https://doi.org/10.17979/spudc.9788497498081.0724

APRENDIZAJE DE COMPORTAMIENTOS DE NAVEGACIÓN EN PLANIFICADORES RRT*

Noé Pérez Higueras, Luis Merino
Universidad Pablo de Olavide, Crta. Utrera km 1, Sevilla, noeperez@upo.es

Fernando Caballero
Universidad de Sevilla, Crta. Camino de los Descubrimientos, s/n, Sevilla, fcaballero@us.es

Resumen

Este trabajo presenta un algoritmo para el aprendizaje de comportamientos de navegación a partir de demostraciones usando árboles de exploración aleatoria óptimos (RRT*) como planificador de caminos. El algoritmo de aprendizaje combina las técnicas de *Inverse Reinforcement Learning* (IRL) y RRT* para aprender los pesos de la función de coste a partir de trayectorias de demostración. Esta función de coste puede ser usada más tarde en el algoritmo RRT* permitiendo al robot reproducir el comportamiento deseado en distintos escenarios. El método ha sido probado primero en simulación y luego usando trayectorias reales de un robot en el laboratorio.

1. INTRODUCCIÓN

Cada día más, los robots móviles están participando de nuestra vida diaria y coexistiendo con nosotros. Como resultado, la planificación del movimientos de robots que comparten con humanos un entorno dinámico es un campo de investigación en auge. Los robots deben respetar las convenciones sociales, garantizar el confort de las personas a su alrededor y mantener la legibilidad, de manera que los humanos puedan entender las intenciones del robot [11]. Este problema, denominado navegación social, involucra diferentes áreas como la percepción, la creación de modelos cognitivos y la planificación de caminos.

El problema fue inicialmente abordado incluyendo costes y restricciones relativos al confort humano en los planificadores para obtener caminos socialmente aceptables [19, 7]. En estos casos, las restricciones eran pre-programadas. No obstante, los comportamientos sociales son difíciles de acotar y programar [3]. En muchos casos (como [8, 16]), estos costes están basados en la teoría Proxémica [5]. Sin embargo, como se muestra en [14], la Proxémica se centra en las personas mientras interactúan, y puede no ser adecuada para la navegación entre personas.

Por lo tanto, una aproximación más correcta puede ser el aprendizaje de estos comportamientos sociales a partir de datos. En particular, en este trabajo se busca la aplicación a robots de telepresencia [18]. Un robot de telepresencia es definido habitualmente como un *Skype sobre ruedas* en el que un operador se puede conectar al robot y navegar, mientras interactúa con otras personas a través de las interfaces de telecomunicación al mismo tiempo. El objetivo es incrementar la autonomía de tales robots, liberando al usuario de las tareas de navegación a bajo nivel. En este marco, se pueden obtener, de manera natural, datos de navegación del usuario, que serán considerados como ejemplos y empleados para el aprendizaje de comportamientos de navegación sociales.

De este modo, en este trabajo se plantea el desarrollo de una aproximación para el aprendizaje de comportamientos de navegación a partir de datos de los usuarios. En particular, se presenta un algoritmo de aprendizaje por demostración que emplea un planificador de caminos basado en muestreo (un RRT* [6] en concreto). Se hace uso de los conceptos de Aprendizaje por Refuerzo Inverso (IRL en sus siglas inglesas) y de los planificadores basados en muestreo para identificar la función de coste de dicho planificador que mejor se ajusta a las trayectorias de ejemplo. A diferencia de las aproximaciones clásicas de IRL basadas en Procesos de Decisión de Markov (MDPs en sus siglas inglesas), el método presentado es rápido y escala muy bien con el tamaño de estado; además de generalizar para ser empleado en diferentes escenarios. Una vez que la función de coste ha sido calculada, puede ser usada online en un RRT* para reproducir el comportamiento deseado en la planificación de caminos del robot.

La estructura de este artículo es la siguiente. Después de un resumen del trabajo relacionado, la siguiente sección describe el algoritmo de aprendizaje por demontración. Posteriormente, la sección 3 describe el problema particular de navegación social considerado. A continuación, la Sección 4 valida la aproximación mediante experimentos en simulación. Finalmente, se presentan las conclusiones y posible trabajo futuro.

1.1. Estado del arte

En los últimos años, muchas han sido las contribuciones en la aplicación de técnicas de aprendizaje a tareas de navegación social. En [20], se emplea aprendizaje supervisado para aprender modelos de predicción de movimiento de personas cuando el robot navega en entornos concurridos. Por otro lado, Luber et al. [14] emplean aprendizaje no supervisado para determinar prototipos de movimiento sociales de los que inferir costes sociales para la planificación de caminos. En [4], se emplea un modelo de navegación basado en fuerzas sociales cuyos parámetros son aprendidos mediante datos dados por los usuarios.

