Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Aplicación de un sistema de posicionamiento óptimo de sensores al diseño de un entorno distribuido de agrupaciones de micrófonos

Autor: Roberto Macho Pedroso

Tutores: Cristina Losada Gutiérrez y Francisco Domingo Pérez

2015

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Aplicación de un sistema de posicionamiento óptimo de sensores al diseño de un entorno distribuido de agrupaciones de micrófonos

Autor: Roberto Macho Pedroso

Directores: Cristina Losada Gutiérrez y Francisco Domingo Pérez

Tribunal:

Presidente: Javier Macías Guarasa

Vocal 1º: Óscar Esteban Martínez

Vocal 2º: Cristina Losada Gutiérrez

Calificación: .....

Fecha: .....

A todos los que han estado a mi lado a lo largo de mi vida...

Agradecimientos

Por fin ha llegado el momento de finalizar la carrera. La consecución de unos de los logros más importantes de mi vida y quiero agradecer a todos aquellos que me han ayudado de una forma u otra para conseguirlo.

En primer lugar a mis tutores del TFG, Cristina Losada Gutiérrez y Francisco Domingo Pérez por su ayuda y paciencia. Ya que sin ellos no sería posible.

A Javi, Jose y los demás miembros del GEINTRA que tanto me han ayudado.

Al resto de profesores de la EPS que me han enseñado todo lo que sé.

A mis padres, Jesús y Conchi, por apoyarme en lo necesario para terminar mi carrera.

A mis hermanos, Héctor y Aurora, que con su apoyo me han animado cuando lo he necesitado.

A mis amigos que han sabido aguantarme todos estos años y me han insistido para terminar.

Por último y más importante a Sabrina por hacerme más llevadera la vida desde que la conozco y por saber sacarme siempre una sonrisa incluso cuando yo creo que no es posible.

A todos ellos, gracias por hacerme llegar hasta aquí.

Resumen

En este trabajo se plantea un sistema desarrollado en MATLAB que permite obtener la ubicación óptima de un conjunto de sensores o micrófonos para la localización de objetivos en un entorno cerrado.

Dicha ubicación óptima se obtiene mediante el uso de un algoritmo genético que minimiza el error de localización para sistemas basados en TDOA y en SRP.

Todo el sistema está gobernado con una interfaz gráfica de MATLAB que permite agilizar el uso de las funciones y modificar muy fácilmente todos los parámetros de éstas.

Palabras clave: Localización, TDOA, SRP, posicionamiento óptimo de sensores, agrupaciones de micrófonos.

Abstract

This work outlines the features and functioning of a specifically created MATLAB-developed system which allows obtaining the optimal positioning of sensors or microphone arrays for the localization of objectives in a closed environment.

This optimal positioning is obtained thanks to a genetic algorithm that minimizes localization errors for TDOA and SRP-based systems.

The whole system is being ruled by a MATLAB graphic interface which allows accelerating the usage of functions and easily modifying all of their parameters.

Keywords: Localization, TDOA, SRP, optimal positioning of sensors, microphone arrays.

Resumen extendido

La localización y el seguimiento de objetivos (persona/s u objeto/s) en un espacio inteligente es fundamental para mejorar los procesos de interacción con el entorno.

En este trabajo se plantea un sistema desarrollado en MATLAB que permite obtener la ubicación óptima de un conjunto de sensores para la localización de objetivos en un entorno cerrado.

El principal problema de la localización en entornos cerrados es la limitación de técnicas que encontramos, ya que no podemos usar técnicas como la localización por GPS o radiobaliza. Mientras que con esas técnicas de localización se obtienen precisión del orden de metros, en entornos cerrados nos enfrentamos al problema de la precisión de localización, ya que en estos entornos un error de metros es demasiado elevado, por lo que se necesitan métodos de localización más precisos.

En el sistema desarrollado se utilizan o bien sensores genéricos con un modelo de propagación basado en un modelo de rayos o bien micrófonos para la localización.

La localización mediante sensores genéricos permite una localización sencilla y poco ruidosa, ya que las señales se trasmiten de forma directa y sin demasiada reverberación.

El uso de micrófonos para la localización es menos invasivo que la localización con otros sensores que necesitan emisores, ya que como usa la voz de las personas o el ruido generado por los objetos no necesita emisores de señal externos.

Para comenzar, en este trabajo se realiza un estudio teórico de los distintos métodos de localización permitidos por el sistema desarrollado, mediante algoritmos basados en *Time Diference of Arrival (TDOA)* y *Steered Response Power (SRP)* (TDOA para todo tipo de sensores y SRP únicamente para micrófonos).

El estudio teórico sigue con la forma de minimizar su error mediante el uso de algoritmos metaheurísticos, en concreto los genéticos. Esta técnica permite reducir considerablemente el tiempo de cálculo de las ubicaciones óptimas de los sensores o micrófonos, ya que se basa en recrear los procesos de evolución de la naturaleza mediante cruces y mutaciones de las distintas soluciones. Al no ser un método determinista, ya que no se prueban las infinitas soluciones posibles, no nos proporciona un resultado absoluto pero si nos proporciona una solución aceptable en un tiempo razonable.

Después del estudio teórico se ha implementado un sistema que permite obtener la ubicación óptima de los sensores para la localización de un objetivo.

Todo el sistema está gobernado con una interfaz gráfica de MATLAB (GUI) que permite agilizar el uso de las funciones y modificar muy fácilmente todos los parámetros de éstas.

Esta interfaz gráfica permite modificar la información a tener en cuenta de la sala (tipo de cómputo de obstáculos), el tamaño de rejilla de puntos objetivo, realizar modificaciones de la topología de los arrays de micrófonos, el número de sensores a utilizar y las posiciones posibles de éstos, las funciones utilizadas por el algoritmo genético,...

Además permite no sólo ver la simulación al acabar el resultado, sino también ver resultados anteriores para poder analizar los distintos métodos de análisis del sistema. También está desarrollado con el fin de poder ver también el funcionamiento del algoritmo genético, ya que esta diseñado para almacenar el mejor resultado cada cinco generaciones, de esta forma en actualizaciones futuras podría permitir ver el funcionamiento del algoritmo en tiempo real a través de la interfaz.

Por último se han evaluado los resultados obtenidos por el sistema implementado y se presentan próximas líneas futuras de desarrollo para un mayor aprovechamiento del sistema desarrollado.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El objetivo de una buena introducción definitiva es que el lector se contente con ella, lo entienda todo y no lea el resto

Como se hace una tesis. Umberto Eco

1.1 Introducción

El análisis automático de espacios inteligentes a partir del procesamiento de múltiples sensores es un área de cada vez mayor actividad científica. En ese contexto, las tareas de detección, localización y seguimiento de objetivos (personas o cualquier otro elemento móvil) son fundamentales para mejorar los procesos de interacción con el entorno, o con otras personas u objetos dentro del mismo.

El Grupo de Ingeniería Electrónica aplicada a Espacios Inteligentes y Transporte (GEINTRA) del Departamento de Electrónica de la Universidad de Alcalá ha arrancado una línea de actividad en la que se han ido planteando trabajos orientados a la generación de tecnología basada en el procesamiento de las señales captadas por agrupaciones de sensores de distintos tipos [1] (basados en señales acústicas, tanto en la banda de audio [2] como en la de ultrasonidos, señales infrarrojas [3], y señales de vídeo, con cámaras de distintos tipos).

Una etapa fundamental en el despliegue de agrupaciones de sensores en un entorno real es la de la decisión acerca de la topología y distribución espacial de todos ellos. El enfoque tradicional asume una topología conocida a priori, y una distribución geométrica que trata de cubrir razonablemente el espacio objeto de análisis. El inconveniente fundamental de esta estrategia es que, en general, no utiliza criterios objetivos y medibles para la toma de decisiones, lo que puede implicar limitaciones en las prestaciones que finalmente se obtendrán de los algoritmos que usan esos sensores.

Una alternativa deseable sería la determinación automática de dicha topología y distribución espacial, utilizando algoritmos de optimización basados en distintos paradigmas y que se basan en el uso de métricas objetivas sobre las que se construyen las funciones a optimizar [4, 5].

Una de estas técnicas de optimización son los algoritmos genéticos [6], que es una búsqueda heurística que imita el proceso de la selección natural. Dicho proceso se basa en aplicar los procesos de mutación, recombinación y selección que tienen lugar en la naturaleza a una población de soluciones con el fin de encontrar la mejor solución posible.

El trabajo fin de grado que aquí se ha realizado se centra en la adaptación del algoritmo de posicionamiento óptimo de sensores infrarrojos en interiores que está siendo desarrollado por Francisco Domingo Pérez [7] (dentro de su tesis doctoral), para usarlo con agrupaciones de micrófonos [2], así como en la realización de una interfaz gráfica para su fácil manejo, lo que permitirá realizar cómodamente modificaciones en los parámetros del entorno, geometría de la habitación, posibles obstáculos, etc.

1.2 Objetivos

El objetivo fundamental de este trabajo es la adaptación del método de posicionamiento óptimo [7], basado en el uso de algoritmos genéticos, para obtener la mejor ubicación para un conjunto de agrupaciones de micrófonos, con el fin de calcular la posición del hablante en una habitación determinada, entendiendo como mejor ubicación aquella que tiene un menor error de posicionamiento.

Los objetivos específicos de este proyecto son los siguientes:

- Adaptar el programa de MATLAB para utilizar el algoritmo diseñado[7] por Francisco Domingo Pérez para el posicionamiento de las agrupaciones de micrófonos.
- Diseñar, implementar y evaluar una interfaz gráfica para su fácil manejo, y que tendrá las siguientes características:
    1. Posibilidad de variar las dimensiones de la habitación.
    2. Posibilidad de introducir obstáculos.
    3. Posibilidad de cambiar parte de los algoritmos.
    4. Visor para ver la ubicación de las agrupaciones de sensores y los posibles puntos “ciegos".
- Realizar pruebas experimentales para evaluar y validar los resultados obtenidos.

1.3 Estructura de la memoria

A continuación se describe la organización y la estructura de cada uno de los capítulos contenidos en esta memoria.

En el Capítulo 2 se realiza el estudio teórico de las distintas técnicas de localización usadas en este trabajo y del algoritmo de ubicación óptima de los sensores.

A lo largo del Capítulo 3 se explica detalladamente el sistema desarrollado en este trabajo, su configuración y los resultados proporcionados.

El Capítulo 4 está dedicado a las conclusiones y las líneas futuras.

En el Capítulo 5 se detallan los requisitos para la utilización del sistema desarrollado.

El Capítulo 6 contiene el presupuesto detallado de este proyecto.

Capítulo 2

Estudio teórico

El conocimiento comienza por la práctica y todo conocimiento teórico, adquirido a través de la práctica, debe volver a la práctica.

Sobre la práctica y la contradicción. Mao Zedong

2.1 Introducción al problema de localización y precisión de la estimación de la posición

El objetivo de la localización es obtener, con la mejor precisión posible, la posición de un objetivo que emita una señal dentro de un habitáculo o recinto.

Para ello es necesario el uso de sensores, ya sean infrarrojos, ultrasonidos, micrófonos,etc. dependiendo del tipo de señal que emita el objetivo. Empleando las señales medidas por dichos sensores, podemos obtener información de la posición, procesando las medidas con un PC o un microprocesador para aplicaciones más reducidas.

Dicha localización se hace de forma pasiva ya que el objetivo únicamente emite una señal pero no hace ningún procesamiento. Este proceso puede ser reversible y ser el objetivo, el sensor que a partir de la información de los emisores fijos, estime su posición de forma activa, de forma análoga al A-GPS que utiliza triangulación de señales de telefonía que llegan al teléfono [8].

Este cálculo de la posición se puede realizar en función del tiempo de llegada (*Time of Arrival - TOA*), la diferencia de tiempo de llegada (*Time Difference of Arrival - TDOA*), la potencia de la señal recibida (*Received Signal Strength - RSS*) o la dirección de llegada (*Direction of Arrival - DOA*) de las señales recibidas por los sensores. TOA, TDOA y RSS proporcionan la información de la posición a partir de la distancia entre la fuente y los sensores, mientras que en DOA la orientación de la fuente es relativa a los sensores.

En la tabla 2.1 se puede ver un resumen de las características de estos métodos, sus ventajas y sus inconvenientes.

La propagación de la línea de visión (LOS) se refiere a la radiación de ondas electromagnéticas o de ondas acústicas. Implica que tiene que haber un camino directo entre el emisor y el receptor.

Tabla 2.1: Comparación entre los diferentes métodos de medida (adaptada de [1])

Método	**Información de la Localización**	**Ventajas**	**Inconvenientes**
TOA	Distancia	Alta precisión	Es necesaria la sincronización temporal tanto en la fuente como en todos los sensores. LOS necesaria
TDOA	Diferencia de distancias	Alta precisión No es necesaria la sincronización temporal en la fuente	LOS necesaria
RSS	Distancia	Simple y económico No es necesaria la sincronización temporal en la fuente	Baja precisión
DOA	Orientación	Sólo son necesarios dos sensores. No es necesaria la sincronización temporal	Son necesarias antenas inteligentes LOS necesaria

Teniendo esto en cuenta, el resto del trabajo se centrará en el uso del método TDOA, ya que ofrece alta precisión eliminando la sincronización con la fuente. Este método utiliza uno de los sensores como referencia y calcula la diferencia de tiempo de llegada de las señales a los otros sensores con respecto al de referencia.

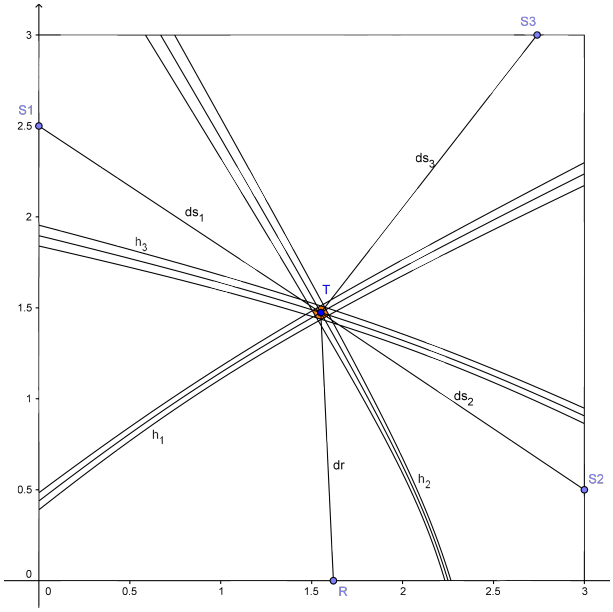


Figura 2.1: TDOA con 4 sensores en 2D

En la Figura 2.1 se puede ver un ejemplo de funcionamiento del método TDOA en dos dimensiones en una habitación con cuatro sensores. En la figura se puede observar cómo usando un algoritmo basado en TDOA se realiza la localización mediante trilateración hiperbólica [9].

La localización se realiza trazando $N-1$ cónicas (hipérbolas), siendo N el número de sensores y la hipérbola el lugar geométrico de los puntos del plano que mantiene constante la diferencia de la distancia a los dos focos, cuyos focos serán el sensor de referencia y cada uno de los $N-1$ sensores respectivamente, que pasa por el objetivo T.

