Máster en Automática y Robótica
Guiado y Navegación de Robots

Simulación y guiado de robot móvil en un hogar de ancianos

Presentado por:

Carlos Prados
Andrea Fontalvo
Yao Hu

UNIVERSIDAD
POLITÉCNICA
DE MADRID

2018

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En la actualidad la planeación, guiado y control de robots móviles constituye una de las principales ramas de investigación en sistemas robóticos dado a los problemas de naturaleza probabilística que estos tipos de robots enfrentan. El interés de movilizar un vehículo o robot en un entorno desconocido trae consigo grandes fuentes de incertidumbre, tanto dependiendo de la naturaleza del entorno, como de la densidad de objetos y agentes presentes en las mismas; adicionalmente la incertidumbre puede verse afectada por los sistemas de percepción y tecnología empleada.

Con el interés de aplicar los conocimientos vistos en la asignatura de Guiado y Navegación de Robots en el marco del máster en Automática y Robótica referentes a lo mencionado anteriormente, se plantea una solución al problema de planeación, guiado y control de un robot móvil proponiendo un entorno y sistema robótico para tal fin, empleando la herramienta de simulación APOLO[1] y el entorno de programación MATLAB.

En el campo de la robótica de servicio, buscando el beneficio directo a los seres humanos en el apoyo de diferentes actividades cotidianas, los robots móviles encuentran un gran abanico de aplicaciones; son ampliamente usados en los hogares para tareas de limpieza, en entornos como museos o escuelas para vigilancia, así como también son usados en aplicaciones de asistencia a personas en hoteles y comercio, entre otras [2].

Siendo de interés las aplicaciones de robótica que puedan impactar de manera positiva directamente al ser humano, y tomando como foco a la población menos familiarizada a estos desarrollos tecnológicos como son las personas de la tercera edad, se propone la situación problema de la movilidad de un robot móvil de servicio de tipo asistencial en una residencia de ancianos para disposición de medicinas en las habitaciones. En la comunidad académica existe un gran interés en esta área, interes que proviene a que la población anciana se encuentra creciendo en una actualidad en donde los robots de servicio es cada vez más un campo desarrollado y explorado [3][4][5].

El entorno propuesto corresponde a un piso con dos áreas comunes, y tres habitaciones, delimitadas por paredes y pasillos, y todas con su respectiva mobiliaria, siendo estos los obstáculos del medio. En cuanto al robot a emplear, se propone el empleo de un robot móvil con locomoción por ruedas tipo diferencial; el sistema de percepción adherido estaría compuesto por tres sensores de ultrasonido dispuestos al frente del robot, y a los dos lados. En las secciones siguientes se detalla en lo concerniente a la construcción del entorno propuesto y la disposición de los sensores en el robot, así como el proceso de calibración de los mismos. Siguiente se explican los algoritmos de planificacion de la trayectoria, localizacion y control para la navegación en el entorno de interés.

2. ENTORNO

Para definir el mapa del entorno sobre el que se va a trabajar, al principio, se ha planteado un plano de la planta con la herramienta AutoCAD, dicha planta consta de tres habitaciones y dos salas, además, está amueblada. Como se muestra el plano por la Figura 1, la línea negra delinea toda el área residencial, la línea rosa se muestra los muebles de la residencia y los puntos rojos se indican las posiciones de las diez balizas, las cotas se presentan por tres colores.

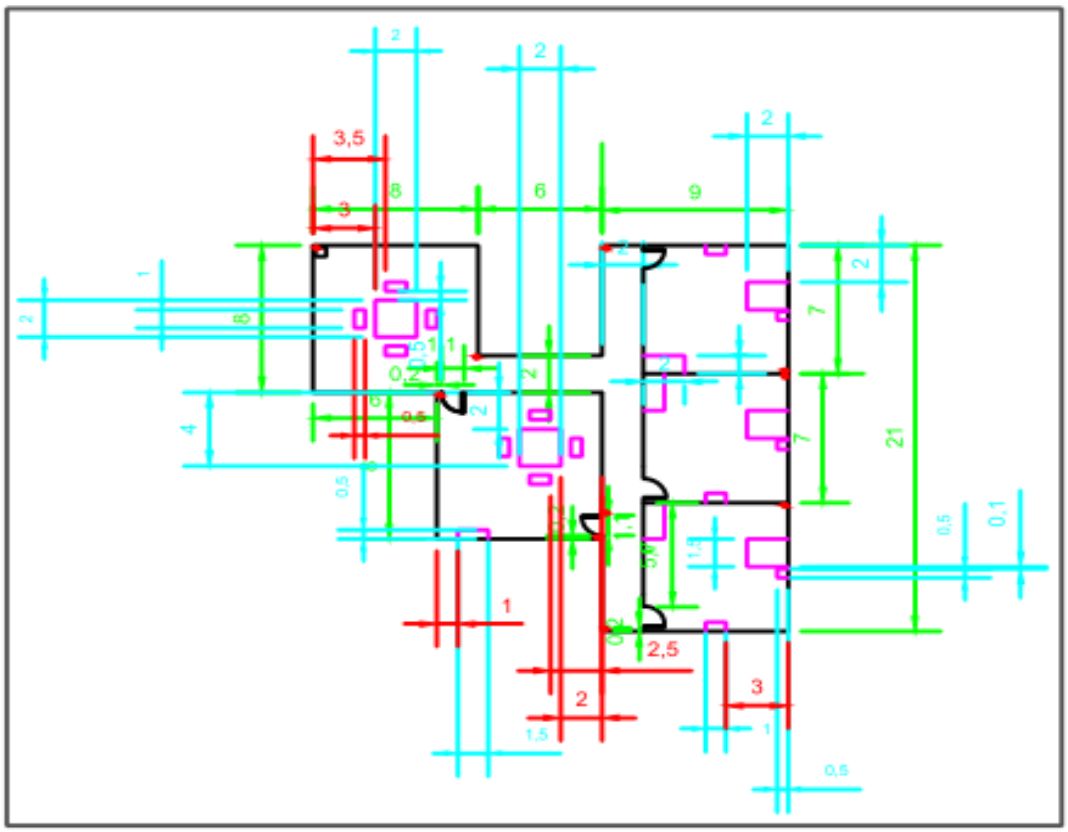

Figura 1. Mapa entorno acotado

Después, se define el mapa de contorno correspondiente al plano de la planta a través de un editor de texto. La Figura 2 se muestra el modelo del texto para definir un plano, con esta forma podemos definir el contorno completo.

```
<face>
      <vertex>  %Definir un plano con la forma definida,a partir del punto origen,el eje X positivo
                hacia derecha,el eje Y positivo hacia arriba,el eje Z positivo es perpendicular a la pantalla,hacia fuera
          {X1 ,Y1, Z1}
          {X2 ,Y2, Z2}
          {X3 ,Y3, Z3}
          {X4 ,Y4, Z4}
      </vertex>
      <orientation>{rotx, roty, rotz}</orientation >%Se rota dicho plano según eje X,Y,Z
      <position>{Xp, Yp, Zp}</position> %se coge un punto p como el punto origen de dicho plano
      <colour r="valor(0-1)" g="valor(0-1)" b="valor(0-1)"/>
  </face>
```

