la arquitectura de componentes software RoboComp, los conceptos y algoritmos son genéricos y se pueden usar en cualquier otra plataforma robótica que disponga de un par de micrófonos.

RobEx es una base robótica cuyo diseño se distribuye como hardware libre bajo una licencia de tipo Creative Commons Share-Alike. La plataforma de software robótico RoboComp se distribuye bajo la licencia GPL. Así pues, ambos proyectos, desarollados en el Laboratorio de Robotica de la Universidad de Extremadura. Son libres y se puede acceder a ellos mediante sus correspondientes páginas web [12], [13]. Dado que los resultados del proyecto de fin de carrera se han incluido dentro de RoboComp, el software desarrollado en él se distribuye bajo la misma licencia.

Motivación.

Una vez resueltos los problemas del control de la mecánica del robot, la locomoción, el control teledirigido, la detección de obstáculos, el reconocimiento de objetos e incluso la navegación autónoma, era hora de empezar a oír, bien para

complementar alguna de las anteriores, bien para iniciar nuevos caminos de investigación.

De la capacidad de oír dependerán en años venideros aplicaciones para interactuar con otros sistemas, algunas de ellas serán:

- Capacidad de atención.
- Orientación auditiva-ocular.
- Localización de fuentes sonoras.
- Reconocimiento de voz.
- Interpretación del lenguaje humano.

Las técnicas que intentan interpretar el sonido como fuente de información, actualmente caminan en la dirección de los reconocedores de voz, buscando la forma de operar con ella los distintos dispositivos que nos rodean, ordenadores, móviles, entornos ofimáticos. En este sentido hay muchos frentes abiertos, mucha gente y muchos recursos, con lo cuál se espera que en los próximos años disfrutemos de estos avances, sin embargo, la percepción del sonido entendido como un elemento revelador en la ubicación relativa de los objetos con relación al entorno de un oyente, no esta muy estudiada, lo cuál lo hacía muy interesante.

Se pretendía, simular el sistema auditivo humano físicamente, es decir, utilizar dos micrófonos a modo de oídos, separados una distancia determinada. Los trabajos que hablan de la localización acústica suelen poseer un array de micrófonos, que normalmente consiste en varios (4 , 6 u 8) colocados de forma equiespaciada, incluso alrededor del sujeto separados en segmentos de 15 a 30 grados. De esta manera y con distintas técnicas se pretende estimar la dirección del sonido.

La técnicas biaurales, propiamente dichas, suelen tener aplicaciones más comerciales y para otros fines lejanos al objeto de este trabajo, aunque interesantes desde el punto de vista conceptual. Se utilizan en métodos de grabación holofónica.

Básicamente, consiste en grabar música o bandas sonoras de manera que simulen la forma natural en la que lo escuchamos. Se graba con un par de micrófonos en la posición de las orejas, normalmente con maniquíes especialmente concebidos para este fin, y posteriormente, al reproducirlos tenemos una sensación de relieve espacial. Así oiremos, por ejemplo, cómo se cierra una puerta detrás nuestra, o cómo hablan a nuestra derecha.

Objetivo.

El objetivo se define de manera sencilla: localizar una fuente sonora, pero veremos a lo largo de este texto como esta capacidad común a todos los animales, y aparentemente sencilla, no es fácil de reproducir en entornos robóticos.

Se pretende conseguir determinar desde dónde proviene el sonido con el aliciente añadido de utilizar solo un par de micrófonos para plantear un esquema, salvando las distancias, lo más humano posible.

Para llevar a cabo esta tarea podíamos dividir en tres partes técnicas este proyecto.

- La primera consistiría en dotar de la capacidad auditiva a RobEx. Obtener el audio de los dispositivos y enviarlo al PC remoto, es decir, para empezar hay que construir un “streaming” de audio, lo más compatible con la amplía gama de elementos sonoros que hay en el mercado y con una buena calidad del audio. Esta tarea que hace pocos años podía ser considerada un proyecto fin de carrera en si misma, es ahora el primer peldaño de otro.

- La segunda consistiría en la monitorización de las señales recibidas, dibujando la señal en el dominio del tiempo y en el de la frecuencia, así como su reproducción continua e incluso grabación convencional con el objeto de afirmar que el proceso auditivo es correcto.

- Una vez conseguidos estos objetivos, nos centraremos en aplicar las técnicas de localización acústica estudiadas, las propias y las inspiradas, tratando de discernir entre la más correcta mediante pruebas, descartando otras y mezclándolas, hasta alcanzar el mayor grado de robustez posible.

Capítulo 2.

Robex y RoboComp.

El entorno en el que desarrollaremos el proyecto queda caracterizado por el robot sobre el que será integrado (familia RobEx) y por el sistema de componentes (RoboComp).

El robot en el que se llevarán a cabo las pruebas es un robot de la serie RobEx. Estos son robots móviles con conectividad wireless y/o Ethernet, disponen de un ordenador de a bordo en el que se ejecutan algunos componentes y desde el que pueden interactuar con otros componentes ejecutados de forma remota.

El sistema de componentes llamado RoboComp podemos definirlo como un grafo de componentes o procesos que pueden ser distribuidos en varios procesadores e interactuar entre si.

El proyecto consistirá en la creación de los componentes necesarios para llevar a cabo los objetivos planteados anteriormente, estos no tienen tienen dependencia estricta del resto, aunque están basados en los antiguos adquiriendo de ellos ideas, conceptos y metodología de desarrollo. Al mismo tiempo se han utilizado componentes fabricados por otros integrantes del grupo para las pruebas, como apoyo en ciertas tareas y para la depuración del código final. Por la razón fundamental de la reutilización en la programación orientada a componentes están pensados con la esperanza de que en un futuro no muy lejano, otros los utilizarán para incorporar nuevas características al robot.

A lo largo de este capítulo se describirá el entorno de desarrollo y herramientas usadas, se hará una breve descripción del robot RobEx, se introducirá el paradigma de la programación orientada a componentes y, finalmente, se hará una breve descripción del repositorio de componentes RoboComp.

RobEx.

El robot RobEx [12] , [14] es de tipo diferencial, esto es, el movimiento del robot depende únicamente de la velocidad de rotación de sus dos ruedas motrices. Además de las ruedas motrices, el robot dispone de una tercera rueda, de giro libre, que sirve de apoyo. La simplicidad de diseño y de los cálculos asociados al modelo diferencial son las principales razones que han definido el diseño, no sólo de RobEx, sino de otros muchos robots y gamas de ellos, como pueden ser: Segway, Kephera, o Roomba.

RobEx es una base robótica libre desarrollada en el Laboratorio de Robótica y Visión Artificial de la UEx, Robolab. Sus planos se distribuyen bajo una licencia Creative Commons de tipo Share-Alike. Por tanto, su diseño está abierto a cualquiera que lo quiera consultar [12] o contribuir a él. Al ser de tipo diferencial, su construcción es relativamente sencilla, si se tienen conocimientos de electrónica, cualquiera la puede construir. RobEx está diseñado para llevar uno o varios ordenadores portátiles a bordo para realizar procesos complejos que sean necesarios, hasta hoy relacionados con la visión o la inteligencia artificial, y desde hoy, con la audición.

Tanto los motores de la base, de corriente continua, como el resto de la electrónica, se alimentan de una batería del tipo que se suele usar para extender la autonomía de los ordenadores portátiles. La base está controlada por un microcontrolador dedicado al que se accede a través de una interfaz RS232 sobre USB. [14], [4].

Su principal orientación es la investigación y la docencia. Aunque cada vez resultan más versátiles y pronto darán el salto a la realización de trabajos concretos en empresas. Llevan utilizándose más de tres años a diario en las asignaturas de Robótica y Teoría de Sistemas que se imparten en la carrera de Ingeniería en Informática de la UEX. El origen de su diseño y desarrollo hasta su estado actual nació de los diferentes proyectos de investigación realizados en Robolab desde su creación en el año 1999. Desde su primera versión, cada mejora y nueva funcionalidad que incorpora se prueba intensivamente tanto en el laboratorio como en las aulas, por lo que se consigue una robustez considerable.

Los objetivos de diseño de los robots RobEx son:

- Ser apropiado para entornos estructurados, pero que a la vez pueda ser modificado para otros tipos de terreno.
- Conseguir un robot de bajo coste y fácil de construir con capacidad de procesamiento intensivo a bordo.
- Que el precio no esté reñido con la calidad: fiabilidad y robustez.
- Poder ser ampliado con diversos accesorios de sensorización y manipulación, así como llevar varios portátiles abordo.

Como se indicó antes, el robot RobEx es hardware libre. El robot y su diseño se distribuyen bajo la licencia “Creative Commons Attribution-Share Alike 3.0” [15]. El software del microcontrolador se distribuye bajo GPL. Con esto se pretende:

- Que cualquier persona tenga acceso al diseño y software del robot, y pueda fabricarlo por sí misma.
- Que la comunidad participe en la mejora y evolución del robot
- Que cualquier empresa pueda usar o vender RobEx, pero que cualquier modificación hecha al robot o a su software se haga pública y mantenga la misma licencia.

En la figura se muestra una representación del diseño de la parte mecánica de RobEx.

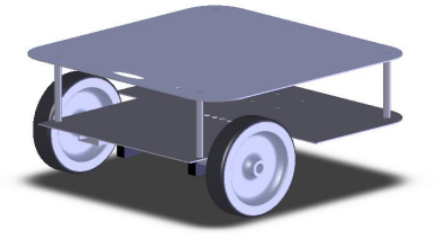

Ilustración 3: Representación del chasis de RobEx.

La estructura del robot está formada por dos planchas de aluminio que se separan y se afianzan entre sí mediante cuatro tubos de acero. Para colocar los motores en el chasis se usan soportes de acero, uno para cada motor. Los soportes tienen dos orificios que se adaptan a los motores y a la vez se sujetan a la plancha de aluminio. Estos orificios son del mismo diámetro que el motor en dichos puntos de apoyo, de forma que éste queda anteriormente sujeto cuando se aprietan los tornillos de fijación. La figura ilustra el diseño de estos soportes.

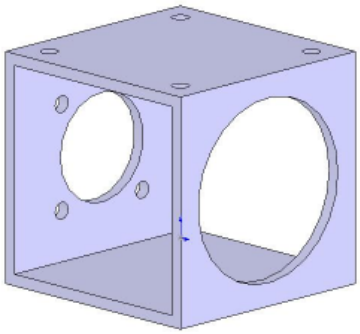

Ilustración 4: Representación de los soportes de los motores

Debido a que el diámetro del orificio de los ejes de las ruedas es de 15 mm, y el de los ejes de los motores es de 6 y 8 mm, se usan unos casquillos en aluminio con un pasador roscado para sujetar éste al motor. La sección mayor de los casquillos

es suficientemente grande como para que puedan quedar unidos a las ruedas por presión. Para asegurar esta unión se utiliza también una gota de cianocrilato. La figura muestra el diseño de los casquillos.

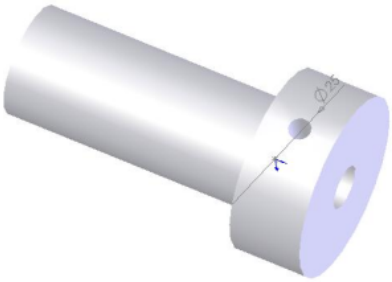

Ilustración 5: Casquillo de transmisión

Para disponer de autonomía energética, el robot incorpora una batería recargable a bordo. Para ello se utiliza un batería de polímero de litio de 21,6V y 3500mAh. Estas baterías son de consumo doméstico común y se suelen utilizar para ampliar la autonomía de ordenadores portátiles, por lo que son asequibles y están sobradamente probadas. Estas baterías proveen al robot de una autonomía de algo menos de dos horas.

La alimentación del robot tiene una tensión de salida de 12V, que corresponde con la tensión de funcionamiento de los motores. Además, tiene una segunda salida de fija de 5V que se puede usar para otros propósitos.

El sistema de control también se encarga de realizar las operaciones y cálculos necesarios para el control PID de los motores. El bucle de control PID de cada motor funciona a una frecuencia de 1 Khz. Por tanto, cada milisegundo se calcula la tensión de salida más apropiada para alcanzar el objetivo actual. En otros términos, el motor sólo está “sin control” o en lazo abierto periodos de 1milisegundo.

La base dispone de un circuito integrado, LSI7266R1, que hace de contador de los pulsos procedentes de los dos codificadores ópticos. Éste se comunica con el microcontrolador mediante dos buses, uno de datos de 8 líneas de entrada y salida y

otro de control de 5 líneas de entrada. Esto permite obtener los datos necesarios para llevar las cuentas de la odometría.

En la figura se pueden ver dos RobEx primitivos a los que se les ha añadido una cámara (ejemplar de la izquierda) y un escáner láser (ejemplar de la derecha).

Ilustración 6: RobEx primitivos, izquierda con cámara y derecha con láser.

Programación orientada a componentes

Dos de los principales problemas que se presentan cuando se crea software son la escalabilidad y la reusabilidad. Estos problemas son especialmente agudos cuando se trata a software que se va a emplear en robótica y es debido a que aprovechar software es algo excepcional en este campo. A pesar de la importancia de la reusabilidad, generalmente se suele perder de vista este aspecto y se acaba creando software monolítico y poco utilizable.

En el ámbito de la robótica es muy común que los investigadores implementen todos los algoritmos con un diseño rígido y orientado a una tarea y/o a un robot específico. De ser así, cuando finaliza la etapa de implementación el software

desarrollado acaba siendo imposible de utilizar. Suele estar tan ligado a una plataforma o tarea específica que resulta más práctico empezar de cero (debido a las dependencias y efectos colaterales derivados de su rigidez).

La programación orientada a componentes [23] surge como solución a este tipo de problemas. Es un enfoque que no tiene necesariamente que ver con concurrencia o computación distribuida, sino con cómo se organiza el software. La programación orientada a objetos representó un gran avance respecto a la programación estructurada, sin embargo, cuando el número de clases y sus interdependencias crece, resulta demasiado difícil entender el sistema globalmente. Es por tanto beneficioso disponer de un grado mayor de encapsulamiento, que aune diferentes clases relacionadas bajo una interfaz única, y permita comprender el sistema con menor grado de detalle. La programación orientada a componentes, que se propuso para solucionar este tipo de problemas, muchos la ven como el siguiente paso tras la programación orientada a objetos. [22]

Un componente es un programa que provee una interfaz que otros programas o componentes pueden utilizar. Es una pieza de código elaborado o a elaborar que codifica una determinada funcionalidad. Son los elementos básicos en la construcción de las aplicaciones en esta arquitectura, que se juntan y combinan para llevar a cabo una tarea concreta. A su vez, estos programas hacen uso de programación orientada a objetos (así como en programación orientada a objetos se hace uso de programación estructurada). Esta división en piezas de software de mayor tamaño que las clases, implementadas como subprogramas independientes ayuda a mitigar los problemas de los que se ha hablado antes y a aislar errores. Además, desde su carácter intrínsecamente distribuido ayudan a repartir la carga de cómputo entre núcleos, incluso, dependiendo de la tecnología usada, entre diferentes ordenadores en red.

Desde el punto de vista del diseño se pueden ver como una gran clase que ofrece métodos públicos. La única diferencia desde este punto de vista es que la

complejidad introducida por las clases de las que depende el componente (o clase) que no son del dominio del problema desaparece porque la interfaz del componente las esconde. Un componente puede ser arbitrariamente complejo, pero un paso atrás, lo único que se ve es la interfaz que ofrece. Esto es lo que lo define como componente.

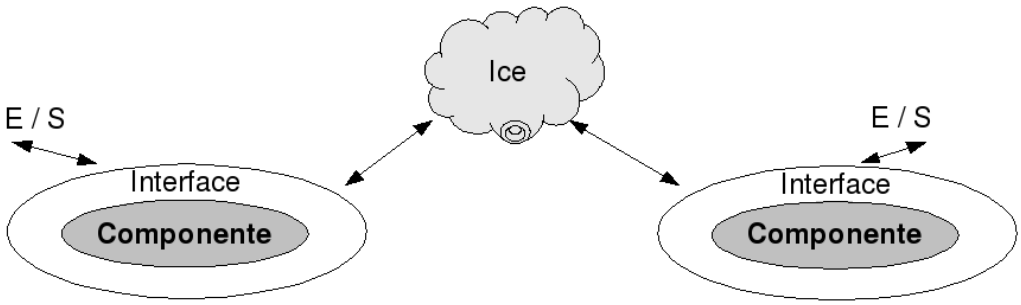

Ilustración 7: Representación genérica de componentes

Por tanto, si cada componente realiza una serie de tareas o responde a una serie de órdenes, necesitamos quien se encargue de esa comunicación. Es donde entra en juego Ice, pieza clave para la comprensión de la arquitectura y para la funcionalidad de la misma. Ice es un framework de comunicación muy interesante para su uso en robótica. A pesar de estar desarrollado por una empresa privada es open-source y está licenciado bajo GPL. Además de su carácter libre hay otras dos razones para elegir Ice.

- La primera razón es su facilidad de uso: al contrario que otras tecnologías como Corba, la filosofía de Ice es soportar sólo aquellas características que los usuarios vayan realmente a necesitar y evitar introducir otras características que raramente se usan y dificultan el aprendizaje.

- La segunda razón es la eficiencia: si bien Ice codifica la comunicación eficientemente tanto en términos de tiempo como de espacio, otras

alternativas suelen codificar la comunicación en XML. El uso de XML tiene ventajas en otras aplicaciones donde es conveniente que los humanos puedan entender el tráfico y no sean factores críticos ni la latencia ni la eficiencia, pero dentro de un robot esto no ocurre.

Ice soporta dos tipos de comunicación, por llamada remota (tipo RPC) o por suscripción (mediante un servicio que hace de servidor de mensajes). Sin embargo, este último introduce un paso intermedio cuya latencia hace desaconsejable su uso en robótica donde la ejecución en tiempo real es fundamental.

Para realizar una conexión con un componente lo único que se ha de conocer es su "endpoint", es decir, la información necesaria para realizar la conexión: la dirección o nombre del ordenador al que se va conectar, el protocolo, el puerto y el nombre de la interfaz. Por ejemplo:

$$< \text{nombre} >:< \text{tcp/udp} > -\text{p puerto} - \text{h host}$$

donde 'nombre' es el nombre de la interfaz del componente al que se quiere hacer la conexión, 'puerto' es el puerto tcp o udp en el que el componente esté escuchando y 'host' el nombre del ordenador donde el componente se está ejecutando.

Desde el punto de vista del programador, una vez la conexión está hecha, el uso de componentes Ice es extremadamente simple. Cuando se realiza una conexión se crea una instancia de un proxy al componente en forma de objeto.

Al ejecutar un método del componente proxy el framework se encarga automáticamente de redirigir la llamada al componente remoto, por lo que la instancia del proxy se utiliza como si se tratase de una instancia de un objeto con la funcionalidad del componente.

Esta idea de usar un recurso remoto dentro de un programa no es nueva. Antes de la llegada de la programación orientada a objetos, ya había un protocolo en Unix llamado RPC para hacer llamadas remotas a procedimientos. Más tarde

surgieron tecnologías como RMI, CORBA, DCOM, o Ice. [8]

Para usar programación orientada a componentes se ha de dividir el diseño del software en piezas que ofrezcan una interfaz. A cambio obtendremos mayor reusabilidad, utilizando los mismos componentes en diferentes contextos, de esta forma se reduce considerablemente el tiempo, el coste y el esfuerzo de desarrollo de nuevas aplicaciones, aumentando a la vez la flexibilidad, reutilización y fiabilidad de las mismas. Será más fácil aislar y encontrar fallos, consiguiendo eliminar la necesidad de contemplar cientos de clases para comprender el software desarrollado.

RoboComp

RoboComp [13], es un repositorio de componentes basados en Ice con aplicaciones en robótica y visión artificial. RoboComp se comenzó a desarrollar en Robolab en 2005. Actualmente el proyecto ha sido migrado a SourceForge, donde, además de tener la página del proyecto (http://sf.net/projects/robocomp), dispone de un wiki (http://robocomp.wiki.sf.net/), donde hay documentación y un repositorio al que se puede acceder incluso directamente con un navegador web (https://robocomp.svn.sf.net/svnroot/robocomp).

Dispone de componentes para captura y visualización de vídeo, control del robot RobEx, detección y mantenimiento de regiones de interés (ROI), lectura y visualización de láser, lectura de joystick y navegación, entre otros muchos. Además dispone de un generador automático de componentes nuevos, de un programa gráfico de manipulación y control de componentes, managerComp y hasta de un componente encargado de grabar los datos producidos por los sensores para su posterior reproducción, replayComp.

La figura muestra un grafo en el que se pueden ver los componentes de los que dispone RoboComp y las dependencias entre ellos.

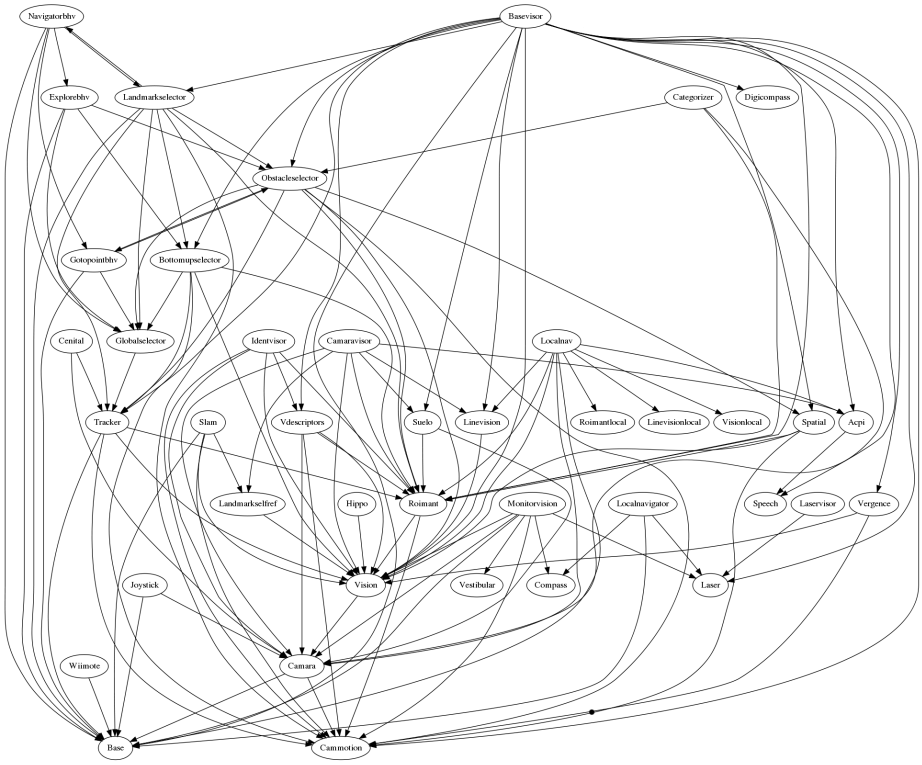

Ilustración 8: Grafo de componentes de RoboComp.

