Integración de un Planificador de Trayectorias Parametrizado en la Arquitectura Robótica ROS

Mario A. Rueda-Castro, Javier Monroy, Francisco-Angel Moreno y Javier Gonzalez-Jimenez
Dpto. Ingeniería de Sistemas y Automática, Instituto de Investigación Biomédica de Málaga (IBIMA),
Universidad de Málaga, 29071, Málaga, España.
{malexrc|jgmonroy|famoreno|javiergonzalez}@uma.es

Resumen

En este trabajo se presenta la integración de un nuevo planificador reactivo de movimiento dentro de la arquitectura robótica ROS. Este planificador, desarrollado originalmente como parte de la librería MRPT y basado en el Espacio de Trayectorias Parametrizado (TP-Space), amplia las posibilidades de navegación de las plataformas robóticas actuales mediante la generación de trayectorias tanto circulares como no-circulares, así como la posibilidad de contemplar la geometría 3D del robot (no restringiéndose a formas prismáticas). La integración se ha realizado garantizando la compatibilidad con el stack de navegación autónoma más popular de la plataforma ROS, adaptando el funcionamiento de dicho reactivo a la interfaz de comunicación estándar de ROS, siendo este compatible con los planificadores globales existentes. Para validar este trabajo, se han realizado múltiples pruebas de navegación en entornos de interior (tanto simulados como reales), con presencia de obstáculos y dónde la plataforma robótica es equipada con diversas cámaras RGB-D y un escáner láser.

Palabras clave: Robótica móvil, MRPT, TP-Space, ROS, Navegación autónoma, Evitación de obstáculos, Generación de Trayectorias, Planificador Reactivo.

1 Introducción

En la actualidad, existe una gran variedad de vehículos que pueden navegar sin necesidad de intervención humana, entre los que destacan los coches autónomos [2], en pleno proceso de expansión, los vehículos aéreos no tripulados (UAVs) como, por ejemplo, los *drones* utilizados para labores de mapeado 3D de superficies [19], los transportadores autónomos (AGVs), de uso extensivo en la industria [24], y los robots móviles asistenciales [6].

Uno de los problemas más exigentes a los que se enfrentan los vehículos autónomos es la *planificación del movimiento*, la cual conlleva una alta complejidad, incluso en escenarios simplificados [11], debido a la variedad de situaciones a las que debe hacer frente, especialmente en entornos dinámicos compartidos con personas. Por esto, tener a disposición un conjunto versátil de algoritmos de planificación capaces de adaptarse al entorno resulta de especial interés.

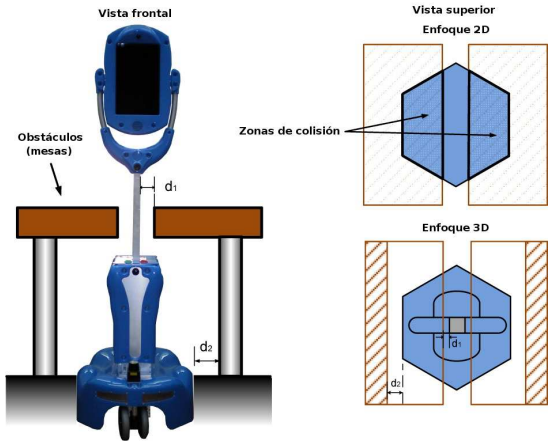


Figura 1: Caso favorable para la navegación de TP-Space 3D. [12] Enfoque 2D: base proyectada en altura, forma prismática. Enfoque 3D: composición de prismas, forma no prismática.

En los últimos años se han desarrollado diferentes alternativas de planificadores de movimiento para robots [3, 8, 1], algunos de los cuales se han implementado como módulos de arquitecturas *software* robóticas entre los que destaca el paquete `move_base` en ROS[1], que se ha convertido en un estándar *de facto* en la comunidad robótica móvil.

Este paquete ofrece una estructura robusta dividida en dos partes: *estática* o *deliberativa* y *dinámica* o *reactiva*, los cuales comparten información para generar trayectorias óptimas de navegación según el algoritmo implementado. La parte *estática* tiene en cuenta factores como el mapa del entorno y la situación de los obstáculos la última vez que se vieron, mientras que la *dinámica* actualiza esta información con los valores actuales de los sensores. Esta separación aumenta la versatilidad del sistema ya que permite sustituir cada parte

[1] https://www.ros.org/

de manera independiente por otro submódulo que modifique el comportamiento del planificador. Debido a esto, el paquete `move_base`se ha utilizado en numerosos proyectos ofreciendo resultados muy positivos [15, 17].