Una aproximación adicional es el aprendizaje por demostración [2], en el que un experto muestra al robot como debe navegar entre las personas. Esta aproximación es particularmente relevante para el caso de los robots de telepresencia. Y una manera de aprender por demostración es mediante IRL [1]. Las observaciones de un experto demostrando la tarea que se desea aprender se emplean para recuperar la función de coste (o recompensa) que el experto está intentando minimizar (maximizar). Posteriormente, dicha función se puede usar para obtener la política de comportamiento del robot.

El problema del IRL ha sido resuelto de diversas maneras. Un método probabilístico basado en máxima entropía se presenta en [21]. En [15], se afronta el problema del alto coste computacional empleando modelos bayesianos mixtos para dividir las observaciones y obtener un grupo de funciones de coste más simples. Desde otro punto de vista, los autores en [13] emplean Procesos Gausianos para representar el coste en lugar de como una combinación lineal de un conjunto de características.

En los modelos mencionados anteriormente, la técnica IRL hace uso de procesos de decisión de Markov (MDPs) para representar el problema matemáticamente. Sin embargo, codificar problemas generales con MDPs es computacionalmente complejo, y aunque muchos autores recurren a la discretización de estados, ésta puede resultar complicada. De este modo, algunos autores han propuesto el uso de diferentes planificadores. En [12], se emplean splines y técnicas de optimización para aprender comportamientos de conducción en coches autónomos empleando IRL. En [9] se emplean representaciones en el contínuo, donde el comportamiento de los peatones es aprendido de observaciones de trayectorias de peatones y del robot siendo tele-operado.

Los árboles aleatorios de exploración óptima (RRT*) [6] son una técnica ampliamente utilizada en planificación de caminos. Son flexibles y fácilmente adaptables a diferentes problemas; razonan implícitamente sobre colisiones con obstáculos con un coste computacional moderado; pueden explorar el espacio de configuraciones para obtener caminos óptimos basados en espacios de costes. Además, se pueden extender para razonar sobre la dinámica del robot.

En este artículo se presenta un algoritmo para el aprendizaje de comportamientos de navegación social a partir de demostraciones utilizando RRT* como planificador. Se pretende crear un nuevo algoritmo de aprendizaje que combine las técnicas de IRL y RRT* con el objetivo extraer los pesos de la función de coste adecuados de acuerdo a caminos de demostración. La función de coste puede ser usada luego en un RRT* regular a fin de reproducir el comportamiento deseado para el robot en diferentes escenarios.

2. APRENDIZAJE DE LA FUNCIÓN DE COSTE DEL PLANIFICADOR RRT*

RRT* [6] es una técnica para la planificación óptima de caminos, que considera que una función de coste está asociada a cada punto x del espacio de configuraciones. El RRT* busca obtener el camino ζ^* que minimice el coste total $c(\zeta)$. Para ello, muestrea aleatoriamente el espacio de configuraciones y crea un árbol hacia el destino. Los conjuntos de configuraciones discretas representan estos caminos $\zeta = \{x_1, x_2, \cdots, x_N\}$.

Sin pérdida de generalidad, se puede asumir que la función de coste en cada punto puede ser expresada como una combinación lineal de un conjunto de sub-funciones de coste, llamadas funciones de características $c(x_i) = \sum_j \omega_j f_j(x_i) = \omega^T f(x_i)$. El coste de un camino es, por tanto, la suma de los costes para todos los puntos en el camino. Concrétamente, el coste es la suma de los sub-costes correspondientes a moverse entre pares de puntos del camino. Normalmente, y en este trabajo, se emplea el coste medio entre los pares de puntos multiplicado por la distancia entre ellos

$$c(\zeta) = \sum_{i=1}^{N-1} c(x_i, x_{i+1}) = \sum_{i=1}^{N-1} \frac{c(x_i) + c(x_{i+1})}{2} \|x_{i+1} - x_i\| \quad (1)$$

De este modo, para unos pesos dados ω, el algoritmo devolverá caminos que intentan minimizar este coste.

