Mejora del Comportamiento Proxémico de un Robot Autónomo mediante Motores de Inteligencia Artificial Desarrollados para Plataformas de Videojuegos

David Fernández, Javier Monroy, Javier Gonzalez-Jimenez
Departamento de Ingeniería de Sistemas y Automática, Instituto de Investigación Biomédica de Málaga, Universidad de Málaga, Campus de Teatinos, 29071, Málaga.
davfercha@uma.es, jgmonroy@uma.es, javiergonzalez@uma.es

Resumen

Este artículo presenta una arquitectura software que permite mejorar las capacidades sociales de robots móviles mediante el uso de motores de inteligencia artificial propios de las plataformas de desarrollo de videojuegos. La arquitectura propuesta se basa en fusionar el ampliamente utilizado Robotic Operating System (ROS) con los avanzados motores de inteligencia artificial de la plataforma de videojuegos Unity 3D. Esta fusión permite integrar comportamientos de alto nivel en la interacción hombre-robot, con los avanzados sistemas de gestión de la incertidumbre propios de arquitecturas puramente robóticas. Como caso práctico presentamos la planificación de trayectorias de un robot móvil en un entorno con presencia humana, detallando tres situaciones típicas de la interacción hombre-robot: esquivar a un humano que afecta en su trayectoria, aproximarse a una persona para establecer un diálogo, y evitar cruzar entre dos personas que están dialogando. Para cada situación, analizamos el entorno desde un punto de vista proxémico, y comparamos la trayectoria generada por el navegador de Unity 3D usando nuestra propuesta con una puramente robótica.

Palabras clave: Habilidades Sociales, Videojuegos, Unity 3D, HRI, Robot

1 Introducción

El envejecimiento de la población en los países desarrollados se ha convertido en un problema importante con una creciente atención por parte de la comunidad científica. Más allá de su dimensión estrictamente demográfica, el envejecimiento preocupa por sus efectos políticos, económicos y sociales [13]. En relación con este último, diversas iniciativas han surgido en los últimos años proponiendo el uso de plataformas robóticas para llevar a cabo actividades de carácter social, por ejemplo, como guías turísticos, asesores personales, o robots educativos.

No obstante, en la actualidad, la presencia real de robots en entornos humanos está aún muy limitada, siendo una de las principales causas la altamente compleja interacción entre el hombre y el robot (HRI, de sus siglas en inglés) [3, 7]. Conseguir que un robot tenga un comportamiento que le permita ser aceptado en entornos sociales es un problema muy complejo y que abarca multitud de facetas. Una de estas es la proxémica [4] o estudio de la organización del espacio en la comunicación entre las personas y los objetos. Para el caso de interacción con robots móviles, diversos trabajos han sido propuestos [12, 11], donde la proxémica juega un papel fundamental a la hora de controlar los movimientos del robot. No obstante, aún existen grandes posibilidades de investigación en este ambito, especialmente atendiendo a la gran variedad de situaciones en las que un robot puede interactuar con un humano [6, 9].

Una de las áreas que más ha avanzado en esta interacción maquina-humano en los últimos años es la relacionada con la industria de los videojuegos. Gracias al potente motor económico que la sustenta, gran variedad de algoritmos y motores de inteligencia artificial (IA) enfocados a la interacción con el humano han sido desarrollados y están disponibles de forma libre, especialmente a través de plataformas de desarrollo de videojuegos. En este campo, los motores de IA se utilizan principalmente para modelar el comportamiento de personajes no jugadores (NPC, de sus siglas en ingés) de forma que sus acciones y movimientos se asemejen a las de un humano cuando interactúan con el jugador, por ejemplo, respetando las distancias proxémicas durante un dialogo [10].

