Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Diseño, programación e integración de un robot autónomo en un festival de danza y nuevos medios

Design, programming and integration of an autonomous robot on a dance and new technologies festival

Autora

Paula Abad Liso

Directores

Ana Cristina Murillo Arnal

Luis Riazuelo Latas

Ingeniería electrónica y automática

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Septiembre 2017

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. PAULA ABAD LISO,

con nº de DNI 25207054-C en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA, (Título del Trabajo) DISEÑO, PROGRAMACIÓN E INTEGRACIÓN DE UN ROBOT AUTÓNOMO EN UN FESTIVAL DE DANZA Y NUEVOS MEDIOS.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, SEPTIEMBRE DE 2017

Fdo: PAULA ABAD LISO

Agradecimientos

En primer lugar, agradezco el apoyo del proyecto CESAR (Centro de Supercomputación de Aragón) de la Universidad de Zaragoza, en especial a Fran y Fermín. También al equipo de Trayectos por ser tan cercanos.

Gracias a las bailarinas, Raquel y Laura, por venir siempre a los ensayos con una sonrisa y por sus esfuerzos por entender las partes más técnicas de este proyecto.

A Ana Cris, por estar siempre dispuesta a ayudar y sacar tiempo de donde no lo hay y guiarme tan bien a lo largo de este trabajo. Por contagiar tanta alegría y optimismo y por ser capaz de darle la vuelta a cualquier tortilla.

A Luis por su dedicación, esfuerzo y por no dudar en sacar tiempo para ayudar. Por resolver los problemas de un modo tan increíblemente rápido y eficaz. Gracias por hacernos reír hasta llorar, aunque al principio no te dieras cuenta.

A la gente del laboratorio, en especial a Íñigo y Ana, por ayudarme tanto y hacer aumentar mi interés por la informática. También por distraerme (que no molestarme) en algunas ocasiones.

A Rosa y a Miguel, por formar parte de este trabajo y por aguantarme.

A mi familia y amigos por haber confiando en mi y por poner tanto interés en lo que hago.

Y por último, quiero agradecer a Danilo por aceptar su derrota en el gran reto de la celebración final.

Resumen

Este trabajo presenta el resultado de la integración de un robot móvil autónomo en un espectáculo de danza contemporánea para un festival de danza y nuevos medios. El principal objetivo es integrar en la danza un robot móvil de bajo coste, con el fin de estudiar las posibilidades que este tipo de plataformas puede ofrecer a los artistas. Este proyecto se ha desarrollado en un entorno multidisciplinar donde han colaborado personas de las áreas de robótica, diseño y danza, en Etopia (centro de Arte y Tecnología en Zaragoza).

El trabajo tiene varias fases diferenciadas, todas ellas descritas en detalle en esta memoria. En primer lugar, se acordó junto con el resto de equipos participantes la temática de la actuación.

A continuación, se estudiaron los diferentes sensores y actuadores disponibles que se podían añadir al robot, además de los tipos de movimientos que este podría realizar. Todo esto teniendo en cuenta los limites de tiempo, recursos y preferencias y restricciones del equipo de baile. Se decidió colocar en el robot un láser que le permitiera localizarse en el espacio, además de unos LEDs programables que le dieran personalidad y sirvieran de apoyo para las bailarinas. El robot se programo para realizar dos tipos de movimientos: movimientos basados en odometría y movimientos basados en objetivos en el mapa (es decir, dirigirse a unas coordenadas concretas del mapa).

Posteriormente, los módulos acordados con el equipo de danza se han ido integrando en la coreografía. Para esta integración, diseño final y programación de los movimientos y acciones del robot, todas las semanas se han realizado reuniones o ensayos con las bailarinas o los diferentes grupos que componen el proyecto, con el fin de construir o perfeccionar la coreografía. La coreografía final consta de un conjunto de *movimientos* de los dos tipos mencionados, tiempos de espera relativos y tiempos de espera absolutos, además de los LEDs programados para los diferentes movimiento de la coreografía. La coreografía programada para el robot incluye movimientos muy precisos y que se ejecutan de manera muy fiable y repetitiva, ya que las bailarinas ejecutan sus movimientos muy cerca y muchas veces sin tener al robot en su campo de vista.

La programación del robot se ha realizado utilizando ROS (Robot Operating System), que se compone de un conjunto de librerías y herramientas que permiten construir aplicaciones robóticas de manera modular, y facilitan la integración sencilla de sensores y actuadores.

Los resultados obtenidos han cumplido perfectamente con los objetivos, siendo incluidos en las actuaciones en directo del festival de danza contemporánea de Zaragoza Trayectos. Esta memoria describe estos resultados finales en vivo, así como un análisis más detallado y técnico para validar más formalmente los datos obtenidos del sistema (errores en trayectorias, desplazamientos y velocidades).

Los principales problemas de este proyecto han sido la dificultad de coordinar y entenderse con equipos tan distintos y las restricciones impuestas a la programación y diseño del robot por las necesidades de los artistas.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Durante los últimos años hemos sido testigos de la introducción de la robótica en muchos y diversos campos. Existen numerosos proyectos que proponen como integrar nuevas tecnologías relacionadas con la inteligencia artificial y la robótica en nuevas áreas de aplicación, incluyendo áreas relacionadas con el arte. Por ejemplo, podemos encontrar proyectos que estudian cómo las artes podrían integrarse y beneficiarse de la robótica: desde aplicaciones más genéricas de las artes y la educación [1], hasta enfoques para crear pinturas hechas por robots [2], construir sistemas escultóricos interactivos [3] o experimentar con orquestas robóticas [4]. Nuestro trabajo se centra en explorar las posibilidades de integrar un robot en un equipo de danza contemporánea.

Este trabajo presenta los resultados de un proyecto experimental en el que un robot móvil autónomo se ha integrado en un festival de danza contemporánea. El festival de danza y nuevos medios, organizado por Trayectos[1], se celebra en el centro de arte y tecnología público *Etopia* (figura 1.1), de Zaragoza. La com-

[1] `http://www.danzatrayectos.com/en/laboratorio-de-danza-y-nuevos-medios/`

(a) (b)

Figura 1.1: (a) Etopia, centro de arte y tecnología. (b) Modelo original del robot Turtlebot 2 utilizado en este proyecto.

Figura 1.2: Robot y equipo de baile en la residencia de Etopia donde se ha desarrollado este proyecto.

pañía de danza involucrada en este proyecto es *Tarde o Temprano Danza*[2]. En la figura. 1.2 se muestran varias imágenes del robot y las bailarinas ensayando en la residencia de Etopia, lugar donde se llevaron a cabo los ensayos y el espectáculo final.

Uno de los desafíos que hace que este proyecto sea original es el objetivo de hacer del robot un componente más en el equipo de baile. La compañía de baile requiere que el robot tenga un comportamiento predefinido y preciso, ya que la actuación no es una improvisación por su parte. Esto significa que el robot también debe *aprender* su parte de la coreografía, para estar perfectamente sincronizado con el resto del equipo. Una de las lecciones más importantes aprendidas durante este proyecto es que los problemas teóricos y técnicos más comunes en la investigación robótica no son siempre los principales desafíos a los que nos enfrentamos en este tipo de aplicaciones. Los requerimientos y expectativas de los equipos no-robóticos de este proyecto multidisciplinario implicaron fuertes restricciones en aspectos como: muy alta precisión de ciertos movimientos, restricciones en el tipo de sensores que se pueden utilizar, comportamientos reactivos que no eran aceptables o limitaciones en las marcadores artificiales o elementos que se pueden añadir al escenario o en la ropa de los bailarines, lo que supuso un reto interesante.

1.2. Objetivos y Tareas

El objetivo general de este proyecto es estudiar las posibilidades que un robot móvil autónomo de bajo coste puede proporcionar a los artistas involucrados en un festival de danza contemporánea, así como diseñar e implementar los módulos necesarios y llevarlos a cabo a tiempo para la actuación.

[2] `http://tardeotempranodanza.wix.com/tardeotempranodanza`

Las tareas realizadas para conseguir la aplicación descrita, y el alcance de cada una de ellas se resumen a continuación:

- Estudio, instalación y familiarización del entorno ROS, *framework* utilizado para gestión de plataformas robóticas y sensores relacionados (previo estudio y familiarización del sistema operativo Ubuntu 14.04). Para ello se han realizado los tutoriales de la *wiki* de ROS[3], donde es posible estudiar los conceptos básicos de ROS, entender su arquitectura y sistema de ficheros y aplicar algunos ejemplos sencillos con el fin de aprender a utilizar sus herramientas. Más información sobre ROS en el Anexo A.

- Trabajo en grupo con el resto de participantes para acordar los módulos a implementar. Para ello, las personas responsables del diseño de la carcasa del robot convocaron una sesión creativa, donde decidimos la idea que se quería transmitir a través de la coreografía y el tipo de movimientos y aparatos interactivos con los que podía ser interesante trabajar.

- Estudio de librerías y módulos existentes compatibles con ROS para generación de trayectorias e interacción con el robot. Por ejemplo se hicieron pruebas con pulsadores, con un lector RFID y con LEDs. Los módulos se explican más adelante, en el Capítulo 2.

- Implementación o adaptación de los métodos estudiados para construir los módulos acordados. Se comenzaron a programar los movimientos del robot teniendo en cuenta las instrucciones iniciales de las bailarinas.

- Evaluación de los módulos con los usuarios finales e integración de los módulos seleccionados para la aplicación final. Se ensayó uno o dos días por semana con las bailarinas para construir la coreografía, poniendo en común ideas y comentando las posibilidades de movimiento de las bailarinas y del robot, y así estudiar cómo el otro podía adaptar sus movimientos.

- Pruebas y demostración de aplicación final en una plataforma robótica real. Demostración en el festival. Se realizaron muchas repeticiones de la coreografía final con el fin de identificar fallos o dificultades y modificarlos.

- Documentación de actividades, evaluaciones, *manual* de uso de las demostraciones preparadas, para permitir el uso de los resultados de este proyecto en futuros eventos. Este *manual* se encuentra en el Anexo D.

La organización temporal de las tareas que se han llevado a cabo en este trabajo se muestra en el *diagrama de Gantt* reflejado en la tabla 1.1.

1.3. Entorno de trabajo

Las herramientas básicas utilizadas son el entorno de ROS (*Robot Operating System*) sobre el sistema operativo Ubuntu 14.04 y librerías para el manejo de los sensores y actuadores utilizados. El lenguaje de programación empleado ha sido C++. En el Anexo A se resumen los principales tutoriales y documentación empleados para aprender a manejar ROS.

[3] `http://wiki.ros.org/ROS/Tutorials`

	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO
Estudio, instalación y familiarización del entorno ROS y del SO Ubuntu 14.04							
Trabajo en grupo con el resto de participantes para acordar los módulos a implementar.							
Estudio y pruebas de trabajos, librerías y módulos para generación de trayectorias e interacción con el robot.							
Implementación o adaptación de los métodos estudiados para construir los módulos acordados.							
Evaluación de los módulos con los usuarios finales e integración de los módulos seleccionados para la aplicación final.							
Pruebas y demostración de aplicación final en una plataforma robótica real. Demostración en el festival.							
Documentación de actividades, evaluaciones, "manual" de uso de las demostraciones preparadas.							

Tabla 1.1: Diagrama de Gantt.

Como plataforma móvil se ha usado un Turtlebot 2, que es un robot móvil de bajo coste, impulsado por los desarrolladores de ROS. La figura. 1.1(b) muestra la versión original de este robot sin ningún tipo de carcasa. En la Sección 2.1 se describe brevemente el proceso de diseño realizado por otros miembros del equipo de este proyecto.

Este proyecto se ha realizado dentro del laboratorio de Robótica, Percepción y Tiempo Real[4], del Instituto de Investigación en Ingeniería de Aragón de la Universidad de Zaragoza. Se ha realizado en colaboración con otros investigadores y artistas dentro del Laboratorio de Danza y nuevos Medios, en el marco del proyecto Europeo *Smart Places*. El festival de danza y nuevos medios, organizado por Trayectos[5], se celebra en un centro de arte y tecnología público (Etopia) de Zaragoza mostrado en la figura. 1.1, y la compañía de danza involucrada en este proyecto es *Tarde o Temprano Danza*[6].

Además, en este proyecto se ha utilizado el equipamiento proporcionado por el proyecto CESAR (Centro de Supercomputación de Aragón)[7], de la Universidad de Zaragoza.

1.4. Trabajo relacionado

Como se ha mencionado previamente, se ha intentado en muchas ocasiones integrar la robótica en el arte, con el fin de estudiar como el arte puede beneficiarse de la robótica y viceversa. Existen recientes publicaciones de in-

[4] `http://robots.unizar.es`
[5] `http://www.danzatrayectos.com/en/laboratorio-de-danza-y-nuevos-medios/`
[6] `http://tardeotempranodanza.wix.com/tardeotempranodanza`
[7] `http://cesar.unizar.es/`

vestigación que recopilan diferentes experimentos sobre robótica y arte [5] y sesiones específicas en los principales congresos de investigación sobre robótica[8]. Tambien podemos encontrar trabajo relacionado con la robótica y el arte en universidades de prestigio involucradas con centros artísticos profesionales[9], así como en otros eventos[10] y centros con fines educativos [6].

1.4.1. Los robots y la danza

Las obras relacionadas con las artes escénicas son especialmente importantes para nuestro proyecto, especialmente la danza y el teatro. Podemos encontrar estudios que se centraron en las posibilidades de expresar la emoción y la intención a través de los movimientos del robot [7]. Entre los trabajos más recientes, se encuentran investigadores que proponen sistemas autónomos capaces de reaccionar a diferentes sonidos y música, adaptándose por ejemplo a los tempos musicales o a las restricciones de movimiento [8][9]. También hay muchos estudios que consideran la danza robótica como una herramienta para actividades terapéuticas y aplicaciones médicas, como por ejemplo el trabajo dedicado a la terapia con niños [10]. Más detalles y ejemplos sobre robótica y danza pueden encontrarse en el estudio realizado en [11].

En cuanto a investigaciones con objetivos más comunes a los nuestros, hay investigadores que estudian cómo integrar robots en actividades de danza o espectáculo. Podemos encontrar estudios sobre como diseñar robots que bailen con personas, analizando la coordinación humano-robot [12][13]. Estos proponen un sistema que reacciona a las interacciones físicas gracias al uso de una base móvil omnidireccional equipada con sensores de fuerza. Hay poco trabajo previo experimental centrado en los requisitos del artista para diseñar la coreografía del robot como parte de la actuación de los artistas [14]. El enfoque de integrar robot como un miembro de un equipo híbrido humano-robot es un componente clave en nuestro trabajo. A diferencia de la mayoría de los trabajos previos sobre danza, hemos enfocado nuestro trabajo para evaluar el alcance de una plataforma de bajo costo, en colaboración con artistas locales, haciendo hincapié en que el robot sea un miembro más en el equipo de baile. Este objetivo ha puesto en relieve la necesidad de ciertas capacidades del robot como por ejemplo la precisión y la sincronización exacta con el tiempo, en lugar de los comportamientos reactivos y la improvisación de otras plataformas robóticas más autónomas.

1.4.2. Navegación autónoma de robots

Además, nuestro trabajo está fundamentalmente relacionado con las áreas de construcción de mapas y localización. Hemos utilizado enfoques bien establecidos para el mapeado 2D [15] y localización basada en filtro de partículas [16], utilizando un sensor de escaneo 2D montado en un robot móvil, como se detalla en las siguientes secciones. Estos algoritmos nos han permitido sacar adelante de manera robusta el espectáculo de danza, incluso con un fondo desordenado

[8] `http://www.roboticart.org/iros2017/`

[9] `http://artpower.ucsd.edu/dancing-robot-huang-yi-kuka/`

[10] `https://www.robofest.net/index.php/current-competitions/graf`

del espacio, ya que parte del contorno del escenario que ve el robot es el público, con frecuentes oclusiones en el campo de vista de este y con condiciones de iluminación y organización del espacio muy heterogéneas de unas ejecuciones a otras.

1.5. Contenido de la memoria

El contenido de esta memoria se distribuye en 4 capítulos, siendo el primero de ellos el presente capítulo introductorio. En el Capítulo 2 se describen los principales módulos del sistema. En primer lugar se explican las limitaciones físicas del entorno y las limitaciones de movimientos y se describen las soluciones empleadas. Posteriormente se describe la arquitectura del sistema, siendo sus dos módulos principales el módulo de movimientos y el módulo de LEDs. En el Capítulo 3 se describe el proceso para construir la coreografía, así como los tipos de movimientos involucrados en ella. En el Capítulo 4 se analizan tanto los datos numéricos como los resultados finales de la actuación. Por último, en el Capítulo 5 se habla de las conclusiones extraídas de estos resultados y se describen las líneas de trabajo futuro.