Entorno de desarrollo

A la luz de lo leído es fácil suponer que para desarrollar una aplicación basada en esta arquitectura, es necesario utilizar un conjunto de herramientas mas o menos complejas y librerías que soporten las tareas exigidas a las capacidades del robot, la naturaleza de los problemas, eminentemente en tiempo real, a su vez exige que el tiempo de proceso sea mínimo y siempre se persigue, por filosofía del laboratorio y en la medida de lo posible, un mínimo consumo de recursos.

En las siguientes páginas describiremos el software usado.

IPP/FrameWave

IPP y FrameWave son dos bibliotecas para procesar señales de una y dos

dimensiones. La elección de estas bibliotecas frente a otras de similar objeto se tomó por su gran eficiencia. Ambas bibliotecas son competencia directa, de hecho, mientras IPP es desarrollada por Intel, FrameWave es desarrollada por AMD. La diferencia en el rendimiento respecto a sus competidores radica en que hacen uso de las instrucciones SIMD que aportan las extensiones de x86, 3DNow!, MMX y SSE y derivadas que ambos fabricantes incluyen en sus procesadores.

Ambas biblotecas ofrecen APIs muy similares, pero se diferencian en la licencia bajo la que se distribuyen: mientras FrameWave se distribuye bajo la licencia libre Apache 2.0, IPP es software privativo. A pesar de que es gratuito para el uso personal bajo GNU/Linux, hay que pagar la licencia si se desea usar comercialmente, y su código fuente no es público.

La eficiencia del software desarrollado, no solo en este proyecto sin en el resto del software se debe en gran parte a la de estas bibliotecas. De ellas se han usado multitud de funciones en otro componentes relacionadas con el tratamiento de imágenes. En este trabajo se ha utilizado para el análisis de la señal en el dominio de la frecuencias.

Qt4

Qt4 es un framework de desarrollo cuyo objeto principal es la creación de interfaces gráficas. A pesar de que sea este su principal uso, engloba una gran cantidad de funcionalidades distintas: interfaz multiplataforma con el sistema operativo (sistema de ficheros, procesos, hilos entre otras cosas), comunicación por red mediante sockets, interfaz con OpenGL, conexiones SQL, renderizado de HTML, y módulos de reproducción y streaming multimedia.

Atendiendo al lenguaje de programación, Qt está escrita en C++, pero existen multitud de bindings que hacen posible su uso desde otros lenguajes como Java, C#, Python, Perl y otros muchos.

Inicialmente fue desarrollada por Trolltech, una empresa noruega, bajo una

licencia privativa. Después de varios cambios en la política de licencias pasó a distribuirse bajo una doble licencia GPL/QPL (esta última privativa). Finalmente, tras la compra de Trolltech por parte de Nokia, Qt fue licenciada bajo LGPL en 2009, eliminando así cualquier debate sobre la licencia de la biblioteca.

Ice

Ice es el middleware del que se habló en la sección anterior que permite crear componentes software. Es usado por RoboComp y, por tanto, por los componentes desarrollados en el proyecto de fin de carrera. Es el principal producto de la empresa estadounidense ZeroC, y se distribuye bajo una doble licencia GPL+privativa para habilitar que las empresas desarrollen software privativo con Ice, a cambio del pago de una licencia. Las principales características de Ice frente a otras tecnologías similares es su rapidez, baja latencia y escalabilidad.

Uno de los problemas a resolver a la hora de crear componentes software es la creación de un lenguaje de definición de interfaces. Ice usa Slice, un lenguaje que se creo especialmente para este propósito.

CMake

CMake es una aplicación que permite generar automáticamente ficheros Makefile y ficheros de proyecto de varios IDE como KDevelop o Eclipse entre otros. CMake permite delegar la creación de ficheros Makefile, consiguiendo un resultado multiplataforma y robusto, sin llegar a perder el control del proceso de compilación.

CMake es una iniciativa libre que nació como respuesta a la ausencia de una alternativa suficientemente buena durante el desarrollo de una librería llamada ITK. Si bien dispone de soporte para Qt y algunas otras extensiones, es muy simple hacer nuevas extensiones.

KDevelop

Es un entorno de desarrollo integrado para sistemas GNU/Linux y otros sistemas Unix, publicado bajo licencia GPL, orientado al uso bajo el entorno gráfico KDE, aunque también funciona con otros entornos, como Gnome.

El mismo nombre alude a su perfil: KDevelop - KDE Development Environment (Entorno de Desarrollo para KDE). A diferencia de muchas otras interfaces de desarrollo, KDevelop no cuenta con un compilador propio, por lo que depende de gcc para producir código binario.

managerComp

La aplicación managerComp permite visualizar, tanto gráficamente como en una lista, el estado de los componentes configurados en tiempo real. A pesar de estar integrado en RoboComp, se detalla independientemente por no ser un componente propiamente dicho. Además de la visualización del estado de los componentes, también permite arrancarlos y apagarlos.

PortAudio

La librería que utilizaremos para tratar los datos de entrada/salida de audio en tiempo real será PortAudio. PortAudio es una API de audio para tiempo real, la cual trabaja en C a bajo nivel. Pertenece al proyecto PortMusic y una de las características a remarcar es que es totalmente gratuito, de código abierto y multiplataforma.

Fue elegida por proporcionar un control eficiente y directo del sonido, sin necesidad de un control hardware complicado.

Alsa

Advanced Linux Sound Architecture (conocido por el acrónimo ALSA) es un componente del núcleo Linux destinado a sustituir a Open Sound System, licenciado bajo GPL.

Algunas de las metas de este proyecto desde su concepción fueron la configuración automática de tarjetas de sonido y el manejo de múltiples dispositivos de sonido en un sólo sistema.

Está por debajo de PortAudio y es quien habla con el hardware, configurándolo con los parámetros propuestos por el componente tanto en la captura como en la reproducción.

OpenSSH

OpenSSH es una impementación libre del protocolo SSH (Secure SHell) que ofrece tanto la parte del servidor como la del cliente. Esta herramienta es importante en la puesta en marcha de proyecto porque, junto a managerComp, nos permite ejecutar rápida y remotamente los componentes que se deseen. Además, gracias a que permite autenticación basada en llaves, se puede trabajar de forma segura sin necesidad de escribir la contraseña cada vez que se quiere cambiar el estado de algún componente.

OpenSSH es una iniciativa de los desarrolladores de OpenBSD. Se distribuye con distintas licencias (dependiendo de la pieza), pero a pesar de esto, todas ellas son libres, GPL o BSD.

GNU/Linux

Finalmente el sistema operativo usado durante el desarrollo y los experimentos, GNU/Linux. Gracias a él disponemos de un entorno de desarrollo gratuito y libre, herramientas de compilación, depuración, y una gran cantidad de bibliotecas complementarias. Se ha utilizado la distribución Kubuntu y Ubuntu pero por razones subjetivas.

Capítulo 3

Estudio del problema.

En este capítulo expondremos las ideas básicas del funcionamiento del oído humano, para comprender el sistema que de alguna manera queremos imitar, posteriormente enunciaremos las distintas estrategias que adoptamos, en el mundo real, para resolver la localización de fuentes sonoras.

El sistema auditivo humano

El oído humano convierte las ondas sonoras en señales eléctricas que se transmiten por el nervio acústico hasta el cerebro, en donde el sonido es interpretado. La audición es un proceso complejo que involucra principalmente el trabajo conjunto del cerebro y el órgano y es debido a esta interacción que poseemos la capacidad de identificar y ubicar diferentes sonidos que se encuentran en el entorno y que llegan a nosotros a través del medio que nos rodea.

El oído se divide en tres partes, oído externo, medio e interno.

- El oído externo está constituido por el pabellón auditivo (oreja), el conducto auditivo y el tímpano. Las ondas sonoras son recogidas por el pabellón que las conduce a través del conducto auditivo hacia la membrana del tímpano

- El oído medio es una cavidad limitada por el tímpano por un lado y por

la base de la cóclea por el otro. En su interior hay tres huesecillos denominados martillo, yunque y estribo. La cabeza del martillo se apoya sobre el tímpano y trasmite vibraciones a través del yunque al estribo. A su vez éste último se apoya en una de las dos membranas que cierran la cóclea, la ventana oval.

- Oído interno, es una cavidad hermética cuyo interior está anegado por un líquido denominado linfa. Consta de tres elementos: los canales semicirculares, el vestíbulo y la cóclea. Los canales semicirculares no tienen relación directa con la audición, tienen que ver con el equilibrio. Las vibraciones de la ventana oval del vestíbulo son transformadas en la cóclea. Las señales de la cóclea son codificadas y transformadas en impulsos electroquímicos que se propagan por el nervio acústico al cerebro.[10]

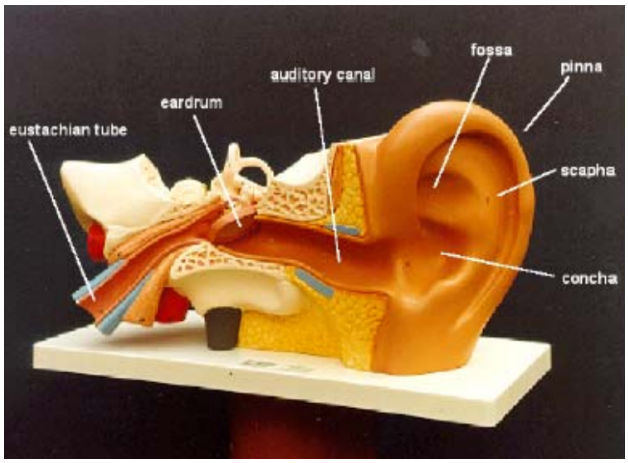

Ilustración 9: Sistema auditivo humano.

Cuando un sonido incide en el oído, una parte de la energía es reflejada mientras que otra parte es absorbida y pasa al interior del oído. El interior del oído está compuesto por múltiples cavidades que resuenan a diferentes frecuencias, es por esto que el sonido que llega a la oreja es modificado y puesto a punto para que los órganos más internos se encarguen de traducirlo a señales que puedan ser interpretadas por el cerebro.

Comprendido lo anterior, estudiemos ahora en detalle las tres partes del oído, entender la naturaleza del órgano a imitar y su funcionamiento, puede ser clave para la toma de decisiones futuras. Una definición más extensa del oído externo la encontramos en [11].

Oido externo

Embriológicamente el pabellón de la oreja y la capa media del tímpano derivan del mesodermo, del primer y segundo arco branquial; el conducto auditivo externo y la capa externa del tímpano del ectodermo superficial, de la primer hendidura braquial; y la capa media del tímpano del endodermo, de la primera bolsa faringea.

El oído esta situado en el hueso temporal del craneo. Está diseñado estructuralmente para, durante el proceso de audición, recoger las ondas sonoras y dirigirlas hacia el interior.

El oído externo consta de:

- El pabellón auricular, un cartílago plano elástico que tiene forma del extremo de una trompeta y esta cubierto por piel gruesa:Compuesto por hélix o borde exterior replegado, antihélix o eminencia central del pabellón que termina en una elevación llamada antítrago, concha o parte centrar y lóbulo, que es la parte inferior.

- El canal auditivo externo, un conducto (tubo) curvo de aproximadamente 2.5 cm de longitud que se encuentra en el hueso temporal. Además posee folículos pilosos, glándulas sebáceas (productoras de cera) y glándulas de ovillo que son las glándulas que dan color a la cera.

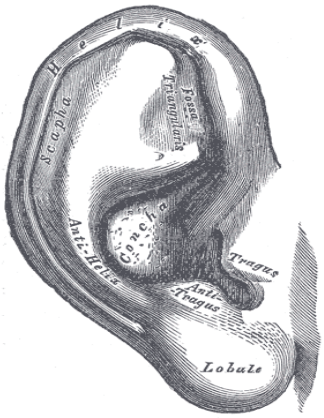

Ilustración 10: Pabellón auditivo.

- La membrana timpánica, también denominada tímpano, que es una porción fina de tejido conectivo fibroso, semitransparente, que se encuentra entre el conducto auditivo externo y el oído medio.

Oído medio

El Oído Medio es una cavidad casi cuadrada situada en el peñasco, región interior del Hueso temporal. Comprende la Caja timpánica, la Cadena de huesecillos y. las Células o celdillas mastoídeas [2].

- La Caja timpánica, es una pequeña cavidad llena de aire, gracias al conducto denominado Trompa de Eustaquio, que la comunica con las fosas nasales, por lo cual la presión de aire contenido en la Caja timpánica es la misma que la del ambiente. Las paredes de la trompa están ordenadamente apiladas unas sobre otras y al efectuarse el acto de deglución se separan para dejar entrar aire nuevo en el oído medio. La Caja timpánica se comunica con el oído interno mediante dos aberturas provistas de una fina membrana la Ventana Oval y la Ventana Redonda.

- La Cadena de huesecillos, se compone de 3 pequeños huesos, Martillo (22-24 mg), Yunque (25 mg) y Estribo (2 mg).

- Las Células o celdillas mastoídeas, están representadas por cavidades irregulares del hueso temporal (Antrum) y su papel en el fenómeno de la audición no se conoce con exactitud, aunque parecen servir para aumentar la cavidad de la caja timpánica.

- La Trompa de Eustaquio conecta la nasofaringe con la caja timpánica. Compuesta por la parte interna o cartilaginosa que mide 24 mm de longitud. Y la externa u ósea que mide 12 mm. La misión de la trompa auditiva consiste en igualar las presiones del oído medio con la atmosférica, para que la membrana timpánica se mueva sin problemas. Al dejar que el aire entre y salga de la cavidad timpánica, esta trompa equilibra la presión a ambos lados de la membrana. En adultos forma un ángulo de 45º con la horizontal, mientras que en lactantes este ángulo es de 15º.

Funcionamiento de la cadena de huesecillos.

El Martillo a través de su "mango" se halla unido a la membrana timpánica, en su extremo opuesto se une firmemente con el Yunque, de manera que siempre que el Martillo se mueve el Yunque se mueve al unísono. El extremo opuesto del Yunque se articula con el "Tallo del estribo" y la base del estribo se apoya en la abertura de la ventana oval, donde los sonidos son transmitidos al oído interno.

La articulación del Yunque con el estribo hace que éste último gire hacia atrás cada vez que el mango del martillo se mueve hacia dentro, y hacia fuera cada vez que el Martillo va hacia fuera, lo cual provoca el desplazamiento hacia dentro y hacia fuera de la Base del Estribo al nivel de la ventana oval, produciéndose el movimiento del fluido coclear en el oído interno.

Como todo sistema vibrante que tiene inercia y un componente elástico, también tiene una frecuencia natural en la cual puede vibrar más fácilmente. Esta es la que recibe el nombre de frecuencia resonante. Como el sistema de huesecillos tiene inercia, y como se halla suspendido de ligamentos elásticos, tiene una

frecuencia de resonancia natural que varía desde 700 a 1400 Hz.

Oído interno.

Es la parte más delicada del sistema auditivo y, por lo mismo, la que se encuentra mejor defendida, pues se halla incrustada en la parte gruesa y petrosa del hueso temporal denominada peñasco. No tiene otra comunicación con el exterior que las ventanas oval y redonda (que dan al oído medio) en tanto que por dentro se relaciona directamente con el cerebro por medio del nervio acústico. El oído interno comprende 3 partes: el vestíbulo, los canales o conductos semicirculares y el caracol o cóclea. [2]

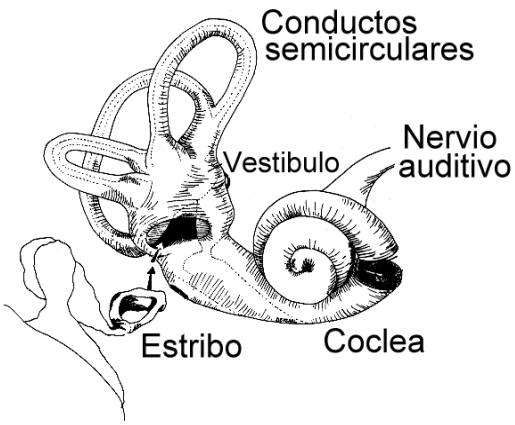

Ilustración 11: Oído interno.

El vestíbulo posee dos orificios (ventanas oval y redonda) tapados por sendas membranas. La ventana oval está unida al estribo y recibe de él sus vibraciones. La cóclea se divide longitudinalmente por la membrana basilar, sobre la que se asientan los filamentos terminales del nervio auditivo. Cuando el estribo empuja la ventana oval, se produce una sobrepresión en la parte superior de la cóclea que obliga a circular el fluido linfático hacia la cavidad inferior a través del helicotrema, mientras que la membrana basilar se deforma hacia abajo. Finalmente, la membrana elástica que cierra la ventana redonda cede hacia afuera.

Cuando el estribo se mueve hacia la izquierda y la derecha, aumentando y disminuyendo la presión del líquido contenido encima de la membrana basilar, aparece una onda que se desplaza de izquierda a derecha a lo largo de la membrana. Esta onda puede visualizarse como un movimiento de traslación hacia arriba y hacia abajo de la membrana. Su velocidad de avance depende de la frecuencia y de las características de la membrana basilar. En algún punto de la cóclea la velocidad es cero. Cerca de ese punto, la membrana oscila hacia arriba y hacia abajo con mayor fuerza y absorbe la energía de la onda. Cada punto de la membrana basilar responde así a una determinada frecuencia.

Cuando el oído recibe un sonido con varias frecuencias, cada una de ellas excita un punto en la membrana basilar, de modo que el cerebro puede interpretar además de la altura del sonido su timbre, sin más que discernir qué terminaciones nerviosas fueron excitadas y con cuánta intensidad. Es decir, el oído interno funciona como un analizador de sonidos.

Audición biaural.

El cerebro humano, para interpretar un sonido, ha de conjugar la información que le llega de ambos oídos. La información que el cerebro recibe de cada uno de los oídos es diferente —salvo cuando están equidistantes de la fuente—, porque ambos oídos están físicamente separados entre sí por la cabeza. Esta diferencia en la situación de los oídos es la que le permite al cerebro localizar la fuente sonora.

En el sistema auditivo la sensación tridimensional está relacionada con la diferencia de amplitud y tiempo que recibe cada oído. Es decir, la localización de los sonidos en el espacio se consigue con el procesamiento por separado de la información de cada oreja y con la posterior comparación de fase y nivel entre ambas señales.

La técnica de grabación del sonido simulando el proceso de audición humano

se conoce como holofonía. [5]

La holofonía es un método de espacialización sonora que viene a ser para el audio lo que la holografía para la imagen. Para conseguir que el cerebro sea capaz de adivinar la posición de la fuente de sonido, se graban las secuencias de cada oído independientemente empleando un torso o cabeza de maniquí, construido con materiales con propiedades físicas similares a los del cuerpo humano y equipado con dos micrófonos omnidireccionales situados en la posición de los tímpanos.

Ilustración 12: Micrófono maniquí Neumann ku100.

Aclarar que no es lo mismo que las grabaciones estéreos convencionales, una grabación estéreo posee dos canales izquierdo y derecho pero nada nos dice de donde se encuentran los instrumentos.

El sonido holofónico fue desarrollado por primera vez, en 1980, por el argentino Hugo Zuccarelli, aplicando el concepto del holograma al sonido. Con esta técnica conseguía perfeccionar aún más el sistema de grabación biaural[5]. Zuccarelli creó su sistema artificial Ringo, con el que afirma que los oídos emiten sonidos que actúan a modo de interferencia, lo cual, es el fundamento de su teoría de localización espacial. Dado que los efectos de interferencia de las ondas de emisión del oído y las de los sonidos a escuchar , son asimétricos, estas interferencias darían al cerebro los

necesarios parámetros para que una localización espacial fuese completa incluso en el plano monoaural. Esta teoría nunca fue aceptada por la comunidad científica

Como curiosidad, y dado que el autor de este texto es un consagrado melómano, se apunta que "The Final Cut" de Pink Floyd fue el primer álbum comercial grabado con esta técnica gracias a la colaboración de Hugo Zuccarelli, quien era tan fanático de esta banda inglesa como de las experimentaciones con el sonido, fue quien les proporcionó la primera prueba de sonido holofónico que terminó convenciendo a los músicos de cambiar el sistema estéreo, que ya no les era suficiente, por el sistema holofónico. Para ello les mostró su famosa grabación de la caja de cerillas [ ].

Ilustración 13: Detalle álbum The final cut de Pink Floyd.

Llegados a este punto, convergemos en que existen señales primarias que hacen posible la habilidad para ubicar sonidos en un contexto tridimensional. Las señales de sonido primarias incluyen la diferencia de intensidad interaural, el retardo de tiempo interaural y la modificación espectral implementada por el oído externo.

Además de las señales de sonido primarias, existen factores secundarios que contribuyen a la capacidad de localización. Entre ellas encontramos las reflexiones en los hombros y el torso, las reflexiones de los objetos del entorno (paredes, muebles, etc.) así como algunas consideraciones psicológicas. Es importante destacar estas últimas ya que predisponen al individuo en el proceso auditivo, por ejemplo, si se escucha un helicóptero se tiende a pensar que se encuentra volando en el aire y no en tierra. Asimismo, si se escucha ladrar a un perro se esperaría que estuviera debajo de nosotros y no encima. Las indicaciones visuales también soportan los factores auditivos, cuando el cerebro combina las imágenes visuales con las imágenes acústicas nos provee con una percepción tridimensional de nuestros alrededores.