En este trabajo proponemos explotar el activo y avanzado desarrollo de los motores de IA de las plataformas de creación de videojuegos, aplicándolos al campo de la robótica móvil, especialmente en el contexto HRI. Más concretamente, proponemos una arquitectura software capaz de fusionar la potencia de los motores de IA de los entornos de videojuegos, con la robustez de los sistemas robóticos a la hora de gestionar la incertidumbre al interactuar en el entorno. Para validar nuestra propuesta, presentamos un caso práctico: la planificación de la trayectoria de un

robot atendiendo a las distancias proxémicas y a los humanos presentes en el entorno, y su posterior navegación.

2 Arquitectura software

Esta sección describe la arquitectura propuesta, presentando de forma genérica los diferentes componentes y su interconexión. Posteriormente, para el caso de uso propuesto, se detalla la estructura del planificador de trayectorias empleado y cómo este es extendido para integrar restricciones de tipo proxémico.

La Fig. 1 muestra la arquitectura global propuesta en este articulo, donde se aprecia que como arquitectura robótica se ha usado ROS[1] (Robot Operating System) por ser la más utilizada en la actualidad. ROS es un framework de código abierto y flexible que permite el desarrollo de software robótico. Consiste en una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robótico complejo y robusto. Por otro lado, como software de desarrollo de videojuegos, se ha escogido Unity 3D [2], un motor gráfico multiplataforma que permite la creación de juegos en 2D y 3D. Entre los diversos frameworks actuales de desarrollo de videojuegos se ha elegido éste por su versatilidad y por la amplia variedad de paquetes software ya existentes e integrables, los cuales permiten agilizan el desarrollo de proyectos explotando alternativas de forma rápida.

Dado que ambas plataformas sowftware están diseñadas para funcionar sobre sistemas operativos distintos (ROS está solo disponible para sistemas operativos Linux, mientras que Unity 3D está basado en Windows), se hace necesario establecer un canal de comunicación bidireccional e independiente del sistema operativo anfitrión. En este trabajo proponemos emplear web-sockets, una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está tecnología, ampliamente usada por los navegadores y servidores web, así como en otros trabajos donde el sistema operativo ROS es conectado con plataformas de realidad virtual [5, 2], nos permite eliminar la dependencia del sistema operativo. No obstante, dado que los mensajes transmitidos pueden verse influenciados por las condiciones de la red de comunicaciones (pudiendo generar latencias no deseadas que influyan en el comportamiento del robot), se recomienda trabjar en conexión local y verificar que la frecuencia de transmisión de datos entre ambos sistemas es

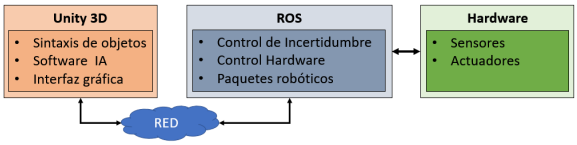

Figura 1: Diagrama representativo de la arquitectura global propuesta.

adecuada para la aplicación a ejecutar.

Concretamente, para realizar la conexión por web-sockets en ROS se ha utilizando *rosbridge_suite*[3] que es un paquete instalable que permite el realizar comunicaciones web por socket mediante mensajes JSON. Para poder comunicarse correctamente, ha sido necesario realizar una adaptación de la estructura de los mensajes que usa ROS para que Unity 3D sea capaz de procesarlos, para ello se ha usado la librería *ROSBridgeLib*[4] la cual posee las clases de los mensajes básicos de ROS.

Finalmente, se ha de comentar que a diferencia de Unity 3D, la plataforma ROS se basa en una arquitectura de comunicación P2P en la que diferentes "paquetes" intercambian información a través de una jerarquía de mensajes denominados "topics". Esta diferencia requiere extender la librería *ROSBridgeLib* para poder establecer una comunicación completa entre ambas plataformas.

2.1 Estructura del Planificador de Trayectorias

De entre las diversas tareas en las que los motores de IA de las plataformas de videojuegos pueden ser usadas para mejorar la interacción HRI, en esta sección nos centramos en la descripción de un planificador de trayectorias que permita al robot desplazarse e interactuar en un entorno con presencia humana.

