



UNIVERSIDAD CATÓLICA
“NUESTRA SEÑORA DE LA ASUNCIÓN”
FACULTAD DE CIENCIAS Y TECNOLOGÍA

INGENIERÍA ELECTROMECÁNICA CON ORIENTACIÓN ELECTRÓNICA

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

Micaela Carolina Jara Ten Kathon

Hernandarias, abril de 2019

UNIVERSIDAD CATÓLICA
“NUESTRA SEÑORA DE LA ASUNCIÓN”
FACULTAD DE CIENCIAS Y TECNOLOGÍA

INGENIERÍA ELECTROMECÁNICA CON ORIENTACIÓN ELECTRÓNICA

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

Micaela Carolina Jara Ten Kathon

Tutor: Lic. Gregorio Ariel Guerrero Moral

Hernandarias

2019

Micaela Carolina Jara Ten Kathon

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

Proyecto Fin de Carrera presentado como requisito parcial para optar al título de Ingeniería Electromecánica con Orientación Electrónica. Facultad de Ciencias y Tecnología, Universidad Católica “Nuestra Señora de la Asunción”

Tutor: Lic. Gregorio Ariel Guerrero Moral

Hernandarias

2019

Jara Ten Kathen, Micaela Carolina. (2019); Estudio e implementación de un algoritmo de navegación autónoma para un auto eléctrico a escala. Hernandarias, Universidad Católica. 153p.

Tutor: Lic. Gregorio Ariel Guerrero Moral.

Defensa de Proyecto de Fin de Carrera

Palabras claves: Algoritmos de navegación. Distancia anticipada. Navegación autónoma. Persecución pura.

A Dios, Jesús y a la Virgen María, por brindarme la fuerza necesaria para cumplir esta meta y por las oportunidades recibidas que me ayudaron a crecer en sabiduría y como persona.

A mis padres, mis hermanos y mis abuelos, quienes me apoyaron incondicionalmente en todos estos años de estudio.

Agradecimientos

Agradezco a Dios, Jesús y la Virgen María, por la fortaleza y perseverancia que me brindaron en todo momento, por encontrar en Ellos la paz y tranquilidad en los momentos difíciles.

Al profesor tutor, por su sabiduría, paciencia y predisposición para ayudar en todos estos meses de trabajo, principalmente por la confianza puesta en mí.

A mis padres, por su inmenso amor, por sus ejemplos, por apoyarme en todas mis decisiones y por motivarme a seguir adelante, son lo mejor que Dios me pudo haber dado.

A mis hermanos y a mis abuelos, por apoyarme y estar siempre en todo momento, por brindarme alegría en momentos difíciles.

A mi novio, por el constante apoyo en todos estos años de mi formación universitaria, por estar siempre presente y por ayudarme en todo lo que estuviera a su alcance.

A mis amigos y a mis compañeros que me motivaron, ayudaron y estuvieron siempre para mí, otorgándome su apoyo.

A los profesores de esta casa de estudio, por confiar en mi capacidad, por brindarme todo su apoyo y por transmitirme las enseñanzas que fueron la base para emprender este propósito, en especial a la profesora Tania Melgarejo, por ser un hombro amigo en todos estos años.

A la Universidad Católica “Nuestra Señora de la Asunción” Sede Alto Paraná, por el apoyo dado en todos estos años.

A todos los miembros del grupo A2G, al Parque Tecnológico Itaipu (PTI-PY) y a la National Instruments, por el apoyo recibido y por toda la ayuda brindada para la participación en la competencia Robocar Race 2018. Véase Apéndice G.

Y finalmente, a todas aquellas personas que no nombre pero que de una o de otra manera me ayudaron a cumplir esta meta.

“La mayor gloria no es no caer nunca, sino levantarse siempre”

Nelson Mandela

Resumen

El presente proyecto de grado sobre el estudio e implementación de un algoritmo de navegación autónoma para un vehículo eléctrico a escala, se realiza en el contexto de investigador junior del Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada de la 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, el cual, actualmente, está en fase de crecimiento dentro del sector automotriz mundial.

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

Para el cumplimiento de los objetivos planteados se utilizó el concepto de reutilización de código y se personalizó para que funcione en la plataforma física “Aguara’í”. De esta manera, se pudo concluir que el desempeño del algoritmo seleccionado, algoritmo de navegación persecución pura, fue muy efectivo y satisfactorio. La implementación se desarrolló en un periodo corto de tiempo y los trayectos recorridos durante la navegación autónoma fueron similares a las rutas establecidas, es decir, fue posible establecer valores, de la distancia anticipada, para que los errores no superen los límites fijados.

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

Abstract

The present project of degree on the study and implementation of an autonomous navigation algorithm for a scale electric vehicle, is made in the context of junior researcher of the Center for Research in Sciences, Technology and Advanced Innovation of the Catholic University "Nuestra Señora de la Asunción" Campus Alto Paraná. This project was carried out in order to train the knowledge regarding autonomous navigation, which is currently in the growth phase of the global automotive sector.

This research includes studies on autonomous navigation, concepts, classifications and the different algorithms that compose it, as well as simulations in a simulation platform selected to analyze the operation of the chosen algorithm and the implementation of the same in the electric vehicle at scale thanks to a selected embedded system.

For the fulfillment of the proposed objectives, the concept of code reuse was used and it was customized to work on the "Aguara'i" physical platform. In this way, it was possible to conclude that the performance of the selected algorithm, pure pursuit navigation algorithm, was very effective and satisfactory. The implementation was developed in a short period of time and the routes traveled during the autonomous navigation were similar to the established waypoints, that is, it was possible to establish values, of the lookahead distance, so that the errors do not exceed the set limits.

Keywords: Autonomous navigation. Navigation algorithms. Pure pursuit. Lookahead distance.

Tabla de Contenidos

Agradecimientos.....	v
Resumen.....	vii
Abstract.....	viii
Tabla de Contenidos	ix
Tabla de Figuras	xiii
Lista de Tablas	xvi
Lista de Abreviaturas	xvii
Introducción	1
Planteamiento del Problema	5
Justificación	7
Marco Teórico	8
Capítulo 1. Fundamentos de la Navegación Autónoma	8
1.1. Auto autónomo.....	8
1.1.1. Reseña histórica	8
1.1.2. Sensores	9
1.2. Navegación autónoma.....	10
1.2.1. Clasificación de la navegación autónoma.....	13
1.3. Sistema de coordenadas	19
1.4. Sistema de navegación inercial	23
1.5. Conclusión del capítulo.....	23
Capítulo 2. Algoritmos de Navegación	24
2.1. Navegación por estima (Dead Reckoning).....	24
2.2. Persecución pura (Pure pursuit)	27
2.3. Inteligencia artificial	29
2.3.1. Algoritmo de búsqueda A* (A estrella o A star)	29

2.3.2. Redes neuronales	32
2.4. Criterios de evaluación de algoritmos de navegación.....	34
2.5. Conclusión del capítulo.....	34
Capítulo 3. Algoritmo Persecución Pura	36
3.1. Reseña histórica.....	36
3.2. Derivación teórica	38
3.2.1. Geometría del algoritmo	39
3.2.2. Distancia anticipada (lookahead distance).....	41
3.3. Implementación.....	42
3.3.1. Representación del camino	42
3.3.2. Algoritmo de búsqueda.....	42
3.4. Conclusión del capítulo.....	43
Marco Metodológico	44
Capítulo 4. Diseño Metodológico	44
4.1. Contexto de la investigación	44
4.2. Alcance.....	45
4.3. Diseño de la investigación.....	45
4.4. Enfoque	46
4.5. Unidad de estudio.....	46
4.6. Técnicas e instrumentos de recolección de datos.....	46
Capítulo 5. Simulación del Algoritmo	47
5.1. Procedimientos	48
5.1.1. Definir condiciones iniciales.....	48
5.1.2. Configuración del robot	50
5.1.3. Controlador de seguidor de ruta.....	51
5.1.4. Funcionamiento del controlador persecución pura	52
5.2. Resultados	53

5.2.1.	Primer mapa: Pista Tacurú Pucú	54
5.2.2.	Segundo mapa: Pista Robocar Race 2018	59
5.3.	Conclusión del capítulo	63
Capítulo 6. Diseño e Implementación del Algoritmo	64
6.1.	Diseño del algoritmo	65
6.2.	Algoritmo de navegación	67
6.2.1.	Odometría	67
6.2.2.	Generación del camino	68
6.2.3.	Seguimiento del camino.....	75
6.2.4.	Interfaz de usuario.....	78
6.3.	Conclusión del capítulo	80
Capítulo 7. Comprobación del Sistema	81
7.1.	Simulación en LabVIEW	83
7.2.	Primer mapa: Pista Tacurú Pucú	87
7.3.	Segundo mapa: Pista Robocar Race 2018.....	88
7.4.	Tercer mapa: Pista Costanera de Hernandarias.....	89
7.5.	Conclusión del capítulo	89
Consideraciones Finales	91
Análisis de Costos	91
Conclusión	93
Trabajos Futuros	96
Bibliografía	97
Glosario	104
Apéndice	105
Apéndice A. Relación entre los sistemas de coordenadas	105
A.1.	Matriz de cosenos directores	105
A.2.	Ángulos de Euler.....	106

A.3.	Cuaterniones.....	109
Apéndice B. Código fuente del algoritmo de implementado.....		110
B.2.1.	Odometría	110
B.2.2.	Generación del camino	111
B.2.1.	Extracción de los puntos de ruta	111
B.2.2.	Inyección de puntos	112
B.2.3.	Suavizado.....	113
B.2.4.	Parámetros.....	114
B.2.5.	SubVI cálculo del radio de curvatura del camino.....	115
B.2.6.	SubVI cálculo de la distancia entre los puntos	116
B.2.7.	SubVI cálculo de la velocidad	116
B.2.8.	SubVI limitador de velocidad	117
B.2.9.	Parte 1: Fragmento del código del subVI Persecución Pura.....	117
B.3.	Seguimiento del camino	119
B.3.1.	SubVI cálculo del punto más cercano.....	119
B.3.2.	SubVI del cálculo del punto anticipado	119
B.3.3.	SubVI del ángulo de corrección.....	120
B.3.4.	SubVI para la visualización del recorrido.....	121
B.3.5.	Parte 2: Fragmento del código del subVI Persecución Pura.....	121
B.4.	Proyecto Algoritmo Persecución Pura	123
Apéndice C. Resultados del concurso del Technical Description Paper.....		126
Apéndice D. Technical Description Paper		127
D.1.	TDP presentado en el concurso	127
D.2.	TDP en proceso de evaluación para ser publicado en el JPAUT	132
Apéndice E. Proyecto presentado en el Labview Student Design Competition		143
Apéndice F. Invitación al evento NIWeek 2019		150
Apéndice G. Nota de Agradecimiento		151

Tabla de Figuras

Figura 1 Niveles de control de un RMA	11
Figura 2 Tipos de navegación en la robótica móvil.....	13
Figura 3 Niveles de autonomía de un vehículo.....	18
Figura 4 El marco de referencia ENU a nivel local en relación con los marcos ECI y ECEF ...	22
Figura 5 Marco de referencia Body de una plataforma móvil	22
Figura 6 Diagrama de flujo del algoritmo navegación por estima	26
Figura 7 Diagrama de flujo del algoritmo persecución pura	28
Figura 8 Pseudocódigo del algoritmo A*	31
Figura 9 Deep Neural Network de un vehículo autónomo	33
Figura 10 Distancia anticipada	39
Figura 11 Geometría del algoritmo.....	39
Figura 12 Distancia anticipada pequeña	41
Figura 13 Distancia anticipada grande.....	41
Figura 14 Puntos de la ruta del robot.....	48
Figura 15 Condiciones iniciales respecto a ubicación y POSE	49
Figura 16 Configuración del robot.....	50
Figura 17 Controlador de seguidor de ruta	52
Figura 18 Controlador persecución pura	53
Figura 19 Pista Tacurú Pucú.....	54
Figura 20 Velocidad = 2 m/s. Distancia Anticipada = 1.8 m	55
Figura 21 Velocidad = 2 m/s. Distancia Anticipada = 2.5 m	56
Figura 22 Velocidad = 2 m/s. Distancia Anticipada = 3.5 m	56
Figura 23 Velocidad = 3 m/s. Distancia Anticipada = 2.5 m	57
Figura 24 Velocidad = 3 m/s. Distancia Anticipada = 0.5 m	58
Figura 25 Pista Robocar Race 2018.....	59

Figura 26 Velocidad = 2.5 m/s. Distancia Anticipada = 0.5 m	60
Figura 27 Velocidad = 2.5 m/s. Distancia Anticipada = 1.7 m	61
Figura 28 Velocidad = 2.5 m/s. Distancia Anticipada = 2.3 m	61
Figura 29 Velocidad = 2.5 m/s. Distancia Anticipada = 3 m	62
Figura 30 Diagrama de flujo del código fuente del algoritmo adaptado	65
Figura 31 Diagrama de flujo del código fuente del algoritmo persecución pura	66
Figura 32 Código fuente del suavizado	70
Figura 33 Método de los 3 puntos	72
Figura 34 Punto anticipado	76
Figura 35 Interfaz de Usuario del Algoritmo.....	78
Figura 36 Interfaz de Ayuda. Parte 1	79
Figura 37 Interfaz de Ayuda. Parte 2	80
Figura 38 Auto eléctrico a escala modificado “Aguara’i”.....	81
Figura 39 Ruta de puntos sin suavizado	84
Figura 40 Ruta suavizada. A = 0.9; B = 0.1	85
Figura 41 Ruta suavizada. A = 0.7; B = 0.3	86
Figura 42 Ruta suavizada. A = 0.1; B = 0.9	86
Figura 43 Ilustración del efecto del orden de las rotaciones del cuerpo.....	107
Figura 44 Código fuente del recabado de datos.....	110
Figura 45 Código fuente de la extracción de los puntos de la ruta.....	112
Figura 46 Código fuente de la inyección de puntos.....	113
Figura 47 Código fuente en LabVIEW del suavizado	113
Figura 48 Parámetros del algoritmo persecución pura	115
Figura 49 Código fuente del cálculo del radio de curvatura.....	115
Figura 50 Código fuente del cálculo de la distancia entre los puntos.....	116
Figura 51 Código fuente del cálculo de la velocidad en un punto.....	116
Figura 52 Código fuente del limitador de velocidad	117

Figura 53 Parte 1: Fragmento del código fuente del subVI Persecución Pura	118
Figura 54 Código fuente del cálculo del punto más cercano	119
Figura 55 Código fuente del cálculo del punto anticipado	120
Figura 56 Código fuente del ángulo de corrección	120
Figura 57 Código fuente de la visualización del recorrido	121
Figura 58 Parte 2: Fragmento del código fuente del subVI Persecución Pura	122
Figura 59 Código fuente del proyecto Algoritmo Persecución Pura. Parte 1	124
Figura 60 Código fuente del proyecto Algoritmo Persecución Pura. Parte 2	125

Lista de Tablas

Tabla 1 Comparación de los algoritmos de navegación	34
Tabla 2 Rango de valores para la distancia anticipada. Mapa Tacurú Pucú	59
Tabla 3 Rango de valores para la distancia anticipada. Mapa Robocar Race 2018.....	63
Tabla 4 Valores recomendados para los parámetros A y B	71
Tabla 5 Valores de los parámetros para la implementación	83
Tabla 6 Valores para la distancia anticipada. Mapa Tacurú Pucú	87
Tabla 7 Valores para la distancia anticipada. Mapa Robocar Race 2018	88
Tabla 8 Valores para la distancia anticipada. Mapa Costanera de Hernandarias	89
Tabla 9 Presupuesto del hardware	91
Tabla 10 Horas de trabajo para el proyecto “Aguara’i”.....	92
Tabla 11 Horas de trabajo para la navegación autónoma	92

Lista de Abreviaturas

CMU.....	Carnegie Mellon University
DARPA.....	Defense Advanced Research Projects Agency
DNN.....	Deep Neural Network
ECEF.....	Earth Centered Earth Fixed
ECL.....	Earth Centered Inertial
ENU.....	East, North, Up
FATEC.....	Facultad de Tecnología de San Pablo
GPS.....	Global Positioning System
IEEE.....	Institute of Electrical and Electronics Engineers
HTML.....	Hyper Text Markup Language
IMU.....	Inertial Measurement Unit
INS.....	Inertial Navigation System
IR.....	Infrarrojo
JPAUT.....	Journal of Production and Automation
LiDAR.....	Light Detection and Ranging
MIT.....	Massachusetts Institute of Technology
MPC.....	Model Predictive Control
NavLab.....	Navigation Laboratory
NED.....	North, East, Down
NL.....	Navegación Local
PGC.....	Planificación Global de Caminos
PID.....	Proporcional, Integral, Derivativo
POSE.....	Position and Orientation
RC.....	Radio Control
RMA.....	Robots (o vehículos) Móviles Autónomos
SAE.....	Society of Automotive Engineers
TDP.....	Technical Description Paper
UFABC.....	Universidad Federal de ABC
VI.....	Virtual Instrument

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 [(e.GO - Mover, 2019), (RINSPEED - microSNAP, 2019)]. 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 (Thorpe C. , Hebert, Kanade, & Shafer, 1988).

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.

Organización del documento

MARCO TEÓRICO

Capítulo 1. FUNDAMENTOS DE LA NAVEGACIÓN AUTÓNOMA. En este capítulo, 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.

Capítulo 2. ALGORITMOS DE NAVEGACIÓN. En este capítulo, se explica de manera básica cuatro algoritmos de navegación. Además, se presentan los criterios para la selección del algoritmo a ser implementado en el auto eléctrico a escala.

Capítulo 3. ALGORITMO PERSECUCIÓN PURA. En dicho capítulo, se explica de forma detallada el algoritmo escogido, desde una reseña histórica hasta su funcionamiento.

MARCO METODOLÓGICO

Capítulo 4. DISEÑO METODOLÓGICO. En dicho capítulo, se explican los procedimientos utilizados para el análisis de la problemática propuesta en dicho proyecto.

Capítulo 5. SIMULACIÓN DEL ALGORITMO. En este capítulo, se muestra el código realizado en el entorno MATLAB para las simulaciones, así como los resultados de los mismos.

Capítulo 6. DISEÑO E IMPLEMENTACIÓN DEL ALGORITMO. En dicho capítulo, se explica el diseño del algoritmo y la teoría del código utilizado para la implementación en el vehículo eléctrico a escala.

Capítulo 7. COMPROBACIÓN DEL SISTEMA. En este capítulo, se muestran los resultados obtenidos en la implementación del algoritmo en el vehículo eléctrico a escala.

CONSIDERACIONES FINALES

ANÁLISIS DE COSTOS. Se presentan tablas respecto al presupuesto del hardware, a las horas de trabajo dedicadas al proyecto “Aguara’i” y a las horas de trabajo dedicadas exclusivamente a la navegación autónoma.

CONCLUSIÓN. Se realiza una conclusión general de los temas abarcados en este libro de tesis.

TRABAJOS FUTUROS. Se presentan sugerencias para trabajos futuros.

GLOSARIO. En esta sección del libro, se definen diversos conceptos.

APÉNDICE

Apéndice A. RELACIÓN ENTRE LOS SISTEMAS DE COORDENADAS. En dicho apéndice, se explica las transformaciones necesarias entre los sistemas de coordenadas Body y de referencia.

Apéndice B. CÓDIGO FUENTE DEL ALGORITMO DE IMPLEMENTADO. En el apéndice, se explica el código fuente implementado en LabVIEW.

Apéndice C. RESULTADOS DEL CONCURSO DE TECHNICAL DESCRIPTION PAPER. En dicho apéndice, se muestra los puestos del concurso de Technical Description Paper.

Apéndice D. TECHNICAL DESCRIPTION PAPER DEL PROYECTO “AGUARA’I”. En el apéndice, se anexa el artículo “Desarrollo de un vehículo autónomo a escala 1:8”.

Apéndice E. PROYECTO PRESENTADO EN EL LABVIEW STUDENT DESIGN COMPETITION. En dicho apéndice, se anexa el artículo presentado en la competencia organizada por National Instruments.

Apéndice F. INVITACIÓN AL EVENTO NIWEEK 2019. En el apéndice, se anexa la invitación de la National Instruments al evento NIWeek 2019.

Apéndice G. NOTA DE AGRADECIMIENTO. En dicho apéndice, se anexa la nota de agradecimiento del equipo A2G enviada al Decano de la Facultad de Ciencias y Tecnología.

Planteamiento del Problema

La demanda global por nuevos ingenieros con conocimientos sobre automatización, sistemas autónomos y sistemas eficientes va aumentando en paralelo con el desarrollo de nuevas tecnologías y nuevas industrias. Según estimaciones de la red global de firmas KPMG, en el 2020, la producción de vehículos en Sudamérica será el 5% del total de unidades fabricadas en el mundo (Moraga, 2017), lo que significa que la industria automotriz en Paraguay también crecerá. Este rápido crecimiento lleva a universidades a desarrollar nuevas formas para adquirir conocimiento en estas áreas, áreas como vehículos autónomos, eficiencia energética, entre otros.

En el contexto nacional, empresas maquiladoras del sector automotriz se han establecido en el Paraguay, dedicándose primariamente en el área de fabricación de cableado automotriz (Fujikura) y ensambladora de vehículos (Grupo Reimpex). Por otro lado, en el contexto internacional, empresas no tradicionales en el rubro automotriz tales como Apple, Google, Uber, se han aliado con empresas del rubro automotriz tales como BMW, Renault, Nissan, Mitsubishi, Audi, entre otros, creando proyectos conjuntos o teniendo su división de vehículos autónomos (Abuelsamid, Alexander, & Jerram, 2017). Esta es una nueva área que, según estudios, demandará mano de obra de ingeniería especializada, que, en el contexto de esta tesis, permite formar los conocimientos que serán demandados por esta industria.

1. Pregunta general

- ¿Qué se debe hacer para que el auto eléctrico a escala pueda conducir de forma autónoma?

2. Preguntas específicas

- ¿Cuáles son algunos de los algoritmos de navegación existentes?
- ¿Cómo analizar el funcionamiento del algoritmo seleccionado sin necesidad del auto eléctrico a escala?
- ¿Cómo comprobar el funcionamiento del algoritmo de navegación utilizando el auto eléctrico a escala?

3. Objetivo general

- Implementar un algoritmo de navegación autónoma en un vehículo eléctrico a escala.

4. Objetivos específicos

- Estudiar los algoritmos de navegación para autos autónomos.
- Realizar simulaciones del funcionamiento del algoritmo seleccionado.
- Implementar el algoritmo seleccionado en el auto eléctrico a escala.

Justificación

Con el aumento de la demanda global por ingenieros con conocimientos en el área de sistemas autónomos, específicamente en vehículos autónomos, las universidades deben buscar alternativas para brindar estos conocimientos a sus estudiantes, ya sean cursos, eventos nacionales o internacionales. Entre los eventos internacionales, se puede destacar la competencia Robocar Race, organizado por la Facultad de Tecnología de Santo André, Universidad Federal del ABC y Robótica Paula Souza (Robocar Race, 2018), evento al cuál, la Universidad Católica “Nuestra Señora de la Asunción” Campus Alto Paraná, fue invitada. Con ello, los alumnos de esta casa de estudio pudieron aprender sobre esta nueva área.

Un factor importante en la construcción y desarrollo del auto autónomo a escala es conocer que algoritmo de navegación es mejor para la situación y es el más factible, puesto que, dependiendo del algoritmo seleccionado, harán que el desempeño del vehículo sea óptimo. Para ello, se realizan estudios y evaluaciones para seleccionar el tipo de algoritmo a implementar.

Marco Teórico

Capítulo 1

Fundamentos de la Navegación Autónoma

En este capítulo, se expone una pequeña introducción a los autos autónomos, su definición y un breve resumen referente a los hechos históricos de los mismos, y a los sensores. Se profundiza con respecto a la navegación autónoma, explicando el concepto y los niveles de control de un robot móvil autónomo, así como la clasificación de la navegación autónoma.

1.1. Auto autónomo

Según la definición de Techopedia (Autonomous Car, s/f), 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.

1.1.1. Reseña histórica

Según el artículo publicado por Li, Díaz, Morantes, & Dorati, (2018), 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 (Thorpe, Hebert, Kanade, & Shafer, 1988), como parte de la Iniciativa de Computación Estratégica de la Agencia de Proyectos de Investigación Avanzados de Defensa (Thorpe C. , Hebert, Kanade, & Shafer, 1991); así también, la Universidad de la Bundeswehr Munich (UniBw Munich, Neubiberg, Alemania) mostró resultados tempranos en la conducción de autopistas de alta velocidad

(Dickmanns & Zapp, 1987). En la demostración final del proyecto EUREKA-PROMETHEUS en 1994, UniBw Munich y Daimler-Benz presentaron la conducción autónoma en el tráfico de autopista de tres carriles, con velocidades de hasta 130 km/h, que incluía el seguimiento de marcas de carriles y otros vehículos. El sistema decidía cuándo cambiar entre carriles por sí mismo, aunque era requerida la aprobación de un conductor humano por razones de seguridad (Dickmanns E. D., 2007).

