Estudio e implementación de un algoritmo de navegación autónoma para un auto eléctrico a escala

Ingeniería Electromecánica con orientación Electrónica

Micaela Carolina Jara Ten Kathen – 45.581

***Resumen*— El presente proyecto de grado se realiza en el contexto de investigador junior del CICTIA, Universidad Católica "Nuestra Señora de la Asunción" Campus Alto Paraná. Dicho proyecto fue realizado con el fin de formar los conocimientos respecto a la navegación autónoma, puesto que, se encuentra en crecimiento dentro del sector automotriz.**

Esta investigación comprende estudios sobre la navegación autónoma y los diferentes algoritmos que la componen; simulaciones en una plataforma de simulación seleccionada, para analizar el funcionamiento del algoritmo escogido; y la implementación del mismo en el vehículo eléctrico a escala gracias a un sistema embebido seleccionado.

Para el cumplimiento de los objetivos, se utilizó el concepto de reutilización de código y se personalizó para que funcione en la plataforma física "Aguara'i". La implementación se desarrolló en un periodo corto de tiempo y fue posible establecer valores, de la distancia anticipada, para que los errores, en la navegación, no superen los límites fijados.

***Palabras clave*— Algoritmos de navegación. Distancia anticipada. Navegación autónoma. Persecución pura.**

I. Nomenclatura

A2G: nombre del equipo formado para el evento Robocar Race 2018. Está constituido por estudiantes de diversos cursos de la carrera de ingeniería electromecánica y por un profesor tutor, todos pertenecientes a la casa de estudios mencionada

Aguara'i: nombre del auto eléctrico a escala autónomo.

CICTIA: Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada.

Robocar Race 2018: competencia internacional de vehículos autónomos a escala, llevada a cabo en la ciudad de San Pablo, Brasil, organizada por la Facultad de Tecnología de Santo André, Universidad Federal ABC y Robótica Paula Souza [1], en la cual, se ha participado en representación de la Universidad Católica "Nuestra Señora de la Asunción" Campus Alto Paraná.

II. Introducción

En estos últimos años, los vehículos autónomos tuvieron gran destaque en el mundo tecnológico, autos con nivel de autonomía 4 fueron presentados en la 89° edición del Salón del Automóvil de Ginebra [2], [3]. Muchas investigaciones e implementaciones han sido realizadas, desde empresas dedicadas a la fabricación de vehículos hasta empresas dedicadas a la fabricación de electrónicos y desarrollo de software.

Lo que pocos conocen, es que, en los años 80, un grupo de la Universidad Carnegie-Mellon (Pittsburgh, E.E.U.U.) desarrolló el proyecto NavLab 1, una Van que tenía la capacidad de conducir de forma autónoma utilizando visión y navegación para robot móvil en un ambiente al aire libre [4].

Para el desarrollo de un vehículo autónomo, distintas áreas de la ingeniería deben ser estudiadas. Uno de esos estudios es sobre la navegación autónoma, puesto que, sin ella, el vehículo no será capaz de conducir de forma autónoma.

En este proyecto, se hará una revisión bibliográfica de una serie de algoritmos de navegación con el fin de seleccionar uno de ellos e implementar en un vehículo eléctrico a escala.

III. Desarrollo del resumen de TFG

Este trabajo se dividirá en cinco grandes secciones, siendo los mismos:

Fundamentos de la Navegación Autónoma. En dicha sección, se realiza una pequeña introducción respecto a los autos autónomos, y se profundizará en navegación autónoma, explicando el concepto y los niveles de control de un robot móvil autónomo. Se presenta su clasificación según la forma en la que se navega y según el nivel de autonomía del vehículo. Además, se explica de manera básica cuatro algoritmos de navegación, dichos algoritmos fueron escogidos por sugerencia del tutor y por el gran uso que actualmente desempeña en el mundo tecnológico. Uno de esos algoritmos fue escogido para la implementación en el auto eléctrico a escala.

Algoritmo Persecución Pura. En esta sección, se explica de forma detallada el algoritmo escogido, así mismo, se presenta las simulaciones realizadas en el entorno de programación MATLAB, con el fin de comprender mejor su funcionamiento.

Diseño e Implementación del Algoritmo. En dicha sección, se explica el diseño del algoritmo y la teoría del código utilizado para la implementación en el vehículo eléctrico a escala.

Comprobación del Sistema. Se muestran, en esta sección, los resultados obtenidos en la implementación del algoritmo en el vehículo eléctrico a escala.

Consideraciones Finales. En esta última sección, se presenta

el análisis de costos, así como las conclusiones y se sugieren algunos cambios e ideas para futuros proyectos.

IV. FUNDAMENTOS DE LA NAVEGACIÓN AUTÓNOMA

A. Auto Autónomo

Según la definición de Techopedia [5], un auto autónomo es un vehículo capaz de conducir por sí solo, es decir, percibe su entorno y acorde a ello, navega hasta el destino deseado. El auto autónomo puede utilizar diversas tecnologías que le permitan percibir el entorno que le rodea, como un sistema de posicionamiento global (GPS), el cual permite conocer la posición absoluta del vehículo; una unidad de medición inercial (IMU), que consiste en una combinación de acelerómetros y giroscopios, permite conocer la posición relativa del auto; encoders, para calcular la velocidad; cámaras; radares; o un LiDAR.

Según el artículo publicado por [6], la conducción autónoma comenzó en la década de 1980. En ese periodo, la Universidad Carnegie Mellon (CMU, Pittsburgh, PA) presentó sus vehículos Navlab, que operaban en entornos estructurados [4], como parte de la Iniciativa de Computación Estratégica de la Agencia de Proyectos de Investigación Avanzados de Defensa [7]; así también, la Universidad de la Bundeswehr Munich (UniBw Munich, Neubiberg, Alemania) mostró resultados tempranos en la conducción de autopistas de alta velocidad [8].

Como menciona [6], en la actualidad, diferentes compañías realizan inversiones millonarias para desarrollar prototipos de vehículos autónomos que realicen diferentes funciones, tales como Toyota, Nissan, Nvidia, Uber, entre otros [9].

B. Navegación Autónoma

Según [10], la navegación autónoma es la capacidad de un robot, en el caso de este proyecto, del vehículo, de ir desde un punto inicial a un punto final, evitando los obstáculos que se encuentran en su camino.

Un robot móvil debe resolver en todo instante las preguntas ¿Dónde estoy?, ¿Dónde voy? y ¿Cómo llego? [11]. [12], comenta que la primera pregunta se resuelve con los sensores instalados en el robot y/o con sensores ubicados en el ambiente de operación. La segunda pregunta se responde con una estrategia global de navegación y la tercera pregunta se responde con una estrategia de navegación local, que puede corresponder a una incapacidad de experimentar cambios (estática) o incorporar capacidades deliberativas que optimicen la evasión de obstáculos (dinámica).

Para tener una mejor visión de la estructura de control para navegación de vehículo autónomos, se menciona el artículo [13], donde se organiza el control en forma de cascada, como se presenta en la Fig. 1. En el nivel más alto (nivel 4) se encuentra la planificación de la navegación y la generación de la trayectoria. Los algoritmos de control para seguimiento de trayectoria basados en los modelos cinemáticos están generalmente localizados en el nivel 3. El nivel 2 es el nivel de percepción y control, es el primer nivel donde existe control del movimiento del robot. Finalmente, en el nivel 1, se encuentran los sensores y actuadores, que son el hardware del robot.

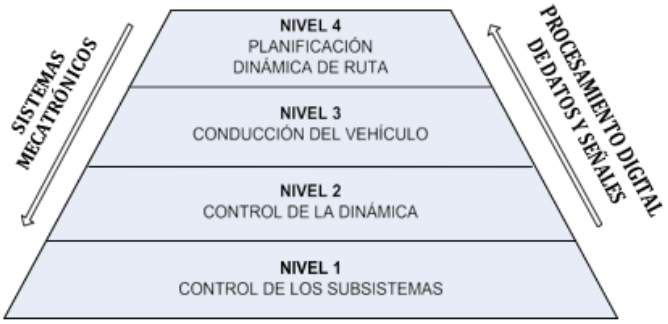

Fig. 1 Niveles de control de un Robot Móvil Autónomo.

1) Clasificación de la navegación autónoma

a) Según el tipo de navegación

La navegación puede realizarse de distintas maneras, en el artículo [14], se describen los tipos de navegación. En la Fig. 2 se resume la clasificación.

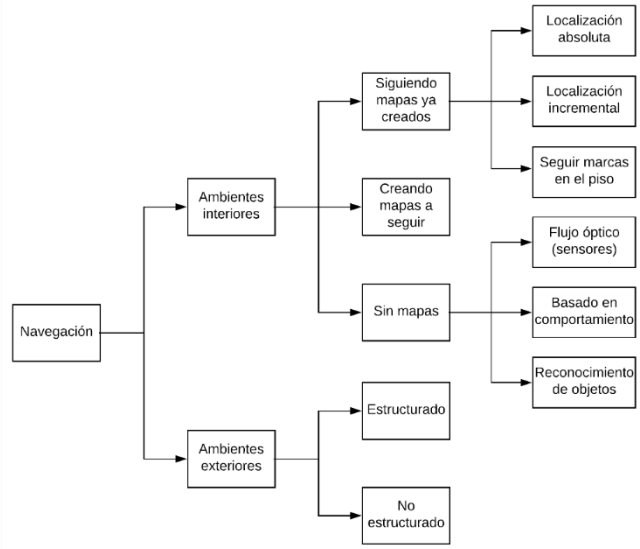

Fig. 2 Tipos de navegación en la robótica móvil.

En el artículo mencionado, [14], los tipos de navegación se dividen en dos grandes grupos, navegación en ambientes interiores y en ambientes exteriores, los mismos, se vuelven a dividir en diversos grupos.

Dentro de la *navegación en ambientes interiores*, se encuentran:

(1) Siguiendo un mapa ya creado

En él, el robot ya posee el mapa que debe seguir [14]. Sin embargo, se puede encontrar en situaciones donde: la posición inicial del robot es desconocida, *localización absoluta*; la posición inicial del robot puede ser conocido al menos aproximadamente, *localización incremental*; y la posición inicial del robot es conocida, *seguir marcas en el piso*.

(2) Creando mapas a seguir

Utilizan sensores para construir sus propios modelos geométricos o topológicos del entorno y luego utilizan estos modelos para la navegación [14].

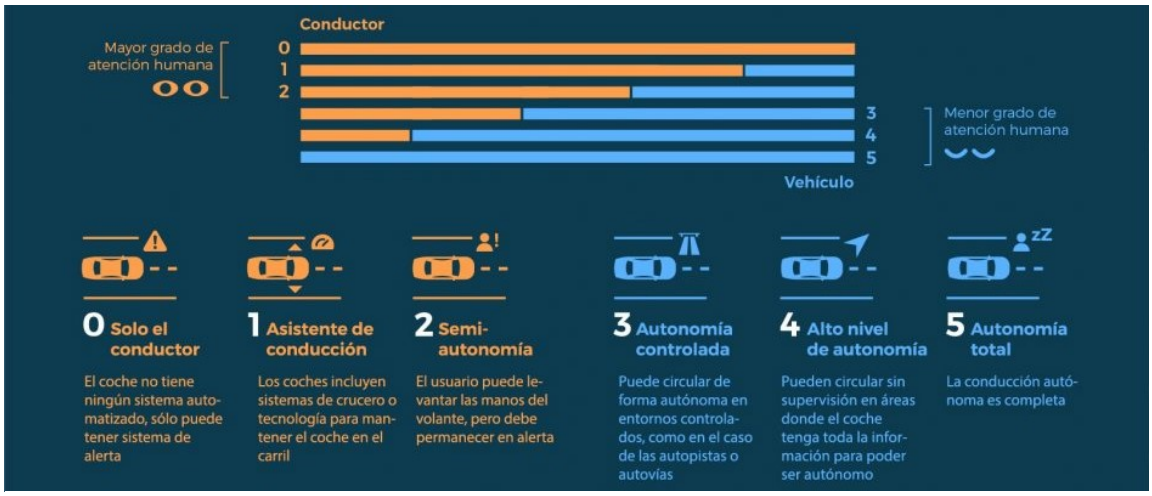

Fig. 3 Niveles de autonomía de un vehículo.