Figura 2. Definición de planos en simulador

Además, los sensores que hemos elegido son tres sensores ultrasonidos. Una vez terminado el archivo.xml, se carga este en la aplicación Apolo. Finalmente, el entorno se observa en la interfaz de simulación como se observa en la Figura 3.

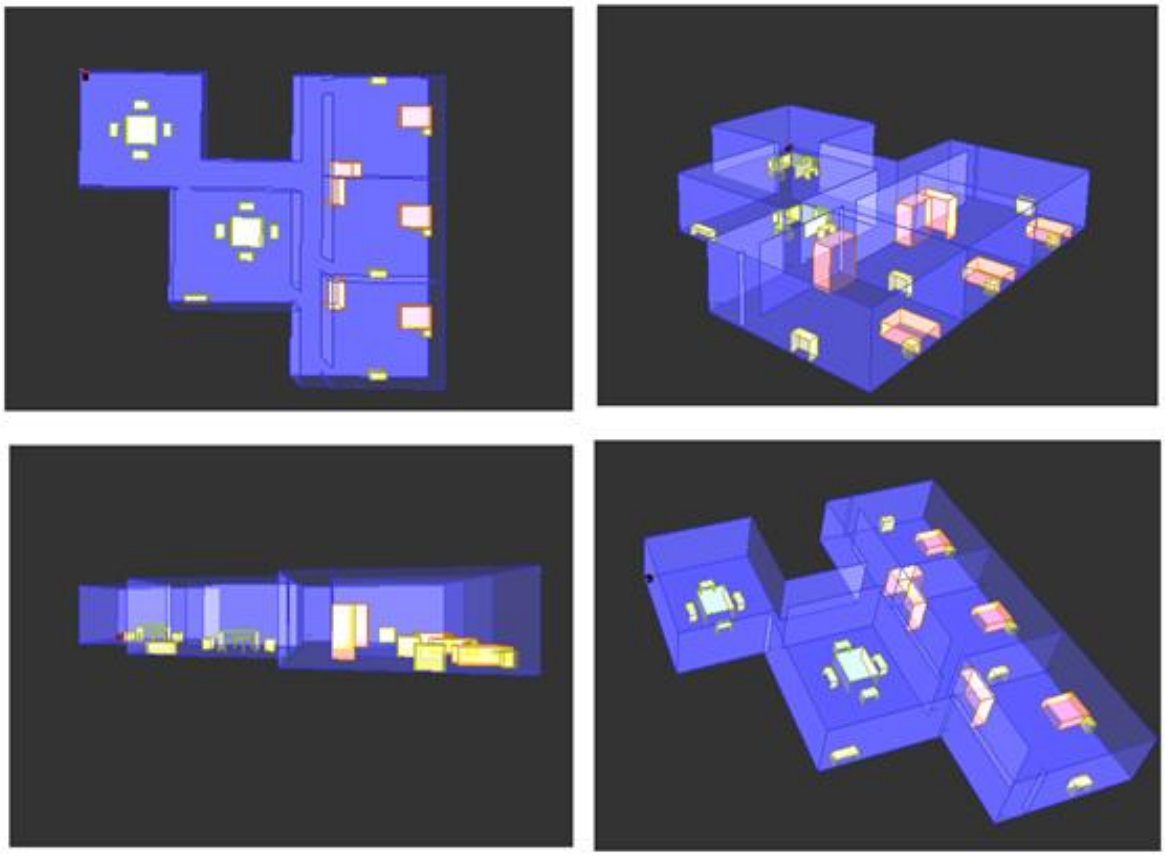

Figura 3. Entorno visto en Apolo

3. ROBOT “Elder”

El robot empleado corresponde a un Pioneer 3AT disponible como robot móvil en la plataforma de simulación de Apolo; al tratarse de un robot para servicio tal opción cabe dentro del diseño del sistema para el problema planteado en la residencia de ancianos. Para la distinción del robot en la plataforma de simulación, se ha denominado este como “Elder”.

3.1. Sensores

Previa revisión bibliográfica [5][6][7], se encontró que el enfoque en el área de la percepción del sistema de navegación propuesto, puede conformarse por la *odometría* que la naturaleza del robot móvil permite acceder; también se estudio la posibilidad de emplear sensores tipo laser (LIDAR) [8], sin embargo, para la aplicación en cuestión se descarta puesto que, si bien seria de utilidad, seria un recurso de muy alto gasto computacional para la problemática a resolver; si se tratase de un problema de mapeado, el empleo de un LIDAR seria una buena opción de tecnología a emplear, como se encuentra en la literatura. Dada la revisión hecha, y observando las necesidades de la situación propuesta, se decide emplear en conjunto con el sistema odométrico del robot, *sensores de ultrasonido* para detectar paredes y obstáculos a los lados derecha, izquierda y frontal del robot móvil, para evitar colisiones durante la navegación.

3.2. Calibración

Los sistemas de percepción a emplear tanto propioceptivos como exteropceptivos, requieren de calibración para el correcto funcionamiento de los algoritmos. La respectiva calibración se da a partir de definición de las matrices R y Q, las cuales describen a las matrices de

covarianza de los sensores y del sistema odométrico, respectivamente. Así, el proceso de calibración correspondió a la definición de estas matrices a través de la recolección de datos y tratamiento en MATLAB.

El primer paso para la calibración de los sensores fue la ubicación de estos en el robot en la herramienta de simulación Apolo. Posterior, se definió un entorno rectangular en Apolo, y un algoritmo en MATLAB en donde se llevaron a cabo repeticiones de mediciones, recolección de los datos y tratamiento de estos para la correcta definición de R y Q.