Dado un conjunto de caminos de demostración $\mathcal{D} = \{\zeta_1, \zeta_2, \cdots, \zeta_D\}$, el problema de aprender de demostraciones, en esta configuración, significa determinar los pesos ω que dirigen al planificador RRT* a comportamientos similares a las demostraciones. De acuerdo a [1], el valor óptimo de ω es aquel que provoca que el valor esperado de las funciones de características de los caminos generados por el planificador sea igual al valor esperado de las funciones de características de los caminos de demostración:

$$\mathbb{E}(f(\zeta)) = \frac{1}{D}\sum_{i=1}^{D} f(\zeta_i) \tag{2}$$

Como se indica en [10], aplicando el principio de máxima entropía al problema de IRL [21] (el cual indica que, para todas las distribuciones potenciales de trayectorias que cumplen (2), la mejor es aquella con la entropía más alta, ya que no está sujeta a otros factores distintos los valores de esperanza de las funciones de características) se obtiene la siguiente densidad de probabilidad para los caminos de demostración del experto:

$$p(\zeta|\omega) = \frac{1}{Z(\omega)} e^{-\omega^T f(\zeta)} \tag{3}$$

donde $Z(\omega) = \int e^{-\omega^T f(\zeta)} d\zeta$ es una función de normalización que no depende de ζ.

El valor de ω que maximiza la verosimilitud (logarítmica) de los caminos demostrados con respecto a (3) se puede obtener considerando el gradiente respecto a ω:

$$\nabla\mathcal{L} = \frac{\partial\mathcal{L}(\mathcal{D}|\omega)}{\partial\omega} = \mathbb{E}(f(\zeta)) - \frac{1}{D}\sum_{i=1}^{D} f(\zeta_i) \tag{4}$$

Al emplear un planificador RRT* en lugar de un MDP, las restricciones a la hora de encontrar la densidad de probabilidad en los costes son distintas. El método RRT* es óptimo asintóticamente; esto es, para un tiempo de planificación infinito, el método converge casi seguro al camino óptimo (coste mínimo). Así que, dado suficiente tiempo, la distribución de costes se asemejaría a una distribución exponencial decreciente desde el coste óptimo. En el caso de emplear tiempos de planificación más realistas, se debería realizar una caracterización de la densidad de probabilidad de los costes de los caminos planificados por el RRT*. Sin embargo, se demostrará empiricamente que los resultados obtenidos aplicando el gradiente derivado de la distribución exponencial son buenos, aunque otras distribuciones de probabilidad podrían describir mejor la distribución de los costes de los caminos del RRT, y por lo tanto, obtener aún mejores resultados.

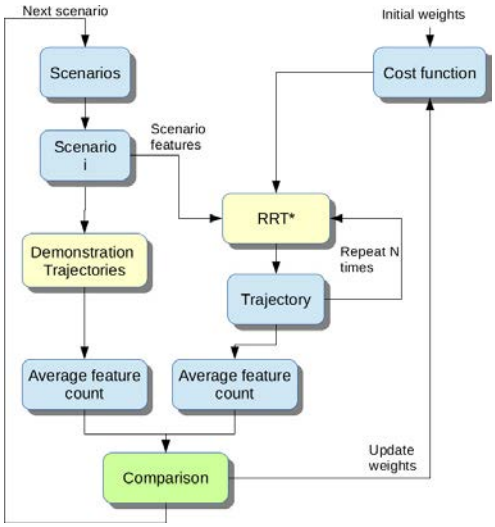


Figura 1: Esquema de aprendizaje general propuesto para ajustar los pesos de la función de coste de un planificador RRT*.

Algorithm 1: RRT*-IRL

Require: Caminos de demostración $\mathcal{D} = \{\zeta_1^1, \ldots, \zeta_D^S\}$ en $\mathcal{S}$ escenarios
Ensure: Pesos de la función de coste $\omega = [\omega_1, \ldots, \omega_J]^T$

1: $\overline{f}_{\mathcal{D}} \leftarrow CaractersticasMedia(\mathcal{D})$
2: $r \leftarrow inicializarAleatorio()$
3: $\omega \leftarrow [r, \ldots, r]^T$
4: **repeat**
5: **for each** $s \in \mathcal{S}$ **do**
6: **for** $repeticiones$ **do**
7: $\zeta_i \leftarrow ejecutarRRTstar(s, \omega)$
8: $f(\zeta_i) \leftarrow Caracteristicas(\zeta_i)$
9: **end for**
10: $\overline{f}^s_{RRT^*} \leftarrow \frac{(\sum_{i=1}^{repeticiones} f(\zeta_i))}{repeticiones}$
11: **end for**
12: $\overline{f}_{RRT*} \leftarrow (\sum_{i=1}^{S} \overline{f}^i_{RRT^*})/S$
13: $\nabla\mathcal{L} \leftarrow (\overline{f}_{RRT*} - \overline{f}_{\mathcal{D}})$
14: $\omega \leftarrow ActualizarPesos(\nabla\mathcal{L})$
15: **until** convergencia
16: **return** ω

La Figura 1 muestra el esquema de aprendizaje propuesto, y descrito en más detalle en el Algoritmo 1. Los caminos de demonstración de los que se quiere aprender son requeridos por el algoritmo, que, como resultado, devuelve los valores de los pesos para la función de coste del RRT* (1).

