# Planificación de movimiento robótico con detección de obstáculos

Campó Beraún osé Javier
*Universidad Nacional de ingeniería*
Lima, Perú
jcampob@uni.pe

Hamer Franklin Jara Ocas
*Universidad Nacional de Ingeniería*
Lima, Perú
hamer.jara.o@uni.pe

Peralta Haro Katheryn Ximena
*Universidad Nacional de Ingeniería*
Lima, Perú
katheryn.peralta.h@uni.pe

Quispe Amao Renzo Renato
*Universidad Nacional de Ingeniería*
Lima, Perú
rrquispea@uni.pe

***Resumen*—La planificación de la ruta es la capacidad de un robot para buscar una ruta factible y eficiente hacia la meta. Utilizando el algoritmo dynamic window approach y una simulación de ray-casting para poder optimizar la planificación de ruta de nuestro vehículo. En el presente trabajo se implementó la simulación con pygames y algunas mejoras a los algoritmos para poder detectar y evitar obstáculos más eficiente disminuyendo el número de obstáculos a calcular.**

**Índice de Términos— Dynamic window approach, ray-casting**

## I. Introducción

Los algoritmos que generan movimiento para robots móviles se pueden dividir en algoritmos de planificación y algoritmos para evitar obstáculos en tiempo real. Los algoritmos de planificación consideran un modelo o mapa del entorno para calcular una ruta desde la posición actual del robot hasta el objetivo, mientras que los algoritmos para evitar obstáculos suelen utilizar información sensorial para determinar un movimiento que evitará la colisión con obstáculos en el entorno. Para la mayoría de las aplicaciones en robótica móvil, el entorno es parcial o completamente desconocido y cambia con el tiempo. En tales circunstancias, las trayectorias generadas por los algoritmos de planificación se vuelven inexactas y se requiere una nueva planificación para alcanzar la meta. Dado que la planificación puede llevar demasiado tiempo para evitar colisiones en tiempo real, los comandos de movimiento para robots móviles generalmente se generan mediante enfoques de evasión de obstáculos en tiempo real y computacionalmente eficientes. Sin embargo, la evasión de obstáculos puramente reactiva puede no resultar en el comportamiento requerido para realizar la tarea del robot. [2] Uno de los objetivos finales de la investigación de robótica móvil en interiores es construir robots que puedan llevar a cabo misiones de forma segura en entornos peligrosos y poblados. Por ejemplo, un robot de servicio que ayuda a los humanos en entornos de oficinas interiores debería poder reaccionar rápidamente a cambios imprevistos, y realizar su tarea bajo una amplia variedad de circunstancias. Para construir robots móviles autónomos, uno tiene que construir sistemas que puedan percibir su entorno, reaccionar ante circunstancias imprevistas y volver a planificar dinámicamente para lograr sus tareas. [3] Este artículo se centra en utilizar el enfoque de ventana dinámica para evitar colisiones con los obstáculos y a su vez detectar con el ray-casting estos obstáculos para poder evitarlos en una simulación en pygames con obstáculos con dimensiones. El siguiente articulo aplica el enfoque de ventana dinámica, nuestro enfoque difiere al algoritmo al simularlo con pygames con obstáculos y una simulación de un radar que mapea los obstáculos a cierta distancia lo que disminuye el número de cálculos e iteraciones a la hora de minimizar la función objetivo. El documento describirá la historia del algoritmo , la ecuaciones de movimiento del robot, el algoritmo de dynamic window approach, y los cambios hechos en una simulación 2D en la cual esta hecha de pixeles. En la penúltima sección se discutirá los efectos de variar parámetros de la función objetivo así como el tiempo que toma cada uno. Para finalizar se mencionará nuestras discusiones.