(3) Sin mapas

Estos son sistemas realizan la navegación sin ninguna representación explícita sobre el espacio donde se encuentran. Los mismos, recurren al reconocimiento de objetos encontrados en el entorno o al seguimiento de esos objetos mediante la generación de movimientos basados en observaciones visuales [14]. Estos sistemas se pueden dividir en: un sistema de flujo óptico que imita el comportamiento visual de las abejas, *flujo óptico* [15]; "memorizando" del entorno, *basado en comportamiento*; y utilizando un enfoque de navegación, *reconocimiento de objetos* [16] y [17].

Dentro del grupo de *navegación en ambientes exteriores* se presentan los ambientes:

(1) Estructurado

La navegación al aire libre en entornos estructurados requiere algún tipo de seguimiento, generalmente poseen modelos de ambientes simples, que contienen información como puntos de fuga, anchos de caminos y carreteras [14].

(2) No estructurado

Un entorno no estructurado, como definen [14], es un entorno al aire libre sin propiedades regulares que podrían percibirse y rastrearse para la navegación.

b) Según la autonomía

No todos los autos tienen el mismo nivel de conducción autónoma. Existe una organización que creó una escala de seis niveles de autonomía, en la cual, pueden ser divididos los vehículos, la Sociedad de Ingenieros Automotrices (SAE). La jerarquía se muestra en la Fig. 3 y es presentada en el estándar SAE J3016 [18].

Para clasificar el nivel de autonomía del vehículo, la SAE International explica que hay cuatro aspectos fundamentales:

1. Movimiento del vehículo.
2. Detección y respuesta ante objetos y eventualidades.
3. Respaldo de la conducción.
4. Condiciones específicas para el funcionamiento del sistema.

C. Algoritmos de Navegación

1) Navegación por estima (Dead Reckoning)

La navegación por estima es un sistema utilizado hace más de 20 años y es uno de los más simples. De acuerdo al artículo [19]:

> "El sistema de navegación por estima determina la ubicación actual utilizando cierta información de posición, trayectoria y velocidad durante un período de tiempo determinado. Este sistema de navegación proporciona al vehículo una posición estimada, puesto que los sensores que necesita son sencillos, los más utilizados son los encoders. Como el encoder mide básicamente la distancia de movimiento, es evidente que los errores del sensor tienen un efecto tanto en el rumbo como en la posición del robot móvil. Los errores son acumulativos y pueden dar lugar a un error de posición grave. Uno de los sistemas refinados de la navegación por estima es el sistema de navegación inercial. El sistema de navegación inercial consiste en sensores inerciales, como giroscopios, acelerómetros y algunos tipos de filtros. Para calcular la posición y la actitud, deben integrarse los datos del giroscopio y el acelerómetro. Por lo tanto, incluso errores muy pequeños en los sensores de inercia causarán un crecimiento ilimitado en la posición y la actitud. Por lo tanto, los errores de navegación del sistema de navegación por estima que utilizan encoders y sensores de inercia deben compensarse con otros mecanismos externos."

En la Fig. 4, se muestra el diagrama de flujo presentado en [20], donde se explica el funcionamiento del algoritmo a muy alto nivel. Se fija la distancia que se desea recorrer y se compara con la distancia viajada o recorrida, si la distancia viajada iguala o supera la distancia fijada, el algoritmo se detiene, caso contrario, el algoritmo continúa hasta que se cumpla la condición. La distancia recorrida se obtiene de los sensores.

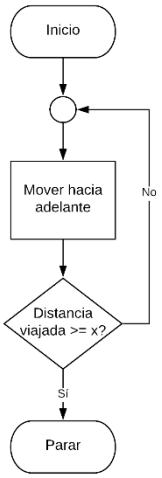

Fig. 4 Diagrama de flujo del algoritmo navegación por estima.

2) Persecución Pura (Pure Pursuit)

Conforme al artículo [21], la persecución pura es un algoritmo de seguimiento que funciona calculando la curvatura que el vehículo se moverá para que, desde su posición actual, vaya a la posición deseada. El objetivo del algoritmo es elegir una meta que se encuentra a una cierta distancia delante del vehículo en la ruta. El nombre persecución pura proviene de la analogía que se utiliza para describir este método, se tiende a pensar que el vehículo está persiguiendo un punto en la ruta que está a cierta distancia delante de él.

El autor mencionado en el párrafo anterior explica que persecución pura es un método para determinar geométricamente la curvatura que conducirá al vehículo a un punto de trayectoria elegido, denominado punto de meta. Este punto de meta es un punto en el camino que está a una distancia en frente de la posición actual del vehículo, esa "distancia delante de él o distancia anticipada" en inglés es conocido como *lookahead distance*. Básicamente, se construye un arco que une el punto actual y el objetivo, la longitud de arco construido es la distancia anticipada y actúa como la tercera restricción para determinar un arco único que une los dos puntos. A continuación, en la Fig. 5, el diagrama de flujo del algoritmo resumido por [21].

Para explicar específicamente como funciona este algoritmo, se recurre a la doctoral presentada por [22], en el MIT:

> "El enfoque de control tiene un solo parámetro ajustable, el parámetro que define un círculo virtual que rodea el vehículo. El punto de meta, mencionado anteriormente, es la intersección de este círculo con la ruta especificada. Luego de ello, el algoritmo de persecución pura especifica una curvatura deseada o una aceleración centrípeta deseada basada en la selección de una trayectoria de arco hacia ese punto de meta. El arco está restringido para ser tangencial al vector de velocidad que se encuentra en el origen del marco fijo del cuerpo, así como está restringido a pasar a través del origen del marco fijo del cuerpo y el punto de meta. Esto proporciona las condiciones necesarias para definir las coordenadas del centro del arco, así como el radio de curvatura."

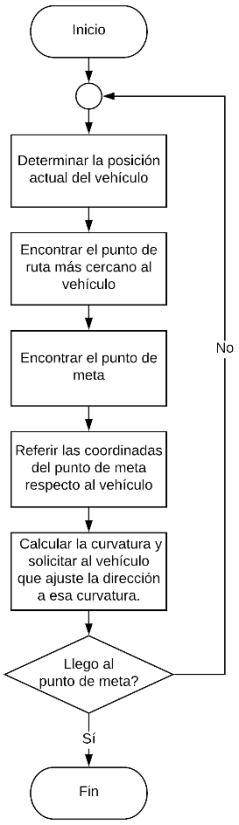

Fig. 5 Diagrama de flujo del algoritmo persecución pura.

3) Inteligencia Artificial

Desde el surgimiento de la inteligencia artificial en el siglo pasado, los estudios sobre este tema no cesaron. La aplicación de ella en los vehículos es uno de los temas actuales en el mundo.

La inteligencia artificial posee varias áreas como el algoritmo genético, redes neuronales, búsqueda heurística, entre otros [23], [24]. La navegación de un vehículo puede ser realizado dentro de una u otra rama. En este proyecto, se explicará acerca del algoritmo de búsqueda A* y redes neuronales, dichos algoritmos son muy utilizados dentro de la inteligencia artificial [25], [26].

a) Algoritmo de búsqueda A (A estrella o A star)*

El algoritmo de búsqueda A* utiliza una combinación de búsqueda heurística y búsqueda basada en la ruta más corta [27]. Este algoritmo es una extensión del algoritmo de Dijkstra, intenta reducir el número total de estados explorados incorporando una estimación heurística del costo para llegar a la meta desde un estado dado [28].

La fórmula mencionada por [28], para dicho algoritmo es:

$$f(v) = g(v) + h(v) \qquad (1)$$

Donde [27], definen que *h(v)* es la distancia heurística de la celda al estado objetivo y *g(v)* es la longitud de la ruta desde el estado inicial hasta el estado objetivo a través de la secuencia de celdas seleccionadas, esta secuencia termina en la celda evaluada actualmente. Cada celda adyacente a la celda alcanzada actualmente es evaluada por el valor *f(v)*. En otras palabras, *h(v)* es el costo heurístico de un nodo a otro, *g(v)* es el costo del camino de un nodo a otro y *f(v)* es el costo total. La celda con el

menor costo *f(v)* es escogida como la siguiente en la secuencia, es decir, es por esa celda que seguirá la ruta.

En la Fig. 6, se observa el pseudocódigo de alto nivel presentado por [29]. En ella, se muestra en forma de código lo que [27] han explicado en su artículo.

```
Put node_start in the OPEN list with f(node_start) = h(node_start) (initialization)
while the OPEN list is not empty {
  Take from the open list the node node_current with the lowest
       f(node_current) = g(node_current) + h(node_current)
  if node_current is node_goal we have found the solution; break
  Generate each state node_successor that come after node_current
  for each node_successor of node_current {
    Set successor_current_cost = g(node_current) + w(node_current, node_successor)
    if node_successor is in the OPEN list {
      if g(node_successor) ≤ successor_current_cost continue (to line 20)
    } else if node_successor is in the CLOSED list {
      if g(node_successor) ≤ successor_current_cost continue (to line 20)
      Move node_successor from the CLOSED list to the OPEN list
    } else {
      Add node_successor to the OPEN list
      Set h(node_successor) to be the heuristic distance to node_goal
    }
    Set g(node_successor) = successor_current_cost
    Set the parent of node_successor to node_current
  }
  Add node_current to the CLOSED list
}
if(node_current != node_goal) exit with error (the OPEN list is empty)
```

Fig. 6 Pseudocódigo del algoritmo A*.

b) Redes Neuronales

"El interés en la red neuronal se deriva del deseo de comprender los principios que conducen de alguna manera a la comprensión de las funciones básicas del cerebro humano y de construir máquinas capaces de realizar tareas complejas. Esencialmente, la red neuronal se ocupa de tareas cognitivas como el aprendizaje, la adaptación, la generalización y la optimización. De hecho, el reconocimiento, el aprendizaje, la toma de decisiones y la acción constituyen los principales problemas de navegación. Para resolver estos problemas se utilizan la lógica difusa y las redes neuronales. Mejoran las capacidades de aprendizaje y adaptación relacionadas con las variaciones en el entorno donde la información es cualitativa, inexacta, incierta o incompleta. El procesamiento de datos imprecisos o ruidosos, por las redes neuronales, es más eficiente que las técnicas clásicas, porque las redes neuronales son altamente tolerantes a los ruidos." [30]

De acuerdo a [31], una red neuronal, para la generación de trayectorias, cuenta con tres capas de neuronas: la de entrada, la oculta y la capa de salida. Las neuronas de entrada, simplemente envían la información del exterior, que es captado por los sensores, hacia las neuronas de la capa oculta. La capa oculta efectúa una expansión no lineal del espacio de entrada en un espacio "oculto" donde las clases son linealmente separables. La capa de salida realiza clasificación lineal y calcula la suma ponderada de las salidas que proporciona la capa oculta. [32] agregan que la red, en función de la información que capta a través de los sensores, tiene que proporcionar una acción de control adecuada para que el vehículo se desplace por el entorno sin colisionar.

