# Comparación de Algoritmos para Planeación de Movimientos: RRT, SST y DIRT

Edgar Granados
*Deparamento Académico de Sistemas Digitales*
*ITAM*
CDMX, México
edgar.granados@itam.mx

***Resumen*—Se presenta el trabajo realizado durante una estancia de verano de investigación en el laboratorio de robótica PRACSYS de Rutgers, la Universidad Estatal de Nueva Jersey así como la continuación de éste. Se describen las simulaciones realizadas y resultados obtenidos utilizando un carro escala 1:10, el simulador Gazebo y el Robot Operating System (ROS). La primera parte consiste en comparar el desempeño de los algoritmos de planeación de movimiento RRT, SST y DIRT mientras que la segunda parte utiliza éstos algoritmos para generar trayectorias seguras.**

***Index Terms*—Robótica, planeación de movimientos, simulación, RRT, SST, DIRT.**

## I. Introducción

La investigación en robótica autónoma ha despertado mucho interés en los últimos años. Ésta se ha enfocado en los carros autónomos particularmente despues del concurso organizado por DARPA [14],[12] y [4]. El problema de navegación autónoma se ha dividido en múltiples capas de decisión, siendo una jerarquía comúnmente utilizada la siguiente: planeación de rutas, selección de comportamientos, planeación de movimientos y control a bajo nivel.

La planeación de rutas consiste en obtener rutas viables en la red de carreteras y caminos disponibles para los vehículos automotores. Es importante notar que la búsqueda se realiza en un ambiente mayormente estático (las calles no suelen ser muy variables e incluso el tráfico suele presentar estacionalidad) y a la salida se esperan puntos a los que se tiene que llegar (p.ej. intersecciones de calles).

La selección de comportamientos involucra definir el objetivo del vehículo en ese momento de acuerdo a las circunstancias: estacionarse, esperar el cambio del semáforo, rebasar un vehículo, detenerse, etc. Como se considera que el total de los comportamientos son pocos, su selección se suele hacer utilizando máquinas de estado cuyas transiciones se dan de acuerdo al sensado del ambiente. Recientemente se ha experimentado con métodos basados en probabilidad para definir el comportamiento del vehículo [15], [5].

La planeación de movimientos clásica consiste en, de acuerdo al comportamiento establecido, definir una ruta que el vehículo sea capaz de seguir (p.ej. un carro necesita realizar una "vuelta en u" para girar 180°, no puede girar sobre su propio eje), esta ruta incluye la evasión de obstáculos. Actualmente, se busca incorporar la dinámica del vehículo a la planeación de movimientos (p.ej. considerar la aceleración que lleva el vehículo para definir el frenado). Éste problema ha sido atacado de diversas formas. Uno de los primeros métodos consistía en generar grafos e intentar resolverlos usando dijkstra o A*. Por la complejidad que representa generar y resolver el grafo, surgieron los métodos de búsqueda incremental que al mismo tiempo muestrean el espacio de búsqueda y construyen el grafo hasta llegar a una solución viable. Una comparación entre los distintos algoritmos de planeación de movimientos se puede encontrar en [13]. En la sección II se presenta una breve descripción de tres algoritmos de búsqueda incremental.

Por último, el control a bajo nivel recibe los valores deseados para las señales de control (para un vehículo: velocidad y ángulo de dirección) para llevar y mantenerlas en dicho valor. Un aspecto que ha tomado relevancia es tomar en cuenta el aspecto humano: cambios bruscos en velocidad o dirección pueden resultar muy incómodos.

La sección II del presente trabajo describe el problema abordado en este proyecto, describiendo de manera general los algoritmos utilizados. En la sección III se presenta el trabajo realizado durante la estancia de investigación. La sección IV muestra los resultados obtenidos y las conclusiones se presentan en la sección V.

## II. Definición del problema abordado

El objetivo de la estancia de investigación fue realizar una comparación de tres distintos algoritmos de planeación de movimientos utilizando una simulación de un carro escala 1:10. Se utiliza un simulador por la demanda en tiempo de cómputo que representan los algoritmos seleccionados, además de facilidades que típicamente ofrecen los simuladores. El robot con el que se trabaja fue seleccionado porque el ITAM cuenta con un robot físico en el cual está basado el simulador: los resultados se pueden implementar en el robot físico. Los tres algoritmos seleccionados están basados en búsqueda incremental: Rapidly-exploring Random Tree (RRT), Stable Sparse-RRT (SST) y Dominance-Informed Region Tree (DIRT).