Técnicas Interaurales

Las orejas están situadas simétricamente en los lados de la cabeza, la cual tiene un ancho aproximado de 16 cm. Cuando una fuente de sonido se encuentra directamente en frente del sujeto, ambas orejas están expuestas al sonido de la misma forma. Sin embargo cuando la fuente de sonido se mueve hacia un lado de la cabeza, una oreja quedará más expuesta al sonido que la otra y esta última queda dentro del espacio de sombra que genera la cabeza; este efecto, que se ilustra en la figura, genera diferencias en las amplitudes de las señales de sonido que llegan a cada oreja desde la fuente.[6]

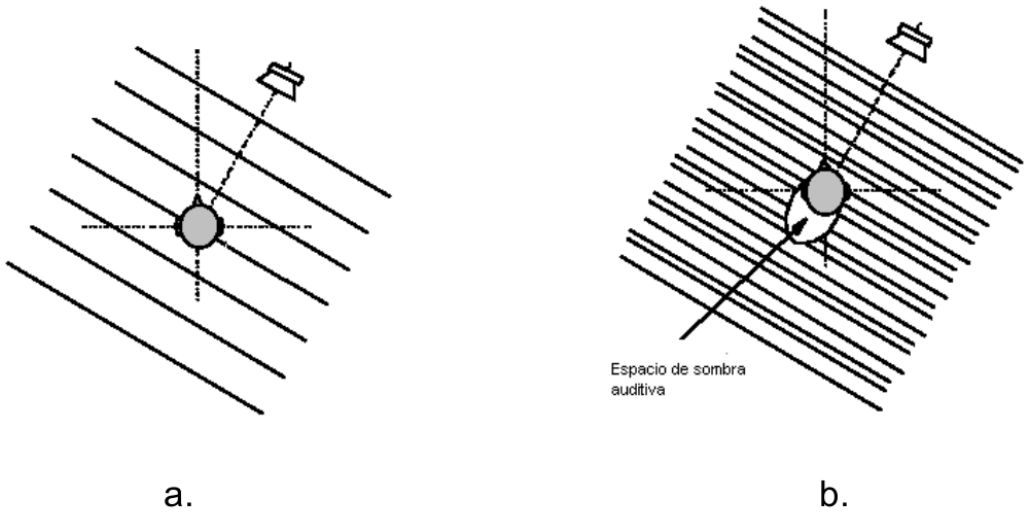

Ilustración 14: a) Difracción para longitud de onda mayor al tamaño de la cabeza.

b) Difracción para longitud de onda menor al tamaño de la cabeza.

Los efectos de difracción alrededor de la cabeza complican los efectos de sombra. Cuando una onda de sonido encuentra barreras, salen a relucir las propiedades de onda de la energía: el sonido se difracta alrededor del obstáculo de una forma análoga a cómo la luz se difracta alrededor de una rendija. Como regla general, las ondas de sonido pueden difractarse eficientemente alrededor de los

objetos cuando su longitud de onda es significativamente mayor que el tamaño del objeto, mientras que cuando el tamaño del mismo es mucho mayor que la longitud de onda, los efectos de difracción son mínimos. En general los efectos de difracción son considerables principalmente a frecuencias entre 700 Hz y 8 Khz.

Diferencia de tiempo interaural.

El retardo interaural (ITD, de sus siglas en inglés *Interaural Time Difference*) se refiere a la diferencia entre el tiempo de llegada de las señales a los oídos izquierdo y derecho. A menos que la fuente de sonido esté en uno de los polos (directamente en frente, detrás, arriba o abajo) el frente de onda llegará a los oídos en tiempos diferentes. Por ejemplo si la fuente se encuentra exactamente en el lado derecho de sujeto, el sonido deberá viajar una distancia mayor para alcanzar el oído izquierdo de la que debe viajar para alcanzar el sonido derecho.

Por lo general se establecen tres planos característicos en los experimentos destinados a estudiar la localización por parte del ser humano. [1]

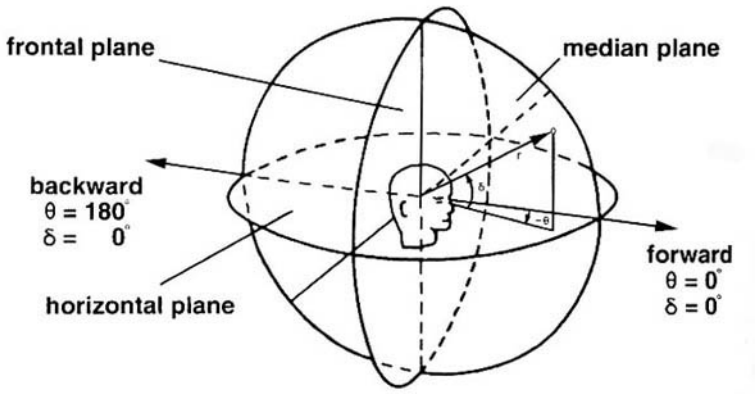

Ilustración 15: Planos de localización.

La localización se realiza a partir de la determinación de una dirección y una distancia. La dirección de una fuente sonora, a su vez, se establece a partir de la determinación de un ángulo lateral y de un ángulo de elevación. Si para simplificar el

cálculo de los valores de desfase interaural en una cabeza conceptual utilizamos una vista plana como en el siguiente dibujo:[9]

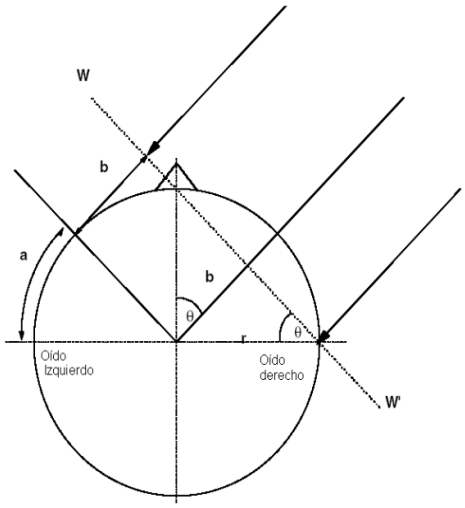

Ilustración 16: Desfase interaural en una cabeza esférica.

Se está recibiendo una señal de sonido desde una fuente con un determinado ángulo de acimut θ (aproximadamente 45º). Cuando el frente de onda llega al oído derecho, puede verse que existe una distancia $a+b$ que debe recorrer antes de encontrar el oído izquierdo. Por la simetría de la configuración, b es igual a la distancia desde el centro de la cabeza hasta el frente de onda comprendido entre W y W' por lo tanto:

$$b = r * sen(\theta)$$

También del diagrama se desprende que a representa el arco de la circunferencia subtendido por el ángulo θ , por lo tanto la distancia $a+b$ es igual a:

$$a+b = \left(\frac{\theta}{360}\right) 2\pi r + r\, sen(\theta)$$

Puede verse que en el límite cuando tiende a cero, la longitud del camino es

igual a 0. También, cuando θ tiende a 90º la longitud de $a+b$ es en promedio 19.3 cm. y el retardo asociado es aproximadamente 560 μs. En la práctica, el ITD ha sido medido y ha resultado un poco más grande, posiblemente debido a la naturaleza no esférica de la cabeza, a complejas situaciones de difracción y a efectos producidos por las superficies [7].

Cabe recordar igualmente que la respuesta espectral del oído externo modifica el sonido que llega al oído interno. Las complejas formas que pueden verse en la oreja resuenan a diferentes frecuencias dependiendo de la dirección de donde viene el sonido, modificando de esta forma el espectro de las señales de sonido antes de que alcancen la membrana del tímpano. El cerebro analiza esta información de ambos oídos como parte del proceso de localización, en conjunto con la diferencia de intensidades.

Diferencia de Intensidad interaural.

La diferencia interaural de intensidades (IID de sus siglas en inglés, Interaural Intensity Difference), se da principalmente a partir de las diferentes distancias que deben recorrer las ondas para llegar a uno y otro oído, pero también por la sombra acústica producida por la cabeza del individuo, la difracción de la onda, que estudiábamos al principio. Una fuente sonora ubicada en una posición directamente al frente, atrás, arriba o abajo del oyente, será percibida con igual nivel de intensidad acústica por cada uno de los oídos. Si la fuente sonora es girada unos cuantos grados a la izquierda o derecha de esta posición inicial, la percepción de nivel se inclinará hacia el oído que se encuentre más cercano a la fuente sonora.

Las neuronas encargadas de registrar las IID son capaces, tras recibir el estímulo en una oreja, de ir inhibiéndolo en la otra oreja. Digamos que la magnitud de la respuesta de la célula depende de la correlación de fuerzas entre las dos entradas, que a su vez, depende de la intensidad de sonido en los oídos. Incluso la respuesta en medidas humanas de los niveles de diferencias varía a distintas frecuencias y también para cada sujeto. En el experimento de B.J.C. Moore [1], imponía la condición de emitir un sonido puro. Los resultados de la diferencia de intensidades se resumen en la siguiente tabla, la cual nos da una idea bastante clara de la complejidad del problema y pone una vez más de manifiesto la capacidad sensorial humana.

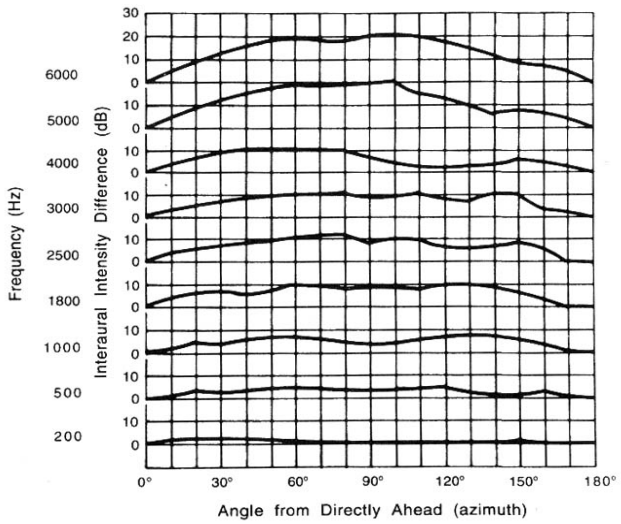

Ilustración 17: Diferencia Interaural de Intensidad.

Un efecto curioso que todos sentimos cuando escuchamos música estéreo y que es utilizado practicamente en todas las mezclas para simular los desplazamientos laterales, en el argot técnico conocido como *paneo,* es el Efecto Hass o Efecto de Precedencia [16] . El cerebro humano toma en consideración las pequeñas diferencias en el tiempo y nivel de llegada de los sonidos a cada uno de los oídos. En el caso de sistemas de altavoces estéreo, al ser emitido un sonido a través de ambos altavoces con el mismo nivel, como en el caso de la figura (a), se crea una imagen fantasma de

la ubicación de la fuente productora del sonido, situándola en el punto medio de los dos altavoces. Si se varía el nivel de una de las señales de los altavoces, el cerebro tiende a posicionar a las fuentes sonoras en ubicaciones más próximas al altavoz con mayor nivel, tal como se muestra en la figura (b). A este fenómeno se lo conoce como el Efecto Hass, y es tal que pueden producirse diferencias de hasta 8dB en la intensidad de la señal secundaria mientras que se seguirá percibiendo la posición inicial como el punto desde el cual el sonido es generado. [16]

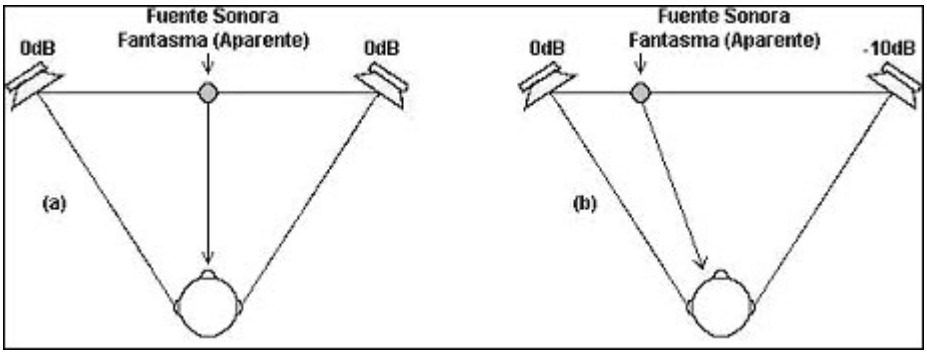

Ilustración 18: Efecto Hass. (a) Señales de igual nivel de sonoridad. (b) Señales de distinto nivel.

Capítulo 4

Diseño y desarrollo del sistema.

Tras estudiar el problema y comprobar las diferentes alternativas, se observa que hay que construir un sistema versátil que nos permita probar las distintas opciones teniendo en cuenta, además, que lo probaremos en un entorno real con las complicaciones derivadas. Así, habrá que proporcionar herramientas para el tratamiento de las soluciones que utilizan la diferencia de tiempo y para las que utilizan la diferencia de intensidades, sin olvidar la utilización de programas externos de edición y análisis de audio que nos permitan ahorrar tiempo a la hora de descartar o aceptar alguna solución.

Al trabajar con la programación orientada a componentes debemos definir los nuevos componentes a crear, sus entradas y salidas, y las funciones que tendrán. A continuación se explica la planificación del proyecto.

Planificación del proyecto.

Inicialmente se pretenden crear tres componentes para añadir al repositorio de RoboComp; microComp captura el sonido, soundComp recibe el sonido lo monitoriza y lo trata si procede, localizadorComp interpreta las señales para determinar el ángulo de procedencia de la fuente sonora enviando el giro a la base.

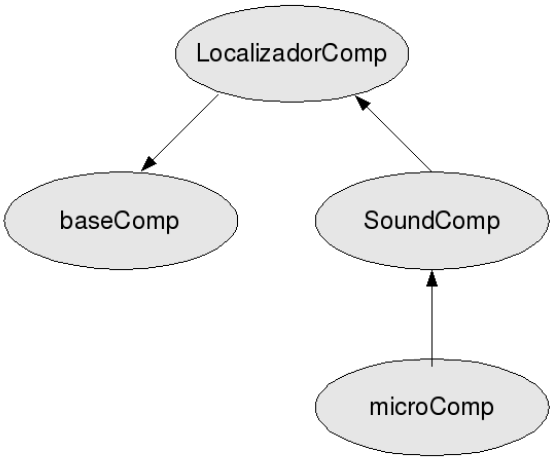

Ilustración 19: grafo de componentes.

Primeramente, se estudiarán las distintas formas de adquirir sonido a través de un medio externo, en nuestro caso micrófonos, una vez comprendidas y examinadas las distintas alternativas que ofrece el mundo del software libre en la captura de audio se utilizará la que más se adapte a nuestra necesidades. El robot dispondrá de dos configuraciones hardware:

- Un par de cámaras web conectadas a sendos puertos USB con micrófonos omnidireccionales.

- Un par de micrófonos omnidireccionales cardiodes, la señal pasará por una digitalizadora, a modo de previo, conectada por USB.

En ambos casos los micrófonos simulan nuestra capacidad auditiva estereofónica antes de llegar al portátil que controla el robot. Este los enviará vía TCP/IP a un PC remoto encargado de controlar/supervisar todo el funcionamiento.

Como decíamos antes, la estructura final del proyecto, supone la realización de tres nuevos componentes para añadirlos a la arquitectura del robot: microComp, soundComp y localizadorComp. Este último estará integrado en soundComp hasta que sea lo suficientemente robusto y versátil o hasta que soundComp sirva el sonido

a otros componentes futuros como reconocedores de voz.

Fase 1: microComp.

El componente microComp simula nuestro oídos. Su función es estar siempre recibiendo las señales acústicas que le llegan desde los micrófonos a través del puerto USB al que está conectada la digitalizadora, o en el caso de las cámaras, recibir de ambos USB simultáneamente. Por tanto se debe construir un driver de captura de audio en Linux. Este componente debe ser muy robusto. Deberá superar pruebas de ejecución en largos periodos de tiempo. Es la parte del cerebro de un oído artificial, capturando todos los sonidos producidos a su alrededor. Es el inicio de un sentido básico en el ser humano. Las dificultades tecnológicas de implementar un driver de audio en cualquier sistema operativo son manifiestas, ya que crear un software capaz de permitirnos tratar los datos de la forma que más nos convenga. No puede haber pérdidas de datos, es decir el flujo de audio debe ser continuo, todo debe ser enviado al PC remoto en paquetes adaptados a la velocidad de la red, debe consumir poco recursos y debe asegurar que el sonido recibido no es adulterado por otros programas que pudiesen estar en ejecución. Por tanto, microComp recoge las señales acústicas del mundo y las envía a soundComp para su posterior tratamiento.

Fase 2: soundComp.

El componente soundComp trata el sonido e interpreta la información que recibe. Para ello se centrará en la parte de la localización de la fuente sonora, sin olvidar que en el futuro se pretende interpretar y reconocer el audio. Debe ser un componente transparente y tendrá alguna funcionalidad para ayudar en la depuración y en la comprobación de que lo escuchado es correcto.

Tendrá una parte dedicada a reproducir de manera estereofónica el sonido capturado. Oiremos en todo momento lo que oye el robot por su oído / micrófono izquierdo por el altavoz o canal izquierdo del PC remoto. Igual ocurrirá con el canal derecho. Al mismo tiempo monitorizará la señal sonora recibida en la pantalla tanto

en el dominio del tiempo como en el de la frecuencia para tener la certeza de que todo funciona como intuimos.

La segunda sección consistirá en filtrar el sonido recibido mediante técnicas de ingeniería acústica, implementando algoritmos capaces de calcular transformadas rápidas de Fourier, para estudiar el sonido en el espectro de las frecuencias. A su vez se deberán fijar umbrales de ruido auto regulables descartando, como nuestro cerebro, lo que no nos interesa. Se podrán realizar cálculos tanto en el dominio del tiempo como en el de la frecuencia, no solo del paquete actual de sonido, sino también en paquetes anteriores para mejorar la interpretación, siempre y cuando no interfieran en la sensación de tiempo real.

La tercera parte consistirá en la implementación de un código capaz de generar ondas sintéticas de una determinada amplitud y frecuencia que puedan ser reproducidas y dibujadas. Esta funcionalidad servirá para depurar errores en este proyecto como son sincronización, reproducción continuada, bloqueos, pérdidas de datos o interrupciones con otros procesos del sistema operativo, pruebas para el correcto funcionamiento de la transformada de Fourier, identificación de un mismo sonido, etc. y como entrenador en un futuro para aspectos relacionados con el reconocimiento de voz.

La cuarta parte será la capacidad de escribir / leer ficheros wav, a partir del audio recibido, los cuales podrán ser reproducidos con otro software y tratados con programas matemáticos donde depurar, descartar, o adoptar soluciones, como por ejemplo Octave.

Una vez que tengamos la certeza de que todo funciona bien entre microComp y soundComp, podremos servir unos datos limpios al tercer componente para que este pueda cumplir la tarea o idea fundamental del proyecto: localizar una fuente sonora en el espacio.

Fase 3: localizadorComp.

El componente localizadorComp (finalmente integrado en soundComp) conseguirá situar una fuente sonora en el espacio que lo rodea, determinar el ángulo y poder acercarse o alejarse según futuras aplicaciones. Se utilizarán técnicas y artículos sobre la materia y nos basaremos en las pruebas y conclusiones de otros investigadores sobre como estimar la posición de un objeto emisor de sonido, adaptando estas ideas a los requerimientos de la tecnología disponible, descartando enfoques erróneos, añadiendo nuevas soluciones y/o modificando las condiciones donde fueron realizadas a las necesidades que implica un sistema en tiempo real como el descrito. En definitiva, debemos converger en una solución que satisfaga las expectativas iniciales, que sea suficientemente robusto para su posterior utilización por parte del grupo de investigación tanto para las asignaturas que imparten, como para sus proyectos.

Creación genérica de un componente.

Antes de proceder con la especificación de los componentes construidos se mostrará como crear un componente genérico. Cuando se incluye un nuevo componente en RoboComp se usa generalmente una herramienta que lo genera automáticamente. El generador de código también incluye el código necesario para crear una conexión a los componentes que se le especifiquen. A los componentes que se generen automáticamente se le pueden añadir después todas las clases que sean necesarias. Además, muchas veces se desea añadir en el fichero de configuración algún parámetro específico y, seguramente, modificar el fichero de interfaz por defecto (que no ofrece ningún método). De no usar el generador automático de código, la creación de componentes, incluso pequeños, sería bastante tediosa, y susceptible de errores ya que es necesario integrar el código del middleware Ice y el código propio del componente.

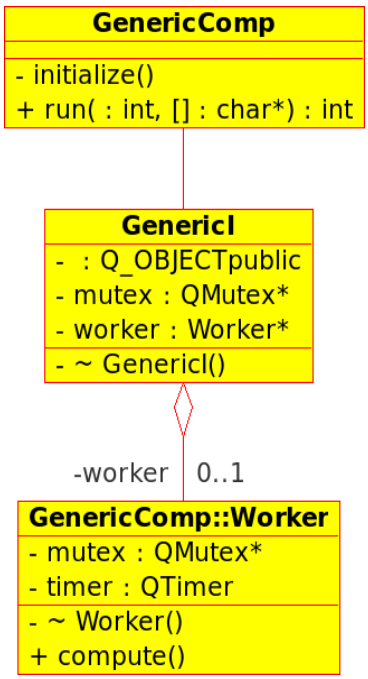

Ilustración 20: Diagrama de clases de un componente genérico.

Como se puede observar en la ilustración, además del código de programa principal, el componente plantilla consta de tres clases: GenericComp[23], GenericI y Worker. Cuando se usa el generador de código éste modifica los nombres de las clases GenericComp y GenericI en función al nombre del componente cambiando Generic por lo que corresponda en función al nombre del componente.

Cada componente consta de, al menos, dos hilos de ejecución, siendo estos el hilo principal y un hilo separado para responder a las llamadas Ice. El último de estos dos hilos se crea cuando se instancia una interfaz Ice (a la clase creada se le tiene que pasar como parámetro la clase que se hace cargo de las llamadas). El hilo principal de ejecución se encarga de la parte activa del componente, de hacer las llamadas Ice necesarias si las hay, y los cálculos asociados al comportamiento del componente. Para facilitar la comprensión del código la clase GenericComp delega en Worker todo el trabajo y GenericI queda como una clase que, de forma transparente y asíncrona, responde a las llamadas remotas.

Las clases Worker y GenericI, que como ya hemos visto se encargan de la parte interna y externa respectivamente, se ejecutan en exclusión mutua gracias a un mutex que comparten. El bloqueo del mutex se realiza cada vez que GenericI recibe una llamada o que Worker va a cambiar alguna parte de su estado que pueda modificar las respuestas de GenericI a sus invocaciones remotas.

Como se puede ver en el diagrama, GenericI tiene acceso a la clase Worker. Dicha condición, que sólo se da en ese sentido, es necesaria porque generalmente toda la información relativa al estado del componente se guarda dentro de Worker.

MicroComp.