Sin embargo, no existe entre estos algoritmos implementados en ROS uno que incorpore información sobre la forma tridimensional del robot y adapte la navegación en función de ésta. En su lugar, habitualmente se utiliza un polígono o circunferencia (denominado *footprint*) cuya extrusión vertical incluya toda la forma del robot. Esta simplificación limita habitualmente el número de trayectorias generadas, incluso impidiendo el paso del robot por zonas que, aun siendo físicamente transitables, no son tenidas en cuenta al ser de un tamaño inferior al *footprint* del robot. En la Figura 1, se muestra un ejemplo de este caso, donde un robot debe desplazarse entre dos mesas alineadas y dicha zona es demasiado estrecha para su base. Si se tiene en cuenta la forma en 3D del robot, la navegación sí sería posible.

Una excepción es el planificador reactivo propuesto como parte de la librería *Mobile Robot Programming Toolkit* (MRPT)[2], formada por conjunto de paquetes y aplicaciones *software* multiplataforma para robótica móvil. Este planificador de trayectorias, además de tener en cuenta la forma real en 3D del robot (aproximada en forma de múltiple bloques 2D), permite la generación de trayectorias no circulares, ofreciendo un mayor abanico de posibilidades a la hora de seleccionar el tipo de camino a seguir para alcanzar el destino deseado. Para ello hace uso del denominado Espacio de Trayectorias Parametrizado (TP-Space) [3], cuyo funcionamiento queda descrito en la sección 2. No obstante, y a pesar de los continuos esfuerzos en la integración de esta librería con la arquitectura robótica ROS (véase por ejemplo [20] dónde Rodríguez *et. al* realizaron una primera aproximación a este problema), dicho planificador de trayectorias no es compatible con la arquitectura de planificación de caminos propuesta en el paquete estándar `navigation`, sino que opera de manera independiente. Este aspecto limita considerablemente su implantación en sistemas robóticos que usen ROS, siendo por tanto una opción no considerada en la mayoría de los casos prácticos.

Este trabajo contribuye con la integración del reactivo 3D de la MRPT como módulo compatible con el paquete de navegación estándar de ROS, de manera que el planificador estático calcule trayectorias de navegación y el nuevo planificador reactivo genere los comandos de movimiento necesarios para efectuar la navegación teniendo en cuenta la forma real en 3D del robot. Esta integración permite, por ejemplo, cambiar de manera sencilla entre diferentes planificadores estáticos de ROS, modificar el tipo de sensores a utilizar o la fuente de la odometría manteniendo las virtudes del reactivo 3D de la librería MRPT.

Más concretamente, este trabajo contribuye con: i) la integración del módulo con el planificador global de ROS, ii) la corrección del ángulo final del robot al llegar a la meta y iii) el diezmado de las nubes de puntos capturadas por los sensores para optimizar el procesamiento. El código desarrollado se ofrecerá como *software* libre para su uso para la comunidad robótica.

El módulo creado se ha validado mediante una serie de experimentos utilizando el modelo de un robot con forma no prismática (Fig. 1) equipado con 2 cámaras de profundidad a diferentes alturas y un escáner láser. Este modelo se ha utilizado tanto en un conjunto de experimentos con obstáculos y mediciones simuladas como en experimentos en el mundo real. Los resultados demuestran que la solución implementada permite la navegación con ROS de robots no prismáticos evitando las posibles colisiones durante la trayectoria.

2 Trabajos Relacionados

La planificación de movimiento consiste en, dado un escenario o entorno de trabajo, una pose inicial y una pose final, encontrar una sucesión de comandos de velocidad que logren una navegación desde la pose inicial a la final en ausencia de colisiones. El enfoque más relevante y difundido para afrontar este problema es el paradigma de planificación híbrido introducido por Brooks [5]. Este aúna conocimiento estático y conocimiento dinámico del entorno para ofrecer una arquitectura robusta y apropiada para la navegación de plataformas robóticas.

En el terreno de arquitecturas destacadas que implementen este paradigma, el *stack* `navigation` de ROS, compuesto por diferentes módulos encargados de la información sensorial, la gestión del mapa de ocupación, o la localización del robot entre otros, ofrece en su núcleo el paquete `move_base`[3]. Se trata de un servidor asíncrono capaz de computar los comandos de velocidad necesarios para alcanzar una meta objetivo, dividiendo el problema en dos niveles: global y local. El nivel global, por una parte, utilizando la información que provee el servidor del mapa e información sensorial, entre otras cosas, crea un mapa de celdas