En ausencia de ruido dichas hipérbolas se cortarían en un único punto, pero teniendo el ruido presente aparecen dos hipérbolas más para cada pareja de sensores, produciéndose una región de estimación.

Para la localización en dos dimensiones son necesarios al menos 3 sensores (uno de referencia y dos más para calcular la diferencia de distancias), sin embargo para la localización en el espacio tridimensional es necesario el uso de al menos 4 sensores, y en vez del uso de hipérbolas es necesario el uso de hiperboloides.

Una vez obtenido el tiempo de llegada de la señal a los N sensores, se selecciona uno de ellos como referencia y se calcula la diferencia de medida entre cada uno de los otros $N-1$ sensores y el de referencia[7]. Las $N-1$ diferencias de medidas pueden ser expresadas como una función de los parámetros a estimar ($\theta = [x\ y\ z]^T$):

$$r = \mu(\theta) + \epsilon \tag{2.1}$$

Donde cada elemento es un vector columna de tamaño $N-1$ con la forma de la ecuación 2.2.

$$r = \begin{pmatrix} \Delta d_{1,r} \\ \Delta d_{2,r} \\ \vdots \\ \Delta d_{N-1,r} \end{pmatrix}; \epsilon = \begin{pmatrix} \epsilon_{1,r} \\ \epsilon_{2,r} \\ \vdots \\ \epsilon_{N-1,r} \end{pmatrix}; \mu(\theta) = \begin{pmatrix} \|\theta - x_1\| - \|\theta - x_r\| \\ \|\theta - x_2\| - \|\theta - x_r\| \\ \vdots \\ \|\theta - x_{N-1}\| - \|\theta - x_r\| \end{pmatrix} \tag{2.2}$$

Asumiendo que sigue una distribución normal, la diferencia de medidas se puede modelar como $r \sim \mathcal{N}(\mu(\theta), \Sigma)$. Siendo ϵ el error en cada medida (con una media nula y varianza $\sigma^2_{x,r}$), μ el vector de las diferencia de normas (distancias) de los sensores al objetivo y la referencia al objetivo y $\Delta d_{x,r}$ es la diferencia de medida entre el sensor x y el de referencia.

La matriz de covarianza es una matriz cuadrada de orden $N-1$ formada por las varianzas y las covarianzas de las variables, que modela el error de los sensores para un método de medida en concreto.

Para el método TDOA, que es en el que se centra este trabajo, la matriz de covarianza es de la forma de la ecuación 2.3. Como el error de la referencia está presente en cada medida hay correlación entre éstas, por lo cual los elementos de fuera de la diagonal principal en la matriz de covarianza no son nulos.

$$\Sigma = \begin{pmatrix} \sigma^2_{1,r} & \sigma^2_r & \cdots & \sigma^2_r \\ \sigma^2_r & \sigma^2_{2,r} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \sigma^2_r \\ \sigma^2_r & \cdots & \sigma^2_r & \sigma^2_{N-1,r} \end{pmatrix} \tag{2.3}$$

Siendo:

$$\sigma^2_{x,r} = \sigma^2_x + \sigma^2_r \tag{2.4}$$

El cálculo de la matriz de covarianza se realiza teniendo en cuenta que el error de los sensores es dependiente de la distancia de éstos al objetivo. Dicha matriz será una matriz simétrica de tamaño $N-1$ con la forma de la ecuación 2.5.

$$\Sigma = \begin{pmatrix} dr^2 + ds_1^2 & dr^2 & \cdots & dr^2 \\ dr^2 & dr^2 + ds_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & dr^2 \\ dr^2 & \cdots & dr^2 & dr^2 + ds_{N-1}^2 \end{pmatrix} \tag{2.5}$$

De forma simplificada se puede calcular suponiendo que el error es independiente de la distancia entre el emisor y el sensor, con unos en su diagonal y 0.5 en el resto. De la forma de la ecuación 2.6.

$$\Sigma = \begin{pmatrix} 1 & 0{,}5 & \cdots & 0{,}5 \\ 0{,}5 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0{,}5 \\ 0{,}5 & \cdots & 0{,}5 & 1 \end{pmatrix} \tag{2.6}$$

Para resolver el problema hay que utilizar algún estimador no lineal. Se sabe que existen estimadores eficientes o asintóticamente eficientes (como máxima verosimilitud). Un estimador es eficiente si la varianza de la estimación alcanza la cota inferior de Cramér-Rao (*Cramér-Rao lower bound - CRLB*), así que podemos utilizarla como función objetivo a minimizar para encontrar un conjunto de sensores que ofrezcan el menor error en la posición estimada posible.

La CRLB permite obtener una cota inferior de la varianza de un estimador insesgado de un parámetro determinista. Ésta se puede calcular con la ecuación 2.7. Esta función se obtiene calculando la inversa de la Matriz de Información de Fisher (*Fisher Information Matrix - FIM*), la cual se obtiene con la segunda derivada de la función de verosimilitud. En algunos casos esa cota no existe dando como resultado infinito, cuando la FIM es singular.

$$CRLB = [J^T \cdot \Sigma^{-1} \cdot J]^{-1} \tag{2.7}$$

Siendo J el jacobiano de la función $\mu(\theta)$ definido en la ecuación 2.2. Así, la fórmula del jacobiano para TDOA es la mostrada en 2.8.

$$J = \begin{pmatrix} (\frac{x_T - x_{S1}}{ds_1} - \frac{x_T - x_R}{dr}) & (\frac{y_T - y_{S1}}{ds_1} - \frac{y_T - y_R}{dr}) & (\frac{z_T - z_{S1}}{ds_1} - \frac{z_T - z_R}{dr}) \\ (\frac{x_T - x_{S2}}{ds_2} - \frac{x_T - x_R}{dr}) & (\frac{y_T - y_{S2}}{ds_2} - \frac{y_T - y_R}{dr}) & (\frac{z_T - z_{S2}}{ds_2} - \frac{z_T - z_R}{dr}) \\ \vdots & \vdots & \vdots \\ (\frac{x_T - x_{S(N-1)}}{ds_{N-1}} - \frac{x_T - x_R}{dr}) & (\frac{y_T - y_{S(N-1)}}{ds_{N-1}} - \frac{y_T - y_R}{dr}) & (\frac{z_T - z_{S(N-1)}}{ds_{N-1}} - \frac{z_T - z_R}{dr}) \end{pmatrix} \tag{2.8}$$

De esta forma se tiene en cuenta tanto el error en las medidas, con la información contenida en la matriz de covarianza, como la geometría de la sala, pues cada fila es una diferencia de vectores unitarios que indican la dirección $\overrightarrow{TS}$ y $\overrightarrow{TR}$. Por lo que optimizando la CRLB se obtiene un conjunto de sensores que consigue un menor error cuadrático medio en el posicionamiento[10].

En este trabajo se tendrán en cuenta los obstáculos fijos de la habitación, como las columnas. Por tanto hay que tenerlos en cuenta en los cálculos del TDOA, en la matriz de covarianza y en el cálculo del jacobiano. Para esto se considera que en caso de que un obstáculo esté situado entre el objetivo y un sensor, la distancia entre ellos será considerada infinita.

2.2 Ubicación óptima de sensores

2.2.1 Definición del problema

La ubicación de los sensores cambia considerablemente el error de posicionamiento obtenido, por eso es importante tratar de buscar la posición óptima de los sensores.

El problema de dicho cálculo es la infinidad de soluciones posibles. En el siguiente ejemplo se puede apreciar cómo, al aumentar el número de sensores, las posiciones de éstos se incrementan exponencialmente, considerando posiciones como ubicaciones posibles del conjunto de sensores y no de cada sensor en particular:

Supongamos que limitamos las posiciones de cada sensor a 100 posibles ubicaciones en una habitación. Ahora analicemos cómo aumentan las combinaciones posibles de posiciones de los sensores con el número de éstos:

- 1 sensor: 100 distribuciones posibles.
- 2 sensores: $\binom{100}{2} = 4950$ distribuciones posibles.
- 3 sensores: $\binom{100}{3} = 161700$ distribuciones posibles.

  .

  .

  .

- N sensores: $\binom{100}{N}$ distribuciones posibles.

Aunque 100 posiciones es un número muy bajo para un ejemplo real en el que las posiciones posibles superan con facilidad los millones (p.e. para una habitación cuadrada de 3·3·3m tenemos 3,6e9 posiciones posibles), esto permite ver cómo calcular todos los resultados y elegir el mejor podría suponer un tiempo no asumible, ya que podría superar los años en un caso real.

Para reducir notablemente el tiempo de estimación de la ubicación óptima es muy recomendable el uso de algoritmos metaheurísticos. De forma general este tipo de algoritmos nos permite obtener una solución más o menos satisfactoria (dependiendo del algoritmo en concreto utilizado y del número de iteraciones) en un tiempo razonable. En concreto en este trabajo se va a utilizar el algoritmo genético que está siendo desarrollado por Francisco Domingo Pérez[7] que será explicado en el siguiente apartado.

2.2.2 Descripción teórica del algoritmo de ubicación óptima

Un algoritmo genético[6] es un algoritmo metaheurístico que imita la teoría de la evolución de Charles Darwin para la resolución de problemas.

Para ello se parte de una población inicial que cumple los requisitos del problema, creada de forma aleatoria, de la cual se seleccionan los individuos más capacitados para luego reproducirlos o cruzarlos y mutarlos para poder obtener una siguiente generación de individuos mejor adaptados que los de la anterior generación.

En la Figura 2.2 se puede ver el diagrama de flujo de un algoritmo genético genérico:

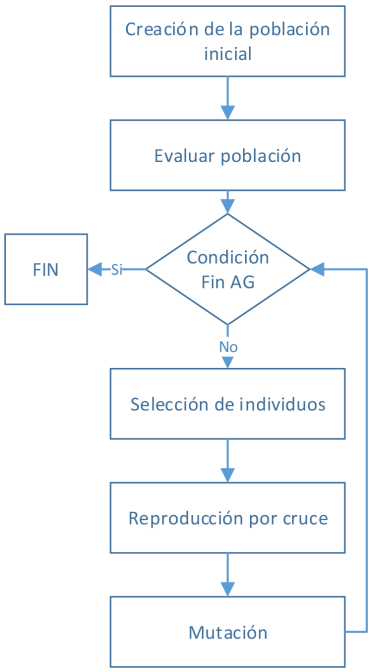


Figura 2.2: Diagrama de bloques general de un algoritmo genético

Esta estructura general se puede adaptar fácilmente a cada problema de la siguiente manera:

1. Se define de la mejor forma posible la población.
2. Se crea la población inicial de forma aleatoria cumpliendo los requisitos del problema.
3. Se definen las funciones de cruce y mutación.
4. Se crea la función de evaluación de la población o función objetivo. Dicha función se centra en evaluar cada individuo de la población asignándole un valor para que después la función de selección pueda escoger los mejores individuos para trabajar con ellos.

   La función objetivo puede adaptarse al problema para garantizar que se cumplan ciertas características de forma que si no cumplen dichas características se penalice la solución.

2.3 Posicionamiento utilizando arrays de micrófonos

Hay muchas aplicaciones en las que es necesario conocer la posición de una persona u objeto en un entorno interior. Hay diferentes alternativas para esto, algunas no funcionan en interiores como la localización mediante GPS o por radiobaliza, otras son intrusivas (ya que requieren que el usuario lleve encima algún tipo de dispositivo) como la localización mediante sensores IR o ultrasonidos, otras pueden no respetar la privacidad de gente como las cámaras que además encarecen el sistema y por último la localización con micrófonos.

Por tanto el uso de micrófonos como receptores para posteriormente calcular la posición del hablante es mucho más aplicable para espacios cerrados que el resto de técnicas, por lo que se justifica el uso de micrófonos para el posicionamiento.

Un posible caso de aplicación en el que sea requerido calcular el posicionamiento de una o varias personas en un espacio cerrado es para poder crear un mapa de calor del movimiento de la/s persona/s

en una sala, una tienda o una parte de un centro comercial para poder usar dicho mapa de calor del movimiento para colocar de manera más precisa los anuncios o los artículos que se quieran vender más.

2.3.1 Explicación teórica del problema de posicionamiento

Para el posicionamiento en una sala se miden los tiempos de llegada o la diferencia de tiempo de llegada de las señales a los sensores. Esas señales pueden llegar de forma directa o por medio de reflexiones. Esas reflexiones producen unos nuevos caminos para que las señales lleguen a los sensores. La suma de todos los caminos posibles se denomina multicamino.

Al igual que para sensores infrarrojos se ha demostrado que es bastante influyente el efecto del multicamino de la señal recibida, en los micrófonos también es aún mas relevante. Por lo que es necesario tenerlo en cuenta para el posicionamiento.

Para el análisis acústico, se ha tenido en cuenta el método geométrico de las imágenes [11], basado en la teoría de rayos, en el que las reflexiones del sonido son calculadas formando imágenes especulares de la fuente primaria con respecto a las diferentes superficies reflectantes del entorno. Esas imágenes se consideran como fuentes secundarias que emiten nuevos rayos sonoros con una atenuación relativa al coeficiente de reflexión de la superficie (el cual depende del tipo de material de la superficie), y con un retardo que depende de la distancia entre la fuente y el punto del receptor de la imagen. Dicho análisis asume que las reflexiones en las paredes son especulares, es decir que el ángulo de reflexión es igual al ángulo de incidencia. Además se modelan las ondas sonoras como rayos, entendiéndose por rayo a la línea que indica la dirección y el sentido de propagación del sonido, con un contenido energético que depende de la energía total radiada, del número de rayos emitidos y de la directividad de la fuente.

Las imágenes especulares son los puntos simétricos del destino con respecto a las distintas superficies de reflexión. En la figura 2.3 se muestra un ejemplo sencillo de la aplicación del método de las imágenes para una habitación rectangular. Siendo S la fuente, T el destino, ST el camino directo, SAT el camino con una reflexion con respecto a S1 obtenido con el método de las imágenes y SBCT el camino con dos reflexiones con las superficies S1 y S2 obtenido con el método de las imágenes.