Capítulo 2

Diseño y Arquitectura del robot

Esta sección describe los principales módulos del sistema: 1) el diseño y los componentes principales de la plataforma robótica; 2) los módulos utilizados para construir el mapa del entorno donde el robot va a realizar sus tareas; 3) la arquitectura del sistema construida para programar todas las tareas del robot.

2.1. Plataforma robótica

Esta sección describe el diseño y los componentes principales de la plataforma robótica utilizada en este proyecto. Para ayudar a transmitir el mensaje que los artistas quieren hacer llegar al público, sobre la influencia mutua entre robot y humano, el equipo de diseño del proyecto desarrolló una carcasa para el modelo de robot utilizado.

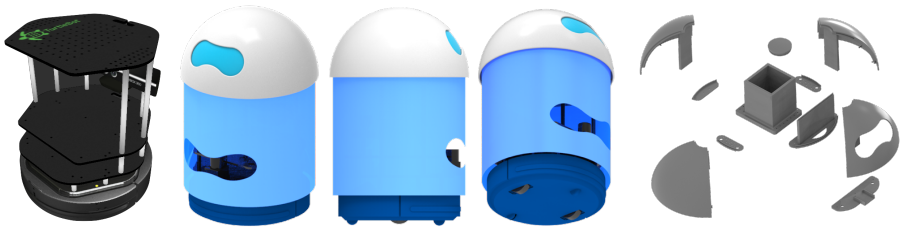
La figura 2.1(a) muestra una imagen renderizada del modelo original del robot utilizado, TurtleBot 2[1]. En la figura 2.1(b) vemos tres puntos de vista diferentes del robot con la carcasa diseñada.

Además de la carcasa del robot, se desarrollaron varias piezas adicionales para servir como elementos de soporte, tanto para la propia carcasa como para otros elementos internos y sensores. La gran mayoría de estas piezas se fabricaron con una impresora 3D y, por lo tanto, debían dividirse en piezas que se ajustasen a las medidas del área de impresión de la impresora 3D disponible. La figura 2.1(c) muestra el modelo 3D de todas estas piezas de soporte, incluyendo las piezas de la carcasa de la cabeza, soportes internos para sensores y otros componentes pequeños. La figura 2.2 muestra diferentes etapas del robot a medida que se construían y unían los diferentes componentes.

2.1.1. Sensores y actuadores

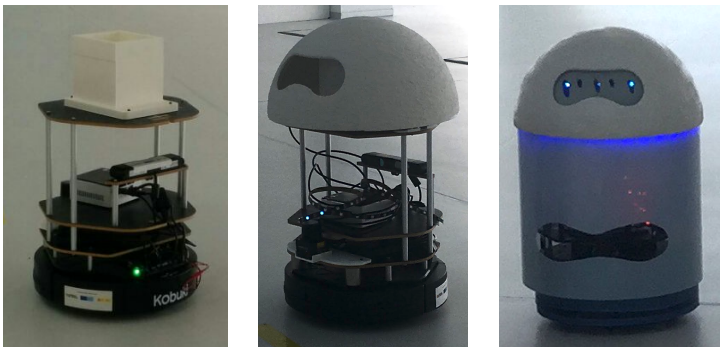
Durante las etapas iniciales de este proyecto distintos sensores fueron considerados, pero muchos fueron descartados principalmente por el tiempo de desarrollo y restricciones de peso o por requisitos de robustez y precisión. Además

[1] `http://www.turtlebot.com/turtlebot2/`



(a) (b) (c)

Figura 2.1: Modelo 3D del Turtlebot 2 original (a), carcasa diseñada para este proyecto, desde diferentes puntos de vista (b) y piezas impresas en 3D, incluyendo partes de la carcasa y piezas de soporte para sensores y LEDs (c).



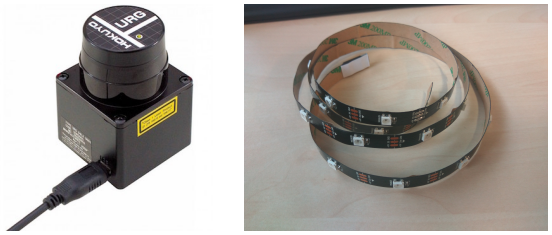
(a) (b) (c)

Figura 2.2: Diferentes estados de la apariencia del robot al ir añadiendo elementos de la carcasa.

de los sensores básicos del TurtleBot 2, como se muestra en la figura 2.3, hemos añadido un *láser Hokuyo* (ya que era la opción más robusta para mapear el entorno dados los requisitos y restricciones del escenario), que hemos colocado en la parte frontal de la plataforma más baja, y las tiras LED programables (ya que eran el único componente adicional disponible para darle capacidades expresivas al sistema que se ajusta al escenario y requisitos y restricciones de los artistas) que están colocadas en diferentes partes del robot. Una tira de LEDs rodea la parte superior de la plataforma robótica y de la otra está colocado un extremo en la parte donde van los ojos, en una pieza impresa en 3D y la otra parte enrollada en un soporte que se encuentra centrado en la parte delantera, en la zona donde iría el corazón. Esto se puede ver en la figura 2.11.

Las principales decisiones adoptadas a este respecto fueron las siguientes:

En primer lugar, para construir el mapa y localizar el robot en él (como se detalla en la Sección 2.2), nuestro sistema utiliza un escaneo láser en 2D. Este es el sensor más robusto dado el entorno y las restricciones existentes. Aunque sus mediciones 2D proporcionan modelos ambientales más simples que otros sensores, por ejemplo una cámara, su comportamiento es también más robusto y fiable. Esto se debe al hecho de que el escenario tiene elementos geométricos



(a) (b)

Figura 2.3: Dos de los componentes principales del robot: escáner láser 2D (a) y tiras de LED (b).

estáticos que permiten una cartografía y localización muy robustas utilizando este sensor.

Otros sensores estudiados descartados para el diseño final. En segundo lugar, para dar capacidades adicionales de expresión al robot, se evaluaron sensores interactivos (tales como sensores de proximidad, sensores de contacto y cámaras). Como sensor de proximidad, se quiso utilizar un lector *RFID* (*Radio Frequency Identification*) llamado *Touchatag* (figura 2.4). Este lector envía señales de radio de corto alcance que son recogidas por una etiqueta/pegatina *RFID*, y posteriormente la etiqueta devuelve una cadena corta de datos [17]. Las etiquetas hubiesen ido pegadas a las manos de las bailarinas y cuando estas las acercasen al lector el robot haría un movimiento prefijado. La distancia máxima a la que se puede colocar la pegatina del lector para que este lea el código son 4 cm. Sin embargo, el mayor problema de esto era que el lector no siempre identificaba la proximidad de la pegatina por lo que tuvimos que descartar esta opción de cara al espectáculo.

En cuanto a sensores de contacto propusimos utilizar un pulsador (pudiendo utilizarse un simple ratón USB para dicho fin, incorporado en la carcasa del robot). Usar un pulsador era más eficiente que utilizar el anteriormente mencionado *RFID*, ya que la pulsación siempre es detectada. Estas dos opciones fueron incorporadas al programa como posibles opciones. Sin embargo, ya que las bailarinas querían estar muy cerca del robot y querían saber en todo momento los movimientos exactos y el plan del robot, aunque estuviera fuera de su vista, se optó por no incluir comportamientos reactivos. Por lo tanto, el único elemento adicional acordado que encajaba con el escenario y con los requisitos y restricciones de los artistas eran los LEDs programables, que fueron programados para tener comportamientos diferentes dependiendo del tipo de movimiento y de la música que sonaba en cada momento. También se consideró la opción de

Figura 2.4: Lector RFID

usar un altavoz colocado en el propio robot que reprodujese sonidos robóticos. Finalmente se decidió que era mejor reproducir estos sonidos desde los mismos altavoces por los que se reproducía la música, ya que el sonido iba a ser más claro.

Por último, se consideró la opción de añadir partes móviles al robot pero esta idea tuvo que ser descartada debido a los límites de peso que admitía la base.

2.2. Modelado y localización del robot en el entorno

2.2.1. Requisitos

El área donde la actuación ha tenido lugar es un espacio abierto en lugar de un escenario convencional (más detalles en la subsección 2.2.2). Esto quiere decir que no existe separación física entre el público y el espacio de la actuación. Para asegurar ciertos componentes geométricos estáticos que el robot puede incorporar fácilmente al modelo (mapa) del entorno de trabajo, se necesitan algunos elementos de separación. Al construir estos elementos, los requisitos principales fueron:

1. Seguir las exigencias estéticas de los artistas.

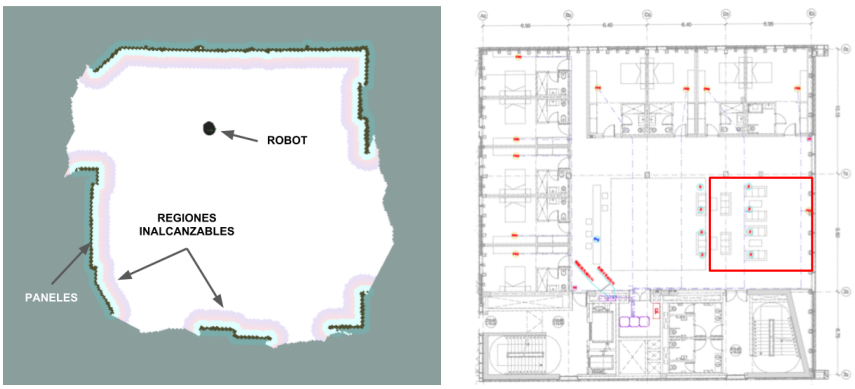
2. Evitar cualquier oclusión innecesaria de la actuación al público.

Como una solución de bajo coste, se hicieron unos pocos bloques de poliestireno expandido y se dispusieron alrededor del espacio de la actuación, con huecos entre unos y otros. En la figura 2.5 podemos ver algunos de estos bloques (pequeños paneles blancos) durante las sesiones de entrenamiento de la coreografía. Estos bloques se pintaron de blanco para darles más apariencia de robustez y se adhirieron adecuadamente al suelo. Tenían una altura de 40 centímetros, lo que fue suficiente para que el *láser Hokuyo* lo viera, ya que este fue colocado en la parte baja del robot.

2.2.2. Modelo o "mapa"del entorno

La odometría no funciona lo suficientemente bien para localizar el robot con buena precisión. Por lo tanto, es necesario la utilización de navegación, que hace uso de mapas precargados con el fin de reubicar al robot con precisión. La

Figura 2.5: Imágenes de sesiones de entrenamiento donde podemos ver los paneles utilizados para limitar el mapa del espacio de la actuación.



(a) (b)

Figura 2.6: Mapa del escenario y sus límites (a) y planta del recinto completo, donde el recuadro rojo indica la zona del escenario (b).

obtención de mapas se llevó a cabo a través de un nodo estándar disponible en ROS para dicha tarea[2] (instrucciones en el Anexo D). Este nodo de mapeo implementa el conocido algoritmo de *SLAM* de *gmapping* [15] y nos permite construir un mapa a partir de los datos obtenidos con un escaneo en 2D, por ejemplo con un láser.

Este mapa permite al robot localizarse en cualquier momento, siempre y cuando la distribución del espacio mapeado no cambie. La figura 2.6 (a) muestra un mapa obtenido del escenario de la actuación. Este mapa se ha obtenido usando una aplicación para generar mapas con la técnica *SLAM* [16]. Obsérvese que la vista del *mapa de obstáculos* mostrada en esa figura ensancha las *regiones inalcanzables* alrededor no sólo de los obstáculos sino también de los límites/paredes del escenario. Esto significa que el robot no puede acercarse demasiado a los elementos físicos reales para evitar colisiones. Esta es una característica de navegación común que debe tenerse en cuenta cuidadosamente al diseñar la coreografía, ya que al utilizar las funciones de navegación del mapa,el robot no puede alcanzar ninguna posición muy cerca de ningún obstáculo (incluidos los bailarines). Necesitamos inhabilitar esta funcionalidad para obstáculos dinámicos si queremos que el robot se mueva muy cerca de los otros bailarines, como se puede ver en muchos de los ejemplos en este proyecto, por ejemplo en la figura 2.5. En la figura 2.6 (a) se señala el robot y los paneles que establecían los límites del escenario. La figura 2.6 (b) muestra la planta del recinto completo, es decir, la sala común de la residencia de Etopia y algunas de las habitaciones. El recuadro rojo indica la zona donde se encontraba el escenario y, por lo tanto, donde se realizaron todos los ensayos. El público se colocó en los espacios libres.

2.2.3. Localización del robot

El robot se localiza en el mapa utilizando los datos de escaneo del láser 2D. Este método mejora el primer intento basado solamente en la odometría, que

[2] `http://wiki.ros.org/gmapping`

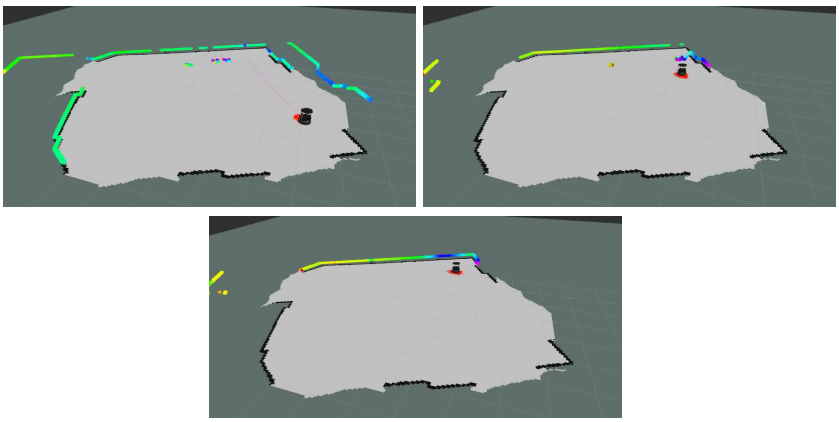


Figura 2.7: Robot planeando y ejecutando una trayectoria para alcanzar un objetivo dado en el mapa (visualizado en *RosViz*). Las líneas azules y verdes que coinciden con el contorno del escenario son puntos escaneados con el láser y alineados con el mapa. La linea delgada rosa que sale del robot representa la trayectoria planeada. Las pequeñas líneas rojas alrededor del robot representan la incertidumbre en la localización del robot.

sólo es útil para cortos periodos de tiempo. El sistema utiliza el nodo *AMCL*[3] disponible de ROS. El *AMCL* es un conocido método probabilístico de localización 2D que implementa un enfoque adaptativo de localización *Monte Carlo* (Adaptive Monte Carlo Localization, según las siglas AMCL) utilizando un filtro de partículas para rastrear la posición del robot en un mapa conocido. La figura 2.7 muestra algunos de los principales ingredientes de este enfoque. La incertidumbre del robot acerca de su ubicación depende de como de bueno sea el algoritmo alineando los datos de exploración 2D en todo momento con el mapa conocido del entorno.

2.3. Arquitectura del sistema

El sistema utilizado para controlar el robot en este proyecto tiene dos componentes principales, el control de los movimientos del robot y el control de los LEDs. En la figura 2.8 se muestra el diagrama de los principales bloques del programa.

2.3.1. Módulo de movimientos

El sistema para controlar el robot ha sido creado usando ROS (Robot Operating System) y sus capacidades disponibles para la plataforma TurtleBot 2[4]. El módulo de movimientos tiene dos elementos fundamentales: la estimación de la odometría y la localización en el mapa del escenario.

El nodo *figuras_node* es el programa principal, en él se encuentran los diferentes movimientos del baile y es el que controla toda la coreografía y el

[3] `http://wiki.ros.org/amcl`
[4] `http://wiki.ros.org/indigo`, `http://wiki.ros.org/turtlebot_navigation/`

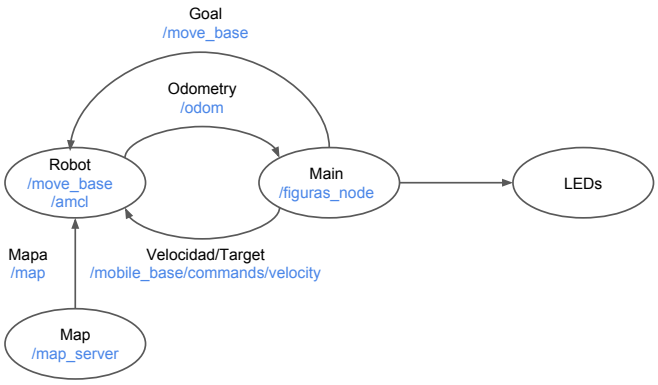


Figura 2.8: Diagrama de los bloques principales.

tiempo en el que esta transcurre. Este nodo publica en un *topic* llamado */mobile_base/commands/velocity* que envía comandos de velocidad al controlador de bajo nivel del robot indicándole la velocidad a la que tiene que ir el robot en cada momento. Además, está suscrito al nodo */odom*, que proporciona la odometría que va calculando el robot. La odometría se calcula a partir de sensores, que leen el giro del propio robot y el giro de las ruedas de este para calcular el desplazamiento. Por lo tanto, con odometría es posible realizar los movimientos con más precisión que si nos basásemos en el tiempo que calculamos que tardaría el robot en recorrer una distancia a una velocidad determinada, ya que el rozamiento y el peso del propio robot y de los componentes que lleva montados en él, hacen que este se frene y realizaría movimientos más cortos. Por último, el nodo principal es *cliente* de otro nodo llamado */move_base*. De esta forma, a través del programa principal solicitamos que el robot se mueva a una posición determinada del mapa y este nodo nos devuelve el movimiento a esa posición (gracias al nodo */amcl_pose*), con muy buena precisión ya que se basa en el mapa y no en la odometría. Este termina devolviendo un mensaje indicando que el robot ha llegado a su destino. Por último, en el nodo *figuras_node* es también donde se decide que hacen los LEDs en cada momento. Esto se explica con más detalle en la Sección 2.3.2.