Este componente se encarga de la captura del audio. Antes de desarrollarlo se trabajó con distinto software de grabación. Se estudio como grabar en Linux desde un puerto USB, se probaron programas conocidos como Arecord (de ALSA) y se estudió el funcionamiento y el código disponible de uno de los programas más conocidos de edición en el mundo del software libre, Audacity.

En primera instancia la grabación se pretendía hacer directamente con ALSA, pero tras un tiempo se observó que el proceso era demasiado complejo. Se consiguieron resultados locales de grabación de ficheros de audio, pero no se conseguieron los datos en buffers de tamaño apropiado para que pudieran viajar entre el portátil abordo del robot y el PC. Había además que manejar el control del puerto, las posibles interrupciones, implementar mecanismos de colas y evitar desbordamientos, por lo que la opción de ALSA acabó desestimada.

Lo ideal sería tener una API que facilitara la comunicación con los dispositivos. Tras leer, preguntar y descartar algún otro método se optó por utilizar PortAudio[19], que es una librería de entrada/salida audio con funciones de grabación y reproducción escrita en C, lo cuál encajaba perfectamente con nuestro esquema de desarrollo. Además, comprobar que Audacity la utilizaba aumento

nuestra confianza.

El proceso resumido en pocas líneas no hace justicia al tiempo invertido en su elección y puesta a punto. Costó bastante unir microComp con PortAudio y conseguir los resultados esperados. Hubo que modificar el código propuesto por PortAudio ya que no existen funciones para grabar y reproducir, de tal forma que la grabación fuera orientada a pequeños buffers los cuales debían ser reproducidos remotamente. PortAudio está pensada para grabar audio durante un tiempo y transcurrido ese tiempo reproducirlo. Además teníamos que tener control total y poder grabar y reproducir del dispositivo que quisiéramos.

En cualquier caso, fue un acierto elegir está librería ya que, solventados los problemas de comprensión y aplicado los parámetros correctos del hardware, no presenta más problemas. Veamos por su importancia una explicación de está API, esperando nos ayude a entender mejor el sistema.

Portaudio.

Para procesar el audio en tiempo real de forma general con PortAudio debemos seguir la serie de pasos descritos en la figura:

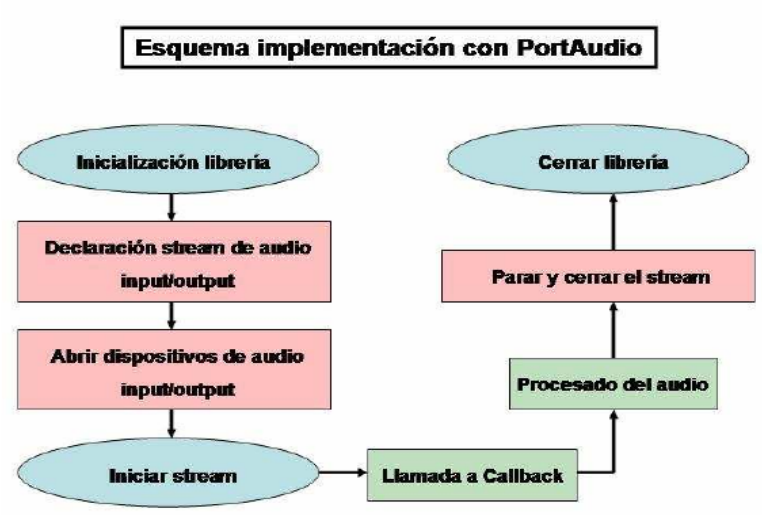

Ilustración 21: Esquema de implementación con PortAudio

Incluimos en nuestro proyecto el archivo "portaudio.h". En él se listan todas las funciones soportadas. Debemos entender PortAudio como un programa ejecutándose en segundo plano, que esta hablando con ALSA y está adquiriendo la información digital brindada por un dispositivo de audio concreto. Y nosotros, periódica y exactamente le reclamaremos esos datos.

Funcionamiento.

Al iniciar microComp arrancamos el hardware a través de PortAudio. Ese elemento de captura tiene una serie de parámetros que debemos conocer, es decir, no todos los micrófonos del mercado pueden operar en el mismo rango de frecuencias, o utilizar la misma resolución en sus datos, lo cual implica tener conocimientos de los materiales utilizados. De la misma forma, necesitamos tener estructuras de datos acordes con sus características, por tanto, necesitamos definir dos estructuras de configuración para el dispositivo, una con la configuración hardware y otra con la configuración deseada adaptada a ese aparato.

La estructura de parámetros hardware que incluye la librería y los parámetros modificables, sin demasiados riesgos, son:

- El dispositivo desde el que queremos grabar, identificado con un número, el cual se puede obtener sabiendo el nombre que tiene para el sistema.

- El número de canales con el que puede trabajar, normalmente mono o estéreo.

- La frecuencia de muestreo, normalmente algún valor de los siguientes 8Khz, 16Khz, 44.1Khz, 48Khz

Como es de esperar, existen otros parámetros pero que son poco relevantes en nuestro caso o que es conveniente mantener en sus valores por defecto.

La estructura encargada de la "forma" de los datos, en nuestro caso buffers, es de tamaño igual a potencia de dos, (512, 1024, 2048), para su posterior tratamiento óptimo con algoritmos de transformadas rápidas de Fourier (FFT en sus siglas en inglés) y esta compuesta por:

- El número frames que se pueden grabar por buffer y por canal.

- El tiempo de grabación.

- El tamaño o resolución de las muestras.

Ambas configuraciones son relativas y la segunda depende de la primera. Así conociendo o bien el tiempo de grabación o bien el tamaño del buffer (número de frames), podemos obtener el resto de parámetros simplemente atendiendo a la frecuencia de muestreo. estos deben cumplir una serie de condiciones proporcionales para que la captura sea correcta. Veamos el siguiente ejemplo para aclarar estos conceptos.

En nuestro caso, para facilitar el proceso de la FFT se elegió un tamaño de 1024 muestras. La frecuencia de muestreo tras estudiar, la primera opción hardware, la de las cámaras USB, fue que estas conseguían una calidad óptima de grabación a 16000 Hz. Esto quiere decir que toman 16000 muestras de sonido en un segundo. Por

lo tanto si queríamos capturar 1024 muestras, el tiempo de grabación en segundos cumple la siguiente ecuación:

$$t = \frac{tamaño\ del\ buffer}{frecuencia\ de\ muestreo}$$

Por lo tanto y para nuestro caso el tiempo 0,064 segundos (64 ms por cada 1024 muestras) es un retraso aceptable en la escucha del sonido, ya que supera los 15 buffer por segundo. Un valor equivalente a los 15 fps a los que se suelen capturar las imágenes en RoboComp.

La resolución o formato de cada muestra varía desde enteros sin signos, hasta reales de 32 bits. Una grabación de un cd de audio normal tiene una resolución de 16 bits y una frecuencia de muestreo de 44.1 Khz. Fijando la frecuencia se eligió el formato "float" de 32 de bits, con valores que varían entre [1,-1] con objeto de representar la onda de manera sencilla y para los cálculos posteriores. Con esta calidad, sabiendo que grabaremos dos canales y que enviamos unos 15 buffers por segundo y que en en el entorno de desarrollo que nos encontramos un "float" ocupa 4 bytes, la tasa de transferencia bruta de datos por segundo es:

$$Tasa = canales * tamaño\ del\ buffer * tamaño\ float * buffers\ por\ segundo$$

$$Tasa = 2 * 1024 * 4 * 15 = 122{,}880\ bytes = ١٢٠\ KB/s$$

Cantidad soportable con creces por el tipo red, LAN, con la que trabajamos.

Cualquier programa realizado con PortAudio requiere un stream o corriente de datos para el audio de entrada. Este stream ha sido asociado con la configuración de las estructuras y de esta forma podríamos tener un flujo de datos asociado por cada dispositivo

```
PaError Pa_OpenStream( PaStream** stream,
                       const PaStreamParameters *inputParameters,
                       const PaStreamParameters *outputParameters,
                       double sampleRate,
                       unsigned long framesPerBuffer,
                       PaStreamFlags streamFlags,
                       PaStreamCallback *streamCallback,
                       void *userData );
```

Los parámetros más relevantes de esta función son los dos últimos: la función callback y el espacio de intercambio de datos entre el usuario (microComp) y PortAudio.

La función Callback es la función principal del proceso de captura. Es llamada una vez se ha creado y abierto el "stream" de audio. Es controlada desde un proceso ajeno a nosotros. Se caracteriza por tener un funcionamiento cíclico, ya que es llamada periódicamente. Se adquiere un determinado número de muestras de audio (marcado por el tamaño del buffer) y se procesan dentro de la función callback. Una vez terminada la función, ésta se vuelve a iniciar de nuevo capturando las siguientes muestras de audio consecutivas y así sucesivamente. Esto se traduce en que la función callback se ejecutará durante el número de muestras del buffer y nosotros podremos recoger, un instante después, los datos del espacio de intercambio para enviarlos al PC remoto.

Al cerrar microComp debemos asegurarnos de cerrar correctamente el dispositivo y de liberar los recursos ocupados (fundamentalmente el espacio de intercambio), así como terminar con la librería y el "stream" asociado. Recordamos que cuando se programan aplicaciones que interfieren con elementos que pueden ser utilizados por otras librerias hay que poner especial énfasis en estas tareas, propias de su naturaleza de driver ya que a menudo se obvian en programas convencionales.

Por tanto y como decíamos, una de las características más importantes de la programación orientada a componentes, es la de que un paso atrás de la interfaz no importa la complejidad del software. Esta afirmación se pone de manifiesto aquí, ya

que la interfaz de microComp es extremadamente sencilla, tanto que mostramos su código a continuación:

```
#ifndef MICRO_ICE
#define MICRO_ICE

module RobolabModMicro
{
        sequence<float> sndType;

        interface Micro
        {
          void getBuffers(out sndType data, out sndType data1 );
        };
};
#endif
```

Un usuario que quiera capturar audio usando microComp solo tendrá que lanzar el componente y escribir una llamada a la función sabiendo que el sonido será devuelto en canales separados. Primero el canal izquierdo luego el canal derecho. Nada tendrá que saber de como funciona el dispositivo. Por ejemplo, existen distintas formas de colocar los datos cuando se graba. Si es mono se graba un dato detrás del otro, si es estereofónico se puede grabar intercalado o sin intercalar. En el intercalado la secuencia de muestras es así: canal izquierdo, canal derecho, canal izquierdo... y si es sin intercalar el canal izquierdo irá colocado al principio y el canal derecho a continuación. Todo esto más lo anteriormente visto queda oculto a la petición del sonido desde fuera del componente.

SoundComp

Es el componente concebido en el diseño para monitorizar las señales recibidas de microComp. Este componente aplicaría un tratamiento a las señales si fuera necesario pero no analizaría la información para determinar la localización de la fuente. Simplemente serviría el sonido a localizadorComp. Sin embargo, la cruda realidad deshizo, para el proyecto, estos planes ya que se trató el audio buscando un

algoritmo que ofreciera buenos resultados y posteriormente se subiría al nivel del localizadorComp, el cual enviaría a baseComp el ángulo hacia donde debíamos dirigirnos.

En cualquier caso sus funciones generales son: la reproducción continua del sonido, de forma que con unos simples cascos conocemos realmente lo que escucha el robot, fundamental para mejorar y entender la sensación con la que debemos jugar, para conseguir el objetivo y dibujar en pantalla la representación de la onda en una ventana típica o similar a los software de edición de audio convencionales y calcular las transformadas rápidas de Fourier y representarlas.

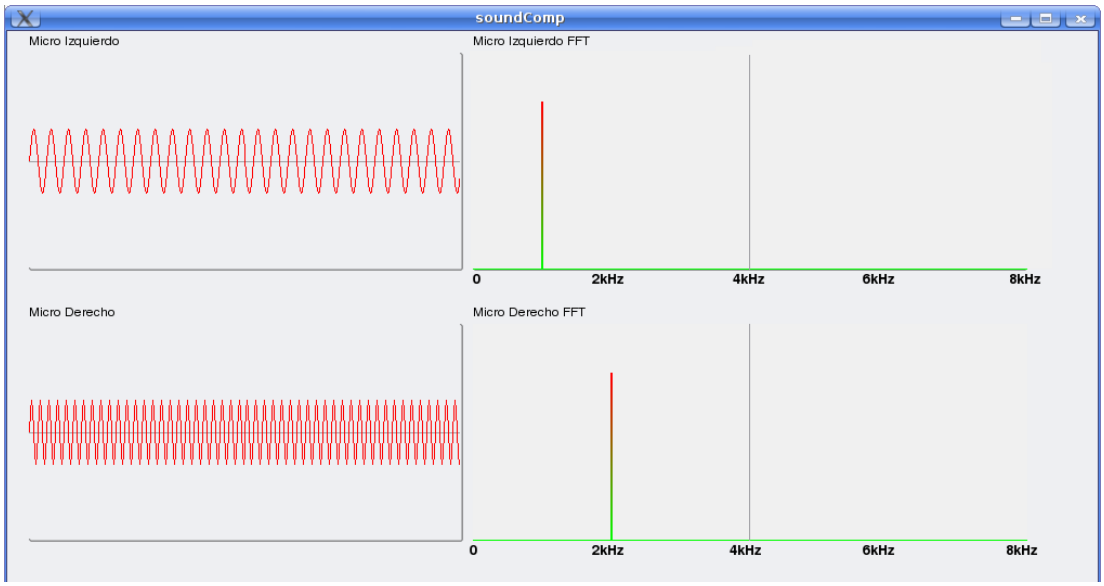

Ilustración 22: soundComp representación de 2 ondas puras sinusoidales de 1Khz y 2Khz con la misma intensidad.

También se escribió el código de una función que permitía suprimir un rango determinado de frecuencias. Dada una frecuencia inicial y una final suprimía la energía contenida y a continuación procedíamos a realizar la transformada inversa de Fourier, volviendo al dominio del tiempo. El supresor de frecuencias, aunque funciona correctamente y elimina la parte que no nos interesa para la orientación, si reproducíamos el sonido, como el corte era muy radical, su reproducción se veía un tanto alterada, resultando a veces hasta graciosa.

La función de lectura y escritura de ficheros de audio sin pérdidas, tipo WAV, resulto muy útil en la verificación de los algoritmos propuestos y conseguimos de esta forma tener un tipo de archivo muy común y utilizado por multitud de aplicaciones externas. Podíamos grabar lo que el robot oía y conseguir un poco de estabilidad en un sistema real donde probar las distintas soluciones teóricas. Este tipo de técnica resulta muy útil para la programación robótica en unas condiciones que al menos se repiten, (un fichero wav siempre va a contener el mismo sonido) y donde se sabe de antemano que ha sucedido, si ha habido una variación significativa de intensidad, o cual es el desfase en la recepción del sonido. Se puede ejecutar la algoritmia planteada y observar los resultados, pudiendo realizar variaciones y comparar resultados, los cuales determinan si las nuevas soluciones son válidas o no, es decir, si sobre las condiciones grabadas no mejora el rendimiento el camino no es correcto. El caso contrario no nos asegura que al enfrentarlo a la realidad vaya a ir mucho mejor, pero al menos es un paso.

La elección de este tipo de ficheros, a parte de por no tener pérdidas, fue debida a dos razones más. La primera por la disponibilidad de la librería libsndfile [20], que es una librería escrita en C, como PortAudio ya que hace medianamente fácil su integración en nuestro código. Está diseñada para la lectura y escritura de ficheros que contienen muestras de sonido del tipo WAV o AIFF entre otros, mediante una interfaz de librería estándar. Se distribuye el código fuente bajo licencia libre, LGPL. Esta licencia tiene la particularidad de que permite copiar y distribuir el software pero no permite la modificación del mismo. Este aspecto es aquí irrelevante puesto que solo necesitamos una API lo suficientemente robusta para manejar los ficheros de audio.

La segunda razón fue que el tipo wav es uno de los formatos que se pueden utilizar sin dificultad con Matlab [21]. Matlab (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo y un lenguaje de programación propio. Como reza el eslogan de su página web, sirve para acelerar la paz entre la ingeniería y la ciencia, y así ha sido. Fue un

elemento crucial para descartar métodos de orientación, y para evitar hacer algoritmos con cálculos tediosos en C++. Era una manera de aplicar más rápidamente soluciones teóricas y demostrar porque no eran válidas en nuestro sistema.

Otra función es la capacidad de crear ondas sintéticas para futuros entrenamientos con técnicas de aprendizaje o reconocimiento de patrones de audio, para propósitos relacionados con el habla. De momento se puede crear una onda pura con una intensidad entre [1,-1] para el formato de audio a reproducir y/o grabar, y frecuencia deseadas. Así podríamos crear un LA puro, como el del teléfono, a 440 Hz y con un intensidad 0.5 y reproducirlo o crear un fichero de audio.

Las funciones de localización y los resultados obtenidos se explicarán en el próximo capítulo mediante la síntesis en un solo experimento de las pruebas realizadas, asociadas a las ideas teóricas en las que están basadas. Ni que decir tiene que para llegar a estas conclusiones se invirtió mucho tiempo y esfuerzo y que cuando hablamos de un experimento o de un escenario, éste fue objeto de muchas reflexiones de adaptaciones y de confianza en la idea teórica que nunca se descartan del todo y quizás en un futuro se retomen o se fusionen, y siempre con la convicción de que el fallo era nuestro o de que algo se había entendido o construido mal.

Capítulo 5

Experimentos

A lo largo de este capítulo describiremos los ensayos realizados partiendo de una primera prueba lejana en el tiempo, base fundamental de la inspiración de todo el proyecto y de la decisión de empezar con el oído como un nuevo sentido para la criatura RobEx. Expondremos sus resultados terminando con la estrategia que obtuvo la mejor aproximación en la localización de la señal.

Punto de partida

El punto de partida o experimento inicial de este proyecto tiene lugar en un local de ensayo de un grupo de rock. Tras decidir estudiar el comportamiento humano en la localización de las fuentes, necesitábamos una grabación para tener una idea de como podía funcionar el proceso. La idea fue grabar una voz pronunciando el nombre del robot, *pulguita*, alrededor de una persona a la misma distancia y a un volumen más o menos igual. Esto simulaba un entorno real más o menos parecido a donde se tendría que desenvolver en el futuro, o eso creíamos. El local es un semi-estudio de grabación y dispone de medios técnicos de calidad. La habitación esta acondicionada. Con paredes gruesas de materiales absorbentes, digamos que la reverberación del sonido es poca, no llega a las condiciones de un estudio de grabación profesional pero es mejor que cualquier habitación normal y, desde luego, que el laboratorio de paredes planas y alicatadas que tenemos en la

Universidad.

El sujeto se colocó con dos micrófonos a la altura de sus orejas formando noventa grados y apuntando las membranas a derecha e izquierda respectivamente teniendo, por tanto, la cabeza en medio de los dos receptores. Los micrófonos son AKG C-1000 de patrón polar cardioide e hipercardiode. Se considera, en general, que este tipo establece el mejor equilibrio entre el sonido incidente y el ambiente. Por estas razones se eligieron de entre los disponibles. además es conocido en el "mundillo" como la navaja suiza por su diversidad de usos. El otro sujeto se situó a una distancia de metro y medio, realizando llamadas de 30 en 30 grados, empezando desde los 90 grados a la derecha y terminando en 90 grados a la izquierda. Las señales recogidas por los micrófonos son enviadas a una mesa de mezclas profesional analógica Yamaha MG 24/14fx encargada de la alimentación fantasma y con funciones de puerta de ruido, que está a su vez conectada al PC a través del sistema de grabación digital M-Audio Delta PCI 1010 de gama media-alta con 10 canales E/S. El sonido es procesado con Cubase, uno de los software más conocidos de edición de audio y el resultado es almacenado en ficheros WAV.

El resultado obtenido se muestra en pantalla. El primer oscilograma representa el canal izquierdo y el segundo el derecho.

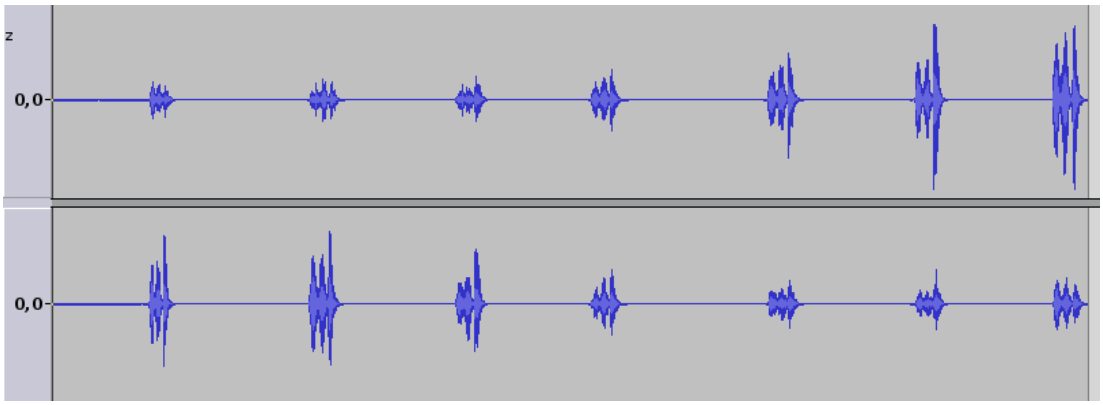

Ilustración 23: Oscilogramas del experimento inicial

Observando la figura de manera general comprobamos efectivamente que la

intensidad del sonido ha ido evolucionando y podemos decir sin dificultad que los tres primeros gritos se dieron desde la derecha, que el cuarto tuvo que ser de frente y que el resto provenía de la izquierda. Cuando se realizó el experimento no había nada implementado en el robot. Se iniciaba la planificación del proyecto y se pretendía calcular de alguna manera la diferencia de intensidades como método de localización. La realidad demostraría que quedaba mucho por resolver. En esa imagen vemos un fragmento del tiempo. Tenemos el proceso completo y podemos ver claramente los estadios entre sonidos y silencios. Ahora, ¿cómo podemos saber cuánto dura un sonido?. Tenemos paquetes de 64 milisegundos con sonido, ¿es suficiente?, ¿cuánto encolamos?, ¿las diferencias entre los micrófonos se mantienen constantes con la distancia? Es decir, un grito a un metro por la izquierda resulta en una diferencia de intensidad, supongamos de dos a favor del micrófono izquierdo, y otra realizada desde el mismo ángulo pero a dos metros, sigue manteniendo la diferencia, ¿decae la intensidad a la mitad por estar al doble de distancia, por ese lado ?, etc.