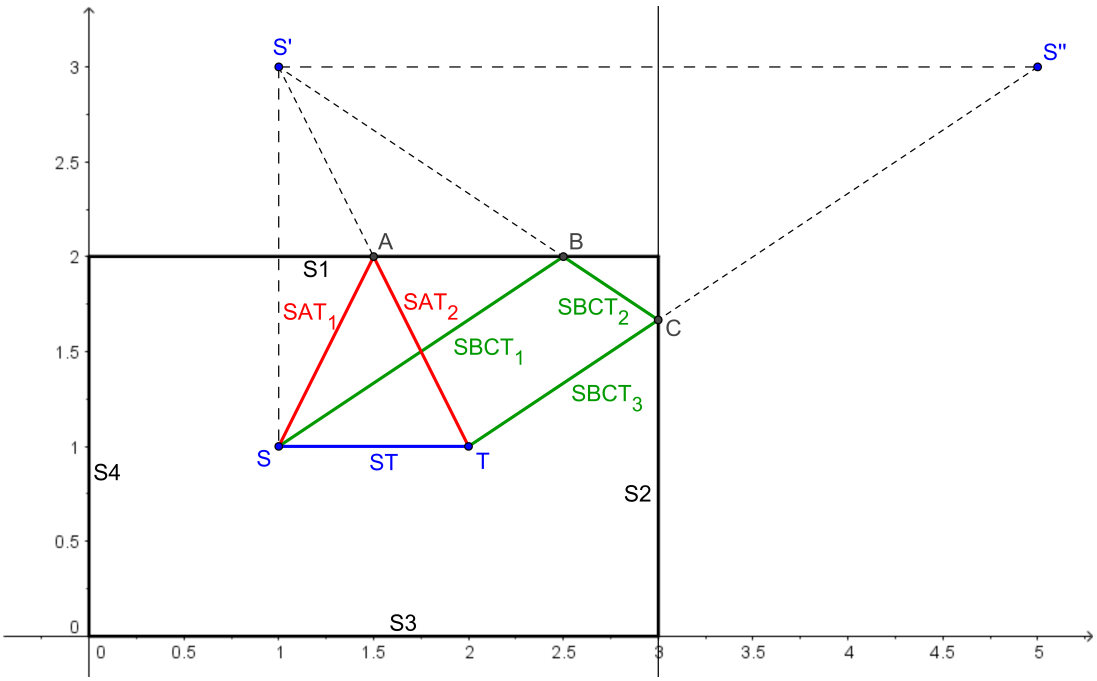


Figura 2.3: Ejemplo de aplicación del método de las imágenes para una habitación rectangular

Una vez obtenidos los puntos imagen para cada objetivo a analizar, se consideran como nuevos objetivos con una atenuación inversamente proporcional a la distancia al origen y teniendo en cuenta el coeficiente de reflexión de las paredes.

La localización en el contexto de grupos de micrófonos, ya sean colocados de forma aislada o como en arrays, puede llevarse a cabo mediante algoritmos basados en TDOA explicado en apartados anteriores o mediante algoritmos basados en *Steered Response Power (SRP)*. A diferencia de los basados en TDOA que necesitan un único sensor de referencia, los basados en SRP calculan la diferencia de tiempos en función de pares, por lo que son menos sensible a los obstáculos.

Por su mejor utilización en entornos ruidosos y con reverberación y por la posibilidad de contrastar las soluciones con los datos reales con lo que contamos, en este trabajo se va a utilizar un algoritmo basado en SRP para la localización mediante micrófonos.

En señales acústicas su longitud de onda varía entre $1{,}715cm$ y $17,15m$ para frecuencias entre $20Hz$ y $20kHz$, por lo que habría que tener en cuenta los obstáculos, debido a las difracciones. Sin embargo, como para la localización mediante micrófonos sólo vamos a tener en cuenta la sala sin obstáculos, ya que los datos reales con que contamos son de una sala como esa, en este trabajo no se van a considerar los obstáculos para la localización con micrófonos. No obstante para casos en los que haya que tener en cuenta los obstáculos, el proceso sería el mismo que para los métodos basados en TDOA, considerando distancias infinitas para cada camino "tapado" por un obstáculo.

Los algoritmos basados en SRP se basan en la capacidad de los micrófonos de apuntar a una determinada dirección o posición del espacio, hacen uso de técnicas de *beamforming*[12]. En el uso de esta técnica para localización, a la salida del beamformer se le conoce como *steered response*.

La forma más sencilla de implementar una steered response es la utilización de un *delay-and-sum beamformer*. Éste aplica los diferentes retardos temporales (en función de la posición del espacio a localizar) a la señal de entrada de cada uno de los micrófonos, produciendo una alineación temporal de éstas, para posteriormente sumarlas para obtener la salida del algoritmo. Otro método, utilizando un *filter-and-sum beamformer*, aplica un filtrado previo a las señales antes de la alineación y la suma de éstas.

En señales acústicas el filtrado, además de eliminar el ruido y las interferencias, aunque no deberían distorsionar significativamente la señal, debe tratar de trasladar la mayor cantidad de potencia posible. Para esto es muy recomendable el uso de la Transformada de Fase (*Phase Transform - PHAT*).

Por esto en este trabajo se va a utilizar un algoritmo de localización basado en SRP-PHAT, en concreto el desarrollado exhaustivamente por José Velasco Cerpa en su artículo [13]. En dicho artículo se establece que la función de la Transformada Discreta de Fourier (*Discrete Fourier Transform - DFT*) de la salida SRP-PHAT es la de la fórmula 2.9.

$$P_{DFT}(\mathbf{q}) = \frac{1}{N_F}\sum_{j=1}^{N_P}\left[\sum_{k=0}^{K}\sum_{l=0}^{K}\frac{2\sin\left(\frac{2\pi(M_0-0{,}5)}{N_F T_s}\left(\Delta\tau(\mathbf{p}_j,\mathbf{q})-\Delta\tau(\mathbf{p}_j,\mathbf{r})\right)\right)}{\tau^*_{j1k}\tau^*_{j2l}\,D(\mathbf{r})\sin\left(\frac{\pi}{N_F T_s}\left(\Delta\tau(\mathbf{p}_j,\mathbf{q})-\Delta\tau(\mathbf{p}_j,\mathbf{r})\right)\right)}\right] \tag{2.9}$$

Una vez obtenido el patrón SRP se tiene una matriz cuadrada de orden M, siendo M el número de puntos objetivo analizados. Suponiendo que x es el punto a localizar e y el punto obtenido en la localización. Para optimizar el error medio es necesario normalizar cada fila del patrón SRP, obteniendo la función de densidad $f_{xy}(y,x)$. Con esta función de densidad se puede calcular el error medio con la ecuaciones 2.10 y 2.11.

$$Error(y) = \sum_{1}^{M} \|y - x\| * f_{xy}(y, x); \tag{2.10}$$

$$Error_{medio} = \sum_{1}^{M} \frac{Error(y)}{M} \tag{2.11}$$

2.3.2 Modelado del error de posicionamiento con arrays de micrófonos

En este trabajo se utilizan arrays de 4 micrófonos colocados en forma de T invertida con una separación horizontal de 20cm y una separación vertical de 30cm, como el de la figuras 2.4 y 2.5.

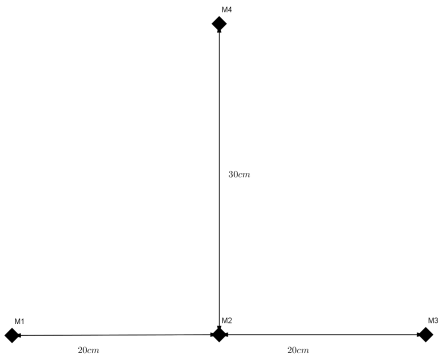


Figura 2.4: Esquema array de micrófonos utilizado en este trabajo

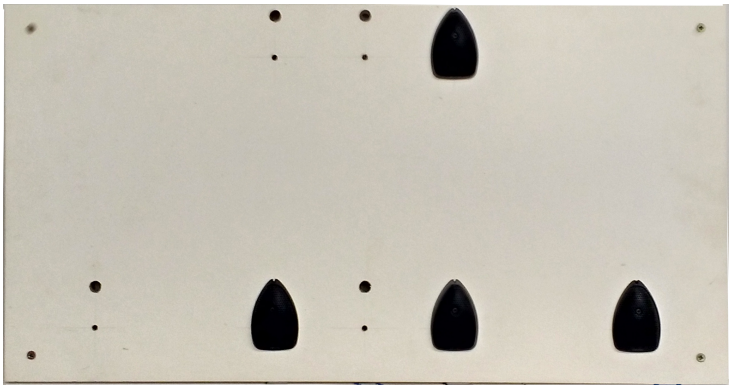


Figura 2.5: Array real de micrófonos utilizado en este trabajo

Por esta razón el algoritmo genético para este tipo de sensores se define de forma que únicamente busque la posición de uno de los micrófonos del array (en este caso el central), posteriormente en la función objetivo se calculan las posiciones del resto de los micrófonos en función de éste y calcula la función objetivo para todos los micrófonos del array.

Al igual que para sensores de otro tipo para el cálculo de la posición dentro del espacio en tres dimensiones es necesario el uso de cuatro micrófonos, en el caso de este tipo de arrays sería posible el posicionamiento con un único array.

Capítulo 3

Desarrollo

Si la hipótesis sobrevive a la prueba experimental, aumenta su prestigio, y al ir siendo aceptada se desarrolla y se extiende en una forma cada vez más comprensiva.

Max Planck

3.1 Introducción

En los siguientes apartados se explicará con detalle el sistema desarrollado en este trabajo. Este trabajo se centra en la creación de un sistema de posicionamiento óptimo de sensores y de arrays de micrófonos para la localización de personas u objetos, adaptando la técnica de posicionamiento óptimo desarrollada por Francisco Domingo Pérez en su tesis doctoral[7]. Esta técnica permitía calcular las posiciones de los sensores en dos dimensiones para habitaciones rectangulares.

3.2 Sistema desarrollado

El sistema desarrollado en este trabajo extiende la técnica de posicionamiento óptimo de sensores de Francisco Domingo Pérez para habitaciones de cualquier forma, para localizar a personas u objetos en el espacio tridimensional y pudiendo poner restricciones de ubicación a los sensores y obstáculos.

Se van a usar además arrays de micrófonos con una topología específica, formados por cuatro micrófonos colocados en forma de T invertida definidos por la separación vertical y horizontal de los micrófonos, como el de la figura 2.4, utilizando una técnica de localización basada en SRP-PHAT. Aunque también permite el uso de micrófonos aislados, así como sensores que usen técnicas de localización basadas en TDOA.

3.2.1 Objetivo y descripción general del sistema desarrollado

El objetivo del sistema desarrollado es obtener la ubicación óptima de los sensores o arrays de micrófonos en una sala con unas características concretas.

El sistema permite además escoger como se tienen en cuenta los obstáculos que haya en la sala, y aunque pueda mostrar todos los obstáculos, sólo se permite tener en cuenta a la hora de calcular el error en la medida los obstáculos fijos (columnas).

Está diseñado también para forzar la ubicación de los sensores en las paredes, el suelo, el techo o combinaciones de éstos.

Todo esto puede configurarse de manera sencilla a través de una interfaz, que ayuda a seleccionar el tipo de sensores a utilizar para el posicionamiento, los obstáculos a tener en cuenta, las restricciones de paredes, altura mínima para colocar los sensores en las paredes, configurar los parámetros del algoritmo genético,...

3.2.2 Datos de entrada y configuración

Aunque la interfaz permite seleccionar la habitación para la cual se quiere realizar el proceso de posicionamiento de los sensores, para configurar las características de dicha sala es necesario definirla en modo texto en un archivo "$.srf''$ como el de la figura 3.1.

```
[ROOMInfo]

version = 1.1
comment = Idiap demo room

numVertexes = 4

roomFloorHeight = −730
roomCeilingHeight = 1670


[Vertexes]

vertex0 = 1800 2200 −730
vertex1 = 1800 −6000 −730
vertex2 = −1800 −6000 −730
vertex3 = −1800 2200 −730

[Obstacle 1]
v_obst0 = 600 800 −30
v_obst1 = 600 −4000 −30
v_obst2 = −600 −4000 −30
v_obst3 = −600 800 −30
obstacle1FloorHeight = −10
obstacle1CeilingHeight = −30
obstacle1Fixed = 0
obstacle1Name = table
```

Figura 3.1: Ejemplo de archivo ".srf" de definición de sala

El archivo "$.srf''$ estará dividido en 2 o más partes (dependiendo de si hay o no obstáculos definidos en la habitación y del número de éstos).

Una primera parte encabezada por $[ROOMInfo]$ contiene la información sobre las características generales de la habitación:

- version: Número de versión.
- comment: Nombre de la sala.
- numVertexes: Número de vértices de la habitación.
- roomFloorHeight: Valor z en mm del suelo.
- roomCeilingHeight: Valor z en mm del techo.

Un segundo apartado encabezado por $[Vertexes]$ contiene las componentes [x,y,z] de los vértices expresadas en mm de la habitación definidos en el suelo. De la siguiente forma: $vertex[x] = xyz$

Los siguientes apartados encabezados con $[Obstaclex]$ si son necesarios son relativos a los obstáculos, uno para cada obstáculo, con los siguientes:

- v_obstx = x y z: Componentes [x,y,z] expresadas en mm de cada vertice x del obstáculo.
- obstacle[x]FloorHeight: componente z inferior del obstáculo expresada en mm.
- obstacle[x]CeilingHeight: componente z superior del obstáculo expresada en mm.
- obstacle[x]Fixed: Valor 0 para obstáculos no fijos (por ejemplo mesas) y valor 1 para obstáculos fijos (por ejemplo columnas).
- obstacle[x]Name: Nombre del obstáculo.

Además en el caso del uso de micrófonos es necesario cargar los puntos imagen posibles para cada punto objetivo que vamos a analizar, ya que únicamente se hará el análisis para tres posibles tamaños de la rejilla (entendiendo como rejilla el conjunto de puntos objetivo separados a una distancia determinada, el *STEP*). Dicha información de los puntos imagen está contenida en archivos *".raw"* para cada punto de la rejilla como el de la figura 3.2.

```
1 0 −1.80000000000000000e+03 7.60000000000000000e+03 9.70000000000000000e+02
1 1 −1.80000000000000000e+03 −8.80000000000000000e+03 9.70000000000000000e+02
1 3 5.40000000000000000e+03 −3.20000000000000000e+03 9.70000000000000000e+02
2 1 0 −1.80000000000000000e+03 1.32000000000000000e+04 9.70000000000000000e+02
2 0 1 −1.80000000000000000e+03 −1.96000000000000000e+04 9.70000000000000000e+02
2 3 0 5.40000000000000000e+03 7.60000000000000000e+03 9.70000000000000000e+02
2 1 3 5.40000000000000000e+03 −8.80000000000000000e+03 9.70000000000000000e+02
2 3 2 −9.00000000000000000e+03 −3.20000000000000000e+03 9.70000000000000000e+02
```

Figura 3.2: Ejemplo de archivo ".raw" de puntos imagen

Donde la primera columna hace referencia al número de rebote al que pertenece el punto imagen, la segunda y la tercera (sólo la segunda para un rebote) se refiere a la pared o paredes con las que rebota y las otras tres incluyen las coordenadas [x,y,z] del punto imagen en mm.

Esos son los únicos parámetros no configurables o no calculados automáticamente a través de la interfaz desarrollada y sus funciones auxiliares.

En las tablas 3.1 y 3.2 se pueden ver las distintas funciones creadas y/o utilizadas en este trabajo.