Nuestra arquitectura está basada en ROS, donde los planificadores de trayectorias se dividen en dos fases o etapas: en primer lugar, un planificador global, el cual se encarga de generar caminos óptimos entre la posición actual del robot y el objetivo al que se desea llegar atendiendo a los obstáculos estáticos del entorno (generalmente en forma de mapa de ocupación), y a las restricciones de movimiento del robot. En segundo lugar, un planificador local que genera las consignas de velocidad, para mediante pequeños desplazamientos, seguir el trazado obtenido por el planificador global. El planificador local integra el comportamiento reactivo del robot puesto que su misión es completar la trayectoria global, evitando

[1] `http://www.ros.org/`
[2] `https://unity3d.com/es`
[3] `http://wiki.ros.org/rosbridge_suite`
[4] `https://github.com/michaeljenkin/unityros`

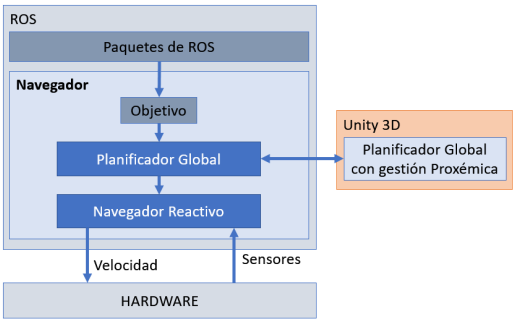

Figura 2: Diagrama representativo de la arquitectura propuesta para el planificador global.

los obstáculos que no se encuentren en el mapa de navegación, mayormente, obstáculos dinámicos (p.e. personas u otros robots).

Ambos planificadores se encuentran definidos en el paquete *move_base*[5], ofreciendo una API para la configuración del algoritmo de generación de trayectorias a emplear en cada fase, así como los parámetros que determinan su comportamiento. Una de las ventajas de este paquete ROS es la posibilidad de definir nuevos planificadores personalizados a través de un sistema modular basado en plugins. Concretamente, explotamos esta propiedad para implementar un nuevo planificador que delegue en Unity 3D la generación de la trayectoria global, para luego ser procesada por el planificador local de ROS (véase Fig. 2). Este tipo de configuración permite explotar la robustez de ROS con respecto a la gestión de incertidumbre y el uso de sistemas avanzados de control reactivo, mientras que se dota de un comportamiento más "social" desde un punto de vista HRI.

Para la obtención del camino global en Unity 3D, se ha utilizado un paquete de generación de trayectorias típico de los entornos de videojuegos: *Pathfinding_Project*[6]. Este paquete permite escanear el entorno virtual previamente generado para obtener el espacio transitable a través del cual, mediante algoritmos como el A*, se obtiene el camino a seguir formado por un conjunto de puntos de interés. Un ejemplo de su uso puede verse en la Fig. 3, en la que se comprueba que en caso de trabajar en un entorno sin presencia humana, el resultado obtenido es muy similar al de un planificador típico en robótica.

No obstante, el objetivo es hacer uso de las características dinámicas y sociales integradas en este tipo de plataformas para trabajar en entornos dinámicos y con presencia humana. Concretamente, este planificador contempla una propiedad

[5] `http://wiki.ros.org/move_base`
[6] `https://arongranberg.com/astar/`

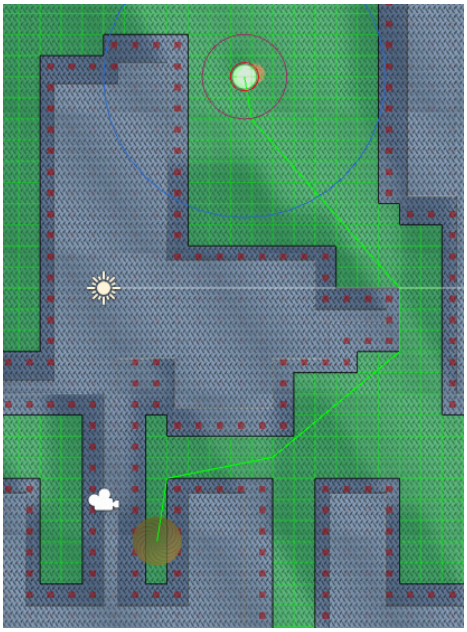

Figura 3: Ejemplo de uso de Unity 3D para la generación de caminos en un entorno sin presencia humana. En este caso el resultado obtenido es muy similar al de un planificador típicamente robótico.