Como menciona Li, Díaz, Morantes, & Dorati, (2018), en la actualidad, diferentes compañías realizan inversiones millonarias para desarrollar prototipos de vehículos autónomos que realicen diferentes funciones. Según el artículo “Autónomo, inteligente y conectado: el vehículo del mañana, una realidad en 2018”, (2018), entre esas empresas se encuentran Toyota, la cual busca producir plataformas autónomas pensadas para el transporte público, el comercio y la logística; Nissan, esta empresa probó con público real una tecnología capaz de interpretar el pensamiento del conductor para predecir las decisiones que se tomarán al volante con anticipación; Nvidia, la empresa tecnológica especialista en procesadores de cálculo, creó un asistente inteligente con reconocimiento facial pensado específicamente para el sector automotriz. En 2019, General Motors Co. desea lanzar un automóvil totalmente autónomo, sin pedales ni volantes. Además de estas empresas, varias uniones han surgido para desarrollar estas tecnologías, como Lyft y Ford; Uber y Toyota; entre otras.

1.1.2. Sensores

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, dicha tecnología engloba el GPS, radares, cámara, entre otros, es una tecnología de teledetección óptica que utiliza la luz de láser para obtener una muestra densa de la superficie de la tierra produciendo mediciones exactas de x, y, z (Principios básicos de LiDAR, 2018).

Para la navegación, se utiliza los datos provenientes del algoritmo de fusión de sensores, esto se realiza para analizar en qué punto se encuentra y cuál es el siguiente punto donde debe ir el vehículo. Una vez hecho el análisis en el algoritmo de navegación, son enviados los parámetros a los controles para que el vehículo pueda moverse. La fusión de sensores es la combinación sensorial de los datos recogidos por los sensores.

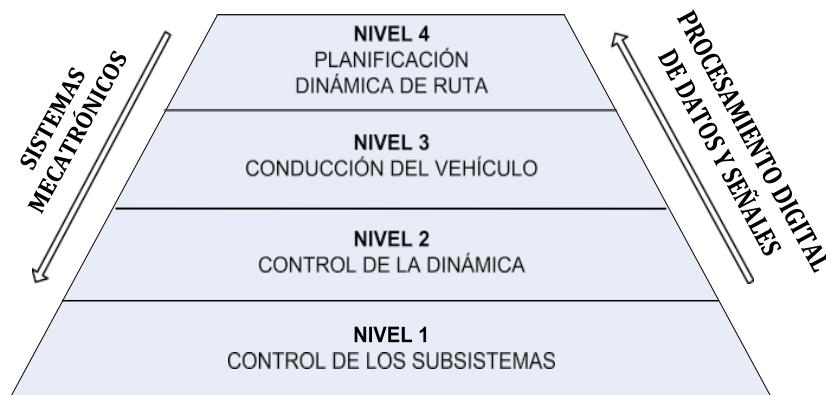
1.2. Navegación autónoma

Según Azkune (2011), 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? (Leonard & Durrant-Whyte, 1991). Valdés Tapia, (2013), 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 de Wit, Crane III, & Armstrong, (2004) donde se organiza el control en forma de cascada, como se presenta en la Figura 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, mientras que el control de la dinámica está en el nivel 2. Finalmente, los sistemas de control sensor/actuador están en el nivel 1.

Figura 1 Niveles de control de un RMA



Fuente: Raffo, Normey-Rico, Rubio, & Kelber, (2009).

Valdés Tapia, (2013), explica en su tesis cada uno de los niveles del control de navegación:

- *Nivel 4*: también llamado Planificación Global de Caminos (PGC) o Navegación Global (NG), consiste en la obtención de un camino geométrico que enlaza la posición inicial del robot con la final, además de asegurarse que el camino esté libre de colisiones. Si este camino está parametrizado en el tiempo se denomina planificación de trayectorias, donde el camino global está compuesto por un conjunto de caminos menores definidos para sortear los obstáculos intermedios, donde estas trayectorias pueden ser calculadas al inicio

de la operación y no recalcularse durante el trayecto o pueden ser dinámicas, dependiendo del nuevo conocimiento del camino, la trayectoria a seguir es calculada. Esta trayectoria global puede estar formada por un conjunto de puntos secuenciales (objetivos intermedios) o por un conjunto de tramos de diversas geometrías como curvas y rectas.

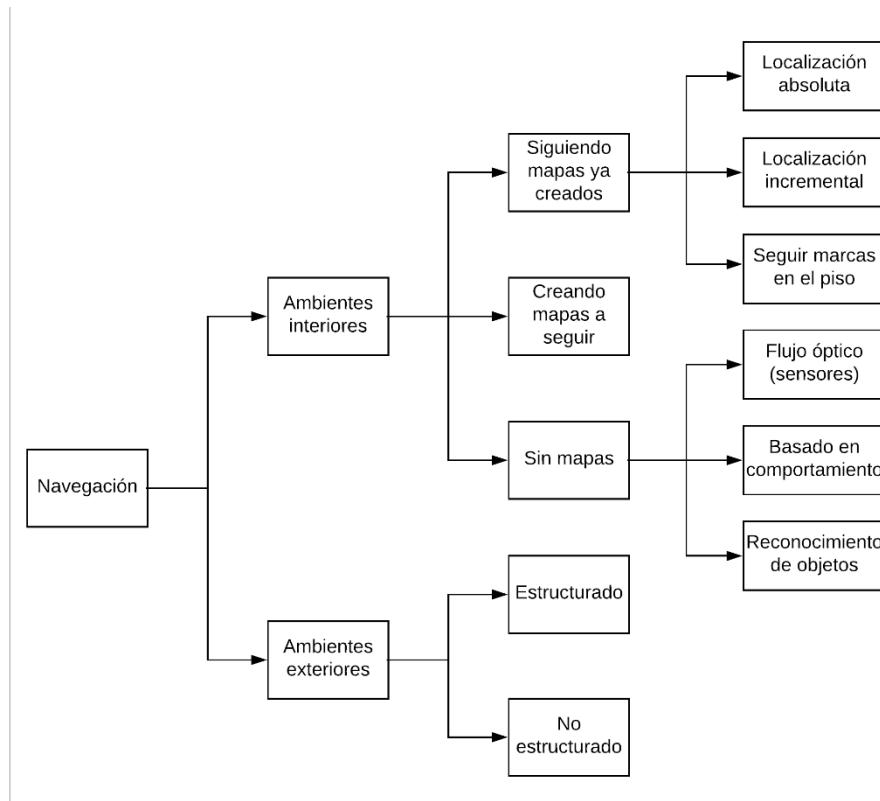
- *Nivel 3:* conducción del vehículo o navegación Local (NL) o guiada, es la encargada de ejecutar los movimientos del robot siguiendo un camino previamente calculado por el planificador global o buscando la posición final del robot, en ambos casos evitando colisiones con el entorno, lo que implica que este nivel debe alimentarse del sistema sensorial del robot. En este nivel se considera la dinámica del robot y las limitaciones que ésta establece para realizar las acciones de control.
- *Nivel 2:* nivel de percepción y control, es el primer nivel donde existe control del movimiento del robot. A diferencia de los anteriores niveles, en este no existe inteligencia. El único control que se puede implementar aquí es uno reactivo, por ejemplo, que envíe señales a los servomotores o a los sensores para conocer las acciones desarrolladas. La robótica móvil educativa orientada a enseñar algoritmos básicos de control se enfoca en este nivel. Lo normal es que la reacción del robot sea generada por el trabajo de los niveles anteriores, cuando éstos existen.
- *Nivel 1:* en el control de subsistemas se encuentran los sensores y actuadores, que son el hardware del robot, en mayor medida electrónica y mecánica, destinada a capturar las señales del entorno y entregar la tracción a las ruedas del robot.

1.2.1. Clasificación de la navegación autónoma

1.2.1.1. Según la forma de navegación

La navegación puede realizarse de distintas maneras, en el artículo de DeSouza & Kak, (2002), se describen los tipos de navegación. En la Figura 2 se resume la clasificación.

Figura 2 Tipos de navegación en la robótica móvil



Fuente: DeSouza & Kak, (2002).

En el artículo mencionado en el párrafo anterior, los autores dividen los tipos de navegación en dos grandes grupos, navegación para ambiente interior (*indoor*) y navegación para ambiente exterior (*outdoor*). Estas grandes categorías se diferencian en la técnica de posicionamiento del robot o auto. La localización en el exterior se realiza casi exclusivamente con el GPS, los sistemas

de localización en interiores han empleado con éxito una variedad de tecnologías (Otsason, Varshavsky, LaMarca, & De Lara, 2005). Esas tecnologías pueden ser Wi-Fi, Bluetooth, ultrasonido e infrarrofo.

A seguir se menciona la explicación de cada categoría dada por los autores DeSouza & Kak, (2002).

Navegación en ambientes interiores.

- a. *Siguiendo mapas ya creados:* como su nombre lo dice, son sistemas que dependen de modelos geométricos creados por el usuario o mapas topológicos del entorno.
 1. *Localización absoluta:* en esta categoría la posición inicial del robot es desconocida, debido a ello, el sistema de navegación debe construir una coincidencia entre las observaciones y las expectativas derivadas de toda la base de datos. Existe la posibilidad que el mismo conjunto de observaciones coincida con múltiples expectativas por el hecho de que las incertidumbres están asociadas con las observaciones. Algunas de las soluciones que proponen los autores para las ambigüedades resultantes en la localización pueden ser los métodos: localización de Markov (Thrun, 2000), localización de Monte Carlo [(Blake & Isard, 1997), (Dellaert, Fox, Burgard, & Thrun, 1999)], filtrado de Kalman de múltiples hipótesis basado en una mezcla de gaussianos (Cox & Leonard, 1994), entre otros. Resumiendo, en palabras sencillas lo expuesto por Atiya & Hager, (1993), lo esencial de su enfoque es reconocer todo aquello que permanece fijo, en la imagen de la cámara, con respecto a la posición y orientación del robot mientras viaja en su entorno (DeSouza & Kak, 2002).

2. *Localización incremental*: los autores explican que debido a que la posición inicial del robot se conoce al menos aproximadamente, el algoritmo de localización debe realizar un seguimiento de las incertidumbres en la posición del robot mientras ejecuta los comandos de movimiento. En esta categoría, los sensores son utilizados para realizar correcciones en la posición una vez que las incertidumbres superan un límite. De igual modo, exponen que las técnicas probabilísticas han evolucionado como el enfoque preferido para la representación y la actualización de las incertidumbres posicionales a medida que el robot se mueve, por ello, presentan como solución el sistema FINALE (Kosaka & Kak, 1992). Este sistema logra una localización incremental mediante el uso de una representación geométrica del espacio y un modelo estadístico de incertidumbre en la ubicación del robot.
 3. *Seguir marcas en el piso*: en esta categoría la localización se realiza mediante el seguimiento de puntos de referencia. DeSouza & Kak, (2002) mencionan que esto se debe a que, tanto la ubicación aproximada del robot como la identidad de los puntos de referencia que se ven en la imagen de la cámara son conocidas y pueden rastrearse.
- b. *Creando mapas a seguir*: estos son sistemas que utilizan sensores para construir sus propios modelos geométricos o topológicos del entorno y luego utilizan estos modelos para la navegación (DeSouza & Kak, 2002).
- c. *Sin mapas*: estos son sistemas que realizan la navegación sin ninguna representación explícita sobre el espacio donde se encuentran, sino que más bien recurren al reconocimiento de objetos encontrados en el entorno o al seguimiento de esos objetos

mediante la generación de movimientos basados en observaciones visuales (DeSouza & Kak, 2002).

1. *Flujo óptico (sensores)*: Santos-Victor, Sandini, Curotto, & Garibaldi, (1993), han desarrollado un sistema de flujo óptico que imita el comportamiento visual de las abejas. Esto es porque, en lugar de usar la información de profundidad, el mecanismo de navegación se realiza con características derivadas del movimiento. Por otro lado, el movimiento de paralaje puede ser mucho más útil, especialmente cuando el insecto está en movimiento relativo con respecto al medio ambiente. Además, la precisión y el rango de operación pueden modificarse cambiando la velocidad relativa (DeSouza & Kak, 2002).
2. *Basado en comportamiento*: en esta categoría, la navegación autónoma se logra mediante la "memorización" del entorno. Es decir, se almacena imágenes o plantillas del entorno para luego asociar esas imágenes con comandos o controles que llevarán al robot a su destino final (DeSouza & Kak, 2002). Los autores citan la solución dada por Gaussier, Joulain, Zrehen, Banquet, & Revel, (1997), y Joulain, Gaussier, Revel, & Gas, (1997), como ejemplo para esta categoría de navegación, el desarrollo de un enfoque basado en la apariencia utilizando redes neuronales para mapear la percepción en acción.
3. *Reconocimiento de objetos*: Kim & Nevatia, (1998) y (1999), han propuesto un enfoque diferente para la navegación automática, utilizar un enfoque de navegación simbólica, donde, el robot toma comandos como "ir a la puerta" o "ir al escritorio frente a usted", y utiliza la información simbólica contenida en estos comandos para

establecer los puntos de referencia que necesita reconocer y la ruta que necesita tomar para alcanzar la meta (DeSouza & Kak, 2002).

Navegación en ambientes exteriores.

- a. *Estructurado*: en general, la navegación al aire libre en entornos estructurados requiere algún tipo de seguimiento. Estos sistemas, generalmente poseen modelos de ambientes simples, que contienen información como puntos de fuga, anchos de caminos y carreteras, es decir, poseen la capacidad de reconocer las líneas que separan los carriles o separan el camino de la berma, la textura de la superficie del camino y las superficies adyacentes, etc. El seguimiento de caminos para los robots de exteriores puede ser similar al de los robots de interiores, a excepción de los problemas causados por las sombras, el cambio de las condiciones de iluminación, el cambio de colores (DeSouza & Kak, 2002).
- b. *No estructurado*: un entorno no estructurado, como definen los autores, es un entorno al aire libre sin propiedades regulares que podrían percibirse y rastrearse para la navegación. En tales casos, el sistema de visión puede utilizar como máximo una caracterización genérica de los posibles obstáculos en el medio ambiente (DeSouza & Kak, 2002).

1.2.1.2. Según el nivel de 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. Esta organización se llama Sociedad de Ingenieros Automotrices (SAE), nació a comienzos del siglo XX con el fin de desarrollar estándares para los fabricantes y ensambladoras de vehículos,

ya sea en el área automotriz o aeroespacial. La jerarquía es presentada en el estándar SAE J3016 y son los siguientes:

Figura 3 Niveles de autonomía de un vehículo



Fuente: SAE International STANDARD J3016, (2016).

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

- Movimiento del vehículo:* ¿Quién se encarga? ¿El humano o la máquina?

El movimiento se diferencia entre longitudinal (o sea, acelerar y frenar) y lateral (o sea, la dirección).

- Detección y respuesta ante objetos y eventualidades:* ¿Quién se encarga mediante sistemas

que monitoricen el entorno del vehículo durante la conducción? ¿El humano o la máquina?

- Respaldo de la conducción:* ¿Quién se encarga en caso de fallo de los sistemas

automatizados, o ante la pérdida de las condiciones para su funcionamiento? ¿El humano o la máquina?

- d. *Condiciones específicas para el funcionamiento del sistema* (horarias, climatológicas, geográficas, tipo de carretera, cantidad de tráfico, velocidad, entre otros): ¿El sistema de conducción automatizada funciona en todo caso, o solo en ciertas condiciones que lo limitan?

1.3. Sistema de coordenadas

Un concepto importante para la navegación es el de sistema de coordenadas, tomando como base la tesis doctoral de Amezcua Paredes & Pineda Salgado, (2013), al hablar de sistema de coordenadas se hace mención a un conjunto de vectores y números sin sentido, los cuales deben ser relacionados con una referencia conocida para que adquieran significado. Existen varios sistemas de coordenadas fundamentales: sistemas ortogonales, dextrógiros y cartesianos. Estos sistemas difieren en el origen, la orientación relativa de sus ejes y el movimiento relativo entre sus planos.

A continuación, se presentan algunos sistemas de coordenadas relevantes para la navegación:

- *Sistema de coordenadas Inercial I-frame*: es un sistema coordenadas donde se cumplen las leyes de Newton del movimiento. Dicho sistema ni rota ni acelera. En la práctica, consiste en un conjunto de ejes perpendiculares entre sí. Sin embargo, debido a que en la práctica es imposible encontrar un sistema realmente inercial, puesto que todos los sistemas se mueven, este sistema es utilizado para la aproximación teórica y para visualizar otros marcos de referencia más fácilmente.
- *Sistema de coordenadas ECI i-frame*: Este sistema de coordenadas se mueve con el planeta, debido a ello, no rota respecto al espacio inercial, aunque dicha afirmación no

es del todo correcta, puesto que la Tierra gira con respecto al Sol, además de su propia rotación, sin embargo, este sistema será tomado como inercial, esto es importante porque, como se mencionó anteriormente, las leyes de newton son casi ciertas y permite visualizar más fácilmente otros marcos de referencia no inerciales.

El origen de este sistema de coordenadas se encuentra en el centro de masas de la Tierra y los ejes del mismo están fijados en las estrellas: El eje Z coincide con el eje polar y el plano perpendicular al eje Z coincide con el Ecuador. El eje X e Y no rotan con la tierra, apuntando X directamente al equinoccio Vernal.

- *Sistema de coordenadas ECEF e-frame*: o sistema de coordenadas geocéntrico. Al igual que el sistema anterior, tiene su origen en el centro de masas de la Tierra, los ejes rotan con la Tierra. El eje Z se dirige directamente al norte a lo largo del eje polar. Los ejes X e Y están en el plano ecuatorial con X dirigido hacia el meridiano de Greenwich (0° latitud, 0° longitud) y el eje Y 90° hacia el Este.
- *Sistema de coordenadas de Navegación n-frame*: Es un sistema local con sus ejes X e Y en el plano tangente al punto de la Tierra donde está el origen, ese origen se encuentra en la localización del sistema inercial (longitud, latitud).

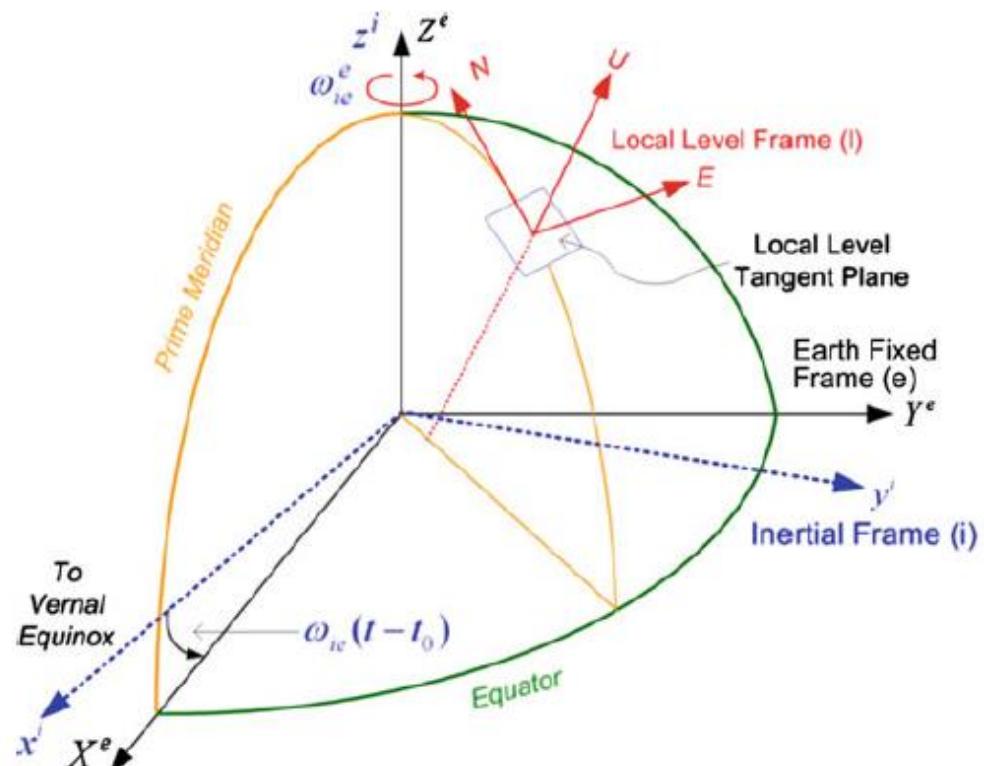
Otra forma de llamarlo es NED (North, East, Down), debido a las direcciones que apuntan sus ejes, el eje X apunta al norte, el eje Y al Este y el eje Z abajo, aunque debe ser especificado puesto que puede ser utilizado también con la configuración ENU (East, North, Up) que sería: el eje X apuntando al este, el eje Y apuntando al norte y el eje Z apuntando hacia arriba.

- *Sistema de coordenadas Body b-frame*: Este sistema es utilizado en plataformas strapdown, debido a que dicho sistema tiene su origen en el centro de masas del vehículo. Dicho de otra manera, los ejes del sistema coinciden con los ejes de los sensores y se mueven con él.

Para comprender mejor como se relacionan dichos sistemas, se muestran las figuras: Figura 4 y Figura 5. En la Figura 4, se observa la relación entre el sistema de coordenadas ENU (rojo) y los sistemas ECI (azul) y ECEF (negro). En caso de se quiera saber la relación entre el sistema NED y los sistemas mencionados anteriormente, solo es necesario cambiar la dirección del eje z, este se orientaría hacia el centro de la Tierra.

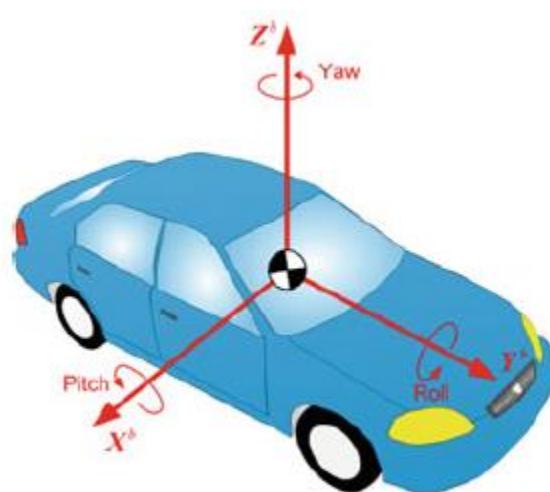
En la Figura 5, se muestra el sistema de coordenadas Body, el cual tiene como origen de coordenadas el centro de masa del cuerpo.

Figura 4 El marco de referencia ENU a nivel local en relación con los marcos ECI y ECEF



Fuente: Noureddin, Karamat, & Georgy, (2012).

Figura 5 Marco de referencia Body de una plataforma móvil



Fuente: Noureddin, Karamat, & Georgy, (2012).

1.4. Sistema de navegación inercial

Otro concepto importante es el *Sistema de Navegación Inercial (INS)*, dicho sistema “consiste en una unidad de medida inercial que incluye los sensores ligados a una plataforma común para mantener las mismas orientaciones relativas y un ordenador para procesar estas medias o cualquier otro cálculo” (Ferrer Mínguez, 2009).

Existen dos configuraciones básicas para la implementación de la navegación inercial: Sistema con plataforma estabilizada o Sistema cardán (*gimbaled*) y Sistema con sensores fijos a la estructura del vehículo (*strapdown*) (Grewal & Andrews, 2008). En este proyecto se utilizará el sistema con sensores fijos a la estructura del vehículo.

“El sistema con plataforma estabilizada pretende aislar la plataforma con los sensores iniciales de los movimientos de rotación externos. Las plataformas estabilizadas están sujetas a un marco rígido que rota de tal manera que aísla el interior de la plataforma de rotaciones externas con los ejes del cuerpo. Generalmente es imposible alcanzar un nivel de aislamiento perfecto y siempre rota algo.”

En el sistema con sensores fijos a la estructura del vehículo, los ejes de los sensores están alineados con los ejes del móvil. El mismo, necesita mayor capacidad de cálculo ya que es necesario “aislar virtualmente” las medidas de los sensores al sistema de referencia inercial.” (Ferrer Mínguez, 2009)

1.5. Conclusión del capítulo

Para hablar de algoritmos de navegación autónoma, primero es necesario conocer conceptos básicos de navegación. Con ese objetivo fue desarrollado este primer capítulo, donde se pudieron conocer los conceptos de auto autónomo, navegación autónoma, sus clasificaciones y dos conceptos que deben ser conocidos para hablar de navegación, sistemas de coordenadas y sistema de navegación inercial.

Capítulo 2

Algoritmos de Navegación

En dicho capítulo, se realiza el estudio de cuatro algoritmos de navegación, con el fin de seleccionar uno para, posteriormente, implementarlo. Se presentan definiciones y revisiones bibliográficas de los algoritmos, además de diagramas de flujo, pseudocódigos o imágenes respecto al funcionamiento de cada uno de ellos. Se mencionan los criterios utilizados para la selección del algoritmo a ser implementado.

Como se mencionó anteriormente, la navegación autónoma es la capacidad del vehículo de ir desde un punto inicial a un punto final. El algoritmo de navegación es el código responsable de realizar los cálculos necesarios para que dicha trayectoria pueda ser realizada, es decir, analiza los valores proveídos por los sensores, calcula el siguiente paso que se debe seguir y arroja los parámetros a los controles para que el vehículo pueda seguir la ruta correspondiente.