El algoritmo RRT fue presentado en 1998 [8] enfocado en encontrar rutas viables para robots no-holonómicos (robots que no se pueden mover en cualquier dirección en cualquier momento) con múltiples grados de libertad. RRT construye un árbol de configuraciones válidas. En cada iteración hace crecer el árbol aplicando una señal de control en un tiempo

fijo que busca minimizar la distancia a una muestra del espacio aleatoria.

El algoritmo SST [11], [9] introduce dos modificaciones a RRT: la primera es utilizar otra función para determinar el nodo más cercano al punto muestreado y la segunda es introducir un método para podar el árbol en cada iteración.

Por último, el algoritmo DIRT [10] busca introducir uno de los beneficios de A*: el uso de heurísticas. Para esto, extiende las ideas de SST para formar un árbol donde cada nuevo nodo se forma a partir de regiones de *dominancia informada*. Esto permite utilizar heurísticas para realizar la propagación en cada iteración.

## III. Trabajo realizado

La primera etapa del proyecto consistió en preparar el ambiente de simulación, para el cual se utiliza el simulador Gazebo versión 8.1 utilizando ROS Kinetic. Para el ambiente de pruebas se diseñó un escenario que consiste en un estacionamiento con múltiples autos y postes de luz (obstáculos), además de herramientas auxiliares para la visualización y verificación. A manera de validación del ambiente de pruebas, se implementó una versión del algoritmo A* para llegar de un punto a otro.

Una vez validado el ambiente de pruebas, se procedió a implementar los tres algoritmos a comparar en el siguiente orden: RRT, SST, DIRT, seleccionando las señales de control de manera aleatoria. Una vez implementados, se establecieron cuatro pruebas que consisten en correr cada algoritmo desde la misma pose inicial a cuatro poses finales distintas variando el número de iteraciones. Para cada prueba, se realizan 10 corridas de cada número de iteraciones preestablecido. El número de iteraciones para las pruebas se estableció de la siguiente forma: $1M$, $\frac{1M}{2^1}$, $\frac{1M}{2^2}$, ..., $\frac{1M}{2^{11}}$.

Por último, se implementó otro tipo de selección de la señal de control conocido como *bangbang*. Éste tipo de controlador consiste en solamente utilizar señales de control *extremas*: para la dirección se utiliza derecho, total izquierda y total derecha mientras que para la velocidad se utiliza la máxima y la mitad de ésta. Se corrieron las mismas pruebas que para la versión aleatoria del controlador.

## IV. Resultados

En la Fig. 1 se muestra un ejemplo del árbol obtenido por el algoritmo RRT mientras que la Fig. 2 muestra el árbol de SST y la Fig. 3 muestra el árbol DIRT. En todos los casos, se muestra en amarillo el punto de inicio y en verde el punto meta. En morado se muestran las ramas del árbol y en azul se muestra la rama seleccionada como la mejor trayectoria encontrada.

El árbol RRT es el más extenso de todos, extendiéndose en todo el espacio de búsqueda. Gracias a la cantidad de nodos que tiene RRT, se puede ver que efectivamente se tiene implementada una evasión de obstáculos. En cambio, el árbol SST tiene un número menor de nodos, en particular se puede ver cómo la función de poda de SST elimina la mayoría de las ramas al compararlo con RRT. El árbol DIRT tiene mayor cantidad de nodos que SST y similar a RRT pero a diferencia de RRT, se puede apreciar que el árbol se encuentra *dirigido* hacia donde se encuentra la meta, logrando encontrar una solución similar a la que potencialmente podría encontrar un ser humano.

Debido a la aleatoriedad de los algoritmos, no se encuentra la misma ruta siempre aunque tienden a encontrar rutas similares en múltiples iteraciones. Adicionalmente, es importante tener en cuenta que al tratarse de un carro y no se está considerando la posibilidad de utilizar reversa para reacomodarse, es difícil que una trayectoria recta sea viable.