[2] https://www.mrpt.org/

[3] http://wiki.ros.org/move_base

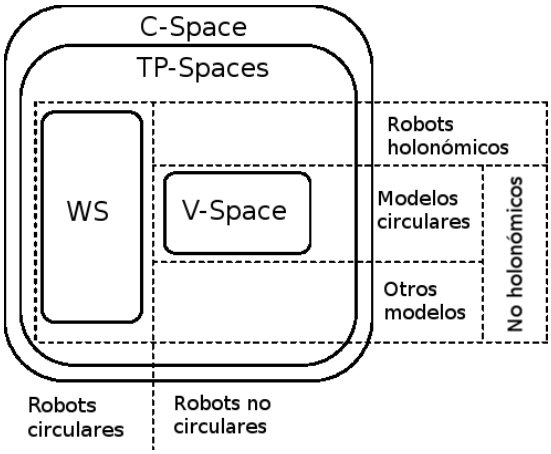


Figura 2: Jerarquía existente en las representaciones del espacio empleadas en los algoritmos de planificación de movimiento. [4]

asignando un valor de coste a cada una de ellas. Sobre este mapa se genera una ruta de navegación utilizando un algoritmo de búsqueda de caminos y se transmite al nivel local. El nivel local, a su vez, crea otro mapa de costes solo empleando el rango de visión del robot. En función de este mapa y la ruta recibida del nivel global se computa el comando de velocidad necesario para alcanzar la meta a tenor del algoritmo de planificación seleccionado.

Los planificadores de movimiento reactivos (*i.e.* planificadores locales en `move_base`) más utilizados en la actualidad emplean métodos de muestreo para elegir el comando de velocidad adecuado en cada momento de entre un abanico finito de posibilidades [8, 10]. Dichos comandos de velocidad están restringidos por la cinemática del vehículo en cuestión, su geometría, grados de libertad, así como por las características del escenario en el que la navegación toma lugar. Para atender a todas estas restricciones a la hora de planificar posibles trayectorias se emplea una abstracción denominada *espacio de configuraciones* o *C-Space* [16]. Esta abstracción eleva la dimensionalidad del problema de navegación atendiendo a las características del robot, siendo por lo general necesario aplicar algún tipo de simplificación o condicionante para hacer el problema computacionalmente eficiente. De entre las diferentes propuestas, limitar los posibles movimientos del vehículo utilizando solo trayectorias circulares (de fácil caracterización) es a día de hoy una de las opciones más empleadas. Este nuevo subespacio, denominado *V-Space* (Fig. 2), es de gran relevancia para la comunidad al estar presente en los módulos más importantes de ROS para la planificación reactiva del movimiento: *Dynamic Window Approach* (DWA) y *Trajectory Rollout* (TR) [8, 10].

Pese a la eficiencia y gran difusión de los métodos basados en el *V-Space*, estos reducen grandemente el espacio de búsqueda de trayectorias, rechazando zonas de navegación válidas y seguras. Con la máxima de conseguir alcanzar estas zonas, surge *TP-Space* [3]. Este nuevo sub-espacio de trabajo se basa en aplicar transformaciones paramétricas al espacio general *C-Space* mediante las denominadas *Parameter Trajectory Generators* (PTGs) [12]. Gracias a esta parametrización, el *TP-Space* contempla múltiples modelos de trayectorias, incluyendo los comúnmente usados modelos circulares y atendiendo a la geometría 3D del robot. En este sentido se puede considerar una generalización del *V-Space* tal y como se aprecia en la Fig. 2, y representa el principal motivo por el que en este trabajo se propone la integración de dicho planificador reactivo en la arquitectura robótica ROS.

3 Planteamiento del Sistema de Navegación

La arquitectura robótica ROS, y más concretamente el *stack navigation*, ofrece una interfaz común para la definición e integración de nuevos planificadores, tanto globales como locales. No obstante, el planificador de trayectorias parametrizado en el que estamos interesados en este trabajo no se adapta a esta interfaz, sino que emplea directamente la información sensorial para generar los comandos de movimiento, no haciendo uso, por ejemplo, de los resultados del planificador global. Por resolver este problema se ha implementado un nodo auxiliar o envoltorio que encapsula el reactivo nativo de la MRPT y traduce sus mensajes para adaptarlos a dicha interfaz. Las diferentes adaptaciones y problemas resueltos son detallados a continuación.