[33] investigaron respecto a la Red Neuronal Profunda (Deep Neural Network) en vehículos autónomos. El DNN recibe la información de diferentes sensores, como la cámara, la detección de luz y el sensor de alcance (LiDAR), y el sensor IR (infrarrojo). Cada capa de un DNN consiste en una secuencia de unidades de computación individuales llamada neuronas. Las neuronas de las diferentes capas están conectadas entre sí a través de los bordes. Cada borde tiene un peso correspondiente (*θs* en la Fig. 7). Cada neurona aplica una función de activación no lineal en sus entradas y envía la salida a las neuronas subsiguientes como se muestra en la Fig. 7. La mayoría de los DNN existentes se entrenan con pendiente de gradiente utilizando la propagación hacia atrás [34]. Una vez terminado el entrenamiento, se puede usar un DNN para la predicción.

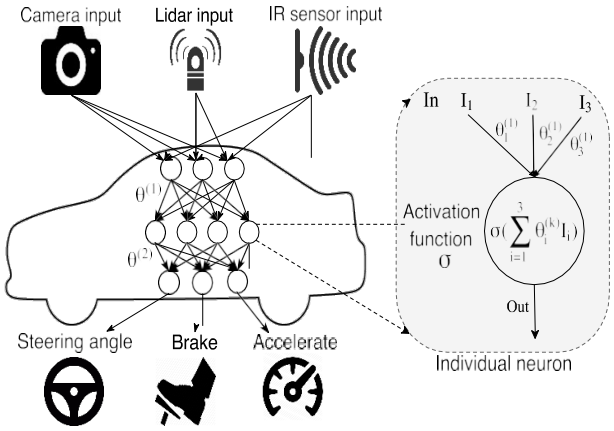

Fig. 7 Deep Neural Network de un vehículo autónomo.

D. Criterios de evaluación de algoritmos de navegación

Para seleccionar el algoritmo adecuado es importante realizar una tabla de comparación entre los algoritmos estudiados. A seguir, se presenta la Tabla 1, donde se comparan los cuatro algoritmos estudiamos previamente. Los criterios de evaluación fueron basados en las necesidades de este proyecto y el nivel dado a cada algoritmo es basado en los estudios realizados y en las experiencias obtenidas durante la investigación.

Los criterios son los siguientes:

1. Nivel de dificultad del algoritmo de navegación con respecto al aprendizaje del mismo.
2. Tiempo de aprendizaje que tomará el algoritmo de navegación.
3. Tiempo de programación o, en el caso de utilizar el algoritmo de terceros, el tiempo que llevará realizar las mejoras necesarias al algoritmo escogido.
4. Tiempo de entrenamiento del algoritmo de navegación.
5. Nivel de adaptación del algoritmo a pistas desconocidas (Cuando la ruta a seguir no es conocida de antemano).

TABLA 1
COMPARACIÓN DE LOS ALGORITMOS DE NAVEGACIÓN

Criterios	Navegación por estima	Persecución Pura	Algoritmo de Búsqueda A*	Redes Neuronales
1	Bajo	Medio	Alto	Alto
2	Bajo	Medio	Alto	Alto
3	Bajo	Medio	Alto	Alto
4	Medio	Bajo	Alto	Alto
5	Bajo	Alto	Medio	Alto

Teniendo en cuenta la Tabla 1 y todas las investigaciones desarrolladas, se escoge el algoritmo persecución pura para simular e implementar en el auto eléctrico a escala. Dicha elección se debe a la simplicidad que presenta el algoritmo y a la

versatilidad del mismo. En otras palabras, permite aprender e implementarlo en un periodo de tiempo corto, y puede ser implementado en pistas desconocidas sin mucha dificultad a diferencia del algoritmo navegación por estima, que es sencillo de aprender, pero a la hora de implementar no es conveniente, puesto que lleva tiempo programarlo en pistas desconocidas.

En el caso del algoritmo de búsqueda A* y las redes neuronales, es necesario un nivel de conocimiento más amplio, por ende, mayor periodo de tiempo. Debido al tiempo limitado que se poseía, no sería posible la implementación de uno de estos algoritmos.

V. Algoritmo Persecución Pura

A. Reseña Histórica

Al hablar del algoritmo persecución pura, uno pensaría que siempre fue utilizado en robots o vehículos, no obstante, en 1969, [35], lo investigaron como método para que los misiles puedan interceptar y perseguir a sus objetivos. En este proceso, el vector de velocidad del misil siempre se dirige hacia la posición objetivo instantánea [36].

En 1985, [37], aplicaron el algoritmo por primera vez como un método donde se calculaba la dirección necesaria para mantener el vehículo en la carretera [21]. [37] lo aplicaron en el Terragator, un robot de seis ruedas con dirección por deslizamiento que fue utilizado para la experimentación de la visión al aire libre (outdoor). Ellos han hecho la primera demostración de un vehículo autónomo, sin embargo, tuvieron varias fallas, esas fallas se debieron principalmente a errores en sus programas, procedimientos de calibración imprecisos y limitaciones del hardware, no limitaciones fundamentales de las técnicas utilizadas.

Unos años más adelante, basado en la idea de [37], [38], proponen el método de búsqueda pura para seguir caminos explícitos [39]. Ellos implementaron, en el proyecto NavLab, tres algoritmos de seguimiento de rutas, el primero es un enfoque de la teoría de control que usa errores en el eje horizontal y en el ángulo de dirección, llamado Teoría de Control, el segundo es el Polinomio de Quintic, intenta ajustar un polinomio adecuado al punto deseado, y el tercero usa arcos sucesivos para alcanzar el punto objetivo, el cual actualmente se conoce como Persecución Pura [38]. Las pruebas desarrolladas de todos estos algoritmos mostraron que el método de persecución pura era el método más robusto y confiable [21].

[21] presenta en su artículo algunos problemas de implementación del algoritmo persecución pura. Desde entonces, la estrategia de búsqueda pura se ha utilizado en muchas aplicaciones para el seguimiento explícito de rutas, incluida la navegación interior y exterior [40], [38].

[39] explican que, a mediados de los años 90, [41] y [42] investigaron acerca de las condiciones de estabilidad del algoritmo. [41] estudió los efectos de los retrasos en el tiempo asociados con el procesamiento visual en las siguientes líneas rectas, mientras que, [42] analizaron la estabilidad para el seguimiento de rutas explícitas con curvatura constante, teniendo en cuenta los retrasos en la computación, la comunicación y los actuadores en el bucle de control.

[40] introdujo un controlador difuso para la supervisión de los parámetros en tiempo real, lo cual permitió desarrollar vehículos no tripulados a alta velocidad con un controlador de seguimiento puro con supervisión difusa [43].

Resumiendo lo expuesto por [36], en 2003, resuelven el problema de que el vehículo está lejos del camino al crear un punto de meta virtual en la distancia de anticipación [44]; en 2011, [45], creó el controlador MPC que se ejecuta en línea para rastrear una ruta planificada, tenía la capacidad de evitar obstáculos, a pesar de la precisión que posee este controlador, el resultado no fue tan bueno como se esperaba; en 2012, [46], propusieron el uso del controlador PID adaptativo para rastrear rutas predefinidas. Para culminar, una de las modificaciones más recientes del algoritmo tradicional es el algoritmo de persecución pura con función polinomial de 2do grado, los resultados experimentales muestran que el algoritmo es más preciso que el algoritmo tradicional al mejorar 54.54% [47].

B. Derivación teórica

Como se mencionó anteriormente, el algoritmo persecución pura es un método para calcular geométricamente el arco necesario para que el vehículo vaya al punto de trayectoria deseado. El método es simple, intuitivo, fácil de implementar. En síntesis, el objetivo del algoritmo es elegir esa "distancia anticipada" adecuada [47].

La distancia anticipada es la propiedad principal de ajuste del controlador Persecución Pura. La distancia delante de él o distancia de anticipación es la distancia que debe recorrer el robot desde la ubicación actual para calcular los comandos de velocidad angular [48]. En la Fig. 8 se muestra el robot y el punto de anticipación. En esta figura, se puede observar de forma gráfica el funcionamiento del algoritmo persecución pura, en ella, la ruta real no coincide con la línea directa entre los puntos de referencia por la propiedad de la distancia anticipada [48].

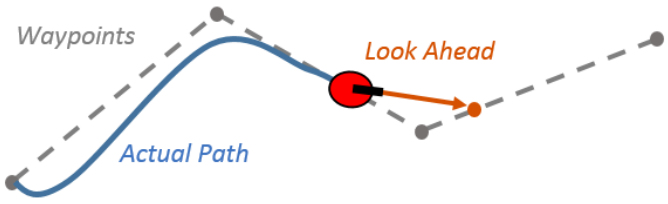

Fig. 8 Distancia anticipada.

1) Geometría del algoritmo

Para comprender mejor sobre cómo funciona el algoritmo y la distancia anticipada, se muestra la Fig. 9, en ella, se observa el vehículo con los ejes del sistema de coordenadas, x e y.

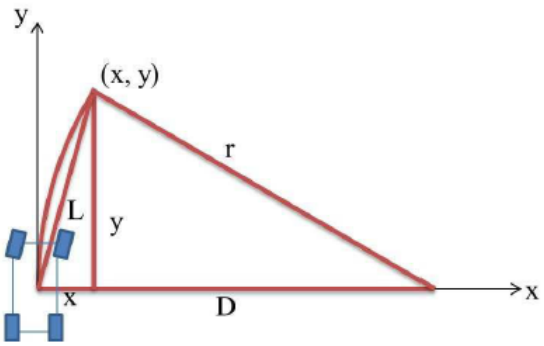

Fig. 9 Geometría del algoritmo.

El punto (*x; y*) es un punto que se encuentra a una distancia anticipada del vehículo. Esa distancia es *L*, y es la longitud del cable del arco que conecta el origen con el punto (*x; y*). El radio de curvatura del arco está representado por la letra *r* [21]. Se tiene por objetivo calcular la curvatura del arco, para ello, [21] presenta las siguientes relaciones:

$$x^2 + y^2 = L^2 \tag{2}$$

$$D^2 + y^2 = r^2 \tag{3}$$

$$x + D = r \tag{4}$$

De las ecuaciones (2), (3) y (4):

$$r^2 - 2rx + x^2 + y^2 = r^2 \tag{5}$$

$$r = \frac{L^2}{2x} \tag{6}$$

Por lo tanto, se tiene que la curvatura del arco es:

$$\gamma = \frac{1}{r} = \frac{2x}{L^2} \tag{7}$$

Es decir, al elegir una distancia de anticipación y al calcular el error de trayectoria *x*, se puede calcular la curvatura requerida para que el vehículo realice la trayectoria requerida [47].

2) Distancia anticipada

Para elegir la distancia anticipada se debe tener en cuenta que este parámetro incide en la forma en que el vehículo rastrea la ruta. Los objetivos principales son recuperar la ruta y mantenerla. Una pequeña distancia anticipada hará que el vehículo recupere rápidamente el camino entre los puntos de la ruta, sin embargo, el robot sobrepasará el camino y oscilará a lo largo del camino deseado como se muestra en la Fig. 10 [48].

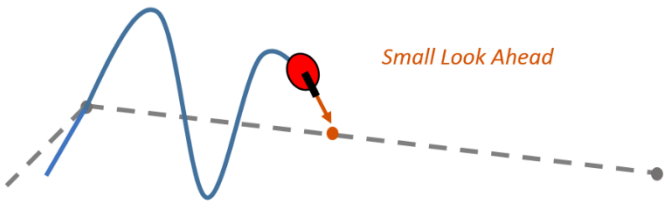

Fig. 10 Distancia anticipada pequeña.