denominada obstáculo dinámico que sirve para especificar que el objeto que la posea puede cambiar de posición o forma durante el tiempo de ejecución. Explotando esta propiedad, diseñamos las diferentes áreas proxémicas de las que disponemos los humanos [11, 14], las cuales vienen dadas por diversos factores como por ejemplo la actividad, estado de ánimo o el tamaño de la persona (Véase Fig. 4). Atendiendo a esta nueva propiedad, la planificación de la trayectoria cambia radicalmente, tal y como ilustramos en la Sección 3 con tres casos comunes de HRI.

2.2 Plugin del Planificador Global y Conexión entre Sistemas

El uso del planificador de trayectorias y el modelado de la proxémica incluidos en Unity 3D vienen condicionados a disponer en dicha plataforma de conocimiento del entorno, de la posición actual del robot, del destino a navegar, y de la presencia de humanos, factores que son generados y/o gestionados por ROS. La Fig. 5 muestra el diagrama de conexiones implementado para este caso de uso.

- **Mapa de Ocupación**: Para la generación del entorno tridimensional en Unity 3D, se realiza una suscripción al topic: */map* de ROS que posee el mapa de ocupación de obstáculos. Este consiste en un mapa de rejilla en la cual cada celda representa la

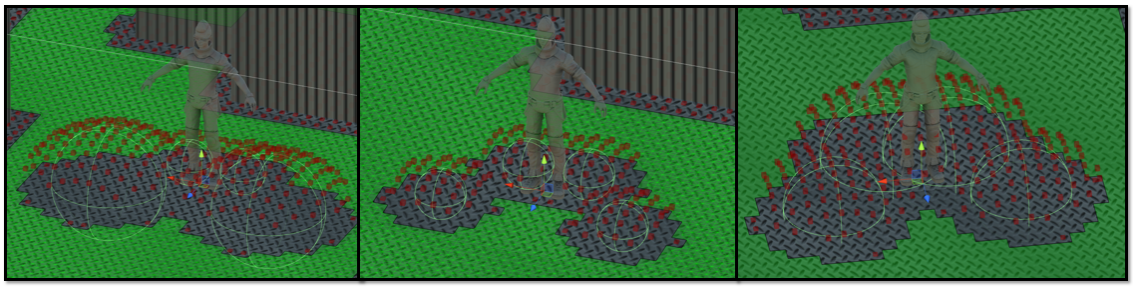

Figura 4: Diferentes áreas proxémicas generadas entorno a una persona mediante Unity 3D.

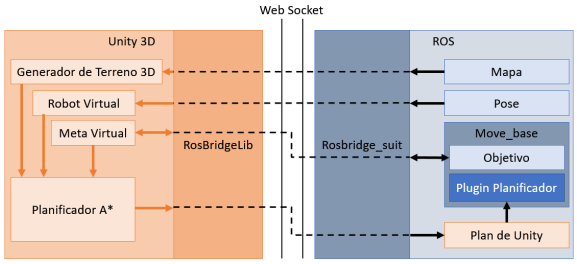

Figura 5: Diagrama representativo del tránsito de información entre Unity 3D y ROS para el caso de la planificación de trayectorias atendiendo a la proxémica.