## II. Estado del Arte

Los enfoques de prevención de colisiones para robots móviles se pueden dividir aproximadamente en dos categórias : global y local. Las técnicas globales, como la hoja de ruta y métodos de campo potenciales, generalmente en estos se dispone de un modelo completo del entorno del robot. La ventaja de enfoques globales radica en el hecho de que una trayectoria completa desde el punto de partida hasta el punto destino se puede calcular fuera de línea. Sin embargo, los enfoques globales

no son apropiados para evitar obstáculos rápidamente. Una desventaja de los métodos globales es su lentitud debido a la complejidad inherente de la planificación del movimiento del robot. en conclusión, planificar en un modelo global suele ser demasiado caro para realizarlo repetidamente.[2] Los enfoques locales o reactivos, por otro lado, utilizan solo una pequeña fracción del mundo modelo, para generar control del robot. Esto tiene la desventaja de que no pueden producir soluciones óptimas. Los enfoques locales quedan atrapados fácilmente en mínimos locales como configuraciones de obstáculos en forma de U. Sin embargo la ventaja clave de las técnicas locales sobre los globales radica en su baja complejidad computacional, que es particularmente importante cuando el modelo mundial se actualiza con frecuencia en función de la información del sensor. Estos métodos son extremadamente rápidos y normalmente considere solo el pequeño subconjunto de obstáculos cerca del robot. Borestein y Koren identificó que tales métodos a menudo no logran encontrar trayectorias entre obstáculos poco espaciados; también pueden producir un comportamiento oscilatorio en pasillos estrechos. en conclusión, los métodos locales suelen ser muy rápidos, y se adaptan rápidamente a cambios imprevistos en el entorno. [3]

### *II-A. Ecuaciones de la dinámica del robot*

Esta sección describe las ecuaciones de movimiento para el robot movil. Asumiendo que el robot tiene movimiento de traslación y rotacional de manera independiente. Las trayectorias candidatas del robot consisten un número finito de segmentos de arcos. Esta representación es conveniente a la hora de comprobar colisiones.

*II-A1. Ecuaciones de movimiento:* El robot tiene un estado en cada instante de variables de la forma : $(x(t), y(t), (t), v(t), w(t))$ que denotan la posición y orientación del robot en un plano 2D, la velocidad y la velocidad angular instante $t$ simulando un tiempo discreto. entonces tenemos las siguiente ecuaciones de movimiento:

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot \cos\theta(t) \cdot dt \tag{1}$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot \sin\theta(t) \cdot dt \tag{2}$$

$$v(t_n) = v(t_0) + \int_{t_0}^{t_n} \dot{v}(t) \cdot dt \tag{3}$$

$$\theta(t_n) = \theta(t_0) + \int_{t_0}^{t_n} w(t) \cdot dt \tag{4}$$

$$w(t_n) = w(t_0) + \int_{t_0}^{t_n} \dot{w}(t) \cdot dt \tag{5}$$

Sustituyendo las ecuaciones 5 en 4, 4 y 3 en 1 tenemos la siguiente ecuación que solo depende de las variables que describen el estado en un tiempo $t$.

$$x(t_n) = \theta(t_0) + \int_{x=t_0}^{x=t_n} (v(t_0) + \int_{t_0}^{t_n} \dot{v}(\hat{t}) \cdot d\hat{t}) \cdot \cos\left(\theta(t_0) + \int_{t_0}^{t_n} (w(t_0) + \int_{t_0}^{t_n} \dot{w}(\tilde{t}) \cdot d\tilde{t}) \cdot d\hat{t}\right) \cdot dt \tag{6}$$