Existen diversos algoritmos de navegación, a continuación, se mencionarán algunos de ellos, junto con una breve reseña de cada.

2.1. 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 presentado por Park, Chung & Lee en 1998:

“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 iniciales, 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.”

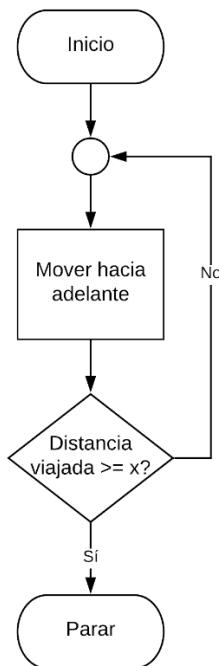
Park, Chung, & Lee, (1998), también explican que los errores del encoder se clasifican en dos categorías, en errores sistemáticos y no sistemáticos, así como los errores de los sensores de inercia, en errores determinísticos y estocásticos [(Borenstein & Feng, 1995), (Siouris, 1993)]. Los errores deterministas pueden compensarse con la función adecuada, pero los errores estocásticos deben compensarse mediante el filtrado con modelos de error. Estos autores mencionan que el filtro extendido de Kalman se ha utilizado ampliamente para la compensación de errores del sistema de navegación, y las características deterministas de los sensores iniciales se reflejan en investigaciones como los de Barshan & Durrant-Whyte, (1995) y Komoriya & Oyama, (1994).

El encoder mide el desplazamiento a lo largo del recorrido. El robot con tracción diferencial mide no solo el desplazamiento sino también el ángulo de dirección, esto se da porque existe la posibilidad de tener dos encoders, uno en cada rueda. Como los errores del encoder son proporcionales a la distancia de movimiento, el error en la posición y en el ángulo de dirección aumentan sin límite. Estos errores pueden ser sistemáticos, estos errores son dominantes en las superficies lisas de los ambientes interiores, entre ellos están: diámetros de rueda desiguales, diferencia entre la distancia real y la distancia nominal entre ejes y desalineación de ruedas; o no sistemáticos, dichos errores se pueden dar en superficies ásperas, entre ellos están: viajar sobre

suelos irregulares, viajar sobre objetos inesperados en el suelo y deslizamiento de la rueda (Park, Chung, & Lee, 1998).

A continuación, en la Figura 6, se muestra el diagrama de flujo presentado en el primer video de la serie Student Competition: Mobile Robotics Training de The MathWorks, Inc, donde se explica el funcionamiento del algoritmo a muy alto nivel. Básicamente, 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.

Figura 6 Diagrama de flujo del algoritmo navegación por estima



Fuente: D'Souza (2017).

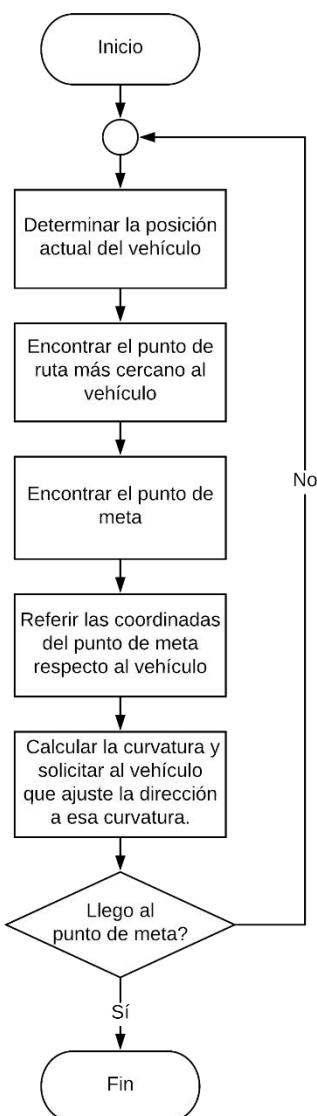
*x es igual a la distancia especificada al inicio del algoritmo

2.2. Persecución pura (Pure pursuit)

Conforme al artículo de Coulter (1992), 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 Figura 7, el diagrama de flujo del algoritmo resumido por Coulter, (1992).

Figura 7 Diagrama de flujo del algoritmo persecución pura



Fuente: Coulter, (1992).

Para explicar específicamente como funciona este algoritmo, se recurre a la tesis doctoral presentada por Campbell, (2007), en el MIT. Campbell describe que

“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.”

El autor además aclara que, en la práctica, esta formalidad en la definición de la trayectoria del arco no será necesaria, aunque debería entenderse implícitamente.

2.3. 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 [(Zhang, Dechter, & Korf, 2001), (Mejía Peñafiel, 2010)]. 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 [(Bekey & Goldberg, 1993), (Patel, s/f)].

2.3.1. 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 (Duchoň, y otros, 2014). 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 (LaValle, 2006).

La diferencia existente entre el algoritmo A* y el algoritmo de Dijkstra es que, en el primer algoritmo, como explica Reddy, (2013), en su artículo, se va calculando el costo de las celdas que siguen la ruta del menor costo conocido, manteniendo una cola de prioridad ordenada de segmentos de ruta alternativos en el camino. Si, en cualquier punto, un segmento de la ruta que se atraviesa tiene un costo más alto que otro segmento de ruta encontrado, abandona el segmento de la ruta de mayor costo y atraviesa el segmento de la ruta de menor costo. Este proceso continúa hasta alcanzar la meta. En cambio, en el segundo algoritmo, se realiza el cálculo de costo de todas las celdas, pertenezcan ellas a la ruta del menor costo conocido o no. Debido a ello, la cantidad de iteraciones es mayor, por lo tanto, el tiempo de cálculo de la ruta con menor costo es mayor con respecto al tiempo de cálculo del algoritmo A*.

A pesar de tener un tiempo de evaluación menor al tiempo del algoritmo de Dijkstra, este algoritmo posee algunas deficiencias, una de ellas es que solo puede ser utilizado en ambientes estáticos, es decir, cuando los obstáculos son fijos. A partir de las mejoras que se fueron realizando sobre el algoritmo A*, surgieron los algoritmos A Dinámico estrella (Dynamic A*), D estrella (D*) y D estrella Lite (D* Lite). La base de estos algoritmos es el A estrella, la diferencia es que son para ambientes dinámicos o tienen una respuesta más rápida (Ganapathy, Yun, & Chien, 2011).

Continuando con el algoritmo de búsqueda A*, la fórmula mencionada por LaValle, (2006), para dicho algoritmo es:

$$f(v) = g(v) + h(v) \quad 2.1.$$

Donde Duchoň, et al., (2014), 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 Figura 8, se observa el pseudocódigo de alto nivel presentado por Abiy, Pang, Tiliksew, Moore, & Khim (s/f). En ella, se muestra en forma de código lo que Duchoň, et al., (2014) han explicado en su artículo.

Figura 8 Pseudocódigo del algoritmo A*

```

1 Put node_start in the OPEN list with  $f(\text{node\_start}) = h(\text{node\_start})$  (initialization)
2 while the OPEN list is not empty {
3     Take from the open list the node node_current with the lowest
4          $f(\text{node\_current}) = g(\text{node\_current}) + h(\text{node\_current})$ 
5     if node_current is node_goal we have found the solution; break
6     Generate each state node_successor that come after node_current
7     for each node_successor of node_current {
8         Set successor_current_cost =  $g(\text{node\_current}) + w(\text{node\_current}, \text{node\_successor})$ 
9         if node_successor is in the OPEN list {
10             if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
11         } else if node_successor is in the CLOSED list {
12             if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
13             Move node_successor from the CLOSED list to the OPEN list
14         } else {
15             Add node_successor to the OPEN list
16             Set  $h(\text{node\_successor})$  to be the heuristic distance to node_goal
17         }
18         Set  $g(\text{node\_successor}) = \text{successor\_current\_cost}$ 
19         Set the parent of node_successor to node_current
20     }
21     Add node_current to the CLOSED list
22 }
23 if(node_current != node_goal) exit with error (the OPEN list is empty)

```

Fuente: Alseda, (s/f)

2.3.2. Redes neuronales

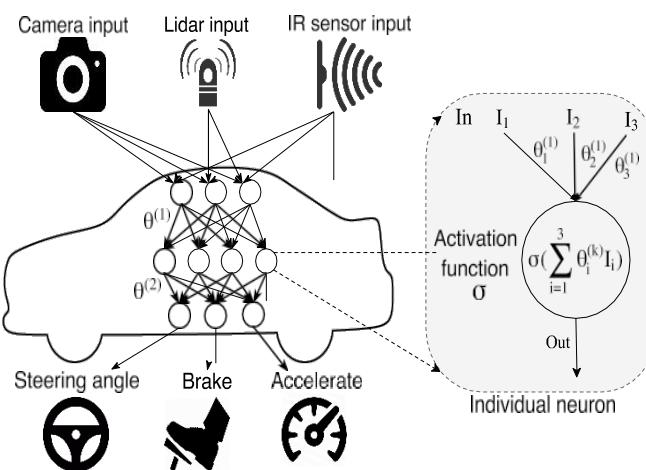
“El interés en la red neuronal se deriva del deseo de comprender los principios que conducen de alguna manera a la comprensión de las funciones básicas del cerebro humano y de construir máquinas capaces de realizar tareas complejas. Esencialmente, la red neuronal se ocupa de tareas cognitivas como el 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.” (Janglová, 2004)

De acuerdo a Baquero Suárez & Cudris Cantillo (2007), 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. Acevedo & Castañeda (2007) 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.

Este tipo de redes se caracterizan por tener un aprendizaje o entrenamiento híbrido, es decir, un aprendizaje en la capa oculta mediante técnicas no supervisadas con funciones de activación radial y aprendizaje mediante corrección del error en la capa de salida realizando un ajuste fino de todos los parámetros mediante la minimización de alguna función de coste (Baquero Suárez & Cudris Cantillo, 2007).

Tian, Pei, Jana, & Ray, (2018), investigaron respecto a la Red Neuronal Profunda (Deep Neural Network) en vehículos autónomos. En su artículo mencionan que el componente clave de un vehículo autónomo es el módulo de percepción controlado por la DNN subyacente. 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) (Figura 9).

Figura 9 Deep Neural Network de un vehículo autónomo



Fuente: Tian, Pei, Jana, & Ray, (2018).

Continuando con la explicación dada en el artículo de Tian, Pei, Jana, & Ray, (2018), 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 (θ s en la Figura 9). 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 Figura 9. La mayoría de los DNN existentes se entrena con pendiente de gradiente utilizando la propagación hacia atrás (Rumelhart, Hinton, & Williams, 1986). Una vez terminado el entrenamiento, se puede usar un DNN para la predicción.

2.4. 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 estudiados 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.

Tabla 1

Comparación de los algoritmos de navegación

Criterios de evaluación	Navegación por estima	Persecución Pura	Algoritmo de búsqueda A*	Redes Neuronales
Nivel de dificultad del algoritmo de navegación con respecto al aprendizaje del mismo.	Bajo	Medio	Alto	Alto
Tiempo de aprendizaje que tomará el algoritmo de navegación.	Bajo	Medio	Alto	Alto
Tiempo de programación o, en el caso de utilizar el algoritmo de terceros, el tiempo que llevará realizar las mejoras necesarias al algoritmo escogido.	Bajo	Medio	Alto	Alto
Tiempo de entrenamiento del algoritmo de navegación.	Medio	Bajo	Alto	Alto
Nivel de adaptación del algoritmo a pistas desconocidas (Cuando la ruta a seguir no es conocida de antemano).	Bajo	Alto	Medio	Alto

Fuente: Elaboración propia.

2.5. Conclusión del capítulo

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.

Capítulo 3

Algoritmo Persecución Pura

En este capítulo, se explica específicamente sobre el algoritmo persecución pura, se realiza una reseña histórica, la derivación teórica del algoritmo, así como se explica la manera de implementar el algoritmo.

Como se mencionó en el Capítulo 2, 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 (Wang, Hsu, & Wu, 2017).

3.1. Reseña histórica

Al hablar del algoritmo persecución pura, uno pensaría que siempre fue utilizado en robots o vehículos, no obstante, en 1969, Scharf, Harthill, & Moose, 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 (Samuel, Hussein, & Mohamad, 2016).

En 1985, Wallace, y otros, aplicaron el algoritmo por primera vez como un método donde se calculaba la dirección necesaria para mantener el vehículo en la carretera (Coulter, 1992). Wallace, et al., (1985) 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 Wallace, et al., (1985), Amidi & Thorpe, (1991), proponen el método de búsqueda pura para seguir caminos explícitos (Morales, Martínez, Martínez, & Madow, 2009). 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 (Amidi & Thorpe, 1991). Las pruebas desarrolladas de todos estos algoritmos mostraron que el método de persecución pura era el método más robusto y confiable (Coulter, 1992).

Coulter, (1992), 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 [(Ollero, García-Cerezo, & Martínez, 1994), (Samuel, Hussein, & Mohamad, 2016)].

Morales, Martínez, Martínez, & Madow, (2009) explican que, a mediados de los años 90, Murphy, (1994), y Ollero & Heredia, (1995), investigaron acerca de las condiciones de estabilidad del algoritmo. Murphy, (1994), estudió los efectos de los retrasos en el tiempo asociados con el procesamiento visual en las siguientes líneas rectas, mientras que Ollero & Heredia, (1995), 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.

Ollero, García-Cerezo, & Martínez, (1994), 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 (Rodríguez-Castaño, Heredia, & Ollero, 2000).

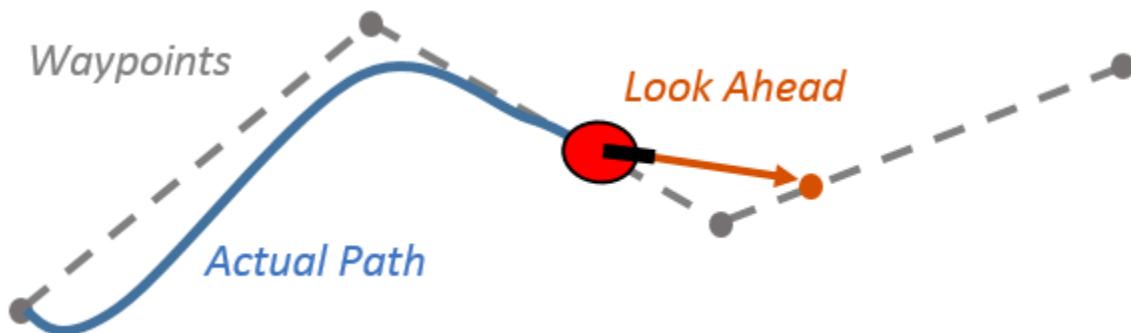
Resumiendo lo expuesto por Samuel, Hussein, & Mohamad, (2016), en 2003, resuelven el problema de que el vehículo está lejos del camino al crear un punto de meta virtual en la distancia de anticipación (Petrinec, Kovacic, & Marozin, 2003); en 2011, Abbas, creó el controlador MPC (Control Predictivo por Modelo) 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, Zhao, y otros, propusieron el uso del controlador PID (Proporcional, Integral y Derivativo) 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% (Wang, Hsu, & Wu, 2017).

3.2. Derivación teórica

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 [The MathWorks, Inc, s/f(a)]. En la Figura 10 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 [The MathWorks, Inc, s/f (a)].

Figura 10 Distancia anticipada

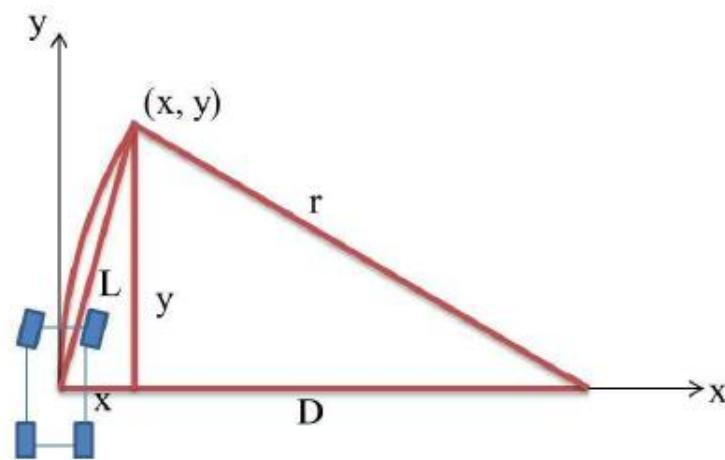


Fuente: The MathWorks, Inc, s/f (a).

3.2.1. Geometría del algoritmo

Para comprender mejor sobre cómo funciona el algoritmo y la distancia anticipada, se muestra la Figura 11, en ella, se observa el vehículo con los ejes del sistema de coordenadas, x e y .

Figura 11 Geometría del algoritmo



Fuente: Wang, Hsu, & Wu, (2017).

El punto $(x; y)$ es un punto que se encuentra a una distancia anticipada del vehículo. Esta 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 (Coulter, 1992). Se tiene por objetivo calcular la curvatura del arco, para ello, Coulter, (1992), presenta las siguientes relaciones:

$$x^2 + y^2 = L^2 \quad 3.1.$$

$$D^2 + y^2 = r^2 \quad 3.2.$$

$$x + D = r \quad 3.3.$$

De las ecuaciones 3.1., 3.2. y 3.3.:

$$r^2 - 2rx + x^2 + y^2 = r^2 \quad 3.4.$$

$$r = \frac{L^2}{2x} \quad 3.5.$$

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

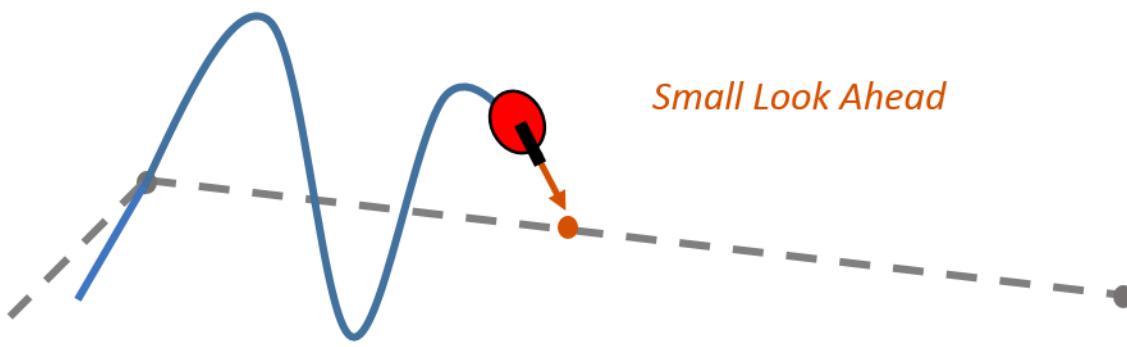
$$\gamma = \frac{1}{r} = \frac{2x}{L^2} \quad 3.6.$$

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 (Wang, Hsu, & Wu, 2017).

3.2.2. Distancia anticipada (lookahead distance)

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 Figura 12 [The MathWorks, Inc, s/f (a)].

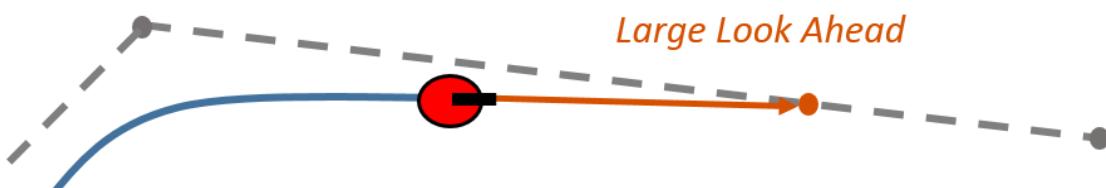
Figura 12 Distancia anticipada pequeña



Fuente: The MathWorks, Inc, [s/f (a)].

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 [The MathWorks, Inc, s/f (a)], dicho de otro modo, observando la Figura 13, la recuperación del camino no se dará de forma rápida.

Figura 13 Distancia anticipada grande



Fuente: The MathWorks, Inc, [s/f (a)].

3.3. 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) (Coulter, 1992).

3.3.1. Representación del camino

Como explica Coulter, (1992), el camino se representa con un conjunto de puntos discretos que son almacenados en la memoria del robot o vehículo. Según el estudio de Coulter, 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.

3.3.2. Algoritmo de búsqueda

Para la implementación del algoritmo en sí, se toma como base los pasos propuestos por Coulter, (1992), mencionados en la Figura 7. 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.

3.4. Conclusión del capítulo

El algoritmo es mayormente utilizado para navegación de ambientes interiores. Sin embargo, puede ser utilizado en ambientes exteriores como complemento. En el proyecto NavLab, el algoritmo persecución pura fue utilizado conjunto con visión computacional, con el fin de mantener al vehículo en la línea del centro de la ruta, véase (Kanade & Thorpe, 1985). Teniendo en cuenta esto, dicho algoritmo puede encontrarse en la categoría de siguiendo mapas ya creados, que sería el caso de introducir al inicio de la navegación una ruta con puntos que debe seguir, o en creando el mapa a seguir, como sería el caso de NavLab, que era posible gracias a las cámaras que utilizaba.

Todo lo mencionado anteriormente depende en gran medida de los sensores seleccionados para la navegación y si es que el algoritmo persecución pura será utilizado solo o en conjunto con otro algoritmo de navegación.

Marco Metodológico

Capítulo 4

Diseño Metodológico

En el presente capítulo, se explican los procedimientos utilizados para el análisis de la problemática propuesta en dicho proyecto.

4.1. Contexto de la investigación

El presente proyecto se realizó en el contexto de investigador junior del Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada de la Universidad Católica “Nuestra Señora de la Asunción” Campus Alto Paraná. Dicha investigación surgió de la invitación a la competencia mencionada en la Justificación de este libro, Robocar Race (2018). El equipo formado para dicho evento se denomina “A2G” y 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. El auto eléctrico a escala autónomo ha sido nombrado “Aguara’i”.

El proyecto “Aguara’i” consiste en un auto eléctrico a escala 1:8 que navega de forma autónoma. Para ello, fue modificado un auto eléctrico a escala RC (Control Remoto), y fueron adaptados sensores (encoders, acelerómetro, giroscopio y magnetómetro) y un dispositivo embebido (myRIO) para realizar la tarea de adquisición, procesamiento de los algoritmos de fusión, control y navegación.

4.2. Alcance

Para dicho proyecto, se abarcó el módulo Path Planner del proyecto “Aguara’í”, véase Apéndice D. Dicho módulo se encarga de realizar la navegación del vehículo tomando los datos del algoritmo de fusión de sensores y procesándolos para enviar los valores correspondientes de velocidad y dirección al algoritmo de control.

En esta investigación se presenta un estudio de algoritmos de navegación para implementar en el auto eléctrico a escala. Se elaboró una taxonomía de algoritmos de navegación y se propuso métricas de evaluación de desempeño. Se evaluó en concreto el algoritmo persecución pura que ha sido implementado en el “Aguara’í”.

No forma parte del alcance de esta tesis, la adaptación del auto eléctrico “Aguara’í” ni el diseño de los algoritmos de: fusión de sensores, y control de velocidad y dirección.

4.3. Diseño de la investigación

El diseño fue experimental. El diseño experimental es

“Una acepción particular de experimento, más armónica con un sentido científico del término, se refiere a un estudio en el que se manipulan intencionalmente una o más variables independientes (supuestas causas-antecedentes), para analizar las consecuencias que la manipulación tiene sobre una o más variables dependientes (supuestos efectos-consecuentes), dentro de una situación de control para el investigador.” (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2010)

Tomando como base el concepto presentado, en esta investigación son manipulados de manera intencional los parámetros del algoritmo diseñado, con el fin de, analizar el comportamiento de la navegación del auto “Aguara’í”.

4.4. Enfoque

Puesto que, en dicha investigación, son utilizados cálculos matemáticos para la navegación del auto, además, son analizados posteriormente con el objetivo de definir su comportamiento, se establece el enfoque del proyecto como cuantitativo. Según Hernández Sampieri et al (2010):

“En el enfoque cuantitativo los planteamientos a investigar son específicos y delimitados desde el inicio de un estudio. Además, las hipótesis se establecen previamente, esto es, antes de recolectar y analizar los datos. La recolección de los datos se fundamenta en la medición y el análisis en procedimientos estadísticos, la investigación debe ser lo más “objetiva” posible. Los estudios cuantitativos siguen un patrón predecible y estructurado (el proceso) y la meta principal de estos estudios es la construcción y la demostración de teorías.”

4.5. Unidad de estudio

La unidad de estudio fue el algoritmo de navegación autónoma.

4.6. Técnicas e instrumentos de recolección de datos

Las técnicas utilizadas para la recolección de datos fueron: revisión de trabajos de grados relacionados en el área; investigación en libros y revistas científicas (ya sean impresos o virtuales); asistencia a cursos referentes al área; revisión de manuales y algoritmos ya hechos; simulaciones en herramientas software; y pruebas físicas con el vehículo eléctrico a escala.

Capítulo 5

Simulación del Algoritmo

El presente capítulo, se muestra el código realizado en el entorno MATLAB. Las simulaciones se llevaron a cabo con el fin de comprender mejor el algoritmo de persecución pura. Así mismo, se presentan los resultados de las mismas.