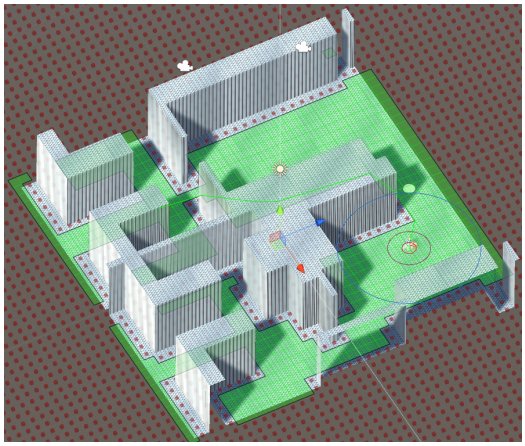

Figura 6: Reconstrucción 3D en Unity del mapa de ocupación de obstáculos utilizado en ROS.

probabilidad de ocupación. Esta información probabilística es binarizada en Unity 3D para generar un mapa del entorno 3D (véase Fig. 6), asignando una celda como libre si $p < 0,5$ y viceversa. Es importante mencionar que a diferencia de los sistemas robóticos, Unity no considera probabilidades ni incertidumbre en los datos, aspecto que solventamos haciendo uso del planificador local de ROS. .

- **Posición del Robot**: Para poder acceder a esta pose y llevar a cabo la planificación de la trayectoria, Unity se subscribe a través de la arquitectura software propuesta al topic */amcl_pose*. Este contiene la creencia sobre la pose del robot en el entorno. Una vez más, esta información es probabilística, representando la pose más probable de entre una serie de candidatos, lo cual implica que esta puede cambiar (incluso de forma drástica) a lo largo del tiempo. Este topic es comúnmente actualizado por módulos ROS de localización probabilística, como es el caso del filtro de Kalaman o el filtro de particulas.

- **Destino de Navegación**: Se prevé la posibilidad de que el destino al que el robot debe ir, pueda venir impuesto desde cualquiera de las dos plataformas, de esta manera, se brinda la opción de mover al robot a los algoritmos diseñados tanto en ROS como en Unity 3D. Para lograr esto, se utiliza el topic */move_base/goal* de manera bidireccional permitiendo a ambos entornos escribir y leer la posición de destino deseada, la cual se vincula con la posición de un objeto virtual, de manera que si se mueve el objeto se publica la nueva posición en ROS y viceversa.

- **Presencia de Humanos**: En este trabajo asumimos que se dispone de un modulo ROS para la detección y localización de las personas en el entorno. A partir de dicha información, Unity 3D asigna el rol de un jugador y por tanto, les añade el área proxémica previamente definida.

Con estos datos, Unity 3D genera la trayectoria global correspondientes, y la publica en el topic de ROS */unity_data/global_plan* para ser gestionado posteriormente por el planificador local de ROS (véase Fig. 2). Dado que este planificador local dispone de más información para la navegación (p.e. detección de obstáculos no contenidos en el mapa de ocupación) es muy probable que este se desvíe del plan global inicialmente trazado. Para mantener una consistencia en el sistema, Unity re-

calcula el camino global periódicamente y en caso de que se obtenga una nueva ruta diferente a la inicial, esta se reenvía a ROS.

3 Caso Práctico: Planificación de Trayectorias

Para validar la arquitectura propuesta y analizar las ventajas que los motores de IA contenidos en Unity 3D ofrecen a la hora de modelar la interacción entre robots y humanos, en esta sección se describen varios experimentos en los que se comparan las trayectorias generadas por el navegador que utiliza ROS de manera predefinida y el navegador de Utniy 3D. Más concretamente, presentamos tres escenarios comunes en la interacción hombre-robot: esquivar a un humano que se encuentra en el área de actuación del robot, aproximarse a una persona para establecer un diálogo, y evitar cruzar entre dos personas que están dialogando.

En todos los casos, modelamos el área proxémica de las personas involucradas mediante la consideración de cuatro esferas (véase Fig. 4), las cuales han sido parametrizadas atendiendo a los estudios presentados en [9]. Estás áreas representan zonas que el robot no debería invadir para lograr un comportamiento natural y social desde un punto de vista HRI.