3.1 Mapas de Coste VS Mapas de Ocupación

El paquete ROS `navigation` hace uso de los denominados *mapas de coste* en los que a cada punto del espacio se asigna un valor numérico que representa su riesgo de navegación. De esta manera se pueden generar trayectorias óptimas (con mínimo coste) tanto en velocidad como en seguridad, penalizando ciertos movimientos con una reducción en su módulo. Este sistema, además, está compuesto de diferentes capas superpuestas. En cada capa se reproduce información que puede ser de diferente carácter semántico como, por ejemplo, la zona de seguridad y aproximación del robot a las personas (costes sociales y proxémicos [22]). Sin embargo, MRPT es incompatible con este tipo de mapas, ya que realiza la navegación sobre esce-

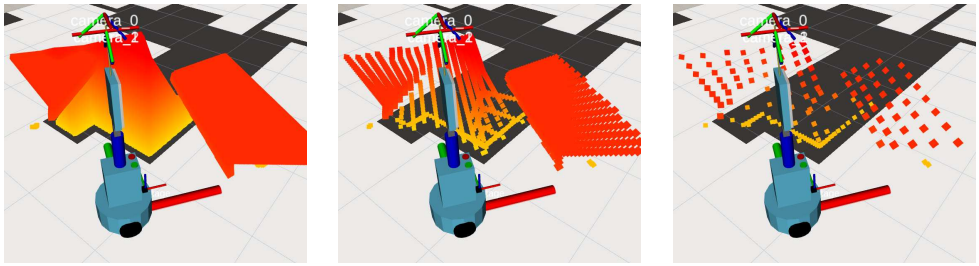


Figura 3: Visualización en RViz del diezmado de nube de puntos. Tres ejecuciones con distintos conjuntos valores de diezmado. Respectivamente (cámaras, láser): (1,1), (100,4) y (1000,8). Color nube de puntos: componente Z (menor-amarillo, mayor-rojo).

narios en forma de *mapas de ocupación* [18]. La información semántica de estos mapas solo hace referencia a la ocupación de las celdas, por lo que no existe una forma sencilla de traducir mapas de coste a mapas de ocupación sin perder información semántica. Recalcar que este problema no se ha resuelto, en su lugar, en este trabajo se ha propuesto la utilización de uno de los nodos incluidos dentro del paquete `mrpt_navigation`, denominado `mrpt_local_obstacles` que está dedicado a recopilar la información sensorial que el robot recibe durante un intervalo de tiempo en un *buffer*. Este *buffer* permite la posibilidad de recordar obstáculos observados en el pasado pero que se encuentran fuera del campo de visión del robot en el instante actual, generando una representación tridimensional de la escena de utilidad. No se cuenta con la semántica de los mapas de coste.

3.2 Comunicación entre planificador global, local y MRPT

Tal y como se mencionó en la introducción, `move_base` ofrece una estructura robusta dividida en dos capas: *reactiva* y *deliberativa*. Una de las misiones de este trabajo es hacer compatible *TP-Space* con la capa *reactiva* de `move_base` asegurando la comunicación con la capa *deliberativa*.

El funcionamiento del sistema comienza con la generación por parte del planificador global de una trayectoria de navegación, la cual es dividida de forma regular en puntos de paso o *waypoints*, por los que el robot va a navegar secuencialmente y de manera reactiva usando el planificador local de la MRPT. Una vez alcanzado un *waypoint*, se determina el siguiente en la secuencia y se continua navegando hasta alcanzar el objetivo final.

En el caso probable de que durante la ejecución de una navegación reactiva llegue una nueva ruta global, la anterior se desecha y se actualiza el objetivo local, permitiendo de esta forma una navegación flexible ante cambios en el plan global.

La selección del siguiente *waypoint* se realiza en función de un parámetro configurable que permite *muestrear* la secuencia de puntos de paso a una determinada distancia, permitiendo que se seleccionen puntos más cercanos o más lejanos los unos de los otros. Este aspecto es importante para el uso de las trayectorias parametrizadas del *TP-Space* ya que la velocidad máxima de navegación depende de la distancia entre puntos de paso, llegando a hacerse imposible la navegación si estos están demasiado cerca entre sí.

Finalmente, una vez alcanzado el último *waypoint*, se procede a alinear la orientación del vehículo con la de la pose final, ya que el reactivo de la MRPT adopta como objetivo para la navegación únicamente la posición de la meta y no la orientación, al contrario de los planificadores locales más populares de ROS. Así, para conseguir diseñar un planificador local que ofrezca las mismas prestaciones que el resto, es necesario corregir la orientación del robot cuando la posición de la meta se haya alcanzado, de forma que el ángulo del robot se alinee con el ángulo deseado en el destino. Para ello, se determinará el sentido de giro apropiado en función de los ángulos actual y objetivo, girando posteriormente a velocidad constante hasta conseguir el alineado.

3.3 Diezmado de las Nubes de Puntos

Para obtener una navegación que tenga en cuenta la forma 3D del robot es necesario utilizar sensores que capturen información no solo de un plano (como los escáneres láser) sino del espacio tridimensional, como los láseres 3D o las cámaras RGB-D. Como se mencionó en la sección 3.2, el nodo desarrollado utiliza un *buffer* para almacenar la información de los sensores, en este caso en forma de nubes de puntos. Sin embargo, debido a la densidad de puntos que se capturan, el cómputo se ralentiza tanto que se hace imposible navegar.