### *II-B. Enfoque de Ventana Dinamica*

El enfoque de ventana dinámica es una estrategia en linea para evitar colisiones para robot móviles desarrollada por Dieter Fox, Wolfram Burgard y Sebastian Thrun en 1997. El método se deriva directamente de la mecánica del robot. El enfoque considera periódicamente la siguiente acción un par de velocidad y velocidad angular en el próximo intervalo de tiempo aproximando las trayectorias buscando una velocidad óptima y permitiendo al robot detenerse de forma segura. Estas velocidades forman la **ventana dinámica** que se centra alrededor de las velocidades actuales del robot en el espacio de velocidades. [3] Entre las velocidades admisibles, la combinación de la velocidad y la velocidad de rotación se elige minimizando una función objetivo. La función objetivo evalúa la dirección, la velocidad de avance y las distancias a los obstáculos. La combinación de todos los objetivos conduce una estrategia de prevención de colisiones muy robusta y elegente. [3] En el enfoque de ventana dinámica se realiza la búsqueda de comandos que controlan el robot directamente en el espacio de velocidades. La dinámica del robot está incorporada en el método reduciendo el espacio de búsqueda a aquellas velocidades que son alcanzables bajo restricciones dinámicas. Además de esta restricción, solo se consideran las velocidades que están a salvo con respecto a los obstáculos. Esta poda del espacio de búsqueda se realiza en la primer paso del algoritmo. En el segundo paso, la velocidad que minimiza la función objetivo es elegido de las velocidades restantes. [3]

### *II-C. Espacio de Búsqueda*

*II-C1. Trayectoria Circular:* Cada curvatura está determinada de forma única por el vector de velocidad $(v_i, w_i)$, a la que nos referiremos simplemente como velocidad. Generar una trayectoria hacia una meta determinada (punto) para los siguientes $n$ intervalos de tiempo, el robot tiene que determinar las velocidades $(v_i, w_i)$ , uno para cada uno de los n intervalos entre $t_0$ y $t_n$ . Esto debe hacerse bajo la premisa de que la trayectoria resultante no se cruza con ningún obstáculo. El espacio de búsqueda para los vectores es exponencial en el número de intervalos considerados. [3] Para

que la optimización sea factible, el enfoque de ventana dinámica considera exclusivamente el primer intervalo de tiempo, y asume que las velocidades en los n-1 intervalos de tiempo restantes son constantes, lo que equivale a asumir cero aceleraciones en $[t_1; t_n]$. [3]

*II-C2. Velocidades Admisibles:* Los obstáculos en el entorno más cercano al robot imponen restricciones a las velocidades de rotación y traslación. Por ejemplo, la velocidad máxima admisible en una curvatura depende de la distancia al siguiente obstáculo en esta curvatura. La velocidad se considera admisible si el robot puede detenerse antes de alcanzar este obstáculo. [3]

*II-C3. Ventana Dinámica:* Para tener en cuenta las aceleraciones limitadas que pueden ejecutar los motores. El espacio de búsqueda se reduce a la ventana dinámica que contiene solo las velocidades que se puede alcanzar en el próximo intervalo de tiempo. [3] La ventana dinámica se centra alrededor de la velocidad real y sus extensiones dependen de las aceleraciones que se puedan ejercer. Todas las curvaturas fuera de la ventana dinámica no se pueden alcanzar dentro del siguiente intervalo de tiempo y, por lo tanto, no se tienen en cuenta para evitar obstáculos. [3]

### *II-D. Minimizando de la función objetivo*

Después de encontrar un espacio de velocidades. Se calcula sobre las velocidades candidatas el mínimo de la siguiente función objetivo:

$$G(v, w) = \alpha \cdot heading(v, w) + \beta \cdot dist(v, w) + \gamma \cdot vel(v, w) \tag{7}$$

1. Heading: mide la alineación del robot con la dirección del objetivo.
2. dist : es la distancia al obstáculo más cercano y el deseo del robot de evadirlo.
3. vel : velocidad de avance del robot.

### *II-E. Raycasting y la prevención de colisiones*

El método raycasting se utiliza como actualizador del mapa, su función es identificar los objetos que se encuentren en el espacio definido, se utilizan rayos para identificar el entorno mediante una búsqueda dentro de un cono de medición.

*II-E1. Enfoque global de prevención de colisiones:* La principal ventaja de estos métodos es que al tener un enfoque global se puede calcular una trayectoria desde el punto de partida hasta el punto final.Sin embargo, al hacer cálculos globales se consume mucho costo computacional además de que no es el mas apropiado en cuanto a esquivar obstáculos rápidamente, su fuerza está en el mapeo completo del espacio.