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. Dicho software posee el Robotics System Toolbox, una herramienta que proporciona algoritmos y conectividad de hardware para desarrollar aplicaciones robóticas autónomas para vehículos aéreos y terrestres, manipuladores y robots humanoides [The MathWorks, Inc, s/f (b)].

Para esta experiencia, fue utilizado el toolbox mencionado y un código proporcionado por The MathWorks, Inc. A seguir, se procede a demostrar cómo funciona el algoritmo persecución pura. Se simuló como un robot navega por diversas trayectorias utilizando la propiedad distancia anticipada de dicho algoritmo. En la simulación, se variaron los valores de velocidad y distancia anticipada, así como las trayectorias a seguir.

5.1. Procedimientos

Como se mencionó anteriormente, para la simulación se utilizó el código de The MathWorks, Inc, [s/f (c)], en el cual, se presenta el comportamiento del algoritmo persecución pura. A dicho código, se le adjunto el mapa deseado.

5.1.1. Definir condiciones iniciales

Para iniciar la simulación, se debe definir las condiciones iniciales. La primera es establecer la ruta que el robot debe seguir, en otras palabras, proporcionarle los puntos del camino, así como se muestra en la Figura 14. Para obtener los puntos de la ruta, se realiza un plano del mapa a ser utilizado, se fijan los puntos en la línea central de la pista y luego se escalan los valores, teniendo en cuenta los parámetros del gráfico definido en la sección 5.1.2.

Figura 14 Puntos de la ruta del robot

```
1 %% Condiciones iniciales
2 % Ruta del robot (waypoints)
3 - path = [7.48    5.34;
4          7.37    6.52;
5          7.48    6.95;
6          7.70    7.48;
7          8.02    8.02;
8          8.55    8.55;
9          9.62    9.62;
10         10.69   10.69;
11         11.54   11.76;
12         11.76   12.83;
13         11.76   13.89;
14         10.69   15.18;
15         8.55    15.18; % Termina Columna 1
16         7.48    13.89;
17         6.73    12.83;
18         6.41    11.76;
19         5.13    9.62;
20         4.17    8.55;
21         2.57    5.34;
22         2.14    4.28;
23         2.14    3.21;
24         3.21    2.35;
25         5.34    2.35;
26         8.55    2.35;
```

Fuente: Elaboración propia.

Luego, se definen (Figura 15):

- La ubicación donde el robot comenzará a moverse: se estableció como ubicación el primer punto definido en la matriz de ruta.
- La ubicación donde el robot debe llegar, es decir, el punto de meta: se estableció como ubicación de meta el último punto de la matriz de ruta.
- La orientación inicial del robot, en otras palabras, el ángulo con respecto al eje x hacia donde se dirigirá el robot: se fija el ángulo de dirección del primer punto al segundo punto.
- La POSE inicial, dicho de otro modo, la posición y orientación del robot, eje “x”, eje “y” y el ángulo de dirección “ θ ”: se determina como POSE inicial la ubicación inicial del robot y la orientación inicial.

Figura 15 Condiciones iniciales respecto a ubicación y POSE

```
61 % Definir la ubicación actual del robot
62 - robotCurrentLocation = path(1,:);
63
64 % Definir el punto de meta del robot
65 - robotGoal = path(end,:);
66
67 % Definir la orientación inicial del robot
68 - initialOrientation = 90;
69
70 % Definir la pose actual (x y theta)
71 - robotCurrentPose = [robotCurrentLocation initialOrientation];
```

Fuente: Elaboración propia.

5.1.2. Configuración del robot

En este ejemplo se utiliza un simulador de robot simple que actualiza y devuelve la POSE del robot de accionamiento diferencial para las entradas de control dadas [The MathWorks, Inc, s/f (b)].

Figura 16 Configuración del robot

```
73 %% Configuración del robot
74 % Radio del robot
75 - robotRadius = 0.4;
76
77 % Definir el mapa a utilizar y la resolución de la misma
78 - robot = ExampleHelperRobotSimulator('emptyMap',2);
79
80 % Habilitación del laser (sensor)
81 - robot.enableLaser(false);
82
83 % Definir el tamaño del robot
84 - robot.setRobotSize(robotRadius);
85
86 % Mostrar la trayectoria
87 - robot.showTrajectory(true);
88
89 % Establecer la pose del robot
90 - robot.setRobotPose(robotCurrentPose);
91
92 % Gráfico de la simulación
93 - plot(path(:,1), path(:,2),'k--d')
94 - xlim([0 25]) % Se define los límites de x
95 - ylim([0 17.67]) % Se define los límites de y
```

Fuente: Elaboración propia.

Para la configuración del robot, Figura 16, se establece el radio del robot a simular, en esta simulación, se mantuvo el valor del código ejemplo. En el caso del mapa, se asignó *emptyMap*, dicha opción proporciona un mapa en blanco, esto es debido a que se adjuntará el mapa creado al término de la simulación de MATLAB; y una resolución de 2, es decir, cada cuadro del mapa tendrá una longitud de 2 metros, este valor se variará dependiendo del mapa que se necesite.

Example Helper Robot Simulator es una clase de MATLAB, el fin del mismo es utilizarlo en demostraciones de algoritmos de navegación, son utilizados en diversos ejemplos. Esta clase crea un simulador simple para un robot de accionamiento diferencial (The MathWorks, Inc, 2016).

Este ejemplo tiene la capacidad de ser utilizado con sensor ultrasonido virtual, dicho de otra manera, si se habilita el robot.enableLaser, puede navegar evitando los obstáculos asignados en el mapa. En esta simulación no se habilitará la función mencionada.

Seguidamente, se define el tamaño del robot, se permite observar el trayecto que el robot recorre, se establece la POSE del robot y se fija los parámetros del gráfico donde se mostrará el camino seguido por el robot y el comportamiento del mismo con las distintas velocidades y distancias anticipadas.

5.1.3. Controlador de seguidor de ruta

En esta sección se expone la propiedad del seguidor de ruta a utilizar, en el caso de esta simulación, es el controlador persecución pura, Figura 17.

Se establece la trayectoria que el robot debe seguir, el cual, fue definido al inicio del código. Se designan los valores de la velocidad lineal, la máxima velocidad angular y la distancia anticipada. Estos valores son modificados para el análisis del comportamiento del algoritmo.

Figura 17 Controlador de seguidor de ruta

```
%% Controlador de seguidor de ruta
% Definir el controlador Pure Pursuit o Persecución Pura
controller = robotics.PurePursuit;

% Definir los waypoints
controller.Waypoints = path; %Se definió al inicio de la simulación

% Establecer el valor de la velocidad lineal
controller.DesiredLinearVelocity = 2.5;

% Establecer el valor de la máxima velocidad angular
controller.MaxAngularVelocity = 5;

% Establecer el valor del lookahead distance o distancia anticipada
controller.LookaheadDistance = 3;
```

Fuente: Elaboración propia.

5.1.4. Funcionamiento del controlador persecución pura

En este apartado, se programa el funcionamiento del algoritmo. Para la simulación, el ejemplo proporcionado por MathWorks, [The MathWorks, Inc, s/f (c)], no fue modificado, Figura 18.

El radio de meta, es la distancia deseada entre la ubicación final del robot y la ubicación de meta, cuando el robot se encuentre a esa distancia del punto de meta, se detendrá.

La ejecución es realizada cada 10 Hz y se ejecuta hasta que la distancia entre el robot y el punto de meta sea menor al valor asignado al radio de meta. Mientras esta condición no sea cumplida, el algoritmo calcula los valores de velocidad lineal “ v ” y velocidad angular “ ω ”. Dichos valores proporcionan las condiciones necesarias al robot para que el mismo pueda desplazarse. Posteriormente, se actualiza el valor de la POSE y la distancia al punto de meta.

Figura 18 Controlador persecución pura

```
113 % Conducción del robot
114 % Definir un radio de meta
115 - goalRadius = 0.1; %Distancia deseada entre la ubicación final y la de meta
116
117 % Distancia actual
118 - distanceToGoal = norm(robotCurrentLocation - robotGoal);
119
120 % Frecuencia de ejecución del controlador
121 - controlRate = robotics.Rate(10);
122
123 % Recorrido del robot hasta llegar al radio de meta
124 - while( distanceToGoal > goalRadius )
125
126     % Calcula las salidas del controlador, es decir, las entradas del robot
127 -     [v, omega] = controller(robot.getRobotPose);
128
129     % Simula el robot usando las salidas del controlador
130 -     drive(robot, v, omega);
131
132     % Extrae la información actual de la ubicación ([X,Y]) de la pose
133     % actual del robot
134 -     robotCurrentPose = robot.getRobotPose;
135
136     % Re-calcula la distancia a la meta
137 -     distanceToGoal = norm(robotCurrentPose(1:2) - robotGoal);
138
139 -     waitfor(controlRate);
140
141 - end
142
143 % Cerrar simulación
144 - delete(robot)
```

Fuente: Elaboración propia.

Para finalizar, se tiene la posibilidad de cerrar la ventana de simulación. Si se desea mantener abierto, se elimina la función *delete(robot)* de la línea 144 o simplemente se pone un % delante de ella.

5.2. 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

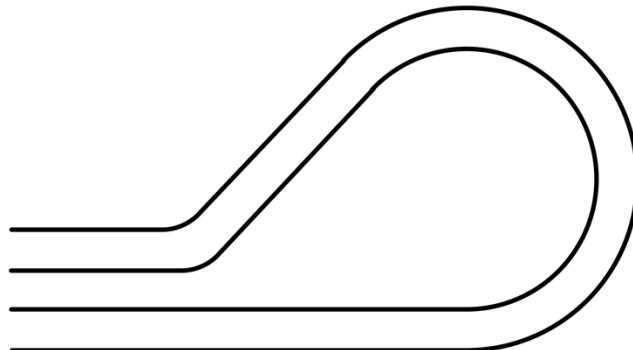
velocidad y distancias anticipadas, no serán mostradas todas las imágenes tomadas y si una tabla al final de cada sección de los mapas con los análisis respectivos.

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.

5.2.1. 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, Figura 19.

Figura 19 Pista Tacurú Pucú

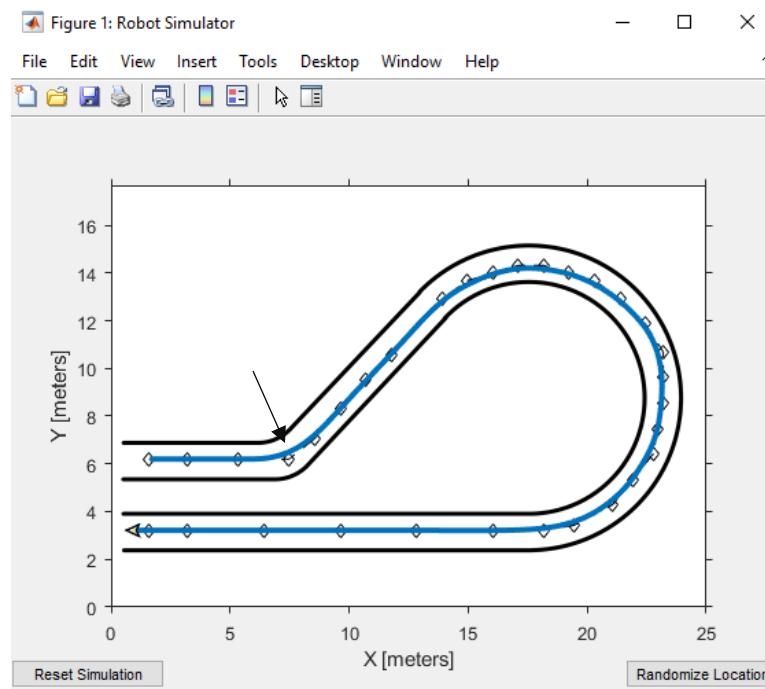


Fuente: Elaboración propia.

A continuación, se muestran las simulaciones realizadas con una velocidad de 2 m/s y distancias anticipadas diferentes, Figura 20, Figura 21 y Figura 22. En la Figura 20, 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 (flecha), no son grandes, son difícilmente notorias.

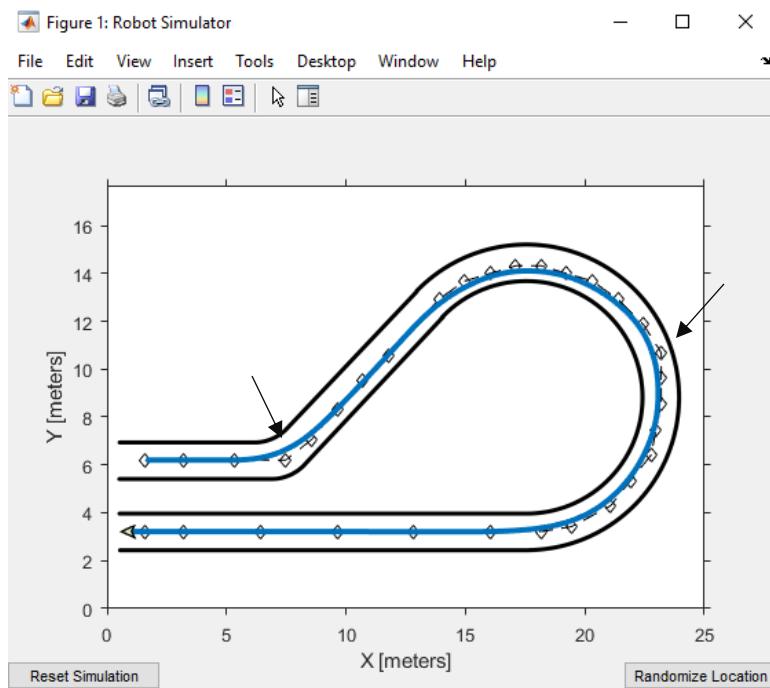
Figura 20 Velocidad = 2 m/s. Distancia Anticipada = 1.8 m



Fuente: Elaboración propia.

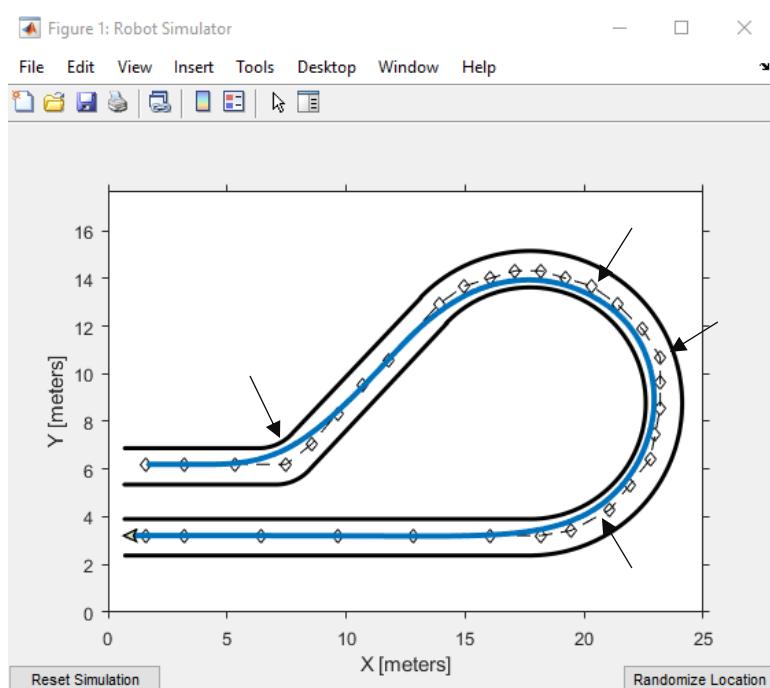
En la Figura 21, el valor de la distancia anticipada es 2.5 metros. En él, se observa que el robot no pasa por algunos de los puntos fijados en las curvas del mapa. En cambio, en la Figura 22, donde la distancia anticipada es 3.5 metros, prácticamente solo en los trayectos rectos el robot pasa por los 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.

Figura 21 Velocidad = 2 m/s. Distancia Anticipada = 2.5 m



Fuente: Elaboración propia.

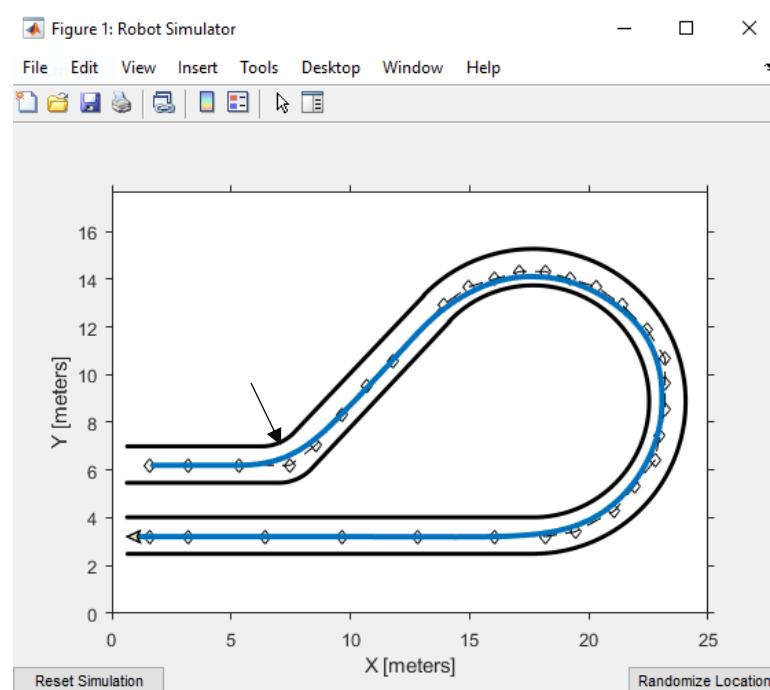
Figura 22 Velocidad = 2 m/s. Distancia Anticipada = 3.5 m



Fuente: Elaboración propia.

En las siguientes figuras: Figura 23 y Figura 24, la velocidad del robot es de 3 m/s. En la Figura 24, se observa el trayecto recorrido con la distancia anticipada igual a 2.5 metros. Al igual que en la Figura 21, el robot no pasa por todos los puntos.

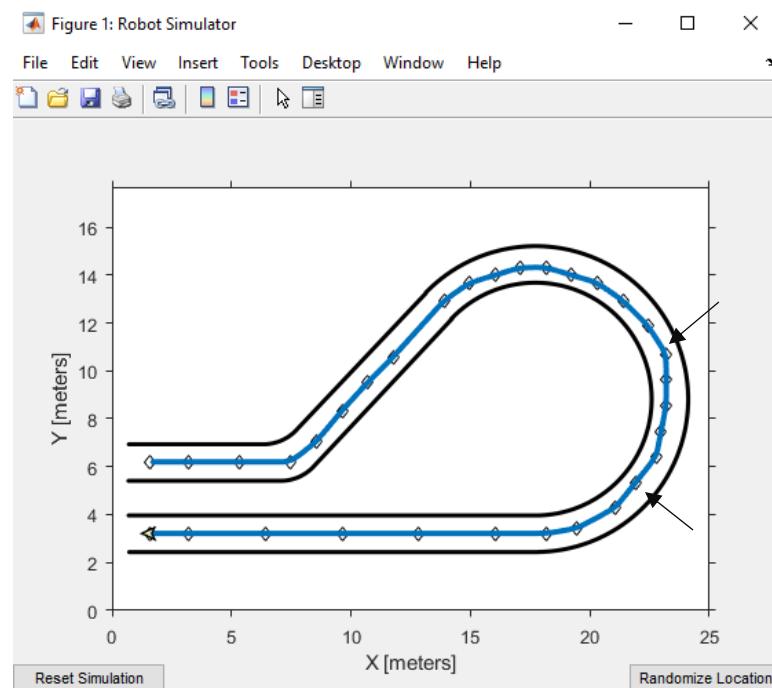
Figura 23 Velocidad = 3 m/s. Distancia Anticipada = 2.5 m



Fuente: Elaboración propia.

En la Figura 24, 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.

Figura 24 Velocidad = 3 m/s. Distancia Anticipada = 0.5 m



Fuente: Elaboración propia.

5.2.1.1. 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

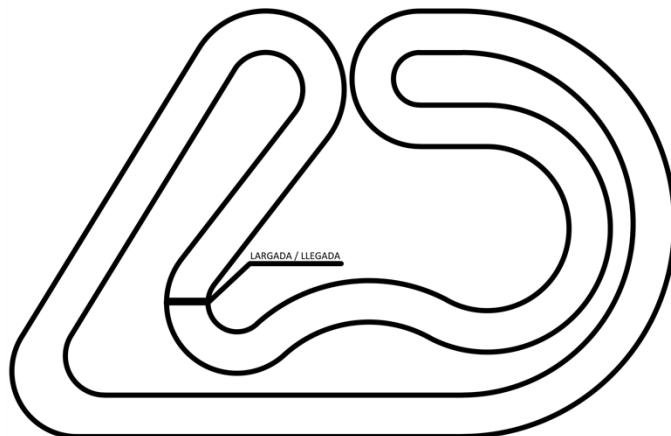
Velocidad (m/s)	Rango de valores de la distancia anticipada (m)
2	1.8 – 3
3	2.2 – 3.2
4	2.5 – 3.2

Fuente: Elaboración propia.

5.2.2. Segundo mapa: Pista Robocar Race 2018

En la Figura 25 se observa el diseño de la pista de la competencia. Luego de ser compartido con los participantes, se recurrió al Arq. Jorge Villanueva, docente de la institución, quién escaló la imagen con el fin de realizar simulaciones y pruebas.

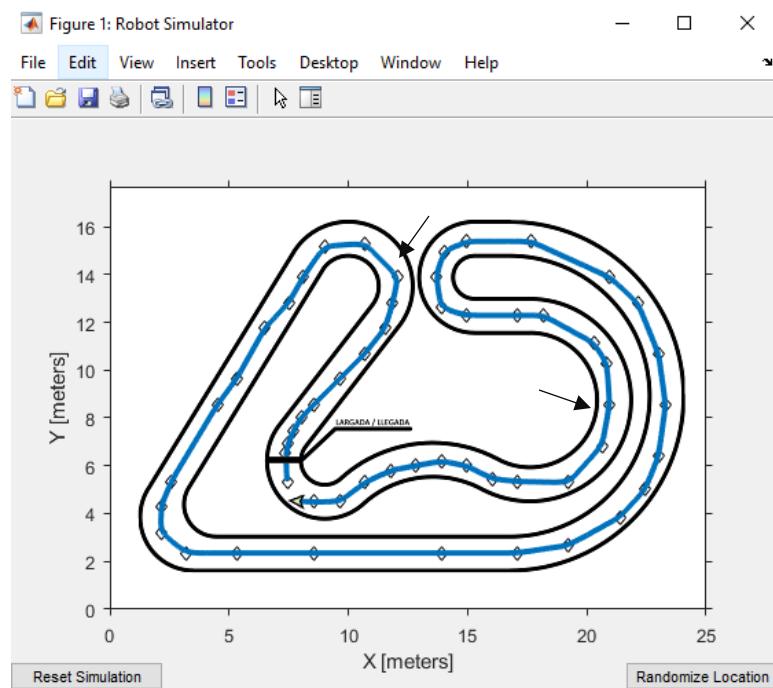
Figura 25 Pista Robocar Race 2018



Fuente: Robocar Race, (2018). Modificado por el Arq. Jorge Villanueva.

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

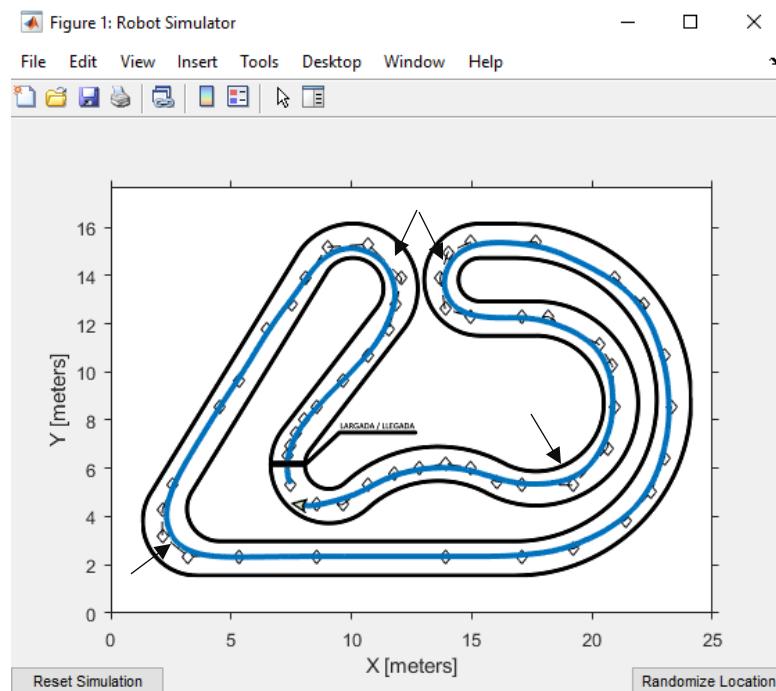
Figura 26 Velocidad = 2.5 m/s. Distancia Anticipada = 0.5 m



Fuente: Elaboración propia.