Tabla 3.1: Funciones creadas y/o utilizadas en este trabajo. Parte 1

Declaración de la función	**Uso de la función**
conv=calc_conv(nwalls,nrebounds)	Calcula las posibles combinaciones para un número de paredes y un número de rebotes
target=calc_grid_points(room,Step)	Calcula y define las posiciones de los puntos objetivo en el espacio 3D
obstacle=checkobstacles(sensorPosition, targetPoint,room)	Busca si hay obstáculos en la trayectoria entre un sensor y un punto objetivo
Population = gacreation4room(GenomeLength, FitnessFcn,options)	Crea una población inicial para el algoritmo genético
varargout = Interface(varargin)	Define las funciones de la GUI y sirve para lanzarla
intermediate_results(result)	Permite visualizar gráficamente los resultados intermedios de una solución y su evolución temporal
valid=is_valid_solution(solution, sensors,room,Matrix_type)	Comprueba si una solución es válida bajo unas restricciones
mJacob = jacobiano(sensors, , target,room)	Calcula la matriz jacobiana. Adaptada de [7]
[super_grid,N_rebotes]= LoadGrid(grid_file,n_rebounds,step)	Lee los ficheros de los puntos imagen para el calculo del grid usando micrófonos
room=LoadRoom(room_file)	Lee un fichero ".srf" de sala
[t, pairs] = location_to_timedelay (geometry, pairs, locations, fs, c)	Calcula los retardos de localización
y = pair_distance(geometry, pairs)	Calcula la distancia entre pares de micrófonos
plotdevelop(room,walls,floor,ceiling)	Permite ver el desarrollo 2D generado de la sala, NO NECESARIA
point3D=point2Dto3D(point,room)	Permite pasar de un punto del desarrollo 2D a un punto en el espacio 3D NO NECESARIA
point2D=point3Dto2D(point,room)	Permite pasar de un punto en el espacio a un punto del desarrollo 2D

Las funciones señaladas como *NO NECESARIA* no son necesarias para el funcionamiento correcto del sistema pero son útiles para realizar comprobaciones.

Tabla 3.2: Funciones creadas y/o utilizadas en este trabajo. Parte 2

Declaración de la función	**Uso de la función**
[xMatrix Output]=positioning (n_sensors,act_room,Step,Matrix_Type, Objective_Type,gaoptions,rebounds)	Calcula la posición optima de los sensores para una sala en unas determinadas condiciones
printwalls(result)	Dibuja en pantalla el resultado en el espacio 3D
save_cluster_script(room,minsensors, maxsensors, Step,Matrix_Type, Objective_Type,gaoptions)	Crea el script ".bash" necesario para lanzar las simulaciones en el cluster
Patron_SRP=SRPpattern_cuboidROOM (super_grid,MICS,pares, N_rebotes, Velocidad, Fo, Fs,N_FFT)	Calcula el patrón SRP[13]
test	Función para ser lanzada en el cluster
result = testsolution(result)	Calcula el error para la solución del algoritmo
objective = traceCRLB(anchors,anchorRef, targets,Matrix_Type,Objective_Type, Functions,rebounds)	Función objetivo del algoritmo Adaptada de [7]
room=upd_room(room)	Calcula y define las paredes, suelo y techo tanto en el espacio 3D como en el desarrollo 2D de éste

Durante la ejecución de la GUI se generan las siguientes estructuras de datos:

- room (tabla 3.3): Contiene la información de la sala. A su vez ésta contiene otras estructuras y cell arrays:
  - obstacles (tabla 3.4): Contiene las distintas estructuras de los obstáculos.
  - surface3D (tabla 3.5): Contiene la información de las paredes en el espacio tridimensional de la sala.
  - surface2D (tabla 3.6): Contiene la información de las paredes en el desarrollo en bidimensional de la sala.
  - MicArray (tabla 3.7): Contiene las medidas de los arrays de micrófonos.
  - sensorPosition (tabla 3.8): Contiene las posiciones posibles de los sensores.
- result (tabla 3.9): Contiene la solución de la ubicación de los sensores obtenida por algoritmo genético. Contiene además estas estructuras:
  - Room (tabla 3.3)
  - GAoptions (tabla 3.10): Contiene las opciones de entrada del algoritmo genético.
  - GAoutput (tabla 3.11): Contiene la solución del algoritmo genético.
  - TestSol: Contiene la información del error de ubicación para una rejilla de un tamaño fijo.

Tabla 3.3: Parámetros de la estructura room

Parámetro	**Tamaño**	**Función**
comment	1xn char	Comentario o nombre de la sala
numVertexes	1x1 double	Número de vértices en el suelo
roomFloorHeight	1x1 double	Componente Z del suelo
roomCeilingHeight	1x1 double	Componente Z del techo
Vertexes	(numVertexes)x3 double	Componentes [x,y,z] de los vértices del suelo en metros
Xsize	1x1 double	Xmax-Xmin
Ysize	1x1 double	Ymax-Ymin
obstacles	1x1 cell	Cell Array de obstáculos
surface3D	1x1 struct	Información 3D de la sala
surface2D	1x1 struct	Información 2D de la sala
adjwall	1x(nwalls) cell	Paredes completas
ignorewalls	1x(ignoredwalls) double	Paredes completas ignoradas
obstacleoptions	1x1 double	Opciones posibles de análisis de obstáculos
MicArray	1x1 struct	Información del array de micrófonos
sensorPosition	1x1 struct	Restricciones de ubicación de los sensores en la sala
minHeight	1x1 double	Altura mínima para ubicar los sensores

Tabla 3.4: Parámetros de la estructura obstacles

Parámetro	**Tamaño**	**Función**
vertexes	(nvertexes)x3 double	Contiene las coordenadas [x,y,z] de los vértices del obstáculo
FloorHeight	1x1 double	Contiene la altura mínima del obstáculo
CeilingHeight	1x1 double	Contiene la altura máxima del obstáculo
Fixed	1x1 double	1: Obstáculo Fijo 0: Obstáculo no fijo
Name	1xn char	Nombre del obstáculo

Tabla 3.5: Parámetros de la estructura surface3D

Parámetro	**Tamaño**	**Función**
walls	(nwalls)x(nvertixes)x3 double	Puntos 3D de los vértices de las paredes
floor	(nvertixes)x3 double	Puntos 3D de los vértices del suelo
ceiling	(nvertixes)x3	Puntos 3D de los vértices del techo

Tabla 3.6: Parámetros de la estructura surface2D

Parámetro	Tamaño	Función
walls	(nwalls)x(nvertixes)x2 double	Puntos 2D de los vértices de las paredes
floor	(nvertixes)x2 double	Puntos 2D de los vértices del suelo
ceiling	(nvertixes)x2	Puntos 2D de los vértices del techo

Tabla 3.7: Parámetros de la estructura MicArray

Parámetro	Tamaño	Función
horizdist	1x1 double	Separación horizontal de los micrófonos del array
vertdist	1x1 double	Separación vertical de los micrófonos del array

Tabla 3.8: Parámetros de la estructura sensorPosition

Parámetro	Tamaño	Función
walls	1x1 double	0 o 1: Imposibilidad o posibilidad de ubicar los sensores en las paredes
floor	1x1 double	0 o 1: Imposibilidad o posibilidad de ubicar los sensores en el suelo
ceiling	1x1 double	0 o 1: Imposibilidad o posibilidad de ubicar los sensores en el techo

Tabla 3.9: Parámetros de la estructura result

Parámetro	Tamaño	Función
Room	1x1 struct	Sala donde se ha ejecutado el algoritmo
n_sensors	1x1 double	Número de sensores
Step	1x1 double	Separación de la rejilla
Matrix_type	1x1 double	Tipo de sensor escogido
Objective_type	1x1 double	Tipo de objetivo seleccionado
GAoptions	1x1 struct	Opciones del algoritmo genético
Rebounds	1x1 double	Número de rebotes analizados
xMatrix	(n_sensors)x3 double	Posición de los sensores
GAoutput	1x1 struct	Salida del algoritmo genético
telapsed	1x1 double	Tiempo transcurrido
x1	(ngenerations/5)x (n_sensorsx2) double	Posiciones 2D de las soluciones intermedias del algoritmo genético
t1	(ngenerations/5)x1 double	Tiempo de las soluciones intermedias

Tabla 3.10: Parámetros de la estructura GAoptions

Parámetro	**Tamaño**	**Función**
walls	1x1 double	0 o 1: Imposibilidad o posibilidad de ubicar los sensores en las paredes
floor	1x1 double	0 o 1: Imposibilidad o posibilidad de ubicar los sensores en el suelo
ceiling	1x1 double	0 o 1: Imposibilidad o posibilidad de ubicar los sensores en el techo

Tabla 3.11: Parámetros fundamentales de la estructura GAoutput

Parámetro	**Tamaño**	**Función**
generations	1x1 double	Número de generaciones alcanzadas
message	1xn char	Mensaje de finalización

A continuación de la tabla 3.12 a la 3.32 se detalla para cada una de las funciones los párametros de entrada-salida (incluidas sus carácteristicas) y su funcionamiento interno.

Tabla 3.12: $function\ conv = calc\_conv(nwalls, nrebounds)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
nwalls	Entrada	Número de paredes con las que calcular los rebotes	1x1 double
nrebounds	Entrada	Número de rebotes	1x1 double
conv	Salida	Posibles combinaciones de rebotes	1x1 cell

Tabla 3.13: $function\ target = calc\_grid\_points(room, Step)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura con la información de la sala	1x1 struct
Step	Entrada	Tamaño rejilla puntos objetivo	1x1 double
target	Salida	Coordenadas 3D de los puntos objetivo	(ntargets)x3 double

Tabla 3.14: $function\ obstacle = checkobstacles(sensorPosition, targetPoint, room)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
sensorPosition	Entrada	Coordenadas 3D del sensor	1x3 double
targetPoint	Entrada	Coordenadas 3D del punto objetivo	1x3 double
room	Entrada	Estructura con la información de la sala	1x1 struct
obstacle	Salida	1:Trayectoria con obstáculo/s 0:Trayectoria libre	1x1 double

Tabla 3.15: $function\ Population = gacreation4room(GenomeLength, FitnessFcn, options)$

Parámetro	Entrada/Salida	Información	Tamaño
sensorPosition	Entrada	Coordenadas 3D del sensor	1x3 double
targetPoint	Entrada	Coordenadas 3D del punto objetivo	1x3 double
room	Entrada	Estructura con la información de la sala	1x1 struct
obstacle	Salida	1:Trayectoria con obstáculo/s 0:Trayectoria libre	1x1 double

Tabla 3.16: $function\ intermediate\_results(result)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct

Tabla 3.17: $function\ valid = is\_valid\_solution(solution, sensors, room, Matrix\_type)$

Parámetro	Entrada/Salida	Información	Tamaño
solution	Entrada	Solución a probar	1x(2xsensors) double
sensors	Entrada	Número de sensores	1x1 double
room	Entrada	Estructura con la información de la sala	1x1 struct
Matrix_type	Entrada	Tipo se sensores	1x1 double
valid	Salida	1:Solución válida 0:Solución no válida	1x1 double

Tabla 3.18: $function\ mJacob = jacobiano(sensors, anchorRef, target, room)$

Parámetro	Entrada/Salida	Información	Tamaño
solution	Entrada	Solución a probar	1x(2xsensors) double
sensors	Entrada	Número de sensores	1x1 double
room	Entrada	Estructura con la información de la sala	1x1 struct
Matrix_type	Entrada	Tipo se sensores	1x1 double
valid	Salida	1:Solución válida 0:Solución no válida	1x1 double

Tabla 3.19: $function\ [super\_grid, N\_rebotes] = LoadGrid(grid\_file, n\_rebounds, step)$

Parámetro	Entrada/Salida	Información	Tamaño
grid_file	Entrada	Nombre del fichero de rejilla	1xn char
n_rebounds	Entrada	Número de rebotes	1x1 double
step	Entrada	Separación de la rejilla	1x1 double
super_grid	Salida	Rejilla con puntos imagen incluidos	(ntargets)x3x(ncov) double
N_rebotes	Salida	Rebotes para cada tipo de rebotes	(ncov)x1 double

Tabla 3.20: $function\ room = LoadRoom(room\_file)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room_file	Entrada	Nombre del fichero de sala	1xn char
room	Salida	Estructura con la información de la sala	1x1 double

Tabla 3.21: $function\ [t, pairs] = location\_to\_timedelay(geometry, pairs, locations, fs, c)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
geometry	Entrada	Disposición de los sensores	3xN double
pairs	Entrada	Parejas de micrófonos	(nPairs)x2 double
locations	Entrada	Coordenadas 3D de los puntos objetivo	3x(ntargets) double
fs	Entrada	Frecuencia de muestreo en Hz	1x1 double
c	Entrada	Velocidad del sonido en el aire (m/s)	1x1 double
t	Salida	Tiempos de retardo	(nPairs)x(ntargets) double
pairs	Salida	Parejas de micrófonos	3x(ntargets) double

Tabla 3.22: $function\ y = pair\_distance(geometry, pairs)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
geometry	Entrada	Disposición de los sensores	3xN double
pairs	Entrada	Parejas de micrófonos	(nPairs)x2 double
y	Salida	Distancias entre pares	(nPairs)x1 double

Tabla 3.23: $function\ plotdevelop(room, walls, floor, ceiling)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
room	Entrada	Estructura de información de la sala	1x1 struct
walls	Entrada	Indica la posibilidad de colocar los sensores en las paredes	1x1 double
floor	Entrada	Indica la posibilidad de colocar los sensores en el suelo	1x1 double
Ceiling	Entrada	Indica la posibilidad de colocar los sensores en las paredes	1x1 double

Tabla 3.24: $function\ point3D = point2Dto3D(point, room)$

Parámetro	**Entrada/Salida**	**Información**	**Tamaño**
point	Entrada	Coordenadas 2D de un punto	1x2 double
room	Entrada	Estructura de información de la sala	1x1 struct
point3D	Salida	Coordenadas 3D de un punto	1x3 double

Tabla 3.25: $function\ point2D = point3Dto2D(point, room)$

Parámetro	Entrada/Salida	Información	Tamaño
point	Entrada	Coordenadas 3D de un punto	1x3 double
room	Entrada	Estructura de información de la sala	1x1 struct
point3D	Salida	Coordenadas 2D de un punto	1x2 double

Tabla 3.26: $function\ [xMatrix\ Output] = positioning(n\_sensors, act\_room, Step, Matrix\_Type, Objective\_Type, gaoptions, rebounds)$

Parámetro	Entrada/Salida	Información	Tamaño
n_sensors	Entrada	Número de sensores	1x1 double
room	Entrada	Estructura de información de la sala	1x1 struct
Step	Entrada	Separación de la rejilla de puntos objetivo	1x1 double
Matrix_type	Entrada	Tipo de sensores	1x1 double
Objective_type	Entrada	Tipo de objetivo	1x1 double
gaoptions	Entrada	Estructura de opciones del AG	1x1 struct
rebounds	Entrada	Número de rebotes	1x1 double
xMatrix	Salida	Ubicaciones 2D óptimas	(2N)x1 double
Output	Salida	Salida del AG	1x1 struct

Tabla 3.27: $function\ printwalls(result)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct

Tabla 3.28: $function\ save\_cluster\_script(room, minsensors, maxsensors, Step, Matrix\_Type, Objective\_Type, gaoptions)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura de información de la sala	1x1 struct
minsensors	Entrada	Número mínimo de sensores	1x1 double
maxsensors	Entrada	Número máximo de sensores	1x1 double
Step	Entrada	Separación de la rejilla de puntos objetivo	1x1 double
Matrix_type	Entrada	Tipo de sensores	1x1 double
Objective_type	Entrada	Tipo de objetivo	1x1 double
gaoptions	Entrada	Estructura de opciones del AG	1x1 struct