Para solucionarlo se ha utilizado la *PointCloud library* [21] para diezmar la nube de puntos antes

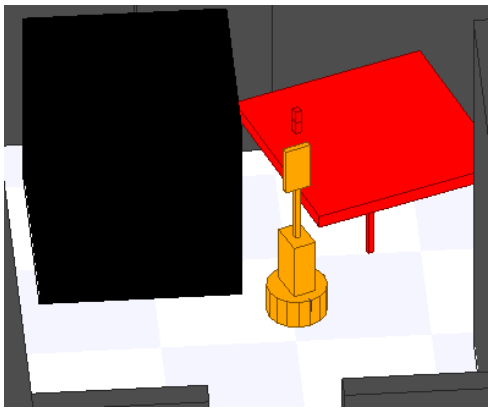


Figura 4: Visualización del simulador Stage. Modelo del robot (de frente) como conjunto de prismas naranja. Equipado con: 3 cámaras RGB-D (prismas rojos superpuestos), 1 escáner láser (plano azul, base del robot). Dos tipos de obstáculos: mesas unipata (rojo), cajas (negro). Paredes (gris).

de la inserción en el *buffer*. El diezmado ha sido realizado con un filtro de índices con parámetros ajustados de forma experimental pero que pueden ser modificados por el usuario según sus requisitos (Fig. 3). Para evitar problemas de sincronización entre los datos 3D y 2D obtenidos, estos últimos también se diezman aunque en distinta medida.

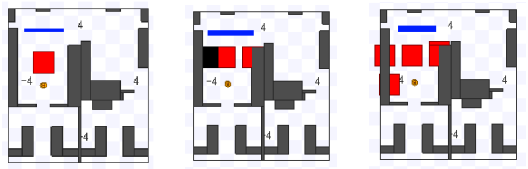
Finalmente hay dos aspectos a tener en cuenta relacionados con este tema: i) dado que la representación de los obstáculos detectados en el *TP-Space* (denominados *TP-Obstacles*) se basa en aplicar a los puntos detectados la forma del robot y la trayectoria elegida, este diezmado no afecta significativamente a la precisión de la navegación, y ii) el diezmado de los datos 2D solo se aplica al reactivo de la MRPT y no en el resto de nodos de ROS que utilizan los escáneres láser (como el localizador), ya que se perdería robustez al navegar.

4 Experimentos y Conclusiones

Para evaluar la integración de este novedoso planificador reactivo dentro de la arquitectura ROS, se presentan a continuación resultados de pruebas realizadas tanto en un ambiente simulado como en el mundo real.

4.1 Experimentos Simulados

El entorno de simulación escogido para esta evaluación ha sido Stage [23]. Este simulador robótico ofrece interesantes características como la simulación de sensores láser y cámaras RGB-D, así como la posibilidad de modelar escenarios 3D de superficie plana de forma sencilla. Tareas imprescindibles para la navegación autónoma como son la localización del robot y el mapeado del entorno [7] se consideran resueltas y no forman parte de esta evaluación.



a b c

Figura 5: Vista superior de tres escenarios de prueba simulados en Stage. (Lineas azules: metas)

El escenario de simulación utilizado está compuesto por varias habitaciones interconexas que asemejan una oficina (Fig. 5) . Se han incorporado dos tipos diferentes de obstáculos: mesas, para probar la capacidad de procesar el espacio tridimensional del reactivo, y cajas, fácilmente detectadas por el sensor láser con la finalidad de eliminar posibles caminos alternativos y, por otra parte, aliviar la divergencia entre ruta global y trayectoria local. Además, se ha simulado el modelo de un robot no prismático equipado con 3 cámaras de profundidad situadas: la primera de ellas situada sobre la cabeza del robot con una inclinación de 60º de pitch, las demás mirando hacia los lados con un yaw y pitch de $\pm 90^o, 50^o$ respectivamente. También se le ha equipado un sensor láser situado en la base del vehículo. (véase Fig. 4)

Para evaluar de forma seria y objetiva los resultados del la integración del planificador de movimientos considerado en este trabajo se hace necesario una comparativa con otros planificadores de movimiento en diferentes entornos. Concretamente, en este trabajo proponemos una comparativa entre: (i) *Dynamic Window Planner (DWA)*, configurado de tal forma que se penaliza severamente trayectorias que difieran mucho de la ruta global previamente computada, (ii) *Trajectory Rollout Planner* (TR) [10], un algoritmo de muestreo básico de comandos de velocidad, y (iii) dos versiones del reactivo basado en *TP-Space*, operando solo con información del escáner láser 2D (TP) y añadiendo información 3D de las cámaras RGB-D (TP3D).