Primero, en la Línea 1 se obtiene la media de los valores de las funciones de características de los ca-

minos de demostración en los diferentes scenarios $\overline{f}_{\mathcal{D}} = \frac{1}{D}\sum_{i=1}^{D} f(\zeta_i)$. Los valores de las funciones de características para un camino se obtienen como la suma de los valores para los pares de puntos del camino evaluado de acuerdo a la Ecuación (1).

$$f(\zeta) = \sum_{k=1}^{N-1} \frac{f(x_k) + f(x_{k+1})}{2} \|x_k - x_{k+1}\| \quad (5)$$

Después, los pesos son iniciados de manera aleatoria (Línea 2). En este caso, por facilidad de comparación, se ha empleado el mismo valor natural aleatorio para todos los pesos, aunque la inicialización con diferentes valores aleatorios arroja los mismos resultados. Es importante comentar que los pesos no son normalizados durante el proceso de aprendizaje, de modo que cambios de valor en un peso no provoquen la variación de los valores del resto de pesos. Una vez finalizado el aprendizaje, entonces se realiza la normalización de los pesos. Por otro lado, los valores de las funciones de características en cada punto sí son normalizados, aunque esto no es un requerimiento del algoritmo. La normalización de pesos y características permite comparar y visualizar fácilmente los costes (mediante el uso de mapas de costes).

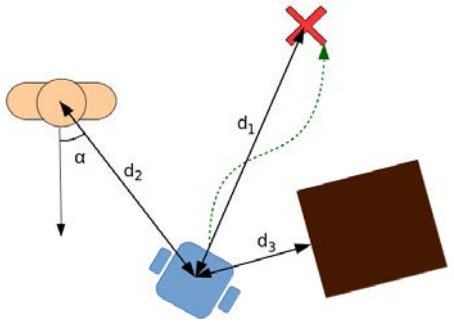
La parte mas notable del algoritmo es el gradiente dado por (4), el cual requiere una comparación de los valores medios de las funciones de características obtenidos de los caminos de demostración y de los caminos del planificador RRT*. Estos últimos son obtenidos ejecutando el planificador repetidas veces utilizando los pesos actuales (Line 7) y obteniendo y normalizando los valores de las funciones de características (Line 8). En las Línesas 10 y 12 se obtienen las medias de estos valores.

Basado en esta comparación, los pesos de la función de coste son actualizados utilizando la técnica del gradiente exponecial descendente (line 14), al igual que en [21]:

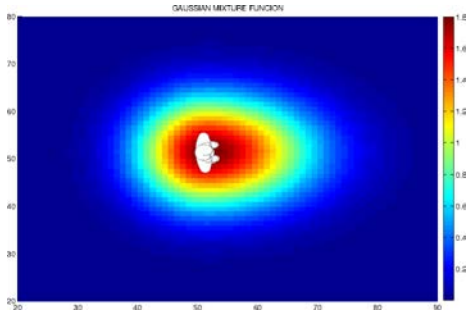
$$\omega_i \leftarrow \omega_i * e^{(\lambda/\phi)*\nabla\mathcal{L}_i} \quad (6)$$

donde ϕ es el número de la iteración actual del algoritmo; λ es un factor de ajuste de la ecuación, y $\nabla\mathcal{L}_i = \frac{\partial\mathcal{L}(\mathcal{D}|\omega)}{\partial\omega_i}$ es el componente i-ésimo del gradiente.

Finalmente, el proceso de aprendizaje termina cuando las variaciones en los valores de los pesos quedan por debajo de un cierto valor de convergencia ϵ.



(a) Características



(b) Función gausiana

Figura 2: (a) Características empleadas en la función de coste aprendida. $d1$, distancia a la meta. d_2, distancia entre el robot y las personas de alrededor. α, ángulo entre el frente de la persona y la posición del robot. d_3, distancia al obstáculo más cercan. (b) Función mixta de dos gausianas situada sobre cada persona. La barra lateral muestra los costes en función del color representado.