Tabla 3.29: $function\ Patron\_SRP = SRPpattern\_cuboidROOM(super\_grid, MICS, pares, N\_rebotes, Velocidad, Fo, Fs, N\_FFT)$

Parámetro	Entrada/Salida	Información	Tamaño
super_grid	Entrada	Rejilla con puntos imagen incluidos	(ntargets)x3x(ncov) double
MICS	Entrada	Disposición de los micrófonos	3xN double
pares	Entrada	Parejas de micrófonos	(nPairs)x2 double
N_rebotes	Entrada	Rebotes para cada combinación	(ncov)x1 double
Velocidad	Entrada	Velocidad del sonido	1x1 double
Fo	Entrada	Frecuencia de corte	1x1 double
Fs	Entrada	Frecuencia de muestreo	1x1 double
N_FFT	Entrada	Número de puntos de la FFT	1x1 double
Patron_SRP	Salida	Solución SRP	(ntargets)x(ntargets) double

Tabla 3.30: $function\ result = testsolution(result)$

Parámetro	Entrada/Salida	Información	Tamaño
result	Entrada	Estructura de la solución	1x1 struct
result	Entrada	Estructura de la solución	1x1 struct

Tabla 3.31: $function\ objective = traceCRLB(anchors, anchorRef, targets, Matrix\_Type, Objective\_Type, Functions, rebounds)$

Parámetro	Entrada/Salida	Información	Tamaño
anchors	Entrada	Posiciones 2D de los sensores	Nx2 double
anchorRef	Entrada	Sensor de referencia	1x1 double
targets	Entrada	Rejilla de puntos objetivo	(ntargets)x3 double
Matrix_type	Entrada	Tipo de sensores	1x1 double
Objective_type	Entrada	Tipo de objetivo	1x1 double
Functions	Entrada	Funciones del AG utilizadas	1x1 double
rebounds	Entrada	Número de rebotes	1x1 double
objective	Salida	Objetivo a minimizar	1x1 double

Tabla 3.32: $function\ room = upd\_room(room)$

Parámetro	Entrada/Salida	Información	Tamaño
room	Entrada	Estructura de información de la sala	1x1 struct
room	Entrada	Estructura de información de la sala	1x1 struct

3.2.3 Obtención de la ubicación óptima de los sensores

La obtención de la ubicación óptima de los sensores se realiza mediante un algoritmo genético que minimiza la función objetivo.

El algoritmo genético de ubicación de sensores está adaptado a partir de un algoritmo genético genérico 2.2.2, de la siguiente forma:

1. Se define de la mejor forma posible la población.

   Como se busca posicionar los sensores en las superficies de las paredes, techo y/o suelo, se ha adaptado el algoritmo de Francisco Domingo Pérez [7], el cual posicionaba los sensores en cualquier punto del suelo, de forma que ubique los sensores en una superficie plana formada por el desarrollo 2D de la sala. Así se definirían los individuos como las coordenadas en ese desarrollo 2D del conjunto de sensores a posicionar, concatenadas en un vector de tamaño $1x2N$, siendo N el número de sensores para el caso de sensores aislados. Por ejemplo para un caso con 4 sensores uno de los individuos se definiría como $[x_1\ y_1\ x_2\ y_2\ x_3\ y_3\ x_4\ y_4]$.

   Cuando se utilizan arrays de micrófonos, se puede realizar el mismo proceso que para un conjunto de micrófonos aislados, con la salvedad de que para un array (o un conjunto de arrays) de micrófonos, éstos están colocados de una forma determinada.

   Para el caso de arrays de micrófonos, moviendo únicamente uno de los micrófonos (en este trabajo el micrófono central), y calculando posteriormente para el cálculo del error los otros tres micrófonos, no sólo se agiliza el tiempo de trabajo del algoritmo genético, ya que de otra forma las soluciones que no están ubicadas con esa topología tendrían que descartarse, sino también es posible que para obtener una solución válida y óptima es posible que no convergiera el algoritmo. Por lo que los individuos serían de tamaño $1x2A$, siendo A el número de arrays de micrófonos.

   De esta forma la población se define como el conjunto de individuos, es decir, como el conjunto de vectores definidos anteriormente. Obteniendo una matriz de tamaño $Px2N$ o $Px2A$, siendo P el tamaño de la población.

2. Se crea la población inicial de forma aleatoria cumpliendo los requisitos del problema. En este trabajo se crea posicionándolos de forma aleatoria sobre las paredes no restringidas de la habitación.

   Para ello se define una rejilla de puntos muy pequeña de forma que haya muchos puntos posibles y se cogen PxN puntos y se colocan sus coordenadas en una matriz de tamaño $Px2N$.

3. Se definen las funciones de cruce y mutación. En este trabajo se utilizan distintas funciones de mutación y cruce de las definidas en MATLAB para los algoritmos genéticos [14].

   Las funciones de cruce usan dos soluciones (padres) para generar una tercera (hijo) dependiente de éstas, ya sea obteniendo el punto central de ambas, mezclando la información de ambas (por ejemplo usando la coordenada x de una solución y la coordenada y de la otra). Ésto se realiza con el fin de que la tercera solución, como sucede en la naturaleza donde los hijos heredan información de los padres, pueda ser una solución mejor que las anteriores.

   Las funciones de mutación en cambio, cogen una solución y la modifican de forma aleatoria, al igual que en la naturaleza.

4. Se crea la función de evaluación de la población o función objetivo. Dicha función se centra en evaluar cada individuo de la población asignándole un valor para que después la función de selección pueda escoger los mejores individuos para trabajar con ellos.

   La función objetivo usada en este trabajo es la llamada *traceCRLB*, que a su salida (la cual hay que optimizar) da información del error medio para esa solución. Dicha función calcula o bien la CRLB para sensores con localización basada en TDOA, o bien el error medio a partir de la SRP para la localización con micrófonos.

   Está adaptada también para garantizar que se cumplan ciertas características, por medio de la función *is_valid_solution*, y si no penalizar la solución. Esta penalización se aplica a las soluciones que no cumplen las características de ubicación restringidas, por ejemplo que el array de micrófonos

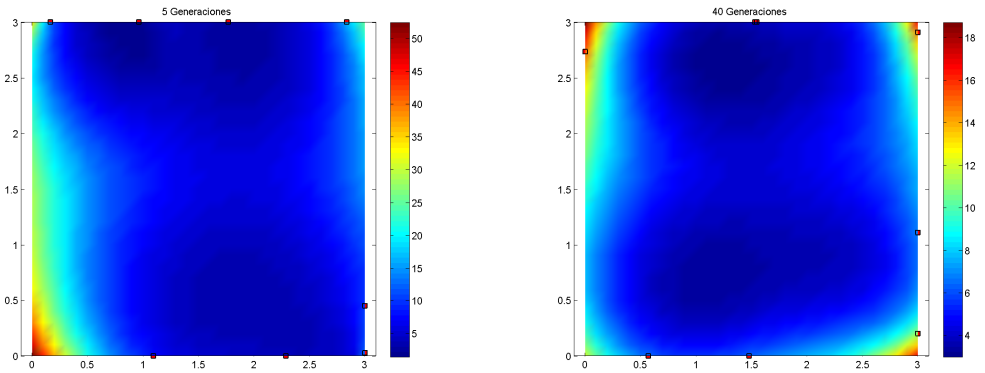
no tenga todos los micrófonos en la misma pared, que cumpla los requisitos de altura mínima o que estén situados dentro de las superficies deseadas.

Para calcular si hay un obstáculo en la trayectoria situada entre un sensor y el punto objetivo, se traza una línea que une estos dos puntos y comprueba si alguno de los puntos de ésta están situados dentro del polígono formado por el obstáculo en el suelo.

En la Figura 3.3 se puede ver el algoritmo genético específico desarrollado en este trabajo.

Figura 3.3: Diagrama de bloques del algoritmo genético empleado en este trabajo

En la figura 3.4 se puede ver el funcionamiento el algoritmo de posicionamiento para ubicar 8 sensores a lo largo de 200 generaciones. Según va avanzando el algoritmo se van desplazando las ubicaciones de los sensores a las posiciones con menor error. El error mostrado se calcula a una altura de 1,7 m teniendo en cuenta la estatura media humana, ya que es aproximadamente la altura donde se producen los sonidos.



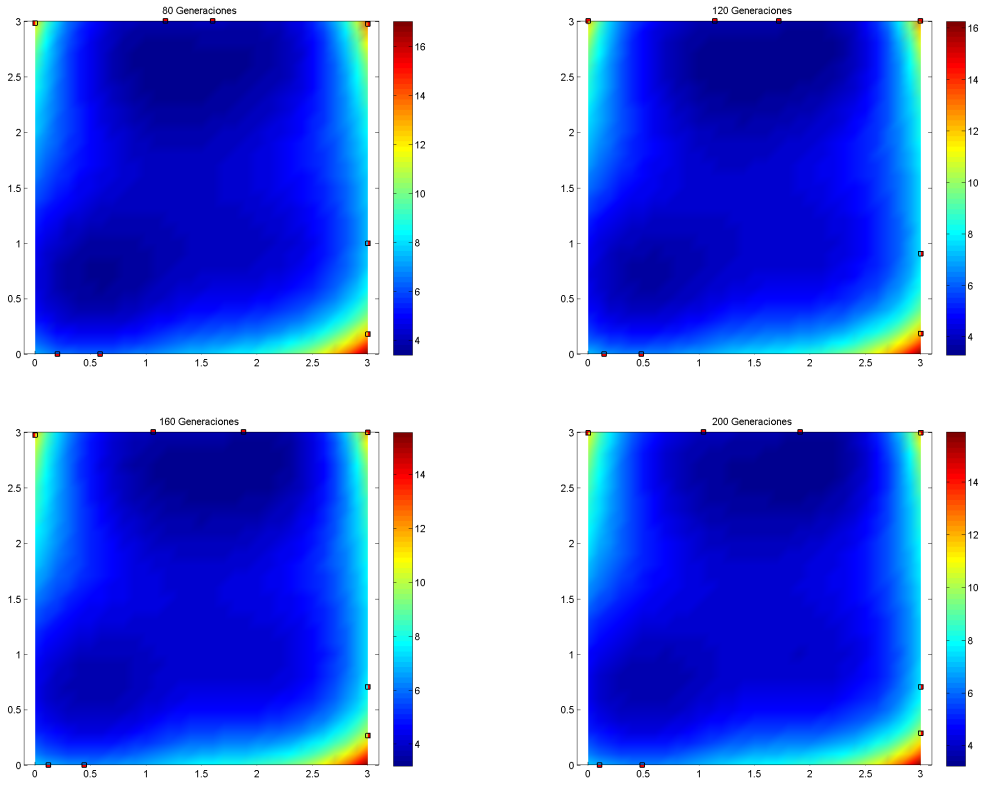


Figura 3.4: Error para 8 sensores a lo largo 200 generaciones

Como se puede ver en la gráfica de la figura 3.5 según va avanzando el algoritmo genético el error medio va disminuyendo hasta que llega a estabilizarse, por lo que podemos decir que el algoritmo converge a partir de ese número de generaciones para la configuración del algoritmo genético del ejemplo. Ésto no significa que sea la mejor ubicación posible ya que ampliando el número de generaciones del algoritmo y la población de éste la solución puede mejorar.

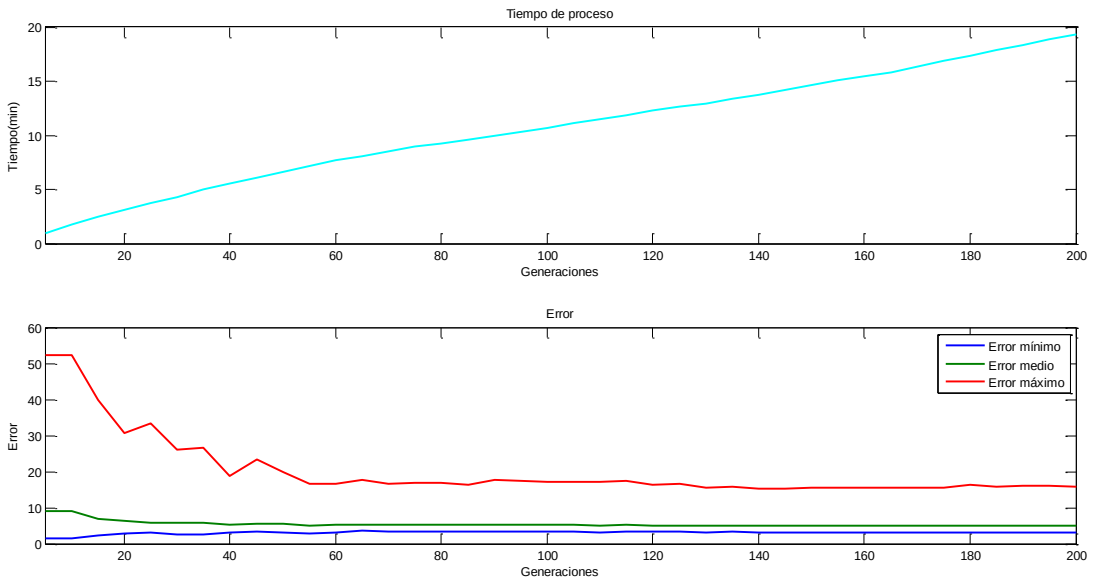


Figura 3.5: Evolución del algoritmo genético a lo largo de 200 generaciones

Por tanto se puede decir que el algoritmo utilizado minimiza el error medio en un tiempo razonable muy inferior al tiempo que se tardaría en probar todas las posibles soluciones, aunque no garantiza la mejor ubicación absoluta, ya que no usa un método determinista como sería probar las infinitas posibilidades.

3.2.4 Resultados proporcionados

A continuación se muestran varios resultados obtenidos con el algoritmo genético usado, tanto para uso sensores tipo IR con y sin obstáculos sin tener en cuenta el multicamino, como para micrófonos (aislados o con topología de array) sin obstáculos.

Los resultados se han obtenido para los espacios inteligentes de IDIAP (una sala rectangular sin obstáculos con la que podemos contrastar datos reales de micrófonos) y ISPACE (una sala con forma irregular, paredes curvas y con obstáculos), así como salas genéricas cuadradas o rectangulares (con y sin obstáculos).

En el apartado anterior 3.2.3 se ha mostrado el resultado obtenido para una sala cúbica de 3 m de lado, con 8 sensores con localización basada en TDOA y sin obstáculos.

En ese mismo contexto localización basada en TDOA, se encuentran los siguientes experimentos.

En la figura 3.6 se puede ver la solución óptima calculada para la misma sala cúbica la figura 3.4 pero con una columna cuadrada en el centro de la sala.