Por otro lado, si se escoge una distancia anticipada mayor, las oscilaciones a lo largo del camino se reducirán. No obstante, esto puede causar curvaturas más grandes cerca de las esquinas [48], dicho de otro modo, observando la Fig. 11, la recuperación del camino no se dará de forma rápida.

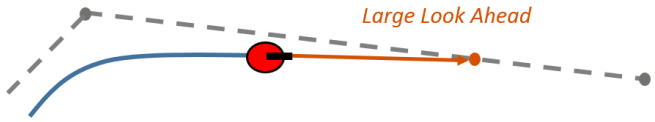

Fig. 11 Distancia anticipada grande.

C. Implementación

El método en sí es bastante sencillo, como lo es la implementación. Los únicos problemas reales de implementación se encuentran en decidir cómo manejar la información de ruta (comunicación, gráficos, actualizar la ruta con nueva información del planificador) [21].

1) Representación del camino

Como explica [21], el camino se representa con un conjunto de puntos discretos que son almacenados en la memoria del robot o vehículo. Según estudios del mismo, normalmente, el conjunto de puntos contiene las siguientes informaciones:

- Localización X.
- Localización Y.
- Ángulo de dirección.
- Curvatura del camino en este punto.
- Distancia (a lo largo de una línea recta) de este punto al principio del camino.

Estos datos pueden variar dependiendo de la forma que se diseñe el algoritmo de navegación. Un ejemplo sería que los puntos *x* e *y* pueden darse en el marco global o en el marco del robot.

2) Algoritmo de búsqueda

Para la implementación del algoritmo en sí, se toma como base los pasos propuestos por [21], mencionados en la Fig. 5. Esos pasos son:

1. Determinar la posición actual del vehículo.
2. Encontrar el punto de ruta más cercano al vehículo.
3. Encontrar el punto de meta.
4. Referir las coordenadas del punto de meta respecto al vehículo.
5. Calcular la curvatura y solicitar al vehículo que ajuste la dirección a esa curvatura.
6. Actualizar la posición del vehículo.

Cuando, al actualizar la posición del vehículo, esta coordenada sea igual al punto de meta, el algoritmo será terminado.

D. Simulación

El rápido avance de los conocimientos en el área de la robótica obliga a los ingenieros a instruirse de manera constante en este ámbito. Emplear MATLAB para estos casos, trae consigo una gran ventaja.

Para esta experiencia, fue utilizado el toolbox Robotics System Toolbox y un código proporcionado por The MathWorks, Inc.

Las simulaciones se llevan a cabo con el fin de comprender mejor el funcionamiento del algoritmo estudiado. Para ello, se simuló como un robot navega por diversas trayectorias utilizando la propiedad distancia anticipada de dicho algoritmo. En el mismo, se variaron los valores de velocidad y distancia anticipada, así como las trayectorias a seguir.

1) Procedimientos

Como ya ha sido mencionado, para la simulación se utilizó el código de [49], en el cual, se presenta el comportamiento del algoritmo persecución pura. A dicho código, se le adjunto el mapa deseado. Los pasos a seguir fueron los siguientes:

1. Definir las condiciones iniciales.
2. Configuración del robot.
3. Controlador de seguidor de ruta.
4. Funcionamiento del controlador persecución pura.

2) Resultados

Las simulaciones se basaron en dos mapas específicos, el primero es una pista que fue utilizada para pruebas de implementación del auto y el segundo es la pista donde se desarrolló la competencia Robocar Race 2018. Estas simulaciones fueron realizadas con distintos valores de velocidad y distancias anticipadas, no serán mostradas todas las imágenes tomadas y si una tabla al final de cada sección de los mapas con los análisis respectivos.

Fueron escogidos estos mapas con el objetivo de comparar los resultados de las simulaciones con los de la implementación del algoritmo en el vehículo a escala.

a) Primer mapa: Pista Tacurú Pucú

Dicha pista fue implementada en el tinglado del Colegio Nacional Tacurú Pucú, ubicado en la ciudad de Hernandarias. El formato de la pista fue diseñado con el fin de probar el funcionamiento del algoritmo en trayectos rectos y en curvas cerradas.

A continuación, se muestran las simulaciones realizadas con una velocidad de 2 m/s y distancias anticipadas diferentes, Fig. 12, Fig. 13 y Fig. 14. En la Fig. 12, la distancia anticipada tiene un valor de 1.8 metros, con él, se puede observar que el robot realiza el recorrido pasando cerca de los puntos, las curvaturas cerca de las esquinas no son grandes, son difícilmente notorias.

En la Fig. 13, el valor de la distancia anticipada es 2.5 metros. En él, se observa que el robot no pasa por algunos de los puntos fijados en las curvas del mapa. En cambio, en la Fig. 14, donde la distancia anticipada es 3.5 metros, prácticamente solo en los trayectos rectos el robot pasa por los puntos. Esto se debe a que al ser mayor la distancia anticipada, más se aleja de los puntos en las curvas, en otras palabras, disminuye la distancia que debe recorrer.

En la Fig. 15, se muestra un ejemplo de una distancia anticipada pequeña, 0.5 metros. En dicha figura, el robot realiza el trayecto punto por punto, se pueden observar pequeñas oscilaciones mientras el robot pasa de un punto a otro. En las próximas figuras serán más perceptibles las oscilaciones del robot con pequeñas distancias anticipadas.

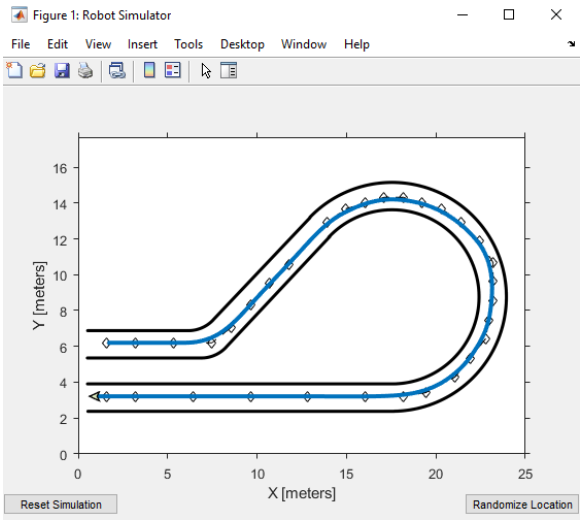

Fig. 12 Velocidad = 2 m/s. Distancia Anticipada = 1.8 m

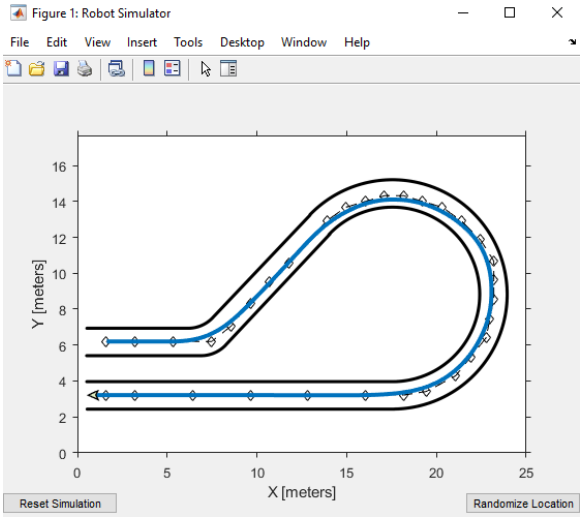

Fig. 13 Velocidad = 2 m/s. Distancia Anticipada = 2.5 m

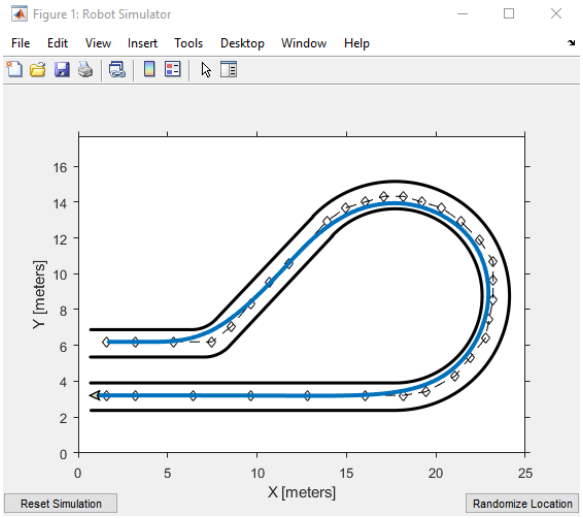

Fig. 14 Velocidad = 2 m/s. Distancia Anticipada = 3.5 m

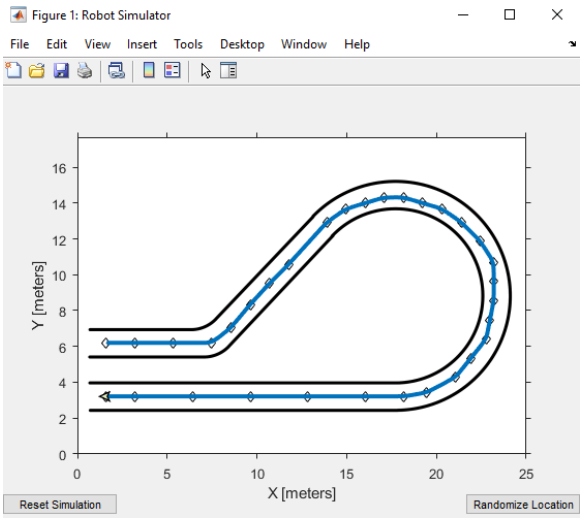

Fig. 15 Velocidad = 3 m/s. Distancia Anticipada = 0.5 m

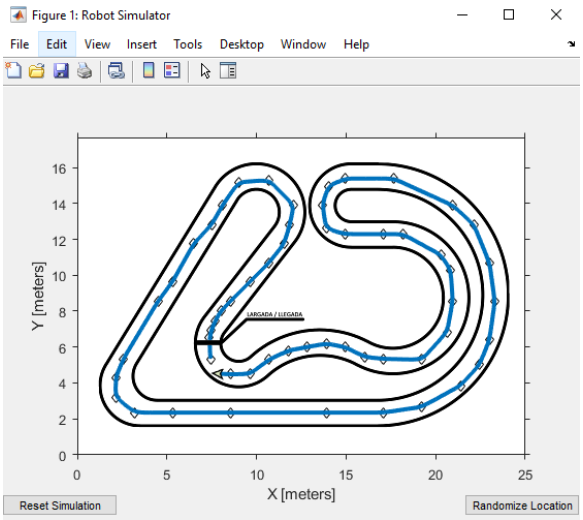

Fig. 16 Velocidad = 2.5 m/s. Distancia Anticipada = 0.5 m

(1) Análisis

El funcionamiento del algoritmo fue óptimo utilizando valores entre los rangos especificados para cada valor de velocidad en la Tabla 2, es decir, el robot se mantenía dentro de los márgenes de la pista. Sin embargo, por debajo del valor mínimo, el algoritmo se comportaba como seguidor de líneas o tenía pequeñas oscilaciones. Por encima del valor máximo, el robot realizaba curvaturas más grandes, los cuales, en una situación real, harían que el robot salga de la pista.

TABLA 2
RANGO DE VALORES PARA LA DISTANCIA ANTICIPADA. MAPA TACURÚ PUCÚ

Velocidad (m/s)	Rango de valores de la distancia anticipada (m)
2	1.8 – 3
3	2.2 – 3.2
4	2.5 – 3.2

b) Segundo mapa: Pista Robocar Race 2018

A continuación, se presentan las simulaciones desarrolladas con una velocidad de 2.5 m/s con diferentes valores de distancia anticipada.