3.1 Caso A: Navegación evitando una presencia humana

En el primer caso expuesto, el robot tiene que navegar de un punto a otro del mapa evitando la presencia humana en su trayectoria. Atendiendo al uso de la proxémica, el robot no debería invadir el espacio personal de la persona, por lo que tiene que considerar alejarse de la persona lo suficiente como para evitar hacerlo.

En la Fig. 7 se muestran los resultados comparados del planificador por defecto que se utiliza en ROS y el planificador de Unity 3D utilizando la arquitectura propuesta en este artículo. En azul se observa el camino que sigue el robot cuando se utiliza el navegador de ROS, el cual considera la persona como un obstáculo más del mapa. Como consecuencia, el robot invade el área proxémica de la persona. Igualmente, en verde se observa el camino generado por el planificador global de Unity 3D, el cual provee un camino que no invade el espacio proxémico generado para la persona, de manera que al usar este camino en un robot mediante la arquitectura propuesta, se consigue un desplazamiento más natural desde el punto de vista de la persona.

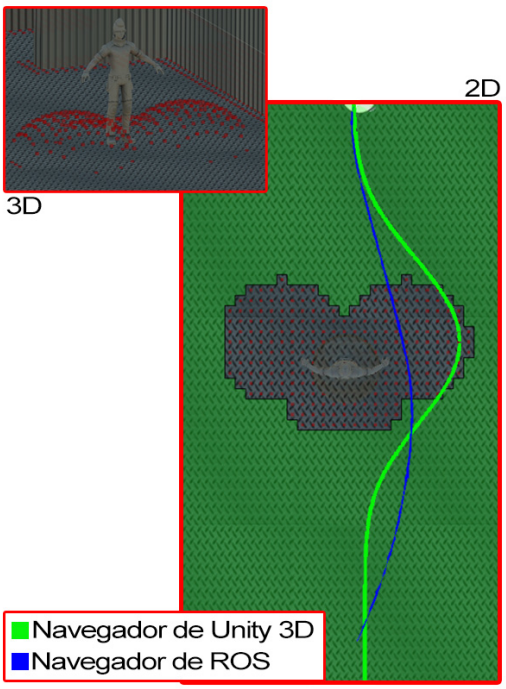

Figura 7: Comparativa de trayectorias cuando el robot se cruza con una persona en su paso.

3.2 Caso B: Interacción Directa con Humano

En este caso, se comprueba como es el acercamiento del robot cuando tiene que ir a interactuar con una persona, por ejemplo para hablar con ella, o entregarle algo. Al igual que en el caso anterior, el comportamiento deseado desde un punto de vista proxémico resulta de realizar un acercamiento sin invasión del área personal. Además, el robot debe ir hacia la persona desde una posición frontal a esta.

El resultado obtenido de la comparación de ambos planificadores se observa en la Fig. 8, donde mediante el uso del navegador por defecto de ROS, el robot invade el área proxémica de la persona rodeandola y acercándose a ella desde el lateral al igual que si esta fuese un obstáculo más del entorno. En cambio, el navegador global de Unity 3D obtiene un camino que evita la invasión del área designada como espacio proxémico, el cual permite que el acercamiento ha la persona se realice encarándola.

3.3 Caso C: Navegación evitando espacios proxémicos de grupos de personas

Por último, se comprueba el comportamiento del robot cuando este debe ir de un punto a otro de la sala pero a diferencia del caso A, se han colocado dos personas encaradas que simulan tener una conversación. Puesto que cada persona tiene su propia distancia personal, al estar cerca una de