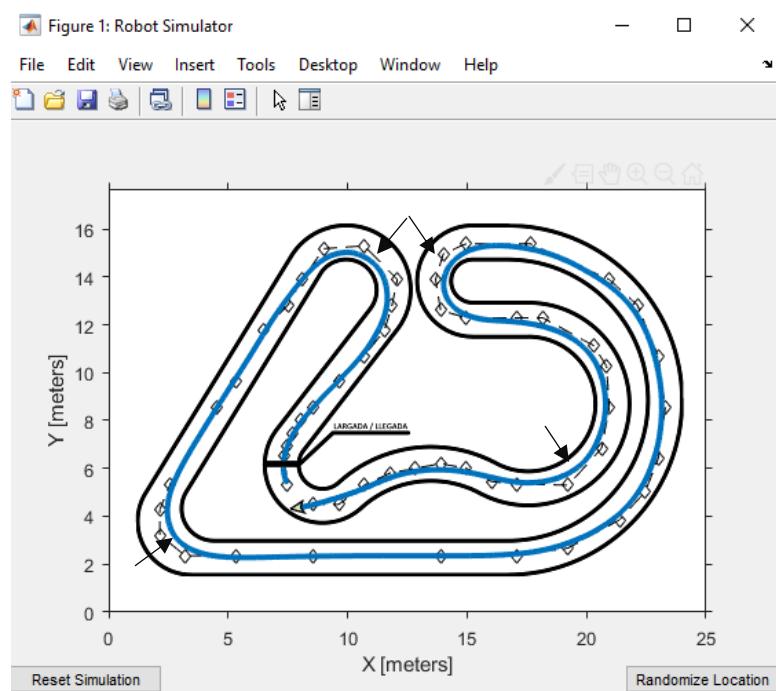
En la Figura 26, como en la Figura 24, posee una distancia anticipada pequeña, igual a 0.5 m y el efecto es el mismo, se producen pequeñas oscilaciones mientras el robot pasa de un punto a otro. A diferencia de la Figura 24, en este se puede observar mejor el efecto producido por pequeñas distancias anticipadas.

Figura 27 Velocidad = 2.5 m/s. Distancia Anticipada = 1.7 m



Fuente: Elaboración propia.

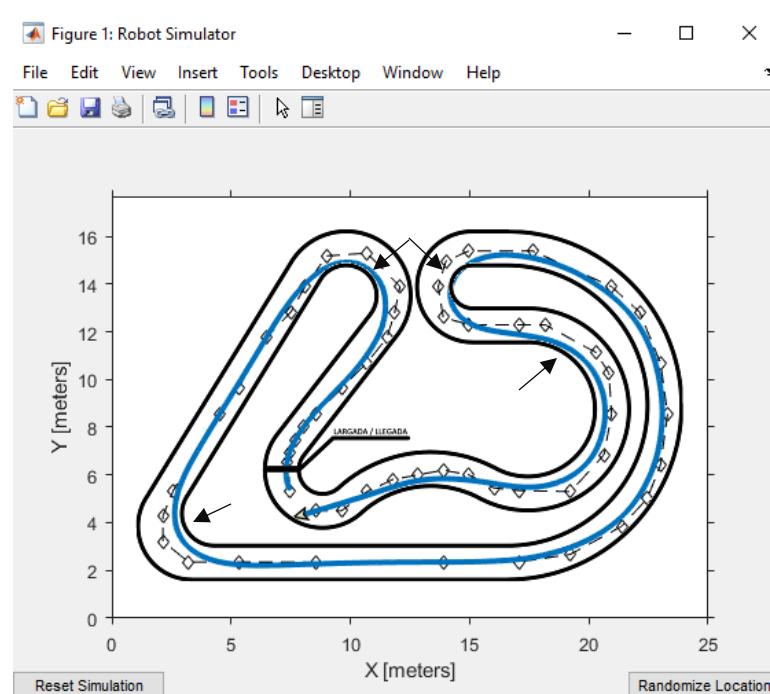
Figura 28 Velocidad = 2.5 m/s. Distancia Anticipada = 2.3 m



Fuente: Elaboración propia.

En la Figura 27 se observa como el algoritmo funciona con una distancia anticipada de 1.7 m. A partir de este valor, dicho algoritmo muestra un óptimo rendimiento. En la Figura 28, 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.

Figura 29 Velocidad = 2.5 m/s. Distancia Anticipada = 3 m



Fuente: Elaboración propia.

En la Figura 29 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.

5.2.2.1. Análisis

Al igual que en la sección 5.2.1.1. de este capítulo, se realizó un análisis del funcionamiento del algoritmo en la pista Robocar Race 2018. 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

Fuente: Elaboración propia.

5.3. Conclusión del capítulo

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.

Capítulo 6

Diseño e Implementación del Algoritmo

En el presente capítulo, se muestra el diseño del algoritmo. Además, se explica la teoría del código utilizado para la implementación en el vehículo eléctrico a escala. El mismo, fue implementado en el sistema embebido myRIO utilizando el ambiente de programación LabVIEW.

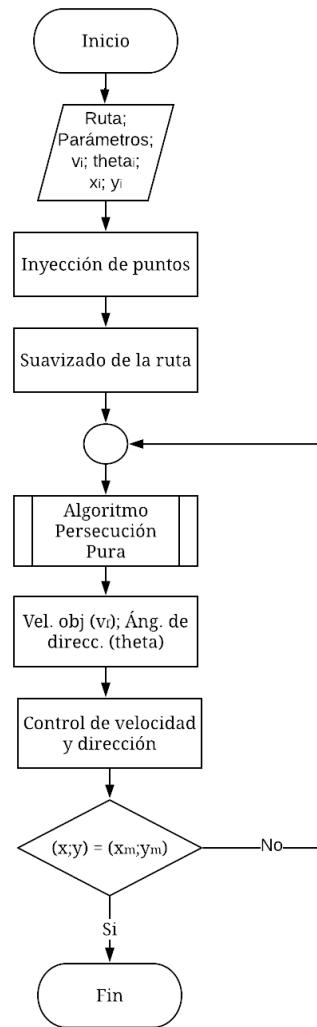
La implementación del algoritmo persecución pura fue realizada a través de los productos de National Instruments, el sistema embebido myRIO y el software LabVIEW. myRIO es un sistema embebido para estudiantes que ofrece la posibilidad de aprender de manera práctica sobre mecatrónica, sistemas de control, entre otros [National Instruments, s/f(a)]. El myRIO se programa a través de LabVIEW, dicho software se enfoca en la programación gráfica, con ello, ayuda a visualizar cada aspecto de la aplicación, esto hace que sea más fácil integrar hardware de medidas de cualquier proveedor, representar una lógica compleja en el diagrama, desarrollar algoritmos de análisis de datos y diseñar interfaces de usuario personalizadas [National Instruments, s/f(b)].

El código a ser implementado en el myRIO es una modificación del algoritmo del Team 1712 (Team 1712, 2018). 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 de este libro de tesis.

6.1. Diseño del algoritmo

El código programado por el Team 1712 posee muchas secciones que no son necesarias para este proyecto. Debido a ello, en la Figura 30, se presenta el diagrama de flujo del algoritmo adaptado que fue implementado en el auto eléctrico a escala.

Figura 30 Diagrama de flujo del código fuente del algoritmo adaptado

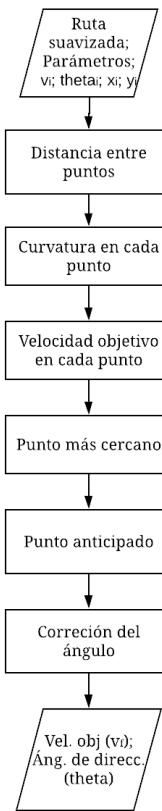


Fuente: Elaboración propia.

Una vez que el código es iniciado, se fija la ruta de puntos que servirá de guía para la navegación, así como parámetros que serán explicados posteriormente. 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. El algoritmo persecución pura está compuesto por los módulos presentados en la Figura 31. 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.

Figura 31 Diagrama de flujo del código fuente del algoritmo persecución pura



Fuente: Elaboración propia.

6.2. Algoritmo de navegación

En esta sección, se explican detalladamente cada módulo de los diagramas presentados en la Figura 30 y Figura 31.

Tomando como guía el artículo presentado por el Team 1712, (Team 1712, 2018), la implementación del algoritmo posee tres etapas: la odometría, generación del camino y el seguimiento del mismo.

El código en LabVIEW, puede ser encontrado en el Apéndice B.

6.2.1. Odometría

La odometría se utiliza para estimar los puntos por los cuales el vehículo debe pasar, en otras palabras, se calcula los valores de x e y de los puntos del camino de puntos (*waypoints*). Los valores x e y se obtienen de las siguientes fórmulas:

$$x_n = x_{n-1} + \cos \theta \quad 6.1.$$

$$y_n = y_{n-1} + \sin \theta \quad 6.2.$$

x_n = x actual

x_{n-1} = x anterior

y_n = y actual

y_{n-1} = y anterior

θ = ángulo de dirección

6.2.2. Generación del camino

En la sección anterior se guardaron los datos para el camino de puntos, en esta sección el camino de puntos será extraído del documento de texto, luego, utilizando el software del Team 1712, dicho software introducirá más puntos de referencia en el camino y lo suavizará. Seguidamente, se calculará y almacenará las siguientes informaciones para cada punto del camino:

- Distancia del punto a lo largo del camino.
- La velocidad que debe poseer el auto en ese punto.
- Curvatura del camino en ese punto.

Cuando se crea la ruta, el usuario debe establecer los máximos valores de velocidad y aceleración al cual puede llegar a alcanzar el auto (Team 1712, 2018).

6.2.2.1. Inyección de puntos

La precisión de los puntos de ajustes de la velocidad y las medidas de curvatura aumenta cuanto más cercanos sean los puntos espaciales, asimismo, el algoritmo de suavizado, desarrollado por Harrilal, (2014), se desempeña mejor con más puntos. Dicho algoritmo de inyección calcula, para cada segmento de línea de la ruta, cuántos puntos espaciados uniformemente cabrían dentro de cada uno de ellos (incluido el punto de inicio, excluyendo el punto final), y luego inserta los debidos puntos a lo largo del segmento. Cuando llega al final agrega el punto de meta a la lista de puntos. A seguir el pseudocódigo hecho por el Team 1712.

```
espacio = distancia deseada entre los puntos  
nuevos_puntos = vector donde serán puestos los nuevos puntos
```

- Para cada segmento de línea
 - vector = punto_final - punto_inicial
 - num_puntos_que_caben = Math.ceil(vector.magnitud()/espacio)
 - vector = vector.normalizado() * espacio
 - para (i = 0; i < num_puntos_que_caben; i++):
 - Sumar (punto_inicial + vector * i) a nuevos_puntos
- Agregar el punto de meta a nuevos_puntos

6.2.2.2. *Suavizado de la ruta*

Para suavizar el camino, fue utilizado el algoritmo del Team 2168. En el artículo, (Team 1712, 2018), se explica que dicho algoritmo toma una matriz 2D de coordenadas XY y devuelve una versión más suave de las coordenadas. Toma tres parámetros: *weight_data* (al cual se le llama *A*), *weight_smooth* (al cual se le llama *B*) y tolerancia. La cantidad que este algoritmo suaviza depende del espaciado entre los puntos y los valores de *A*, *B* y tolerancia. Cuanto más grande el valor de *B*, más suave será el camino.

El autor del algoritmo de suavizado, (Harrilal, 2014), explica que el parámetro *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. El parámetro *B* controla las curvas que compondrán el camino, dicho de otra forma, qué curvas se considerarán para el camino. La tolerancia define la región dentro de la cual se tienen en cuenta todas las coordenadas. Harrilal añade que cuanto más cerca esté el parámetro *A* de 1, más cerca estará el camino suave de la línea original, y cuanto más cerca esté *B* de 1, mayor será el radio del camino. El autor advierte que estos parámetros son requisitos conflictivos, y por eso es un algoritmo de optimización, un ejemplo sería que, al aumentar el valor de *A*, se empuja los puntos hacia la ruta original, y en otra manera, si se aumenta *B*, se aleja los

puntos de la línea original y hay una lucha constante entre ellos. El algoritmo converge cuando ambos coinciden con el lugar donde se empujan los puntos.

En la Figura 32 se puede observar el código fuente del suavizado hecho por Harrilal, (2014). Dicho código es implementado por el Team 1712 en LabVIEW, el cual puede ser visualizado en el Apéndice B.

Figura 32 Código fuente del suavizado

```
public double[][] smoother(double[][] path, double a, double b, double tolerance){

    //copy array
    double[][] newPath = doubleArrayCopy(path);

    double change = tolerance;
    while(change >= tolerance)
    {
        change = 0.0;
        for(int i=1; i<path.length-1; i++)
            for(int j=0; j<path[i].length; j++)
            {
                double aux = newPath[i][j];
                newPath[i][j] += a * (path[i][j] - newPath[i][j]) + b *
                (newPath[i-1][j] + newPath[i+1][j] - (2.0 * newPath[i][j]));
                change += Math.abs(aux - newPath[i][j]);
            }
    }
    return newPath;
}
```

Fuente: Harrilal, (2014).

Los valores de A y B deben estar entre 0 y 1. En la Tabla 4 se puede observar los valores recomendados por el autor. Para el algoritmo de suavizado, los valores por defecto son: A igual a 0.7 y B igual a 0.3, estos valores se mantuvieron en la implementación.

Tabla 4

Valores recomendados para los parámetros A y B

A	B
Cualquier valor entre 0 y 1	0.0 – 0.4
Cualquier valor entre 0 y 0.9	0.5
Cualquier valor entre 0 y 0.7	0.6
Cualquier valor entre 0 y 0.5	0.7
Cualquier valor entre 0 y 0.3	0.8
Cualquier valor entre 0 y 0.1	0.9
Ningún valor	1.0

Fuente: Harrilal, (2014).

6.2.2.3. Distancia entre los puntos

La distancia de los puntos deberá de ser almacenado, puesto que, es utilizado para calcular la curvatura y la velocidad que debe tener el auto en cada punto. Esta distancia no es necesariamente igual al espaciado utilizado en el algoritmo de inyección porque los puntos se mueven cuando se suaviza la ruta (Team 1712, 2018).

La distancia del punto actual es igual a la distancia del punto anterior más la distancia entre el punto anterior y el actual, expresando en forma matemática se tiene:

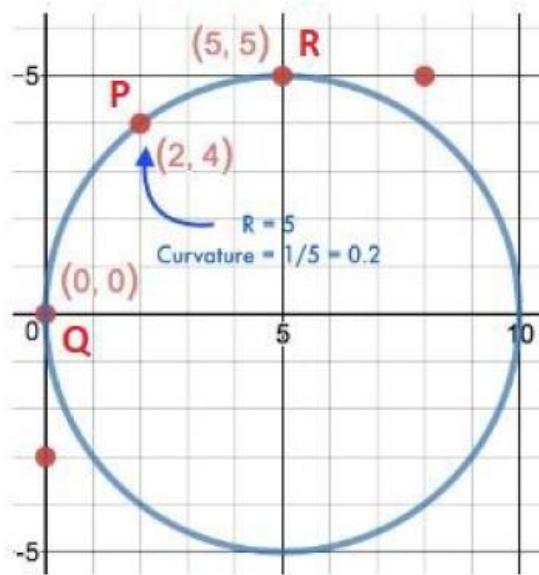
$$dist_punto_i = dist_punto_{(i-1)} + dist_formula [punto_i, punto_{(i-1)}] \quad 6.3.$$

6.2.2.4. Curvatura del camino

La curvatura de la trayectoria en un punto se utiliza para ajustar la velocidad del robot. Tomando como ejemplo, si es que el auto realiza un giro brusco (curvatura grande), él debe disminuir la

velocidad (Team 1712, 2018). Para calcular la curvatura se utiliza la ecuación 3.6. donde la curvatura es igual a la inversa del radio. Team 1712 explica que es posible encontrar el radio del círculo que interseca el punto y los dos puntos a cada lado de la siguiente forma:

Figura 33 Método de los 3 puntos



Fuente: Team 1712, (2018).

Dado tres puntos $P(x_1; y_1)$, $Q(x_2; y_2)$ y $R(x_3; y_3)$, Figura 33, se encuentra el centro $C(a; b)$ y el radio r que se interseca con los tres puntos mencionados anteriormente, donde:

$$r = |PC| = |QC| = |RC| \quad 6.4.$$

La solución es:

$$k_1 = 0.5 * (x_1^2 + y_1^2 - x_2^2 - y_2^2) / (x_1 - x_2) \quad 6.5.$$

$$k_2 = (y_1 - y_2) / (x_1 - x_2) \quad 6.6.$$

$$b = 0.5 * \frac{x_2^2 - 2 * x_2 * k_1 + y_2^2 - x_3^2 + 2 * x_3 * k_1 - y_3^2}{x_3 * k_2 - y_3 + y_2 - x_2 * k_2} \quad 6.7.$$

$$r = \sqrt{(x_1 - a)^2 + (y_1 - b)^2} \quad 6.8.$$

Al hallar r , solo es necesario aplicar la fórmula 3.6. que la curvatura de la trayectoria en ese punto ya es conocido.

El Team 1712 recomienda agregar un valor suficientemente pequeño (0.001) a x_l , puesto que, cuando $x_l = x_2$, se obtiene un error de división por cero. La fórmula entonces da una aproximación muy cercana sin ningún error. En el caso que el resultado sea NaN, se tiene una línea recta en la trayectoria, dado que el radio es infinito y la curvatura es 0.

La curvatura en cada punto se encuentra utilizando las fórmulas mencionadas recientemente. Se toma cada punto del trayecto y los puntos a cada lado de este. Los puntos inicial y final poseen una curvatura de 0 (Team 1712, 2018).

6.2.2.5. *Velocidades*

Parte 1: Velocidad máxima.

Para cada punto en el trayecto, el auto intentará alcanzar una velocidad objetivo. 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. Para reducir la velocidad del auto alrededor de los giros, se establece la velocidad de cada punto en el valor mínimo de (velocidad máxima de trayectoria, $k / \text{curvatura}$ en el punto), donde k es una constante de 1 a 5, según la lentitud con la que desee que el robot gire, esto, además de reducir la

velocidad, asegura que la velocidad objetivo del auto nunca esté por encima del máximo (Team 1712, 2018).

Parte 2: Límite de aceleración.

Para que el auto no acelere bruscamente, al comienzo se agregó un suavizador para la aceleración. Sin embargo, como el valor de la velocidad en el punto inicial es 0, el auto no sería capaz de moverse, esto llevó a que los programadores eliminan la aceleración suave y solo mantengan el de la desaceleración. Pero, en consecuencia, las velocidades objetivo ya no obedecen a la aceleración máxima, para solucionar este problema se necesitará un limitador de velocidad. En tiempo real, a medida que el auto recorre el camino, el limitador de velocidad restringe la velocidad de cambio de la velocidad objetivo de entrada (Team 1712, 2018).

Parte 3: Cálculos.

Para calcular la velocidad objetivo se aplica la ecuación de Torricelli, donde se considera que el auto posee la aceleración máxima desde el final al comienzo, con su velocidad limitada por las velocidades máximas calculadas en la parte 1. Dicho esto, la velocidad objetivo en un punto es el mínimo de: la velocidad actual del punto y la velocidad más alta que el robot puede alcanzar en ese punto cuando comienza en el último punto (Team 1712, 2018).

$$v_f = \sqrt{v_i^2 + 2 * a * d} \quad 6.9.$$

v_f = velocidad máxima que se puede alcanzar en un punto

v_i = velocidad del punto anterior

a = aceleración máxima establecida

d = distancia entre los puntos

Limitador de velocidad.

Team 1712 explica que la salida del limitador de velocidad intenta obtener el mismo valor que la entrada, pero limitando a la rapidez con la que puede cambiar, teniendo como entrada el valor que se desea limitar y la velocidad máxima de cambio para la misma. La función Restricción limita el primer parámetro para estar en el rango de los dos segundos.

6.2.3. Seguimiento del camino

Tomando como guía el artículo de Team 1712, el seguimiento del camino consta de las siguientes partes.

6.2.3.1. Encontrar el punto más cercano

Para hallar el punto más cercano, se debe calcular la distancia del punto a todos los demás puntos y tomar la más pequeña.

6.2.3.2. Encontrar el punto anticipado

El punto anticipado 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 (Team 1712, 2018). Para encontrar los puntos de intersección, se calcula dos valores de t , donde t varía de 0 a 1 y

representa proporcionalmente a qué distancia del segmento está el punto de intersección. Si se presentan los siguientes casos significa que el círculo no interseca el segmento: $t < 0$ o $t > 1$.

Para hallar el punto de intersección, el Team 1712 presenta la siguiente fórmula:

$$Punto = E + (valor\ de\ t\ en\ la\ intersección) * d \quad 6.10.$$

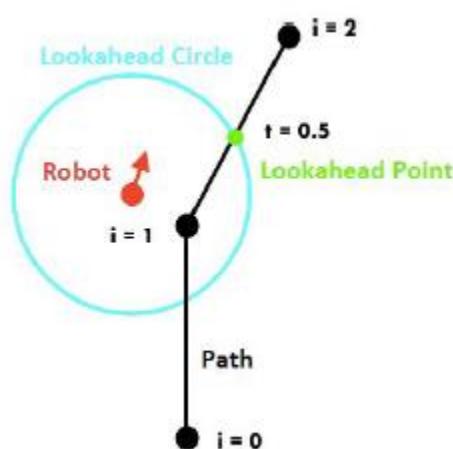
E = punto inicial del segmento de línea

L = punto final del segmento de línea

d = vector de dirección, de inicio a fin ($L - E$)

Por lo tanto, el valor que toma el punto anticipado es el valor t más el índice del punto de inicio del segmento. Como ejemplo, se puede observar la Figura 34, donde el valor del punto anticipado es 1.5.

Figura 34 Punto anticipado



Fuente: Team 1712, (2018).

El algoritmo diseñado por Team 1712, itera a través de los segmentos de línea de la ruta para ver si hay un punto de intersección válido ($0 \leq t \leq I$) entre el segmento y el círculo de búsqueda anticipada (círculo que posee como radio la distancia anticipada) con el fin de hallar el punto anticipado. El nuevo punto anticipado es la primera intersección válida cuyo subíndice es mayor que el índice del último punto de búsqueda anticipada. Esto asegura que el punto de anticipación nunca retroceda. En caso que el siguiente punto anticipado sea inválido, se utiliza el último punto anticipado válido.

6.2.3.3. Corrección del ángulo

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. El nuevo ángulo de dirección se obtiene de la siguiente forma:

$$\theta = (-1) * \tan^{-1}[(y_{dist.anticipada} - y_{auto}) / (x_{dist.anticipada} - x_{auto})] \quad 6.11.$$

$y_{dist.anticipada}$ = nuevo valor calculado de y .

y_{auto} = valor de y donde se encuentra el auto.

$x_{dist.anticipada}$ = nuevo valor calculado de x .

x_{auto} = valor de x donde se encuentra el auto.

6.2.3.4. Comunicación con los controles: de velocidad y de dirección

En esta parte, la velocidad calculada y el ángulo de dirección son comunicados a los controles de velocidad y de dirección, con el fin de realizar la navegación autónoma. Los datos son enviados

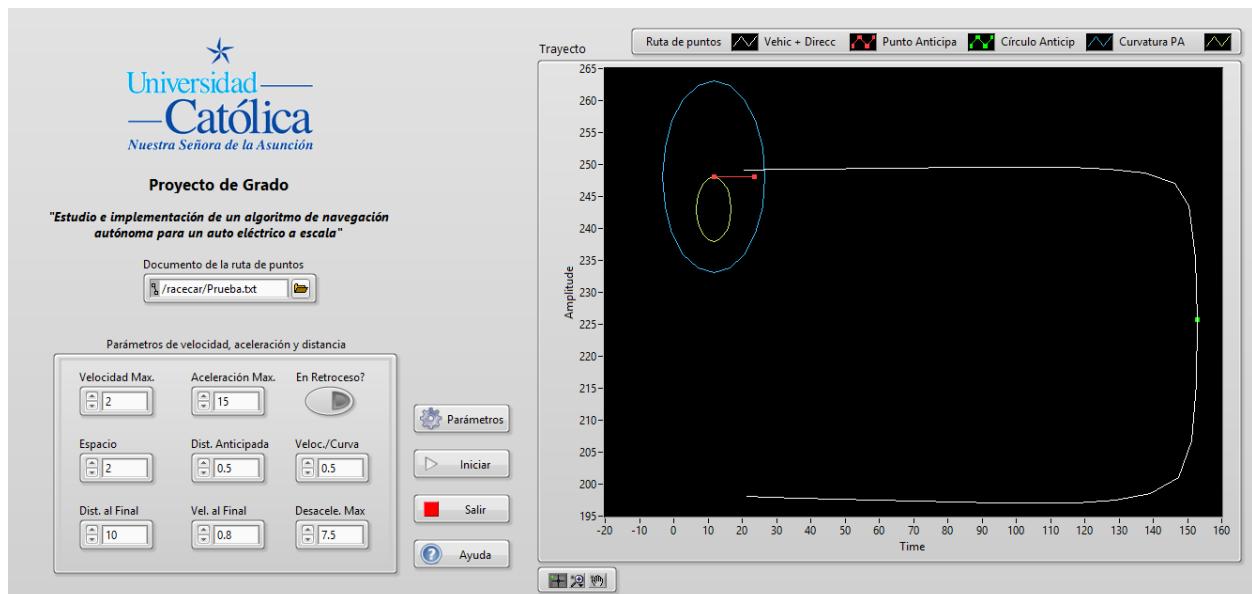
cada 52 ms a los controles, esto se debe al tiempo que lleva recabar la información de los sensores y procesarlos. Si el tiempo de iteración es menor, la navegación del auto no se realiza de manera correcta, causando fallas en los cálculos, y como consecuencia, el auto se estrella.