La matriz R, correspondiente a la covarianza de los sensores empleados, en este caso 3 sensores de ultrasonido, se construyó tomando medidas de estos tres sensores a una distancia estipulada (2 metros) repetidas veces dentro del entorno de calibración generado en Apolo. La dimensión de esta matriz R depende del número de sensores, resultando de 3x3. Se encontró, durante el proceso de la construcción de esta matriz, que la definición de la ubicación de los sensores en Apolo afectaba los resultados, de manera que cuanto mas preciso el numero que configuraba la dirección de cada uno, la medida se veía afectada, es decir, el numero de cifras decimales empleadas en la ubicación del sensor en apolo tiene efecto en que tan precisa es la medida que proporciona. Teniendo esto en cuenta, y buscando un entorno de simulación con similitud a uno real, se estipuló usar solo 2 cifras decimales. Entre mas cifras usadas en la ubicación del sensor en apolo, menor se observaba la varianza de los sensores. La varianza de cada sensor (diagonal de R) resultante de estos experimentos y empleada en el algoritmo de localización fue finalmente como se muestra en la Tabla 1.

Varianza sensor derecha	5.3718e-10
Varianza sensor izquierda	5.3797e-10
Varianza sensor frontal	2.4403e-30

Tabla 1. Varianza sensores de ultrasonido

En cuanto a la matriz Q, correspondiente a la covarianza del sistema odométrico en este caso, se llevaron también a cabo experimentos en el entorno de prueba mencionado anteriormente, en donde se recopilaron los datos con los respectivos comandos de odometría. En esta matriz, la diagonal corresponde a la varianza de la velocidad lineal y la varianza de la velocidad angular obtenida en las pruebas llevadas a cabo. La dimensión de esta matriz es de 2x2, con las varianzas obtenidas mostradas en la Tabla 2.

Varianza velocidad lineal	1.8494e-07
Varianza velocidad angular	1.2229e-06

Tabla 2. Varianza odometría

4. PLANIFICACIÓN

En este apartado, se va a programar un planificador capaz de generar una trayectoria válida en el mapa del entorno escogido. El entorno, que describe una residencia de ancianos, dispone de tres habitaciones, que corresponde cada una a un anciano distinto. Cada uno de los ancianos tiene en su cama un pulsador para solicitar que el robot “Elder” acceda a su habitación, de tal manera que se genere una trayectoria válida entre la posición actual del robot y la habitación solicitada, o home.

Para planificar la trayectoria entre los diferentes puntos requeridos del mapa, se ha decidido realizar una planificación basada en muestreo. El algoritmo de planificación implantado ha sido el PRM (Probabilistic Roadmaps) que consiste en la generación de un mapa de carreteras en Cfree. El mapa generado determinará las trayectorias a seguir en posteriores planificaciones.

El mapa de carreteras generado corresponde a un grafo, de tal manera que se puede implantar un algoritmo de búsqueda discreta sobre un espacio que es completamente continuo Cfree.

4.1. Generación del roadmap

El algoritmo PRM es especialmente eficiente en planificadores múltiple-query, por lo que va a ser útil para nuestra aplicación, donde el punto de partida y el punto destino no tienen por qué ser el mismo. Además, solo es necesario generar el roadmap una única vez, ya que el resto de las veces se hará uso de dicho roadmap para la generación de trayectorias.

Los nodos, o vértices, del grafo se corresponden con las configuraciones del robot escogidas de forma apropiada dentro del entorno Cfree, es decir, posiciones en el mapa libres de colisión, donde el robot no choca con ningún objeto de la residencia de ancianos, ni con las paredes de esta.

Las aristas del grafo se corresponden con las trayectorias sencillas entre dos nodos libres de colisión, es decir, que la unión entre dos nodos no pase por una posición del mapa donde exista colisión.

Inicialmente, se ha generado un plano donde se muestran los diferentes objetos de la residencia y su localización en un plano visto desde arriba. En dicho plano (*Figura 4*) se pueden observar en negro las paredes, en rojo las balizas y en rosa los diferentes objetos con los que no debe colisionar el robot.

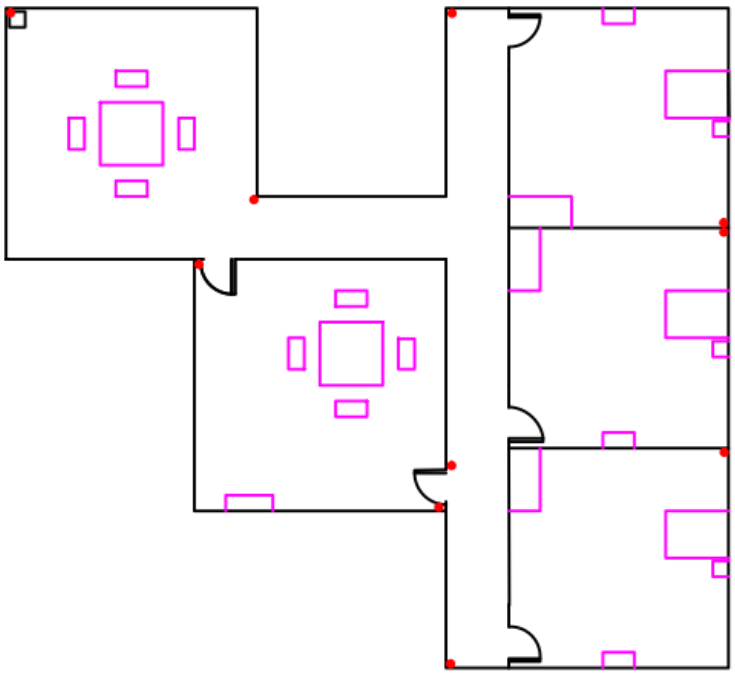

Figura 4. Plano de la residencia de ancianos (sin cotas).

Partiendo del plano de la residencia, se ha generado un mapa de ocupación de celdilllas que indica la posición de los diferentes objetos. Las puertas y balizas se han suprimido, mientras que las paredes y los objetos pertenecientes a Cobs se han engordado 30 cm para que el robot móvil se represente únicamente con una celdilla en el mapa (ancho del robot de 40cm), y de esta manera evitar choques. El resultado, tras diferentes operaciones basadas en técnicas de visión por computador, se puede observar en la *Figura 5*, donde las partes en negro corresponden a Cobs, mientras que las partes en blanco corresponden a Cfree.