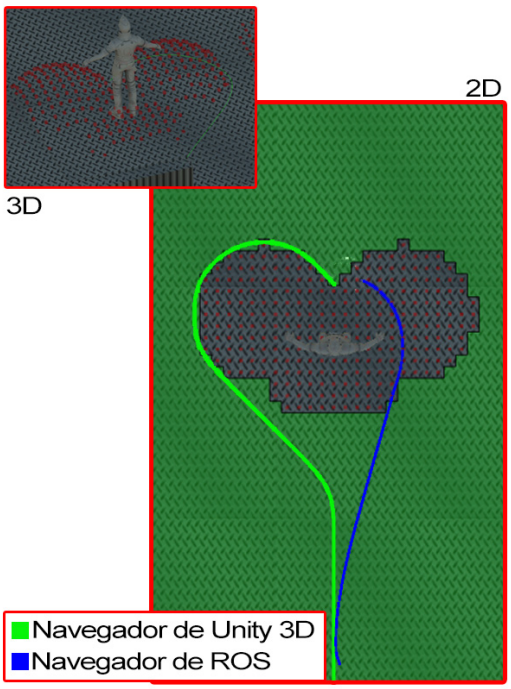

Figura 8: Comparativa de trayectorias cuando el robot va hacia una persona.

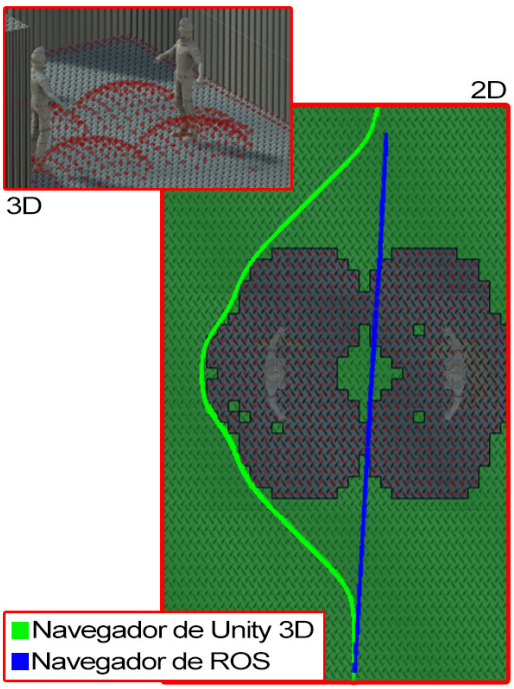

Figura 9: Comparativa de trayectorias cuando el robot se encuentra con un grupo de personas.

otra, se combinan generando un área mayor [8]. Cuando el robot navegue debe evitar pasar entre las personas ya que además de resultar poco natural, interrumpiría a las personas sin necesitar su atención.

En la Fig. 9 se muestran los resultados obtenidos de realizar este caso. Se muestra como el camino generado por el navegador de ROS, cruza entre las dos personas por lo que resultaría un comportamiento poco adecuado en base a la proxémica de las personas. Por el contrario, el camino generado por el navegador de Unity 3D al considerar las áreas asignadas a las dos personas genera un camino que rodea a ambas sin interrumpirlas.

Para concluir con esta sección, cabe destacar que en todos los casos expuestos, se puede comprobar que el navegador de Unity 3D, al considerar la persona como un obstáculo dinámico, pospone la reacción a dicho obstáculo hasta que la invasión del área proxémica es inminente. Esta decisión consigue que si la persona se ha desplazado durante el desplazamiento del robot, para cuando el camino se actualice mantendrá la trayectoria óptima. En comparación, el navegador predefinido en ROS prevé que tiene que desviarse por la existencia de un objeto estático, aunque quizás no sea necesario cuando se aproxime y por tanto se habrá desviado del camino óptimo.

4 Conclusiones

En base a los resultados obtenidos, se puede afirmar que Unity 3D se puede utilizar como un elemento de ayuda para el desarrollo de las HRI brindando un amplio conjunto de paquetes libres que pueden ser utilizados para mejorar el comportamiento de los robots. En este artículo se ha utilizado uno de estos paquetes para el diseño de un navegador que contempla el área proxémica de las personas. El diseño de estas áreas pueden realizarse en ROS mediante mapas de coste pero resulta más tedioso y conlleva más tiempo de desarrollo para realizar ensayos.