Partimos con la idea de que todo iba ser más o menos estable pero, como veremos, no fue así. En algunos casos un sonido emitido desde la una posición acababa recibiéndose más alto en la otra posición, o emitido a igual distancia desde el otro lado era percibido con diferencias de intensidad.

Si observamos la figura inicial con atención, podemos observar que los gritos se emitieron el primero a 90 grados a la derecha y, luego, a 60 grados del canal izquierdo. Podemos decir que la variación en el sonido es mínima, sin embargo el canal derecho parece que aumenta. Reflexionando, lo normal o lo que cabría esperar sería que al gritar perpendicular a la membrana receptora la intensidad recogida fuera mayor que al hacerlo desde una posición inclinada. Sin embargo, a la luz de la figura, parece que esto no ocurre. En el canal izquierdo desde este ángulo menos protegido por la cabeza del sujeto, cabría esperar que la intensidad aumentase, sin embargo tampoco ocurre.

Al pasar del centro, parece que el comportamiento es el esperado, aunque si entráramos en detalle, el grito a 60 grados en el canal derecho registró menos intensidad. Es suficiente para entender la problemática a la que nos enfrentamos.

¿Por qué ocurrió esto?, quizás el sujeto llamador no fue capaz de mantener la intensidad en su voz, pero el ser humano si es capaz de ubicarlo y, además, si trabajamos con robot en entornos reales, estas situaciones ocurrirán. Es muy probable que nuestros oídos sean mejores que los micrófonos y no tengan estas variaciones, ¿cuánto de importante es la función de la cabeza en la distorsión y difracción del sonido?.

Pensando en el sistema futuro, si partimos de una serie de datos recogidos por dos sensores, que como todos los dispositivos del mercado no hay ninguno igual, una vez calibrados volvemos al lugar inicial, el local de ensayo, y recordando la enfatización en el hardware utilizado para poner de manifiesto que hablamos de un entorno más ideal, comprobamos la variabilidad en la respuesta ofrecida. Tras un año de trabajo afirmo que el problema fundamental para desarrollar los algoritmos que resuelvan el problema de la localización acústica es que **las medidas son inestables**.

Del silencio al sonido

Uno de los primeros hitos era conseguir averiguar cuándo se producía un sonido que reclamara nuestra atención y obviar el ruido de fondo. De la misma forma que nosotros podemos estar charlando en un entorno en el que hay música puesta o hay ruido producido por una máquina de aire acondicionado, por ejemplo. Si nuestro compañero esta a la izquierda y nos llama, distinguimos su voz sobre el ambiente y atendemos. Notamos que su voz suena por encima de ese ambiente y nos planteamos, ¿hay un umbral de audición para la atención?, ¿el umbral depende del entorno?, así un susurro una mañana de domingo, en buena compañía, en una habitación tranquila consigue nuestra atención y es efectivamente menos intenso que la llamada del

compañero. ¿Cómo conseguir en nuestro robot esa capacidad?, ¿cómo reducir el nivel de ruido?.

Necesitamos escribir un código que estime el nivel medio del ruido y decida, si lo recibido en un instante, con respecto a lo que había, es una llamada de atención o no. A lo largo del desarrollo se consiguieron distintas formas de solventarlo. Una de las primeras y de mis favoritas es utilizar la media acumulativa o running average.

Según la wikipedia "La media acumulativa es un método utilizado para analizar un conjunto de datos en modo de puntos para crear series de promedios. Así las medias móviles son una lista de números en la cual cada uno es el promedio de un subconjunto de los datos originales. Una serie de medias móviles puede ser calculada para cualquier serie temporal. Se usa para demanda estable, sin tendencia ni estacionalidad; suaviza las fluctuaciones de plazos cortos, resaltando así las tendencias o ciclos de plazos largos". Exactamente lo que buscábamos.

Nuestra serie de datos estará compuesta por las medias de los buffers. Para evitar problemas en el sonido y teniendo en cuenta que la onda adopta valores entre -1 y 1, calcularemos la media en valor absoluto. Llegan a razón de unos 15 por segundo y no podemos ir almacenando todos, sumarlos, y dividirlos por el total de muestras y así obtener la media aritmética. Además el sistema siempre ascendería y sinceramente ¿para que queremos saber el nivel de ruido hace por ejemplo 10 segundos?, el pasado no importa. En la "running average" también reiniciamos los valores pero estos enseguida se adaptan, ya que la media del primer paquete es la nueva media acumulativa y no es difícil implementar la opción de no reiniciarse si estamos recibiendo un grito.

Entonces la media acumulativa en un instante es igual a la media anterior más la diferencia entre el dato actual, menos la media acumulada dividido por el numero de elementos.

$$avgC_{(i+1)} = avgC_i + [(dato_{(i+1)} - avgC_i)/i+1]$$

Donde *avgC* es la media acumulativa, *i* el número de elementos y *dato* es valor a añadir a la media acumulativa, que curiosamente en nuestro caso es la media aritmética del paquete de sonido recibido.

Experimento 1: Orientación con la intensidad.

Para realizar estas pruebas contábamos con el robot en su forma básica. El portátil sobre la base y una torreta éstero donde van las dos cámaras, entre ellas no hay ningún material susceptible de causar difracción. RobEx tenía esta forma:

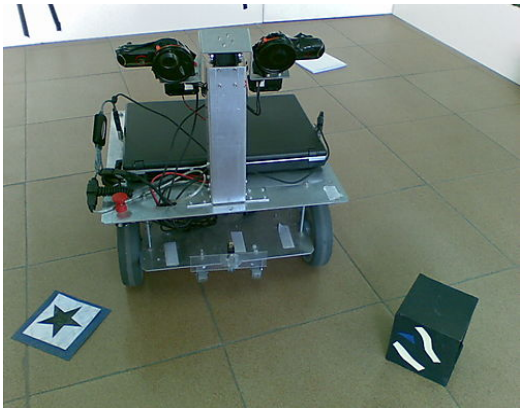

Ilustración 24: Robex con la torreta.

Esta forma es la típica que tiene en el laboratorio para otros usos relacionados con la visión. Así pues, intentamos las primeras soluciones.

Suponiendo que el sonido se recogerá más alto en el micrófono más cercano a al fuente de sonido y, ya implementado, la detección entre el silencio y el sonido, la primera idea es, recibido un sonido en cualquiera de los dos oídos, calculemos la diferencia de intensidad en ese instante. Nuestra unidad mínima de cálculo era el paquete de audio enviado (el buffer de 1024 muestras). Entonces, conociendo la media del paquete simplemente restábamos uno sobre el otro y si el resultado era a favor del primero, el sonido provenía de esa dirección. Se intuía que valores alrededor del cero indicarían que el sonido venía del frente y que valores grandes indicarían

uno u otro lado. Lo ideal sería que la resta se dividiera en niveles lo más estables posibles y que estos fueran de la mano con la posición del sonido. Es decir si emitimos un sonido desde 90 grados a la izquierda con una diferencia x y luego otro desde 45 a la izquierda la diferencia deseada sería $x/2,$ en el micro derecho se habría producido un aumento de la energía y en el izquierdo una disminución que hubiese acortado su diferencia. El problema es que no siempre ocurre esto y la recepción en el micro varía. Estamos en un entorno real donde la reflexión del sonido es impredecible, trabajamos con dispositivos analógicos compuestos de membranas que no tienen porque recibir siempre la misma presión sonora.

Se intentó conseguir un sistema básico que distinguiese tres estados: sonido proveniente de la izquierda, del frente o de la derecha. Tras muchas pruebas se definió un umbral. Si la resta entre el paquete izquierdo y el paquete derecho estaba entre ese umbral negativo y el umbral positivo, el sonido (cercano a 0) había sido recibido con la misma intensidad por ambos, proviniendo del frente, si el sonido era mayor que el umbral negativo, provenía del la derecha y si era mayor que el umbral provenía de la izquierda. Pero esto no era estable y el sonido emitido desde el mismo punto a la izquierda o derecha podía, según el paquete de audio recibido, tener más energía en el micrófono más alejado, dando secuencias de resultados imprecisos como esta:

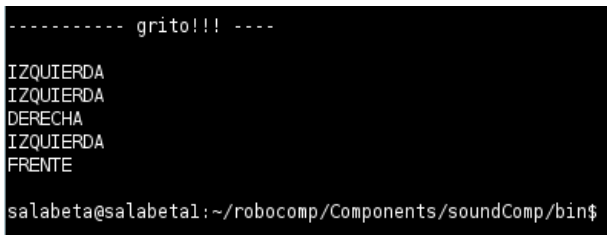

Ilustración 25: Detalle experimento 1

Por supuesto que se calcularon decisiones agrupando paquetes. Al utilizar la misma técnica de encontrar la diferencia producida bien por la media de los n paquetes o por la suma de ambos canales, los resultados tendían a hacerse semejantes

y por tanto la decisión era el frente.

Abandonamos este camino de momento y, sin dejar los métodos basados en la diferencia de intensidades, lo intentamos en el dominio de la frecuencia.

Experimento 2: Orientación mediante el espectro.

Tras las conclusiones del primer experimento y observando durante tantos días como monitorizaba los datos soundComp, sumado la esperanza de buscar de forma sencilla una medida más estable, se optó por utilizar los datos de la transformada de Fourier. Teníamos el software para hacerlo y además creímos ver algo de luz en la forma que solía aparecer en la ventana del espectro de frecuencia.

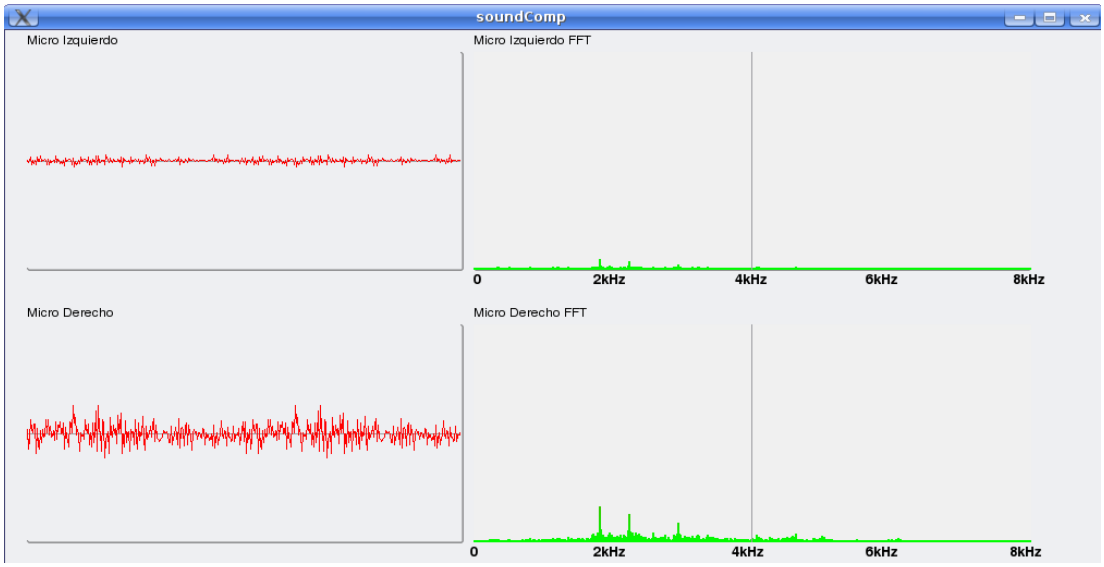

Ilustración 26: Orientación con el espectro de frecuencia.

Atendiendo a la imagen, podemos observar una diferencia notable en el espectro de frecuencias. Así, el de debajo, acorde con la intensidad de la señal, muestra que el sonido que se está recibiendo alrededor de 2kHz es de una magnitud mayor que el primero. Viendo esto, cualquiera de nosotros podría afirmar que el sonido proviene del lado derecho. Una vez más nos disponíamos a analizar los datos

y buscar si había o no homogeneidad en los mismos. Seguíamos buscando una escala entre la diferencia de ambos oídos y la posición ocupada en el espacio por la fuente. Tropezamos de nuevo con errores similares a los primeros. No había nada que lo garantizará estable y se producían casos atípicos en los que aún con la emisión continuada de un sonido, generalmente un pitido entorno a los 1000 Hz, desde una posición fija, se registraban picos en el micrófono más alejado incluso superiores al más cercano. La verdad es que empezaba a ser desesperante.

Recurrimos a consultar a profesores de la titulación de Ingeniería de Telecomunicaciones en Imagen y Sonido que se imparte en la Universidad de Extremadura. Nos recordaron que los micrófonos comerciales tienen una respuesta o están diseñados para otras misiones bastante alejadas de las del oído humano. Puntualizo y añado, que por eso cada instrumento musical se graba con un determinado tipo de micrófono y nosotros no necesitamos distintos tímpanos para disfrutar de un buen concierto. El entorno lo consideraban fundamental y las condiciones reales con el ruido y reverberación asociadas, dificultarían el trabajo de la búsqueda de medidas exactas. Demostraron esta afirmación con un ejemplo sencillo. Al principio de curso ponen una fuente sonora emitiendo perpendicular a la pared y un sonómetro a una determinada distancia entre la pared y la fuente. Observan el valor de la señal y luego alejan el sonómetro de la fuente acercándolo a la pared, una distancia conocida. Al observar la señal registrada el valor aumenta respecto a la primera medida. La explicación está en la onda estacionaria producida por el rebote con la plana pared de azulejos de los laboratorios.

Aún con estas malas expectativas se intentó una nueva solución. La idea era olvidar todo lo que pudiera dar o quitar energía al sonido original. Para ello con la transformada de Fourier podíamos detectar la frecuencia fundamental y quedarnos solo con el valor de ese pico, y determinar la diferencia de energías con la misma frecuencia en el otro canal. Para alcanzar mejores resultados se amplió al intervalo de frecuencias cercanas a la fundamental y se encolaron paquetes para tomar una decisión sobre una muestra mayor. Las predicciones en la orientación mejoraron pero

no adquirieron calidad suficiente. Obviamente, el sistema se comportaba mejor cuando las pruebas se realizaban con pitidos que cuando se hacían con grabaciones de voz.

Experimento 3: Buscando una cabeza.

No es un experimento en sí, pero es la base de otros y, como suele ocurrir en este campo, otra posibilidad más de volver a las primeras soluciones por si pudieran valer o se pueden mejorar. La idea era poner material entre los micrófonos para que absorbieran energia, perturbara y difractara el sonido.

El primer intento consistió en un tubo de PVC, de los convencionales para fontanería, relleno con lana de roca para que obstaculizase la propagación del sonido, A ambos extremos se le colocaron las cámaras asomando el micrófono por cada lado pero sin sobresalir. El resultado se intuye en la figura.

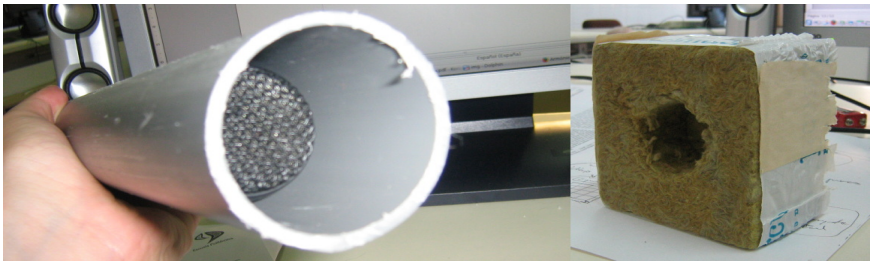

Ilustración 27: Tubo de PVC mostrando el micrófono y la lana de roca utilizada en el relleno.

Los resultados no mejoraron y la intensidad seguía campando a sus anchas.

La solución más elegante y la que se intentará en el futuro es conseguir reproducir a precio económico una cabeza tipo la comentada en el capítulo tres. La Neuman ku100, valorada en 6.200 euros y que por supuesto no es financiable para un proyecto fin de carrera. Si ya fue difícil y costó tiempo y esfuerzo por parte del profesor conseguir la digitalizadora, y menos mal que los micrófonos bastantes más caros eran de mi propiedad, más difícil es pedir la cabeza para grabaciones biaurales.

Para el futuro la idea consiste en rellanar un maniquí convencional con una sustancia de densidad similar al agua e introducir en ella los micrófonos, horadando los orificios de la oreja. Esta pieza no pudo realizarse para este trabajo y sigue en proyecto.

Las últimas pruebas del último experimento se realizaron con los micrófonos dentro de una caja de plástico rellena de arena de gato en bolsas, para no dañarlos, y conectados a la digitalizadora. Una especie de cabeza muy rudimentaria que mejoró algo, aunque no en exceso, la estabilidad de la intensidad de las grabaciones.

Ilustración 28: Detalle Micrófono entre arena.

Buscando nuevas posibilidades conseguimos un par de maniquíes, y valieron para probar. Encitamos las cámaras alrededor de la cabeza, con los micrófonos, en la posición de las orejas. El sujeto quedó de esta guisa:

Ilustración 29: Robex equipado con la cabeza de Maniquí.

Se volvieron a realizar los experimentos, y se colocó de una y otra forma el hardware utilizado. La capacidad de orientación no mejoraba. Todos los experimentos desde este punto se probaban sobre una matriz de sonidos, aprovechando la forma del suelo de baldosas del laboratorio. Se emitía un "*pip*" a intervalos de dos segundos a la misma intensidad y distancias, desde un altavoz de PC. Los pitidos se realizaban desde las siguientes posiciones.

	p00.replay. (-0.9,0.9)			p03.replay (0,0,0,9)			p06.replay (0,9,0,9)
	P30(-0,9, 0,0)			Robot ^---^			p36.replay (0,9 , 0,0

Ilustración 30: matriz de sonidos utilizada en las pruebas

De esta forma podía ir registrando resultados con distintas configuraciones hardware, evaluar y estudiar las grabaciones posteriormente

Como colofón a esta lucha por conseguir una diferencia interaural estable se

visualizan dos detalles de la grabación ocurrida sobre el maniquí desde la posición p30, 90 grados a la izquierda del robot. El “pip” se repetía unas 20 veces. Se han superpuesto los valores de los dos canales de audio registrados a lo largo el tiempo:

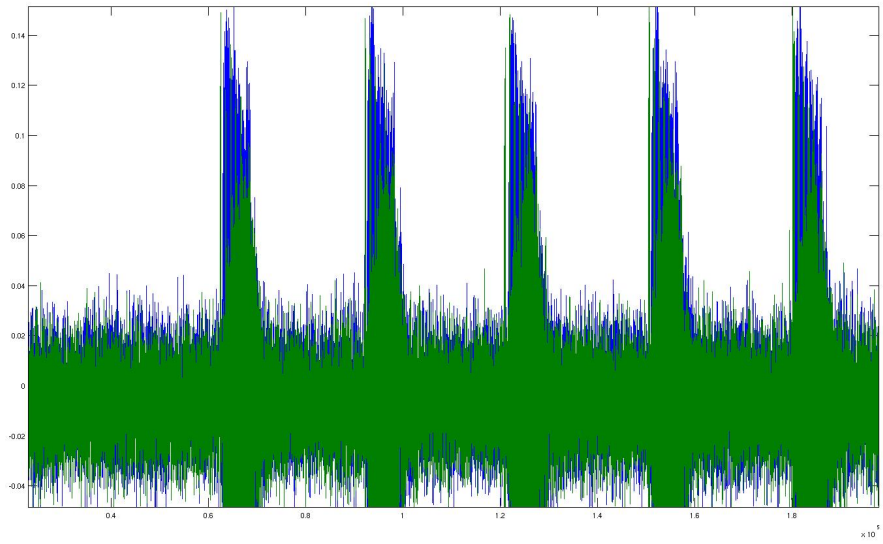

*Ilustración 31: Prueba de sonido desde p30, 90 grados a la izquierda parte **inicial** del fichero.*

El canal izquierdo es de color azul y el canal derecho de color verde. Se observa como siempre va ganando el canal izquierdo, justo lo que deseamos. En la siguiente figura se muestra lo extraño del experimento. Las condiciones no fueron alteradas y el experimento se repitió con resultados análogos igual de increíbles.

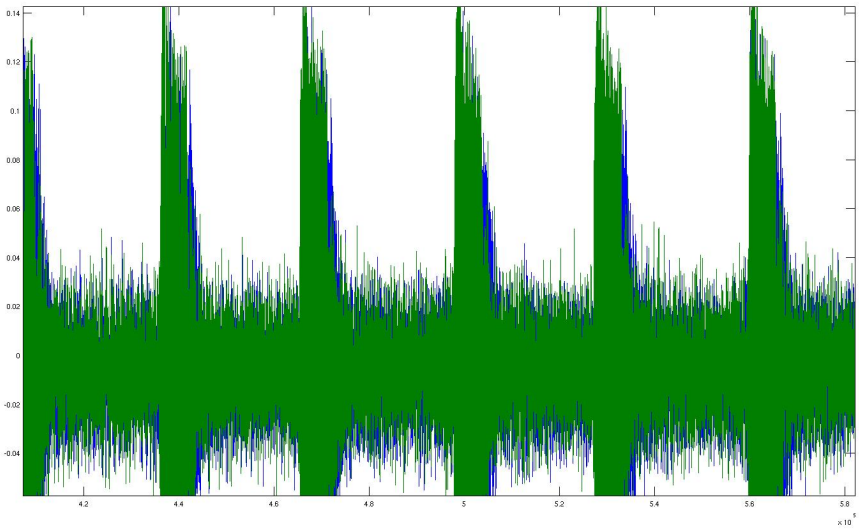

*Ilustración 32: Prueba de sonido desde p30, 90 grados a la izquierda parte **final** del fichero.*

El canal derecho acaba capturando más intensidad que el canal izquierdo, aún encontrándose más lejos y teniendo por medio la cabeza del maniquí que aunque hueca, algo debía interferir. Ante estos datos aparecía la tentación de rendirse. Sin embargo, el aliciente de estos trabajos es la búsqueda de nuevos caminos en los que, al menos, mantener la esperanza.

Ahora empezaría el dominio del tiempo y la prueba de una compleja tesis desarrollada en el Instituto Tecnológico de Georgia[3], basada en un espacio de probabilidad que mezcla la diferencia de tiempos y de energía para estimar la ubicación de la fuente.

Experimento 4: Orientación mediante el tiempo.