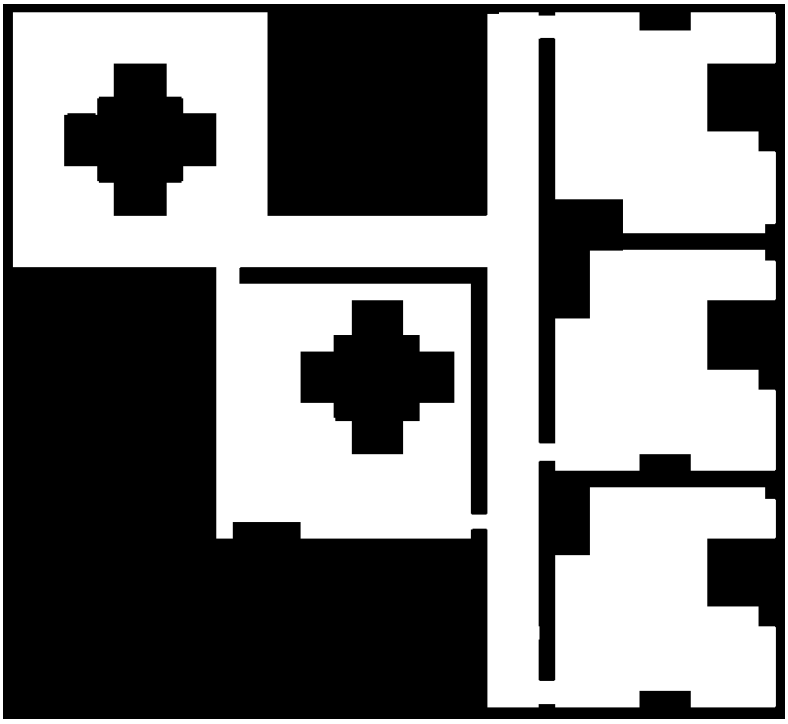

Figura 5. Espacio C de la residencia de ancianos.

El siguiente paso es realizar un muestreo sobre el espacio de las configuraciones de la residencia. En este paso, se van a generar N posiciones aleatorias sobre el espacio Cfree (cada una de ellas corresponde a un nuevo nodo) y se van a almacenar todos estos nodos en un vector denominado “grafo”, cuyo contenido se almacena en forma de una clase denominada “Vértice”.

Esta clase se ha desarrollado a medida para nuestra aplicación y tiene como elementos de la clase variables útiles para la definición de un grafo, como número de identificación, vértices adyacentes, pesos de los vértices adyacentes, padre de la búsqueda, distancia del origen, etc. Esta clase también tiene una función de inicialización, donde se inicializa el número de identificación, los vértices y pesos adyacentes y la posición del punto al que representa en el espacio C.

El número de muestras realizadas N se puede variar, pero un valor comprobado que aporta buenos resultados es N igual a 200.

Además, existen zonas donde hay problemas para la unión entre habitaciones, como son las puertas, o la unión entre pasillos. En estas zonas, se han generado un número M de muestras aleatorias para garantizar la unión entre los departamentos. El número M se puede variar, pero un valor comprobado que aporta buenos resultados es M igual a 5.

Un ejemplo se muestra en la *Figura 6* donde se muestra el departamento superior izquierdo y la puerta de unión a el departamento central. Se puede observar una serie de puntos generados aleatoriamente (rojos) pertenecientes a los N vértices, y una serie de puntos generados aleatoriamente (azules) pertenecientes a una de las secciones críticas del espacio Cfree.

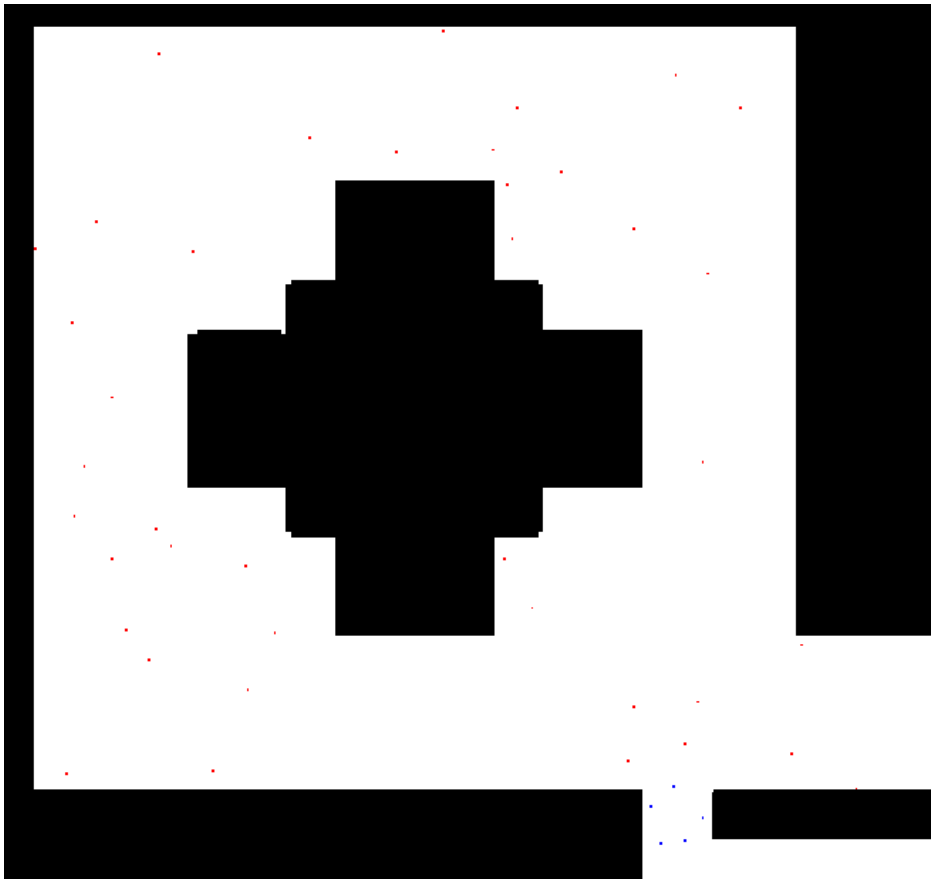

Figura 6. Ejemplo de muestreo en el espacio de las configuraciones.