Siguiendo la línea de contribución de este artículo, podría implementarse al navegador diferentes reglas que modifiquen el área proxémica de la persona en función del contexto en que se encuentre. Esto supondría que el robot tendría comportamientos diferentes en función de la hora del día, o de las situaciones de las personas de su entorno generando como resultado un comportamiento mucho más natural y por tanto contribuyendo en la mejora de las HRI.

Aunque en este artículo se ha explotado Unity 3D para la navegación del robot, hay disponibles multitud de paquetes que pueden contribuir en otras áreas de la robótica, por ejemplo aportando una interfaz intuitiva para el usuario, mediante el uso de la realidad virtual, o incluso usando la inteligencia artificial que aportan los videojuegos

para generar el comportamiento del robot. Las ventajas de Unity 3D ya han sido usadas por otros artículos previamente como por ejemplo [1] donde explotan el motor de animaciones para generar movimientos en un robot.

En conclusión, existen numerosos frentes de interés en donde los entornos de desarrollo de videojuegos pueden complementar la programación de los robots, agilizando o añadiendo soporte a nuevas áreas de desarrollo que en conjunto mejorarán las capacidades sociales de los robots.

Agradecimientos

Este trabajo ha sido parcialmente financiado a través del plan nacional, gobierno de España (proyecto DPI2014-55826-R), y por la Unión Europea, dentro del programa Horizonte 2020-ICT (proyecto 732158 - MoveCare).

Referencias

[1] C. Bartneck, M. Soucy, K. Fleuret, and E. B. Sandoval. The robot engine; making the unity 3d game engine work for hri. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 431–437, Aug 2015.

[2] R. Codd-Downey, P. M. Forooshani, A. Speers, H. Wang, and M. Jenkin. From ros to unity: Leveraging robot and virtual environment middleware for immersive teleoperation. In *2014 IEEE International Conference on Information and Automation (ICIA)*, pages 932–936, July 2014.

[3] M.J. Matarić D. Feil-Seifer. Human robot interaction. *Encyclopedia of Complexity and Systems Science*, pages 4643–4659, 2009.

[4] Edward T Hall, Ray L Birdwhistell, Bernhard Bock, Paul Bohannan, A Richard Diebold Jr, Marshall Durbin, Munro S Edmonson, JL Fischer, Dell Hymes, Solon T Kimball, et al. Proxemics [and comments and replies]. *Current anthropology*, 9(2/3):83–108, 1968.

[5] Yuchao Hu and Wei Meng. Rosunitysim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning. *SIMULATION*, 92(10):931–944, 2016.

[6] B. Jensen, N. Tomatis, L. Mayor, A. Drygajlo, and R. Siegwart. Robots meet humans-interaction in public spaces. *IEEE Transactions on Industrial Electronics*, 52(6):1530–1546, Dec 2005.

[7] K. Kosuge and Y. Hirata. Human-robot interaction. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 8–11, Aug 2004.

[8] Thibault Kruse, Patrizia Basili, Stefan Glasauer, and Alexandra Kirsch. Legible robot navigation in the proximity of moving humans. In *Advanced Robotics and its Social Impacts (ARSO), 2012 IEEE Workshop on*, pages 83–88. IEEE, 2012.

[9] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[10] J. E. Laird. Using a computer game to develop advanced ai. *Computer*, 34(7):70–75, Jul 2001.

[11] Ross Mead, Amin Atrash, and Maja J Matarić. Automated proxemic feature extraction and behavior recognition: Applications in human-robot interaction. *International Journal of Social Robotics*, 5(3):367–378, 2013.

[12] Jonathan Mumm and Bilge Mutlu. Human-robot proxemics: physical and psychological distancing in human-robot interaction. pages 331–338, 2011.

[13] Begoña San Miguel and Mª Jose Gonzalez. El envejecimiento de la población española y sus consecuencias sociales. *Cuadernos de trabajo social*, (9):19–45, 2001.

[14] Emrah Akin Sisbot, Luis F Marin-Urias, Rachid Alami, and Thierry Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.