3. FUNCIÓN DE COSTE PARA NAVEGACIÓN SOCIAL

La tarea que se plantea es la de navegación social en un entorno de hogar, con distintas habitaciones, pasillos, obstáculos y varias personas situadas en diferentes localizaciones a las que el robot tiene que evitar de manera social para alcanzar su meta. Para ello, se ha hecho uso de un conjunto de funciones de características que se consideran necesarias para la realización de esta tarea. Cabe destacar que este artículo se centra en el uso de esas funciones para navegación social, pero no es su objetivo el discutir sobre la naturaleza e importancia de las funciones de características en si, lo cual es aún una cuestión de investigación abierta.

Las características de la función de coste que se pretende aprender son: la distancia al destino, la distancia desde el robot a las diferentes personas en el escenario, el ángulo del robot respecto a cada persona del escenario α, y la distancia del robot al obstáculo más cercano, como puede observarse en la Figura 2a.

De este modo, se combinan tres funciones de características para obtener la función de coste utilizada por el planificador RRT*. Estas funciones se calculan para cada muestra x_k del espacio de configuraciones. La primera función de características es la distancia euclídea desde la posición del robot hasta la meta.

$$f_1(x_k) = \|x_k, x_{goal}\| \tag{7}$$

La segunda función de característica representa un coste proxémico respecto a las personas en el escenario, inspirado por el modelo utilizado por Kirby et *al.* [8]. La función está definida por una mezcla de funciones gausianas, cuya forma puede verse en la Figura 2. Su valor depende de la distancia (d_{jk}) y del ángulo relativo (α_{jk}) de la posición del robot x_k respecto a cada persona j del escenario. El valor teniendo es cuenta todas las personas del escenario se integra de acuerdo a la siguiente expresión, donde P es el ńumero total de personas:

$$f_2(x_k) = \prod_{j=1}^{P} (p(d_{jk}, \alpha_{jk}) + 1) - 1 \tag{8}$$

La Figura 2b muestra esta función mezcla de dos funciones gausianas para una persona. La primera función gausiana es asimétrica y se sitúa en el frente de la persona, con una varianza $\sigma_f = 1{,}20m$ en el frente de la persona, y una varianza menor lateralmente $\sigma_l = \sigma_f/1{,}5$. La segunda función gausiana se sitúa a la espalda de la persona con $\sigma_f = \sigma_l = 0{,}8$.

La tercera y última función de características se basa en la distancia al obstáculo más cercano para cada muestra x_k, $d_{obs}(x_k)$. La formulación del RRT* descarta estados en los que el robot puede colisionar con obstáculos, así que el objetivo es motivar al robot a mantener cierta distancia de los obstáculos. Esta función se basa en el *costmap* usado por el sistema de navegación de ROS [17], en el que se define una área de inflación alrededor de cada obstáculo. De esta manera el valor es cero si el robot está lo suficientemente lejos de cualquier obstáculo ($\delta = 2$ metros en nuestro caso):

$$f_3(x_k) = \begin{cases} 0, & \text{if } d_{obs}(x_k) > \delta, \\ \lambda e^{(-\beta(d_{obs}(x_k) - r))}, & \text{otherwise} \end{cases} \tag{9}$$

donde r es el radio de la circunferencia inscrita en el footprint del robot, $\lambda = 253$ y $\beta = 3$ en nuestra implementación.

Los valores de las n (3 en este caso) funciones de características son normalizados, y la función de coste para cada muestra x_k se calcula como la suma de estos valores multiplicados por unos pesos:

$$c(x_k) = \sum_{i=1}^{n} \omega_i f_i(x_k) \tag{10}$$

donde $\omega_i \in [0, 1]$ y $\sum_i \omega_i = 1$.

Finalmente, el coste total a lo largo de los Q puntos del camino ζ es obtenido de acuerdo a la función de coste de movimiento empleada por el algoritmo RRT* para calcular el coste del desplazamiento de un punto hacia el siguiente. en este caso, se calculo de igual manera que en la Ecuación 1.

4. RESULTADOS EXPERIMENTALES

Se ha llevado a cabo un conjunto de experimentos para validar la aproximación y evaluar si el algoritmo es capaz de recuperar los pesos que dirigen al planificador a comportarse de manera similar a las trayectorias de demostración.

Todos los experimentos presentados han sido realizados empleando una librería de algoritmos RRT desarrollada por los autores, y que se encuentra disponible, junto con su adaptación para ser usada en ROS, en el Github del Laboratorio de Robótica de Servicio[1] bajo licencia BSD. The hardware empleado fue un computador con un procesador $i7$ 3770 con 12 GB de memoria DDR3.