El siguiente paso sería generar las aristas libres de colisión, con el objetivo de generar la huella del grafo, es decir, la red de carreteras. Para ello se ha desarrollado una función denominada “Generador_Aristas”, que, partiendo del grafo existente, hace una comprobación con el resto de los nodos del grafo.

Para cada nodo, comprueba la distancia **euclídea** que existe entre ambas muestras. Se dispone de un vector (en la clase “Vertice”) de tamaño K (se ha establecido a valor 10), que almacena los vértices más cercanos y la distancia a los mismos. De esta manera, se comprueba si entre un punto y otro se pasa por Cobs.

Si dos vértices se encuentran a una distancia menor que las distancias almacenadas previamente en el vector de la clase “Vertice”, entonces se realiza la comprobación, en caso contrario no.

La forma de comprobar que en la unión entre dos vértices se pasa por Cobs, es a través de una interpolación entre ambos vértices, avanzando una posición en el mapa de celdillas en cada paso de la interpolación, sin dejar ninguna celdilla intermedia entre dos puntos sin explorar. De una manera más gráfica se muestra en la *Figura 7,* donde se puede observar que en la dirección de mayor incremento se va explorando con un paso de un píxel, mientras que en la dirección de menor incremento se explora con un paso menor, redondeando al píxel más cercano.

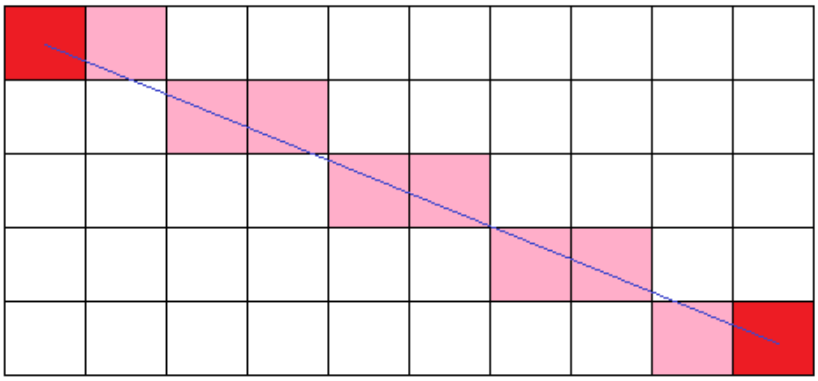

Figura 7. Interpolación en la comprobación de aristas libres de colisión.

Cuando uno de los píxeles indica que esa configuración pertenece a Cobs se detiene la interpolación y la unión entre los dos vértices asociados no es válida.

El resultado es un vector para cada vértice de los vértices adyacentes más cercanos y su distancia euclídea, es decir, las aristas libres de colisión que unen a los vértices más cercanos. En resumen, ya se dispone del mapa de carreteras donde existen un número N+M vértices en el espacio Cfree, unidos por un número máximo de K aristas libres de colisión salientes por vértice.

El siguiente paso es incluir las posiciones inicial y final de la trayectoria. La posición inicial queda determinada por la posición actual del robot, que inicialmente será en la esquina superior izquierda. La posición final, sin embargo, dependerá de las solicitudes por parte de las distintas habitaciones.

Una vez añadidos los puntos inicial y final, y añadidos al vector “grafo”, se deberán generar las aristas libres de colisión de tales puntos, es decir, añadir los vértices al mapa de carreteras. Este proceso se hace de una manera idéntica al caso anterior.

En resumen, tenemos un mapa de carreteras unido a los puntos inicial y final, de tal manera que es necesario encontrar la ruta que une ambos de la mejor forma posible, minimizando el recorrido del robot.

4.2. Búsqueda discreta de la trayectoria

Para encontrar el camino a seguir entre el nodo inicial y el nodo final del grafo, se ha implantado un algoritmo A*. Este algoritmo es una combinación entre búsquedas BFS y DFS. Utiliza una función de evaluación para explorar el grafo, teniendo en cuenta el costo de un camino óptimo entre el nodo inicial y un nodo dado, y una estimación del costo de un camino óptimo entre el nodo dado y el nodo final. Además, es heurístico, informado, completo y sistemático.

El algoritmo inicialmente inicializa todos los nodos y posteriormente realiza la exploración. La exploración consiste en elegir al mejor nodo según la función de evaluación, siendo el costo entre el nodo inicial y el actual la distancia óptima hasta dicho punto, y siendo la estimación del costo entre el actual y el final la distancia euclídea.

Una vez que se tiene el camino óptimo hacía todos los nodos, se hace un backtraking desde el nodo final hasta el inicial (teniendo en cuenta al de cada nodo padre). El resultado se puede observar en la *Figura 8*, donde se muestran diferentes ejemplos.