En la Fig. 16, como en la Fig. 15, posee una distancia anticipada pequeña, igual a 0.5 m y el efecto es el mismo, se producen pequeñas oscilaciones mientras el robot pasa de un punto a otro. A diferencia de la Fig. 15, en este se puede observar mejor el efecto producido por pequeñas distancias anticipadas.

En la Fig. 17, la distancia anticipada es de 2.3 m, en ella, se muestra claramente el funcionamiento de la propiedad distancia anticipada, en las curvas el robot "acorta" el camino.

En la Fig. 18 se muestra un ejemplo de una distancia anticipada grande. En él, el robot se desplaza fuera de los márgenes de la pista, puesto que la distancia es mayor, dicho robot "acorta" mucho más de lo debido el camino.

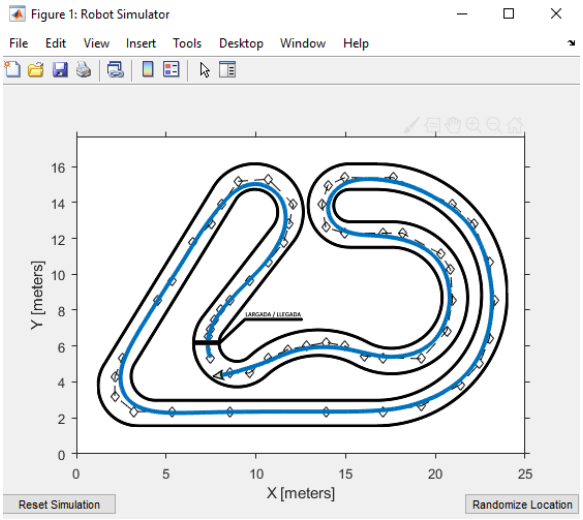

Fig. 17 Velocidad = 2.5 m/s. Distancia Anticipada = 2.3 m

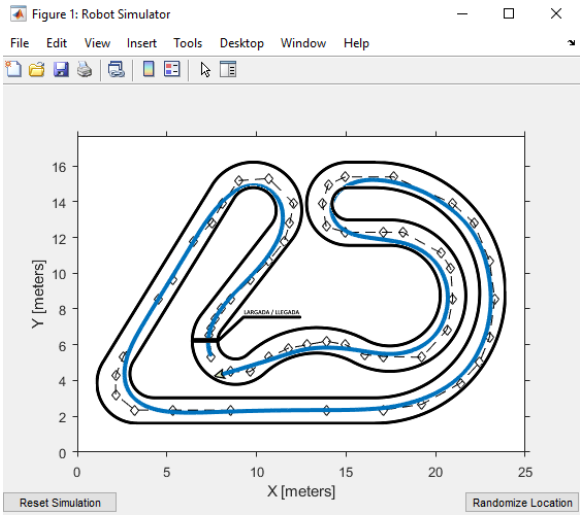

Fig. 18 Velocidad = 2.5 m/s. Distancia Anticipada = 3 m

(1) Análisis

El robot se mantenía dentro de la pista utilizando valores entre los rangos especificados para cada valor de velocidad en la Tabla 3. Los comentarios son los mismos que con el primer mapa, por

debajo del valor mínimo, el algoritmo se comportaba como seguidor de líneas o tenía pequeñas oscilaciones y por encima del valor máximo, el robot realizaba curvaturas más grandes, los cuales, en una situación real, harían que este salga de la pista.

TABLA 3
RANGO DE VALORES PARA LA DISTANCIA ANTICIPADA. MAPA ROBOCAR RACE 2018

Velocidad (m/s)	Rango de valores de la distancia anticipada (m)
2.5	1.7 – 2.3
3	1.8 – 2.4
4	1.9 – 2.5

c) Comparación de los resultados

Realizar simulaciones es muy importante para comprender mejor el funcionamiento del algoritmo. Observando la Tabla 2 y la Tabla 3, se puede ver que el intervalo en el rango de valores en la tabla del primer mapa es mayor que en la tabla del segundo mapa, principalmente para las dos primeras velocidades. Esto se debe al formato de la pista, puesto que, en el segundo mapa, las curvas son más pronunciadas y las coordenadas de los puntos de la ruta de puntos se encuentran más cercas unas de otras. Con esto, se concluye que el rango de valores de distancia anticipada posee pequeñas variaciones entre una pista y otra.

VI. DISEÑO E IMPLEMENTACIÓN DEL ALGORITMO

La implementación del algoritmo persecución pura fue realizada a través de los productos de National Instruments, el sistema embebido myRIO y el software LabVIEW.

El código a ser implementado en el myRIO es una modificación del algoritmo del Team 1712 [50]. La adaptación del algoritmo a la plataforma física "Aguara'i" fue hecha por los miembros del equipo A2G; Jesús Franco, Alfredo Galeano, Sebastian Reckzeigel, Jorge Bareiro y Micaela Jara, autora del artículo.

El código programado por el Team 1712 posee muchas secciones que no son necesarias para este proyecto. Debido a ello, en la Fig. 19, se presenta el diagrama de flujo del algoritmo adaptado que fue implementado en el auto eléctrico a escala.

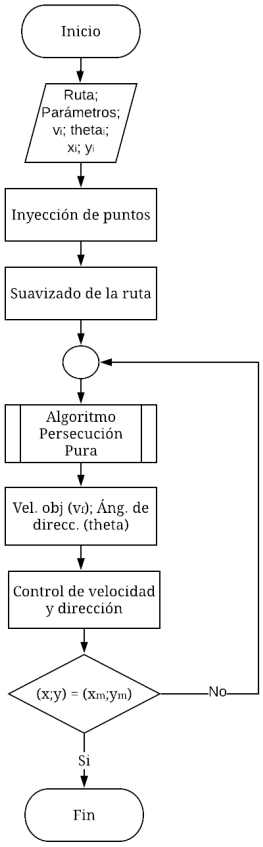

Fig. 19 Diagrama de flujo del código fuente del algoritmo adaptado

Una vez que el código es iniciado, se fijan los debidos parámetros, que serán explicados posteriormente, y la ruta de puntos, que servirá de guía para la navegación. Para la ruta de puntos, pueden ser recolectados y guardados los puntos por donde debe pasar el vehículo, en un documento de texto, con un VI. Luego, se extraen los puntos de dicho documento para la ruta de puntos.

Para que el algoritmo funcione correctamente, los puntos de la ruta deben encontrarse cerca unos de otros y estos deben componer curvas suaves para que el auto no gire de manera brusca, para ello, son utilizados los módulos de inyección de puntos y suavizado de la ruta. Para la inyección de puntos, es utilizado el código fuente del Team 1712 [50] y para el suavizado, el código fuente de [51].

El algoritmo persecución pura está compuesto por los módulos presentados en la Fig. 20. Los datos que ingresan de manera constante al algoritmo son los valores x, y, velocidad actual y ángulo de dirección actual del auto, con estos datos, junto con los parámetros mencionados anteriormente, el algoritmo calcula la velocidad objetivo y el ángulo de dirección para llegar al siguiente punto de la ruta. Estos valores son enviados al control de velocidad y dirección, el cual envía las señales a los actuadores del auto. Esto se repite hasta que las coordenadas del punto actual ($x; y$) sean iguales a las coordenadas del punto de meta ($x_m; y_m$). Cumplida esta condición, el código se detiene.

Fig. 20 Diagrama de flujo del código fuente del algoritmo persecución pura

El código del algoritmo de persecución pura posee los siguientes procesos:

- Calcular la distancia entre los puntos, donde, la distancia del punto actual es igual a la distancia del punto anterior más la distancia entre el punto anterior y el actual. La distancia es utilizada para calcular la curvatura y la velocidad en cada punto.
- Calcular la curvatura de la trayectoria en un punto para ajustar la velocidad del vehículo. La curvatura puede ser hallada encontrando el radio del círculo que interseca el punto y los dos puntos a cada lado, para luego aplicar la ecuación (7).
- Calcular la velocidad objetivo en cada punto. Para calcular esa velocidad, se tiene en cuenta la curvatura en ese punto, puesto que, si se encuentra en una curva cerrada, el algoritmo disminuye la velocidad del auto; y la velocidad máxima definida para la ruta.
- Hallar el punto más cercano. Para hallar dicho punto se debe calcular la distancia entre el mismo y todos los demás puntos y tomar la más pequeña.
- Calcular el punto anticipado, que es el punto en la ruta que se encuentra a una distancia anticipada del auto. El punto anticipado es el punto de intersección del círculo que tiene como radio la distancia anticipada y como centro la ubicación del robot, y los segmentos de la ruta [50].
- Corregir el ángulo de dirección. Una de las modificaciones agregadas fue la corrección del ángulo de dirección. Esto se debe a que el ángulo proveniente de la fusión del magnetómetro con el giroscopio se orientaba en sentido horario.

A. Parámetros del algoritmo

- A: controla la distancia entre la línea original y los puntos suaves, es decir, qué tan cerca de la línea original se desea que estén los puntos suaves.
- B: controla las curvas que compondrán el camino, dicho de otra forma, qué curvas se considerarán para el camino.
- Tolerancia: define la región dentro de la cual se tienen en cuenta todas las coordenadas
- Ubicación Inicial: coordenadas del punto inicial del robot.
- Espacio: distancia deseada entre los puntos, es el mismo valor puesto en PointSpacing.
- Velocidad Max.: máxima velocidad al cual el vehículo irá.
- Aceleración Max.: máxima aceleración al cual el vehículo irá.
- ¿En Retroceso?: si el vehículo irá en retroceso, a diferencia de los demás parámetros, ¿En Retroceso? es un control booleano.
- Espacio: es el valor de la distancia deseada entre los puntos, debe ser mismo valor establecido en la inyección de puntos.
- Dist. Anticipada: es el valor de la distancia anticipada.
- Veloc./Curva: conversión de la curvatura de un punto en el camino a la velocidad máxima del robot en este punto.
- Dist. al Final: distancia a la que el robot viajará a una velocidad más lenta al final del recorrido.
- Vel. al Final: la velocidad a la cual irá el robot al final del recorrido.
- Desaceler. Max.: máximo desaceleración que el vehículo puede tener.

B. Interfaz de usuario

En la Fig. 21 se muestra la interfaz que el usuario observa cuando ingresa al VI. En él, se puede acceder a la opción de ayuda, donde se explica cómo se puede poner en marcha la navegación.

Para poner en marcha la navegación se deben seguir los siguientes pasos:

1. Seleccionar el documento de texto que contiene la ruta de puntos.
2. Establecer los parámetros de velocidad, distancia y aceleración.
3. Clicar el botón Parámetros. En él, se tiene la opción de mantener los valores por defecto o modificar los valores de Espacio, A, B y Tolerancia.
4. Una vez definido todos los parámetros, se habilita la opción de iniciar la navegación.
5. La navegación puede ser detenida clicando en Salir.
6. Se puede observar el recorrido en tiempo real en el gráfico Trayecto.

En la Fig. 22 y Fig. 23 se muestran las ventadas de Ayuda, en él, además de explicar los pasos a seguir, se explican los parámetros que deben ser definidos para la puesta en marcha.