Con el objetivo de validar el algoritmo, se ha empleado el planificador RRT* con un conjunto de pesos conocidos en su función de coste para generar caminos de ejemplo en un conjunto de escenarios. Después, estos caminos se han utilizado como demostraciones para el algoritmo de aprendizaje de la función de coste. El tiempo de planificación utilizado por el RRT* fue de 2 segundos, tanto para grabar los caminos de ejemplo como durante el aprendizaje. Por lo tanto, los caminos obtenidos utilizando la función de coste obtenida por el aprendizaje, deberían imitar el comportamiento de los caminos de ejemplo.

Particularmente, se han utilizado 25 escenarios diferentes (distintas posiciones iniciales del robot, posiciones de meta y número y posiciones de personas) en distintas partes o habitaciones de un mapa de una casa. 15 de estos escenarios se han empleado para realizar el aprendizaje de los pesos de la función de coste; y los 10 restantes, se han utilizado para comparar los caminos resultantes por validación cruzada. Además, para cada escenario, se han grabado 25 caminos del RRT*. La Figura 3

[1] https://github.com/robotics-upo

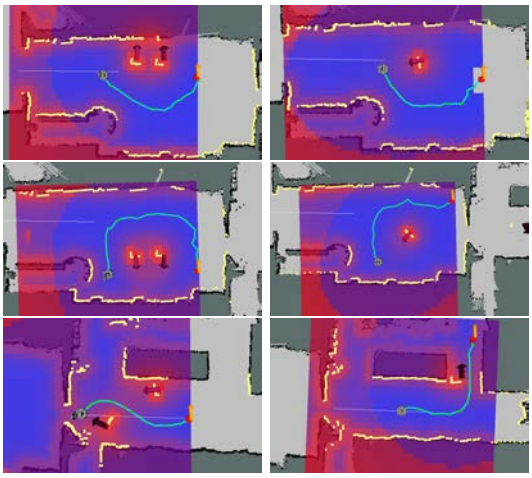


Figura 3: Algunos de los escenarios empleados en la validación cruzada. También se muestra un mapa de costes coloreado basado en la función de coste empleada en el RRT*

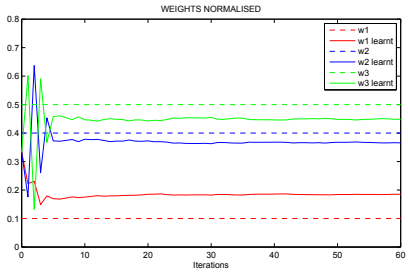
muestra 6 de los 10 escenarios usados para validar los resultados.

La evolución de los valores de los pesos normalizados durante las iteraciones del algoritmo de aprendizaje se pueden ver en la Figura 4. Caber notar que los pesos son inicializados con el mismo valor entero aleatorio. La Figura 4 también muestra la evolución de las medias de las sumas de las funciones de características y los gradientes. Como puede observarse, los pesos convergen a valores próximos a los originales en pocas iteraciones, cometiendo un error final de un 16 % aproximadamente. La diferencia en los valores de las funciones de características, la cual es el objetivo de optimización en (2), se aproxima a cero rápidamente, y por lo tanto, los gradientes y los pesos se estabilizan. El error relativo cometido en los pesos aprendidos respecto a los pesos empleados en las demostraciones se calcula como:

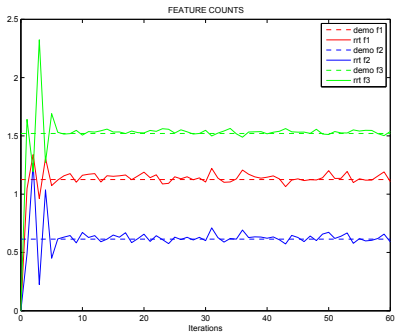
$$RE_\omega = \frac{\|\bar{\omega}_\mathcal{D} - \bar{\omega}_{RRT^*}\|}{\|\bar{\omega}_\mathcal{D}\|} = 0{,}1620 \qquad (11)$$

Una vez finalizado el proceso de aprendizaje, se pueden comparar los caminos de demostración y los caminos obtenidos por el planificador RRT* usando los pesos aprendidos en los escenarios restantes para validación cruzada. La Figura 5 muestra una comparación cualitativa de los caminos en 6 de los 10 scenarios de validación. Como puede observarse, el comportamiento es muy bien reproducido en todos los casos.