Cada uno de los escenarios (Fig. 5) se plantea para recabar información con diferente finalidad: (a) Se utiliza para evaluar la correcta detección de una mesa y el impacto que tiene la contradicción entre planificador global y local. (b) Se utiliza para comprobar si el planificador consigue atravesar un pasillo estrecho formado por mesas. (c) Es un escenario genérico estándar donde se plantean

diferentes alternativas para alcanzar la meta pero todas conllevan un correcto uso de la tridimensionalidad.

La ejecución de cada prueba consiste en la navegación desde una pose fija en cada escenario (Pose A) hasta un conjunto de poses meta (Pose B). La pose A se toma como $A = [-2.56, 0, \pi/2]$, La pose B se extrae de la recta $B = [x, 4, 0]$ con $x \in [-3.5, -1]$, seleccionando 6 valores equidistantes. Estas pruebas se replican un total de 10 veces, dando como resultado un total de 180 simulaciones.

Tabla 1: Resultados de navegación. B: Bloqueo, E: Éxito, C: Choque (Fig. 5)

	DWA	TR	TP	TP3D
a	B	E	C	E
b	B	B	C	E
c	B	B	C	E

La Tabla 1 muestra los resultados obtenidos, donde se puede apreciar como, a excepción del planificador TR en el escenario (a), para los tres primeros algoritmos el robot no superó los obstáculos (ya sea por colisionar con algún elemento del entorno o por caer en un mínimo local durante la generación de trayectorias, quedando este bloqueado). Tan solo el planificador TP con información 3D fue capaz de gestionar exitosamente este tipo de situación. Para este último, la Tabla 2 muestra los valores de velocidad lineal y angular medias, tiempo de navegación y errores en la pose final (posición y *yaw* por separado). Se observa que la desviación de los resultados en velocidad es amplia en relación a su media, concordando con los resultados de Rodríguez *et al.* [20]. El tiempo de navegación medio es cercano al obtenido en una habitación sin obstáculos, aproximadamente de 18 segundos, aunque con una desviación amplia, puesto que a menudo se eligen caminos diferentes dependiendo de la pose final y de los obstáculos, que pueden ser subóptimos. Afinar los parámetros de generación de trayectorias es determinante para mejorar este resultado. Mencionar que el campo de visión usualmente restringido de las cámaras RGBD y el número limitado de ellas, dificultan encontrar una configuración de las mismas libre de ángulos muertos. Se observó un 2% de colisiones para cada caso debido a esto, todas ellas causadas por esquinas muy cercanas al cuello del robot.

4.2 Experimentos con Robot Real

Para realizar un recorrido en una habitación poblada con obstáculos en el mundo real, se ha utilizado la plataforma robótica motorizada Giraff [17].Toda la funcionalidad que antes ofrecía el simulador Stage es ahora provista por dicho robot valiéndose de drivers OpenNI en forma de nodos ROS. Como equipo de sensores fueron elegidos tres ligeras cámaras RGB-D ASUS Xtion Pro. La primera situada en la cabeza del robot, la segunda situada a su izquierda y la otra sobre su derecha. Las tres con una leve inclinación para visualizar el suelo. Además, se equipó al robot con un láser Hokuyo sobre su base.

Tabla 2: Observaciones en las replicaciones de TP-Space en 3 dimensiones. V_m (m/s): Velocidad lin. media, W_m (rad/s): Velocidad ang. media. T (s): Media de tiempo de navegación. P_e (m): Error medio de posición final. Θ_e (rad): Error medio de orientación final.

	(a)	(b)	(c)
V_m	0.20 ± 0.13	0.19 ± 0.14	0.20 ± 0.14
W_m	0.22 ± 0.21	0.25 ± 0.24	0.27 ± 0.25
T	20.45 ± 4.00	22.41 ± 3.68	23.25 ± 12.66
P_e	0.19 ± 0.01	0.19 ± 0.01	0.19 ± 0.01
Θ_e	0.16 ± 0.02	0.16 ± 0.02	0.16 ± 0.01

A pesar de contar con una representación estática del escenario de navegación, la necesidad de localizarse dentro de este mapa conocido conlleva el uso de odometría. Giraff-Plus posee *encoders* en sus ruedas, sin embargo, debido al estado actual de la plataforma móvil, resulta más efectivo encontrar otras vías para computar su odometría. La elegida, para este caso, se basa en el uso de información sensorial láser recibida del escenario. [14, 13] Para conseguir una localización más robusta, se aplica además AMCL. [9]

En cuanto a la planificación reactiva, sucesivos experimentos mostraron que, cuando se eligen puntos intermedios de la ruta global cercanos al robot, una velocidad lineal máxima adecuada para el uso de PTGs ronda los 0.3 m/s.