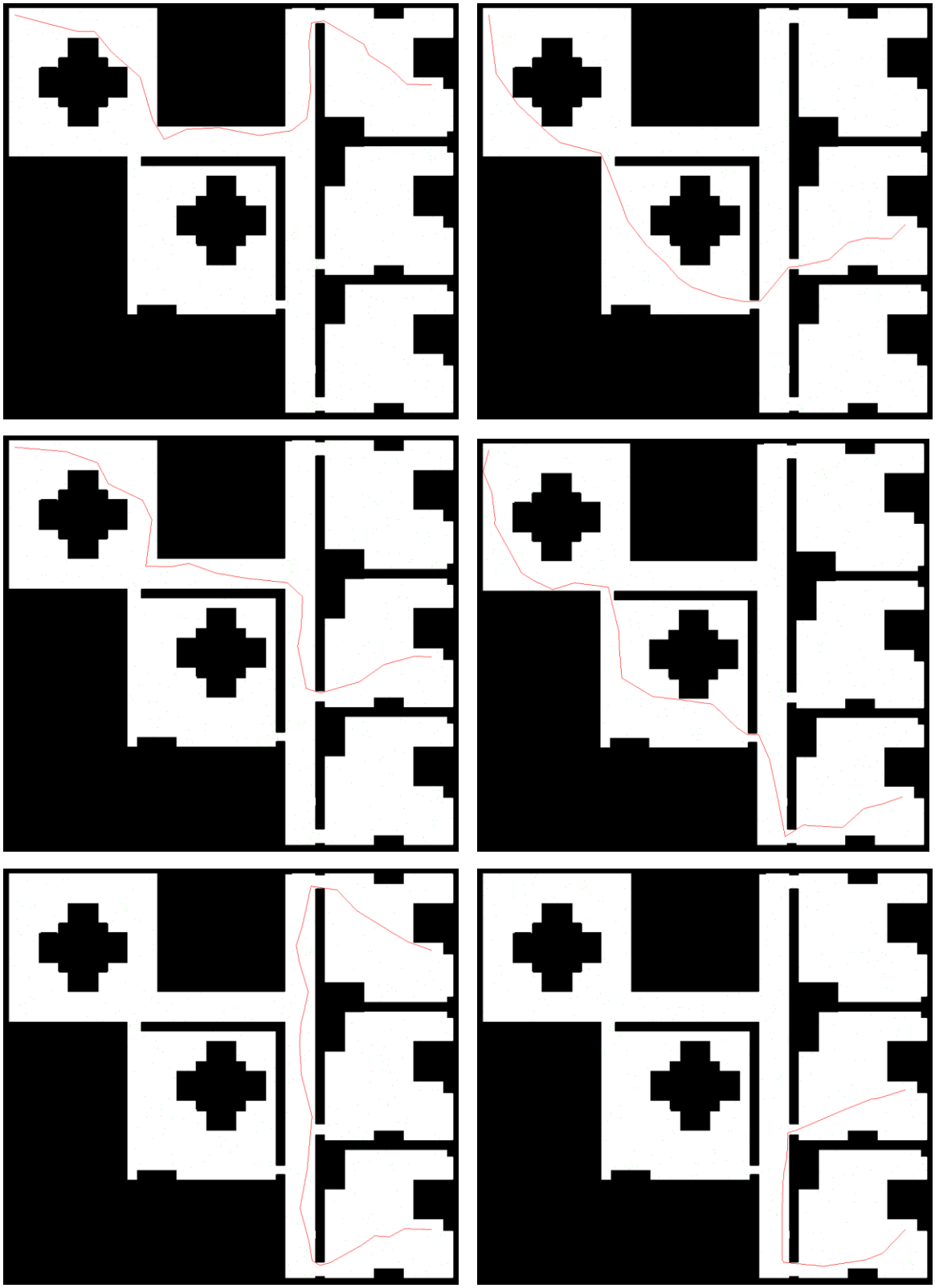

Figura 8. Ejemplos de trayectorias generadas entre diferentes habitaciones y entre la posición inicial (HOME) y las diferentes habitaciones.

En resumen, nuestro programa genera un roadmap (en caso de no existir) gracias a un mapa de ocupación almacenado en una imagen. Para la generación de la trayectoria, en primer lugar, se asigna un vértice del grafo (el más cercano a la posición estimada del robot) como la posición de inicio, es decir, el nodo fuente. En segundo lugar, se solicita al usuario el

destino, es decir, el nodo destino (0-home, 1-Habitación de arriba, 2-Habitación del medio, 3-Habitación de abajo). Posteriormente, se realiza una búsqueda heurística basándose en el algoritmo A* para encontrar el camino más rápido a través del roadmap entre la posición inicial y final, obteniendo una serie de punto intermedios, cuya posición en el mapa de la residencia se conoce.

5. LOCALIZACIÓN

En cuanto a la localización, se empleó el algoritmo correspondiente al filtro de Kalman extendido mediante un enfoque odométrico. Teniendo en cuenta las matrices que describen y modelan la varianza del sistema de percepción del robot en cuestión, se llevo a cabo, previo a incorporación con la planificación y control, un proceso de ajuste de los parámetros que permiten el funcionamiento y estimación de la localización.

El algoritmo de localización en sí toma las mediciones que aporta el sistema de odometría del robot Elder, luego lleva a cabo la respectiva estimación del paso próximo teniendo en cuenta esta información previa; siguiente a esto, lleva a cabo una comparación del valor dado por el robot (simulador) y aquel proporcionado por el algoritmo en si, y finalmente la corrección teniendo en cuenta umbrales para las tres variables de interés que conforman las variables de estado de interés*: posición en X, posición en Y, y ángulo.*

Describiendo una trayectoria de prueba circular, empleando el mismo entorno de prueba mencionado en el apartado de calibración, se observó la incidencia de la matriz de covarianza del sistema y su influencia en la etapa de estimación de la posición y así como también la incidencia y efecto de la matriz H para correcto funcionamiento del algoritmo. Cuando estos parámetros no se encontraban sintonizados con el entorno de simulación, las salidas en cuanto a localización no lograban proporcionar una estimación correcta para la situación.

El funcionamiento del algoritmo de localización se centra en la estimación de la posición del robot. En primer lugar, se cargan los valores obtenidos en la calibración, es decir, las varianzas de los sensores y de la odometría.

Cada vez que el robot se mueve se hace una llamada al algoritmo de localización. Este algoritmo, partiendo de la posición previa, realiza una estimación de la posición del robot a través de la odometría. Con esta posición estimada del robot, se calcula una estimación del valor aportado por los sensores, gracias a la matriz H.

A su vez, observa el valor aportado por los sensores Sonar. El valor que nos aportan los sensores y el valor que se ha estimado, se comparan, dando como resultado una corrección de la estimación de la posición, dando más importancia a la de menor incertidumbre (entre odometría y sensorización).

El resultado del algoritmo es, por tanto, una estimación de la posición. Esta estimación es aportada al controlador del robot, que decide las acciones a realizar sobre el robot.

6. CONTROL Y RESULTADOS

En este apartado, se va a programar un controlador capaz de seguir una trayectoria válida generada por el planificador, en el mapa del entorno escogido. Para ello, se parte de una serie de puntos en el mapa por los que el robot debe pasar.

El siguiente paso es, por tanto, pasar secuencialmente por todos los puntos del camino. Para ello, tomando la posición exacta del robot y la posición precisa del siguiente punto del camino, se genera un ángulo de orientación en el que debe posicionarse el robot. Este ángulo se compara con la orientación actual del robot y se genera una velocidad angular (en el sentido

en el que el tiempo de giro sea mínimo), hasta que la orientación del robot y la orientación hacia el siguiente punto coincidan (con un determinado umbral).

Una vez orientado el robot hacia el siguiente punto del camino, se genera un avance del robot hasta que el mismo. Para ello se calcula la distancia hasta el punto y se compara con un umbral, es decir, el punto de destino tiene un circulo de aceptación de R≈5cm.