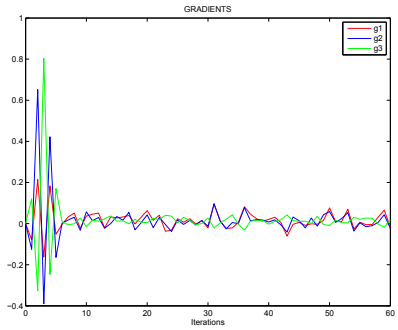
Como el propio planificador RRT* ha sido utilizado para grabar los caminos de demostración, los costes de los caminos también pueden ser comparados a fin de validar los resultados. El error



(a) Pesos



(b) Características



(c) Gradientes

Figura 4: (a) Evolución de los pesos durante las iteraciones de aprendizaje. (b) Evolución de los valores de las funciones de características. (c) Gradientes.

relativo cometido en los costes, y también en las funciones de características, ha sido calculado siguiendo la formulación en (11). La Figura 6 muestra estos errores junto con sus desviaciones estándar. Como puede verse, el error en las funciones de características está por debajo del 8 % en todos los casos, siendo el error en los costes aún menor (por debajo del 4 %). Además, se puede observar en la Figura 5 como en algunos de los escenarios con mayor error (1 y 10) el comportamiento sigue siendo muy bien reproducido.

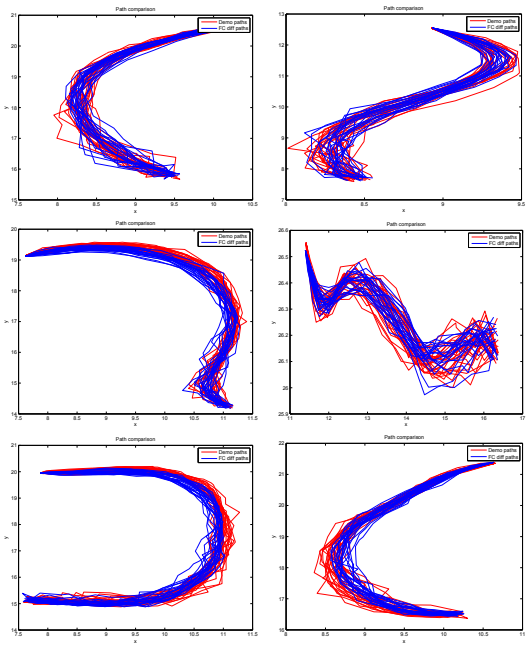
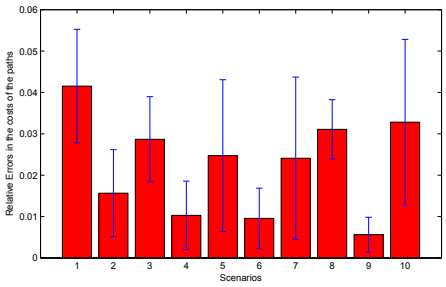
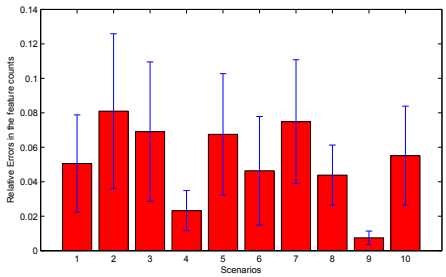


Figura 5: Comparación cualitativa de los caminos de demostración (líneas rojas) y los caminos RRT* obtenidos usando los pesos aprendidos (líneas azules). Los escenarios 1, 4, 5, 6, 7 y 10 se presentan de izquierda a derecha comenzando por la esquina superior izquierda.



(a) Errores relativos en los costes de los caminos



(b) Errores relativos en las características

Figura 6: (a) Errores relativos cometidos entre los costes de los caminos de demostración y los caminos RRT∗ usando los pesos aprendidos. (b) Errores relativos cometidos en los valores de las funciones de características.

5. CONCLUSIONES Y TRABAJO FUTURO

Este artículo presenta una aproximación para el aprendizaje de comportamientos de navegación basado en demostración. Con este fin, se ha implementado un método basado en los conceptos de IRL y enlazado con un planificador RRT* a fin de aprender los pesos de su función de coste, de manera que reproduzca lo más fielmente posible el comportamiento mostrado en las demostraciones. El método es simple de implementar y permite superar los problemas clásicos asociados a los algoritmos IRL basados en MDPs. Además, el método propuesto es significativamente menos costoso computacionalmente que los MDPs y simplifica la generalización del comportamiento gracias a los beneficios intrínsecos de los RRTs.