El robot es comandado recorrer una habitación sorteando obstáculos tridimensionales, cruzar una puerta y alinearse con la pose final. Los obstáculos utilizados son una mesa de cuatro patas (Ims. 1-2) y unas cajas de cartón sobre una plataforma más estrecha (Ims. 2-3). Nótese cómo los obstáculos se sitúan tratando de obstaculizar la puerta. En (a) se representa un momento previo a (1), en él, el robot tuvo que girar sobre sí mismo para alejarse de la mesa. En (b) se representa un momento anterior a (3), en la nube de puntos aparece abajo a la izquierda la torre de cajas y puede observarse cómo el giro a la derecha es elegido porque el espacio libre parece mayor en esa dirección. (Fig. 6)

En la Figura 7, se muestra la odometría com-

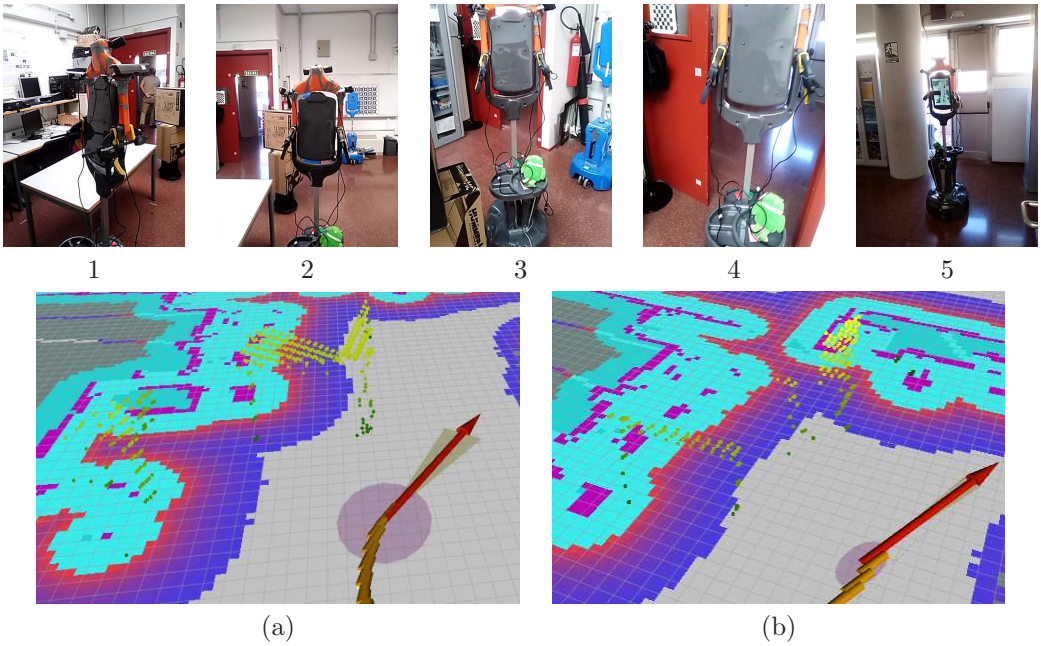


Figura 6: (1-5) Fotografías del recorrido realizado por el robot Giraff-Plus en una habitación. (a-b) Nubes de puntos visualizadas en los puntos 1 y 2 (aprox.). Coloreado según componente Z. (Amarillo: menor, Verde: mayor)

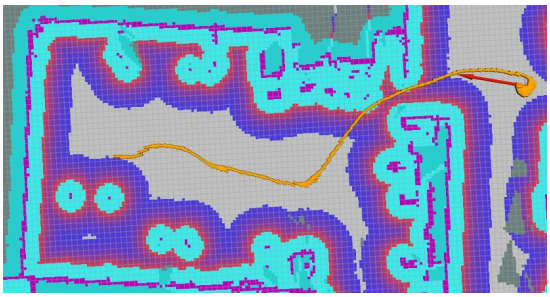


Figura 7: Recorrido del robot Giraff-Plus durante la prueba documentada. Representación del escenario: Costmap global. Ocre: Odometría.

putada por el robot para cada instante del recorrido. Dada la presencia de obstáculos, puede observarse un recorrido sinuoso que se traduce en mayor tiempo hasta alcanzar la meta. (52.54 s)

Agradecimientos

Este trabajo se ha desarrollado en el marco del proyecto WISER (DPI2017-84827-R), financiado por el Ministerio de Ciencia e Innovación contando con fondos del Fondo Europeo de Desarrollo Regional (FEDER), y por la beca de colaboración del Ministerio de Educación, Cultura y Deporte (UMA-998142).