Como ya se mencionado anteriormente en la Sección 2.2.3, al disponer de un mapa estático y de un láser 2D para ubicar al robot gracias al método *AMCL*, sabremos con exactitud donde se encuentra el robot en cada momento. La figura 2.7 muestra varias imágenes del robot ejecutando una trayectoria siguiendo un planificador local para alcanzar una meta. Sin embargo, en la mayoría de las situaciones es imposible realizar este movimiento planificado hacia un objetivo. Esto se debe a que a menudo los movimientos que tiene que realizar el robot tienen que ser muy precisos en cuanto a posición y tiempo de ejecución, ya que este tiene que estar sincronizado con la música. Puesto que, al hacer uso de este servicio, el robot planifica una trayectoria al destino desde dónde se encuentre, no siempre tarda lo mismo en llegar a la meta ni la trayectoria es siempre la misma, así que es conveniente que al realizar este tipo de movimientos exista un tiempo de espera hasta un momento concreto de la

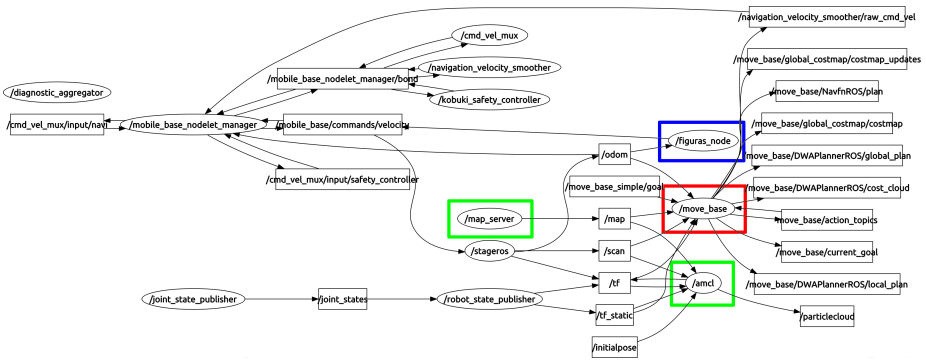


Figura 2.9: Diagrama de los nodos de ROS utilizados en nuestro sistema.

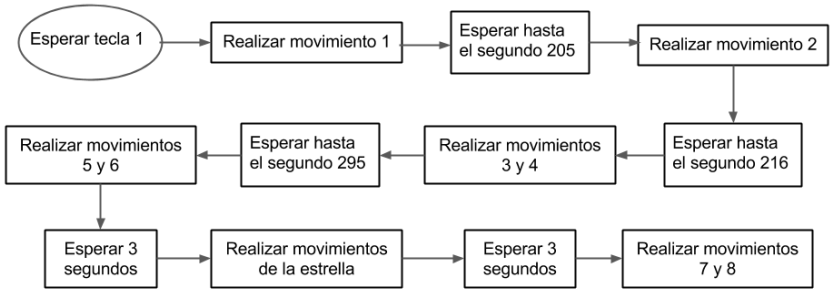


Figura 2.10: Esquema de la ejecución del programa principal.

música para que las bailarinas y el robot puedan sincronizarse de nuevo. Por lo tanto, en la mayoría de los casos, el robot tiene que ejecutar movimientos basados únicamente en su odometría.

En la figura 2.9 se muestra un diagrama, generado con *rqt_graph*[5], de los principales nodos y topics de ROS usados en nuestro sistema. Está compuesto por varios módulos de ROS que proporcionan al robot la capacidad de navegación autónoma (en rojo *move_base*) y localización (en verde *map_server* y *amcl*). La navegación se basa en un método de evitación reactiva de obstáculos y además, como ya hemos comentado, incorpora un planificador para calcular el camino a la meta asignada. Los módulos de localización proporcionan la ubicación del robot durante la actuación. Están basados en un mapa pre-construido y un filtro de partículas que proporciona la localización. El módulo enmarcado en azul *figuras_node* está sincronizado con la música, y está a cargo del control de toda la coreografía del espectáculo (los detalles sobre las trayectorias específicas generadas se dan en la siguiente sección).

En la figura 2.10 se muestra un esquema donde aparecen los movimientos, tiempos de espera y retrasos del programa principal y el orden en el que estos se ejecutan. La coreografía empieza a los ocho tiempos de empezar la música, que es cuando habrá que presionar la tecla 1 del teclado para que el robot comience la coreografía. El baile está dividido en 9 *movimientos*, cada uno en una función diferente. Cada *movimiento* se compone de un conjunto de instrucciones que se

[5] `http://wiki.ros.org/rqt_graph`

le dan al robot de lo que debe hacer y se compone tanto por giros, desplazamientos (basados tanto en mapa como en odometría) y tiempos de espera, como por los hilos de ejecución de los LEDs. Lo que tienen en común los pasos de baile de cada *movimiento*, tanto de las bailarinas como del robot, es que estos deben ejecutarse de manera seguida y tienen un estilo, ritmo y mensaje común. En cada *movimiento* alguna de estas características es diferente a los otros *movimientos*. Además era conveniente separarlos de esta manera de cara a los ensayos, ya que las bailarinas ensayaban cada una de estas partes por separado. Por esto es por lo que el programa comienza a ejecutarse al pulsar la tecla 1, porque para los ensayos cada movimiento tenía asignada una tecla diferente según la parte que quisieran ensayar.

Uno de los movimientos tiene otro nombre: *estrella*. Esto simplemente es por diferenciarlo de los otros movimientos y lo único que tiene de especial es que cuando se ejecuta, la música se apaga y solamente se oyen unos sonidos robóticos. Además, el único que tiene recorridos amplios aquí es el robot, mientras que las bailarinas se encuentran *dormidas* y este intenta despertarlas acercándose a ellas y haciendo que se muevan levemente. El nombre de *estrella* se debe a que inicialmente la trayectoria que recorría el robot tenía forma de estrella, pero tuvo muchas modificaciones y finalmente la forma que adoptó es la que aparece en la figura 3.3 (g).

Además, cada uno o dos movimientos hay un tiempo de espera absoluto (en segundos) referenciado al inicio de la coreografía donde se coge la estampa del tiempo global. Aquí el robot espera quieto hasta que se ha alcanzado ese tiempo, entonces comienza a ejecutar el siguiente movimiento. Estas esperas sirven para sincronizar de nuevo al robot con la música, con el fin de eliminar errores temporales que se van acumulando cada vez más a lo largo del tiempo. Por último, hay dos tiempos de espera relativos de 3 segundos antes y después del movimiento de la estrella, esto es solamente para dar tiempo a las bailarinas de colocarse tumbadas y levantarse para seguir al finalizar este movimiento, además de permitir un corto periodo de tiempo para la incertidumbre del público.

2.3.2. Módulo de LEDs

Hemos usado 2 tiras de LEDs, cada una con 32 componentes LED *RGB* programabes. Utilizamos la API de alto nivel porporcionada por el fabricante[6] que se centra en la programación del color, intensidad y tiempo de cada LED. Los LEDs tienen el principal objetivo de humanizar al robot. Una tira de LEDs se utiliza para emular los ojos y el corazón del robot, mientras que la segunda tira recorre el perímetro del robot con fines estéticos o para simular la boca. Esto se muestra en la figura 2.11 donde se pueden ver las tiras de LEDs en sus respectivos soportes además de adheridos con cinta americana para mejorar la sujeción. Vemos que la tira que sirve de ojos y corazón tiene la parte central de los LEDs sin utilizar (el segmento que cruza de arriba a abajo) y que en la parte de abajo (el corazón) está enrollada a un soporte. Además de los fines estéticos, estos LEDs ayudan a las bailarinas a ver en que módulos el robot está funcionando en cada momento. Esto les permite sincronizarse con el robot y

[6] `https://github.com/arvydas/blinkstick-python`

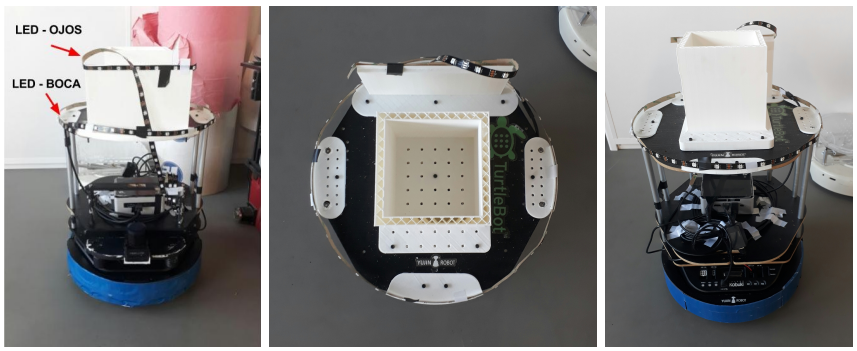


Figura 2.11: Diferentes vistas del robot sin la carcasa exterior. (a) Vista frontal, donde se pueden ven las dos tiras de LEDs. (b) Vista superior. (c) Vista trasera.

entre ellas utilizando los LEDs como referencia.

El módulo de LEDs se ejecuta en todo momento en hilos de ejecución (threads) paralelos al proceso principal, seleccionando diferentes comportamientos dependiendo de la parte de la coreografía, como avanzar, retroceder o acelerar, parado mirando al público, parado esperando para sincronizarse con los tiempos o girando para alinearse con el público. La tira de LED permite la programación separada del color RGB de cada LED, por lo tanto, una vez que las tiras estaban unidas al robot, pudimos configurar fácilmente qué LEDs correspondían a los ojos, el corazón o la cabeza del robot.

La tabla 2.1 describe los distintos comportamientos programados para los LEDs, según el tipo de movimiento a representar. Los nombres con los que se ha nombrado a las dos tiras de LEDs son LED-boca y LED-ojos (que esta además de los ojos enciende el corazón). Se muestra además en la tabla cual es la tira de LEDs para la que está programada cada función y cuales de los LEDs se encienden en cada una de ellas.

Tabla 2.1: Comportamientos programados para los LEDs.

Nombre	LEDs	**Descripción**
Avanzar	Boca - todos	Cuando el robot avanza y en momentos donde no hay otra función establecida. LEDs de color azul permanente.
Saludo	Boca - todos	Cuando el robot mira parado hacia público. El tono de los LEDs cambia progresivamente, recorriendo el círculo cromático de colores.
Parpadeo	Boca - todos	Al girar para *saludar* al público en amarillo y al retroceder en verde, siendo la velocidad de parpadeo proporcional a la velocidad del robot en este caso.
Acelerar	Boca - 8 leds no consecutivos	Cuando la velocidad de avance es elevada. Color aleatorio. Barrido de los 8 LEDs.
Aleatorio	Boca - todos	En algunos momentos con el robot parado. Parpadeo con colores aleatorios en cada LED.
Latido	Ojos - 0-15	Al final de la coreografía. El corazón se enciende y se apaga progresivamente.
Progresivo	Boca - todos	Al *mirar* a las bailarinas al inicio de la coreografía, en giros de 90^o y en algún momento esperando para sincronizarse. Se encienden y apagan progresivamente en blanco.
Ojos	Ojos - 25-31	Encendidos siempre. Los dos ojos del robot en azul claro.
Guiño	Ojos - 29-31	Se enciende sólo un ojo del robot en azul claro. El otro se apaga. Cuando el robot se *enchufa* y *desenchufa* lo hace por partes: primero un ojo, luego el otro y finalmente la boca.

Capítulo 3

Diseño de la coreografía

3.1. Coreografía

La coreografía consta de varias figuras cuya ejecución dura casi 13 minutos, y está sincronizada con tres tramos de sonidos distinto: dos temas musicales y una parte sin música con sonidos de robot en medio del segundo tema musical.

La coreografía final diseñada para el robot esta representada en los diagramas de la figura 3.3. El proceso de diseño, así como los dos tipos de movimientos que se ejecutan en las mismas (basados en odometría u objetivos en el mapa), son detallados a continuación.

3.2. Diseño de la coreografía

El proceso para diseñar y acordar los movimientos a realizar por el robot ha sido un proceso largo con distintas fases.

Reuniones previas multidisciplinares. En primer lugar, hubo una reunión con todos los participantes del proyecto para conocer que podía aportar cada uno de los equipos. En la reunión se explicó cual era el robot y cuales eran sus posibilidades y por parte de las bailarinas, se comentó cual era el género de danza con en que trabajaban (danza contemporánea) y en que consistía.

Más adelante, los responsables del diseño de la carcasa del robot convocaron una sesión creativa (figura 3.1), dónde se decidió entre todos cual era el significado que se quería dar a la actuación. Como resultado de esta sesión se acordó que queríamos transmitir la idea de la influencia robot-humano y humano-robot, mezclando movimientos fluidos y artísticos con otros más geométricos y exactos.

Reuniones para diseñar la coreografía. Las reuniones para ensayar con las bailarinas eran uno o dos días por semana (los primeros días las reuniones se convocaban con menos frecuencia y los días anteriores a la actuación estas reuniones y ensayos eran diarias), de aproximadamente 4 horas cada sesión.

(a) (b)

Figura 3.1: Sesión creativa con los participantes del proyecto. Presentación de la contribución de cada uno de los equipos del proyecto: robótica, diseño y danza (a). Puesta en común de las conclusiones de los participantes después de los ejercicios de creatividad (b).

Para diseñar las distintas *figuras*, se programó un conjunto de elementos básicos de movimiento: avances, giros y arcos. Dónde en cada uno de ellos se podía modificar fácilmente la velocidad lineal o angular, la distancia o arco recorrido, el sentido de movimiento y, en el caso de los arcos, el radio. Así que se hicieron pruebas combinando alguno de estos movimientos mientras las bailarinas trabajaban con la improvisación. También se hizo alguna prueba de improvisación conjunta, con el robot teledirigido y las bailarinas realizando movimientos de baile. Esto consigue inspiración de las bailarinas en los movimientos del robot y viceversa.

Al inicio las reuniones consistían en compartir qué podían aportar el robot o las bailarinas, que tipo de movimientos podía hacer el robot y las ideas que ellas tenían sobre que les gustaría que hiciera. Por ejemplo, querían que hiciera cosas imposibles para el robot, como saltar o girar sobre una rueda con la otra sin apoyar en el suelo. Así que hubo que hacer un trabajo de abstracción para poder explicar algunos aspectos técnicos del robot para que pudieran entender con facilidad su funcionamiento.

Las reuniones posteriores fueron más homogéneas. Cada día se comenzaba hablando sobre las nuevas *figuras* programadas que realizaba el robot, los problemas que habían surgido o nuevas ideas de movimientos o interacciones que se podían incluir en el baile. Posteriormente se probaban, con o sin música, las nuevas *figuras* y se corregían detalles como por ejemplo algún radio, velocidades, finalizar una figura en otro punto del escenario, etc., hasta que la coreografía cuadrase en tiempo, espacio y coordinación y quedara visualmente atractiva. Al finalizar cada reunión se comentaban las impresiones de cada uno y se concretaba que *deberes* había que realizar de cara a la siguiente sesión. La figura 3.2 muestra algún ejemplo de como fueron los ensayos.

Decisiones técnicas consensuadas en las reuniones multidisciplinares. Enseguida se vio que los movimientos basados únicamente en odometría acu-

(a) (b)

Figura 3.2: Ensayos con la compañía de danza taiwanesa *B. Dance*[2] consultada durante el proyecto (a) y ensayo un día normal (b).

mulaban mucho error, así que decidimos incorporar un mapa y funcionalidad para localizarse en el, como se ha explicado en la Sección 2.2. De este modo, la coreografía final se compone por dos tipos de movimiento que se detallan a continuación.

3.3. Tipos de movimientos

Como se ha mencionado anteriormente, los movimientos del robot durante la coreografía (descritos en la figura 3.3) pueden dividirse en dos tipos.