En este punto decidimos cambiar por completo el frente de ataque del problema y dejamos las diferencias en intensidad y energía para pasar a la diferencia entre los tiempos de llegadas. A priori, y con lo descrito en el capítulo 3, parecía poco probable que tuviéramos resolución suficiente con el material disponible para estimar con el escaso milisegundo de retardo máximo entre los micrófonos, pero debíamos

demostrarlo y confiábamos que la seudo-cabeza retrasará la llegada de la señal cuando era recibida desde los laterales.

El método a utilizar es la correlación cruzada[17]. Es una medida de similitud entre dos señales frecuentemente usada para encontrar características relevantes. Dada dos señales se va desplazando una sobre la otra, y en el momento en que son más parecidas el valor será mayor. Si una señal fuera idéntica a otra, el valor máximo se alcanzaría para desplazamiento cero. Si una señal está desplazada x posiciones respecto a otra, (porque se recibió más tarde), cuando la correlación cruzada evalue ese desplazamiento, el resultado será el máximo. De esta forma podríamos averiguar cuantas posiciones de diferencia hay cuando se produce el máximo, lo cual nos indicaría, la diferencia de tiempo. Expliquemos el concepto gráficamente y de manera sencilla.[17]

Si tenemos dos secuencias de números, una señal no es más que una secuencia de datos como la de la figura:

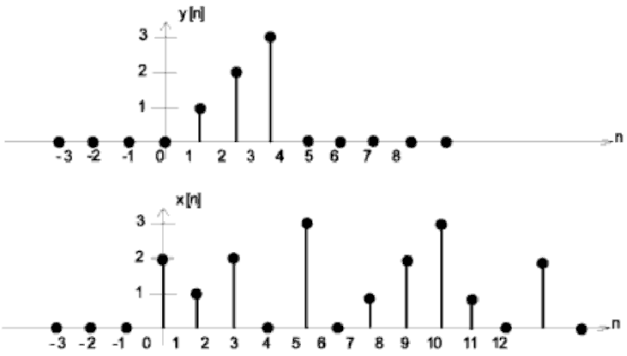

Ilustración 33: Secuencias de las que se va a obtener la correlación cruzada. El eje x representa el tiempo.

Si ahora vamos variando m posiciones, por ejemplo 4, equivale a ir desplazando la secuencia $y[n+m]$, o la secuencia $x[n-m]$, sobre el eje del tiempo y obtenemos:

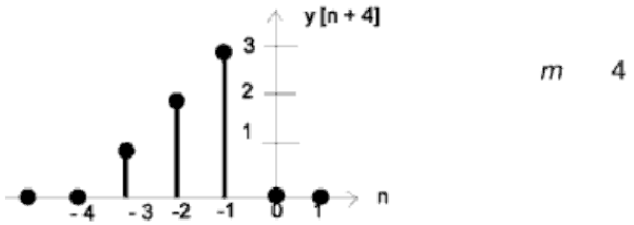

Ilustración 34: Desplazamiento temporal de una de las secuencias.

Si para cada valor de m (de ese instante de tiempo), hacemos la suma de los productos coincidentes, y los representamos:

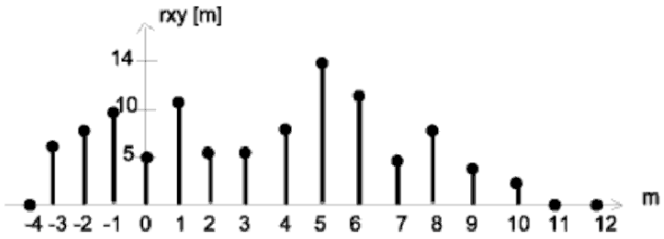

Ilustración 35: Resultado gráfico de la correlación cruzada.

Esto demuestra el grado de igualdad en el tiempo. Se observa que el valor máximo se produce para un desplazamiento de $m=5,$ la primera secuencia $x[n]$ tiene una subsecuencia que comienza en $n=\delta$ que es la más parecida a la secuencia $y[n].$ En la realidad las secuencias de datos suelen ser más grandes en nuestro caso tendrán 1024 muestras de valores comprendidos entre 1 y -1, pero el concepto es extensible.

Las pruebas iniciales de las correlación cruzada se hicieron utilizando la herramienta de generación de ondas sintéticas de soundComp, y se estudiaron los resultados desfasando y/o adelantando hasta un máximo de 90 grados una onda respecto a otra.

Nuestra frecuencia de muestreo es 16000 Hz. Tomamos 16000 muestras en un

segundo. Si generamos una onda sintética de, 1000hz ¿cuántas muestras necesitamos para dibujar una longitud de onda completa?

1000 Hz, quiere decir que se reproduce 1000 veces en un segundo la onda completa. Por tanto en las 16000 muestras de resolución se dibujó 1000 veces la forma de la onda, lo que quiere decir que una longitud de onda se representa en 16 muestras. Por tanto la resolución en grados por muestras es $360/16=22.5$ grados. Así por ejemplo si creáramos dos ondas sintéticas, donde en el instante $t=0$, la primera naciera en 0 grados y la segunda en 90 grados (es decir la segunda ha empezado antes), la correlación cruzada de ambas debía obtener un pico en la muestra 4.

Para una representación más atractiva se utilizan unas capturas en Matlab con las ondas descritas:

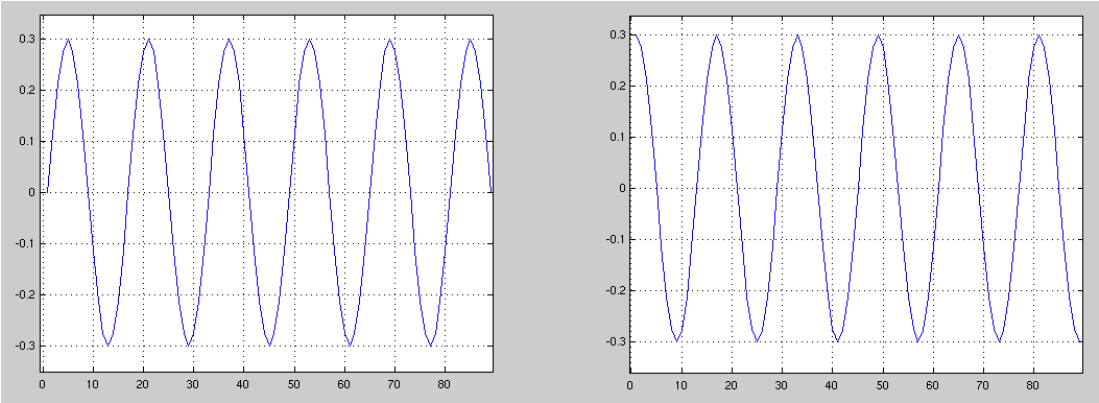

Ilustración 36: Onda de 1000 Hz sin desfase (izquierda) y desfasada 90 grados (derecha).

Se escribió un pequeño código en Matlab que dibuja una onda sinusoidal captada por un micrófono. Los parámetros eran amplitud, frecuencia, frecuencia de muestro, tiempo y desfase en radianes. Después se utilizó la función *xcorr* para realizar la correlación cruzada. El resultado obtenido fue el esperado como detalla la figura:

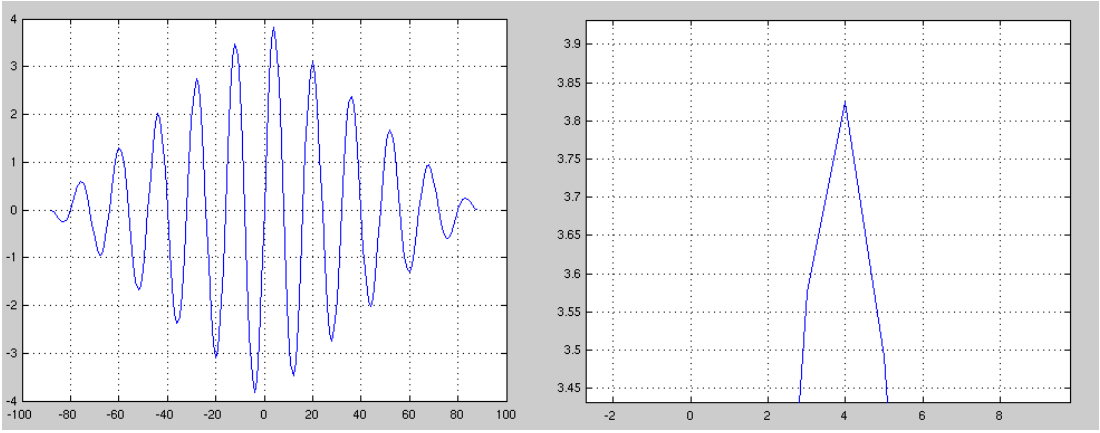

Ilustración 37: Correlación cruzada y detalle con la muestra pico.

Una vez probados teóricamente los conceptos y su acierto, debíamos enfrentarlos a la realidad. Nuestra realidad esta compuesta por paquetes de grabación sobre los cuales calcularíamos la correlación cruzada y posteriormente decidiríamos hacia donde girar. Aún observando como el retraso era manifiesto entre los comienzos de las dos ondas, incluso mayor de lo esperado. Si bien si poníamos dos micrófonos sin nada entre ellos el tiempo que tardaba el frente de onda en llegar de uno a otro no llegaba al milisegundo, cuando interpusimos las distintas seudo-cabezas los tiempos se alargaron superando desde las posiciones más laterales los 20 milisegundos. Pero había dos problemas:

- Las llamadas desde el frente registraban desfases temporales.
- En una captura real su dibujo no tiene que ser necesariamente igual o tan igual como necesitan este tipo de técnicas, para contener la misma información.

Para explicar esto vuelve a ser mucho más ilustrativo recurrir a la imagen.

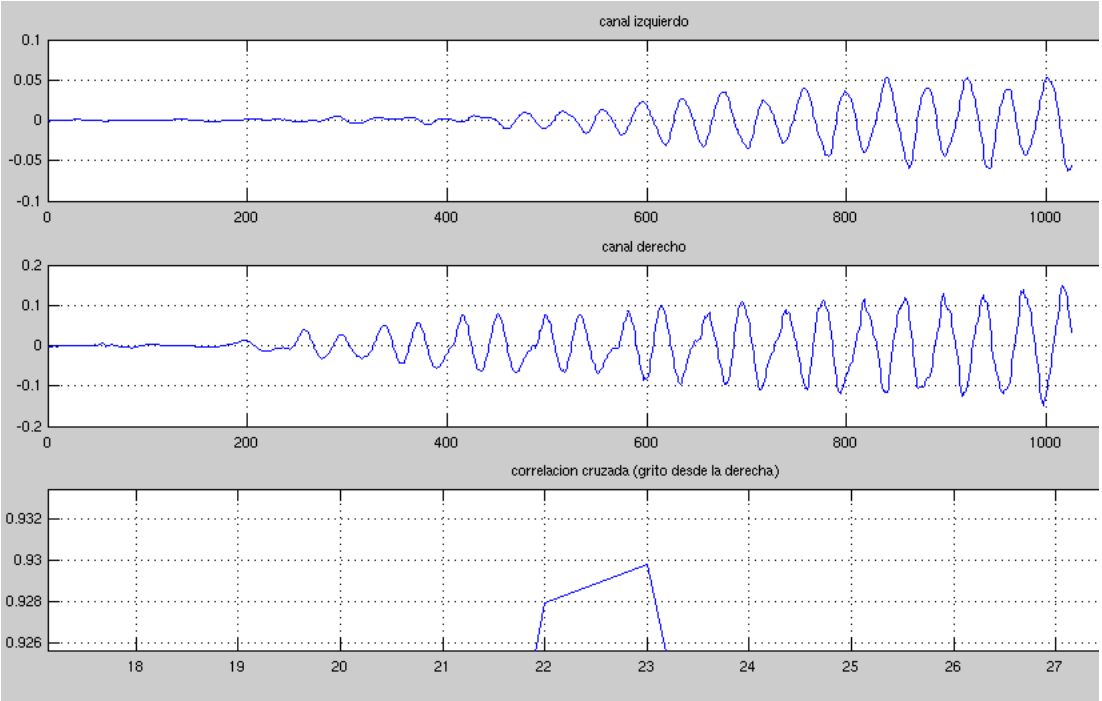

Ilustración 38: Detalle del valor máximo en la correlación cruzada para una onda real, recibida desde 90 grados la derecha.

La captura fue tomada desde 90 grados a la derecha. La correlación tenía una forma parecida a la de las figuras teóricas y se observaba el desplazamiento a la derecha. No se ha incluido su captura por considerar innecesaria sustituyéndola, por el detalle del máximo en la parte inferior de la figura. Buscábamos un orientador que se desplazara a la izquierda o a la derecha independiente de si el sonido provenía del mismo sitio pero en ángulos de signo contrario, teniendo en frente el ángulo 0, a la izquierda negativos y a la derecha positivos. En la primera aproximación se desplazaba 23 muestras a la derecha. Si al repetir el sonido desde el frente se desplazaba 0 podíamos albergar alguna esperanza.

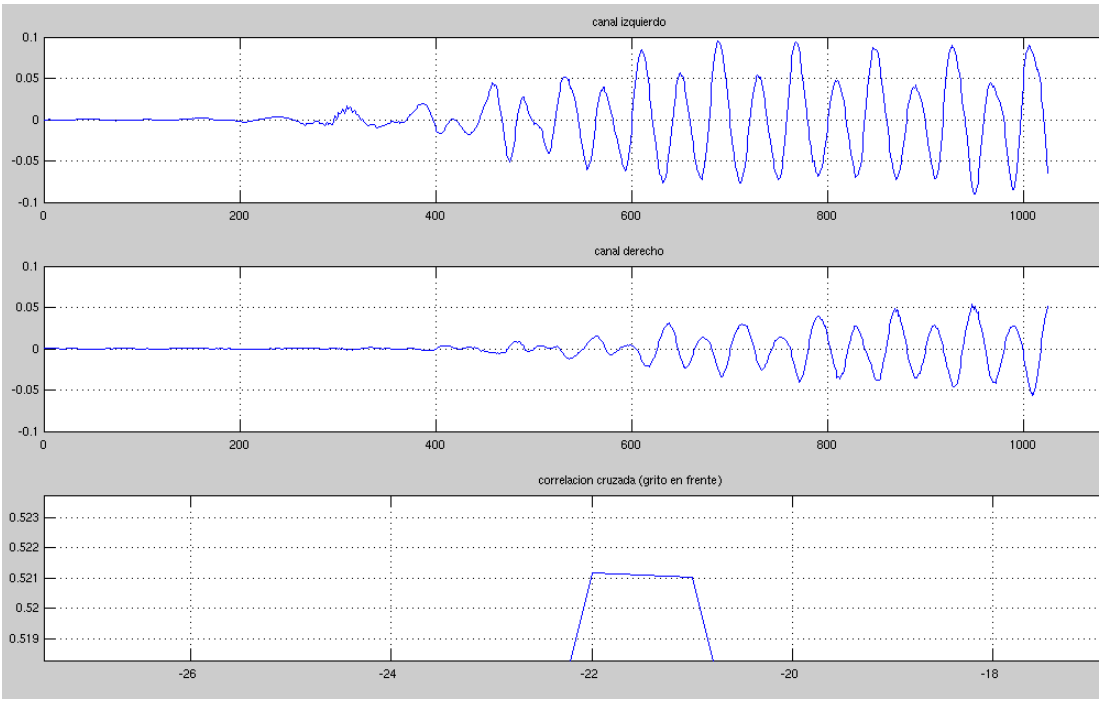

Ilustración 39: Detalle del valor máximo en la correlación cruzada para una onda real, recibida desde en frente, 0 grados.

Lamentablemente no ocurrió. El sonido desde el frente llegó antes al canal izquierdo. Incluso se registró con mayor intensidad (por no olvidar los métodos pasados). La determinación en -22 muestras como el máximo, no se debe achacar a que el método de la correlación cruzada no funciona. Quizás haya que adaptarlo o este más adaptado a condiciones más estables. Al mismo tiempo los dibujos, aún conteniendo el mismo sonido a simple vista no son muy semejantes, ni parece que tengan la misma longitud de onda. Cuando analizamos sonidos complejos que son perturbados por el medio puede haber ligeras variaciones que estropeen métodos tan exactos y teóricos como los propuestos. Nuestro cerebro es en cambio capaz de identificar sin problemas la procedencia del sonido en las mismas condiciones. El único camino posible es seguir estudiándolo e intentar adaptar nuestras soluciones innatas al sistema que tenemos. En verdad, si somos objetivos y no supiéramos que el grito venía desde el frente y miráramos la imagen afirmaríamos que el sonido estaba ubicado a la izquierda, como demuestran el inicio de las perturbaciones en la imagen y como reafirma la correlación.

Experimento 5: Espacio probable.

Este experimento está basado en la tesis de Martin Ericson [3]. Su objetivo es generar una evidencia del entorno basándose en técnicas acústicas, algo demasiado ambicioso para el objeto de este proyecto. Dispone de un robot provisto de cuatro micrófonos, dos delante y dos detrás separados la misma distancia, unos 30 cm, e implementa un algoritmo que realiza operaciones entre cada par. La idea es construir alrededor del robot una especie de tablero imaginario y ubicar, en las distintas celdas, la probabilidad de que la fuente se encuentre en esa posición. Esta es una de las ideas más interesantes descubiertas y es parecido a la idea de los círculos concéntricos[16] , teniendo así distintos umbrales de distancias a las posiciones de las fuentes, pero al mismo tiempo adaptada a la estructura de datos matricial, tan utilizada en programación.

Cuando aborda la localización de la fuente, parte en primer lugar de las diferencias de tiempo estimando una curva de 1 milisegundo, entre la izquierda y la derecha (como ya hemos explicado), y deduciendo que la resolución es muy difícil con los dispositivos actuales. Aplica el método de la correlación cruzada en el dominio de las frecuencias, estimando su energía para cada posición del espacio probable. En definitiva, conociendo la diferencia de tiempo entre la llegada al par de micrófonos y, con la señal capturada por ambos, calcula para esos niveles de energía la probabilidad que tiene la fuente de estar situada en esa casilla. Una vez construido el mapa de probabilidad, estima el ángulo desde el que provenía la fuente.

El pseudocódigo creado por Martin se adjunta en la siguiente figura. Nosotros lo aplicamos tras detectar un sonido. En la tesis sugiere un tamaño del buffer de 2048 muestras. Nosotros hicimos una cola de dos elementos para procesarlo correctamente. La potencia de 2 vuelve a ser clave para el uso de las conocidas transformadas rápidas de Fourier.

/*** the spatial likelihood is estimated pairwise… So for each microphone pair, estimate the chance of there being a sound source at each of the desired locations, and then sum the results across all microphone pairs ***/

1. **for** each pair of microphones [i,j]
2. [f1real,f1imag] = FFT of signali
3. [f2real,f2imag] = FFT of signalj
4. ffreal = f1real.*f2real + f1imag.*f2imag /** The operator ".*" indicates an element-wise multiplication of arrays **/
5. ffimag = f2imag .*f1real - f2real.*f1imag

/*** estimate the weights for the phase transform ***/

6. $G = \frac{1}{magnitude(f1_{real}, f1_{imag}) .* magnitude(f2_{real}, f2_{imag})}$
7. **for** cell [a,b] in the array

/*** estimate the time delay for this microphone pair ***/

8. [x,y] = real coordinates of cell [a,b], relative to the array center
9. $d_1 = \sqrt{(x - mic\_pose_i.x)^2 + (y - mic\_pose_i.y)^2 + +(H - mic\_pose_i.z)^2}$
10. $d_2 = \sqrt{(x - mic\_pose_j.x)^2 + (y - mic\_pose_j.y)^2 + +(H - mic\_pose_j.z)^2}$
11. $TD = (d_1 - d_2)/343 - m/s;$

/*** build generalized cross correlation variables ***/

12. $X_{real} = G.*(\cos(-w*TD).*ff_{real} - \sin(-w*TD).*ff_{imag})$
13. $X_{imag} = G.*(\cos(-w*TD).*ff_{imag} - \sin(-w*TD).*ff_{real})$

/*** perform trapezoidal integration across the half sample size***/

14. $S_{real} = 0.5*\sum_{i=2}^{1024}[(w[i] - w[i-1])*(X_{real}[i] - X_{real}[i-1])]$
15. $S_{imag} = 0.5*\sum_{i=2}^{1024}[(w[i] - w[i-1])*(X_{real}[i] - X_{real}[i-1])]$
16. $SpLikelihood[a][b] += magnitude(S_{real}, S_{imag})$
17. **end for**
18. **end for**

1. P = zeros[360];
2. **for** ang = 0: $ang\_increment$:2π
3. $num = 0$
4. $den = 0$
5. **for** each cell [a,b] in SpLikelihood
6. theta = Th[a][b] – ang, normalized to -π<=theta<π

$W = \frac{e^{-theta^2/2\sigma^2}}{\sigma\sqrt{2\pi}}$

7.
8. num+=W*SpLikelihood[a][b]
9. $den += W$
10. **end for**

$P[ang] = \frac{num}{den}$

11.
12. **end for**
13. $Best\_Angle$ = ang, where $P[ang]$ is maximized

Ilustración 40: Pseudocódigo propuesto por Martin Ericcson.

Se crearon las estructuras necesarias para su implementación en C++ así como todas las operaciones necesarias. La estimación del ángulo, teniendo en cuenta que está diseñado para girar 360 grados y que el robot de la tesis tiene cuatro micrófonos, en general nunca obtuvo un resultado coherente. Sin embargo la matriz se comportaba algo más correcta en relación a las pruebas, aunque la estimación no fuera exacta. Los ángulos, en cambio, eran impredecibles.

Primeramente se probó el algoritmo de forma sintética en los casos que podíamos reproducir cercanos a la realidad (aunque alejados) puesto que no podíamos saber exactamente el nivel de las señales recogidas. Realizamos las pruebas desde los laterales y las posiciones delante y detrás, y enviamos una onda desde el frente, con igual intensidad, como se supone que sería recibida, y en los resultados aparecían máximos en este eje. Si enviamos una señal desde un lateral, y al otro enviamos un silencio, aparecen máximos entonces el eje horizontal.