*II-E2. Enfoque local de prevención de colisiones:* En este caso se utiliza solo una porción del espacio detectado, su principal desventaja es su atasco en mínimos locales, sin embargo, su fortaleza esta en su bajo costo y rapidez de respuesta, esto permite una actualización frecuente y una adquisición de información constante con lo cual se pueden evitar obstáculos de buena manera.

*II-E3. Ponderación basada en la extensión:* El raycast junto con los algoritmos de línea habituales supone implícitamente que el ancho del rayo es igual a la resolución del mapa, dado que este no es claramente el caso, usamos las extensiones locales de los grupos en dirección perpendicular al haz del sensor como un peso adicional para modelar el ancho real del haz.
En general, un grupo $C$ de $N = |C|$ puntos, es una distribución muestral con media muestral $m$ y matriz de covarianza muestral $\sum$.
Sean $\lambda_1, \lambda_2$, tal que $\lambda_1 \geq \lambda_2$ los autovalores y $v_1, v_2$ los autovectores de $\sum$.
Entonces, $C$ forma una elipse de desviación estándar con radios a=$\sqrt{\lambda_1}$ y b=$\sqrt{\lambda_2}$
Sea $m'$ un vector perpendicular al rayo desde $o$(origen) hasta $m$.Entonces el ángulo entre el vector propio $v_1$ y $m'$ es:

$$\theta = \tan^{-1}\left(\frac{m'_y}{m'_x}\right) - \tan^{-1}\left(\frac{v_{1,y}}{v_{1,x}}\right)$$

El vector $m'$ corta esta elipse en un punto $P(x, y)$, cuyas coordenadas en el marco de referencia de la elipse, definido por los vectores propios $v_1, v_2$, se pueden calcular mediante la ecuación paramétrica de la elipse como:

$$x = a * \cos(\theta)$$
$$y = b * \text{sen}(\theta)$$

La distancia euclidiana entre $m$ y $P$ es entonces la mitad de la componente perpendicular deseada $d_{m'}$ de la elipse, es decir:

$$\frac{d'_m}{2} = \sqrt{a^2 * \cos^2(\theta) + b^2 * \text{sen}^2(\theta)}$$

Ahora para modelar la resolución angular $\alpha$ del escáner y para controlar los casos donde $N = |C| = 1$, adicionalmente extendemos el componente perpendicular dependiendo de $\alpha$ y la distancia de $m$ al origen del sensor $o$, utilizando:

$$d_{\alpha,m} = \tan(\alpha) * ||m - o||_2$$

El peso basado en la extensión del grupo $|C|$ se compone entonces de $d'_m$ y $d_{\alpha,m}$, establecidos en relación con la resolución $r$, es decir:

$$w_{c,C,Extent} = \frac{d'_m + d_{\alpha,m}}{r}$$

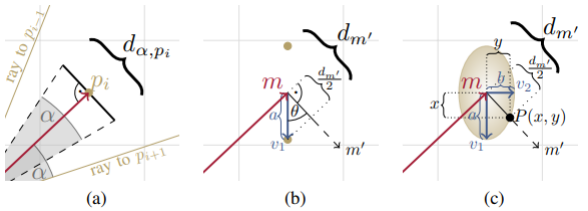


Figura 1. Se muestran los tres casos de ponderación basada en la extensión, dependiendo de la número N de puntos del grupo: (a) $N = 1$, (b) $N = 2$ y (c) $N > 2$.Los puntos de medición, los rayos y los grupos se dibujan en amarillo, las medias de los grupos y los rayos a las medias del grupo en rojo, y los vectores propios y -valores en azul.