Movimientos basados en odometría. La mayoría de las trayectorias específicas tenían que seguir ciertas formas geométricas y recorrer distancias exactas en intervalos de tiempo específicos. Con el fin de tener un control estricto sobre esto, muchas trayectorias fueron diseñadas para ser ejecutadas utilizando la generación de trayectorias basada puramente en la estimación de odometría y verificación (trayectorias azules en la figura 3.3).

Movimientos basados en objetivos en un mapa. En varios momentos clave a lo largo de la coreografía, el robot ejecutó trayectorias utilizando el módulo de navegación autónoma (trayectorias verdes en la figura 3.3).Estas trayectorias se ejecutan de acuerdo con el algoritmo de localización (nodo *AMCL* de ROS) que localiza el robot dentro del mapa existente del entorno. Esto requiere variar los tiempos de espera al final de estas figuras/movimientos, que no son ideales para los artistas, pero fueron necesarios para evitar colisiones debido a los errores de odometría inevitablemente acumulados con el tiempo.

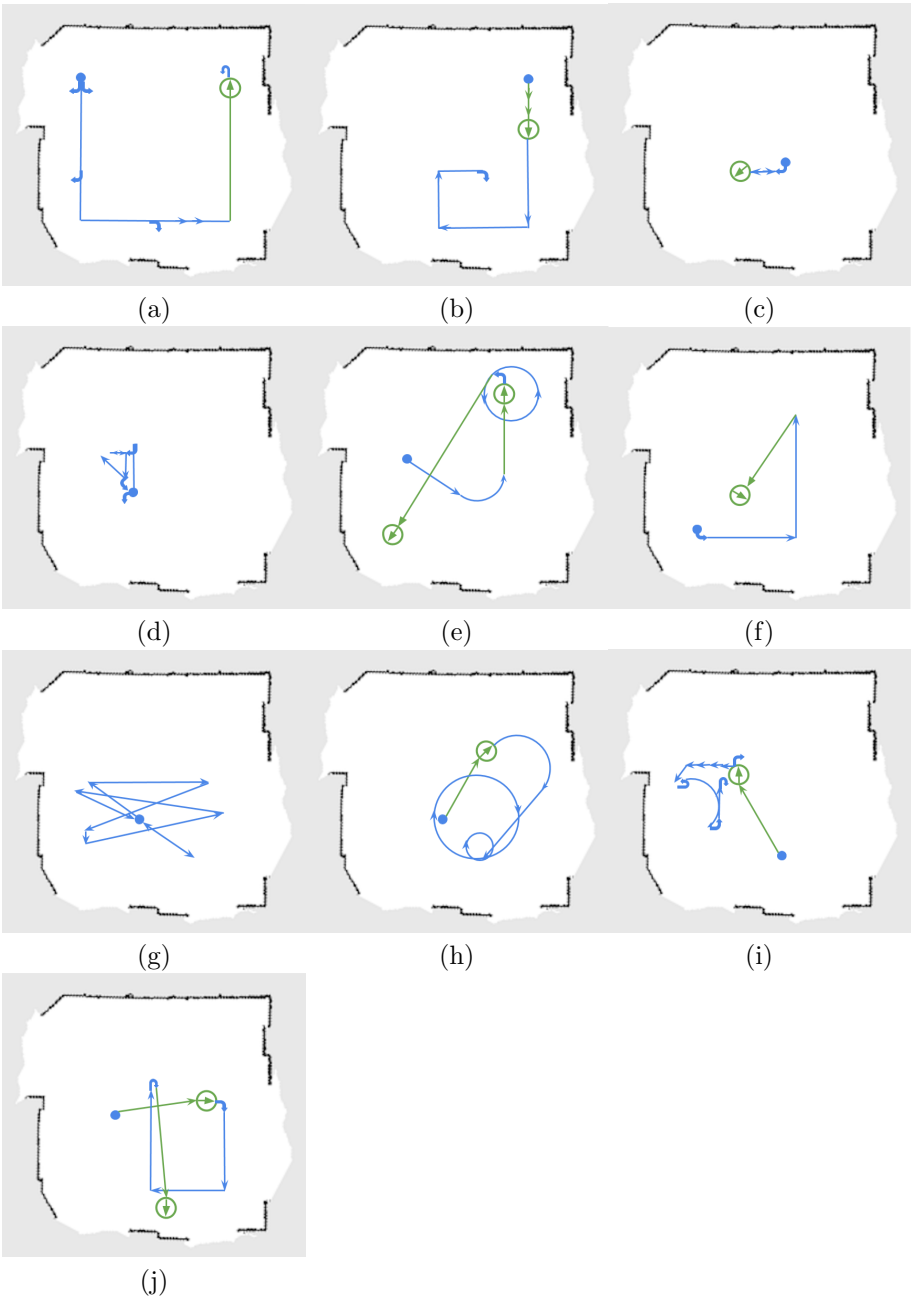


Figura 3.3: Coreografía diseñada para el robot. Los diagramas (a)-(j) representan los movimientos ejecutados por el robot. En azul: movimientos *basados en odometría*. En verde: movimientos *basados en objetivos en el mapa*. El punto azul de cada diagrama representa el punto de inicio de cada parte. Las flechas azules indican rotaciones y direcciones de los movimiento.

Capítulo 4

Evaluación y experimentos

Este capítulo detalla los distintos experimentos realizados para evaluar el sistema desarrollado y los resultados de la actuación final.

4.1. Entorno de experimentación

La configuración final y los sensores de la plataforma robótica utilizados en todos los experimentos han sido detallados en la Sección 2.1.

La **plataforma** TurtleBot 2 tiene un ordenador incorporado para controlar los módulos de movimiento de bajo nivel del robot y un ordenador adicional montado en las placas del robot donde están instalados los principales componentes de software, incluyendo ROS. Nuestra plataforma está equipada con un Intel NUC 6i5SYH (Intel Core i5-6260U, 8GB DDR4), el ordenador principal que lleva encima el robot, donde se ha instalado la distribución Indigo de ROS[1] sobre Ubuntu 14.04.

El **escenario** para la actuación de este proyecto es un gran espacio abierto que pertenece a la residencia del centro de Arte y Tecnología Etopia[2], mostrada en la figura 4.1. No todo el espacio fue mapeado por el robot, ya que una gran parte había que mantenerla libre para el público.

[1] `http://wiki.ros.org/indigo`
[2] `https://www.zaragoza.es/ciudad/etopia/`

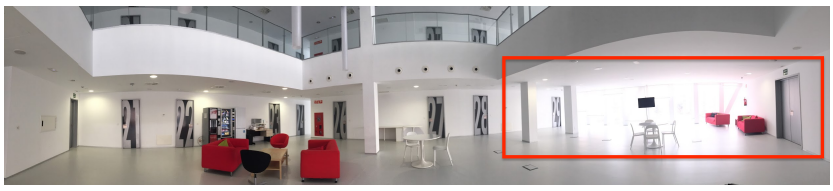


Figura 4.1: Vista panorámica del lugar donde se encuentra el escenario. El escenario es la zona que se encuentra dentro del rectángulo rojo. El público ocupaba el resto del espacio.

En cuanto a los **datos** a utilizar para el análisis detallado de la calidad y errores de los movimientos, en primer lugar, probamos a ejecutar todas las trayectorias de la coreografía en el entorno de simulación disponible para TurtleBot 2 de ROS, *Stage*. Sin embargo, observamos que los tiempos de ejecución y los errores de la odometría no eran equivalentes a las reales, probablemente debido a la variación de peso, equilibrio y configuración de nuestra plataforma en comparación con el modelo TurtleBot 2 original incorporado en el simulador. Por lo tanto, para realizar el análisis de los desplazamientos y las velocidades, hemos ejecutado la coreografía y almacenado todos los datos posibles del sistema en un *Rosbag* [3], el sistema estándar para almacenar datos y *logs* en ROS. Esto nos permite disponer del valor de todos los mensajes de cada *topic* en cada momento y poder analizar así los errores de los desplazamientos y velocidades, y hacer un análisis cuantitativo de estos.

El programa implementado para la **ejecución** de la coreografía, se lanza en el ordenador incorporado en el robot. El programa se puede lanzar conectándose directamente al ordenador del robot (con un teclado y pantalla) o conectándose al mismo de manera remota, vía conexión ssh. Más detalles de como comunicar un portátil con el ordenador incorporado en el robot en el Anexo D.

4.2. Análisis de las trayectorias realizadas.

Como parte de la evaluación del sistema implementado, queremos analizar los errores y robustez de los distintos componentes. En primer lugar se explica de que dependen los errores de cada uno de los tipos de movimientos utilizados (basados en odometría o mapa), y se analizan las trayectorias obtenidas en cada uno de ellos. Por último se analizan los errores en las velocidades y desplazamientos en ambos casos.

Movimientos *basados en objetivos en el mapa.* El error aceptable en estos casos se configura directamente en el nodo de los parámetros de configuración de ROS, porque nos permiten establecer el umbral de aceptación para considerar que el robot ha alcanzado una posición.

Movimientos *basados en odometría.* En estos casos hay una mayor variabilidad en los valores de error. El error acumulado de la odometría no es significativo cuando la exactitud de la forma de la trayectoria y el momento son más importantes para los artistas que la ubicación particular en el mapa donde se está realizando (por ejemplo, las trayectorias circulares en la figura 3.3(h)). Sin embargo, en otras figuras coreográficas un pequeño error en la estimación de odometría significa una colisión con las paredes (por ejemplo al final de la figura 3.3(a)) o con las bailarinas (por ejemplo al principio de la figura 3.3(i)). Cuando el robot atraviesa un espacio estrecho cerca de las bailarinas, es muy probable que las bailarinas empujen accidentalmente al robot y afectan fuertemente a la odometría. En estas partes, el robot se mueve a una ubicación específica en el mapa en lugar de seguir una trayectoria y velocidades concretas.

[3] `http://wiki.ros.org/rosbag`

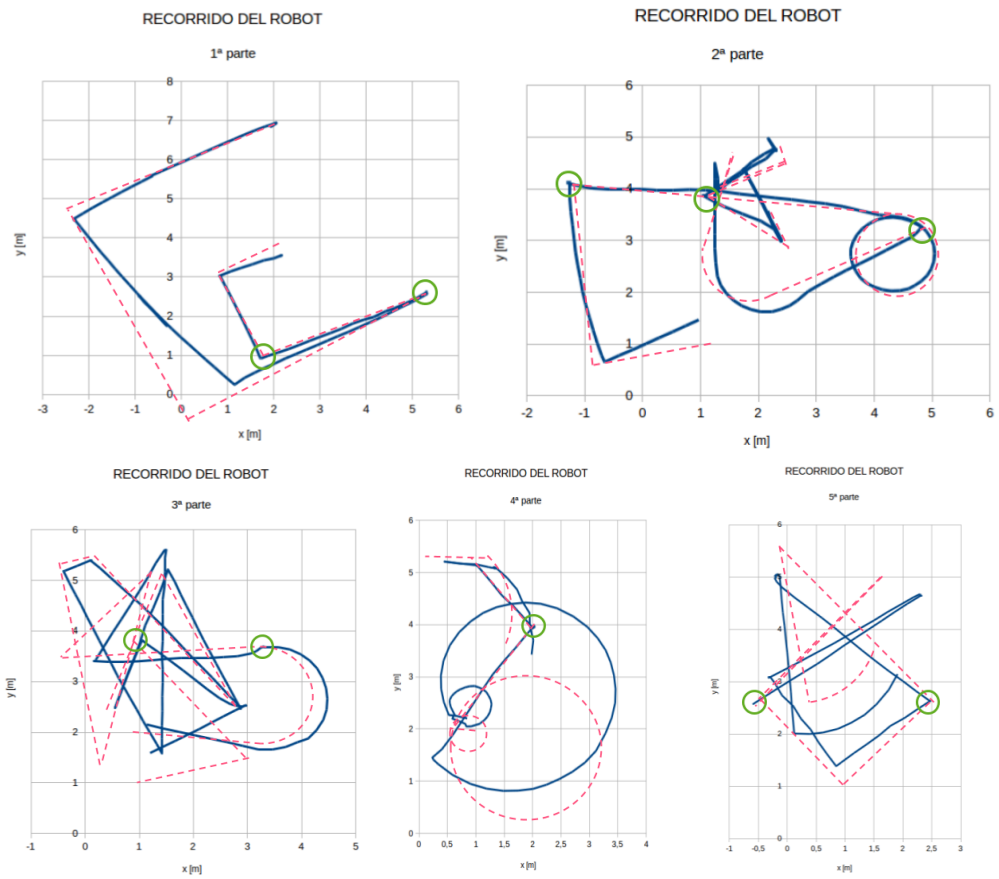


Figura 4.2: Trayectoria real realizada por el robot (linea azul continua) y trayectoria ideal diseñada (linea rosa discontinua). Los círculos verdes son puntos que se alcanzan con movimientos basados en objetivos en el mapa. Toda la coreografía esta dividida en 5 partes para mejor visualización.

Trayectoria real *vs* Trayectoria ideal. En la figura 4.2 se muestran cinco gráficas de la trayectoria del robot. La trayectoria azul es la trayectoria real del robot, la rosa discontinua es la trayectoria ideal programada que debería seguir y los círculos verdes son objetivos del mapa. Estas gráficas se han hecho con los datos extraídos del *Rosbag*. Los datos de las posiciones que el robot ha ido calculando (publicadas en el *topic /amcl_pose*) se han dividido en cinco tramos para una mejor visualización de las gráficas.

Se puede observar que las dos trayectorias no coinciden perfectamente. Esto es debido a que el robot va acumulando error en sus cálculos durante los movimientos basados en odometría, por múltiples motivos, como se discute más adelante. Los puntos en los que las lineas azules si coinciden con las rosas, marcados con un círculo verde, son puntos que se alcanzan con movimientos basados en objetivos en el mapa, como se ha explicado anteriormente. En las gráficas se ve que estos movimientos basados en el mapa son necesarios porque el error hasta ese momento ya era bastante significativo.

La odometría del robot va acumulando error por ejemplo debido a zonas del suelo poco propicias para que pasase el robot sobre ellas. En la zona izquierda del escenario, como se ve en las partes 1ª y 2ª de la figura 4.2, hay una ondulación en el suelo, difícil de apreciar a simple vista pero que afectaba bastante al robot, haciendo que este se desvíe más hacia la izquierda. Este error se iba acumulando pero con realizar movimientos basados en el mapa de vez en cuando, se corregía sin problemas.

También había cuatro trampillas a lo largo del escenario (con enchufes y cable de Ethernet en su interior) que hacían que el robot se tambalease mucho (incluso daba un pequeño salto), que aumentaba mucho el error de la odometría y a veces se quedaba atascado sin poder continuar o incluso se llegó a desenchufar el cable que conectaba la batería con el ordenador, haciendo que este se apagase. La mejor manera de solucionar esto era evitando las trampillas durante las trayectorias. Sin embargo, esto no siempre era posible porque había movimientos en los que tenía que pasar por una u otra, así que por seguridad, pusimos una velocidad baja en esos tramos, y añadimos sujeción extra tanto en las baterías del robot como en los bordes de las cuatro trampillas, con el fin de suavizar esos baches. Las trampillas pueden verse en las figuras 1.2 y 2.5.

Trayectoria calculada mediante odometría *vs* Trayectoria calculada mediante *AMCL* La figura 4.3 además de representar la trayectoria real del robot obtenida mediante el algoritmo de localización *AMCL* (azul) y la trayectoria ideal (rosa discontinua), representa la trayectoria calculada mediante odometría (verde). Con este gráfico comparamos lo diferentes que son las trayectorias obtenidas con el algoritmo de localización *AMCL* y con cálculos de la odometría. La trayectoria obtenida con el algoritmo de localización *AMCL* se corresponde con la trayectoria real, así que verificamos que hubiese sido imposible no utilizar el nodo *AMCL* (que permite realizar al robot movimientos basados en objetivos en el mapa), ya que con solo la odometría en 13 minutos de baile el robot se hubiese chocado o salido del escenario.

4.2.1. Análisis de velocidades.

Velocidad en movimientos basados en odometría. La figura 4.4 (a) muestra un ejemplo de velocidades lineales del robot a lo largo del tiempo, que corresponde con un desplazamiento hacia delante con una velocidad establecida de 0,17 m/s (publicada en el *topic /mobile_base/commands/velocity* a través del programa principal). Se ha representado calculando varias muestras de la velocidad de la siguiente manera: calculando la distancia incremental en x y en y mediante la resta de muestras consecutivas y dividiendo por la resta de tiempos consecutivos. Posteriormente, se han representado las velocidades calculadas respecto del tiempo, en segundos. Vemos que las muestras finales, la velocidad alcanza los 0.17 m/s, la velocidad de consigna. Sin embargo, la mayoría del tiempo la velocidad es menor que 0.17 m/s. Debido a esto, el tiempo que tarda el robot en recorrer una distancia es mayor que el tiempo estimado, por lo que durante los ensayos hubo que ajustar la velocidad o distancia recorrida, a fin de que el robot cumpliese con los tiempos especificados.