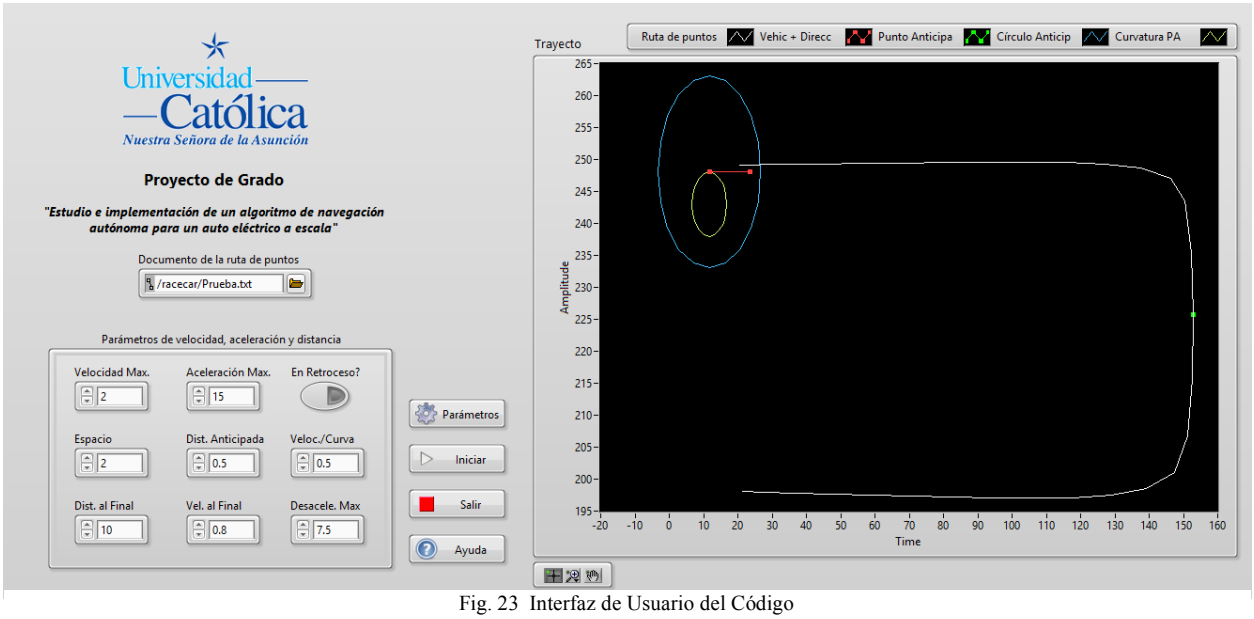

Fig. 23 Interfaz de Usuario del Código

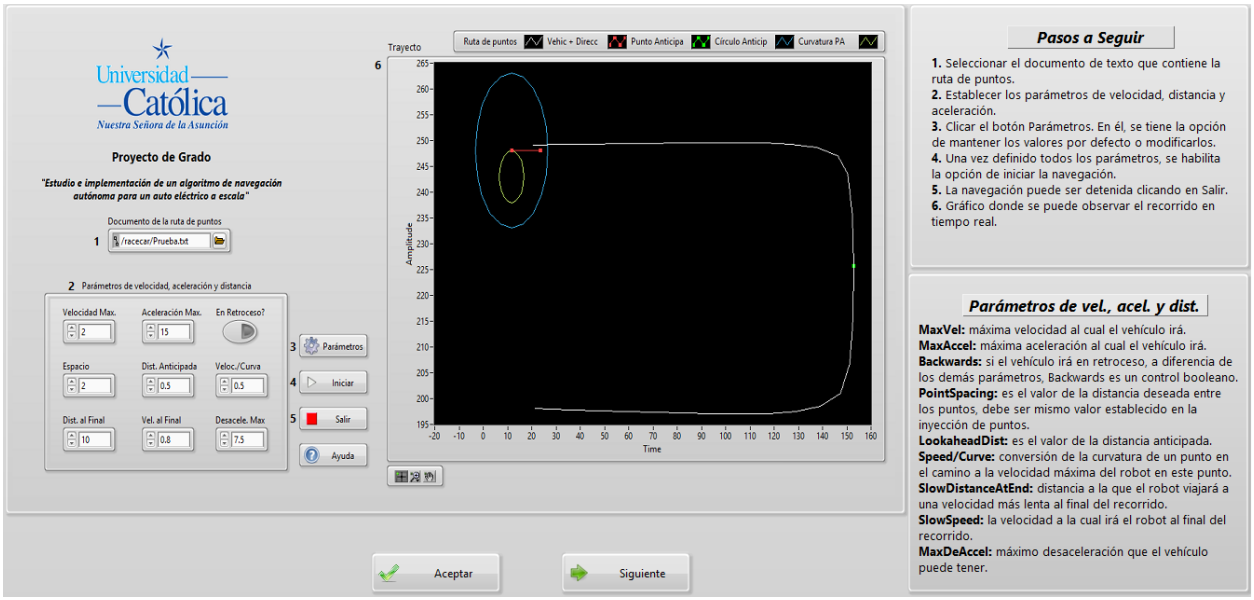

Fig. 21 Interfaz de Ayuda. Parte 1

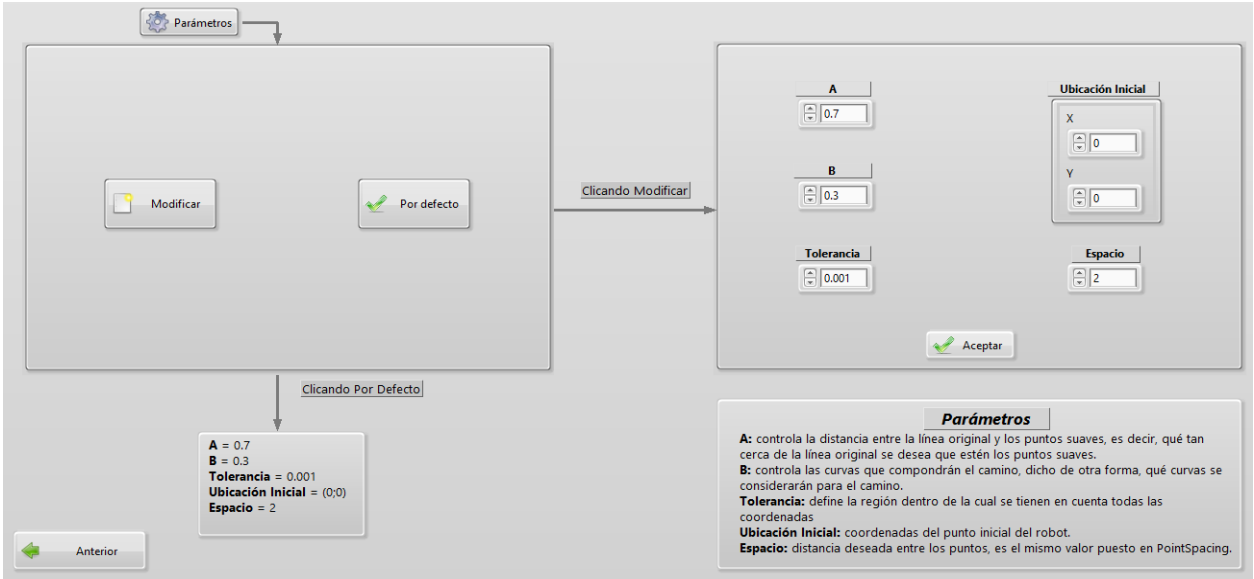

Fig. 22 Interfaz de Ayuda. Parte 2

C. Diagrama de bloques

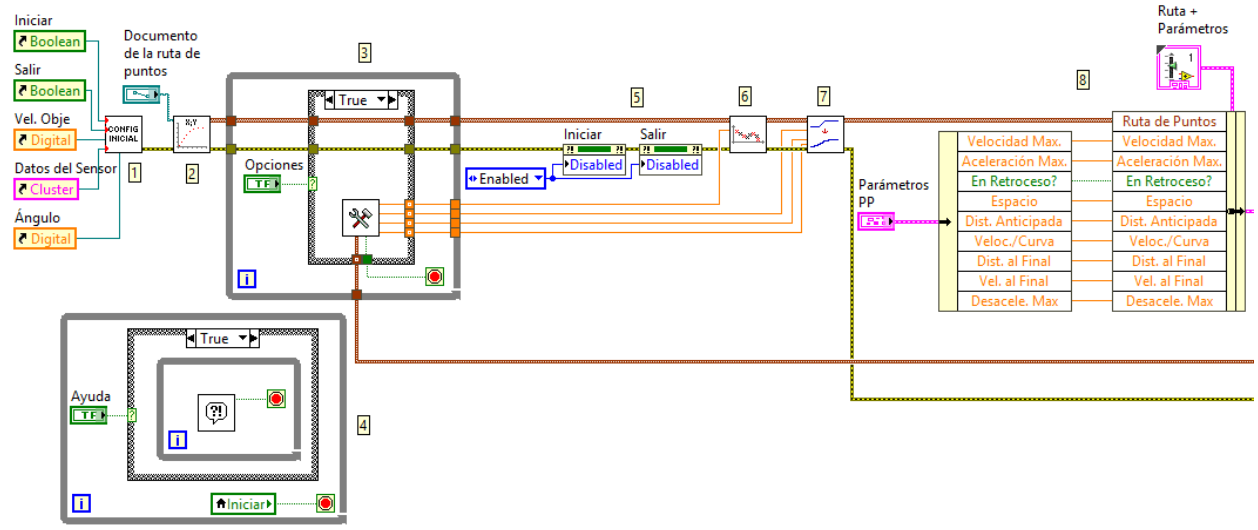

Fig. 24 Código fuente del proyecto Algoritmo Persecución Pura. Parte 1

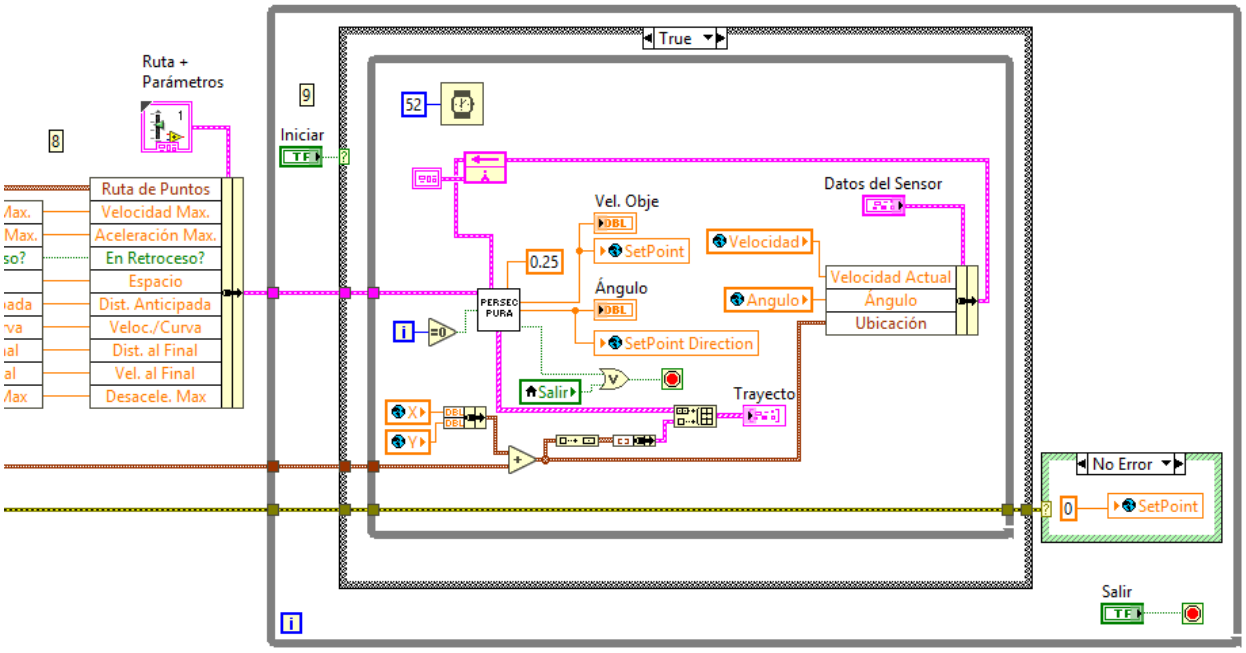

Fig. 25 Código fuente del proyecto Algoritmo Persecución Pura. Parte 2

En la Fig. 24 y Fig. 25 se muestra el código fuente del proyecto del algoritmo de navegación. A continuación, se explica cada parte del mismo:

1. Se realizan las configuraciones iniciales para la visualización de la interfaz.
2. Se extraen las coordenadas (*x; y*) del documento de texto.
3. Se fijan los valores de Espacio, A, B y Tolerancia.
4. Permite visualizar la ventana de Ayuda.
5. Una vez fijados los valores mencionados en el punto 3, habilita los botones de Iniciar y Salir.
6. Se inyectan los puntos en la ruta de puntos a la distancia fijada en Espacio.
7. Realiza el suavizado del camino.
8. Trasmite los parámetros de velocidad, aceleración y distancia, y la ruta de puntos.
9. Al clicar Iniciar, se pone en marcha la navegación del auto, en el subVI Persecución Pura se realizan los cálculos ya mencionados anteriormente. Se crea un cluster de las variables de entrada, X, Y. Velocidad y Ángulo y son enviadas al subVI Persecución Pura. Las variables de salida, la velocidad objetivo (SetPoint) y el ángulo de dirección (SetPoint Direction), son transmitidos por variables globales al control de velocidad y dirección para que el auto pueda realizar el recorrido. Asimismo, son transmitidos los datos necesarios para la visualización del recorrido en el gráfico.

Una vez concluido el recorrido, si no hay error, el algoritmo manda el valor de 0 al SetPoint (velocidad objetivo), esto fue añadido para que el auto se detenga al culminar.

El valor 0.25 que se muestra encima del subVI es el ancho entre las ruedas del vehículo, dicha información es utilizado para el cálculo de las velocidades de la rueda izquierda y derecha, sin embargo, como esta implementación no es necesario el cálculo de las mismas, ese dato puede ser omitido.

VII. Comprobación del Sistema

Las implementaciones se llevaron a cabo en tres mapas diferentes, dos de ellas son las mismas utilizadas para la simulación y el tercer mapa es un segmento de la ciclovía de la costanera de Hernandarias.

En la Fig. 26 se observa el vehículo eléctrico a escala modificado donde fue implementado el algoritmo persecución pura.

Fig. 26 Auto eléctrico a escala modificado "Aguara'i"

Se llevaron a cabo distintas pruebas con valores diferentes de todos los parámetros especificados en la sección anterior. Con ellos, se pudieron concluir que, los parámetros que más tuvieron impacto cuando sus valores eran modificados fueron: la velocidad y la distancia anticipada.

En la implementación, al variar los valores de Espacio, A, B, y Tolerancia, el vehículo realizaba recorridos similares, sin diferencias notorias. En otras pruebas, se fijaron dichos parámetros con los valores que se muestran en la Tabla 4 y se variaron los demás parámetros mostrados en dicha tabla. La respuesta del vehículo a dichas variaciones fue óptima. Al no probar con altas velocidades, las variaciones realizadas en los parámetros de Aceleración Max. y Desacele. Max. no se observaron grandes diferencias. Sin embargo, los cambios fueron notorios al modificar los valores de los parámetros Veloc./Curva, Dist. al Final y Vel. Final, el vehículo mantenía la velocidad o reducía en las curvas dependiendo del valor, así como cerca del final del trayecto.

A diferencia de la implementación, los parámetros de Espacio, A, B, Tolerancia, Aceleración Max. y Desacele. Max. fueron probadas en la simulación en LabVIEW, donde se obtuvieron las respuestas necesarias para analizar el comportamiento del algoritmo. Como los puntos de la ruta fijada se encontraban cercas unos de otros, la variación del parámetro Espacio, no tuvo mucho impacto en el comportamiento del algoritmo. Así mismo, la variación del parámetro Tolerancia, tampoco afectó el comportamiento del mismo. Sin embargo, variando los parámetros A y B, se modificaba el formato de la ruta, es decir, suavizaba sus curvas, habiendo hasta, aproximadamente, un metro de diferencia entre la ruta real y la suavizada (A = 0.1; B = 0.9).

Al establecer velocidades altas, el funcionamiento de los parámetros Aceleración Max. y Desacele. Max fue claramente observado. Al fijar aceleraciones altas, el auto rápidamente alcanzaba la Velocidad Max., sin embargo, con aceleraciones bajas, el auto necesitaba mayor tiempo para alcanzar la velocidad fijada.

Debido a lo mencionado en los párrafos anteriores, para los resultados presentados en este capítulo, se decidió fijar los parámetros con los siguientes valores:

Tabla 4
Valores de los parámetros para la implementación

Parámetros	Valores
Espacio	2
A	0.7
B	0.3
Tolerancia	0.001
Aceleración Max.	15
Veloc./Curva	1
Dist. al Final	10
Vel. al Final	0.8
Desacele. Max.	7.5

A. Primer mapa: Pista Tacurú Pucú

La recolección de datos fue realizada recorriendo por la línea central de la pista, fijando una velocidad en el control de velocidad y dirigiendo el vehículo de manera manual, es decir, la dirección se realizaba con el control remoto.

Fueron realizadas varias pruebas, en las cuales se probaron distintos valores de velocidad y distancia anticipada. En la Tabla 5 se presentan los valores, para cada velocidad, de la distancia anticipada con las que el recorrido realizado de manera autónoma no tenía un error mayor a 1.5 metros. Para visualizar esa aproximación, se fijaron marcas en el piso de la ruta de puntos inicial, esas marcas servían de referencia.

Tabla 5
Valores para la distancia anticipada. Mapa Tacurú Pucú

Velocidad (m/s)	Distancia anticipada (m)
2	1.7 – 1.8
2.5	2.2 – 2.3
3	2.7
3.5	3 – 3.2
4	3.8
4.5	4 – 4.3

A diferencia de la simulación en MATLAB, en la implementación fue difícil conseguir un rango de valores para la distancia anticipada. Una variación de 0.1, en varios casos, ya afectaba en gran manera al desempeño del vehículo.

Si el valor de la distancia anticipada era menor al valor establecido en la Tabla 6, el auto realizaba el recorrido con oscilaciones y el error, en tramos del recorrido, superaba los 1.5 metros. Así como, si el valor de la distancia era mayor, el trayecto hecho por el vehículo, se alejaba de las marcas fijadas en el suelo, más bien, acortaba en gran medida el camino superando el valor límite del error.

En esta pista, se observó como el auto desaceleraba en las curvas gracias al parámetro Vel./Curva del algoritmo.

B. Segundo mapa: Pista Robocar Race 2018

Para este segundo mapa, Los valores de velocidad fueron dos básicamente, puesto que el algoritmo fue implementado en la competencia y no fueron probadas más velocidades. Las pruebas se llevaron a cabo en conjunto con otros equipos, por lo tanto, implementar altas velocidades era difícil.

En dicha pista, ya se obtuvo un rango de valores para la distancia anticipada, puesto que, se fijó como límite de error 2.5 metros. Como en la pista anterior, los datos se recabaron recorriendo la línea central de la pista, la misma tenía un ancho de 5 metros. Los valores que están en negrita en la Tabla 6, fueron los que menos errores generaron, los mismos, oscilaban entre 1.5 y 2 metros, debido a las interferencias en el magnetómetro.

TABLA 6
VALORES PARA LA DISTANCIA ANTICIPADA. MAPA ROBOCAR RACE 2018

Velocidad (m/s)	Distancia anticipada (m)
2.5	1.9 – **2.3**
3	2.3 – **2.5**

C. Tercer mapa: Pista Costanera de Hernandarias

En esta pista pudo ser probado el control de velocidad, siendo implementado en un segmento de la ciclovía de la costanera donde posee desnivel. Debido a ello, el auto tiende a ir más rápido por la inercia. Sin embargo, como al inicio se fija un valor máximo para la velocidad, el algoritmo no permite al auto superar dicha velocidad, en consecuencia, el auto desacelera para ajustarse al valor fijado inicialmente.

En la Tabla 7, se muestra los resultados obtenidos en esta pista. Así como en la pista de Tacurú Pucú, no se pudo determinar un rango de valores para la propiedad Distancia Anticipada. El límite de error era de 1 metro.

TABLA 7
VALORES PARA LA DISTANCIA ANTICIPADA. MAPA COSTANERA DE HERNANDARIAS

Velocidad (m/s)	Distancia anticipada (m)
2	1.7
2.5	1.8
3	2.2
4	2.5
5	2.8

VIII. CONSIDERACIONES FINALES

A. Conclusiones

un algoritmo de navegación autónomo para un auto eléctrico a escala. Para el propósito, se realizó una revisión bibliográfica respecto a la navegación autónoma y de diversos algoritmos de navegación, tales como los algoritmos: navegación por estima, persecución pura, algoritmo de búsqueda A* y redes neuronales. Dicho estudio tuvo como objetivo seleccionar el algoritmo que mejor se adecue a las necesidades y condiciones de este proyecto, donde, la condición primordial era poder implementarlo en un periodo corto de tiempo.

El algoritmo escogido fue persecución pura, dicho algoritmo fue simulado en un entorno de programación e implementado en el auto eléctrico a escala debido a la simplicidad que presenta el algoritmo y a la versatilidad del mismo.

La reutilización de códigos de terceros es un factor importante a resaltar, puesto que, es de gran ayuda a la hora de realizar cualquier proyecto porque se reduce el tiempo de programación, lo cual permite implementarlo después de un periodo corto de tiempo. Utilizando este concepto, fueron realizados las simulaciones y la implementación del algoritmo, puesto que, para la simulación, fue utilizado el código presentado por [49] y, para la implementación, el código fuente tuvo como base los códigos creados por: [50] y [51].

El algoritmo fue simulado en el entorno de programación MATLAB, con el objetivo de conocer mejor el funcionamiento del mismo, fueron simulados dos trayectos utilizando diferentes valores de velocidad y distancia anticipada. Así mismo, para llevar a cabo la implementación, se utilizó el ambiente de programación LabVIEW y el sistema embebido myRIO. Las modificaciones en el código base fueron hechas por Jesús Franco, Alfredo Galeano, Sebastian Reckzeigel, Jorge Bareiro y la autora de este libro de tesis.

Se observó el excelente desempeño del algoritmo persecución pura en las simulaciones realizadas, tanto en MATLAB como en LabVIEW, y en la implementación en el auto eléctrico a escala. Para cada pista que fue implementado dicho algoritmo, el valor del error era distinto, esto se debe a la diferencia en el ancho de esas pistas. Debido a ello, en algunos casos, no se pudieron obtener rango de valores de la distancia anticipada para cada velocidad. Sin embargo, si fue posible, hallar al menos un valor, con el cual, el recorrido realizado por el vehículo, cumpliera con las condiciones determinadas.

Cabe resaltar, como complemento, que la participación del equipo A2G, en representación de la Universidad Católica "Nuestra Señora de la Asunción" Campus Alto Paraná, en la competencia Robocar Race 2018, llevada a cabo en la ciudad de San Pablo, Brasil, fue un éxito.