Cuando se llega al siguiente punto del camino, se realiza el mismo proceso, es decir, girar el robot en la orientación adecuada y avanzar hasta el siguiente punto.

Un ejemplo del funcionamiento del sistema se puede encontrar en la siguiente página web: https://www.youtube.com/watch?v=-ESwtUmP2zg&t=20s. Donde se hace uso de la posición real aportada por el simulador de Apolo, es decir, la posición exacta del robot sin incertidumbre. Se pueden observar diferentes aspectos:

- El *roadmap*, y por tanto el grafo, solo se generan una única vez (el sistema tarda aproximadamente 5 segundos en generarlo, con 400 muestras, en la primera ejecución del programa).
- A pesar de detener al robot antes de llegar al destino, la cancelación del destino fijado y el establecimiento de uno nuevo hace que se genere una nueva trayectoria. Esto se debe a que la posición inicial se toma de acuerdo con la posición actual del robot.
- El robot sigue fielmente la trayectoria generada y mostrada en la imagen ya que no existe incertidumbre en la posición del robot.

Seguido a parametrización de las variables influyentes en el filtro de Kalman, se procedió a incorporar la estimación de la posición con los algoritmos de control para seguir la trayectoria definida previamente.

Cuando se hace uso de la posición estimada por el filtro de Kalman (sensorización y odometría), el resultado es muy similar. Sin embargo, la dinámica de trabajo es diferente. Al igual que en el caso anterior, se conoce la serie de puntos a seguir gracias al planificador de trayectorias.

El paso de rotación, donde se orienta al robot hacia el siguiente punto es similar, cambiando la posición exacta del robot por la posición estimada por el filtro de Kalman.

Una vez orientado el robot hacia el siguiente punto del camino, se genera un avance del robot hasta que el mismo. Para ello se calcula la distancia hasta el punto y se compara con un umbral. A diferencia del caso anterior, se ha desarrollado una metodología para garantizar que se llega al siguiente punto del camino.

Es decir, una vez que el robot queda orientado hacia el siguiente punto del camino se avanza. Durante el avance, debido al error en la odometría, el robot puede desviarse de la trayectoria deseada. De esta manera, durante el avance del robot se comprueba si la orientación hacia el siguiente punto es la correcta. En caso negativo, se genera una velocidad angular que posicionará al robot en la trayectoria correcta.

Cuando se llega al siguiente punto del camino, se realiza el mismo proceso, es decir, girar el robot en la orientación adecuada y avanzar hasta el siguiente punto.

Un ejemplo del funcionamiento del sistema, cuando tiene funcionando al filtro de Kalman para la localización del robot, se puede encontrar en la siguiente página web: https://www.youtube.com/watch?v=0JhDVC0NiL4&feature=youtu.be. Donde se hace uso de la posición estimada (con incertidumbre). Se pueden observar diferentes aspectos:

- El *roadmap*, y por tanto el grafo, solo se generan una única vez (el sistema tarda aproximadamente 5 segundos en generarlo, con 400 muestras, en la primera ejecución del programa), al igual que en el caso anterior.
- El funcionamiento es idéntico al caso anterior.

- El robot no sigue exactamente la trayectoria deseada, ya que la posición estimada no es exacta (se puede observar una de las figuras al final del video y otro ejemplo en la *Figura 9*). Sin embargo, la estimación de la posición es extremadamente buena, obteniendo valores muy reducidos de las varianzas.
- Durante la trayectoria entre dos puntos del camino, el movimiento del robot no sigue una línea recta, sino que durante dicha trayectoria se corrige la orientación del robot para garantizar que se llega al siguiente punto del camino con un rango de aceptación de radio 5cm.
- Cuando el robot llega al destino deseado, se muestran imágenes representativas de las posiciones por las que pasa el robot y los valores de las incertidumbres en las posiciones.

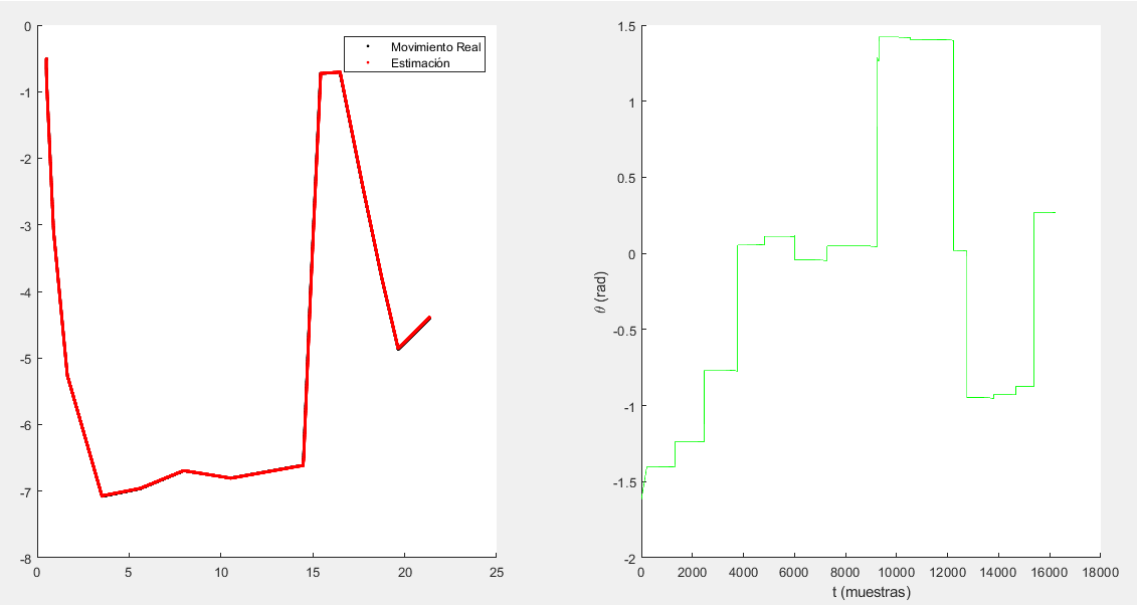

Figura 9. Ejemplo de los resultados de la navegación del robot móvil entre la posición "Home" y la habitación 1. En la figura de la izquierda se muestra la trayectoria seguida por el robot (real y estimada) y en la figura de la derecha las orientaciones del robot a lo largo de la trayectoria.