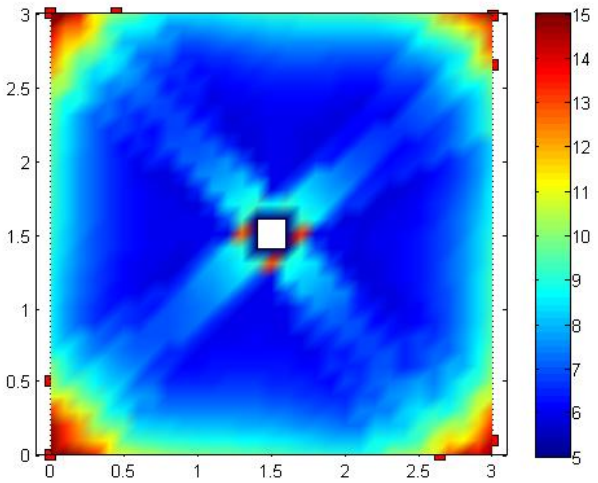


Figura 3.6: Localización con 8 sensores basada en TDOA con un obstáculo central

El la figura 3.7 se puede ver la posibilidad de análisis para una sala con obstáculos (ISPACE), con localización basada en TDOA con 10 sensores, si se tienen o no en cuenta los obstáculos. Ambas pruebas se han realizado limitando las paredes posibles para el posicionamiento de los sensores a las dos paredes donde están ubicados, ya que en la sala real en esas dos paredes no pueden ubicarse al haber ventanas y armarios en éstas.

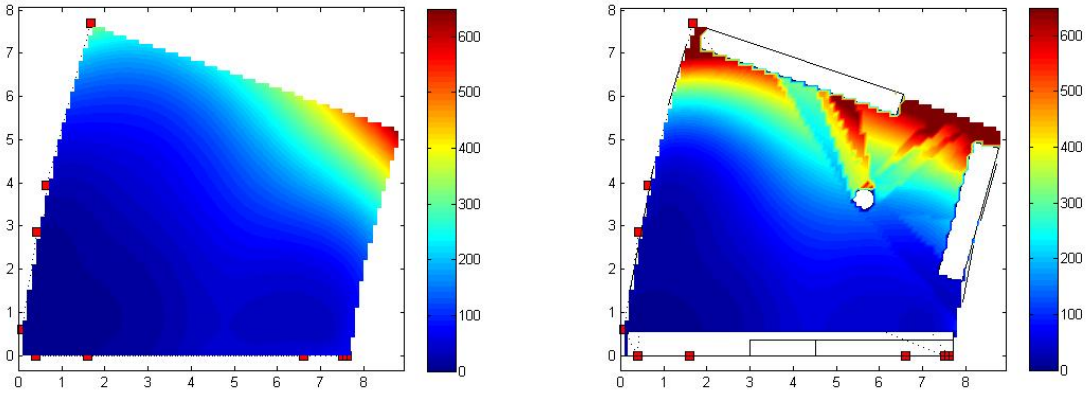


Figura 3.7: Contraste de solución sin obstáculos con solución con obstáculos

Como se puede apreciar en ambas gráficas, el error máximo de localización se encuentra en la esquina superior derecha ya que es el punto más alejado a todos los sensores y el error es dependiente de la distancia al cuadrado. Si se compara con la sala cuadrada el error es mucho mayor por la misma razón, ya que las paredes son más de dos veces más largas. Para el análisis teniendo en cuenta los obstáculos como se puede apreciar aparecen zonas con mayor error detrás de la columna de la sala, lo cual coincide con lo que se podría esperar en la realidad. El error máximo al tener en cuenta los obstáculos aumenta al tener en cuenta la penalización por obstáculo, por lo que en la gráfica aparece una zona con un valor mayor que el máximo valor representado (zona rojo oscuro).

Los experimentos realizados con micrófonos se han realizado todos sin rebotes y con un rebote para poder analizar la importancia de los rebotes en la localización.

Para comprobar el buen funcionamiento del sistema desarrollado se contrastan ubicaciones fijas de los micrófonos con ubicaciones obtenidas con el posicionamiento óptimo. En la figura 3.8 se puede ver la diferencia de error entre la imagen de la izquierda, que corresponde a la solución con la ubicación de un array fijado en la pared estrecha, con la imagen de la derecha que corresponde con la solución óptima para un array teniendo en cuenta todas las paredes, ambas realizadas sin tener en cuenta los rebotes.

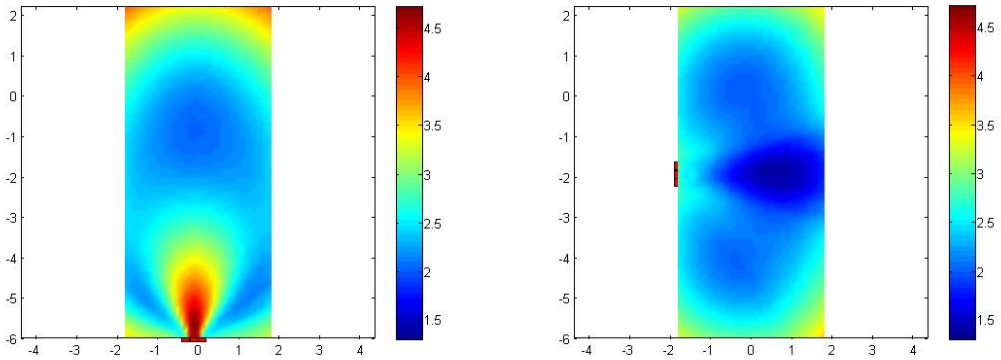


Figura 3.8: Contraste de una solución fijada en la pared estrecha con la solución óptima calculada sin rebotes

En la figura anterior se observa como con la solución optima nos proporciona un error más reducido. Si realizamos el mismo análisis con 1 rebote, como el de la figura 3.9, se observa un resultado similar al análisis sin rebotes con la salvedad de que en el caso de la pared estrecha el resultado es aún peor.

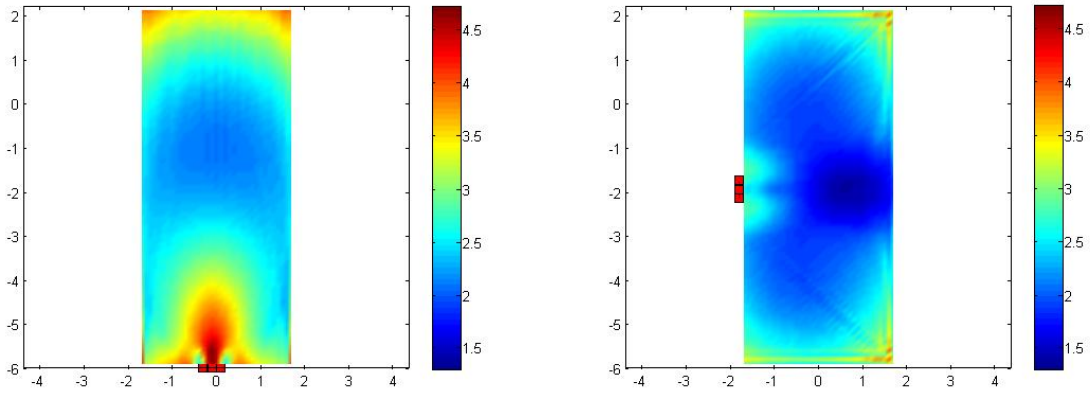


Figura 3.9: Contraste de una solución fijada en la pared estrecha con la solución óptima calculada con un rebote

De forma análoga se puede analizar la diferencia entre usar la topología en array y analizar el uso de micrófonos aislados y como se ve en la figura 3.10 tienden a juntarse también en torno a un punto central. Como se observa si se compara con las figuras 3.8 y 3.9 el uso de arrays proporciona un error simétrico, más uniforme y con un valor inferior, lo que demuestra que el uso de esta tipología de arrays de micrófonos mejora significativamente el error de localización.

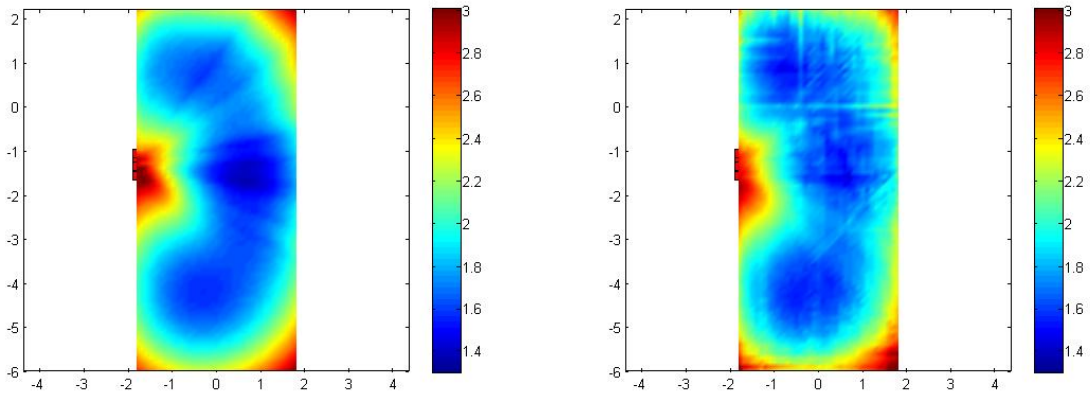


Figura 3.10: Solución para la sala IDIAP con 4 micrófonos aislados sin rebotes y con un rebote

En la figura 3.11 la solución con dos arrays de micrófonos con 1 rebote o sin ellos. Como se observa al aumentar el número de arrays igual que pasaba con 1 solo éstos tienden a posicionarse juntos en el centro de la sala, consiguiendo un error menor. Esto se debe a que la diferencia de tiempo de llegadas al estar posicionados cerca tiene mayor precisión que sólo con cuatro micrófonos.

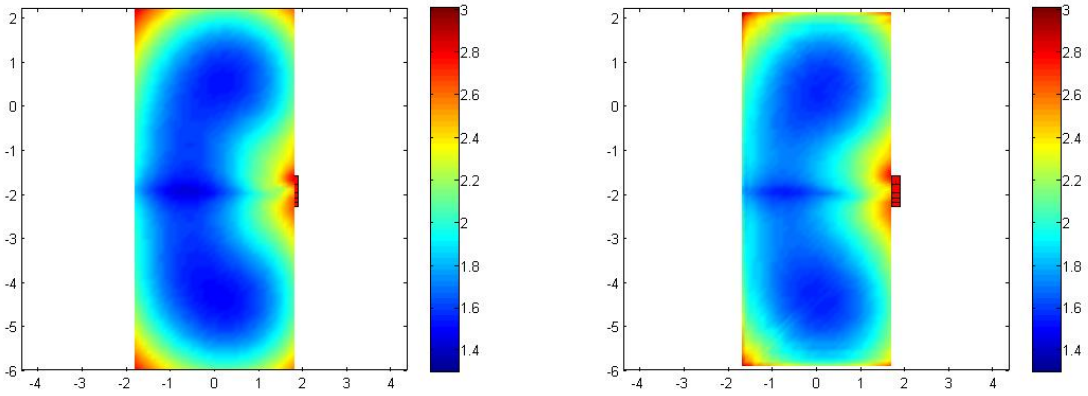


Figura 3.11: Solución para la sala IDIAP con 2 arrays de micrófonos sin rebotes y con un rebote

Capítulo 4

Conclusiones y líneas futuras

4.1 Conclusiones

En este capítulo se hará una revisión del contenido de esta memoria, haciendo énfasis en las conclusiones a las que se ha llegado a lo largo del desarrollo del proyecto.

En en Capítulo 1, se han expuesto los objetivos del presente proyecto y el punto de partida de éste.

En el Capítulo 2, se ha realizado el estudio teórico necesario para entender este proyecto. Se han desarrollado los conceptos de localización, distintos métodos (TDOA basado en distancias y SRP con puntos imagen) para conseguirlo y la forma de minimizarlos mediante el uso de un algoritmo genético.

En el Capítulo 3, se ha realizado una descripción detallada del sistema desarrollado, haciendo énfasis en su estructura interna, sus funciones. También se ha desarrollado en profundidad el método para minimizar el error de localización mediante el uso del algoritmo genético, explicando en detalle el método que usa para calcular las distintas soluciones, las funciones de mutación y cruce que usa y la forma de generar y probar las distintas soluciones. En este capítulo también se ha demostrados cómo este algoritmo reduce el tiempo necesario para obtener una solución aceptable en un tiempo razonable, mediante un ejemplo práctico.

El capítulo de desarrollo incluye además los resultados proporcionados por el sistema desarrollado. Dichos resultados nos llevan a afirmar que el algoritmo usado para calcular la ubicación óptima es altamente recomendable para este uso.

En los resultados mostrados, se aprecian varios tipos de errores con distintos rangos. A continuación se detalla el porqué de cada uno de ellos:

- En localización basada en TDOA está representada la solución de la CRLB, que es dependiente de la distancia al cuadrado por lo que cuanto más grande es la sala mucho mayor es este valor.
- En la localización basada en SRP está representado el error medio para cada punto.

En cuanto al tiempo de procesamiento para la obtención de la solución óptima, éste es muy dependiente del tiempo de proceso para cada solución, siendo mucho mayor el tiempo de procesamiento para la localización basada en SRP y siendo el tiempo de procesamiento de ésta muy dependiente del número de rebotes utilizado para el cálculo de la solución óptima.

Un buen método para obtener una mejor solución para la localización basada en SRP que requiere mucho tiempo de procesamiento, debido a los resultados obtenidos en el apartado 3.2.4, consiste en

obtener la solución óptima sin rebotes para un tamaño de rejilla suficientemente grande para que el tiempo de ejecución del algoritmo sea aceptable y una vez obtenida esa solución realizar nuevamente el proceso de posicionamiento de los micrófonos o arrays de micrófonos ya introduciendo uno o más rebotes limitando la ubicación de éstos a una única pared. De esta forma obtenemos una solución más precisa en menor tiempo, ya que reducimos considerablemente las ubicaciones posibles de micrófonos.

En conclusión se han conseguido todos los objetivos originales con la excepción de probar el algoritmo de ubicación óptima para micrófonos en otras salas.

4.2 Líneas futuras

A continuación se describen las futuras mejoras que se han ido planteando durante la realización de este proyecto.

4.2.1 Introducción en el sistema del cálculo de los puntos imagen para SRP

Los puntos imagen utilizados para la localización basada en SRP de la sala sin obstáculos en la que se han realizado las distintas pruebas han sido obtenidos de forma aislada con otras funciones, en una versión futura dichos puntos deberían poderse calcular dentro de este sistema con el fin de poder calcularlos para cualquier sala posible, con el fin de hacer el sistema completamente autónomo.

4.2.2 Añadir procesamiento de obstáculos para la localización basada en SRP

Como se ha explicado en el apartado 2.3.1, para la localización basada en SRP no se han tenido en cuenta los obstáculos, pero como se explica en ese apartado se puede utilizar la misma función para detectar obstáculos para los distintos caminos formados por los puntos imagen y en caso de estar “tapados” asignar a esos puntos imagen un punto suficientemente alejado donde la atenuación de la señal produzca que no tenga efecto.