El problema ocurrió al probarlo en los casos reales, donde los resultados se volvieron inestables, obteniendo máximos a la izquierda y posteriormente a la derecha, como muestra la figura:

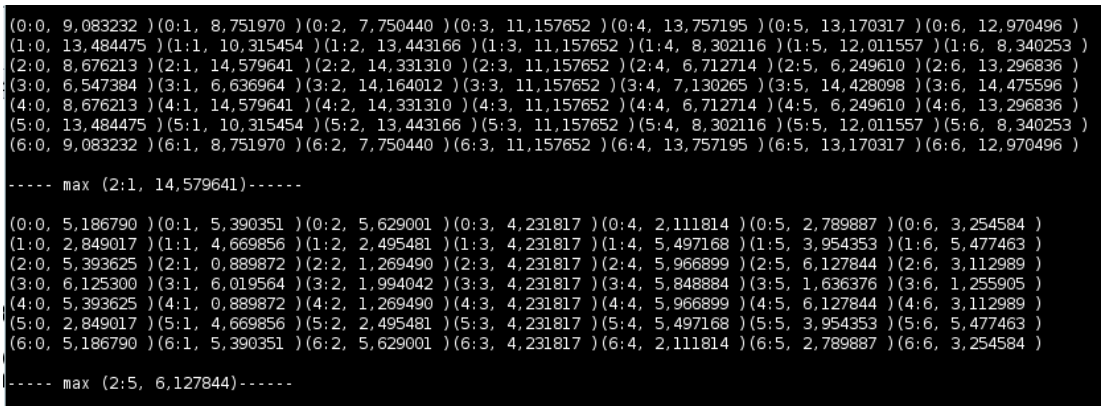

Ilustración 41: Matriz de espacio probable.

Finalmente, no se utilizó este método aunque la idea del espacio probable es prometedora y se seguirá estudiando en el futuro. Es una forma interesante de determinar el mapa sonoro de alrededor y de precisar la localización. La configuración distinta de nuestro robot respecto al suyo, es una de las causas que puede haber influido de manera negativa en la capacidad de ubicación. O quizás en el texto no estén expuestos todos los secretos de la tesis debido a otros intereses.

Experimento 6: Vuelta a la diferencia de intensidad.

Tras los distintos esfuerzos por conseguir buenos resultados con las pruebas realizadas, se decidió volver al origen. Volver a la orientación mediante la intensidad en el dominio del tiempo. Las razones principales parten de la intuición y la reflexión del propio comportamiento humano. Si bien no podemos afirmar tan claramente cuando oímos por un oído primero que por otro, sí somos capaces de diferenciar los niveles, sobre todo en sonidos prolongados. La observación de multitud de oscilogramas sugería siempre una orientación por la diferencia de altura alcanzada en ambos canales. Al verlo no sabíamos decir cuanto, pero sí intuir desde dónde venía el

sonido. Otra razón es debida al efecto Hass y a nuestra imagen mental de un sonido formado a la izquierda cuando tiene más nivel. Así, si estamos escuchando música, podemos ser capaces de oír la guitarra por la izquierda y localizarla en un punto intermedio entre totalmente a la izquierda y el frente, si está lateralizada (*paneada)* con un nivel del 70% al 30%, de izquierda a derecha. Sabiendo que la resolución en la diferencia de intensidades para un ordenador (y por ende para nuestro sistema) es más fácilmente medible y cuantificable, siempre y cuando encontremos una escala estable. Otra razón, que apoya a la anterior, es que la precisión en el tiempo, al menos con el hardware disponible, es prácticamente imposible ya que debemos decidir, en el caso de tener los micrófonos al aire (sin seudo-cabeza), en unos tiempos máximos de 1 ms, que apenas nos dan un par de decenas de muestras para tomar una decisión. En en el caso de tener la seudo-cabeza, los retrasos no logran superar el tamaño del paquete de sonido recibido.

Por otro lado, en los experimentos comprobamos que cuando utilizamos las configuraciones hardware de las dos cámaras, no podíamos asegurar que la lectura de los USB se producía de manera síncrona, No porque nuestra llamada fuera asíncrona, sino porque no sabemos como planifica las lecturas el puerto el sistema operativo. Además esta extraña sensación de que a veces no era perfectamente síncrono, coincidió con pruebas de convergencia en las imágenes por parte del equipo de Robolab, para las cuales se utiliza el mismo modelo de cámaras. En ellas se podían contemplar pequeños saltos producidos entre el vídeo de ambos canales. Por lo cual se optó por no volver a esta configuración y utilizar la digitalizadora externa.

El problema por el cuál abandonamos la orientación por intensidad, es la imposibilidad de obtener resultados estables. Partíamos de que los micrófonos no iban a darnos exactamente lo mismos valores para un mismo sonido emitido desde el mismo punto a la misma intensidad y a la misma distancia, ni de un día para otro, ni si quiera en la misma prueba y, suponíamos que la cabeza, aunque no siempre, mitigaba los efectos rebotes del sonido. A parte, el trabajo y el tiempo dedicados, nos habían mostrado empíricamente que si disponíamos de colas donde almacenar los

paquetes y los procesamos después, en conjunto, los resultados mejoraban. Es decir, si para decidir sobre la posición de un sonido esperábamos a oírlo durante más tiempo, la localización se volvía más exacta. Esto también ocurre a los humanos, Por ejemplo, al escuchar un silbido de un par de segundos, cuando termina solemos girar la cabeza hacía el lugar de la fuente sonora. En nuestro caso utilizamos colas por valores de medio segundo o un segundo aproximadamente. Aunque en los trabajos leídos de otros investigadores en los que se ha abordado de manera computacional el problema de la orientación espacial, como el del experimento anterior, se suelen utilizar paquete pequeños, en torno a las cien milisegundos.

El punto de partida seguía siendo una vez disparada la alerta por sonido, para lo cuál se había conseguido bastante fiabilidad por distintos caminos, al procesar el audio para conseguir una dirección. La idea que buscábamos era conseguir una medida que fuera uniforme, lo más relativo posible a los valores recibidos y alejado de umbrales predefinidos. Sin olvidar que debía ser independiente de los valores positivos y negativos.

El primer punto del algoritmo era buscar “algo” que diera estabilidad. Necesitábamos una medida que premiara las posiciones de la onda alejadas del cero y que al mismo tiempo cuando andaban cercanas al cero las hiciera tender a él. La idea era buscar algo que dibujara un contorno estable sobre la onda y eso, cuando observamos un oscilograma, se parece a dibujar almenas mentalmente. Gráficamente:

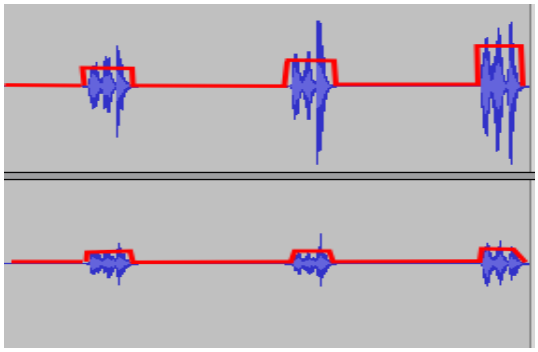

Ilustración 42: Idea de contorno, sobre la onda.

Inicialmente la medida elegida fue la RMS (root mean square), la cuál es una medida muy utilizada para calcular la media de amplitud sobre el tiempo en las ondas sinusoidales. Esta medida suele ser utilizada en la comparación de niveles entre amplificadores. La función que describe este concepto es como sigue:

$$X_{RMS} = \frac{\sqrt{\left(x_1^{۲} + x_2^2 + x_3^2 + \cdots + x_n^2\right)}}{\sqrt{n}}$$

El objetivo era, una vez alcanzado ese pico estable, restarlo respecto al otro. De esta forma la diferencia máxima teórica se produciría a 90 grados, donde el sonido sería recibido solo por un oído. La resta entre un canal y otro nos daría la máxima intensidad que sería igual al canal por donde se estaba recibiendo el sonido. Es decir, una vez calculada la diferencia, observaríamos cuánto se acerca al valor el canal que recibió la señal más alta, y así determinar los grados del giro.

Al aplicar el cálculo de la RMS y, dado que aveces el sonido recibido no se aleja de la primera décima, ocurría que los valores siempre eran muy cercanos a cero, con lo cual la resta era tan pequeña que hacia imposible la subdivisión en intervalos para discernir cuanto había que girar. Con un numerador escaso, y además divido por el denominador que es igual a la raiz cuadrda del tamaño del buffer $\sqrt{۱۰۲۴} = 32$, el resultado era inservible para nuestro caso. El asunto desprendido de la observación, era que el numerador si que ofrecía unas condiciones más favorables para nuestras cuentas. Este proceso por si solo en un vector, la raíz cuadrada de la suma al

cuadrado de sus elementos, que es conocido en matemáticas como la norma Euclídea [17] y representa la distancia de un punto al origen del sistema de referencia.

Así pues se eligió este camino. Una vez disparada la alerta de sonido se calculaba la norma euclídea y se aprovechaba la cola implementada, para almacenar un número determinado de resultados de esa norma, a las cuales se les aplicaba una media aritmética convencional con objeto de suavizar los posibles picos o valles.

Imaginemos el robot con el ángulo de giro 0 grados al frente, entendiendo como 90 grados negativos el giro máximo a la izquierda y 90 grados positivos el giro máximo a la derecha. La resta de intensidad siempre será Derecha menos Izquierda para obtener el signo correcto.

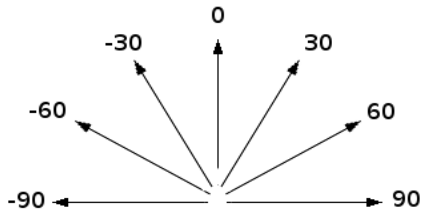

Ilustración 43: Sectores de orientación.

Suponemos por tanto cuatro límites para los tres intervalos por cada lado. Entonces si el valor está entre las líneas, pertenece a ese intervalo y representa un giro de 30 grados. Ya que dicho intervalo se calcula dividiendo el máximo giro entre los intervalos y, debido a que debemos tener en cuenta que el cero, es decir no girar, debe realizarse cuando estemos cerca del cero, tanto en valores positivos como negativos.

Se eligió el sistema de intervalos ante la imposibilidad de crear una función continua debido a la naturaleza de los datos. Así era preferible que un objeto situado en frente a los ojos de un observador, aunque estuviera un poco desplazado a derecha o izquierda, fuera localizado como en frente a que se estimara como desplazado un ángulo superior.

Los resultados en este experimento son los mejores y más estables, Además parte de la idea que si un sonido viene desde la izquierda, lo primordial es iniciar bien el giro. Luego, seguramente, como en la vida real, el sonido se vuelva a producir, es decir nos vuelvan a llamar, y volvamos a girar hacia el mismo sitio o simplemente sigamos de frente al encuentro con el sonido. El problema es a la hora de obtener los 90 grados ya que las diferencias tienen que ser máximas y normalmente un giro de 90 grados se queda en 60 grados. Si aumentamos la resolución en intervalos, por ejemplo poniendo 10 y obteniendo una resolución de 10 grados, el calculo de 90 grados en algunos impulsos se coloca en 70º y en otros cae a los 50º, y en los recibidos de frente puede desplazarlos a mayores de 10º.

Se muestra la localización de la fuente alrededor del robot. La prueba como siempre consistió en la toma de datos desde las posiciones descritas en la figura 30, Las muestras se hacen en tiempo real, es decir, apenas pasan cinco minutos desde que se inicia la prueba. Así si en algún caso la recepción no fue tan buena como se esperaba o si, en general, como se aprecia en la figura los niveles son bajos, no se repite, ya que si no falsearíamos el intento de simular las condiciones reales. El sonido emitido estaba a 90 cm, dos veces el tamaño del robot, y consistía en una grabación mono-fónica de una voz humana pronunciando su nombre, y grabada (escuchada) por el mismo. Es decir, consistía en la reproducción de un fichero wav, grabado con el propio sistema. Para su representación, se optó una vez más, por utilizar Matlab. Se adjunta captura del fichero de audio y resultados.

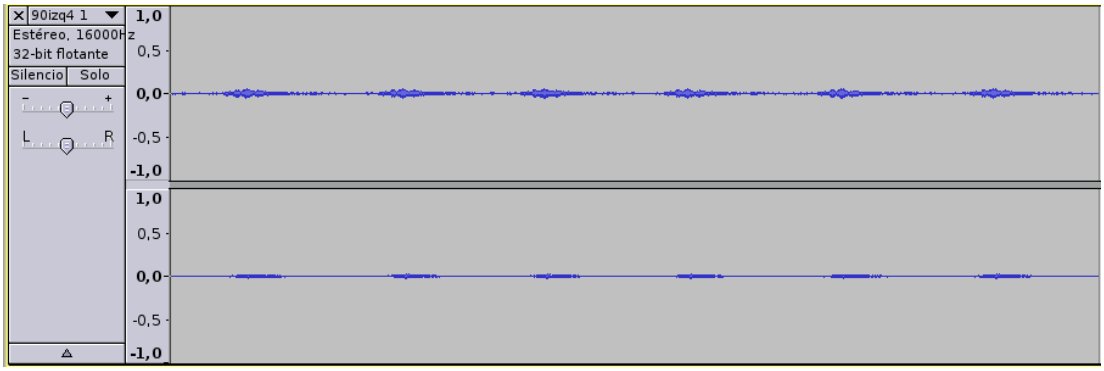

Ilustración 44: Señal recibida desde 90 grados a la izquierda. Figura mostrada con audacity, quien dibuja en el nivel superior el canal izquierdo.

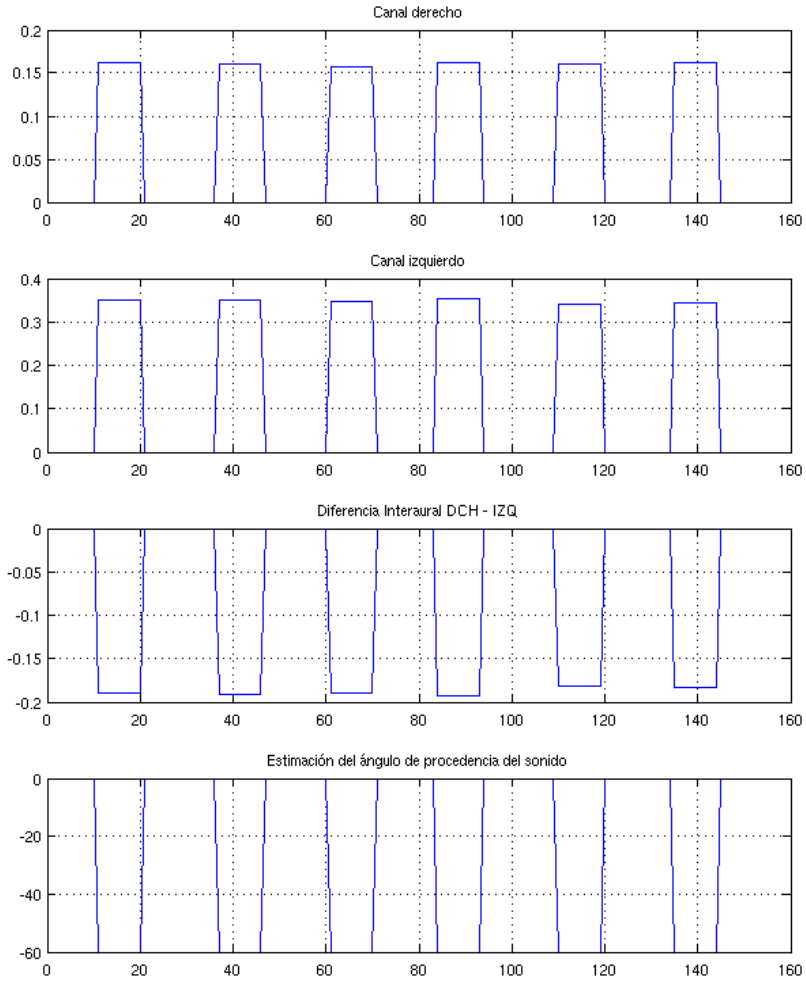

Ilustración 45: Operaciones y resultado de la estimación en grados desde 90 a la izquierda.

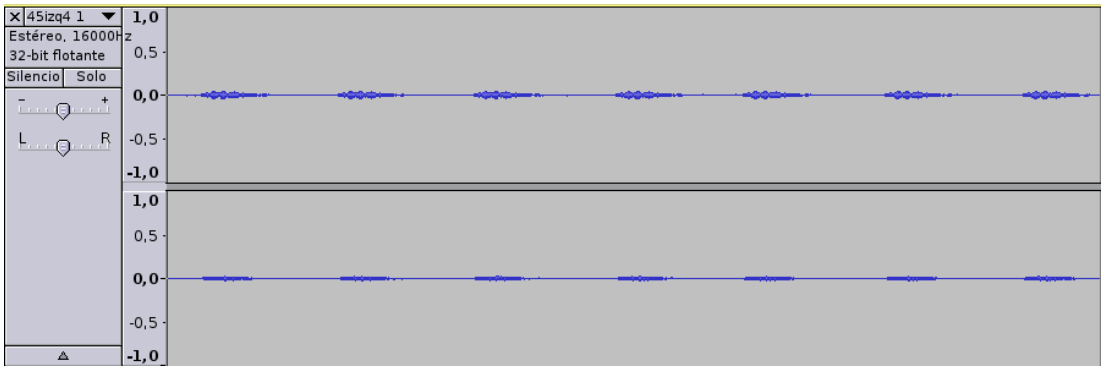

Ilustración 46: Señal recibida desde 45 grados a la izquierda.

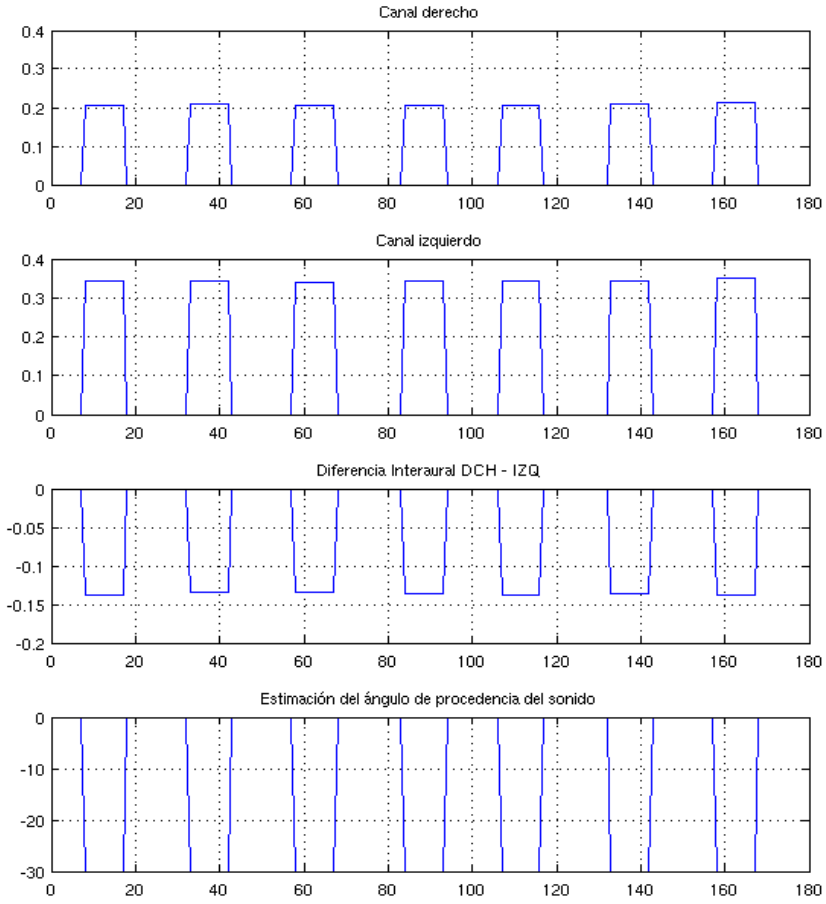

Ilustración 47: Operaciones y resultados de la estimación en grados desde 45 a la izquierda.

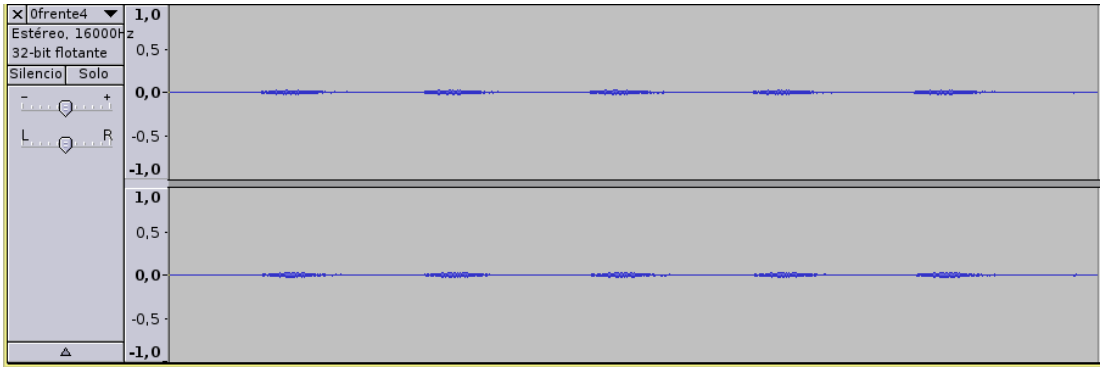

Ilustración 48: Señal recibida desde 0 grados, desde el frente.

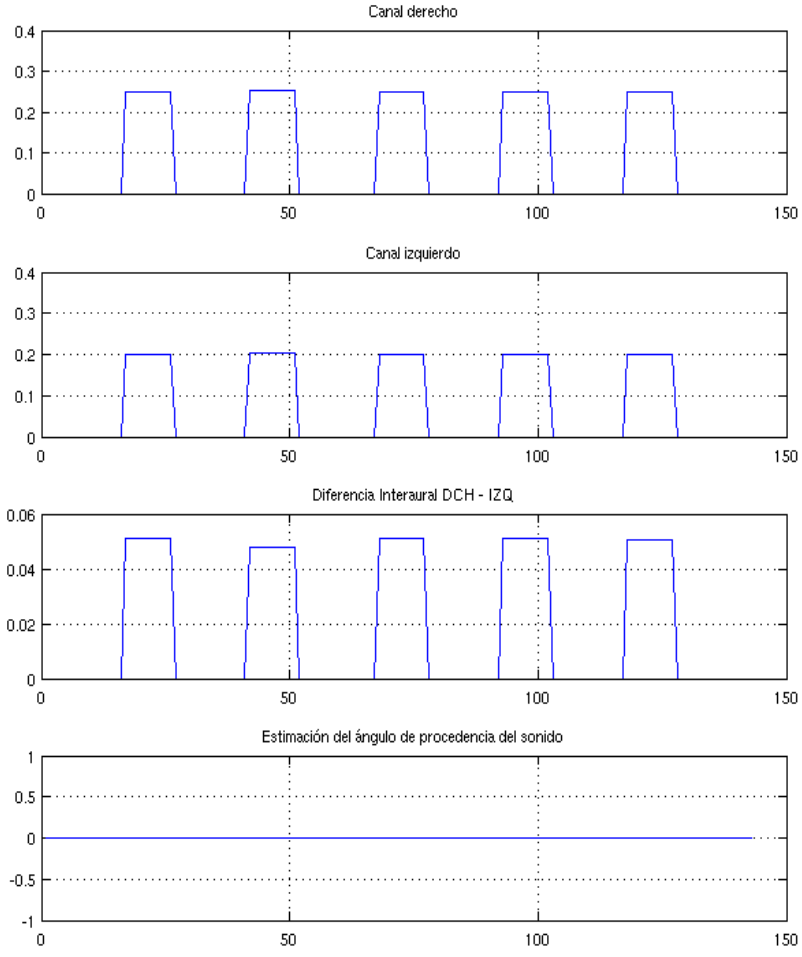

Ilustración 49: Operaciones y resultado de la estimación en grados desde 0. Enfrente.