En el caso de realizar la simulación en LabVIEW, no son enviados los datos al control. Se simula un robot con el módulo LabVIEW Robotics, dicho modulo permite realizar simulaciones de un robot con las entradas y salidas necesarias. Para llevar a cabo las simulaciones, se mantuvo el código fuente propuesto por Team 1712, con pequeñas modificaciones en la interfaz de usuario.

6.2.4. Interfaz de usuario

En la Figura 35 se muestra la interfaz que el usuario observa cuando ingresa al VI (Virtual Instruments), (ver Glosario). En él, se puede acceder a la opción de ayuda, donde se explica cómo se puede poner en marcha la navegación.

Figura 35 Interfaz de Usuario del Algoritmo



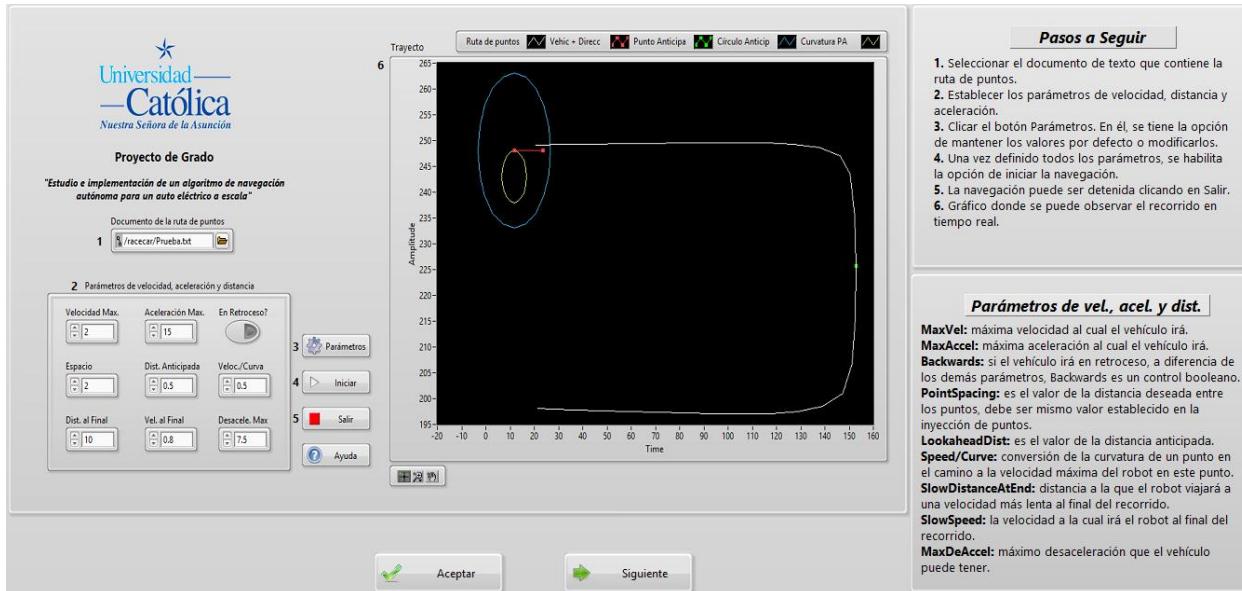
Fuente: Elaboración propia.

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.

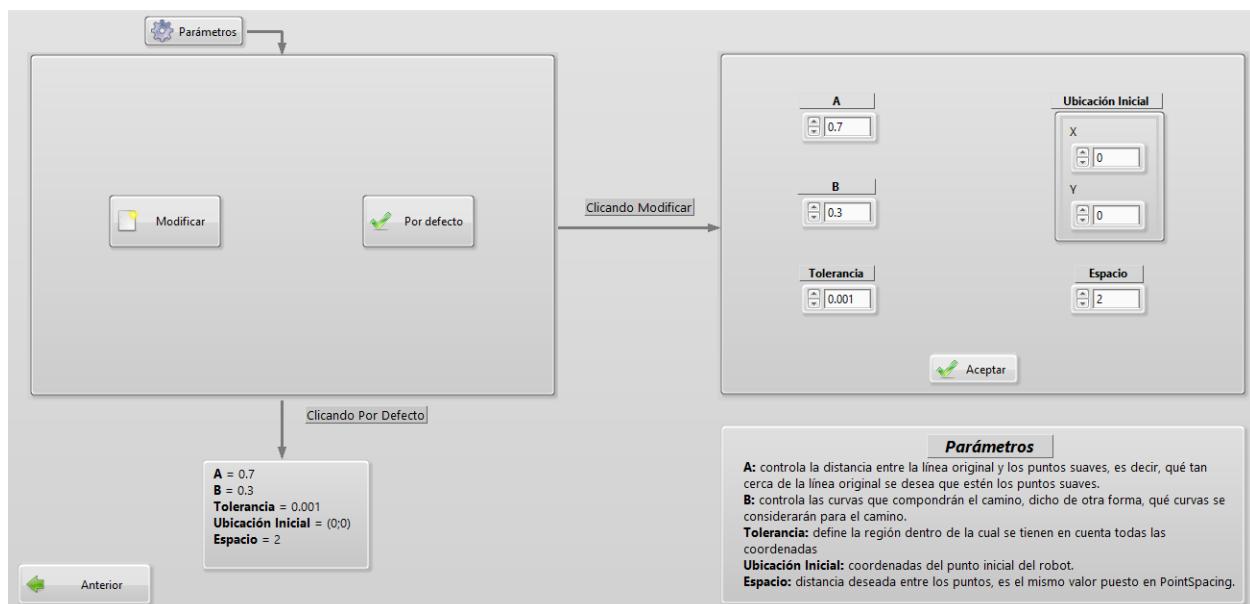
En la Figura 36 y Figura 37 se muestran las ventanas de Ayuda, en él, además de explicar los pasos a seguir, se explican los parámetros que deben ser definidos para la puesta en marcha.

Figura 36 Interfaz de Ayuda. Parte 1



Fuente: Elaboración propia.

Figura 37 Interfaz de Ayuda. Parte 2



Fuente: Elaboración propia.

6.3. Conclusión del capítulo

Reutilizar códigos de terceros y modificarlos según la necesidad, ayuda a ahorrar tiempo programando y comprobando la funcionalidad. Un código que tomaría un año programar, llevaría algunas semanas o meses estudiando y aprendiendo el funcionamiento si se reaprovecha un algoritmo de terceros. Estas técnicas son comunes actualmente, algoritmos ya existentes son modificados para ser utilizados de una manera diferente o son mejorados.

Gracias al reaprovechamiento de los códigos, las simulaciones pudieron ser realizadas en el tiempo de una semana y para la implementación, los involucrados llevaron dos semanas a aprender del mismo y modificarlo a conveniencia.

Capítulo 7

Comprobación del Sistema

En dicho capítulo, se muestran las pruebas y los resultados de la implementación del algoritmo en el vehículo eléctrico a escala.

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 Figura 38 se observa el vehículo eléctrico a escala modificado donde fue implementado el algoritmo persecución pura.

Figura 38 Auto eléctrico a escala modificado “Aguara’i”



Fuente: Foto tomada en la competencia Robocar Race 2018, por Luiz Celiberto.

Se llevaron a cabo distintas pruebas con valores diferentes de todos los parámetros especificados en el capítulo 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 5 y se variaron los demás parámetros mostrados en dicha tabla. La respuesta del vehículo a dichas variaciones fue óptima, el recorrido realizado por el mismo, era similar a la ruta de puntos establecida al inicio de la ejecución. Al no probar con altas velocidades, las variaciones realizadas en los parámetros de Aceleración Max. y Desacele. Max. no tuvieron mucho impacto. 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.

Los parámetros de Aceleración Max. y Desacele. Max. fueron probados en la simulación realizada en LabVIEW, donde se obtuvieron buenas respuestas, el desempeño del robot simulado fue óptimo, al igual que en la implementación, el trayecto recorrido por el robot se aproximó a la ruta de puntos fijada. Al establecer velocidades altas, el funcionamiento de estos parámetros 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 dicha velocidad.

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 5

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

Fuente: Elaboración propia.

El significado de cada uno de los parámetros de la Tabla 5 puede ser encontrado en el Apéndice B.

7.1. Simulación en LabVIEW

Cabe resaltar, que las simulaciones fueron realizadas con los datos recabados en la pista Robocar Race 2018. En dicha pista, interferencias electromagnéticas afectaron al magnetómetro, debido a ello, los valores de x e y , no fueron cien por ciento correctos. Sin embargo, la navegación se acercó a la ruta real que debía seguir, es decir, se repetían los errores de la recolección de datos en la navegación.

Debido a lo mencionado anteriormente respecto a los parámetros Espacio, A, B y Tolerancia, se decidió simular en LabVIEW el algoritmo, con el objetivo de poder visualizar las diferencias existentes al variar dichos parámetros.

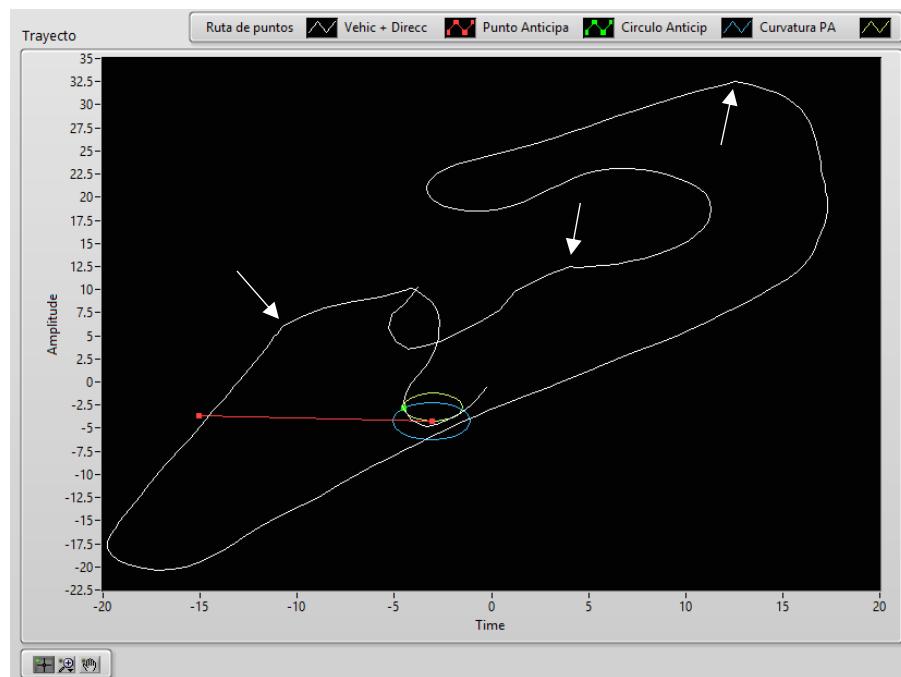
Al variar los valores de Espacio, no se vieron diferencias en la ruta de puntos. Así también, fue eliminado el subVI (ver Glosario), de Inyección de puntos, con el fin de saber si existe diferencia

en la ruta. Al eliminar, la ruta permaneció sin diferencias. Esto se debe a que, al recolectar los puntos cada 100 ms, véase Apéndice B, los puntos se encontraban bastante cercas unos de otros.

Tomando la misma idea con el subVI de Suavizado, el mismo fue eliminado. Sin embargo, la diferencia fue notoria. Al volver a conectar el subVI, se variaron los valores de los parámetros A, B y Tolerancia. La variación del valor de Tolerancia, no afectó en gran medida la ruta de puntos, en cambio, las variaciones en los parámetros A y B, si afectaron el camino.

En la Figura 39, se puede observar la ruta de puntos sin el subVI de Suavizado. En él, se muestra como la ruta posee pequeñas curvas bruscas (flecha blanca).

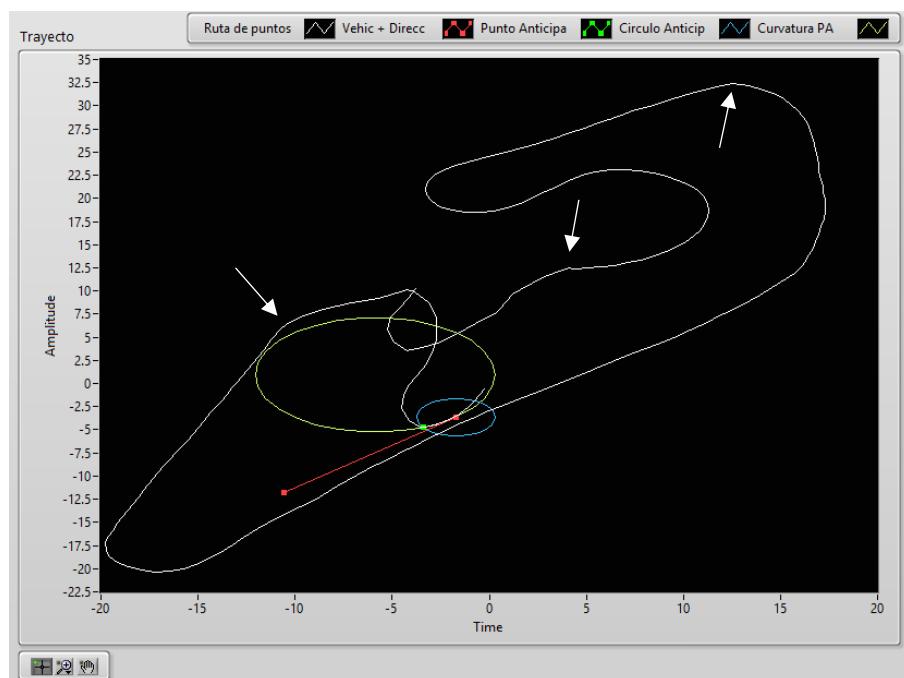
Figura 39 Ruta de puntos sin suavizado



Fuente: Elaboración propia.

Sin embargo, al conectar el subVI nuevamente, y fijar los valores como se muestra en la Figura 40, las curvas bruscas que se tenían en la figura anterior, se suavizaron sin alejarse de la ruta de puntos establecida al inicio.

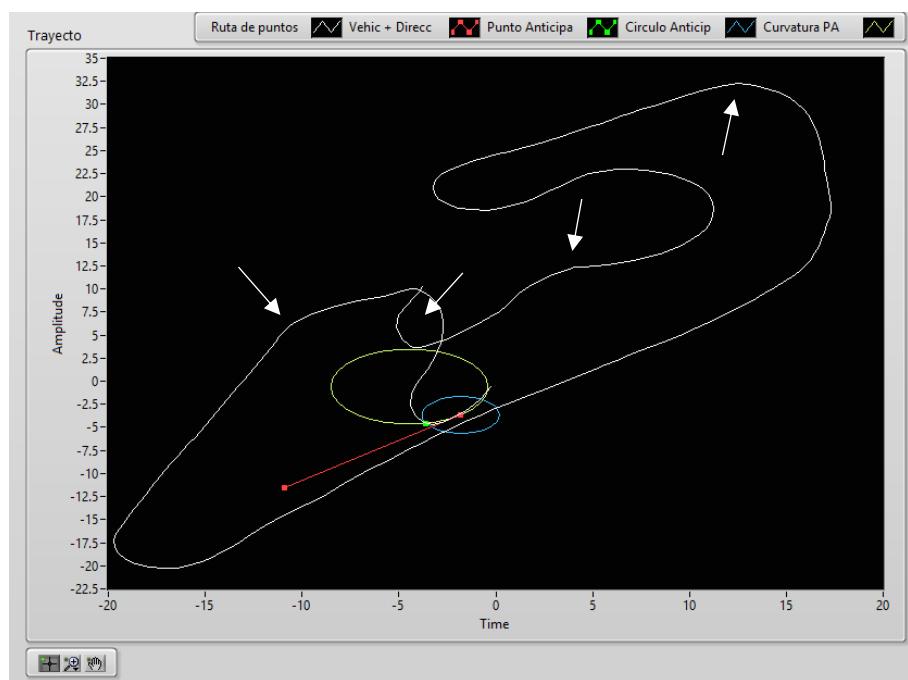
Figura 40 Ruta suavizada. $A = 0.9$; $B = 0.1$



Fuente: Elaboración propia.

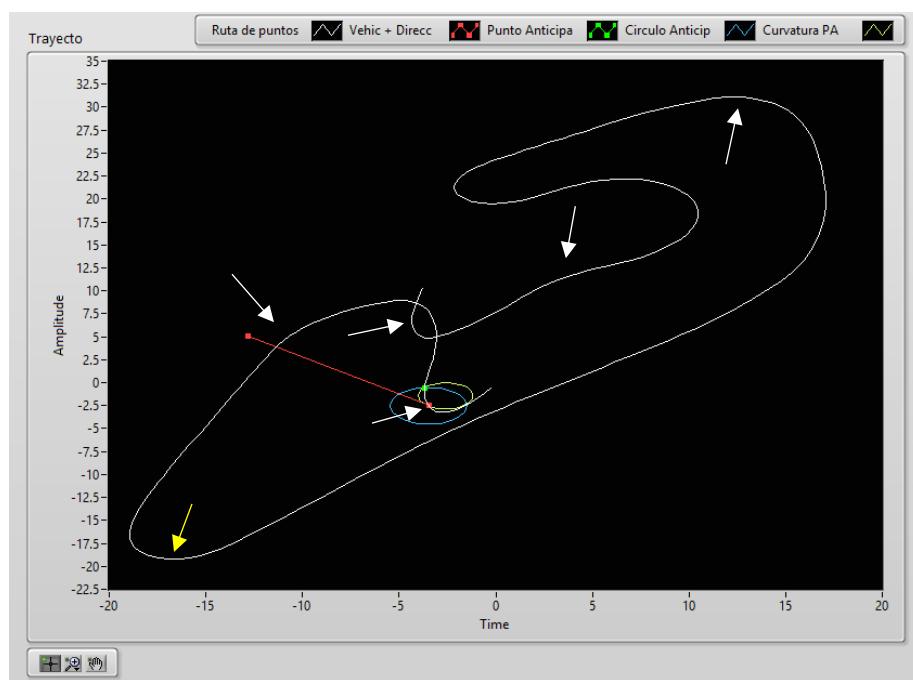
En la Figura 41, se muestra la ruta que fue utilizada para la implementación del algoritmo. En él, la curva en el final de la ruta, se encuentra más suave que en las figuras anteriores. Sin embargo, en la Figura 42, donde el valor de B es mucho mayor que A , se observa que la ruta suavizada se aleja de la ruta inicial. Tomando como ejemplo el punto donde se encuentra la flecha amarilla, en la ruta no suavizada, dicho punto tiene las coordenadas $(-17; -20.4)$ y en la suavizada, sus coordenadas son $(-17; -19.2)$. Por lo tanto, la diferencia entre ambas coordenadas es cercano a un metro. La resolución del gráfico es aproximadamente un metro.

Figura 41 Ruta suavizada. A = 0.7; B = 0.3



Fuente: Elaboración propia.

Figura 42 Ruta suavizada. A = 0.1; B = 0.9



Fuente: Elaboración propia.

7.2. Primer mapa: Pista Tacurú Pucú

La pista se mostró en la Figura 19. 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 6 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 6

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

Fuente: Elaboración propia.

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 medida 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.

7.3. Segundo mapa: Pista Robocar Race 2018

Para este segundo mapa, Figura 25, 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 7, fueron los que menos errores generaron, los mismos, oscilaban entre 1.5 y 2 metros, debido a las interferencias en el magnetómetro.

Tabla 7

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

Fuente: Elaboración propia.

7.4. 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 8, 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 8

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

Fuente: Elaboración propia.

7.5. Conclusión del capítulo

En general, el algoritmo persecución pura se desempeñó de manera correcta en las simulaciones en LabVIEW y en la implementación. La implementación de dicho algoritmo se pudo llevar a cabo sin inconvenientes.

Un detalle a resaltar respecto a los parámetros A y B trata de que, si son fijados valores que no se encuentran en el rango de valores de la Tabla 4, el algoritmo no converge, por lo tanto, no se obtiene una salida de la sección de suavizado de ruta, a causa de ello, no puede ser ejecutado la navegación.

Las simulaciones que se llevaron a cabo en LabVIEW, se debieron a que las pruebas no pudieron ser implementadas en el vehículo, como la reacción del vehículo con velocidades, aceleraciones y desaceleraciones altas. Además de simulaciones para la comprobación del funcionamiento del mismo con ciertos parámetros como Espacio, A, B y Tolerancia.

Para la implementación, se debe tener en cuenta que, si es escogido un valor pequeño para la distancia anticipada, el vehículo oscilará durante el recorrido. Sin embargo, si el valor es muy grande, el vehículo se alejará de la ruta fijada, por lo tanto, saldrá de los márgenes de la pista.

Consideraciones Finales

Análisis de Costos

En esta sección, se presenta el presupuesto del hardware “Aguara’i”, Tabla 9, y las horas hombres respecto al trabajo empleado en el proyecto y en la navegación autónoma, Tabla 10 y Tabla 11.

Tabla 9

Presupuesto del hardware

Partes	Especificaciones	Cantidad	Precio (Gs)
Auto eléctrico *	Hobao Hyper VSE 1/8 buggy	1	3.850.000
Controlador de vuelo **	Crius AIOP v2.1	1	292.000
Cargador *	iMAX B6	1	418.000
Batería LiPo *	SMC 4S 5400mAh	1	440.000
Batería LiPo ***	VEX 2S 3000mAh	1	220.000
Encoder óptico *	FC-03	1	40.000
Controlador ****	myRIO 1900	1	5.000.000
Otros *****	Impresión 3D, tornillos, cables, etc.	-	1.000.000
Total			11.260.000

Fuente: Team A2G, (2018).

Es importante aclarar que el proyecto “Aguara’i” fue financiado de la siguiente forma:

- * Universidad Católica “Nuestra Señora de la Asunción” Campus Alto Paraná.
- ** Ing. Walter Benítez e Ing. Yessica Bogado.
- *** Lic. Ariel Guerrero.
- **** National Instruments.
- ***** Universidad Católica “Nuestra Señora de la Asunción” Campus Alto Paraná junto con alumnas de la carrera de arquitectura.

Tabla 10*Horas de trabajo para el proyecto “Aguara’i”*

Actividad	Detalles	Hs
Planificación del proyecto	Investigación, especificaciones de hardware y software, planificación del desarrollo del proyecto.	50
Identificación de sistema	Modelado del auto eléctrico a escala, para el diseño de los algoritmos de control.	40
Sistema de control	Diseño de los algoritmos de control (PID) para la velocidad y el control de dirección del vehículo autónomo a escala.	50
Fusión de datos	Fusión de sensores para estimar la posición y orientación del vehículo autónomo a escala.	50
Desarrollo de hardware	Modificaciones necesarias para dotar el vehículo eléctrico de los dispositivos necesarios para una conducción autónoma.	40
Algoritmo de navegación	Diseño e implementación del algoritmo de navegación (Pure pursuit o persecución pura).	60
Comprobación del sistema	Se llevan a cabo las pruebas de la fusión de los algoritmos desarrollados de manera independiente en las etapas previas.	50
Cierre	Elaboración de la documentación correspondiente (TDP), organización del viaje y otras actividades relacionadas a la participación en la competencia.	20
Total de horas		360 Hs

Fuente: Equipo A2G.

Tabla 11*Horas de trabajo para la navegación autónoma*

Semana	Actividades	Hs/semana
1	Investigación	15
2	Investigación	15
3	Diseño del algoritmo	15
4	Simulación y pruebas	15
Total de horas		60 Hs

Fuente: Equipo A2G.

Conclusión

El presente trabajo consistió en el estudio e implementación de 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 The MathWorks, Inc, [s/f (c)] y, para la implementación, el código fuente tuvo como base los códigos creados por: Team 1712, (2018) y Harriral, (2014), integrante del Team 2168.

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).

Véanse Apéndice C (Resultados del concurso de TDP) y Apéndice D (TDP).

- 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. Véase Apéndice E.

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.

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.