4.2.3 Definición de nuevas salas

En este proyecto se ha trabajado con salas rectangulares (con y sin obstáculos) además de la sala IDIAP y la sala ISPACE, pero sería interesante probarlo en salas con otras configuraciones como por ejemplo la de la figura 4.1.

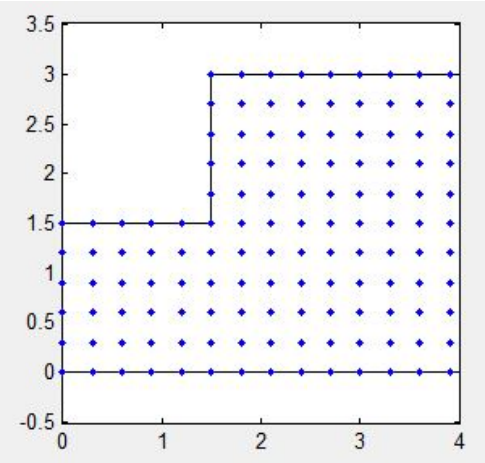


Figura 4.1: Sala con forma de L

4.2.4 Añadir nuevos tipos de sensores y topologías de arrays de micrófonos

Además de los sensores con localización basada en TDOA y los micrófonos usados en este trabajo podrían introducirse nuevos tipos de sensores con sus respectivas funciones de error de localización y las restricciones de ubicación que fueran necesarias (para algunas topologías de arrays de micrófonos es útil posicionarlos dentro del espacio de la sala).

4.2.5 Añadir posibilidad de visionar el funcionamiento en tiempo real de procesamiento

Ya que el sistema está desarrollado para almacenar el mejor resultado intermedio para cada 5 generaciones, podría extenderse esta capacidad para mostrar en la interfaz en tiempo real de procesamiento el funcionamiento del algoritmo genético.

Capítulo 5

Pliego de condiciones

Para la correcta utilización del sistema desarrollado en este trabajo, debe disponerse de un hardware y un software que cumpla unos requisitos mínimos.

5.1 Requisitos de Hardware

- Procesador 32/64 bits 2GHz o superior.
- 1 GB de memoria RAM o superior.
- Recomendado el uso de pantalla panorámica.
- Interface de red para enviar las simulaciones al cluster.
- Al menos 150 MB libres en el disco duro para las funciones y datos.

5.2 Requisitos de Software

- Windows 7 o superior.
- Matlab 2012a o superior
- Matlab Global Optimization Toolbox

Capítulo 6

Presupuesto

6.1 Costes de equipamiento

- Equipamiento hardware utilizados:

Tabla 6.1: Costes de equipamiento hardware

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal(€)**
PC Intel Core I5	1	500 €	500 €
Coste total HW			**500 €**

- Recursos software utilizados:

Tabla 6.2: Costes de recursos software

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal (€)**
Windows 8	1	500 €	500 €
Matlab 2015	1	2000 €	2000 €
Matlab Global Optimization Toolbox	1	1000 €	1000 €
Software LaTeX	1	0 €	0 €
Coste total SW			**3500 €**

6.2 Costes de mano de obra

Tabla 6.3: Costes debidos a mano de obra

Concepto	**Cantidad**	**Coste Unitario**	**Subtotal (€)**
Desarrollo SW	300	65 €/hora	19500 €
Mecanografiado y maquetado de este documento	100	15 €/hora	1500 €
Coste total HW			**21000 €**

El número de horas de ingeniería equivale a cuatro meses a media jornada. El mecanografiado y maquetado se ha estimado en 5 semanas a media jornada.

6.3 Coste total del Presupuesto

Tabla 6.4: Coste total del presupuesto

Concepto	**Subtotal (€)**
Costes de equipamiento hardware	500 €
Costes de recursos software	3500 €
Costes mano de obra	21000 €
Coste total HW	**25000 €**

El importe total del proyecto asciende a la cantidad de: *VENTIÚN MIL EUROS*

En Alcalá de Henares,___ de _____ de 20___.

Fdo: Roberto Macho Pedroso

Graduado en Ingeniería Electrónica de Comunicaciones

Bibliografía

[1] S. A. Zekavat and R. M. Buehrer, *Handbook of Position Location: Theory, Practice, and Advances*. IEEE, 2012.

[2] M. Brandstein and D. Ward, Eds., *Microphone Arrays : Signal Processing Techniques and Applications*. Springer, 2001.

[3] E. M. Gorostiza, J. L. Lázaro Galilea, F. J. Meca Meca, D. Salido Monzú, F. Espinosa Zapata, and L. Pallarés Puerto, "Infrared sensor system for mobile-robot positioning in intelligent spaces," *Sensors*, vol. 11, no. 5, pp. 5416–5438, 2011.

[4] A. N. Bishop, B. Fidan, B. Anderson, K. Doğançay, and P. N. Pathirana, "Optimality analysis of sensor-target localization geometries," *Automatica*, vol. 46, no. 3, pp. 479–492, 2010.

[5] C. Yang, L. Kaplan, and E. Blasch, "Performance measures of covariance and information matrices in resource management for target state estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 3, pp. 2594–2613, Jul. 2012.

[6] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT press, 1998.

[7] F. Domingo-Perez, J. Lazaro-Galilea, E. Martin-Gorostiza, D. Salido-Monzu, and A. Wieser, "Evolutionary optimization of sensor deployment for an indoor positioning system with unknown number of anchors," in *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2014*, Nov 2014, pp. 195–202.

[8] J. M. Shu Wang and B. K. Yi, "Location based services for mobiles: Techonogies and stardards," 2008, http://to.swang.googlepages.com/ICC2008LBSforMobilessimplifiedR2.pdf [Último acceso 07/julio/2015].

[9] E. Martín, "Sistema de posicionamiento absoluto basado en infrarrojos para el guiado de robots móviles en espacios inteligentes." Ph.D. dissertation, Alcalá, Alcalá de Henares, 2012.

[10] S. Chepuri and G. Leus, "Sparsity-promoting sensor selection for non-linear measurement models," *Signal Processing, IEEE Transactions on*, vol. 63, no. 3, pp. 684–698, Feb 2015.

[11] V. R. Lorenzo, "PFC - Diseño, Implementación y Evaluación de herramientas de simulación y entornos acústicos reverberantes: aplicación a sistemas de reconocimiento automático de habla," Ph.D. dissertation, UPM, 2006.

[12] D. Alonso, "Estado del arte sobre localización, seguimiento y estimación de pose de múltiples locutores usando fusión audiovisual," pp. 1–205, 06/2009 2009.

[13] J. Velasco, C. J. Martín-Arguedas, J. Macias-Guarasa, D. Pizarro, and M. Mazo, "Proposal and validation of an analytical generative model of srp-phat power maps in reverberant scenarios," *Signal Processing*, vol. 119, pp. 209–228, 08/2015 2015.

[14] "Página de documentación de mathworks para algorítmos genéticos," http://es.mathworks.com/help/gads/genetic-algorithm.html [Último acceso 03/julio/2015].

Apéndice A

Manual de usuario

A.1 Introducción

La GUI desarrollada en este TFG para MATLAB y las funciones necesarias para su uso han sido creadas para calcular la posición óptima de un conjunto de sensores en una habitación determinada por el usuario, con el fin de calcular la posición del hablante en dicha habitación.

A.2 Manual

A.2.1 Instalación

Para el uso de estas funciones es necesario tener instalado MATLAB en el equipo y seguir los siguientes pasos:

- Copiar el fichero "galincon.m" en la carpeta "C:/Program Files/MATLAB/VERSION/toolbox/globaloptim/globaloptim/private", los cambios realizados en dicha función permite obtener resultados intermedios del proceso de optimización. Es conveniente guardar la versión original del archivo aunque si no se hace no perjudica su uso para otras aplicaciones.

- Copiar la carpeta de las funciones en la raíz de "C:/"

A.2.2 Ejecución Interfaz

Iniciar la GUI a través de MATLAB, abriendo el fichero "Interface.m" y corriendo el código mediante F5 o "Run", o bien tecleando "Interface" por línea de comandos estando en la carpeta donde se copiaron los ficheros. Lo que arrancará la pantalla de inicio de la interfaz A.1.

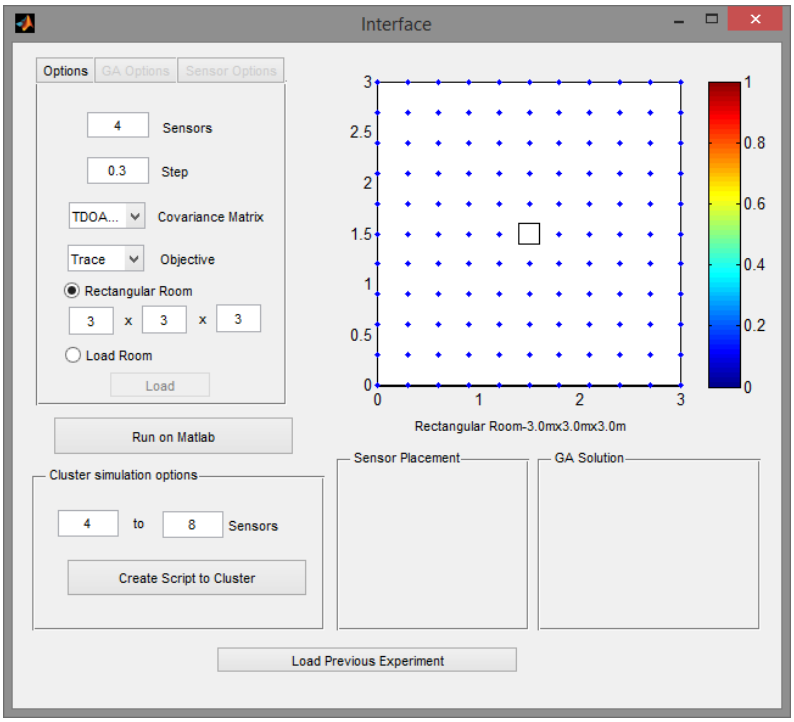


Figura A.1: Interfaz Gráfica

La interfaz de puede subdividir en 4 partes como en A.2: opciones (1), creación de script(2), visualización(3) y resultados (4).

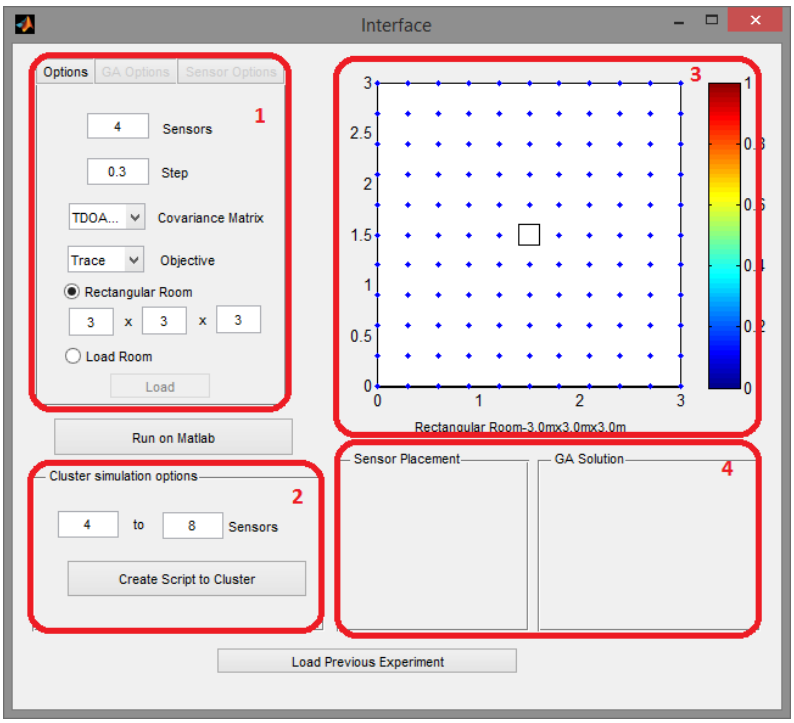


Figura A.2: Partes de la interfaz

A.2.2.1 Opciones

La parte de opciones tiene los siguientes submenús:

1. Submenú Options A.3: Permite cambiar las opciones básicas.

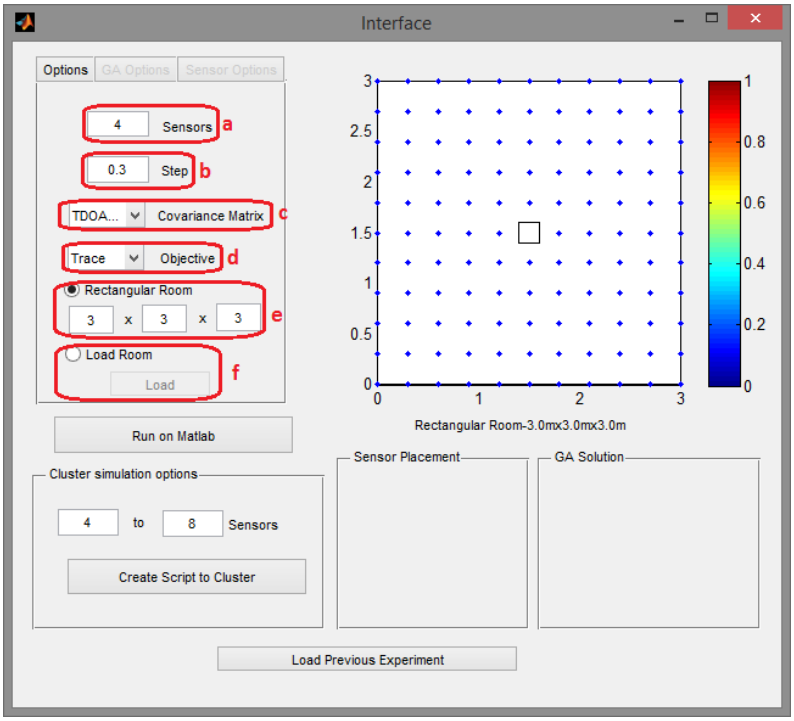


Figura A.3: Opciones del submenú Options

(a) Determina el número de sensores o arrays de micrófonos que se quieren posicionar. Dicho valor está limitado inferiormente a 4 para sensores aislados o a 2 para arrays de micrófonos.

(b) Determina la separación de la rejilla que se va a analizar. Dicho valor esta limitado a 0.1, 0.2 o 0.3 para arrays de micrófonos pero sin limitaciones para sensores aislados.

(c) Sirve para seleccionar el tipo de sensores y el tipo de matriz de covarianza usado.

- TDOA (eye-0.5): Selecciona sensores aislados con la matriz de covarianza A.1, que indica que el error es independiente de la distancia a los sensores.

$$\Sigma = \begin{pmatrix} 1 & 0{,}5 & \cdots & 0{,}5 \\ 0{,}5 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0{,}5 \\ 0{,}5 & \cdots & 0{,}5 & 1 \end{pmatrix} \tag{A.1}$$

- TDOA (sq. dist. dep.): Selecciona sensores aislados con la matriz de covarianza A.2, que indica que el error es dependiente al cuadrado de la distancia a los sensores.

$$\Sigma = \begin{pmatrix} dr^2 + ds_1^2 & dr^2 & \cdots & dr^2 \\ dr^2 & dr^2 + ds_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & dr^2 \\ dr^2 & \cdots & dr^2 & dr^2 + ds_{N-1}^2 \end{pmatrix} \tag{A.2}$$

- Isolated Microphones: Selecciona micrófonos como tipo de sensores para la localización basada en SRP.
- Microphone Array(SRP): Selecciona el tipo de sensor como arrays de micrófonos con topología de T invertida con localización basada en SRP, con o sin rebotes.