7. CONCLUSIONES

Finalmente, se ha conseguido realizar una correcta estimación de la posición en el mapa del entorno desarrollado gracias al filtro del Kalman extendido. Además, el planificador inicialmente genera un roadmap que permite generar trayectorias óptimas entre dos puntos con un tiempo mínimo.

Ante un fallo en el sistema, el localizador, junto al planificador, sabrán la posición del robot, generando una trayectoria entre el punto en el que se encuentra el robot y el punto de destino deseado.

El controlador garantiza que el robot pase por todos los puntos del camino generado por el planificador, ya que, ante una desviación en la orientación, el robot girará posicionándose de manera frontal al siguiente punto del camino.

El mapa de ocupación ha sido generado de tal manera que, ante pequeños fallos en la estimación de la posición y ante el giro del robot, no se produzcan choques ni con paredes ni con obstáculos del entorno.

El filtro de Kalman es inteligente, por lo que da más importancia a la odometría o a la sensorización según la incertidumbre de cada uno de ellos, garantizando una buena localización en el entorno.

8. EJECUCIÓN DEL PROGRAMA

En el archivo zip enviado se pueden observar los siguientes archivos:

- Ocupación.png: Imagen que indica el mapa de ocupación del entorno.
- Trayectoria.png: Imagen que muestra el muestreo (puntos en verde) y la trayectoria a seguir por el robot (en rojo).
- Vertice.m: Clase para la utilización del grafo.
- Q y R: Matrices características del filtro de Kalman.
- Puesta_a_cero.m: Se debe ejecutar en caso de fallo en la búsqueda de trayectoria o en caso de choque del robot contra algún obstáculo. Resetea la posición estimada del robot y la odometría de Apolo, localizando al robot en su posición inicial.
- Pos_Kalman.m: Función que al ser llamada realiza, mediante el filtro de Kalman extendido, una estimación de la posición del robot.
- Controlador.m: Encargado de realizar el control del robot para el seguimiento de la trayectoria definida. Hace llamadas al filtro de Kalman y genera comandos del movimiento al robot.
- Planificador.m: Planificador de trayectorias. Es el encargado de solicitar la posición de destino deseada, generando una trayectoria desde la posición actual del robot y la misma, buscando la mejor ruta como ya se ha explicado. Hace una llamada al controlador para aportarle la serie de puntos de la trayectoria a seguir y que este genere comando para llegar al destino.
- Muestra_Camino.m: Muestra la trayectoria generada en la imagen “Trayectoria.png”.
- Generador_Aristas.m: Genera el grafo. Partiendo de una serie de vértices generados por muestreo, une a los más cercanos cuya unión pasa por Cfree.
- Practica GUIADO.xml: Descripción del entorno y del robot ‘Elder’.

Para poder ejecutar el programa hay que ejecutar en el archivo “Planificador” y se indicarán el nodo actual y la serie de nodos del grafo a seguir para alcanzar el destino indicado.

9. TRABAJO REALIZADO POR CADA COMPONENTE

Yao Hu: Desarrollo del archivo .xml, donde se describe el entorno, el robot y la sensorización anidad al mismo. Desarrollo de un entorno en CAD que describe la residencia de ancianos para la posibilidad de desarrollar un mapa de ocupación, es decir, desarrollo de un mapa en 2D del entorno. Desarrollo de la parte del informe correspondiente.

Andrea Fontalvo: Calibración de los sensores y del sistema de locomoción (odometría), es decir, cálculo de las matrices de ruido de los diferentes parámetros del filtro de Kalman. Ajuste de los parámetros del filtro de Kalman para su puesta en funcionamiento. Desarrollo de la parte del informe correspondiente al trabajo realizado, incluyendo una sección del filtro de Kalman.

Carlos Prados: Desarrollo del mapa de ocupación partiendo de un mapa en 2D del entorno. Desarrollo del planificador basado en muestreo, del algoritmo de búsqueda heurística A* y de pruebas de su funcionamiento del planificador de trayectorias. Desarrollo del controlador que permite que el robot vaya pasando por una secuencia de puntos de una trayectoria previamente definida por el planificador de trayectorias (sin control reactivo). Integración del filtro de Kalman con el controlador y el planificador para su correcto funcionamiento y ajuste final de los parámetros del filtro de Kalman para conseguir una correcta estimación de la

posición. Testeo del funcionamiento del sistema integrado y ajuste final de parámetros. Preparación de un demostrador final que integre todos los componentes anteriores. Desarrollo de una guía de funcionamiento para la ejecución de la simulación en Apolo y descripción de los archivos desarrollados. Desarrollo de la parte del informe correspondiente al trabajo realizado, incluyendo una sección del filtro de Kalman, y de las conclusiones finales.

Trabajo desarrollado conjuntamente: Planteamiento del problema de la navegación de un robot terrestre, concluyendo como aplicación la asistencia en una residencia de ancianos. Elección de los sensores y sistema de locomoción (definido previamente). Planificación del trabajo a desarrollar por cada uno de los componentes.

REFERENCIAS

[1] UPM CAR, “APOLO V1.0.” .

[2] N. Chivarov, D. Chikurtev, M. Pleva, and S. Ondas, “Exploring Human-Robot Interfaces for Service Mobile Robots,” in *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, 2018, pp. 337–342.

[3] Y. Kobayashi *et al.*, “Assisted-care robot dealing with multiple requests in multi-party settings,” in *Proceedings of the 6th international conference on Human-robot interaction - HRI ’11*, 2011, p. 167.

[4] C. Park, S. Kang, J. Kim, and J. Oh, “A study on service robot system for elder care,” in *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2012, pp. 546–547.

[5] T. D. Larsen, K. L. Hansen, N. A. Andersen, and Ole Ravn, “Design of Kalman filters for mobile robots; evaluation of the kinematic and odometric approach,” in *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)*.

[6] Yong-Shik Kim *et al.*, “Localization of ubiquitous environment based mobile robot,” in *2007 International Conference on Control, Automation and Systems*, 2007, pp. 2355–2358.

[7] N. Ganganath and H. Leung, “Mobile robot localization using odometry and kinect sensor,” in *2012 IEEE International Conference on Emerging Signal Processing Applications*, 2012, pp. 91–94.

[8] Y.-C. Lee and W. Yu, “The design of practical mapping system for mobile robots using laser range sensor,” in *2009 IEEE Sensors*, 2009, pp. 1482–1486.