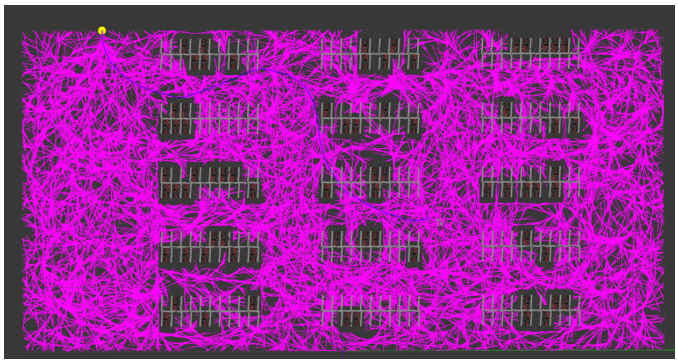

Figura 1. Árbol RRT

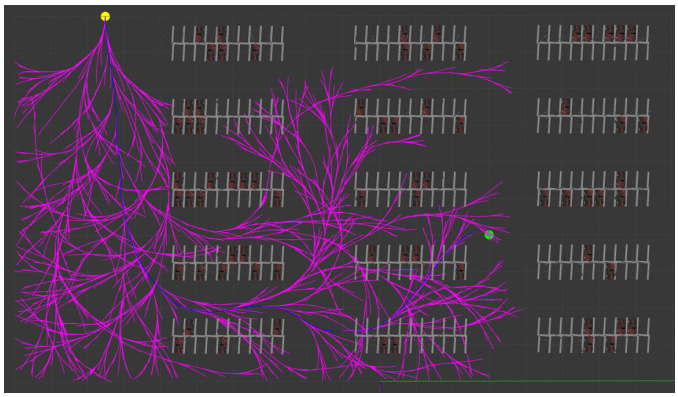

Figura 2. Árbol SST

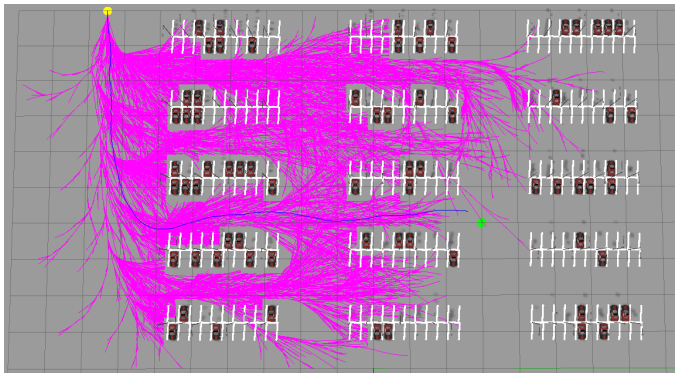

Figura 3. Árbol DIRT

En la Fig. 4 se muestran los resultados para la meta $(-4, +3, \frac{\pi}{2})$: el tiempo de cómputo, el total de nodos en el árbol, la calidad de la solución (duración en segundos) y el porcentaje de corridas que encontraron solución. Ambos casos de SST muestran los menores tiempos de cómputo y DIRT los mayores tiempos, siendo la diferencia más notable en las corridas con mayor número de iteraciones. En cuanto al número de nodos, SST es claramente el que utiliza menor número, consecuencia de la poda del árbol (lo cual contribuye a tener un menor tiempo de cómputo). Tanto RRT como DIRT tienen número de nodos similares. En cuanto a la calidad de la solución (se considera calidad cero como solución no encontrada), es importante notar que en los casos en los que se encuentra solución, salvo DIRT random, el resto tiene soluciones con calidad similar. La calidad de las soluciones tiene una tendencia a tener menor duración a mayor número de iteraciones además de presentar menor variabilidad. En general, SST bangbang presenta mejores soluciones (menor duración) mientras que DIRT random presenta las peores. Por último, en el porcentaje de corridas que encontraron solución, DIRT bangbang es el algoritmo que logra obtener el 100 % de los casos con menor número de iteraciones mientras que SST random es el que necesita mayor número de iteraciones.

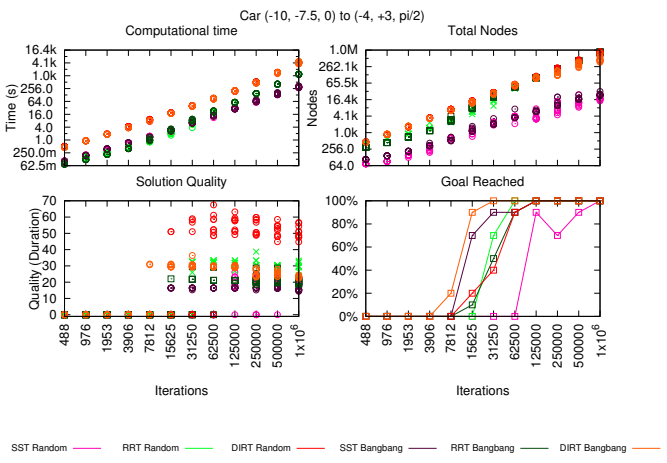


Figura 4. Resultados de $(-10, -7{,}5, 0)$ a $(-4, +3, \frac{\pi}{2})$

## V. Conclusiones

Dentro del área de planeación de movimientos RRT es uno de los algoritmos más populares debido a las ventajas que ofrece frente a A* (no caer en mínimos locales, considerar las restricciones del robot, etc) y algoritmos similares. A pesar de esto, es fácil ver que RRT tiene limitaciones importantes. SST mejora el tiempo de cómputo necesario para encontrar soluciones viables al mantener árboles más pequeños. Ambos algoritmos carecen de poder tener una versión *informada* que, mediante una heurística puedan *dirigir* las soluciones intermedias hacia la meta. DIRT incorpora el tener heurísticas que permitan encontrar soluciones con menor cómputo.