English summary

Integration of a Parametric Trajectory Planner in the ROS Robotic Architecture

Abstract. *This paper presents a new reactive movement planner integration within ROS robotic architecture. Said planner, originally developed as a part of MRPT library and based on Trajectory Parameter Space (TP-Space), broadens current robotic platforms' navigation possibilities through both circular and non-circular trajectories generation, as well as an option to contemplate the robot's 3D geometry (not being restricted to prismatic forms alone). This integration has been carried out guaranteeing compatibility with the most popular autonomous navigation stack in the ROS platform, fitting reactive operation to the standard ROS communication interface, while being compatible with existing global planners also. As validation, multiple navigation tests have been performed in obstacle-populated indoor environments (both simulated and real) by a robotic platform equipped with various RGB-D cameras and a laser scanner.*

Keywords: Mobile Robotics, MRPT, TP- Space, ROS, Autonomous Navigation, Obstacle Avoidance, Trajectory Generation, Reactive Planner.

Referencias

[1] George A Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.

[2] Keshav Bimbraw. Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario

and the Expected Future of Autonomous Vehicle Technology. In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 191–198, 2015.

[3] Jose-Luis Blanco, Javier González, and Juan-Antonio Fernández-Madrigal. Extending obstacle avoidance methods through multiple parameter-space transformations. *Autonomous Robots*, 24(1):29–48, Jan 2008.

[4] José-Luis Blanco, Javier Gonzalez-Jimenez, and Juan-Antonio Fernández-Madrigal. *Foundations of Parameterized Trajectories-based Space Transformations for Obstacle Avoidance*, chapter 2. Mobile Robots Motion Planning: New Challenges. I-Tech Education Publishing, 2008.

[5] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, March 1986.

[6] Silvia Coradeschi, Amedeo Cesta, G Cortellessa, Luca Coraci, Cipriano Galindo, Javier González-Jiménez, Lars Karlsson, A Forsberg, Susanne Frennert, F Furfari, Amy Loutfi, Andrea Orlandini, Filippo Palumbo, Federico Pecora, Stephen Von Rump, A Štimec, Jonas Ullberg, and B Ötslund. *GiraffPlus: A System for Monitoring Activities and Physiological Parameters and Promoting Social Interaction for Elderly*, volume 300, pages 261–271. 07 2014.

[7] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, June 2006.

[8] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, March 1997.

[9] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. pages 343–349, 01 1999.

[10] Brian P. Gerkey and Kurt Konolige. Planning and control in unstructured terrain. In *In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, 2008.

[11] J.E. Hopcroft, J.T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; pspace- hardness of the "warehouseman's problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.

[12] Mariano Jaimez, José-Luis Blanco, and Javier Gonzalez-Jimenez. Efficient reactive navigation with exact collision determination for 3d robot shapes. *International Journal of Advanced Robotic Systems*, 12(63), 2015.

[13] Mariano Jaimez, Javier Monroy, and Javier Gonzalez-Jimenez. Planar odometry from a radial laser scanner. a range flow-based approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4479–4485, jun 2016.

[14] Mariano Jaimez, Javier Monroy, Manuel Lopez-Antequera, and Javier Gonzalez-Jimenez. Robust planar odometry based on symmetric range flow and multi-scan alignment. *IEEE Transactions on Robotics*, pages 1623–1635, 2018.

[15] Nicolas Limpert, Stefan Schiffer, and Alexander Ferrein. A local planner for ackermann-driven vehicles in ros sbpl. 10 2015.

[16] Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, Feb 1983.

[17] Matteo Luperto, Javier Monroy, Francisco-Angel Moreno, J. R. Ruiz-Sarmiento, Nicola Basilico, Javier Gonzalez-Jimenez, and N. Alberto Borghese. A multi-actor framework centered around an assististe mobile robot for elderly people living alone. In *IEEE International Conference on Intelligent Robots - Workshop on Robots for Assisted Living (IROS)*, 2018.

[18] L. Matthies and A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 727–733 vol.2, April 1988.

[19] Francesco Nex and Fabio Remondino. Uav for 3d mapping applications: A review. *Applied Geomatics*, 6, 03 2014.

[20] Enrique Rodríguez, José-Luis Blanco, Jose Luis Torres, Jose Carlos Moreno, Antonio Gimenez, and Jose Luis Guzmán. A ros reactive navigation system for ground vehicles based on tp-space transformations. In *XXXVII Jornadas de Automatica*, 2016.

[21] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

[22] David V. Lu and William Smart. Towards more efficient navigation for robots and humans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1707–1713, 11 2013.

[23] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2:189–208, 2008.

[24] Iris FA Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170(3):677–709, 2006.