- Microphone Array(TDOA): Selecciona el tipo de sensor como arrays de micrófonos con topología de T invertida con localización basada en TDOA, sin rebotes.
  Todas la posibilidades cuando son seleccionadas activan o desactivan (según cual esté seleccionada) características de la interfaz que son relevantes o no para este ese de sensores con la localización seleccionada, como la distancia entre los micrófonos, la limitación del número de sensores o la activación o desactivación de la selección de objetivo.

(d) Determina el objetivo a minimizar por el algoritmo genético:

- Trace: Minimiza el valor medio de la Traza de la CRLB.
- Determinant: Minimiza el valor del determinante de la Traza de la CRLB.
- Max Trace: Minimiza el valor máximo de la Traza de la CRLB.

(e) Si está activo selecciona una habitación rectangular por defecto un cubo de 3 metros de lado con una columna cuadrada en el centro y establece las dimensiones X, Y y Z de la habitación rectangular.

(f) Carga una habitación definida en un fichero ".srf". En el que se define el nombre de la habitación, los vértices que la forman, las alturas y los obstáculos que hay en ella. Como el de la figura **??**

```
[ROOMInfo]

version = 1.1
comment = Idiap demo room

numVertexes = 4

roomFloorHeight = −730
roomCeilingHeight = 1670




[Vertexes]

vertex0 = 1800 2200 −730
vertex1 = 1800 −6000 −730
vertex2 = −1800 −6000 −730
vertex3 = −1800 2200 −730

[Obstacle 1]
v_obst0 = 600 800 −30
v_obst1 = 600 −4000 −30
v_obst2 = −600 −4000 −30
v_obst3 = −600 800 −30
obstacle1FloorHeight = −10
obstacle1CeilingHeight = −30
obstacle1Fixed = 0
obstacle1Name = table
```

Figura A.4: Ejemplo de archivo ".srf" de definición de sala

2. Submenú GA Options A.5: Permite modificar los parámetros del Algoritmo Genético

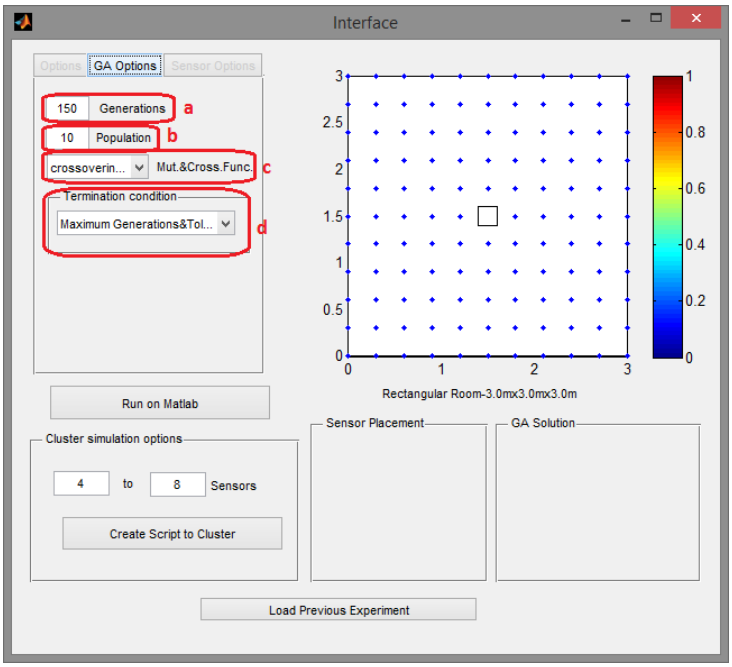


Figura A.5: Opciones del submenú GA Options

(a) Determina el número máximo de generaciones que alcanzará el algoritmo genético.

(b) Fija el tamaño de la población del algoritmo genético.

(c) Selecciona las funciones de Mutación y Cruce utilizadas por el algoritmo genético.

(d) Impone la condición de finalización del algoritmo genético.

3. Submenú Sensor Options A.6: Permite modificar los sensores.

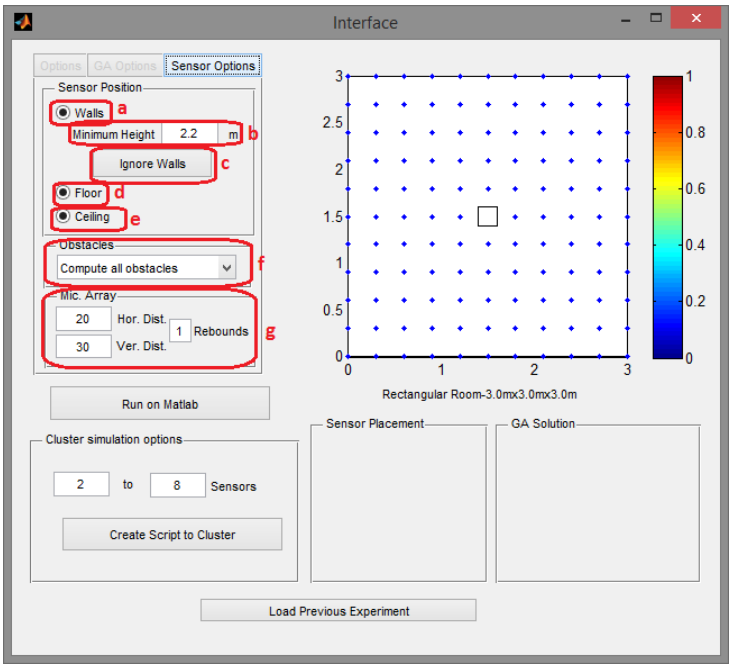


Figura A.6: Opciones del submenú Sensor Options

(a) Si esta seleccionado permite la ubicación de los sensores en las paredes.

(b) Determina la altura mínima de los sensores en las paredes.

(c) Permite seleccionar las paredes donde no se pueden ubicar los sensores.

(d) Si está seleccionado permite la ubicación de los sensores en el suelo.

(e) Si está seleccionado permite la ubicación de los sensores en el techo.

(f) Determina si se tienen en cuenta o no los obstáculos y cuales se pintan.

(g) Está desactivado para sensores aislados y sirve para determinar la separación entre micrófonos y el número de rebotes que se tienen en cuenta para el cálculo.

A.2.2.2 Cluster

Permite crear un script ".bash" ejecutable en el cluster para realizar la misma prueba con un distinto número de sensores de forma más rápida. Como el de la figura A.7

```
#!/bin/bash

# General definitions
MATLAB_PROGRAM=test

SWEEP_Step="0.30"
SWEEP_Matrix_Type="3"
SWEEP_n_sensors="3 4 5"
SWEEP_room="_ISPACE_room_definition_.mat"
SWEEP_Objective_Type="1"
SWEEP_Generations='200'
SWEEP_Population='50'
SWEEP_Functions='2'

Step=$SWEEP_Step
room=$SWEEP_room
Objective_Type=$SWEEP_Objective_Type
Generations=$SWEEP_Generations
Population=$SWEEP_Population
Functions=$SWEEP_Functions

for Matrix_Type in $SWEEP_Matrix_Type
do
        for n_sensors in $SWEEP_n_sensors
                do
                        echo "--------------------------------------"
                        echo "Submitting $MATLAB_PROGRAM for n_sensors=$n_sensors"
                        geintra_run_matlab $MATLAB_PROGRAM "Step=$Step;
                        room=$room;Matrix_Type=$Matrix_Type;n_sensors=$n_sensors;
                        Generations=$Generations;Population=$Population;
                        Objective_Type=$Objective_Type;Functions=$Functions"
                done
        done
done
```

Figura A.7: Ejemplo de archivo ".bash" de script para el cluster

A.2.2.3 Visualización

La parte de visualización tiene dos cometidos.

El primero es el de visualizar cómo es la habitación que vamos a tener en cuenta, los obstáculos que se tienen o no en cuenta y el tamaño de la rejilla. Como los ejemplos de A.8.

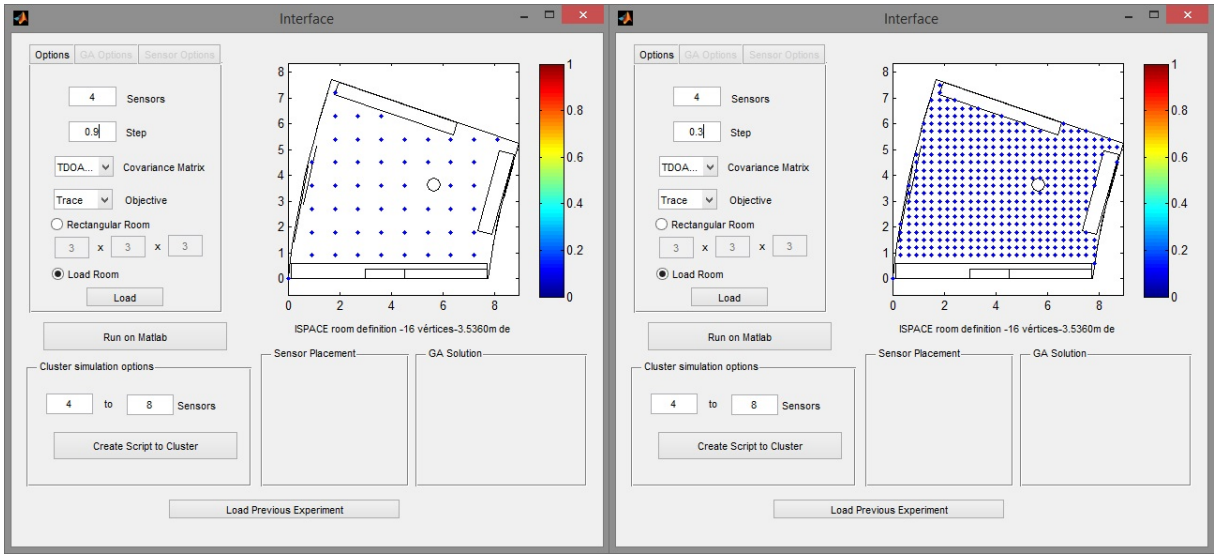


Figura A.8: Distintas rejillas para sala ISPACE

El segundo cometido es el de visualizar el resultado en dos dimensiones como en A.9. Dicho resultados también se ve en 3D en uno de los pop-up que aparecen cuando se finaliza la ejecución o cuando se carga un resultado anterior, como en la figura A.10, así como el la mayoría de las opciones de configuración aparecen en otro pop-up.

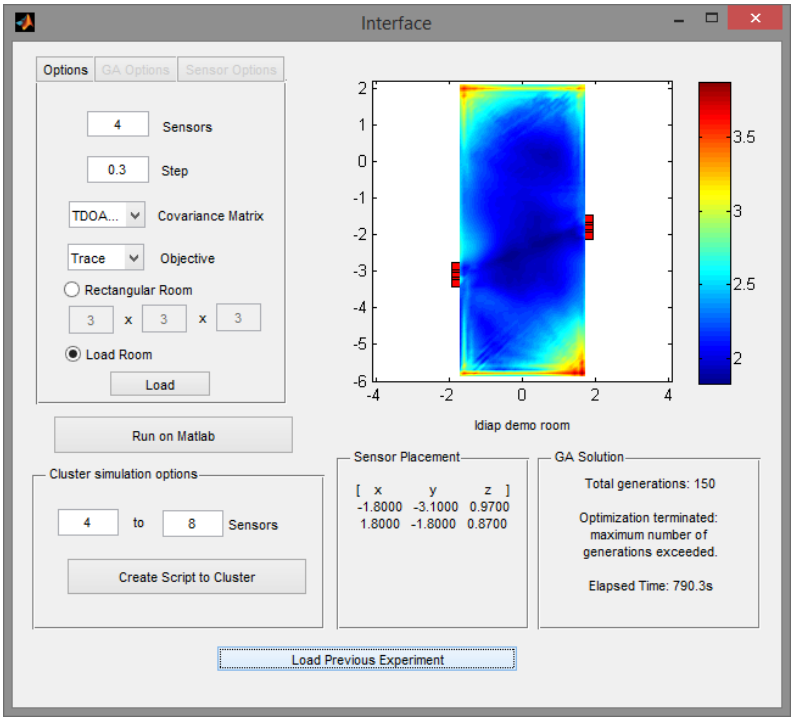


Figura A.9: Resultados

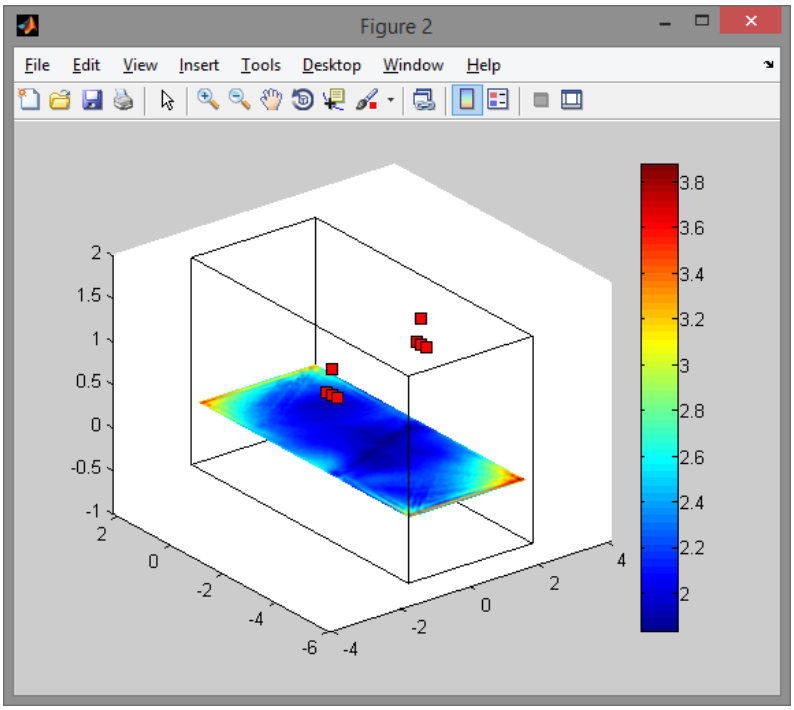


Figura A.10: Resultado 3D

A.2.2.4 Resultados

La parte de resultado se divide en dos: la que nos aporta la posición de los sensores (en el caso de los arrays de micrófonos nos aporta la posición del micrófono central) y la parte que nos aporta la solución del algoritmo genético. En la figura A.9 se puede ver un ejemplo de resultado.

La interfaz también permite visualizar resultados anteriores, para ello de forma automática guarda en el fichero "LastIntefaceResult.mat" la última solución obtenida y así se puede ir guardando dicho fichero con otro nombre para otras futuras visualizaciones

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá