Sinergia Mecatrónica, Capítulo 19, pp. 241 - 253, 2019.
ISBN: 978-607-9394-17-2.

Generación de Trayectorias para Robot Móvil Mediante una Variación del Método RRT

Salas Medina Raúl Alfredo, García Luna Francesco José y Luviano Cruz David

Universidad Autónoma de Ciudad Juárez.
Departamento de ingeniería Industrial y Manufactura, Ingeniería Mecatrónica
al137613@alumnos.uacj.mx

Resumen

Es bien sabido que en la época actual el avance de la robótica móvil ha sido a pasos agigantados, se han logrado desarrollar vehículos autónomos con la capacidad de realizar el traslado de pasajeros de un punto a otro sin la necesidad de un conductor al volante, robot móviles han sido aplicados en líneas de producción para el cuidado y mantenimiento de éstas, robots manipuladores aunados a robots móviles han sido utilizados en almacenes para el manejo y traslado de materiales , lo cual implica un gran campo de aplicaciones posibles para esta tecnología

El presente trabajo plantea la programación de un algoritmo, el cual tendrá la tarea de generar trayectorias entre dos puntos, es decir. Para la generación de las trayectorias se utilizará una variante del método "Rapidly-Exploring Random Trees", a diferencia de este algoritmo, el nuestro genera una sola ramificación en dirección al objetivo y al momento de que un punto generado se encuentre en colisión con un objeto se creará un paso de manera aleatorio para evitar encontrarse nuevamente con el obstáculo. El proceso se realiza hasta llegar al punto objetivo. Después de haber generado la ruta principal, se buscan los puntos mínimos necesarios que de igual manera permitan llegar al objetivo. Finalmente, obtenidos los puntos se ajustan los puntos mediante un spline.

Palabras clave: Generación de trayectorias, robots móviles, RRT.

1. Introducción

Durante el constante avance de la tecnología ha habido grandes cambios en las últimas décadas con respecto al ramo de la robótica, en un pasado se contaba con una cantidad limitada de robots que desempeñaban tareas para optimizar procesos industriales por su precisión, fuerza y velocidad, robots fijos con restricciones de distancia y movimiento [1], pero conforme el tiempo avanza, las innovaciones se han hecho presentes tanto que pueden verse en la creación de robot móviles que se pueden manejarse en dos y tres dimensiones.

Una de las mayores razones para la implementación de robot móviles en diversas áreas, recae en lo que una persona no puede realizar o a donde no puede llegar. Robots móviles han sido utilizados en exploraciones en lugares donde la participación de seres humanos supone un riesgo para su salud, un ejemplo claro son los robots utilizados para la exploración de la planta nuclear de Chernóbil.

Tres aspectos cinemáticos que se deben tomar en cuenta al diseñar un robot son: la movilidad, el control y el posicionamiento. Donde el primero determina cómo será el movimiento del robot para intentar arribar al objetivo, el segundo lidia con las variables cinemáticas, como velocidades, para controlar el movimiento y el tercero es para la localización actual del robot en el área de trabajo, la cual es un aspecto fundamental de la navegación móvil de los robots [2].

Ingredientes básicos para la planeación de trayectorias, según LaValle [3], son el estado, el tiempo, acciones, estado inicial y final, un criterio, si es importante la eficiencia de llegar de un punto a otro y finalmente un plan, el cual determinará que se hará en las tomas de decisiones.

La navegación parte importante de la robótica merece su estudio por separado, ya que, gracias a esta, la interacción con el entorno es posible. Robin R. Murphy [4], en su libro "*Introduction to AI robotics"*, hace mención de cuatros preguntas que son indispensables para la navegación de robots móviles, ¿a dónde voy?, ¿cuál es la mejor manera de hacerlo?, ¿dónde he estado?, ¿dónde estoy?, haciendo alusión que él está en el papel del robot, el cual no tiene una capacidad para recordar, ni para pensar por sí mismo.

La planificación de la ruta del robot es un aspecto importante del diseño del comportamiento de control de navegación. El robot debe alcanzar la configuración final / objetivo con un tiempo mínimo y, al mismo tiempo, la distancia debe ser mínima. Los otros criterios son una menor complejidad computacional y un menor consumo de energía, que solo pueden ser posibles cuando el robot se mueva en la ruta más corta desde la configuración de inicio hasta la configuración del objetivo.

Es un hecho que la demanda de aplicación de robots móviles para realizar actividades que son tardadas y cansadas va en aumento debido a la falta de mano de obra humana. La tarea realizada por humanos dentro de un almacén ha sido repetitiva y exhaustiva. Por ello la implementación de robots autónomos dentro de este ambiente para recoger y colocar objetos en puntos específicos en el almacén supone un ahorro de tiempo y dinero para este tipo de negocios. El método para la evasión de obstáculos propuesta [5] considera que los obstáculos, en este caso los estantes, son estáticos, se cuenta con un robot móvil y espacio libre. Posteriormente creará los caminos evaluando si hay objetos a más de un metro de distancia, sino es así continuará para llegar al punto en donde tomará o dejará el objeto en el estante.

Rapidly-Exploring Random Trees [6], es un algoritmo probabilístico usado para exploración rápida de áreas y planificación de caminos en espacios. El espacio puede ser con obstáculos o sin obstáculos. El algoritmo tiene la posibilidad de trabajar en dos o más dimensiones y para robots holonómicos o no-holonómicos. A pesar de las buenas características, este método tiene dos deficiencias. No es óptimo porque depende de probabilidad y un crecimiento aleatorio en el espacio, además del número de puntos redundantes que se utilizan para explorar la ruta desde el origen hasta el destino es grande.

Planificadores de caminos basado en árboles han sido mostrados como una buena solución para problemas que se presentan al planificar una trayectoria. Estos algoritmos analizan el problema de encontrar una secuencia factible de movimientos de un robot para maniobrar entre obstáculos desde un punto inicial hasta el punto objetivo.

Se han propuesto algoritmos que generan nodos en áreas difíciles, llamados pasajes estrechos. Estos métodos funcionan bien cuando el área de trabajo es similar a la configuración del espacio. Algunos planificadores usan la información conocida del área de trabajo para ayudar a la configuración de muestreo, estos métodos son llamados planificadores que usan indicaciones del espacio de trabajo. En el método propuesto en *"An Obstacle-based RRT"* [7], se usa un vector que delimita los bordes de los obstáculos para facilitar la dirección en la que debe crecer el árbol del método RRT. Además, que la implementación de este ayudará a una mejor generación de puntos en pasajes estrechos.

El movimiento basado en muestreo se ha convertido en un poderoso marco de referencia para la solución de tareas de movimiento para robots. También se ha vuelto casi esencial con el paso del tiempo la planeación de trayectorias de manera efectiva en ambientes dinámicos complejos. La planificación de movimiento apunta a encontrar una secuencia de configuraciones en el robot para llevarlo del estado inicial hasta el estado final u objetivo, con ciertas restricciones del ambiente y la dinámica interna del robot mismo. El algoritmo RRT básico hace crecer un árbol de trayectorias factibles arraigadas en el estado inicial de manera incremental y devuelve una solución cuando una de las trayectorias libres de colisión alcanza el punto objetivo [8].

2. Metodología

Para la realización de esta implementación se llevó una metodología en el siguiente orden: elección de método a utilizar, generación de pseudocódigo, programación de algoritmo, simulación de algoritmo, validación de programación.

2.1 Elección de algoritmo

Para la elección del algoritmo se hizo una investigación de una variedad de algoritmos de planeación de trayectorias, algoritmos como A*, Dijkstra, Potential Field, RRT, entre otros, los cuales tiene en común el objetivo de encontrar la trayectoria la cual lleve desde un punto inicial hasta un punto final. Distintos algoritmos de planeación lo hacen de manera aleatoria, otros de manera incremental u otros seccionan el área de trabajo en partes iguales, pero de igual manera que todos logran el objetivo, pero con distintos procedimientos.

A* es un algoritmo compacto y eficiente [9]. A* es un algoritmo típico de inteligencia artificial de búsqueda heurística. Comparado contra otros métodos de inteligencia artificial, tiene algunas ventajas, como un tiempo de ejecución más corto, alta eficiencia, fácil implementación.

El algoritmo Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo que se encarga de encontrar el camino más corto entre dos puntos. La idea de este método es conocer todos los caminos y conociendo el costo de cada uno elegir el óptimo [1].

Potential Field [10] es un método inspirado en las cargas eléctricas de los objetos. Se puede observar al robot como una partícula con carga positiva y a su vez los obstáculos tendrían el mismo tipo carga con intensión de que se repelasen entre ellos y evitar colisiones entre robot y objetos. Por su parte el punto final debe de ser de carga distinta al del robot creando una atracción entre estos.

Rapidly-Exploring Random Trees, el método elegido como base para elaboración de este proyecto, es un algoritmo probabilístico que genera de ramificaciones de manera aleatoria partiendo de un nodo inicial, estas ramificaciones crecen en toda el área de trabajo y se detienen al llegar al objetivo. Este algoritmo tiene la habilidad de trabajar en dos o más dimensiones y puede ser implementada en robots holonómicos o no holonómicos [8].

Tabla 1. Comparación de métodos

Método	Ventajas	Desventajas
A*	Rápido con buena heurística	Los caminos tienden a estar cerca de los obstáculos
Dijkstra	Camino más corto óptimo	Lento, no hay garantía de separación entre camino y obstáculos
RRT	Rápido	No encuentra el camino más corto, no hay garantía de tener una separación entre los obstáculos y la trayectoria
Potential Field	Se garantiza separación entre obstáculos y la trayectoria	Mal desempeño en pasajes estrechos

2.2 Funciones de programación

Para el desarrollo de este programa de manera eficiente y estructurada, se separó en funciones para agilizar la ejecución debido que al ser un programa iterativo demanda bastantes recursos computacionales

2.2.1 Generación de obstáculos

Esta función está encargada de generar obstáculos de manera aleatoria dentro del área de trabajo.

entrada: mapa, n, alto, ancho
salida : w, h
$n \leftarrow$ cantidad obstaculos
for *i* **to** *n* **do**
  Centroide $w \leftarrow$ Generar un punto aleatorio en el espacio $\mathbb{R}^2$
  $w \leftarrow$ anchura del obstáculo $=$ centroide $\pm$ ancho
  $h \leftarrow$ altura del obstáculo $=$ centroide $\pm$ alto
end

Algoritmo 1: Generación obstáculo

2.2.2 Generación de punto inicial y punto final

La función para generar los puntos inicial y final puede ser utilizada de manera aleatoria o de manera fija para tener un mejor control de la ruta a generar.

Esta función cuando es elegida en manera aleatoria evalúa la colisión con los obstáculos, si se encuentra en esta situación vuelve a generar el punto en otro punto.

entrada: mapa, w, h
salida : pp
$\text{Punto}_{inicial} \leftarrow$ Generar un punto aleatorio en el espacio $\mathbb{R}^2$
while *$\text{Punto}_{inicial}$ choca con obstáculos* **do**
  $\text{Punto}_{inicial} \leftarrow$ generar un punto aleatorio en el espacio $\mathbb{R}^2$
end

Algoritmo 2: Generación Pi y Pf

2.2.3 Verificación de colisión

Esta función tiene como entrada todas las dimensiones de los obstáculos generados por una de las funciones anteriores, las dimensiones del mapa y un punto a evaluar. Se tiene la variable dentroMapa la cual considera los límites del área de trabajo y la variable dentroObs que son los límites de cada uno de los obstáculos.

La variable de salida flag es un indicador booleano que notifica si se ha cumplido la condición dentro de la función de verificación de colisión, es decir, cuando un punto se encuentra dentro del área de trabajo y a su vez dentro de los límites de algún obstáculo.

Una vez obtenidos estos parámetros se analiza si el punto está dentro del mapa y en colisión con un obstáculo, lo cual dará como respuesta un *1 booleano* para indicar que el punto en cuestión está dentro de las fronteras de algún obstáculo.

```
entrada: x, y, xObstáculo, yObstáculo, mapa
salida  : flag
n ← cantidad obstaculos
for i to n do
    dentroMapa ← Revisión de punto dentro del área de trabajo
    dentroObs ← Revisión de punto en colisión con un obstáculo
    if dentroMapa & dentroObs then
        flag = true else
        end
        flag = false
    end
end
```

Algoritmo 3: Verificación de colisión

2.2.4 Generación de ruta

Generación de ruta es la función donde recae más importancia de todo el experimento, obtenidos los puntos inicial y final, los obstáculos, el siguiente paso es generar la ruta.

La función comienza calculando la distancia entre el punto final y el punto inicial, obtenida la diferencia es necesario calcular la normal de dicho intervalo para generar pasos de magnitud uno. Si la distancia entre dichos puntos es menor a uno el paso por dar será de menor magnitud con la intensión de evitar exceder el punto objetivo.

Posteriormente se genera un nuevo punto en dirección hacia el objetivo, el cual está dado por la suma de incremento de magnitud uno y el punto anterior. Es requerido evaluar si este punto se encuentra en colisión con alguno de los obstáculos generados, de ser así se generará un paso con dirección aleatoria con la intención de evadir el obstáculo. Dicho punto aleatorio se evaluará nuevamente bajo la misma condición, una vez que el punto no se encuentra en colisión se trazará sobre el plano.

entrada: $P_{inicial}$, P_{final}, w, h, mapa, reps
salida : ruta
$reps \leftarrow$ cantidad iteraciones
$P_{nuevo} \leftarrow P_{inicial}$
$P_{anterior} \leftarrow P_{nuevo}$
$P_{auxiliar} \leftarrow P_{nuevo}$
$avance \leftarrow P_{final}$ - P_{nuevo}
$Lambda \leftarrow$ Norma de distancia entre puntos
while *iterador* $<$ *reps* **do**
  if *norma*$(avance) > 1$ **then**
    Lambda $= 1$ / norma$(avance)$
    else
      Lambda $= 1$
    end
  end
  incremento $= Lambda$ * $avance$
  $P_{Auxiliar} = P_{anterior}$ + incremento
  while $P_{Auxiliar}$ *colisione con algún obstáculo* **do**
    $P_{Auxiliar} \leftarrow$ generar punto aleatorio en $\mathbb{R}^2$
    $avance_{colisión} = P_{anterior} - P_{Auxiliar}$
    if *norma*$(avance_{colisión}) > 1$ **then**
      lambda$_2$ $= 1$/norma$(avance_{colisión})$
      else
        Lambda$_2$ $= 1$
      end
    end
    incremento$_2$ $= lambda_2$ * $avance_{colisión}$
    $P_{auxiliar} = P_{anterior}$ + incremento$_2$;
  end
  $P_{nuevo} = P_{anterior}$
  ruta $\leftarrow$ Almacenar puntos generados $= P_{nuevo}$
  $avance = P_{final}$ - P_{nuevo}
  $P_{nuevo} = P_{anterior}$ + incremento;
  iterador $=$ iterador $+ 1$;
end

Algoritmo 4: Generación de ruta

2.2.5 Generación de ruta de puntos mínimos

Está función tiene como entrada inicial la ruta obtenida de la función anterior. Sabiendo los puntos generados en la función ruta se realiza una evaluación para utilizar los puntos mínimos.

Se discretiza la distancia entre el punto inicial y los puntos siguientes para analizar si hay colisión durante esa distancia. Cuando haya colisión se guardará el punto anterior al donde hubo colisión, es decir, en el último punto viable y ese punto se convertirá en el punto inicial y se continuará el proceso hasta llegar al punto final.

Una vez obtenidos los puntos mínimos por medio de la función se genera una interpolación de las posiciones creando una curva que pasa por todos los puntos y sin colisión con algún obstáculo.

```
entrada: ruta, w, h, mapa
salida   : puntos
ii ← 1 : Primer elemento del arreglo ruta
jj ← 2 : Segundo elemento del arreglo ruta
slices ← cantidad de particiones
ruta ← puntos almacenados de la ruta inicial
puntos ← puntos mínimos almacenados
while jj <= longitud de ruta do
    P_inicial ← primer elemento de ruta
    P_posible ← segundo elemento de ruta
    distancia = P_inicial - P_posible
    for k = 1 to Slices do
        lambda = k/slices;
        incre = lambda * dist
        pAux = Pini + incre
        P_auxiliar = P_inicial + incremento;
        if P_Auxiliar colisiona con algún obstáculo then
            ii = jj-1
            puntos ← guardar elemento actual de ruta
            break
        end
        if k == slices then
            jj = jj+1
            break
        end
    end
end
puntos ← guardar último elemento de ruta
```

Algoritmo 5: Generación de ruta de puntos mínimos

El siguiente diagrama de flujo describe de manera gráfica el proceso por el cual este sistema genera una trayectoria.

Primero se genera el área de trabajo, se generan los límites del mapa y los obstáculos aleatorios. Seguido de esto se genera el punto inicial y final, los cuales son evaluados para conocer si se encuentran en colisión con algún objeto generado, si la condición es verdadera se generan nuevamente los puntos.

Posteriormente se generarán los pasos en dirección al objetivo mientras el iterador sea menor a la variable reps. Cada paso será evaluado bajo la función de colisión, de ser así se generará un punto aleatorio.

Así mismo una vez terminada la ruta inicial se ejecutará la función de puntos mínimos, tal que se guardará otra ruta que tomará solo los puntos necesarios de la ruta inicial, minimizando la cantidad de posiciones guardada.

Finalmente tomando los puntos mínimos se realiza una interpolación de dichos puntos para trazar una trayectoria curva más agradable para la aplicación en un robot.

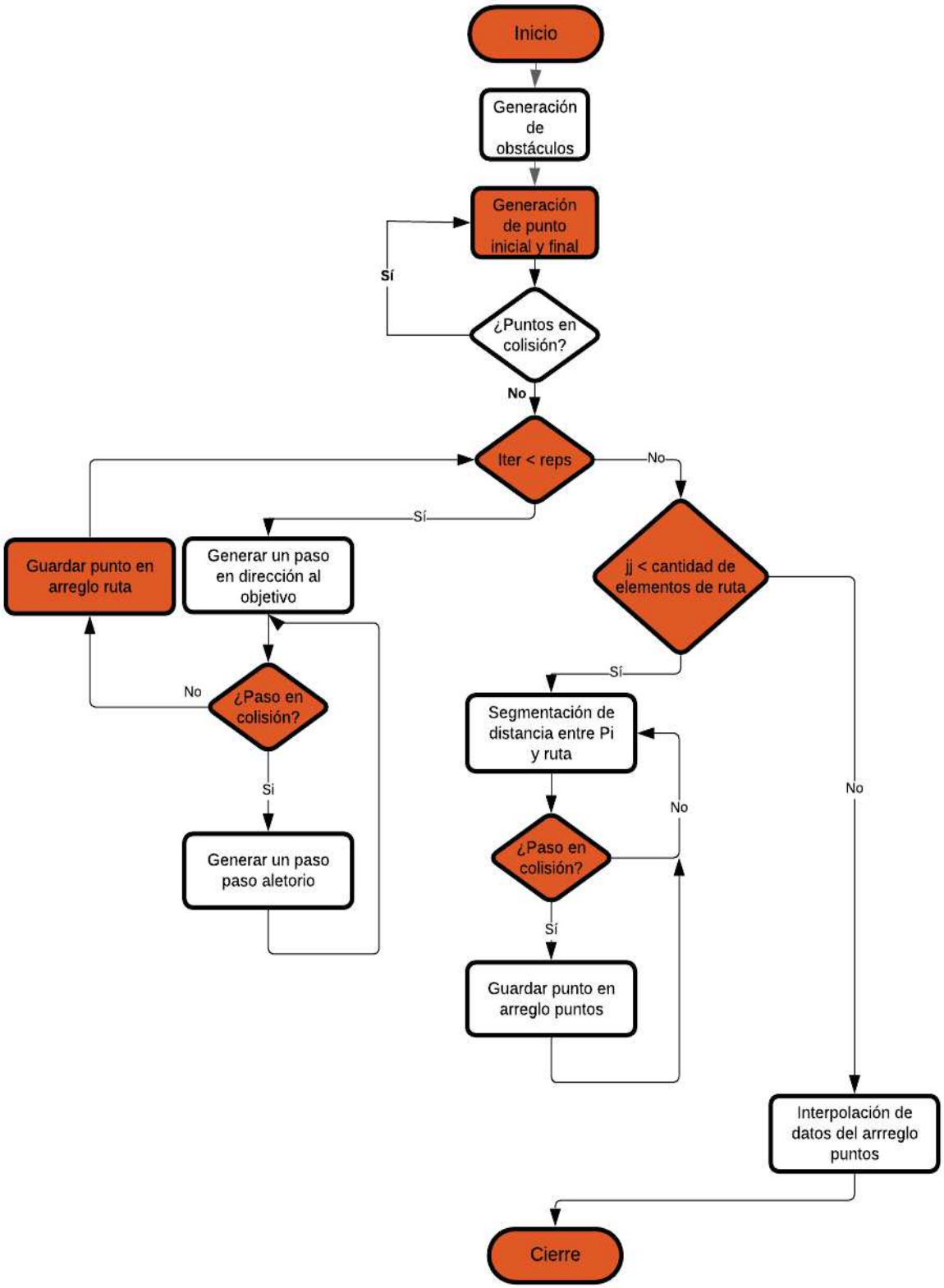

Ilustración 1. Diagrama de flujo del sistema

3. Resultados

En la figura 1 se pueden apreciar los resultados de la función generación de ruta, donde se muestran veinte obstáculos generados aleatoriamente de ancho y alto de dos unidades, dentro de un área de trabajo delimitada.

La ruta se ve generada por los puntos obtenidos en la función siempre en dirección hacia el objetivo a excepción cuando evita colisionar.

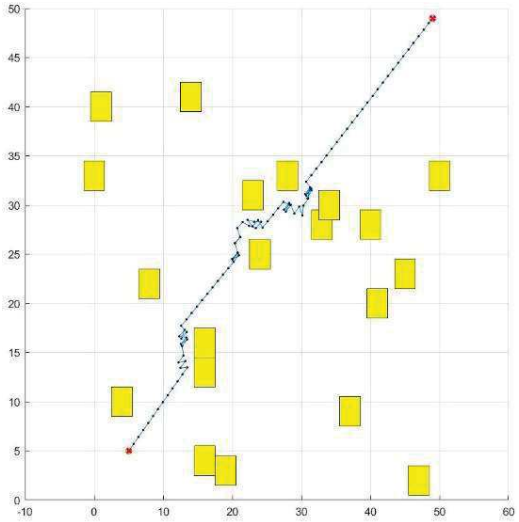

Figura 1. Ruta generada en un área de trabajo con 20 obstáculos con dimensiones de 2x2, punto inicial y final fijos.

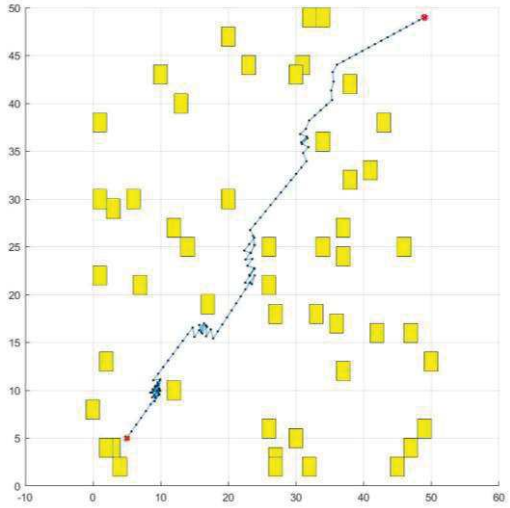

Figura 2. Ruta generada en un área de trabajo con 50 obstáculos con dimensiones de 2x2, punto inicial y final fijos.

En esta figura 2 se muestra un segundo experimento en el cual se generaron 50 obstáculos, lo cual dificulta la generación de la trayectoria, debido que al seguir en dirección al objetivo constantemente resulta en colisiones. Además, en secciones de la ruta generada se observa que realizar un paso de manera aleatoria no es suficiente y se deben realizar dos o más.

La generación de la ruta en un área de trabajo con las mismas condiciones, mismos obstáculos y en las mismas posiciones, resultará siempre en rutas distintas o muy parecidas, pero nunca iguales esto debido a la condicionante de colisión que genera un paso aleatorio en respuesta al punto inválido propuesto.

En la siguiente figura en color rojo, sobre la ruta generada inicialmente, se aprecian los puntos mínimos aplicando la función ruta de puntos mínimos. Puntos que pueden unirse con una recta en forma secuencial sin provocar una colisión con algún obstáculo.

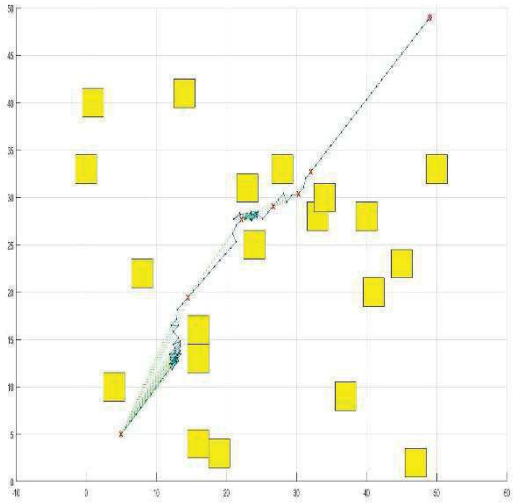

Figura 1. Aplicación de función puntos mínmos en la ruta inicial generada

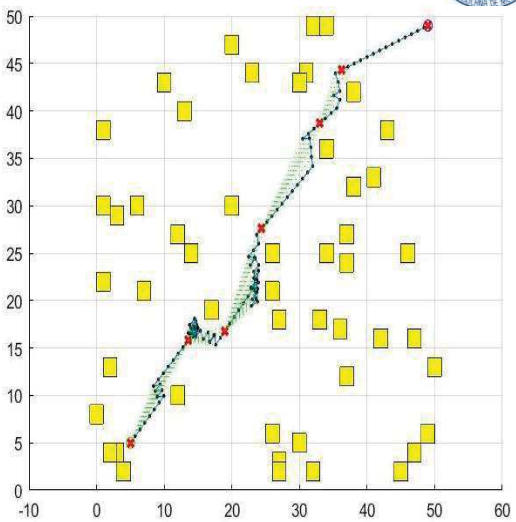

Figura 3. Aplicación de función puntos mínmos en la ruta inicial generada.

En escenarios con una cantidad mayor de obstáculos generados se obliga a la ruta inicial a tener una complejidad mayor, es decir, la ruta tendrá más desviaciones lo cual se puede observar en la figura 4, y a su vez un número mayor de puntos mínimos a diferencia del primer experimento.

Como lo indica la función ruta de puntos mínimos, se evalúa la distancia entre el punto inicial y el siguiente lo cual sucede hasta encontrar una colisión, en la figura 4 con líneas punteadas en verde se puede observar la evaluación entre cada uno de estos puntos y en forma de cruz los puntos sobre los cuales se pueden trazar rectar y unirlos evitando cualquier colisión.

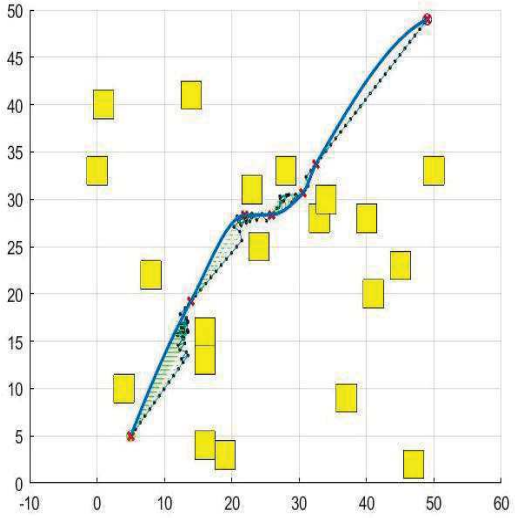

Figura 4. Spline generado con los puntos mínimos, primer experimento.

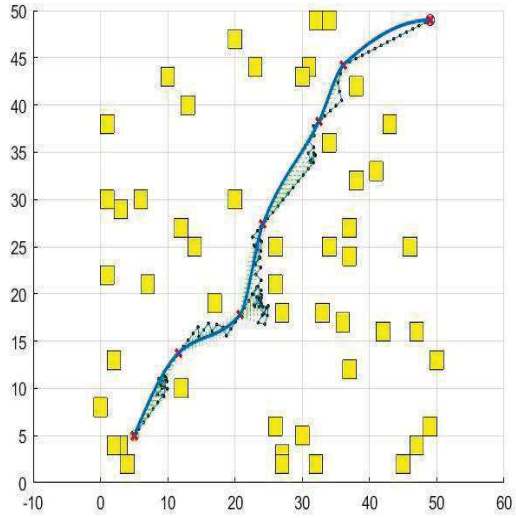

Figura 5. Spline generado con los puntos mínimos, segundo experimento.

Como parte final del experimento se realiza una interpolación de los puntos en color rojo, lo cual resulta en una trayectoria más agradable de ser implementada en un robot móvil.

La figura 7 muestra el error del experimento, dicho error debe converger a cero indicando que se eliminó el error una vez el objetivo fue alcanzado.

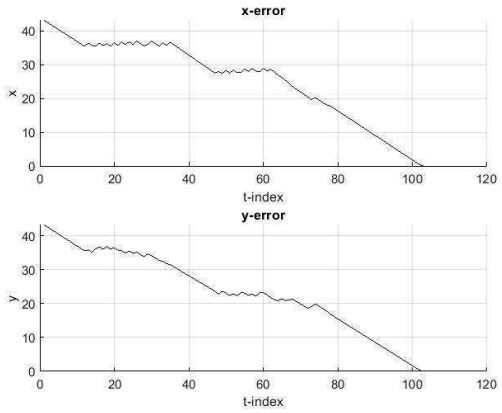

Figura 6. Error calculado de la rua inicial, primer experimento

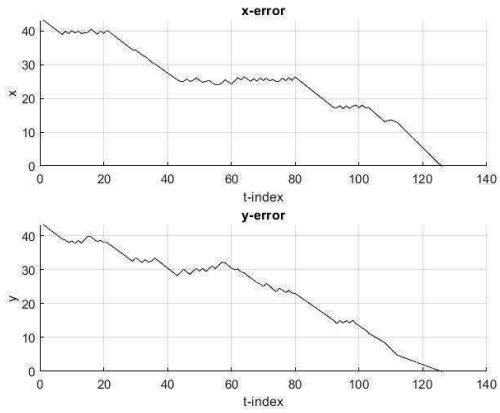

Figura 7. Error calculado de la rua inicial, segundo experimento.

En comparación a la gráfica del error del primer experimento, la figura 8 presenta más perturbaciones, como se mencionó anteriormente, debido a la mayor cantidad de obstáculos generados.

Hay que hacer mención que en el eje de las abscisas es el iterador quien denota cómo se comporta el error con cada repetición.

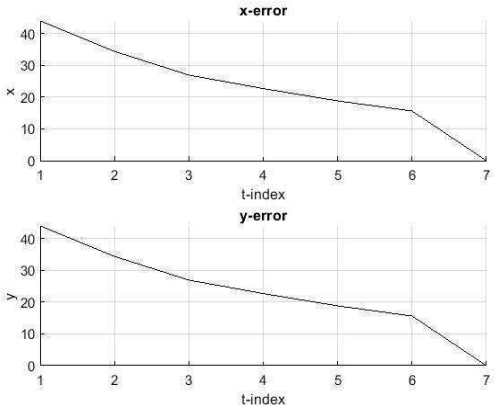

Figura 8. Error calculado del Spline con respecto al punto final, segundo experimento.

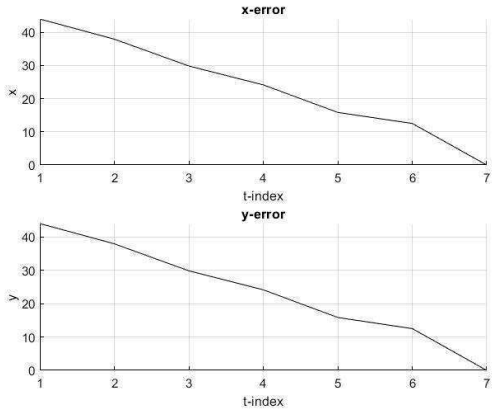

Figura 9. Error calculado del Spline con respecto al punto final, segundo experimento.

En la figura 9 y 10, se observa que la cantidad de iteraciones es solo siete en ambos casos, es debido a que son los puntos mínimos que encontró la función encargada de esto y se puede observar que la convergencia del error es de manera más suave en diferencia a la figura 7 y 8.

4. Conclusiones

Concluimos que esta variación del método RRT para la generación de trayectorias es una buena herramienta cuando el área de trabajo no está tan ocupada por obstáculos que impidan el tránsito, ya que la característica de ir siempre al objetivo lo hace rápido.

Por otra parte, podemos decir que no nos encontramos muy alejados de la era en donde la mayoría los medios de transporte serán tripulados autónomamente, por lo cual, la generación de trayectorias es un tema de interés del cual se seguirán elaborando más tecnologías con la intención de facilitar el transporte de las personas o cargas.

En última instancia, creemos importante mencionar que el *path-planning* tiene una gran cantidad de aplicaciones. No solo se limita a autos para transporte de personas, también puede utilizarse para manejo de material dentro de un almacén e inclusive en líneas de producción.

5. Trabajo futuro

Actualmente se trabaja con la obtención de los parámetros necesarios para la implementación en un robot de tipo diferencial, es decir, conocer los criterios para que le modelo cinemático permita al robot desplazarse siguiendo la trayectoria generada anteriormente.

De igual manera, se plantea escalar a una tercera dimensión este mismo algoritmo con la intención de utilizarlo en dispositivos aéreos, con el cual controlando los parámetros de movimiento pueda ejecutar una ruta en espacio.

Referencias

[1] Siegwart, R., Nourbakhsh, I. R., Scaramuzza, D., & Arkin, R. C. *Introduction to Autonomous Mobile Robots,* MIT Press, England, segunda edición, 2004.

[2] Batlle, J.A., Barjau, A. *Holonomy in mobile robots*, *Robotics and Autonomous systems*. Elsevier, vol. 57, no. 4, pp. 433-440, 2009.

[3] LaValle, S. *Planning algorithms, Cambridge university press*, England, primera edición, 2004.

[4] Murphy, R., Murphy, R. R., & Arkin, R. C. *Introduction to AI robotics, MIT press*, England, primera edición, 2000.

[5] Kumar, N. V., & Kumar, C. S. *Development of collision free path planning algorithm for warehouse mobile robot*, *Procedia computer science*. Elsevier, Vol. 133, pp. 456-463, 2018.

[6] Abbadi, A., Matousek, R., Jancik, S., & Roupec, J. *Rapidly-exploring random trees: 3D planning*, *18th International Conference on Soft Computing MENDEL*, MENDEL, pp.594-599, Czech Republic, 2012.

[7] Rodriguez, S., Tang, X., Lien, J. M., & Amato, N. M. *An obstacle-based rapidly-exploring random tree*, IEEE International Conference on Robotics and Automation, 2006, IEEE, 895-900, Orlando, Florida, 2006.

[8] Adiyatov, O., & Varol, H. A. *A novel RRT*-based algorithm for motion planning in dynamic environments*, *IEEE International Conference on Mechatronics and Automation,* IEEE, pp. 1416-1421, Takamatsu, Japan, 2017.

[9] Peng, J., Huang, Y., & Luo, G. *Robot path planning based on improved A* algorithm*, *Cybernetics and Information Technologies.* De Gruyter Open, vol. 15, no.2, pp.171-180, 2015.

[10] Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., & Thrun, S. *Principles of robot motion: theory, algorithms, and implementation,* MIT press, England, primera edición, 2005.