## III. Trabajos relacionados

Cachumba[b1] presenta la implementación de un asistente inteligente de conducción autónomo, con el fin de prevenir colisiones a causa de la desatención o impericia de los conductores. Para lograr lo requerido, se desarrolla un entorno gráfico en Unity 3D, en el que puede interactuar el usuario mediante los dispositivos hápticos con el vehículo. Para el vehículo se toma como referencia el modelo cinemático de un robot tipo car-like, modelado en Matlab, este permite recrear el comportamiento de un vehículo real. La evasión de obstáculos se da cuando un objeto se encuentra en la trayectoria del vehículo y sobrepasa la distancia mínima de maniobra, en tal caso el controlador toma el mando del vehículo, realiza la evasión, posiciona el vehículo dentro de la trayectoria original y le devuelve el control al conductor. Durante este proceso de evasión se tiene una retroalimentación de fuerza en el volante, que gira de acuerdo a la posición del obstáculo y el pedal de freno, que se acciona según la velocidad del automóvil.otro trabajo realizado sobre la conducción autónoma para la prevención de colisiones usando Ray Casting es la investigación[b5] de la implementación de un asistente de conductor autónomo para la prevención de colisiones en un entorno virtual que permita la inmersión del usuario en el entorno; el coche se controla con dispositivos hápticos y la escena se puede ver en las gafas de realidad virtual, el asistente autónomo solo funciona cuando se detecta un obstáculo, este asistente evita la colisión con el obstáculo y luego devuelve el coche a su paso. Los resultados se basan en las pruebas per-formadas comprobando que el modelo cinemático y la ley de seguimiento de trayectoria están implantados del asistente autónomo permiten demostrar su función y la correcta trayectoria siguiendo la ley asegurando el correcto funcionamiento del asistente de conducción autónoma.

## IV. Metodología

En esta sección se describen las herramientas y la métodología que se usarán en el presente trabajo. El trabajo esta hecho en pygames en el cual simulamos un automóvil con un radar que mapea un área determinada en busca de obstáculos, el radar hace visible los obstaculos que el algoritmo de dynamic window approach tendrá en cuenta a la hora de hacer cálculos. El algoritmo esta basado de python Robotic algorithm de Atsushi Sakai con modificaciones para que pueda ser simulado en un entorno pixeleado. El algoritmo mapea un un área circular alrededor de él encontrando obstáculos y añadiéndolo al conocimiento del robot para poder calcular la ventana dinámica y minimizar la función objetivo para hallar una trayectoria libre de obstáculos respetando la dinámica del robot.

## V. Experimentación y resultados

La experimentación se ejecutó en un entorno virtual python, utilizamos las siguientes librerías:

- numpy 2.19.0
- pygame 2.0.1

### V-A. *Numpy*

NumPy es un proyecto de código abierto que tiene como objetivo permitir la computación numérica con Python. Fue creado en 2005, basándose en el trabajo inicial de las bibliotecas Numeric y Numarray. NumPy siempre será un software 100b % de código abierto, de uso gratuito para todos y publicado bajo los términos liberales de la licencia BSD modificada.

### V-B. *pygame*

Pygame es un conjunto de módulos de Python diseñados para escribir videojuegos. Pygame agrega funcionalidad además de la excelente biblioteca SDL . Esto le permite crear juegos y programas multimedia con todas las funciones en el lenguaje Python. Pygame es altamente portátil y se ejecuta en casi todas las plataformas y sistemas operativos.Pygame es gratis. Lanzado bajo la licencia LGPL, puede crear juegos de código abierto, freeware, shareware y comerciales con él. Consulte la licencia para obtener todos los detalles.

### V-C. *Desarrollo de la simulación*

Para la simulación en pygame los obstáculos son leídos desde un un fichero la cual contiene el centro de los obstáculos. Con pygame dibujamos estos obstáculos como cuadrados con lado definido, por defecto es 20 píxeles. Así mismo nuestro robot es un automóvil de forma circular o rectangular , sus dimensiones están almacenadas en el atributo *config* del robot y con esta información esta dibujada la imagen del automóvil en nuestro plano 2D.