La figura 4.4 (b) muestra la velocidad angular del robot a lo largo del tiempo en un giro sin desplazamiento. El sentido de giro es positivo (hacia la izquierda) y de 90º, con una velocidad angular de consigna de 0.55 rad/s (también publicada

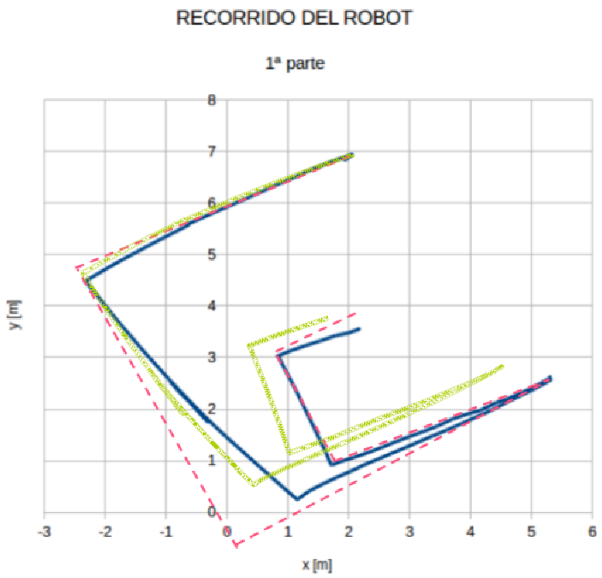


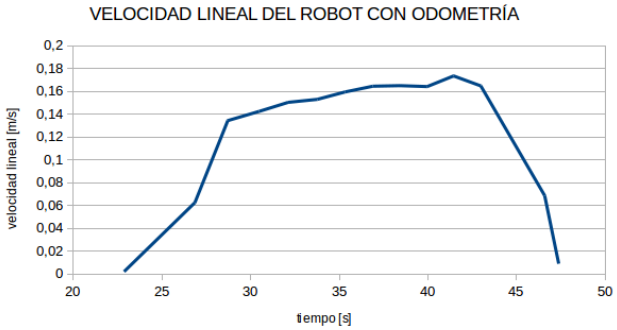
Figura 4.3: Comparación de trayectoria real obtenida gracias al sistema de localización *AMCL* utilizado (azul), trayectoria calculada mediante odometría (verde) y trayectoria ideal (rosa discontinua).

en el *topic /mobile_base/commands/velocity* a través del programa principal). Para obtener las velocidades angulares se ha seguido el mismo procedimiento que en caso anterior de la velocidad lineal, con la diferencia de que para obtener el ángulo girado se ha tenido que hacer una transformación de *cuaternios* a *ángulos de Euler*. Vemos que la velocidad angular no llega a alcanzar los 0.55 rad/s (velocidad angular de consigna).

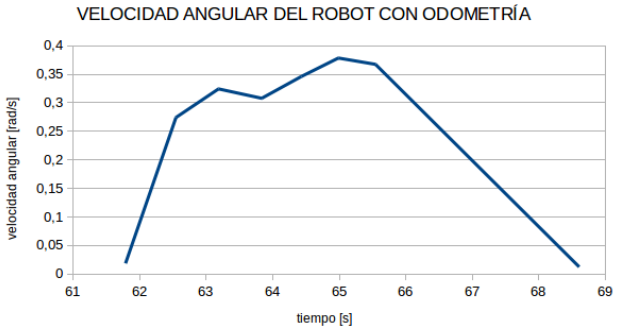
Las muestras al final de ambos recorridos (especialmente e de velocidad angular) están muy espaciadas temporalmente. Esto puede hacer parecer que la deceleración del robot sea lenta. Sin embargo, la realidad es que las aceleraciones y deceleraciones del robot son bastante rápidas. Esto se demuestra a continuación.

La tabla 4.1 contiene datos relacionados con las muestras representadas en la figura 4.4 (b): los incrementos temporales de una muestra a otra y los incrementos de ángulo. Es decir cuanto tiempo le cuesta girar determinado ángulo al robot en cada periodo de tiempo. De las muestras 2 a 7, el tiempo que le cuesta girar 12 radianes aproximadamente es parecido. Así que el robot gira con una velocidad mas o menos constante. En cambio, en la última de las muestras el incremento de tiempo Δ t es bastante superior a los demás y el incremento de ángulo $\Delta\ \theta$ muy inferior. Esto se traduce en que el robot estaba frenando y ya había parado prácticamente cuando se midió la muestra número 7.

Velocidad en movimientos basados en objetivos del mapa. Por otro lado, la velocidad lineal y la velocidad angular del robot cuando se dirige hacia un objetivo podemos verlas en la figura 4.5. Esta figura muestra como ejemplo las velocidades del robot al dirigirse hacia el primero de los objetivos, que



(a)



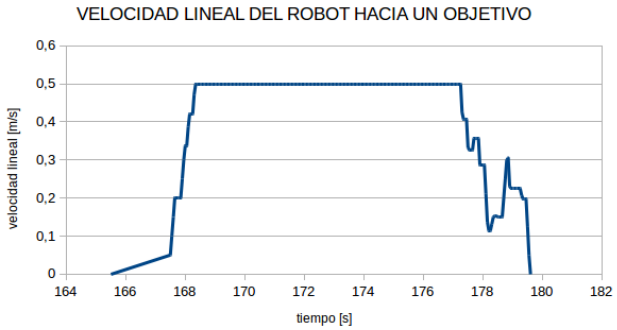
(b)

Figura 4.4: Velocidades lineal y angular del robot con odometría, en un desplazamiento hacia delante y en un giro positivo, respectivamente.

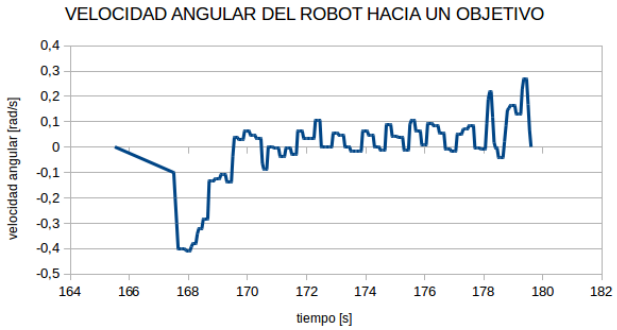
Tabla 4.1: Evolución de los incrementos de ángulo ($\Delta\ \theta$ (rad)) en relación a los incrementos temporales (Δ t (s)).

Nº de muestra	**Δ t (s)**	**$\Delta\ \theta$ (rad)**
2	0.76	11.92
3	0.63	11.88
4	0.65	11.46
5	0.6	11.8
6	0.56	12.08
7	0.56	11.74
8	3.06	2.21

sería el círculo verde de la figura 3.3. Estas figuras han sido realizadas cogiendo las muestras de velocidad correspondientes a ese movimiento (del *topic /mobile_base/commands/velocity*) y representándolas respecto del tiempo, en segundos. En la figura 4.5 (a) vemos que el robot una vez ha acelerado alcanza la velocidad máxima, 0.5 m/s, y se mantiene en ella hasta que tiene cerca el objetivo. Vemos que después de comenzar a frenar da un pequeño acelerón. Esto concuerda con lo que hacía el robot realmente en la mayoría de los movimientos basados en objetivos en el mapa: comenzaba a frenar cuando todavía le queda-



(a)



(b)

Figura 4.5: Velocidades lineal y angular que toma el robot al dirigirse al primer objetivo, desde que arranca hasta que frena.

ban unos centímetros para llegar al objetivo y volvía a acelerar un poco antes de frenar definitivamente. En la figura 4.5 (b) se ve en la zona central que el robot se encuentra continuamente corrigiendo su posición angular, para colocarse de cara a su objetivo. Además, al principio y al final del recorrido, este gira lo necesario para colocarse hacia la dirección del movimiento.

A la hora de establecer las metas, se pretendió que el robot se quedase con el ángulo de llegada y si tenía que girar hacerlo mediante movimientos basados en odometría. Esto se debe a que, aunque a las coordenadas x e y del punto llegaba con bastante precisión, si lo último que realizaba era un giro, se posicionaba con mucho error angular.

4.2.2. Análisis de posición.

Error de posición en movimientos basados en odometría. La tabla 4.2 muestra algunos ejemplos de movimientos lineales basados en odometría. En ella se recogen tanto los desplazamientos y velocidades lineales teóricas (instrucciones que se le dan al robot en el programa principal), como las reales calculadas a partir de los datos recogidos del *Rosbag* (estos datos corresponden al *topic /amcl_pose* aunque los cálculos de desplazamiento hayan sido realizados con odometría, ya que este *topic* nos da las distancias reales recorridas).

Tabla 4.2: Distancias y velocidades lineales teóricas y reales, realizando movimientos basados en odometría.

Distancia teórica (m)	**Distancia real (m)**	**Vel. lineal teórica (m/s)**	**Vel. lineal real (m/s)**
3	3.02	0.17	0.13
2	1.99	0.17	0.14
2.4	2.32	0.17	0.15
1	1.06	0.51	0.4
-1	-0.97	-0.17	-0.1
3	3.04	0.17	0.14
4	3.85	0.65	0.54
-3	-2.9	-0.5	-0.36

Tabla 4.3: Ángulos y velocidades angulares teóricas y reales, realizando movimientos basados en odometría.

Ángulo teórico (°)	**Ángulo real (°)**	**Vel. angular teórica (rad/s)**	**Vel. angular real (rad/s)**
-90	-72.4	-0.55	-0.34
180	164.58	0.55	0.25
-90	-83.45	-0.55	-0.21
-90	-82.56	-0.55	-0.19
90	82.98	0.55	0.29
90	89.76	0.55	0.14
-90	-70.43	-0.55	0.32
90	70.53	0.55	0.34
90	83.01	0.55	0.28
-180	-166.52	-1.65	-1.03

En todas las muestras la distancia real recorrida se aproxima bastante a la teórica, esto es gracias a que los cálculos de odometría no funcionan del todo mal y el robot considera que ha terminado el recorrido basándose en la distancia recorrida. En cuando a las velocidades podemos observar que la diferencia aquí entre la velocidad lineal real y la teórica es mayor, debido al peso del robot y de los componentes que lleva montados en el y al rozamiento del suelo. Todas las velocidades lineales reales son menores que las teóricas. Comparando entre unos desplazamientos y otros no podemos afirmar que el error dependa de la velocidad, es decir, a mayor o menor velocidad no hay más porcentaje de error. Además, entre muestras con misma distancia teórica y misma velocidad teórica los errores son variables. Por lo que las diferencias entre estos errores dependen en mayor medida de las condiciones del suelo, como ya se ha explicado en la Sección 4.2.

En la tabla 4.3 aparecen algunos ejemplos de movimientos angulares basados en odometría. Aparecen tanto los ángulos y velocidades angulares teóricas, como las reales. En este caso, el ángulo real es bastante diferente del ángulo teórico (es decir, el ángulo deseado). Este hecho ya lo hemos comprobado antes en la Sección 4.2, al ver lo diferentes que eran las trayectorias ideales de las reales. Aquí se reitera la necesidad de realizar algunos movimientos a los largo de la coreografía basados en objetivos en el mapa. En cuanto a la velocidad angular ocurre lo mismo que con la lineal: la real es bastante menos que la teórica debido a rozamientos. En este caso tampoco se puede decir que la diferencia

Tabla 4.4: Error en las coordenadas de los puntos objetivo de la coreografía

Objetivo	Posición Objetivo			Posición Alcanzada			Error		
nº	**x (m)**	**y (m)**	**θ (rad)**	**x' (m)**	**y' (m)**	**θ' (rad)**	**x-x'**	**y-y'**	**θ-θ'**
1	5.35	2.63	0.49	5.31	2.57	0.78	0.04	0.06	0.30
2	3.71	1.80	-2.63	3.71	1.81	-2.74	0.00	0.01	0.11
3	0.98	3.97	3.14	1.04	3.88	2.75	0.06	0.09	0.39
4	4.87	3.30	0.27	4.81	3.25	0.62	0.06	0.05	0.35
5	-1.35	4.21	2.67	-1.29	4.11	2.62	0.06	0.10	0.04
6	0.99	3.90	-1.72	1.05	3.82	-2.01	0.06	0.08	0.29
7	3.40	3.75	-0.25	3.30	3.68	0.10	0.10	0.07	0.34
8	2.12	4.07	0.52	2.03	3.98	0.70	0.09	0.09	0.18
9	2.56	2.60	-1.05	2.47	2.63	-0.84	0.09	0.03	0.21
10	-0.79	2.43	-2.66	-0.60	2.57	-2.53	0.19	0.14	0.13

entre los errores dependan del valor de la velocidad, o de si el ángulo es positivo o negativo.

Error de posición en movimientos basados en objetivos del mapa. En la tabla 4.4 aparecen los datos de desplazamientos en x y en y y giros de cada uno de los movimientos basados en objetivos en el mapa que lleva a cabo el robot. Se muestran las coordenadas objetivo, las alcanzadas y el error (es decir, la diferencia entre la alcanzada y la objetivo, en valor absoluto).

Los datos de las coordenadas objetivo son las establecidas en el programa principal. Estas se obtuvieron llevando al robot a la posición deseada con la orientación deseada y almacenando los valores de x, y y θ recogidos por el láser, con la siguiente instrucción en el terminal: *rostopic echo /amcl_pose* (más detalles en el Anexo D). Estos datos se tuvieron que recoger de esta manera debido a que los ejes x e y del mapa no estaban alineados con los laterales mapa y no se podía calcular la posición ellos de forma precisa.

Los datos de las coordenadas x, y y θ alcanzadas se han recopilado de los datos del *topic* donde se muestra la ubicación del robot (*/amcl_pose*) almacenado en el *Rosbag*. Estos datos no son muy exactos ya que en este topic las muestras se encuentran muy separadas temporalmente (hay pocas muestras), entonces no tenemos datos en los momentos concretos de tiempo que nos interesan.

La tolerancia en x y en y, es decir, el error máximo permitido, es 0.1 metros. Vemos que casi todos los errores de x y de y entran dentro de la tolerancia (el objetivo número 10 lo estudiaremos más adelante).

La tolerancia en *tita* es 0.05 radianes. Vemos que casi todos los errores de textittita son muy elevados en comparación a la tolerancia. Realmente el robot si que alcanza el ángulo objetivo con un error máximo de 0.05 radianes, lo que ocurre es que las muestras existentes están tomadas con saltos de tiempo muy variables.

Cuando el robot se encuentra en movimiento las muestras de la ubicación del robot se encuentran bastante seguidas temporalmente, pero cuando se aproxima una objetivo del mapa deja de haber muestras unos segundos antes de que el robot llegue a ésta. Vuelve a haber muestras en el comienzo del siguiente movimiento. Lo último que hace el robot al llegar al objetivo es orientarse, y por eso no tenemos datos que nos indiquen que el robot ha llegado realmente a su objetivo (en cuanto a la orientación), ya que no tenemos muestras del *topic* correspondientes a ese espacio temporal.

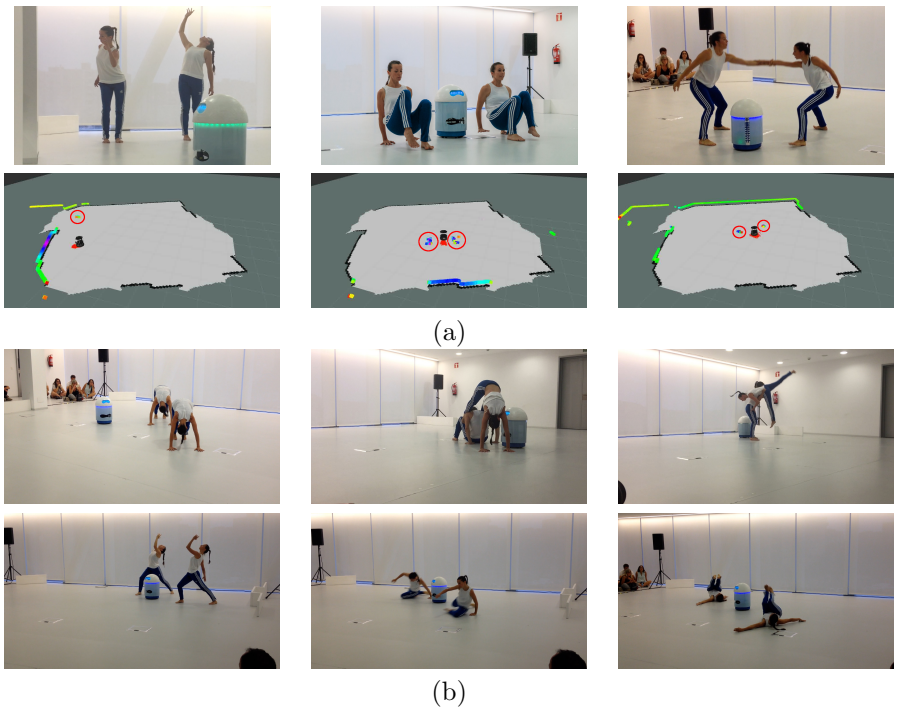


Figura 4.6: Resultados de la actuación. (a) Imágenes desde la perspectiva del público (arriba) y desde el ordenador supervisando el robot (abajo) usando el visualizador RViz. Las áreas grises y blancas corresponden con el mapa, las zonas coloridas corresponden con las medidas del escáner láser 2D. Nótese que los puntos del láser que corresponden con las bailarinas están marcados con un círculo rojo. (b) Ejemplos adicionales de figuras donde la sincronización de la velocidad y de la longitud de la trayectoria es esencial.