En dicha competencia, se lograron los siguientes resultados:

- Segundo puesto en la calidad del TDP (Technical Description Paper).
  El artículo de descripción técnica será publicado en la prestigiosa revista JPAUT (Journal of Production and Automation) de Brasil. Los artículos presentados en el concurso fueron evaluados por el Prof. Dr. Edson Kitani (FATEC Santo André) y el Prof. Dr. Luiz Celiberto Jr. (UFABC).
- Cuarto puesto en la competencia (Categoría libre).

Además de la competencia Robocar Race 2018, el proyecto del auto autónomo "Aguara'i" fue presentado en la competencia organizada por la National Instruments, Labview Student Design Competition. En la mencionada competencia son presentados proyectos estudiantiles que utilicen los productos de dicha empresa.

Debido al impacto que tuvo el proyecto "Aguara'i" en la Universidad Católica y en la competencia Robocar Race 2018, además de la participación en la competencia Labview Student Design Competition, la National Instruments ha invitado al equipo principal de A2G al evento NIWeek 2019, véase invitación en el Apéndice F. Dicho evento tiene una duración de cuatro días, en ese tiempo son desarrollados sesiones técnicas, conferencias, pueden ser observados los proyectos desarrollados por ingenieros de la empresa o asociados a él, entre otros.

B. Sugerencias para trabajos futuros

- Realizar navegación absoluta, junto con la navegación relativa, agregando un GPS al vehículo autónomo a escala.
- Simular en realidad aumentada el algoritmo persecución pura en un vehículo de tamaño real.
- Implementar un algoritmo de navegación distinto en el auto eléctrico a escala.

IX. Agradecimientos

Se agradece al apoyo del equipo A2G y de las siguientes instituciones: Universidad Católica "Nuestra Señora de la Asunción", Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada, Fundación Parque Tecnológico Itaipu – Paraguay, National Instruments.

X. Bibliografía

[1] «Robocar Race,» 2018. [En línea]. Available: http://www.roborace.com.br/.

[2] «e.GO - Mover,» 2019. [En línea]. Available: https://www.gims.swiss/premieres/swiss-premieres/mover. [Último acceso: 24 marzo 2019].

[3] «RINSPEED - microSNAP,» 2019. [En línea]. Available: https://www.gims.swiss/premieres/european-premieres/microsnap. [Último acceso: 24 marzo 2019].

[4] C. Thorpe, M. H. Hebert, T. Kanade y S. A. Shafer, «Vision and navigation for the Carnegie-Mellon Navlab,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 10, nº 3, pp. 362-373, 1988.

[5] «Autonomous Car,» s/f. [En línea]. Available: https://www.techopedia.com/definition/30056/autonomous-car. [Último acceso: 20 noviembre 2018].

[6] Y. Li, M. Díaz, S. Morantes y Y. Dorati, «Vehículos autónomos: Innovación en la logística urbana,» *Revista de Iniciación Científica,* vol. 4, nº 1, pp. 34-39, 2018.

[7] C. Thorpe, M. H. Hebert, T. Kanade y S. A. Shafer, «Toward autonomous driving: The CMU Navlab. part i-perception,» *IEEE Intelligent Systems,* nº 4, pp. 31-42, 1991.

[8] E. D. Dickmanns y A. Zapp, «Autonomous High Speed Road Vehicle Guidance by Computer Vision1,» *IFAC Proceedings Volumes,* vol. 20, nº 5, pp. 221-226, 1987.

[9] «Autónomo, inteligente y conectado: el vehículo del mañana, una realidad en 2018,» 23 January 2018. [En línea]. Available: https://www.hibridosyelectricos.com/articulo/tecnologia/autonomo-inteligente-conectado-vehiculo-manana-realidad-2018/20180122143227016969.html. [Último acceso: 13 diciembre 2018].

[10] G. Azkune, «Navegación autónoma,» 12 Noviembre 2011. [En línea]. Available: https://cuentos-cuanticos.com/2011/11/12/navegacion-autonoma/.

[11] J. J. Leonard y H. F. Durrant-Whyte, «Mobile robot localization by tracking geometric beacons,» *IEEE Transactions on robotics and Automation,* vol. 7, nº 3, pp. 376-382, 1991.

[12] H. A. Valdés Tapia, Navegación Local de Robots Móviles en ambientes desconocidos utilizando Programación Orientada a Comportamientos (Tesis de Grado), Universidad Católica del Norte, 2013.

[13] J. Wit, C. D. Crane III y D. Armstrong, «Autonomous ground vehicle path tracking,» *Journal of Robotic Systems,* vol. 21, nº 8, pp. 439-449, 2004.

[14] G. N. DeSouza y A. C. Kak, «Vision for mobile robot navigation: A survey,» *IEEE transactions on pattern analysis and machine intelligence,* vol. 24, nº 2, pp. 237-267, 2002.

[15] J. Santos-Victor, G. Sandini, F. Curotto y S. Garibaldi, «Divergent stereo for robot navigation: Learning from bees,» de *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, 1993.

[16] D. Kim y R. Nevatia, «Recognition and localization of generic objects for indoor navigation using functionality,» *Image and Vision Computing,* vol. 16, nº 11, pp. 729-743, 1998.

[17] D. Kim y R. Nevatia, «Symbolic navigation with a generic map,» *Autonomous Robots,* vol. 6, nº 1, pp. 69-88, 1999.

[18] SAE Internacional, Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, SAE Internacional, 2014.

[19] K. Park, H. Chung y J. G. Lee, «Dead reckoning navigation for autonomous mobile robots,» *IFAC Proceedings Volumes,* vol. 31, nº 3, pp. 219-224, 1998.

[20] C. D'Souza, «Student Competition: Mobile Robotics Training, Part 1: Controlling Robot Motion,» 27 Jun 2017. [En línea]. Available: https://www.mathworks.com/videos/student-competition-mobile-robotics-training-part-1-controlling-robot-motion-1498677744306.html. [Último acceso: 15 enero 2019].

[21] R. C. Coulter, «Implementation of the pure pursuit path tracking algorithm (No. CMU-RI-TR-92-01),» Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[22] S. F. Campbell, Steering control of an autonomous ground vehicle with application to the DARPA urban challenge (Tesis Doctoral), Massachusetts Institute of Technology. Cambridge, 2007.

[23] W. Zhang, R. Dechter y R. E. Korf, «Heuristic search in artificial intelligence,» *Artificial Intelligence,* vol. 129, nº 1-2, pp. 1-4, 2001.

[24] F. Mejía Peñafiel, «Algoritmos genéticos,» 21 Septiembre 2010. [En línea]. Available: http://nando1-utb.blogspot.com/p/algoritmos-geneticos.html.

[25] G. A. Bekey y K. Y. Goldberg, Neural Networks in robotics, New York: Springer Science+Business Media, 1993.

[26] A. Patel, «Introduction to A*,» s/f. [En línea]. Available: http://theory.stanford.edu/~amitp/GameProgramming/ASTarComparison.html. [Último acceso: 11 enero 2019].

[27] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico y L. Jurišica, «Path planning with modified a star algorithm for a mobile robot,» *Procedia Engineering,* vol. 96, pp. 59-69, 2014.

[28] S. M. LaValle, Planning algorithms, Cambridge: Cambridge university press, 2006.

[29] T. Abiy, H. Pang, B. Tiliksew, K. Moore y J. Khim, «A* Search,» s.f. [En línea]. Available: https://brilliant.org/wiki/a-star-search/. [Último acceso: 16 enero 2019].

[30] D. Janglová, «Neural Networks in Mobile Robot Motion,» *International Journal of Advanced Robotic Systems,* vol. 1, nº 1, pp. 15-22, 2004.

[31] M. A. Baquero Suárez y J. C. Cudris Cantillo, Generación de trayectorias y procesamiento digital de imágenes para la navegación de un robot móvil (Tesis de Grado), Universidad Santo Tomás. Bucaramanga, 2007.

[32] H. G. Acevedo y C. A. Castañeda, «Estudio comparativo de tres técnicas de navegación para robots móviles,» *Revista UIS Ingenierias,* vol. 6, nº 1, pp. 77-88, 2007.

[33] Y. Tian, K. Pei, S. Jana y B. Ray, «Deeptest: Automated testing of deep-neural-network-driven autonomous cars,» de *Proceedings of the 40th International Conference on Software Engineering*, 2018.

[34] D. E. Rumelhart, G. E. Hinton y R. J. Williams, «Learning representations by back-propagating errors,» *nature,* vol. 323, nº 6088, p. 533, 1986.

[35] L. Scharf, W. Harthill y P. Moose, «A comparison of expected flight times for intercept and pure pursuit missiles,» *IEEE Transactions on Aerospace and Electronic Systems,* nº 4, pp. 672-673, 1969.

[36] M. Samuel, M. Hussein y M. B. Mohamad, «A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle,» *International Journal of Computer Applications,* vol. 135, nº 1, pp. 35-38, 2016.

[37] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker y T. Kanade, «First Results in Robot Road-Following,» de *IJCAI*, 1985.

[38] O. Amidi y C. E. Thorpe, «Integrated mobile robot control,» de *Mobile Robots V*, 1991.

[39] J. Morales, J. L. Martínez, M. A. Martínez y A. Mandow, «Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2D laser scanner,» *EURASIP Journal on Advances in Signal Processing,* vol. 2009, pp. 1-10, 2009.

[40] A. Ollero, A. García-Cerezo y J. L. Martínez, «Fuzzy supervisory path tracking of mobile reports,» *Control Engineering Practice,* vol. 2, nº 2, pp. 313-319, 1994.

[41] K. N. Murphy, «Analysis of robotic vehicle steering and controller delay,» de *Fifth International Symposium on Robotics and Manufacturing (ISRAM)*, 1994.

[42] A. Ollero y G. Heredia, «Stability analysis of mobile robot path tracking,» de *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, 1995.

[43] A. Rodríguez-Castaño, G. Heredia y A. Ollero, «Analysis of a GPS-based fuzzy supervised path tracking system for large unmanned vehicles,» *IFAC Proceedings Volumes,* vol. 33, nº 25, pp. 125-130, 2000.

[44] K. Petrinec, Z. Kovacic y A. Marozin, «Simulator of multi-AGV robotic industrial environments,» de *Industrial Technology, 2003 IEEE International Conference on*, 2003.

[45] M. A. Abbas, Non-linear model predictive control for autonomous vehicles (Tesis Doctoral), University of Ontario Institute of Technology. Oshawa, 2011.

[46] P. Zhao, J. Chen, Y. Song, X. Tao, T. Xu y T. Mei, «Design of a control system for an autonomous vehicle based on adaptive-PID,» *International Journal of Advanced Robotic Systems,* vol. 9, nº 2, 2012.

[47] W. J. Wang, T. M. Hsu y T. S. Wu, «The improved pure pursuit algorithm for autonomous driving advanced system,» de *Computational Intelligence and Applications (IWCIA), 2017 IEEE 10th International Workshop on*, 2017.

[48] The MathWorks, Inc, «Pure Pursuit Controller,» s/f (a). [En línea]. Available: https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html. [Último acceso: 7 febrero 2019].

[49] The MathWorks, Inc, «Path Following for a Differential Drive Robot,» s/f (c). [En línea]. Available: https://www.mathworks.com/help/robotics/examples/path-following-for-differential-drive-robot.html. [Último acceso: 26 febrero 2019].

[50] Team 1712, «Implementation of the Adaptive Pure Pursuit Controller,» 2018. [En línea]. Available: https://www.chiefdelphi.com/t/paper-implementation-of-the-adaptive-pure-pursuit-controller/166552. [Último acceso: 09 marzo 2019].

[51] K. Harrilal, «Smooth Path Planner,» 2014. [En línea]. Available: https://github.com/KHEngineering/SmoothPathPlanner. [Último acceso: 10 marzo 2019].