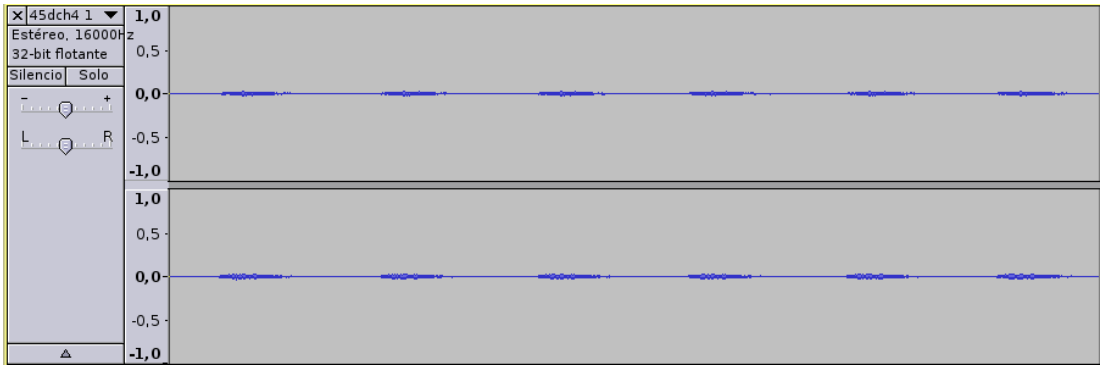

Ilustración 50: Señal recibida desde 45 grados a la derecha.

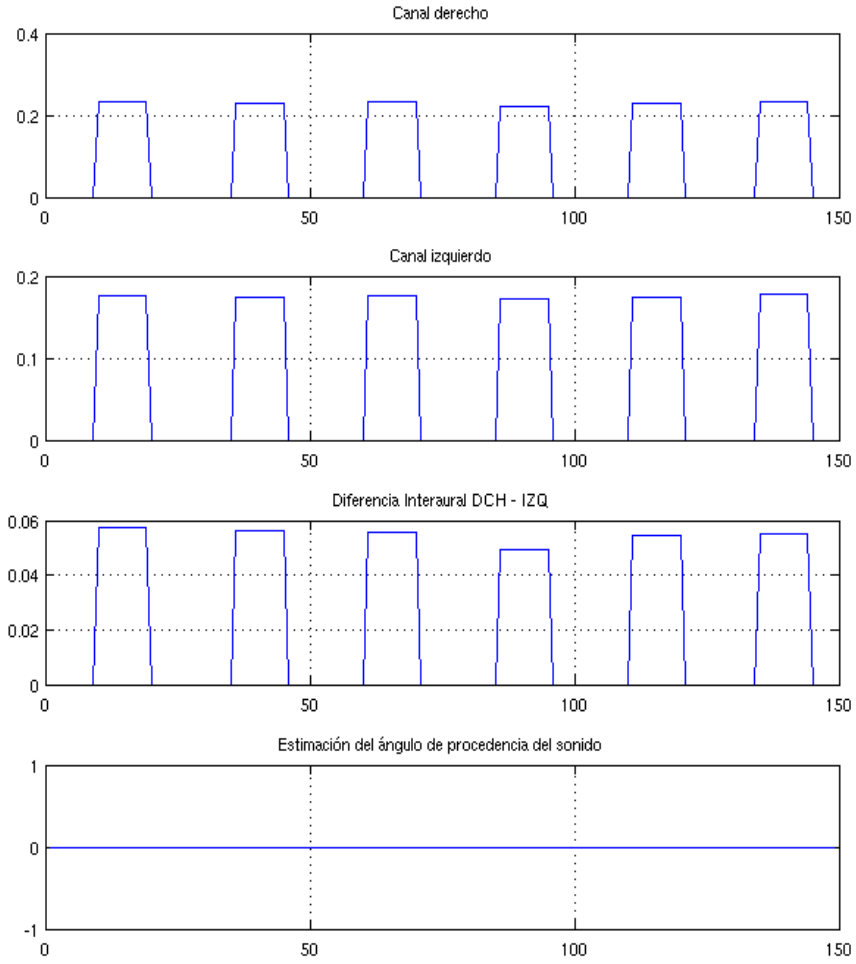

Ilustración 51: Operaciones y resultado de la estimación en grados desde 45 a la derecha.

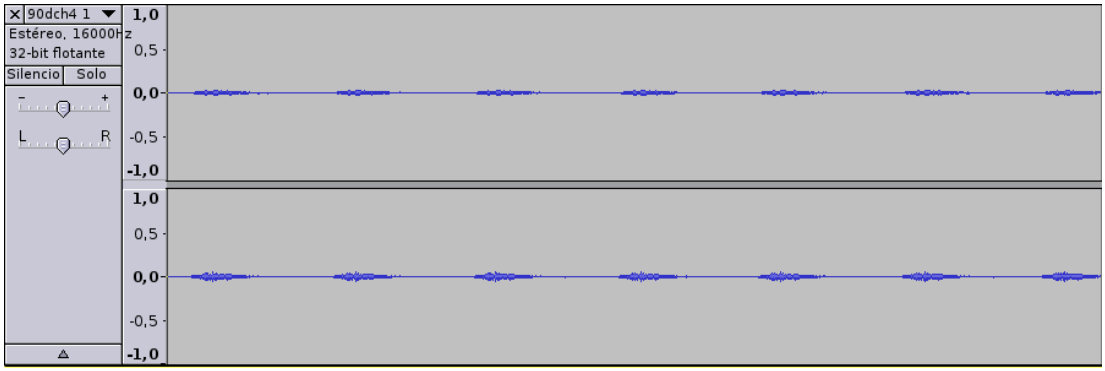

Ilustración 52: Señal recibida desde 90 grados a la derecha.

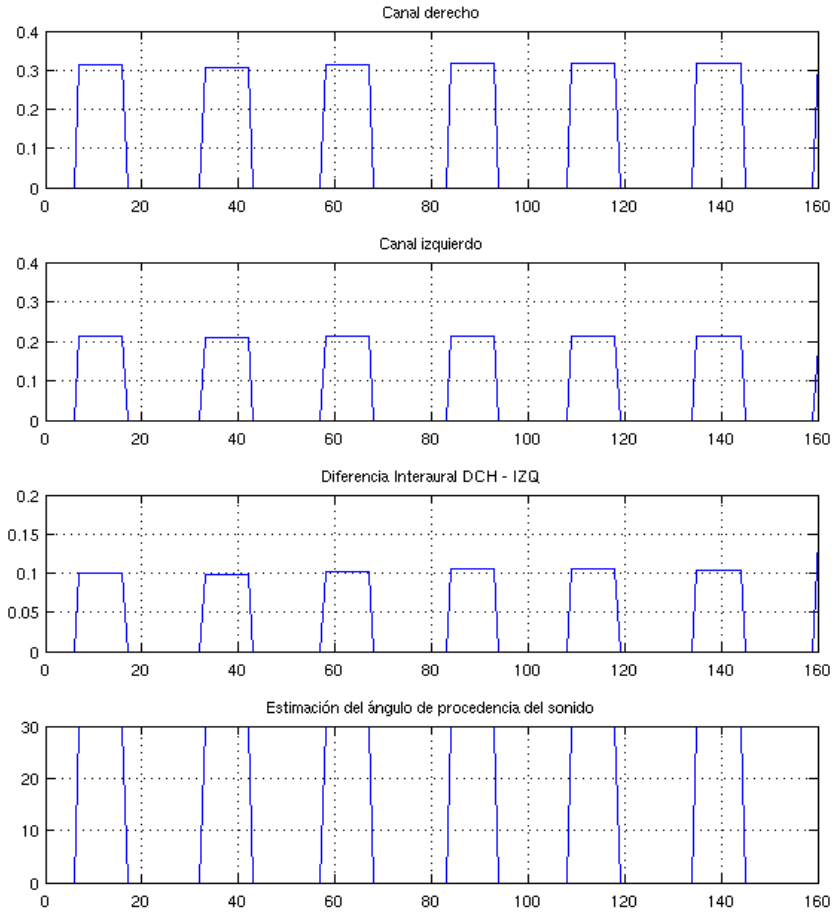

Ilustración 53: Operaciones y resultado de la estimación en grados desde 90 a la derecha.

Observando los resultados y recordando que los intervalos eran de 30 grados, el sistema responde de forma esperada por la parte izquierda. En cambio, cuando llegamos a la parte derecha, el micrófono recoge peor los datos y estima en menos cantidad los grados. Este es el problema básico y conocido: lo inestable de la medidas. Si extrapolamos al mundo humano, si alguien nos llamara desde una ubicación y no nos enteráramos gritaría más fuerte. Si procediéramos en el ejemplo de igual forma, el algoritmo también funcionaría. La idea principal de conseguir una igualdad, con esa especie de almenas, ofrece buenos resultados. Cabe reseñar que al menos no se detectan posiciones drásticamente erróneas. Esto significa que si el giro viene desde la derecha giramos hacia él o no nos movemos, pero al menos no giramos a la izquierda. La secuencia mostrada en las capturas anteriores es una secuencia difícil y elegida a propósito. Los niveles registrados fueron bajos y, por tanto, complican la capacidad de estimación. Como colofón final se muestra el algoritmo trabajando con el fichero inicial. Aquel grabado en el local de ensayo en el que la voz del que suscribe se movía alrededor de los micrófonos colocados a la altura de las orejas, perpendiculares a la cabeza y se puede observar cómo la tendencia a averiguar la ubicación se mantiene, si bien no despunta en los extremos y también vuelve a suceder un aumento susceptible en la intensidad para el segundo grito, debido tal vez a la fuerza con la que se pronunció. Cabe reseñar que cuando el sonido provino del frente el resultado fue correcto. De igual forma que al pasar de la mitad en la secuencia de llamadas y volver a recibir el sonido desde la parte izquierda, la tendencia evoluciona conforme a la resolución, pasando de 0, a 30 y 60 grados respectivamente. Seguramente los micrófonos siguen la ley básica de que no hay dos dispositivos iguales o, quizás, RobEx es “duro de oído” por la parte derecha.

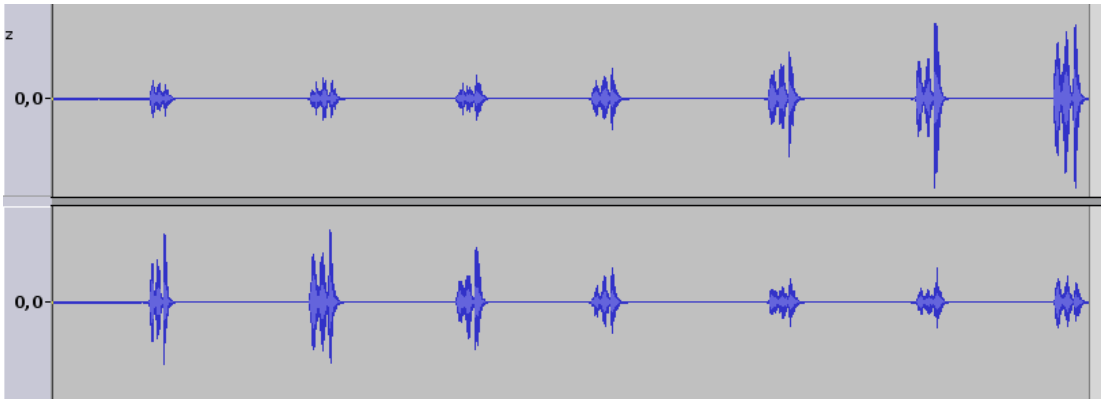

Ilustración 54: Representación sonido registrado en el experimento inicial.

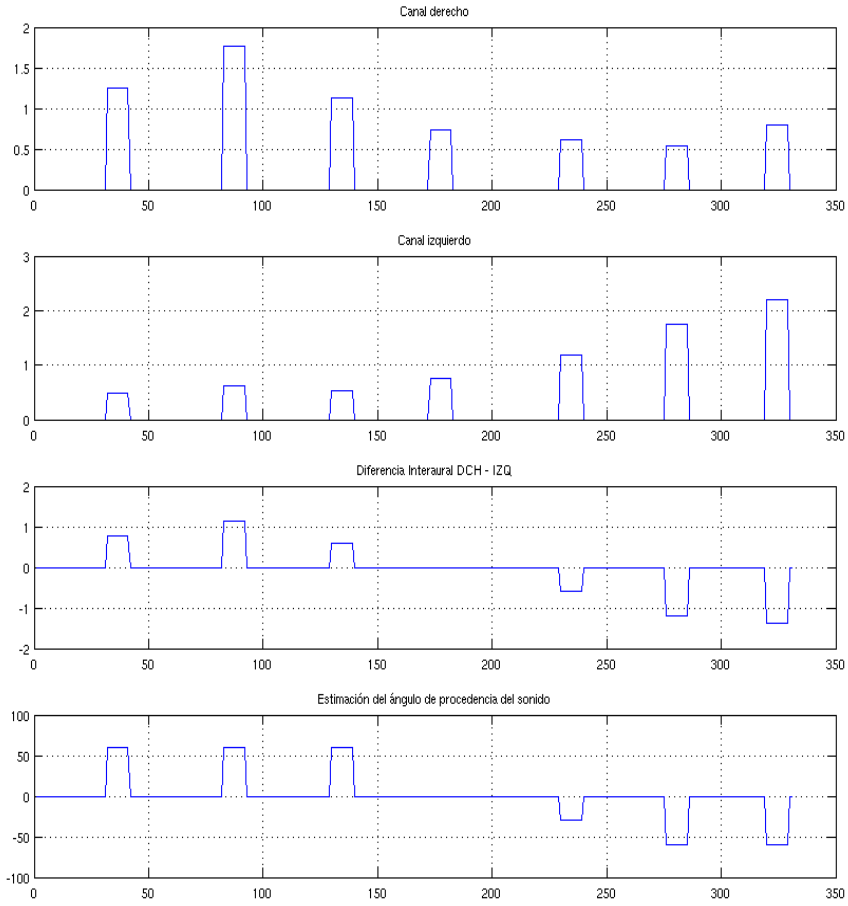

Ilustración 55: Operaciones y resultado en la estimación de los distintos ángulos, para la llamada alrededor del sujeto.

Capítulo 6

Conclusión y trabajos futuros.

De forma general el proyecto fin de carrera ha supuesto una experiencia gratificante en la introducción al mundo científico. Abordar un trabajo de esta envergadura ha sido una lección importante en cuanto al aprendizaje en aspectos como la planificación y metodología de desarrollo, la estimación de hitos, las relaciones entre el coste y el tiempo de lo logros y el conocimiento derivado de los fracasos. Ha supuesto también una lección en las formas de argumentar las ideas vertidas apoyándose en teorías o demostrando empíricamente si pueden ser factibles, y volviendo a defender, el porque sí o el porque no, de la manera más objetiva. Se ha aprendido a analizar las situaciones a priori, de manera fría, y antes de invertir un minuto en la primera idea que aparece en el horizonte, pensar con hipótesis reales los posibles resultados de ese camino. Se descubre el uso de herramientas externas para apoyar esas conjeturas iniciales con las que poder descartar, al menos, los pensamientos erróneos. Aparece una forma nueva de pensar que consiste en adaptar las bases teóricas a los materiales existentes y a descubrir, por tanto, cómo las ideas fuera del papel no son tan perfectas. La estrategia de utilizar métodos de grabación y reproducción donde poder ensayar y modificar los algoritmos propuestos y así conferir un poco más de estabilidad a las condiciones, se vuelve imprescindible en entornos robóticos.

En el plano concerniente al objetivo de la localización esteroacústica sobre robot móviles, como norma general, se advierte la necesidad de asegurar un mínimo grado de exactitud a las medidas sensoriales. El problema fundamental es que no siempre el sistema se comporta como nosotros. Incluso nada asegura que nosotros tengamos una alta fiabilidad. Lo que sí demuestra es que la conjugación de las tres opciones sobre la señal del oído con las que trabaja el cerebro, la diferencia de intensidad, el retardo en el tiempo y la descomposición en frecuencias, apoyadas en unas capacidades preconcebidas debidas al aprendizaje, le da un poder de decisión y precisión al ser humano que dista mucho de este sentido primigenio de la audición que nace a partir de este proyecto.

Para el que escribe la cabeza es fundamental en nuestra habilidad de ubicación y desde ya, se retomará la creación de la imitación más factible y real. La cabeza retrasa y absorbe el sonido dotándonos de tiempo y diferencia para analizarlo. Al mismo tiempo surgen muchas expectativas de futuro y parece necesario aprender técnicas para obtener información de las ondas que no llegan directamente sino rebotando, para imitar la capacidad de ir bajando el nivel de intensidad del oído más lejano a la fuente. Se plantean cuestiones como, ¿el volumen desciende o no se le presta atención?, ¿que es no prestar atención y cómo se puede traducir a la computación?.

Y una vez conseguida la ubicación, se plantean serias cuestiones todavía sin respuesta. La más obvia, incluso con una cabeza perfecta ¿cómo averiguaremos si el sonido está detrás nuestra? Esa pregunta sigue abierta. Los ejemplos estudiados suelen apoyarse en más de dos micrófonos para saberlo.

La estimación de la distancia aparece como otro punto difuso. Los conceptos de lejos y cerca son objeto de estudio. Se utilizan sistemas de umbrales en círculos concéntricos y técnicas de HRTF cuando el problema es al revés. Cuando se generan escenarios tridimensionales en los cuales se pretende sumergir al oyente y, en este caso, es el sistema auditivo quien localiza, con la precisión a la que nos tiene

acostumbrado.

¿Cómo se podrían crear métodos que fueran capaces de trabajar en condiciones en las que, a parte de un ruido ambiente alto, hubiera más de una fuente emitiendo sonido?. ¿Podríamos implementar código que estimará dónde están las dos fuentes, las cuales han sido recibidas en el mismo o en los mismos paquetes, mezcladas en una señal acústica?. ¿Es el análisis de frecuencias la solución? ¿Hay que pensar de nuevo, en la información de las señales enmascaradas?

Al mismo tiempo se pueden explorar los caminos de los reconocedores de voz y conferir a RobEx la capacidad de entendernos e incluso de reaccionar, obedecer y por qué no, de discutir.

Los caminos abiertos por la introducción de un nuevo sentido pueden ser múltiples. Ahora esta empezando a oír. Simplemente registra el entorno sonoro y toma una pequeña decisión de orientación. Hasta que imite nuestras capacidades, las ramificaciones de este trabajo se hacen innumerables. Mi camino pasa por conseguir la atención de RobEx, es decir, conseguir que al llamarlo me ubique y posteriormente, al verme, venga hacia mi o incluso reconozca mi cara. El futuro de la intersección entre el sentido actual de la vista y del oído, ese bucle cognitivo, está amaneciendo en Robolab.

Bibliografía.

[1] B.J.C. Moore. *An Introduction to the Psychology of hearing*. Quinta edición. 2008.

[2] Cristian Sepulveda. C*reación de un modelo eléctrico análogo al sistema auditivo completo, es decir Oído Externo, Medio e Interno.* Tesis doctoral. 2006

[3] Eric Martinson. *Acoustical awareness for intelligent robotic action.* Tesis doctoral. 2009.

[4] Pilar Bachiller Burgos. *Percepción dinámica del entorno en un robot móvil.* Tesis doctoral, 2008.

[5] J. A. torres viveros. *Aplicación de técnica de grabación y mezcla biaural para audio comercial y/o publicitario.* Proyecto fin de carrera. 2009.

[6] Alastair Sibbald. *Virtual Ear Technology*. 2002.

[7] Alastair Sibbald. *Hearing in Three Dimensions.* 2004

[8] Michi Henning and Mark Spruiell. *Distributed Programming with Ice.*, revision 3.3.0. 2008.

[9] H. Giorgi y J. Moreno. *Implementación en tiempo real de modelos de sonido biaural dentro de un sistema tridimensional integrado de imagen y audio.* Tesis doctoral. 2005.

[10] Alastair Sibbald. *An Introduction To Sound And Hearing.* 2002.

[11] Wikipedia. http://es.wikipedia.org/wiki/Oído_externo.

[12] Robolab. Robex arena - http://robexarena.com. 2009.

[13] Robolab. RoboComp project - http://robocomp.wiki.sourceforge.net. 2009.

[14] Agustín Sánchez Domínguez. *Diseño y construcción de un controlador de motores dc basado en microcontroladores*. Proyecto fin de carrera. 2007.

[15] Creative Commons España. http://es.creativecommons.org. 2009.

[16] J. L. Fernández. *Estudio Comparativo entre sistema de Audio 3D y Sistemas 5.1 Aplicado en Video Juegos.* Tesis doctoral. 2006.

[17] James and Jim M. Van Verth, Lars M. Bishop. *Essential Mathematics for Games and Interactive Applications.* Segunda edición 2008.

[18] Eduard Bertran Alberti. *Procesado digital de señales. Fundamentos para comunicación y control.* 2006.

[19] PortAudio. http://www.portaudio.com.

[20] libsndfile. http://www.mega-nerd.com/libsndfile/api.html.

[21] Matlab. http://www.mathworks.com/.

[22] Luis Joyanes., *Fundamentos de programación. Algoritmos, estructuras de datos y objetos.* 3ª edición 2003.

[23] Luis J. Manso. *Navegación visual en robots móviles*. Proyecto fin de carrera. 2009.

Índice de ilustraciones.