4.3. Experimentos de integración y actuación

En cuanto a experimentos de integración y actuación final, la coreografía se ejecutó completa mas de 20 veces, con lo cual el robot ha recorrido mas de 2000 m (100 m por cada coreografía completa) en los ensayos de integración, además de múltiples pruebas de cada uno de los movimientos por separado.

Como estaba previsto, la actuación final de la coreografía se realizó en el *festival de Trayectos*[4]. La actuación del festival consistió en dos pases de la coreografía, ambas ejecutadas perfectamente sin ningún problema, en un escenario completo, con más de 100 personas de público en cada una de ellas. La figura 4.6(a) muestra varios momentos de la actuación desde la perspectiva del público y desde el sistema de monitorización del robot. Nótese lo cerca que que se mueve el robot y las bailarinas en algunas de las figuras. La figura 4.6(b) muestra imágenes adicionales de la actuación donde podemos apreciar la dificultad para sincronizar algunas de las figuras coreográficas diseñadas.

El evento tuvo múltiples menciones en los medios locales y redes sociales:

[4]Vídeo disponible online: `http://robots.unizar.es/data/videos/robot17Etopia.mp4`

- https://www.facebook.com/danzatrayectos/videos/1105997732866164/
- http://www.danzatrayectos.com/portfolio/danzayrobotica/
- http://estoyenetopia.es/2017/06/12/los-robots-pueden-bailar
- https://www.facebook.com/etopia.milladigital/videos/1607487259284100/
- http://www.elperiodicodearagon.com/noticias/escenarios/trayectos-auna-tecnologia-danza-amplia-sus-escenarios_1202042.html
- https://videos.heraldo.es/aragon/un-robot-bailarin-el-protagonista-del-festival-trayectos-JeX5VR/

Capítulo 5

Conclusiones

5.1. Conclusiones del proyecto y Trabajo futuro

En este trabajo se ha integrado un robot de bajo coste en un espectáculo de danza contemporánea. El principal objetivo fue integrar el robot como otro miembro del equipo de baile. Por lo tanto, el robot debía realizar una coreografía precisa y repetible, sincronizada con la de las bailarinas. La coreografía fue diseñada y ejecutada con éxito en el festival de *Trayectos* según lo previsto, gracias al esfuerzo de un equipo multidisciplinar, compuesto por la compañía de danza y los ingenieros e investigadores de robótica y diseño.

Una de las principales lecciones aprendidas de este experimento han sido los esfuerzos necesarios para acortar las distancias entre el mundo del arte y el mundo de la robótica, empezando por los requerimientos y objetivos de cada uno de los grupos. Desde el punto de vista artístico, la estética y los movimientos precisos y repetibles eran más críticos que los sofisticados algoritmos de localización o los comportamientos reactivos, que no eran factibles de utilizar de forma sincronizada como se requería.

Como conclusiones más técnicas sobre los métodos utilizados, se ha comprobado la necesidad de navegar utilizando un modelo (mapa) del entorno y realizar algunos de los movimientos en base a este modelo, ya que el uso sólo de odometría acumula demasiado error. En cuanto a los resultados, se ha conseguido realizar con éxito trayectos bastante complejos de manera fiable y repetible, ya que en ninguna de las ejecuciones finales del festival hubo ningún choque del robot con las paredes o las bailarinas, pese a moverse de manera independiente muy cerca de ellas.

Este primer experimento ha abierto nuevos caminos a futuras colaboraciones. Como posibilidades de trabajo futuro podría integrarse un equipo de robots coordinados que se comuniquen entre ellos. Además se podrían incorporar algunos sensores adicionales, como los comentados en la subsección 2.1.1 o algún movimiento de la carcasa, como por ejemplo girar la cabeza alrededor de un eje.

5.2. Conclusiones personales

Con este trabajo no sólo he aprendido a utilizar el sistema operativo Ubuntu, ROS y otras herramientas sino lo que es más importante, he aprendido a aprender. Es decir, he aprendido a solucionar los problemas que iban surgiendo y he conocido la metodología de trabajo de un proyecto de experimentación.

Además, este proyecto me ha dado experiencia y responsabilidad al tratarse de un trabajo real con fecha límite. En primer lugar, debido a tener que tratar con distintos grupos de personas, concretar continuas reuniones con ellos, tomar decisiones conjuntas e ir mostrándoles la evolución del trabajo. Trabajar con gente que se dedica a otros campos (arte y diseño) me ha permitido ver la robótica desde otros puntos de vista, lo que ha hecho que me interesase más por ella. En segundo lugar, he tenido una responsabilidad real al tener que lograr un buen resultado de la programación y puesta en marcha del robot, consiguiendo exactitud y repetibilidad de los movimientos en los límites de tiempo establecidos, para un buen resultado de la actuación final.

Bibliografía

[1] CJ Chan Jin Chung. Integrated steam education through global robotics art festival (graf). In *Integrated STEM Education Conference*, pages 1–6. IEEE, 2014.

[2] Shunsuke Kudoh, Koichi Ogawara, Miti Ruchanurucks, and Katsushi Ikeuchi. Painting robot with multi-fingered hands and stereo vision. *Robotics and Autonomous Systems*, 57(3):279–288, 2009.

[3] Matthew TK Chan, Rob Gorbet, Philip Beesley, and Dana Kulič. Curiosity-based learning algorithm for distributed interactive sculptural systems. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.*, pages 3435–3441. IEEE, 2015.

[4] Ajay Kapur, Michael Darling, Dimitri Diakopoulos, Jim W Murphy, Jordan Hochenbaum, Owen Vallis, and Curtis Bahn. The machine orchestra: An ensemble of human laptop performers and robotic musical instruments. *Computer Music Journal*, 35(4):49–63, 2011.

[5] Damith Herath, Christian Kroos, et al. *Robots and Art: Exploring an Unlikely Symbiosis*. Springer, 2016.

[6] Chris Martin and Janet Hughes. Robot dance: Edutainment or engaging learning. *Proc. of the 23rd Psychology of Programming Interest Group PPIG 2011*, 2011.

[7] Toru Nakata, Tomomasa Sato, Taketoshi Mori, and Hiroshi Mizoguchi. Expression of emotion and intention by robot body movement. In *Proc. of the 5th Int. Conf. on Autonomous Systems*, 1998.

[8] Jean-Julien Aucouturier, Yuta Ogai, and Takashi Ikegami. Making a robot dance to music using chaotic itinerancy in a network of fitzhugh-nagumo neurons. In *Neural information processing*, pages 647–656. Springer, 2008.

[9] Takahiro Okamoto, Takaaki Shiratori, Shunsuke Kudoh, Shinichiro Nakaoka, and Katsushi Ikeuchi. Toward a dancing robot with listening capability: Keypose-based integration of lower-, middle-, and upper-body motions for varying music tempos. *IEEE Trans. on Robotics*, 30(3):771–778, 2014.

[10] Ryo Suzuki, Jaeryoung Lee, and Ognjen Rudovic. Nao-dance therapy for children with asd. In *ACM/IEEE Int. Conf. on Human-Robot Interaction*, pages 295–296, 2017.

[11] Hua Peng, Changle Zhou, Huosheng Hu, Fei Chao, and Jing Li. Robotic dance in social robotics—a taxonomy. *IEEE Trans. on Human-Machine Systems*, 45(3):281–293, 2015.

[12] Kazuhiro Kosuge, Tomohiro Hayashi, Yasuhisa Hirata, and Ryosuke Tobiyama. Dance partner robot-ms dancer. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 4, pages 3459–3464, 2003.

[13] Takahiro Takeda, Kazuhiro Kosuge, and Yasuhisa Hirata. Hmm-based dance step estimation for dance partner robot-ms dancer. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3245–3250, 2005.

[14] Taeyong Choi, Hyunmin Do, Gukhwa Kim, Jinho Kyung, and Jun Yong Moon. An example of performing art with robot. In *Int. Conf. on Ubiquitous Robots and Ambient Intelligence*, pages 905–906. IEEE, 2016.

[15] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. on Robotics*, 23(1):34–46, 2007.

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press.

[17] Tom Igoe. *Getting Started with RFID*. O'Reilly Media, Inc., 2012.

Anexo A

ROS

ROS (Robot Operating System) es un *framework* flexible de código abierto utilizado para gestión de plataformas robóticas y sensores relacionados. Esta compuesto por un conjunto de librerías software y herramientas que ayudan a construir aplicaciones robóticas dotadas de un comportamiento complejo y robusto. Actualmente ROS sólo funciona en plataformas basadas en Unix.

A.1. Conceptos básicos de los tutoriales realizados

La *wiki* de ROS tiene una gran variedad de tutoriales, desde la instalación del software hasta tutoriales de aplicaciones más complejas. En cada uno de los tutoriales está indicada la dificultad de los mismos y la explicación de estos es clara y ordenada. A continuación se explican las ideas clave de los tutoriales realizados.

- Los **paquetes** son la unidad de organización de software del código de ROS. Cada paquete puede contener librerías, ejecutables, scripts u otros tipos de ficheros. Cada vez que se realiza algún cambio en un programa o fichero contenido en un paquete, hay que colocarse (en el terminal) en el *path* donde se encuentra el paquete y compilar con la instrucción *catkin_make*.
- ***Manifests*** (ficheros con extensión .xml). Este tipo de ficheros contienen la descripción del paquete. Sirve para definir dependencias entre paquetes.
- Los **nodos** son ejecutables que utiliza ROS para comunicarse con otros nodos. Los nodos se comunican a través de **topics**. Un nodo puede tanto *publicar* mensajes en un topic, como *suscribirse* a un topic para recibir mensajes. Los **mensajes** son el tipo de datos que utiliza ROS para suscribirse a topics o publicar en ellos y pueden contener varios datos que pueden ser de diversos tipos. El **Master** es el nodo principal, que hace que los nodos de ROS se comuniquen entre ellos.
- Los **servicios** son otra forma en la que los nodos pueden comunicarse entre si. Los servicios permiten a los nodos enviar una **solicitud** y recibir una **respuesta**.

Anexo B

Manual de uso

En este *manual* se detallan las instrucciones para construir un mapa del entorno y almacenar en sus coordenadas los puntos para la realización en la coreografía de los movimientos basados en objetivos del mapa. También se explica como comunicarse desde un portátil con el ordenador incorporado en el robot y como lanzar el programa. Todo ello para permitir el uso de los resultados de este proyecto en futuros eventos. A partir de ahora en este manual: **PC** es el portátil con el que hay que conectarse en remoto y **Robot** es el ordenador de a bordo del robot.

B.1. Comunicar PC con Robot

Los pasos para poder manejar el Robot de manera remota son los siguientes:

1. Enchufar el *router* disponible a la red con las antenas colocadas y encenderlo.
2. Conectar PC y Robot a la red proporcionada.
3. Abrir NoMachine (aplicación de escritorio remoto gratuita) y seleccionar M̈irar las conexiones...: Seleccionar el ordenador de robot. Una vez hecho esto tendremos una ventana con el escritorio del Robot.

Si alguno de estos pasos da error o el PC no encuentra al Robot, es muy probable que sea porque PC y Robot no se *ven*. Para solucionar esto hay que seguir los siguientes pasos:

1. En el terminal del PC y del Robot escribir *ifconfig*. Apareceran las direcciones IP de ambos. La que nos interesa es la correspondiente a WLAN.
2. En el terminal de PC y Robot: *sudo gedit /etc/hosts*. Se abrirá un fichero. En este fichero tenemos que modificar la IP correspondiente al otro ordenador (donde aparece el nombre de usuario de este) y escribir la obtenida en el paso anterior.

Este paso habrá que seguirlo muchas de las veces si el Robot no tiene asignada una IP fija.

B.2. Obtención del mapa

Si se quiere usar visualización remota lo primero que hay que hacer es indicarselo al PC. Le indicamos que el *MASTER* es el Robot y así lo que el PC visualiza son los datos del Robot. En el terminal del PC escribimos:

- export ROS_HOSTNAME=etopiapc
- export ROS_MASTER_URI=http://ugv-1:11311

Los pasos a seguir para obtener un mapa del entorno son los siguientes:

1. En el Robot: *roslaunch turtlebot_bringup minimal.launch*. Para lanzar el robot.
2. En el Robot: *roslaunch turtlebot_navigation gmapping_demo.launch*. Necesario para realizar el mapa.
3. En el Robot: *roslaunch turtlebot_teleop keyboard_teleop.launch –screen*. Para teledirigir al robot moviéndolo por el entorno mientras crea el mapa con lo que ve con el láser.
4. En el PC: *roslaunch turtlebot_rviz_launchers view_navigation.launch*. Para visualizar lo que va añadiendo el láser al mapa y lo que está viendo en cada momento.
5. Recorrer el espacio del escenario hasta que tengamos un modelo completo de este.
6. Para guardar el mapa escribimos en el Robot: *rosrun map_server map_saver -f path_donde_se_quiera_guardar*

B.3. Lanzar el programa

Los pasos a seguir para lanzar el programa son muy simples pero hay que seguirlos correctamente. Estos son los siguientes pasos:

1. En el Robot: *roslaunch turtlebot_bringup minimal.launch*. Para lanzar el robot.
2. En el Robot: *roslaunch amcl_demo.launch map_file:= /home/robotica/Escritorio/residenciaDefinitivo.yaml*. Esta ruta en nuestro caso, si no habría que poner aquí la ruta al fichero del mapa *yaml*. Para abrir el mapa y poder hacer uso de la navegación con *AMCL*.
3. En el Robot: *rosrun simple_navigation_goals figuras_node*. Para lanzar el programa. *figuras_node* es el nombre del programa y *simple_navigation_goals* es el paquete donde se encuentra.
4. En el PC: *rosrun rviz rviz*. Cargar el fichero *danza.rviz* (o el nombre que le hayamos puesto), donde se encuentran los datos necesarios del mapa. Con esto visualizaremos desde el PC las trayectorias del robot, lo que está viendo el láser en cada momento y su incertidumbre.

5. Al abrir *rviz* lo primero que habrá que hacer será decirle al robot dónde se encuentra, para que a partir de ahí pueda ubicarse por si mismo. Esto se hace seleccionando en la ventana principal *2D pose estimated* y con el ratón seleccionando el punto del mapa donde se encuentra y apuntamos con la flecha que aparece hacia la direccion en la que está enfocada la parte delantera del robot.

B.4. Almacenar coordenadas

Para realizar movimientos basados en objetivos en el mapa es necesario almacenar previamente las coordenadas a las que queremos que el robot se dirija, para poder indicárselo en el programa principal cuando corresponda.

Para ello lo único que hay que hacer es colocar al robot en los puntos a los que queremos que se dirija, con todos los nodos de la Sección B.3 lanzados, exceptuando el nodo del programa principal *figuras_node*. En el programa *rviz* tendremos que comprobar que el robot conoce su ubicación (es decir, si alinea correctamente los límites del mapa con los limites del escenario que ve el láser). Lo único que hay que hacer es escribir en el terminal *rostopic echo /amcl_pose* y almacenar los valores que ahí nos aparecen.

Anexo C

Imágenes de ensayos y actuaciones

De manera complementaria a las imágenes que aparecen en la memoria de este trabajo, se muestran en este Anexo más imágenes de los ensayos y de las actuaciones finales.

C.1. Imágenes de los ensayos

Las imágenes correspondientes a los ensayos son las siguientes: C.1, C.2, C.3, C.4 y C.5.

C.2. Imágenes de la actuación final

La imagen correspondiente a la actuación final es la imagen C.6

Figura C.1: Comienzo de la coreografía. Puesta en marcha del robot y robot realizando sus trayectorias.

Figura C.2: Distintas etapas del primer *movimiento* de la coreografía.

Figura C.3: *Movimiento* de la coreografía en la que las bailarinas simulan ser robots.

Figura C.4: Comentando con otros participantes del proyecto posibles problemas o soluciones. Bailarinas calentando.