Bibliografía

- Abbas, M. (2011). *Non-linear model predictive control for autonomous vehicles (Tesis Doctoral)*. University of Ontario Institute of Technology. Oshawa, Canada.
- Abiy, T., Pang, H., Tiliksew, B., Moore, K., & Khim, J. (s.f). *A* Search*. Recuperado el 16 de enero de 2019, de Brilliant: <https://brilliant.org/wiki/a-star-search/>
- Abuelsamid, S., Alexander, D., & Jerram, L. (2017). *Navigant Research Leaderboard Report: Automated Driving*. Navigant Consulting, Inc.
- Acevedo, H., & Castañeda, C. (2007). Estudio comparativo de tres técnicas de navegación para robots móviles. *Revista UIS Ingenierias*, 6(1), 77-88.
- Alsedà, L. (s/f). *A* Algorithm pseudocode*. Recuperado el 12 de marzo de 2019, de Universitat Autònoma de Barcelona: <http://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf>
- Amezcua Paredes, R., & Pineda Salgado, A. (2013). *Sistema de referencia inercial: análisis de funcionamiento, fundamentos y evolución (Tesis Doctoral)*. México: Escuela Superior de Ingeniería Mecánica y Eléctrica UP Ticomán.
- Amidi, O., & Thorpe, C. (1991). Integrated mobile robot control. *Mobile Robots V*. 1388, págs. 504-524. International Society for Optics and Photonics.
- Atiya, S., & Hager, G. (1993). Real-time vision-based robot localization. *IEEE Transactions on Robotics and Automation*, 9(6), 785-800.
- Autónomo, inteligente y conectado: el vehículo del mañana, una realidad en 2018. (23 de January de 2018). Recuperado el 13 de diciembre de 2018, de Híbridos y eléctricos: <https://www.hibridosyelectricos.com/articulo/tecnologia/autonomo-inteligente-conectado-vehiculo-manana-realidad-2018/20180122143227016969.html>
- Autonomous Car. (s/f). Recuperado el 20 de noviembre de 2018, de Techopedia: <https://www.techopedia.com/definition/30056/autonomous-car>
- Azkune, G. (12 de Noviembre de 2011). *Navegación autónoma*. Obtenido de Cuentos cuánticos. Recuperado el 20 de noviembre de 2018 de: <https://cuentoscuanticos.com/2011/11/12/navegacion-autonoma/>
- Baquero Suárez, M., & Cudris Cantillo, J. (2007). *Generación de trayectorias y procesamiento digital de imágenes para la navegación de un robot móvil (Tesis de Grado)*. Universidad Santo Tomás. Bucaramanga, Colombia.
- Barshan, B., & Durrant-Whyte, H. (1995). Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3), 328-342.
- Bekey, G., & Goldberg, K. (1993). *Neural Networks in robotics*. New York: Springer Science+Business Media.

- Benítez, W., & Bogado, Y. (2015). *Desarrollo de un prototipo de VANT (Vehículo Aéreo No Tripulado) para inspección visual de líneas eléctricas aéreas. (Tesis de Grado)*. Hernandarias: Universidad Católica "Nuestra Señora de la Asunción".
- Blake, A., & Isard, M. (1997). The condensation algorithm-conditional density propagation and applications to visual tracking. *Advances in Neural Information Processing Systems*, (págs. 361-367).
- Borenstein, J., & Feng, L. (1995). Correction of systematic odometry errors in mobile robots. *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on.* 3, págs. 569-574. IEEE.
- Campbell, S. (2007). *Steering control of an autonomous ground vehicle with application to the DARPA urban challenge (Tesis Doctoral)*. Massachusetts Institute of Technology. Cambridge, United States of America.
- Coulter, R. (1992). *Implementation of the pure pursuit path tracking algorithm (No. CMU-RI-TR-92-01)*. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.
- Cox, I., & Leonard, J. (1994). Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66(2), 311-344.
- Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte carlo localization for mobile robots. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on.* 2, págs. 1322-1328. IEEE.
- DeSouza, G., & Kak, A. (2002). Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2), 237-267.
- Dickmanns, E. D. (2007). *Dynamic vision for perception and control of motion*. London, U.K.: Springer-Verlag.
- Dickmanns, E., & Zapp, A. (1987). Autonomous High Speed Road Vehicle Guidance by Computer Vision1. *IFAC Proceedings Volumes*, 20(5), 221-226.
- D'Souza, C. (27 de Jun de 2017). *Student Competition: Mobile Robotics Training, Part 1: Controlling Robot Motion*. Recuperado el 15 de enero de 2019, de MathWorks: <https://www.mathworks.com/videos/student-competition-mobile-robotics-training-part-1-controlling-robot-motion-1498677744306.html>
- Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., & Jurišica, L. (2014). Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96, 59-69.
- e.GO - Mover. (2019). Recuperado el 24 de marzo de 2019, de Geneva International Motor Show: <https://www.gims.swiss/premieres/swiss-premieres/mover>
- Ferrer Minguez, G. (2009). *Integración Kalman de sensores inerciales INS con GPS en un UAV (Tesis de Grado)*. Universidad Politécnica de Cataluña.

- Ganapathy, V., Yun, S., & Chien, T. (2011). Enhanced D* Lite algorithm for autonomous mobile robot. *International Journal of Applied*, 1(1), 58-73.
- Gaussier, P., Joulain, C., Zrehen, S., Banquet, J., & Revel, A. (1997). Visual navigation in an open environment without map. *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*. 2, págs. 545-550. IEEE.
- Grewal, M., & Andrews, A. (2008). *Kalman Filtering: Theory and Practice Using MATLAB* (3rd ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.
- Grewal, M., Weill, L., & Andrews, A. (2001). *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons.
- Guerrero, A., Jara, M., Bogado, A., Pacheco, E., & Franco, J. (2019). *Development of an autonomous vehicle at a 1:8 scale*. Recuperado el 30 de marzo de 2019, de National Instruments: <https://forums.ni.com/t5/LabVIEW-Student-Design/nbsn-Development-of-an-autonomous-vehicle-at-a-nbsn-1-8-scale/ta-p/3898677>
- Harrilal, K. (2014). *Smooth Path Planner*. Recuperado el 10 de marzo de 2019, de GitHub: <https://github.com/KHEngineering/SmoothPathPlanner>
- Henderson, M. (1977). *Euler angles, quaternions, and transformation matrices for space shuttle analysis*. United States: NASA.
- Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2010). *Metodología de la investigación* (5ta ed.). México: McGraw-Hill/Interamericana Editores.
- HoBao Enterprising Co., Ltd. (2010). *Hyper VSe*. Recuperado el 22 de marzo de 2019, de Hobao Racing: http://hobao-racing.com/en/products_OffRoad_buffya1.php?id=162
- Janglová, D. (2004). Neural Networks in Mobile Robot Motion. *International Journal of Advanced Robotic Systems*, 1(1), 15-22.
- Joulain, C., Gaussier, P., Revel, A., & Gas, B. (1997). Learning to build visual categories from perception-action associations. *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*. 2, págs. 857-864. IEEE.
- Kanade, T., & Thorpe, C. (1985). *CMU strategic computing vision project report: 1984 to 1985*. Carnegie-Mellon University.
- Kim, D., & Nevatia, R. (1998). Recognition and localization of generic objects for indoor navigation using functionality. *Image and Vision Computing*, 16(11), 729-743.
- Kim, D., & Nevatia, R. (1999). Symbolic navigation with a generic map. *Autonomous Robots*, 6(1), 69-88.
- Komoriya, K., & Oyama, E. (1994). Position estimation of a mobile robot using optical fiber gyroscope (OFG). *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*. 1, págs. 143-149. IEEE.

- Kosaka, A., & Kak, A. (1992). Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *CVGIP: Image understanding*, 56(3), 271-329.
- Krotkov, E. (1989). Mobile robot localization using a single image. *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on* (págs. 978-983). IEEE.
- LaValle, S. (2006). *Planning algorithms*. Cambridge, United Kingdom: Cambridge university press.
- Leonard , J., & Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3), 376-382.
- Li, Y., Díaz, M., Morantes, S., & Dorati, Y. (2018). Vehículos autónomos: Innovación en la logística urbana. *Revista de Iniciación Científica*, 4(1), 34-39.
- Mejía Peñafiel, F. (21 de Septiembre de 2010). *Algoritmos genéticos*. Obtenido de Mastereduca. Recuperado el 23 de noviembre de 2018 de: <http://nando1-utb.blogspot.com/p/algoritmos-geneticos.html>
- Moraga, E. (05 de Octubre de 2017). *Sudamérica concentrará el 5% de la producción automotriz mundial en 2020*. Recuperado el 22 de noviembre de 2018, de Pulso: <http://www.pulso.cl/empresas-mercados/rxm/sudamerica-concentrara-5-la-produccion-automotriz-mundial-2020/>
- Morales, J., Martínez, J., Martínez, M., & Mandow, A. (2009). Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2D laser scanner. *EURASIP Journal on Advances in Signal Processing*, 2009, 1-10.
- Murphy, K. (1994). Analysis of robotic vehicle steering and controller delay. *Fifth International Symposium on Robotics and Manufacturing (ISRAM)*, (págs. 631-636).
- National Instruments. (2012). *LabVIEW Core 1. Manual de curso*. Estados Unidos: National Instruments.
- National Instruments. (2016). *Inverse Tangent (2 Input) Function*. Recuperado el 14 de marzo de 2019, de National Instruments: http://zone.ni.com/reference/en-XX/help/371361N-01/glang/inverse_tangent_2_input/
- National Instruments. (s/f (a)). *¿Qué es myRIO?* Recuperado el 09 de marzo de 2019, de National Instruments: <http://www.ni.com/es-cr/shop/engineering-education/portable-student-devices/myrio-student-embedded-device/what-is-myrio.html>
- National Instruments. (s/f (b)). *¿Qué es LabVIEW?* Recuperado el 09 de marzo de 2019, de National Instruments: <http://www.ni.com/es-cr/shop/labview.html>
- Noureldin, A., Karamat, T., & Georgy, J. (2012). *Fundamentals of inertial navigation, satellite-based positioning and their integration*. Springer Science & Business Media.

- Ollero, A., & Heredia, G. (1995). Stability analysis of mobile robot path tracking. *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on.* 3, págs. 461-466. IEEE.
- Ollero, A., García-Cerezo, A., & Martínez, J. (1994). Fuzzy supervisory path tracking of mobile robots. *Control Engineering Practice*, 2(2), 313-319.
- Otsason, V., Varshavsky, A., LaMarca, A., & De Lara, E. (2005). Accurate GSM indoor localization. *International conference on ubiquitous computing* (págs. 141-158). Springer, Berlin, Heidelberg.
- Park, K., Chung, H., & Lee, J. (1998). Dead reckoning navigation for autonomous mobile robots. *IFAC Proceedings Volumes*, 31(3), 219-224.
- Patel, A. (s/f). *Introduction to A**. Recuperado el 11 de enero de 2019, de Red Blob Games: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>
- Petrinec, K., Kovacic, Z., & Marozin, A. (2003). Simulator of multi-AGV robotic industrial environments. *Industrial Technology, 2003 IEEE International Conference on.* 2, págs. 979-983. IEEE.
- Premerlani, W., & Bizard, P. (2009). Direction cosine matrix imu: Theory. *Diy Drone: Usa*, 13-15.
- Principios básicos de lidar.* (Julio de 2018). Recuperado el 20 de noviembre de 2018, de ArcGIS: <http://desktop.arcgis.com/es/arcmap/10.3/manage-data/las-dataset/what-is-lidar-data-.htm#>
- Raffo, G., Normey-Rico, J., Rubio, F., & Kelber, C. (2009). Control predictivo en cascada de un vehículo autónomo. *Revista Iberoamericana de Automática e Informática Industrial*, 6(1), 63-74.
- Reddy, H. (2013). PATH FINDING-Dijkstra's and A* Algorithm's. *International Journal in IT and Engineering*, 1-15.
- RINSPEED - microSNAP. (2019). Recuperado el 24 de marzo de 2019, de Geneva International Motor Show: <https://www.gims.swiss/premieres/european-premieres/microsnap>
- Robocar Race. (2018). Obtenido de Robocar Race. Recuperado el 18 de noviembre de 2018 de: <http://www.roborace.com.br/>
- Rodríguez-Castaño, A., Heredia, G., & Ollero, A. (2000). Analysis of a GPS-based fuzzy supervised path tracking system for large unmanned vehicles. *IFAC Proceedings Volumes*, 33(25), 125-130.
- Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *nature*, 323(6088), 533.
- SAE Internacional. (2014). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. United States of America: SAE Internacional.

- Samuel, M., Hussein, M., & Mohamad, M. (2016). A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle. *International Journal of Computer Applications*, 135(1), 35-38.
- Santos-Victor, J., Sandini, G., Curotto, F., & Garibaldi, S. (1993). Divergent stereo for robot navigation: Learning from bees. *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on* (págs. 434-439). IEEE.
- Scharf, L., Harthill, W., & Moose, P. (1969). A comparison of expected flight times for intercept and pure pursuit missiles. *IEEE Transactions on Aerospace and Electronic Systems*(4), 672-673.
- Siouris, G. (1993). *Aerospace Avionics Systems - A Modern Synthesis*. San Diego: Academic Press, Inc.
- Stout, B. (1996). Smart moves: Intelligent pathfinding. *Game developer magazine*, 10, 28-35.
- Sugihara, K. (1988). Some location problems for robot navigation using a single camera. *Computer vision, graphics, and image processing*, 42(1), 112-129.
- Team 1712. (2018). *Implementation of the Adaptive Pure Pursuit Controller*. Recuperado el 09 de marzo de 2019, de chiefdelphi: <https://www.chiefdelphi.com/t/paper-implementation-of-the-adaptive-pure-pursuit-controller/166552>
- The MathWorks, Inc. (2016). *ExampleHelperRobotSimulator*. Recuperado el 12 de marzo de 2019, de MATLAB File Help.
- The MathWorks, Inc. (s/f (a)). *Pure Pursuit Controller*. Recuperado el 7 de febrero de 2019, de MathWorks: <https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html>
- The MathWorks, Inc. (s/f (b)). *Robotics System Toolbox*. Recuperado el 26 de febrero de 2019, de MathWorks: <https://www.mathworks.com/help/robotics/>
- The MathWorks, Inc. (s/f (c)). *Path Following for a Differential Drive Robot*. Recuperado el 26 de febrero de 2019, de MathWorks: <https://www.mathworks.com/help/robotics/examples/path-following-for-differential-drive-robot.html>
- Thorpe, C., Hebert, M. H., Kanade, T., & Shafer, S. A. (1991). Toward autonomous driving: The CMU Navlab. part i-perception. *IEEE Intelligent Systems*(4), 31-42.
- Thorpe, C., Hebert, M., Kanade, T., & Shafer, S. (1988). Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3), 362-373.
- Thrun, S. (2000). Probabilistic Algorithms in Robotics. *Ai Magazine*, 21(4), 93.

- Tian, Y., Pei, K., Jana, S., & Ray, B. (2018). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *Proceedings of the 40th International Conference on Software Engineering* (págs. 303-314). ACM.
- Titterton, D., & Weston, J. (2004). *Strapdown inertial navigation technology* (2nd ed.). IET.
- Valdés Tapia, H. (2013). *Navegación Local de Robots Móviles en ambientes desconocidos utilizando Programación Orientada a Comportamientos (Tesis de Grado)*. Universidad Católica del Norte, Chile.
- Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W., & Kanade, T. (1985). First Results in Robot Road-Following. *IJCAI*, (págs. 1089-1095).
- Wang, W., Hsu, T., & Wu, T. (2017). The improved pure pursuit algorithm for autonomous driving advanced system. *Computational Intelligence and Applications (IWCIA), 2017 IEEE 10th International Workshop on* (págs. 33-38). IEEE.
- Wit, J., Crane III, C., & Armstrong, D. (2004). Autonomous ground vehicle path tracking. *Journal of Robotic Systems*, 21(8), 439-449.
- Zhang, W., Dechter, R., & Korf, R. (2001). Heuristic search in artificial intelligence. *Artificial Intelligence*, 129(1-2), 1-4.
- Zhao, P., Chen, J., Song, Y., Tao, X., Xu, T., & Mei, T. (2012). Design of a control system for an autonomous vehicle based on adaptive-PID. *International Journal of Advanced Robotic Systems*, 9(2).

Glosario

- **Inercia:** es la propiedad de los cuerpos a mantener un estado de reposo permanente o una velocidad de rotación y de translación constante a no ser que se vean afectados por fuerzas o momentos respectivamente (primera ley de Newton).
- **LabVIEW:** es un entorno de programación gráfico que se puede utilizar para crear aplicaciones rápidas y eficientes con interfaces de usuario profesionales.
- **Navegación Absoluta:** la navegación es realizada con sensores que obtienen la localización mediante balizas, puntos de referencia o señales satelitales (GPS).
- **Navegación Inercial:** estima la posición, velocidad y actitud de un vehículo utilizando giróscopos y acelerómetros puestos en el mismo.
- **Navegación Relativa:** los datos de posición y orientación se obtienen utilizando la información proporcionada por varios sensores a bordo (giroscopio, acelerómetro, encoder).
- **Sensores Inerciales IMU (Inertial Measuring Unit):** miden la variación de rotación (giróscopos) y la aceleración (acelerómetros). Estos sensores permiten conocer la posición relativa del auto.
- **SubVI:** son VIs que se crean para usar dentro de otro VI.
- **Virtual Instruments (VI):** programas de LabVIEW.

Apéndice

Apéndice A

Relación entre los sistemas de coordenadas

En el Capítulo 1 del Marco Teórico se describieron los sistemas de coordenadas, sin embargo, es necesario algún tipo de método para pasar de un sistema a otro. Como anexo, se tratará en este apéndice algunos de esos métodos, tomando como base el libro de Titterton & Weston, (2004), y el proyecto final de Ferrer Minguez, (2009).

A.1. Matriz de cosenos directores

La matriz de cosenos directores es un sistema de cambio de coordenadas. Su objetivo es rotar un vector de un marco de referencia a otro, en este caso, representan vectores unitarios de los ejes de Body proyectados a ejes de referencia.

$$\mathbf{C}_b^n = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \quad \text{A.1.}$$

La matriz A.1. está compuesta por los cosenos de los ángulos entre los ejes de los dos sistemas de coordenadas, es decir, que cada componente de la matriz es un coseno. Acaba siendo una matriz de cambio de coordenadas.

Una cantidad definida de vectores en los ejes de Body, \mathbf{r}^b , pueden ser expresados en los ejes de referencia multiplicando previamente el vector por la dirección de la matriz del coseno de la siguiente manera:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^n = \mathbf{C}_b^n \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}^b \quad A.2.$$

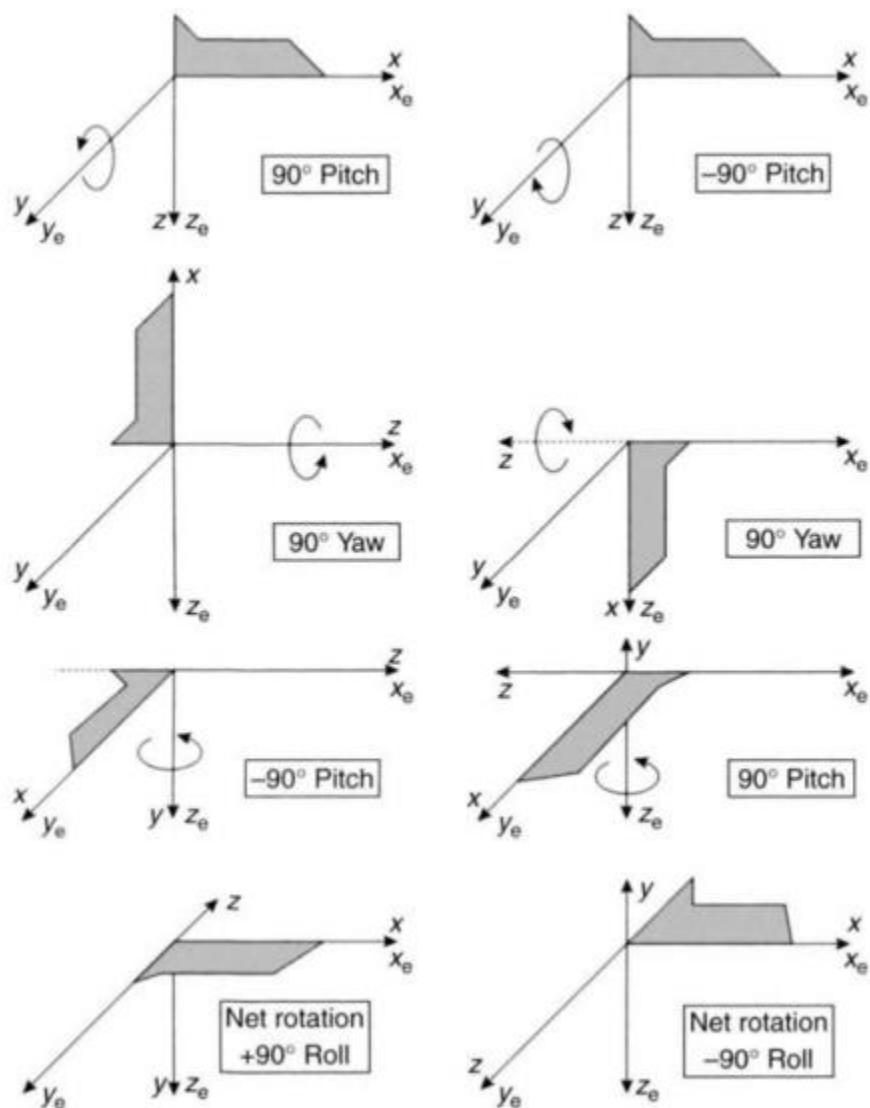
$$\mathbf{r}^n = \mathbf{C}_b^n \mathbf{r}^b \quad A.3.$$

A.2. Ángulos de Euler

En este método son utilizados los tres ángulos de Euler. Dichos ángulos definen el cambio de un sistema de coordenadas mediante una sucesión ordenada de giros. Los ángulos de Euler (phi, theta, psi) corresponden con los ángulos convencionales de roll (ϕ), pitch (θ), yaw (ψ) que se utilizan en navegación para especificar la actitud de un móvil (Figura 43).

Un dato importante es que las rotaciones siempre se deben hacer en el mismo orden, esto es porque, si son aplicados con órdenes diferentes, las transformaciones pueden ser diferentes, debido a que los ángulos de Euler no están únicamente definidos y existen ambigüedades.

Figura 43 Ilustración del efecto del orden de las rotaciones del cuerpo



Fuente: Titterton & Weston, (2004).

Una transformación de un marco de coordenadas a otro se puede llevar a cabo como tres rotaciones sucesivas sobre diferentes ejes (Titterton & Weston, 2004). Dichas rotaciones pueden ser expresados de forma matemática como tres matrices de coseno de dirección separadas:

Rotación ψ sobre el eje z:

$$\mathbf{C}_1 = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad A.4.$$

Rotación θ sobre el eje y:

$$\mathbf{C}_2 = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \quad A.5.$$

Rotación ϕ sobre el eje x:

$$\mathbf{C}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \quad A.6.$$

Como mencionan Titterton & Weston, (2004), una transformación de la referencia a los ejes de Body se puede expresar como el producto de estas tres transformaciones separadas de la siguiente manera:

$$\mathbf{C}_b^n = \mathbf{C}_1 \mathbf{C}_2 \mathbf{C}_3 \quad A.7.$$

Del mismo modo, la transformación inversa de Body a los ejes de referencia viene dada por:

$$\mathbf{C}_b^n = \mathbf{C}_n^{bT} = \mathbf{C}_1^T \mathbf{C}_2^T \mathbf{C}_3^T \quad A.8.$$

$$\mathbf{C}_b^n = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad A.9.$$

Resolviendo la ecuación A.9. se tiene el siguiente resultado:

$$C_b^n = \begin{bmatrix} \cos \theta \cos \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix}$$

Ecuación A.10.

La ecuación A.10. es la matriz de cosenos directores expresados en términos de los ángulos de Euler (Titterton & Weston, 2004).

A.3. Cuaterniones

Otro método utilizado es la representación de actitud de cuaternion, este permite que una transformación de un marco de coordenadas a otro se efectúe mediante una sola rotación alrededor de un vector definido en el marco de referencia, a diferencia del método de los ángulos de Euler. El cuaternion es una representación vectorial de cuatro elementos, cuyos elementos son funciones de la orientación de este vector y la magnitud de la rotación (Titterton & Weston, 2004). Pueden ser consultados los siguientes documentos para más información respecto a este método, sean explicaciones o deducciones: (Titterton & Weston, 2004), (Benítez & Bogado, 2015), (Henderson, 1977).