Particularmente, a partir de los resultados de éste trabajo se puede trabajar en dos vertientes. La primera es probar con distintas heurísticas que permitan mejorar la calidad de las soluciones de DIRT además de mejorar el tiempo en el que se encuentran soluciones. La segunda es incorporar uno de estos algoritmos al proyecto presentado en [2] y [3] como una mejora en la forma en que se generan trayectorias. Adicionalmente, se pueden realizar pruebas similares a las presentadas para medir cómo responde cada algoritmo en un sistema físico.

### V-A. *Trabajo Futuro*

A partir de los resultados obtenidos, se propone diseñar un algoritmo de *re-planeación* dinámica, basado en [1], donde se explora la posibilidad de utilizar planes de contingencia como una forma de proveer garantias de seguridad siendo computacionalmente eficiente. En [6] y [7] se presentan los fundamentos teóricos para realizar *re-planeación en línea*. A partir de lo presentado, se puede realizar una intersección entre los algoritmos asimptoticamente óptimos y los enfocados en *re-planeación* segura y eficiente.

## Referencias


[1] Kostas E Bekris y Lydia E Kavraki. *Greedy but Safe Replanning under Kinodynamic Constraints*. Inf. téc. URL: https://www.cs.rutgers.edu/~kb572/pubs/greedy_but_safe_0.pdf.

[2] Jorge A Delgado, Edgar Granados y Marco Morales. «An Application of POMDPs for Autonomous Driving». En: *POMDPs in Robotics: State of The Art, Challenges, and Opportunities at RSS* (2017).

[3] Jorge A Delgado y col. «Integrated Task and Motion Planning for Instances of Autonomous Driving». En: *RSS 2017 Workshop on Integrated Task and Motion Planning* (2017).

[4] Dmitri Dolgov y col. «Path planning for autonomous vehicles in unknown semi-structured environments». En: *The International Journal of Robotics Research* 29.5 (2010), págs. 485-501.

[5] Enric Galceran y col. «Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment». En: *Autonomous Robots* (2017), págs. 1-16.

[6] Kris Hauser. *Adaptive Time Stepping in Real-Time Motion Planning*. Inf. téc. URL: http://motion.pratt.duke.edu/papers/wafr2010-adaptivereplanning.pdf.

[7] Kris Hauser. *On Responsiveness, Safety, and Completeness in Real-Time Motion Planning*. Inf. téc. URL: http://motion.pratt.duke.edu/papers/AuRo2011-RealTime-preprint.pdf.

[8] Steven M LaValle. «Rapidly-exploring random trees: A new tool for path planning». En: (1998).

[9] Yanbo Li, Zakary Littlefield y Kostas E Bekris. *Sparse Methods for Ecient Asymptotically Optimal Kinodynamic Planning*. Inf. téc. URL: https://www.cs.rutgers.edu/~kb572/pubs/stable_sparse_rrt_WAFR14_LLB.pdf.

[10] Zakary Littlefield y Kostas E Bekris. *Efficient and Asymptotically Optimal Kinodynamic Motion Planning via Dominance-Informed Regions*. Inf. téc. URL: https://www.cs.rutgers.edu/~kb572/pubs/iros_dirt.pdf.

[11] Zakary Littlefield, Yanbo Li y Kostas E Bekris. *Efficient Sampling-based Motion Planning with Asymptotic Near-Optimality Guarantees for Systems with Dynamics*. Inf. téc. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.705.9184&rep=rep1&type=pdf.

[12] Michael Montemerlo y col. «Junior: The stanford entry in the urban challenge». En: *Journal of field Robotics* 25.9 (2008), págs. 569-597.

[13] B Paden y col. «A survey of motion planning and control techniques for self-driving urban vehicles». En: *ieeexplore.ieee.org* (). URL: https://ieeexplore.ieee.org/abstract/document/7490340/.

[14] Chris Urmson y col. «Autonomous driving in urban environments: Boss and the urban challenge». En: *Journal of Field Robotics* 25.8 (2008), págs. 425-466.

[15] Junqing Wei y col. «A point-based mdp for robust single-lane autonomous driving behavior under uncertainties». En: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, págs. 2586-2592.