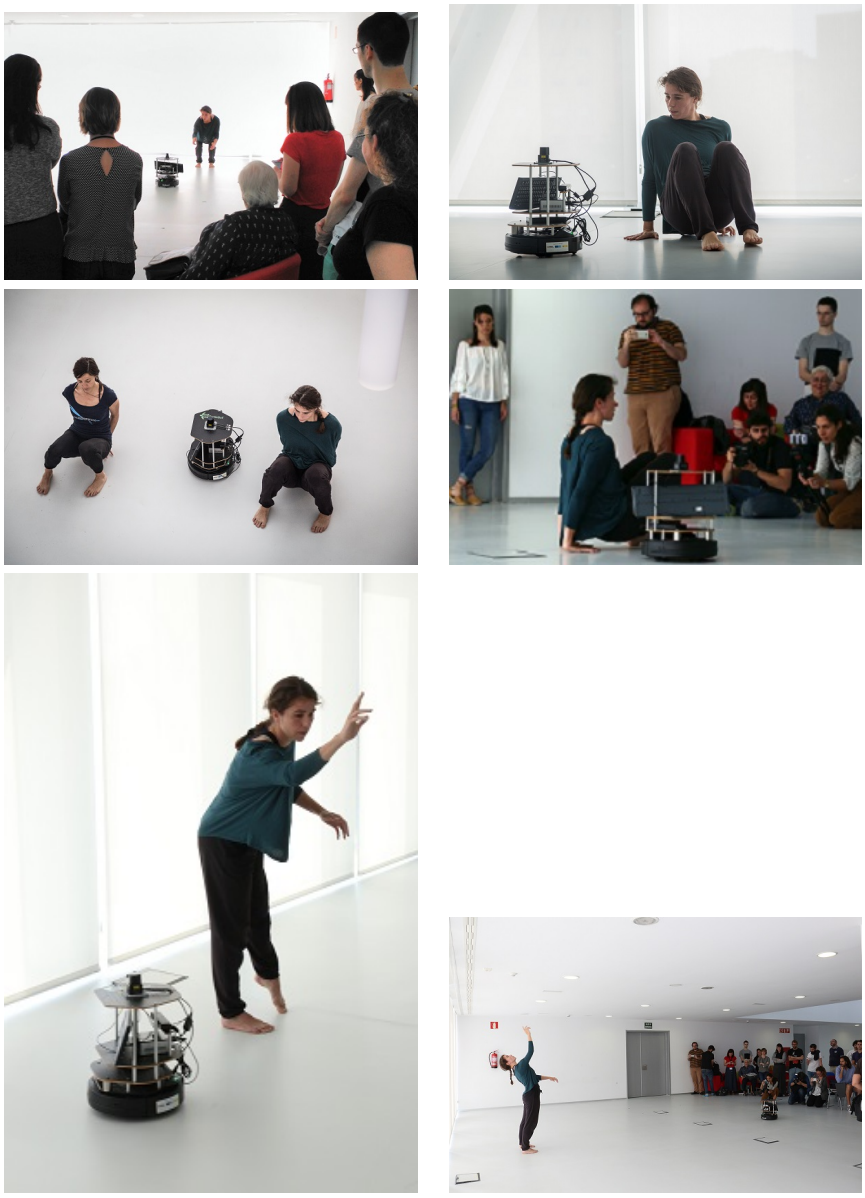


Figura C.5: Rueda de prensa, donde varios periodistas formularon sus preguntas y realizaron fotos y vídeos.

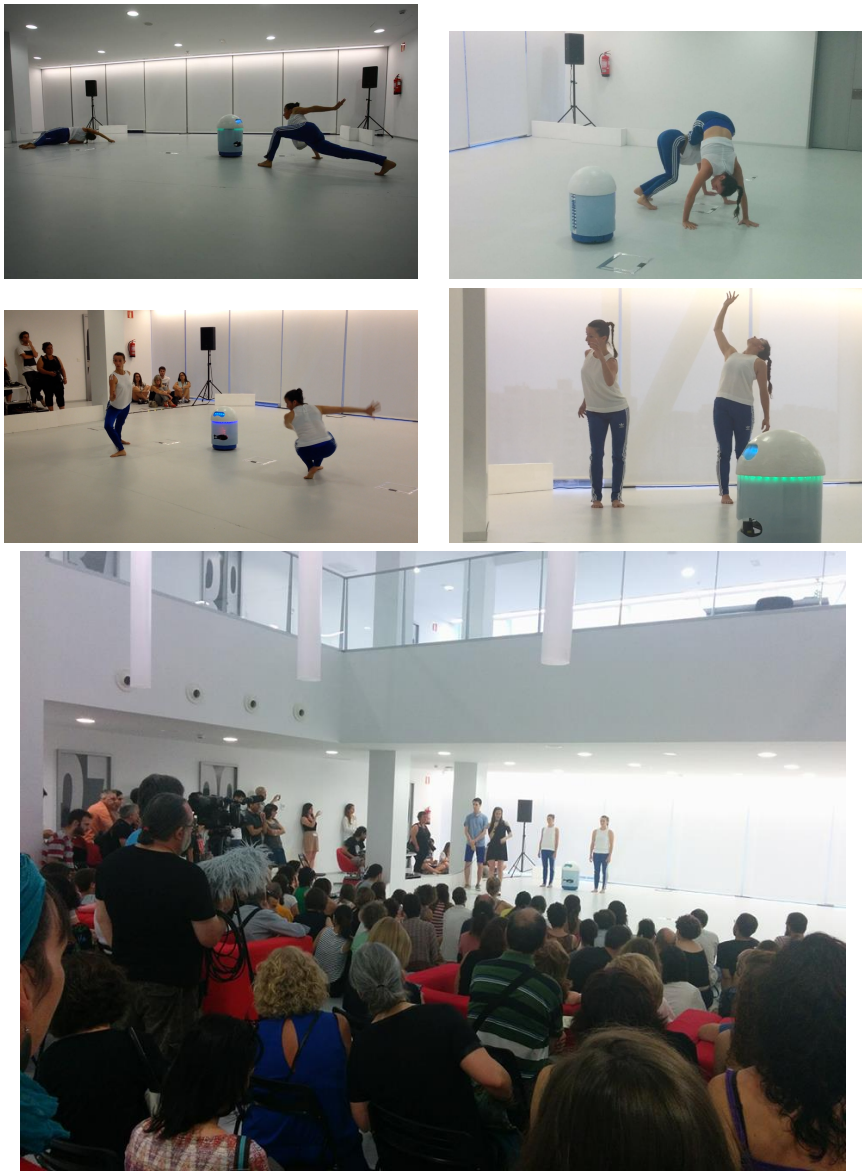


Figura C.6: Imágenes que muestran partes de la actuación final del festival de *Trayectos*, donde se ve a las bailarinas y al robot en el escenario e imagen de la presentación al público, donde se explicó en que consistía la coreografía.

Anexo D

Integrating an autonomous robot on a dance and new technologies festival

Se ha enviado el siguiente resumen en inglés de este trabajo a una conferencia sobre robótica (*ROBOT 2017 - Third Iberian Robotics Conference*[1]), y ha sido aceptado. Está pendiente de publicación y presentación a finales de 2017.

[1] `https://grvc.us.es/robot2017/`

Integrating an autonomous robot on a dance and new technologies festival

P. Abad[1], M. Franco[1], R. Castillón[1], I. Alonso[1], A. Cambra[1], J. Sierra[2], L. Riazuelo[1], L. Montano[1], A. C. Murillo[1]

[1] DIIS-i3A. University of Zaragoza, Spain
[2] CUD. Zaragoza, Spain

Abstract. This paper presents the results of a project to integrate an autonomous mobile robot into a modern dance performance at a dance and new technologies festival. The main goal is to integrate a simple low cost mobile robot into the dance performance, in order to study the possibilities that this kind of platforms can offer to the artists. First, this work explains the process and design to embed the robotic platform into the choreography theme. Another contribution described in this work is the system architecture proposed and built to make the robot behaviours match the artists requirements: precise, synchronized and robust robot movements. Finally, we discuss the main issues and lessons learned for this kind of robotics and arts applications and summarize the results obtained, including the successful final live performance results.

Keywords: Autonomous robots. Arts. Modern Dance.

1 Introduction

Over the last years we have witnessed the introduction of robotics into many and diverse fields. There are numerous recent attempts to integrate new AI & robotics related technologies on novel areas of application, and the Art fields are not an exception. There are many recent initiatives to study how Arts could integrate and benefit from robotics: from more generic arts and education applications [4], to approaches to create paintings made by robots [9], to build interactive sculptural systems [2] or to experiment with robotic orchestras [7]. Our work is focused on exploring possibilities of integrating a robotic member into a modern dance team.

This paper presents the results of an experimental project where an autonomous mobile robot is integrated in a performance of a modern dance festival. The main goal is to study the possibilities that a low cost mobile robot can provide artists involved on this performance. The dance and new technologies festival, Trayectos[3], is being held in a public Art & Technology Center (Etopia) in Zaragoza, and the dance company involved in this project is *Tarde o Temprano Danza*[4].

[3] http://www.danzatrayectos.com/en/laboratorio-de-danza-y-nuevos-medios/
[4] http://tardeotempranodanza.wix.com/tardeotempranodanza

Fig. 1. This project presents the results of an experiment in collaboration with different teams at an *Arts and Technology* center to integrate a TurtleBot robot into a dance and new technologies festival.

One of the challenges that makes this project original is the goal of making the robot one member of the dance team. The dance company requires the robot to have a pre-defined and accurate behaviour, since the performance is not an improvisation from their side. This means that the robot *dancer* should also *learn* its part of the choreography, to be perfectly synchronized with the rest of the team. One of the most important lessons learned during this project is that common theoretical and technical problems in robotic research are often not the main challenges faced in this type of application. The requirements and expectations from the non-robotics teams of this multi-disciplinary projects implied strong restrictions in aspects such as: very high accuracy of certain movements, restrictions on the kind of sensors that can be used, reactive behaviours that were not acceptable or limitations on the artificial markers or elements that can be added to the scenario or dancers outfit.

The main results and contributions presented in this work are:

- The analysis and proposed design of a **housing** to cover the base robotic platform (TurtleBot 2) in order to embed it into the choreography theme.
- Proposed **architecture**, software and hardware modules, to achieve the different robot behaviours following the dance company requirements.
- Details of the final *robot choreography* and **results** of the **live performance** at the festival *Trayectos* 2017.
- Discussion of the main issues and **lessons learned** for this kind of robotics and Arts application.

1.1 Related Work

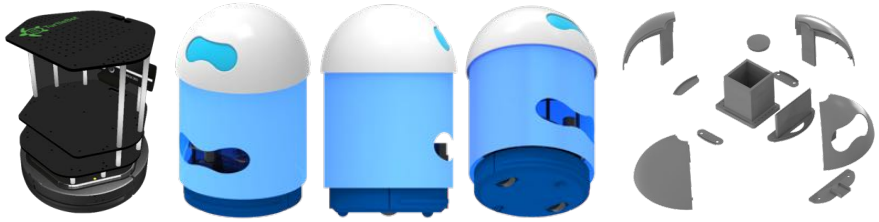
As previously mentioned, we find multiple attempts to integrate robotics in Art fields, not only to study what Art can gain from robotics, but also the other way around. There are recent research publications compiling different experiments on robotics and art experiments [6] and specific sessions on top robotics research venues[5]. We can find related work from top universities involved in professional artistic venues[6], or dissemination actions involving different Arts and robotics experiments and venues[7] for educational purposes [10].

Robots and Dance. Particularly relevant to our project are works related to performing Arts, in particular dance and theater. We can find earlier studies that were focused on the possibilities of expressing emotion and intention trough the robot body movements [11]. Among more recent works, we find researchers that propose autonomous systems that are able react to different sounds and music, adapting for example to the music tempos or motion restrictions [1][12]. There are also many studies that consider robotic dance as a tool for therapy activities and medical applications, such as the proposed work for therapy with children [14]. More details and examples of robotics and dance applications can be found on the survey work in [13].

Closer to our goals, there are previous approaches that study how to integrate robots on a dance activity or performance. There are researchers studying how to design a robotic dance partner, analyzing the human-robot coordination [8][15]. They propose a system that reacts to physical interactions thanks to the use of an omni-directional mobile base equipped with force sensors. There is little experimental prior work focused on the artist requirements to design the robot choreography as part of the artists performance [3]. The focus on integrating the robot as one member of a hybrid human-robot team is a key component in our work. Differently from most prior works on dance, we have targeted our work to evaluate the reach of a low cost platform, in collaboration with local artists, making emphasis on the robot being one more member in the dance team. This goal has highlighted the need of certain capabilities in the robot such as accuracy and exact time synchronizations, rather than reactive behaviours and improvisation from more autonomous robotic platforms in prior work.

Autonomous robot navigation. Additionally, there is essential related work to our project in the areas of mapping and localization. We have used well established approaches for 2D mapping [5] and particle filter based localization [16], using a 2D scanning sensor mounted on a mobile robot, as detailed in the following sections. These algorithms have enabled us to robustly run the dance performance, even with cluttered background and frequent large occlusion to the robot sensors field of view and with very heterogeneous lighting and clutter conditions across executions.

[5] http://www.roboticart.org/iros2017/
[6] http://artpower.ucsd.edu/dancing-robot-huang-yi-kuka/
[7] https://www.robofest.net/index.php/current-competitions/graf



(a) (b) (c)

Fig. 2. 3D models of original Turtlebot 2 (a), housing designed in this project, from different points of view (b) and 3D printed pieces, including housing parts and support pieces to hold sensors and LEDs (c).

2 Robot Design

This section describes the design and main components of the robotic platform used in this project. To help convey the message of the artists for the performance, about the mutual influence between robot and human, a housing was developed for the robot model used, TurtleBot 2[8]. Fig. 2 shows a rendered image of the original TurtleBot 2 model together with three different points of view rendered of the robot equipped with the designed housing.

Robot Housing. The housing was designed and manufactured in collaboration with the choreography team to establish a common conceptual idea. This idea emerged in a creative session that took place at the beginning of the project, lead by the design specialists of the team. It served so that the different parts of the project followed the same line of work with some common final objectives towards the final demo at the dance festival. The design of the housing was strongly influenced by some restrictions inherent to the robotic platform. The most significant ones are:

- The size of the open hole around the laser sensor, to allow its operation.
- The maximum weight restriction of 5kg on the base.
- The need to leave easy access to charging and device ports.

Additionally, the artists and choreographers from the dance company had strong preferences to make the robot look *alive* and cute for the public. This limited the position of robot *face* and eyes and encouraged the possibility to simulate *breathing*. One of the main challenges at this step was to give unity to all these different requirements and restrictions. As it can be seen in Fig. 2 (b), besides the main housing (head built from 3D printed pieces and body cover consisting of a translucent and light cylinder), there were multiple small additions to give a clean and modern appearance, such as the eyes area transparent cover and multiple stickers to make the look as homogeneous as possible. Fig. 1 shows the final appearance of the robot.

[8] http://www.turtlebot.com/turtlebot2/

(a) (b) (c) (d)

Fig. 3. Different stages of the final appearance of the robot (a-c) and main components, scanning laser rangefinder and LED stripes, used in the final version (d).

Sensors and actuators. In addition to the robot housing, several additional pieces were developed to serve as support elements, both for the housing itself and for other internal elements and sensors. The vast majority of these parts have been manufactured with a 3D printer, and therefore need to be split into pieces that fit the measurements of the printing area of the 3D printer available. Fig. 2(c) shows the 3D model of all these support pieces, including head cover pieces, internal holds for sensors and other small components. Fig. 3 shows different stages of the robot as the different components were built and attached.

Different sensors were considered during the initial stages of this project, but many were discarded mainly for development time and weight restrictions or robustness and accuracy requirements. Besides the basic TurtleBot on-board sensors, as shown in Fig. 3d, we added a Hokuyo laser scan (since it was the most robust choice to map the environment give the requirements and restrictions of the scenario) and programmable LED stripes (since they were the only available additional component to give expressive capabilities to the system that fit the scenario and artists requirements and restrictions).

Finally, we considered adding additional moving parts to the robot that had to be discarded due to weight limits on the base platform.

3 System Description

This section describes the main modules of the proposed system: 1) the modules used to map the environment where the robot is going to perform its tasks; 2) the system architecture built to program all the mobile robot tasks.

3.1 Modeling and localizing the robot in the environment

Requirements and Set up. The location where the performance is happening is an open space rather than a conventional scenario (more details in next Section). This means there is no physical separation between the public and the performance space. To ensure certain static geometric components that the robot can easily incorporate into the model (map) of the working environment, some separating elements are needed. When building these elements, the main requirements were:

(a) (b)

Fig. 4. Images from training sessions where we can see the panels used to limit the robot map of the performance space (a). Sample map from the performance area (b).

1. To follow the aesthetic requirements of the artists.
2. To avoid any unnecessary occlusions of the performance to the public.