**Algorithm 1:** Dynamic window approach

**Input:**estado actual $x_t$
**Result:** Ventana dinámica, un espacio de velocidades
Vs = $[V_{min}, V_{max}, w_{min}, w_{max}]$
Vd = $[V_t - \dot{V} \cdot dt, V_t + \dot{V} \cdot dt, w_t - \dot{w} \cdot dt, w_t + \dot{w} \cdot dt]$
dw = $[max(Vs[0], Vd[0]), min(Vs[1], Vd[1]),$
max(Vs[2],Vd[2]) ,min(Vs[3],Vd[3]) ];
**return** *dw*;

**Algorithm 2:** Algoritmo de control de la trayectoria de la ventana dinámica

**Input:**estado actual $x_t$, ventana dinámica *dw*
**Result:** par de valores u = [velocidad,velocidad angular],trayectoria, trayectoria candidatas
x_inicial $\leftarrow x$
costo_mínimo $\leftarrow \infty$
mejor_velocidad $\leftarrow [0, 0]$
mejor_trayectoria $\leftarrow x$
trayectoria_candidatas $\leftarrow$[ ]
**for** *v,y in dw[:2]* **do**
trayectoria$\leftarrow$ $predecir\_trayectoria(x\_inicial, v, y)$
trayectoria_candidatas $\leftarrow$ *trayectoria*
Calcular el costo de la trayectoria
Comparar el costo & escoger el de menor costo
**return**
*mejor_velocidad, mejor_trayectoria, trayectoria_candidatas*;

| # | objetivo | velocidad | obstáculos | tiempo |
|---|---|---|---|---|
| 1 | 0.3 | 20 | 20 | 142.38 seg |
| 2 | 3.0 | 20 | 20 | 152.45 seg |
| 3 | 1500 | 20 | 20 | 146.46 seg |
| 4 | 15000 | 20 | 20 | 162.25 seg |
| 5 | 0.3 | 2000 | 20 | 142.75 seg |
| 6 | 0.3 | 0.2 | 20 | 141.41 seg |
| 7 | 0.3 | 20 | 2000 | 278.493 seg |
| 8 | 0.3 | 20 | 200 | 112.996 seg |
| 9 | 0.3 | 20 | 2 | 141.97 seg |
| 10 | 0.3 | 20 | 0.2 | 143.45seg |
| 11 | 0.3 | 20 | 0.02 | 142.86seg |

Cuadro I
VARIACIÓN DE LAS CONSTANTES DE LA FUNCIÓN OBJETIVO Y EL TIEMPO ESTIMADO EN LLEGAR A LA META

### *V-D. Algoritmo de Dynamic window approach*

Estos dos 1 y 2 son los algoritmos principales para construir un enfoque de ventana dinámica y calcular la trayectoria óptima.

Como el algoritmo solo detecta la colisión de puntos se propuso trabajar con una muestra de puntos que simulen ser un objeto. Así mismo se simula un radar que mapea 100 píxeles alrededor del robot en busca de obstáculos.

### *V-E. Resultados*

En el cuadro I se muestran los resultados comparativos de la variación de las constantes(costos) de la función objetivo y el tiempo medido en completar la tarea de llegar a la meta. Se varió considerando una medida pequeña, mediana y grande.

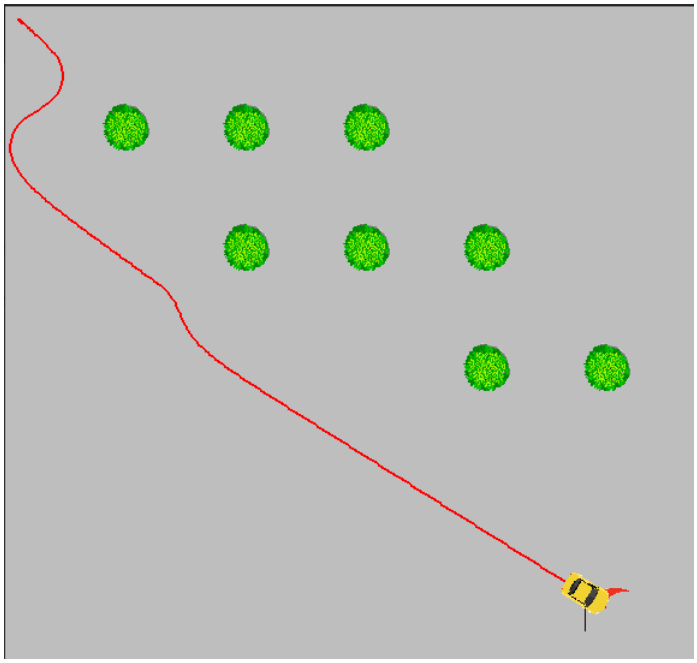

Figura 2. Visualización de la fila 1 del cuadro I del camino recorrido cuando el costo del objetivo es pequeño a comparación de los otros costos.