El método ha sido comprobado en simulación donde las funciones de coste aprendidas son capaces de imitar el comportamiento deseado de manera apropiada. Las medias de los valores en las funciones de características siempre convergen a valores muy cercanos a las demostraciones, y los pesos obtenidos también permiten reproducir el comportamiento con fiabilidad.

Como trabajo futuro se considera la inclusión de las restricciones kinodinámicas del robot en el planificador RRT* y el uso de un conjunto de funciones de características más amplio incluyendo las velocidades del robot y las personas cercanas. Además, la correcta selección de las características involucradas en las tareas de navegación social es aún una cuestión abierta que será investigada. Finalmente, un análisis y caracterización matemáticas del comportamiento del planificador RRT*, así como el estudio de otras técnicas de optimización para resolver el problema, serán consideradas.

Referencias

[1] Abbeel, P., and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning* (New York, NY, USA, 2004), ICML '04, ACM, pp. 1–.

[2] Argall, B., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstrations. *Robotics and Autonomous Systems 57* (2009), 469–483.

[3] Feil-Seifer, D., and Mataric, M. People-aware navigation for goal-oriented behavior involving a human partner. In *Proceedings of the IEEE International Conference on Development and Learning (ICDL)* (2011).

[4] Ferrer, G., Garrell, A., and Sanfeliu, A. Robot companion: A social-force based approach with human awareness-navigation in crowded environments. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (Nov 2013), pp. 1688–1694.

[5] Hall, E. T. *The Hidden Dimension*. Anchor, Oct. 1990.

[6] Karaman, S., and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research 30*, 7 (2011), 846–894.

[7] Kirby, R., J. Forlizzi, J., and Simmons, R. Affective social robots. *Robotics and Autonomous Systems 58* (2010), 322–332.

[8] Kirby, R., Simmons, R. G., and Forlizzi, J. Companion: A constraint-optimizing method for person-acceptable navigation. In *RO-MAN* (2009), IEEE, pp. 607–612.

[9] Kretzschmar, H., Kuderer, M., and Burgard, W. Inferring navigation policies for mobile robots from demonstrations. In *Proc. of the Autonomous Learning Workshop at the IEEE International Conference on Robotics and Automation (ICRA)* (Karlsruhe, Germany, 2013).

[10] Kretzschmar, H., Kuderer, M., and Burgard, W. Learning to predict trajectories of cooperatively navigating agents. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (2014), IEEE, pp. 4015–4020.

[11] Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A. Human-aware robot navigation: A survey. *Robot. Auton. Syst. 61*, 12 (Dec. 2013), 1726–1743.

[12] Kuderer, M., Gulati, S., and Burgard, W. Learning driving styles for autonomous vehicles from demonstration. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), Seattle, USA* (2015), vol. 134.

[13] Levine, S., Popovic, Z., and Koltun, V. Nonlinear inverse reinforcement learning with gaussian processes. In *Neural Information Processing Systems Conference* (2011).

[14] Luber, M., Spinello, L., Silva, J., and Arras, K. Socially-aware robot navigation: A learning approach. In *IROS* (2012), IEEE, pp. 797–803.

[15] Michini, B., Cutler, M., and How, J. P. Scalable reward learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)* (2013), IEEE.

[16] Pacchierotti, E., Christensen, H., and Jensfelt, P. Evaluation of passing distance for social robots. In *IEEE Workshop on Robot and Human Interactive Communication (ROMAN)* (Hartfordshire, UK, Sept. 2006).

[17] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software* (2009).

[18] Shiarlis, K., Messias, J., van Someren, M., Whiteson, S., Kim, J., Vroon, J., Englebienne, G., Truong, K., Evers, V., Perez-Higueras, N., Perez-Hurtado, I., Ramon-Vigo, R., Caballero, F., Merino, L., Shen, J., Petridis, S., Pantic, M., Hedman, L., Scherlund, M., Koster, R., and Michel, H. Teresa: A socially intelligent semi-autonomous telepresence system. In *Workshop on Machine Learning for Social Robotics at ICRA-2015 in Seattle* (2015).

[19] Sisbot, E. A., Marin-Urias, L. F., Alami, R., and Siméon, T. A Human Aware Mobile Robot Motion Planner. *IEEE Transactions on Robotics 23*, 5 (2007), 874–883.

[20] Trautman, P., and Krause, A. Unfreezing the robot: Navigation in dense, interacting crowds. In *IROS* (2010), IEEE, pp. 797–803.

[21] Ziebart, B., Maas, A., Bagnell, J., and Dey, A. Maximum entropy inverse reinforcement learning. In *Proc. of the National Conference on Artificial Intelligence (AAAI)* (2008).