As a low cost solution, a few expanded polystyrene blocks were made and arranged throughout the performance space. In Fig. 4(a) we can see some of these blocks (white small panels) during training sessions of the choreography.

chrome://bookmarks/

Mapping. The map collection was carried out using a standard mapping node available in ROS[9]. This mapping node implements the well known SLAM algorithm from [5] and allows us to easily build a map from 2D laser scan data.

This map allows the robot to locate itself in any moment as long as the mapped place layout does not change. Fig. 4(b) shows a map obtained on the performance scenario area. Note that the *obstacle-map* view shown in that figure enlarges the unreachable regions around not only the obstacles but also the limits/walls of the scenario. This means that the robot is not allowed to get too close to the actual physical elements to avoid collisions. This feature is a common navigation safety feature that needs to be taken into account carefully when designing the choreography, since when using the map navigation features, the robot can not reach any position very close to any obstacle (including the dancers). We need to disable this for dynamic obstacles if we want to be able to have the robot move very close to the other dance team members, as it can be seen in many of the examples in this paper, such as Fig. 4(a).

Localization. The robot localizes itself on the available map using the 2D laser scan data. This method enhance the first attempt based only on odometry which is only useful for short periods of time. The system uses the available AMCL localization ROS node[10]. This localization approach is a well known probabilistic 2D localization method which implements an adaptive Monte Carlo localization approach using a particle filter to track the robot pose on a known map, as described in [16]. Fig. 5 shows some of the main ingredients of this approach. The robot uncertainty about its location depends on how well the algorithm is able to align the current 2D scan data with the known map of the environment.

[9] http://wiki.ros.org/gmapping
[10] http://wiki.ros.org/amcl

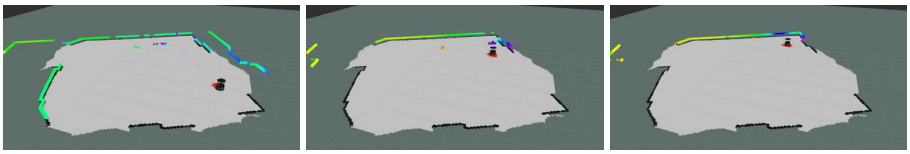


Fig. 5. Robot planning and executing a trajectory to reach a given target location in the map (visualization in *RosViz*). The thick green and blue contours matching map contours are laser scan points aligned with the map. The thinner pink line starting at the robot represents the planned trajectory. The red segments around the robot represent the robot location uncertainty.

3.2 Architecture of the System

The system used to control the robot in this project has two main components, to control the robot movements and to control the LEDs.

Motion module. The system to control the robot has been built using ROS (Robot Operating System) and its available capabilities for the TurtleBot2 platform [11]. There are two essential elements for our motion module: odometry estimation and localization on the scenario map. Figure 5 shows several snapshots of the robot executing a trajectory following a local planner to reach a goal. However, there are many situations where it is impossible to perform this planned movement towards a goal. This is because often the movement to be performed by the robot must be more precise in both position and time of execution, because it has to be synchronized with the music. In these cases, the robot has to execute movements purely based on its odometry.

A diagram of the main ROS nodes and topics used in our system is shown in Fig. 6. It is composed by several ROS modules which provide the robot with the capability of autonomous navigation (red on *move_base*) and localization (green ones *map_server*, *amcl*). The navigation is based on a reactive obstacle avoidance method and also incorporates a planner for computing the path to the goal assigned. The localization modules provides the location of the robot during the performance. It is based on a pre-built map and a particle filter which provides a localization. The blue module *figuras_node* is synchronized with the music, and it is in charge of the control of all the performance choreography (details on the specific trajectories generated are given in next section). It sends velocity commands to the low lever robot controller in order to perform a movement, and also sends goals in the global frame of the map to the navigation system.

LEDs module. We used 2 LED strips, each one containing 32 *RGB*-programmable LED components. We use the high-level API provided by the manufacturer [12] and focus on the colour and timing programming of each LED. The LEDs humanize the robot. One strip is used to emulate the robot eyes and other body

[11] http://wiki.ros.org/indigo, http://wiki.ros.org/turtlebot_navigation/
[12] https://github.com/arvydas/blinkstick-python

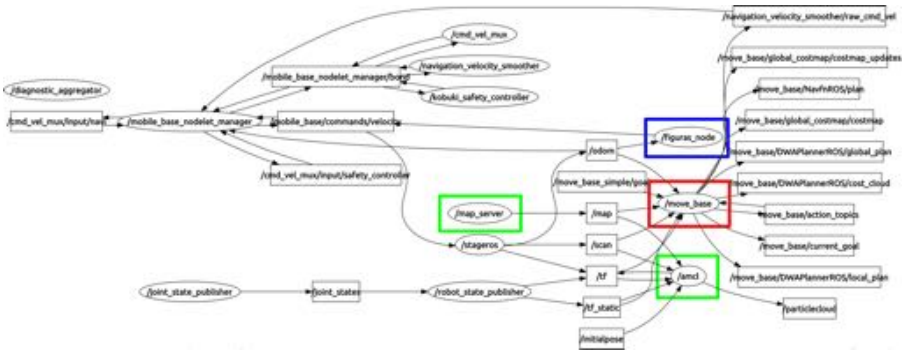


Fig. 6. Diagram of the main ROS modules used in our system.

parts. The second strip is just for aesthetic purposes. In addition to the aesthetic goals, these LEDs help the dancers to see in which modules the robot is running at all times. This allows them to synchronize with the robot and between them using the LEDs as a reference.

The LEDs module is run in parallel threads to the main process at all times, selecting different behaviours depending on the part of the choreography, such as moving forward, backward or accelerating, stopped looking to the public, stopped waiting to synchronize with the timings or rotating to align with the public. The LED stripe enables separated programming of the RGB color of each LED, therefore once the stripes were attached to the robot we could easily configure which led correspond to the eyes, heart or head of the robot.

4 Experiments

This section details the set up used and the main results obtained from the experiments performed before and during the dance festival.

Set up. The final configuration and on-board sensors of the robotic platform used in all the experiments have been detailed in previous Sec. 2. About the particular details of the software platform, our system uses the Indigo ROS distribution[13] over Ubuntu 14.04 installed on the main on-board computer. The TurtleBot 2 platform has an embedded computer to control the low level motion modules of the robot, and an additional computer mounted on the robot plates where we the main software components, including ROS, are installed. Our platform is equipped with a Intel NUC 6i5SYH (Intel Core i5-6260U, 8GB DDR4).

Scenario. The performance of this project is executed on a large open space that belongs to a residence in the center of Art and Technology Etopia[14], shown in

[13] http://wiki.ros.org/indigo
[14] https://www.zaragoza.es/ciudad/etopia/

Fig. 7. Panoramic view of live performance scenario. Red rectangle highlights where the actual performance happens. The public occupied the rest of the open space.

Fig. 7. Not all the space is mapped by the robot because a large part needs to be kept free for the public.

4.1 Choreography

The final choreography designed for the robot is represented in the diagrams from Fig. 8. The choreography consists of several figures whose execution took around 12 minutes, with three different music themes combined.

Robot movement types. As previously mentioned, the movements of the robot can be divided on two groups.

odometry-based movements. Most of the specific trajectories had to follow certain geometric shapes and traverse exact distances in specific time intervals. In order to have tight control over this, many trajectories were designed to be executed using trajectory generation based purely on odometry estimation and verification (blue trajectories on the figures).

map-goal-based movements. In several key steps along the choreography, the robot executed trajectories using the autonomous navigation module (green trajectories in the figures). These trajectories are executed according to the localization algorithm (AMCL ROS node) that locates the robot within the existing environment map. This requires varying waiting times at the end of this figures/movements, which are not ideal for the artists, but were necessary to avoid collisions due to inevitable accumulated odometry drift over time.

Robot movement error analysis. We executed the whole choreography trajectories in the simulation environment available for TurtleBot within ROS, but we observed that the execution times and odometry drifts were not equivalent, probably due to the varying weight, balance and set up of our platform compared to the original TurtleBot model incorporated in the simulator. It is not possible to provide quantitative analysis of errors for specific movements, because of the difficulty of repeating exactly the same experiment multiple times.

For the *map-goal-based* movements, the acceptable error is directly configured in the ROS node configuration parameters, because they allow us to establish the acceptance threshold to consider that the robot has reached a position.

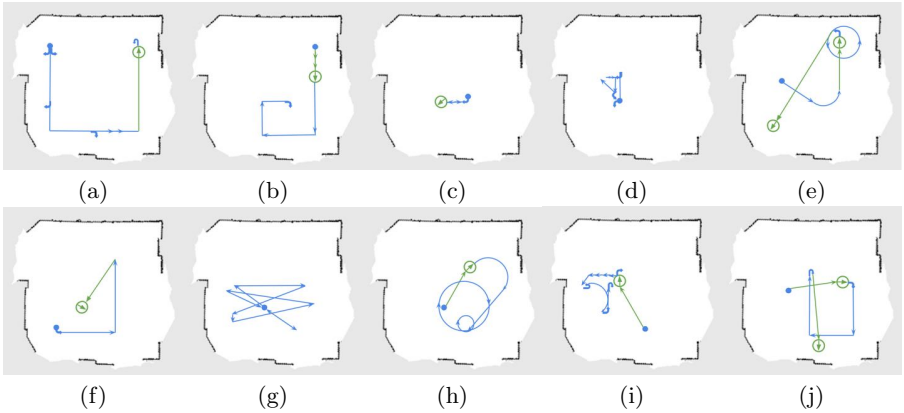


Fig. 8. Robot Choreography. Sorted diagrams (a)-(j) representing the movements performed by the robot. Blue: *odometry-based* movement. Green: *map-goal-based* movement. Thick blue dot in each diagram: starting point in that part. The blue arrows indicate rotations and directions of movement.

For the *odometry-based* movements, there is higher variability on the error values and significance. The accumulated error from the odometry was not significant when the accuracy of the trajectory shape and timing were more important for the artists than the particular location in the map where it is being performed (e.g., the circular trajectories in Fig. 8(h)). However, in other choreography figures a small error in odometry estimation means a collision with the walls (e.g., end of Fig. 8(a)) or with the dancers (e.g., beginning of Fig. 8(i)). When the robot traverses a narrow space near the dancers, it is very likely that the dancers accidentally push the robot and strongly affect the odometry. In this parts, the robot moves to a specific location in the map rather than following a particular trajectory and speed.

4.2 Live Performance Results

As planned, the final demonstration of the choreography was performed on the *Trayectos festival*[15]. The festival performance consisted on two executions of the choreography, both of which were executed perfectly without any issues, in a full scenario, with more than 100 people in the public for each of them. Figure 9(a) shows several moments from the performance from the public perspective and from the robot monitoring system. Note how close the robot and the dancers move in some of the figures. Figure 9(b) shows additional images of the performance where we can appreciate the difficulty to synchronize some of the designed choreography figures.

[15] Video available online: http://robots.unizar.es/data/videos/robot17Etopia.mp4

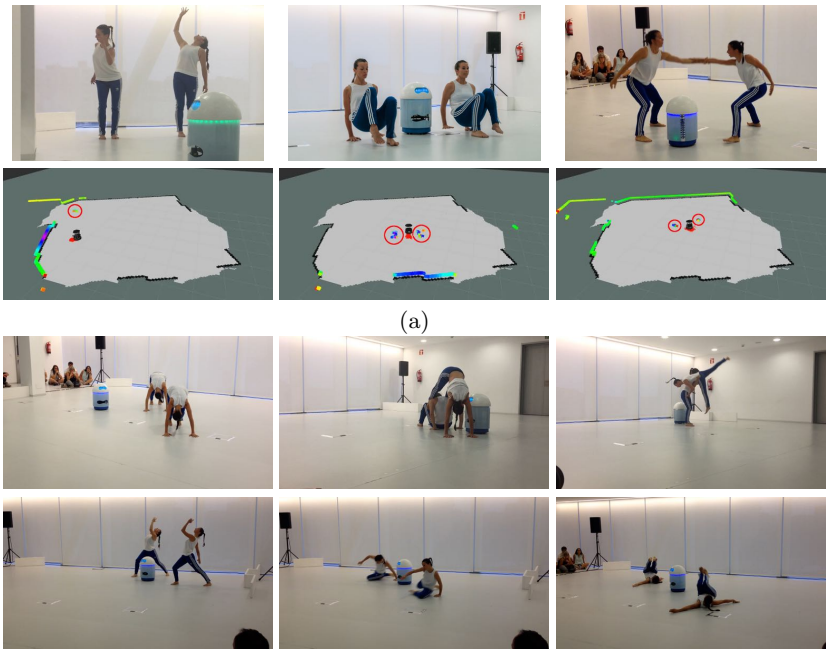


Fig. 9. Results from the live performance. (a) images from the public perspective (top) and from the robot monitoring logs (bottom) using the RViz viewer. The white and gray areas are the loaded map, the coloured points correspond to the current 2D laser scan measurements. Note the laser points that correspond to the dancers marked with a red circle. (b) Additional examples of synchronization figures where speed and length of the trajectory synchronization is essential.

5 Conclusions

This paper has presented our experience and results integrating a low cost robotic platform on a dance performance. The main goal was to integrate the robot as part of the team in the dance performance. Therefore, the robot should have an accurate and repeatable choreography synchronized with the human dancers. The choreography was successfully designed and executed on the *Trayectos* festival as planned, thanks to a multidisciplinary team effort, between dance company, design and robotic engineers and researchers. Some of the most interesting lessons learned from this experiment have been the efforts required to bridge the gaps between the Art and Robotics worlds, starting from requirements and goals from each of the groups. From the artistic side, the aesthetics, accurate and repeatable movements were more critical than sophisticated localization algorithms or reactive behaviours, which were not feasible to used in a synchronized manner as required. This initial experiment has open new paths to future

collaborations, where integrating coordinated multi-robot teams and additional sensors and reactions are part of future work possibilities.

References

1. Aucouturier, J.J., Ogai, Y., Ikegami, T.: Making a robot dance to music using chaotic itinerancy in a network of fitzhugh-nagumo neurons. In: Neural information processing. pp. 647–656. Springer (2008)
2. Chan, M.T., Gorbet, R., Beesley, P., Kulič, D.: Curiosity-based learning algorithm for distributed interactive sculptural systems. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. pp. 3435–3441. IEEE (2015)
3. Choi, T., Do, H., Kim, G., Kyung, J., Moon, J.Y.: An example of performing art with robot. In: Int. Conf. on Ubiquitous Robots and Ambient Intelligence. pp. 905–906. IEEE (2016)
4. Chung, C.C.J.: Integrated steam education through global robotics art festival (graf). In: Integrated STEM Education Conference. pp. 1–6. IEEE (2014)
5. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. on Robotics 23(1), 34–46 (2007)
6. Herath, D., Kroos, C., et al.: Robots and Art: Exploring an Unlikely Symbiosis. Springer (2016)
7. Kapur, A., Darling, M., Diakopoulos, D., Murphy, J.W., Hochenbaum, J., Vallis, O., Bahn, C.: The machine orchestra: An ensemble of human laptop performers and robotic musical instruments. Computer Music Journal 35(4), 49–63 (2011)
8. Kosuge, K., Hayashi, T., Hirata, Y., Tobiyama, R.: Dance partner robot-ms dancer. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. vol. 4, pp. 3459–3464 (2003)
9. Kudoh, S., Ogawara, K., Ruchanurucks, M., Ikeuchi, K.: Painting robot with multi-fingered hands and stereo vision. Robotics and Autonomous Systems 57(3), 279–288 (2009)
10. Martin, C., Hughes, J.: Robot dance: Edutainment or engaging learning. Proc. of the 23rd Psychology of Programming Interest Group PPIG 2011 (2011)
11. Nakata, T., Sato, T., Mori, T., Mizoguchi, H.: Expression of emotion and intention by robot body movement. In: Proc. of the 5th Int. Conf. on Autonomous Systems (1998)
12. Okamoto, T., Shiratori, T., Kudoh, S., Nakaoka, S., Ikeuchi, K.: Toward a dancing robot with listening capability: Keypose-based integration of lower-, middle-, and upper-body motions for varying music tempos. IEEE Trans. on Robotics 30(3), 771–778 (2014)
13. Peng, H., Zhou, C., Hu, H., Chao, F., Li, J.: Robotic dance in social roboticsa taxonomy. IEEE Trans. on Human-Machine Systems 45(3), 281–293 (2015)
14. Suzuki, R., Lee, J., Rudovic, O.: Nao-dance therapy for children with asd. In: ACM/IEEE Int. Conf. on Human-Robot Interaction. pp. 295–296 (2017)
15. Takeda, T., Kosuge, K., Hirata, Y.: Hmm-based dance step estimation for dance partner robot-ms dancer. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. pp. 3245–3250 (2005)
16. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press