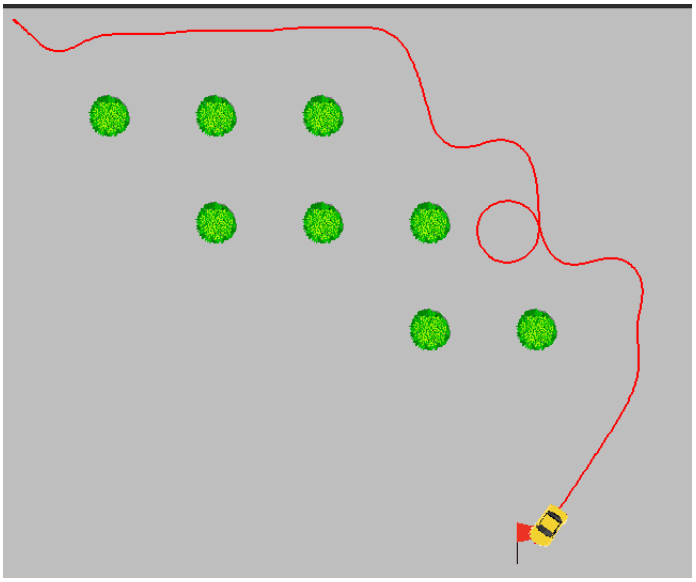

Figura 3. Visualización de la fila 7 del cuadro I del camino recorrido cuando el costo de de obstaculos es grande a comparación de los otros costos.

## VI. Discusión de resultados

- En cuanto a la variación del costo de meta que calcula un ángulo de error con respecto a la dirección de la meta. En la simulación no mostró una variación significativa de la ruta además se evidenció que aumentaba el tiempo de llegada.
- En cuanto a la variación del costo de velocidad se podía intuir que iba a desplazarse con su velocidad máxima pero solo aumentó el tiempo de llegada , y además casi invertía su dirección al acercase a un obstáculo.
- Por último la variación del costo de obstáculos si ayudaba a disminuir el tiempo de llegada. Un costo grande , no extremadamente grande cambio la ruta y empezó a bordear los obstáculos del mapa. Si es demasiado grande como la fila 6 I obtendrá demasiados giros al encontrarse cerca a un conjunto de obstáculos en forma curva.

## VII. Conclusiones

- El algoritmo funciona más rápido mapeando los obstáculos a la vez que avanza. Disminuye el tiempo de cálculo del algoritmo original ya que cálcula todos los obstáculos en cada iteración.
- El algoritmo aún necesita escoger un óptimo conjunto de parámetros como velocidades de traslación y rotación, es decir calcular un espacio de velocidades(dynamic window) apropiado para poder calcular el siguiente estado óptimo.a

## Referencias


[1] Intelligent Autonomous Driving Assistant to Avoid Obstacles and Collisions

[2] High-speed Navigation Using the Global Dynamic Window Approach Oliver Brock 0ussama.Khatib

[3] El enfoque de ventana dinamica para evitar colisiones - Dieter Fox y Wolfram Burgard y Sebastian Thrun

[4] PythonRobotics: a Python code collection of robotics algorithms. Atsushi Sakai https://arxiv.org/pdf/1808.10703.pdf.

[5] Santiago J. Cachumba, Pablo A. Briceño, Víctor H. Andaluz, Germán Erazo ICETC 2019: Proceedings of the 2019 11th International Conference on Education Technology and Computers https://doi.org/10.1145/3369255.3369296.