Apéndice B

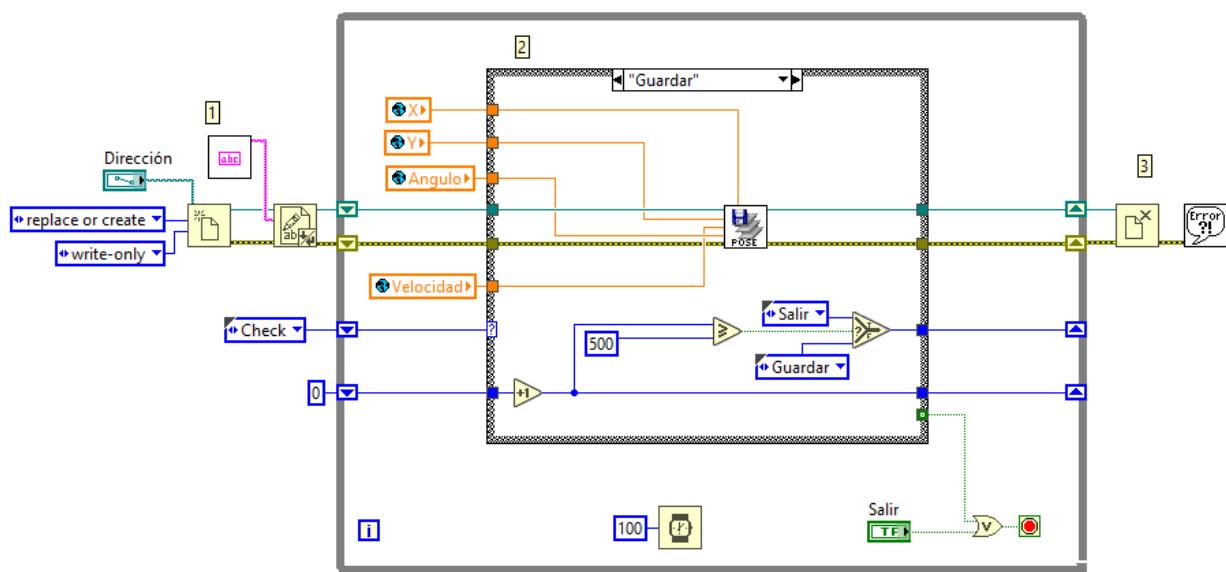
Código fuente del algoritmo de implementado

El algoritmo de implementación tuvo como base el código programado por el Team 1712. En él, se realizaron varias modificaciones, puesto que el código original está diseñado para simulaciones, y para cumplir con uno de los objetivos de este proyecto, el algoritmo tuvo que ser implementado en un auto eléctrico a escala.

B.2.1. Odometría

Como se explicó anteriormente, los valores de x e y son calculados para luego utilizarlos como ruta de puntos. Estos valores son guardados en un documento de texto a través de un Virtual Instruments (VI) diseñado por Jorge Bareiro y Micaela Jara, miembros del grupo A2G.

Figura 44 Código fuente del recabado de datos



Fuente: Bareiro, & Jara, (2018).

En la Figura 44 se puede observar el VI utilizado para recabar los datos. Inicialmente, (1) se crea el documento de texto donde se guardarán los datos, con los debidos encabezados. La programación posee 3 casos (2), Chequear, Guardar y Salir. Chequear fue diseñado con el fin de estipular una condición para el inicio del caso Guardar. La condición que se debe cumplir es respecto a la velocidad del auto, esa velocidad debe igualar o sobrepasar los 0.05 m/s. En el caso Guardar, se recaba los datos de hora del recorrido, coordenada x , coordenada y , velocidad y ángulo de dirección, en el orden mencionado. Los puntos son recabados cada 100 ms. Una vez que son guardados 500 puntos, pasa al caso Salir, donde simplemente detiene la estructura Mientras para luego cerrar el documento de texto (3). En caso de error, se despliega un mensaje con descripciones de la misma.

La cantidad de puntos a guardar y el tiempo de iteración pueden variar dependiendo de la longitud de la pista. Fue seleccionado 500 puntos debido a la memoria del myRIO, si la cantidad de puntos sobrepasa este valor, se sobrescriben los datos en el documento de texto.

Los valores son obtenidos a través de variables globales. Los valores del ángulo provienen de la fusión del magnetómetro con el giroscopio y la distancia proviene de la medición de un encoder óptico. Con estos datos son calculados las coordenadas x e y , y la velocidad del vehículo.

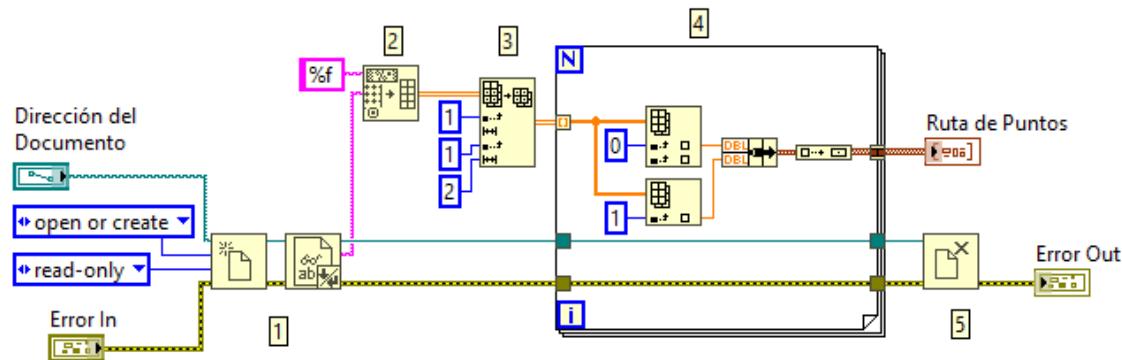
B.2.2. Generación del camino

B.2.1. Extracción de los puntos de ruta

En esta sección, los datos guardados en el documento son extraídos y convertidos en un vector de clusters con coordenadas x e y (Figura 45). El documento de texto es abierto y leído (1), puesto que los datos que contiene está en formato string, debe ser convertido en formato double (2). En

(3), como el vector posee 5 columnas de datos y la primera fila es el encabezado, son extraídos los valores de las columnas que se encuentran en la posición 1 y 2, a partir de la posición $i=1$ y $j=1$, puesto que son las columnas correspondientes a las coordenadas x e y . Este vector es descompuesto en los valores de x e y para que puedan ser formados los puntos, $(x; y)$, en clusters y estos compongan un vector de puntos (4), el waypoint o ruta de puntos. Una vez que el proceso es concluido, el programa cierra el documento de texto.

Figura 45 Código fuente de la extracción de los puntos de la ruta

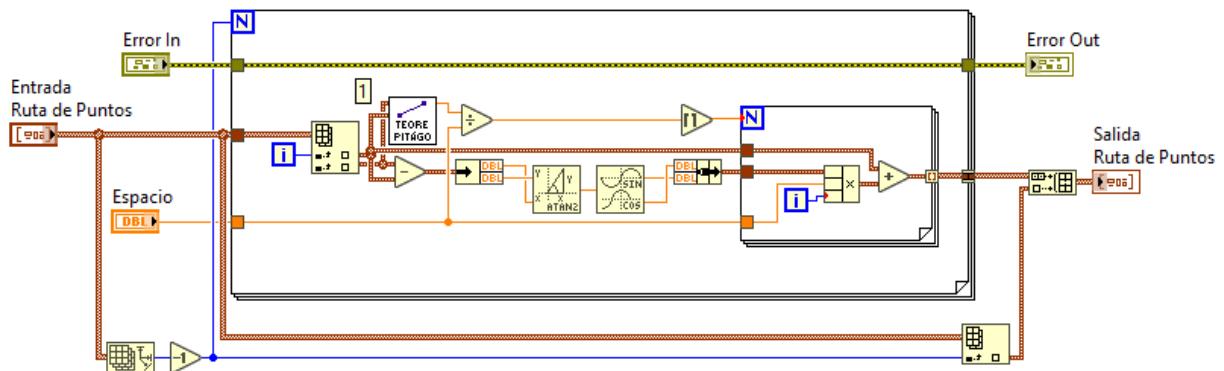


Fuente: Franco, Galeano, & Reckzeigal, (2018).

B.2.2. Inyección de puntos

El funcionamiento del código fuente (Figura 46) fue explicado previamente en Capítulo 6.

Figura 46 Código fuente de la inyección de puntos

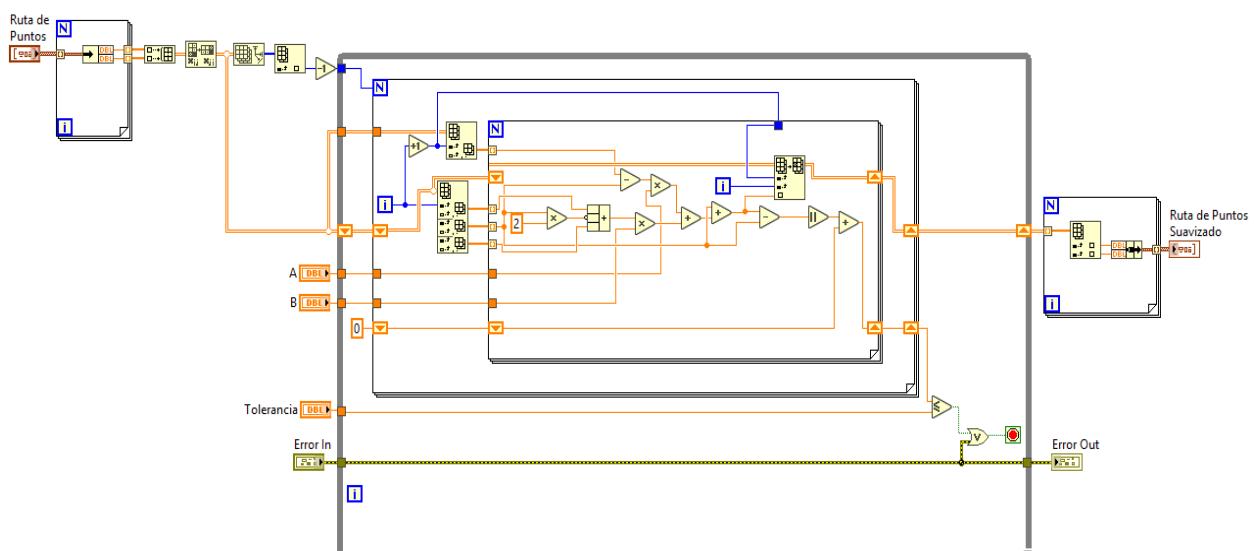


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.3. Suavizado

En la Figura 47 se presenta en código fuente de la sección de suavizado en LabVIEW, hecho por el Team 1712, (2018). La función que cumple es la misma presentada en la subsección Suavizado del Capítulo 6.

Figura 47 Código fuente en LabVIEW del suavizado



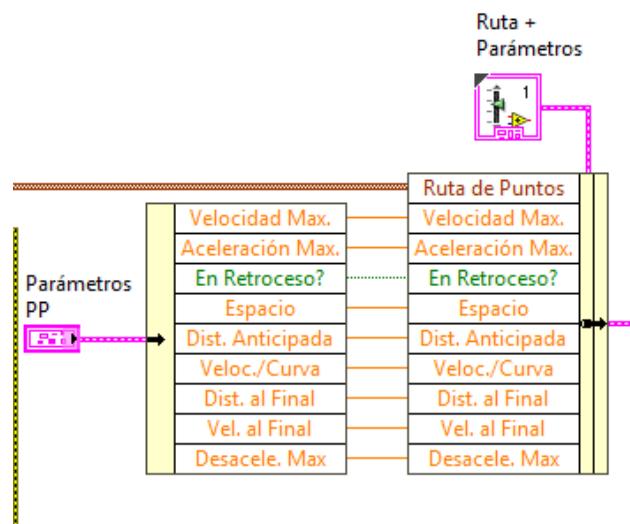
Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.4. Parámetros

A continuación, en la Figura 48, los parámetros que deben ser establecidos para el funcionamiento del algoritmo (Team 1712, 2018).

- 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.

Figura 48 Parámetros del algoritmo persecución pura

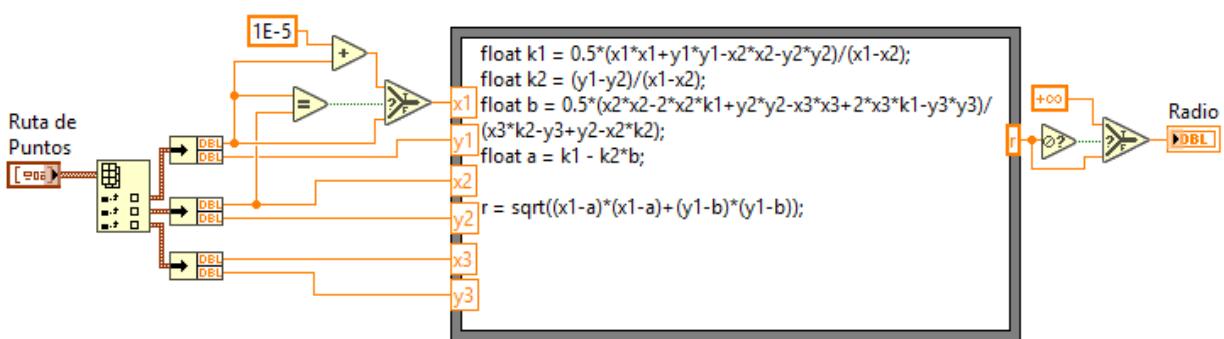


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.5. SubVI cálculo del radio de curvatura del camino

En la Figura 49 se presenta el código para el cálculo del radio de curvatura a partir de 3 puntos conocidos explicado en el Capítulo 6.

Figura 49 Código fuente del cálculo del radio de curvatura

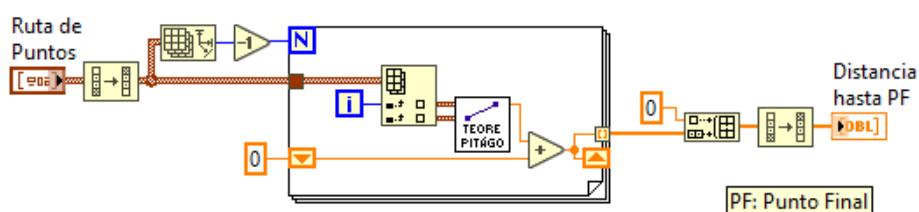


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.6. SubVI cálculo de la distancia entre los puntos

En la Figura 50, se muestra el código que describe a qué distancia se encuentra cada punto del final de la ruta, a lo largo de la ruta (no la distancia en línea recta).

Figura 50 Código fuente del cálculo de la distancia entre los puntos

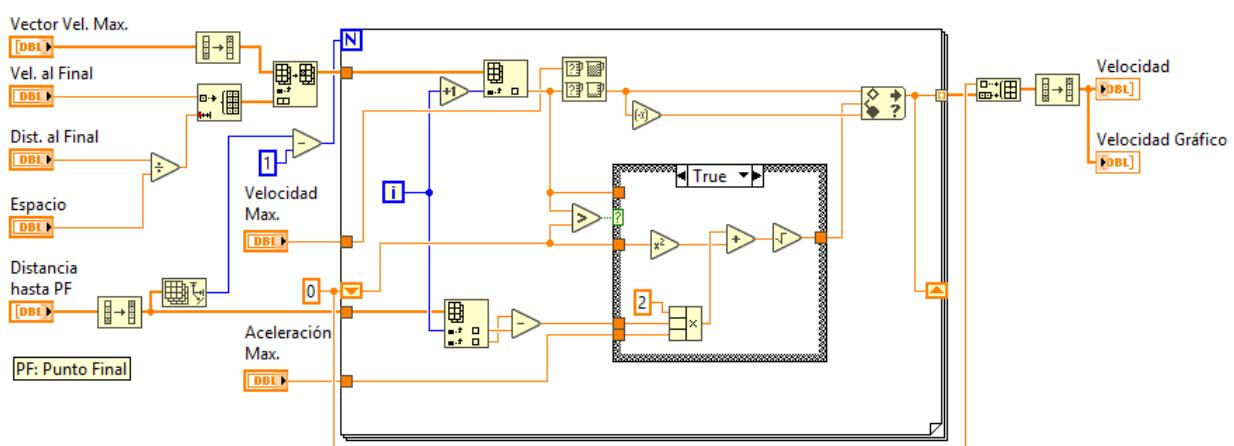


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.7. SubVI cálculo de la velocidad

En esta sección del código, se realiza el cálculo de la velocidad en un punto aplicando la fórmula de Torricelli (Figura 51).

Figura 51 Código fuente del cálculo de la velocidad en un punto

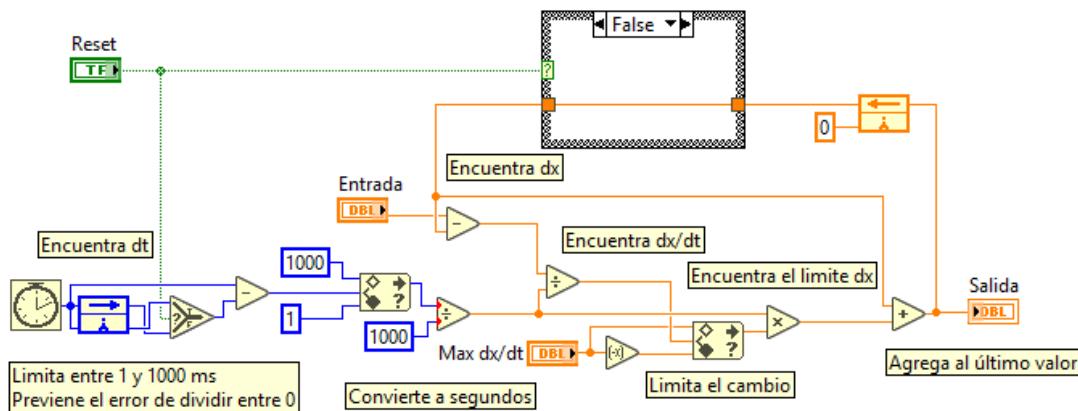


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.8. SubVI limitador de velocidad

En la Figura 52 se muestra el código utilizado para limitar la velocidad. Como se mencionó anteriormente, la salida intenta alcanzar el valor de la entrada, pero solo puede cambiar a una velocidad de Max dx/dt.

Figura 52 Código fuente del limitador de velocidad

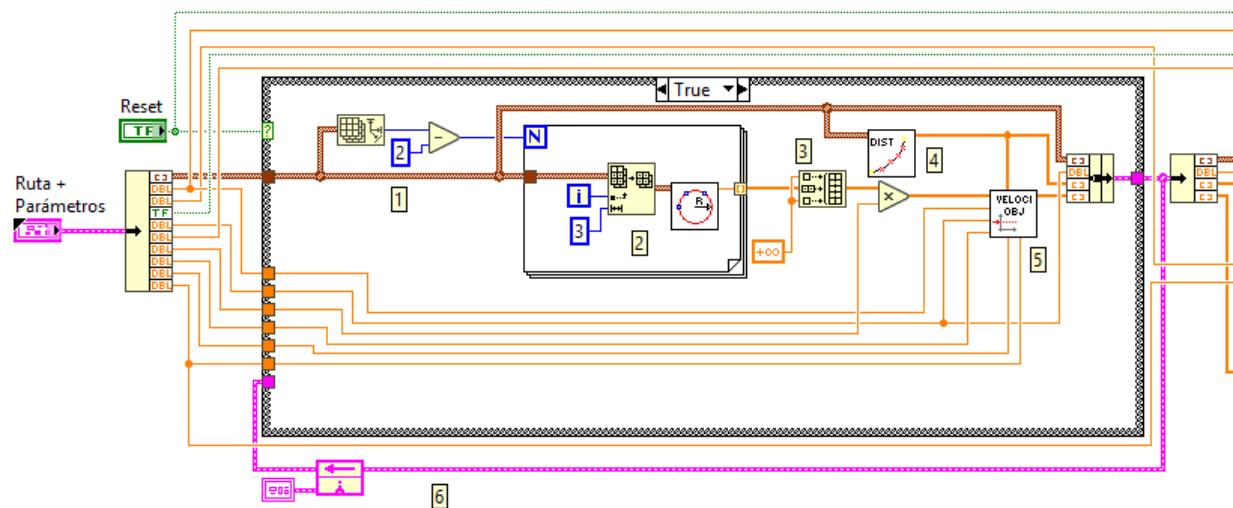


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.2.9. Parte 1: Fragmento del código del subVI Persecución Pura

En la Figura 53 se puede observar el fragmento del código que engloba los tres subVIs mencionados anteriormente.

Figura 53 Parte 1: Fragmento del código fuente del subVI Persecución Pura



Fuente: Franco, Galeano, Reckzeigl, & Jara (2018).

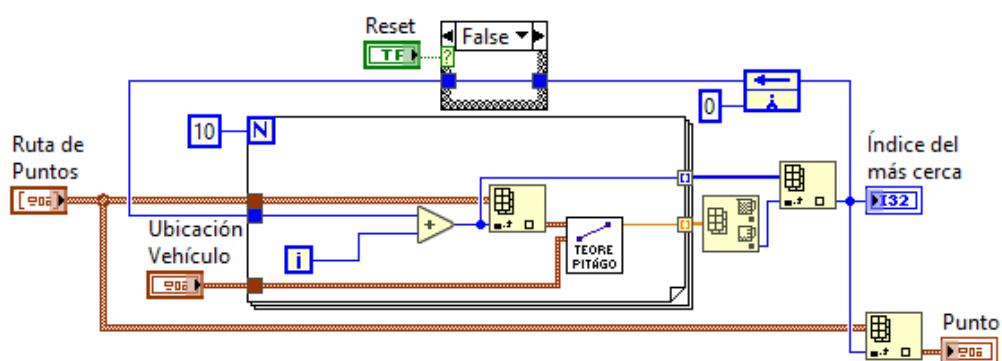
El cluster que contiene la ruta de puntos y los parámetros establecidos al inicio son descomprimidos. En el caso que se active el Reset (inicio de la navegación), ingresa la ruta de puntos a la estructura Caso, se calcula el radio de todos los puntos (2) contenidos en la ruta excepto del punto inicial y el final (1), para dos puntos, se define radio igual a infinito (3), puesto que la curvatura es igual a cero. Se calcula la distancia entre cada punto y el punto final del trayecto (4), para así, poder calcular la velocidad objetivo que debe tener el vehículo en cada uno de los puntos (5). En caso que Reset sea falso, los valores calculados cuando inicia la navegación son guardados en el nodo de realimentación (6) y estos valores son los que pasan a la segunda parte del subVI Persecución Pura.

B.3. Seguimiento del camino

B.3.1. SubVI cálculo del punto más cercano

En la Figura 54 se muestra el código del subVI para encontrar el punto más cercano. Este código procesa los puntos que están por delante del último procesado (Team 1712, 2018).

Figura 54 Código fuente del cálculo del punto más cercano

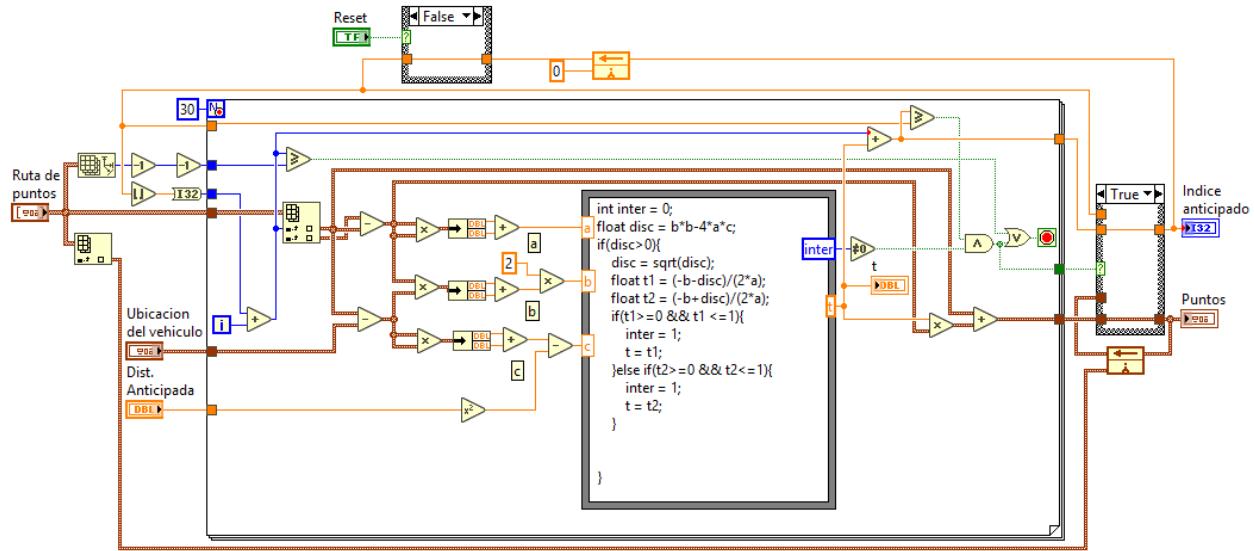


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.3.2. SubVI del cálculo del punto anticipado

En la Figura 55 se presenta el subVI perteneciente al cálculo del punto anticipado, previamente explicado en el Capítulo 6.

Figura 55 Código fuente del cálculo del punto anticipado

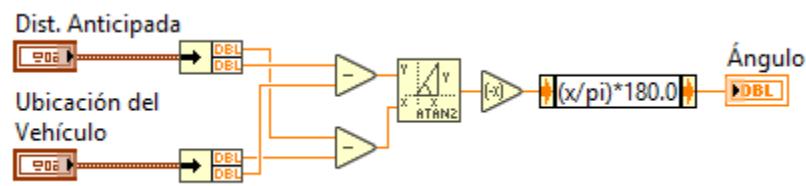


Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.3.3. SubVI del ángulo de corrección

En la Figura 56 se muestra el subVI del código del ángulo de corrección. Se ha utilizado la función atan2, porque calcula el valor del arcotangente para ángulos en cualquiera de los cuatro cuadrantes del plano x - y (National Instruments, 2016).

Figura 56 Código fuente del ángulo de corrección

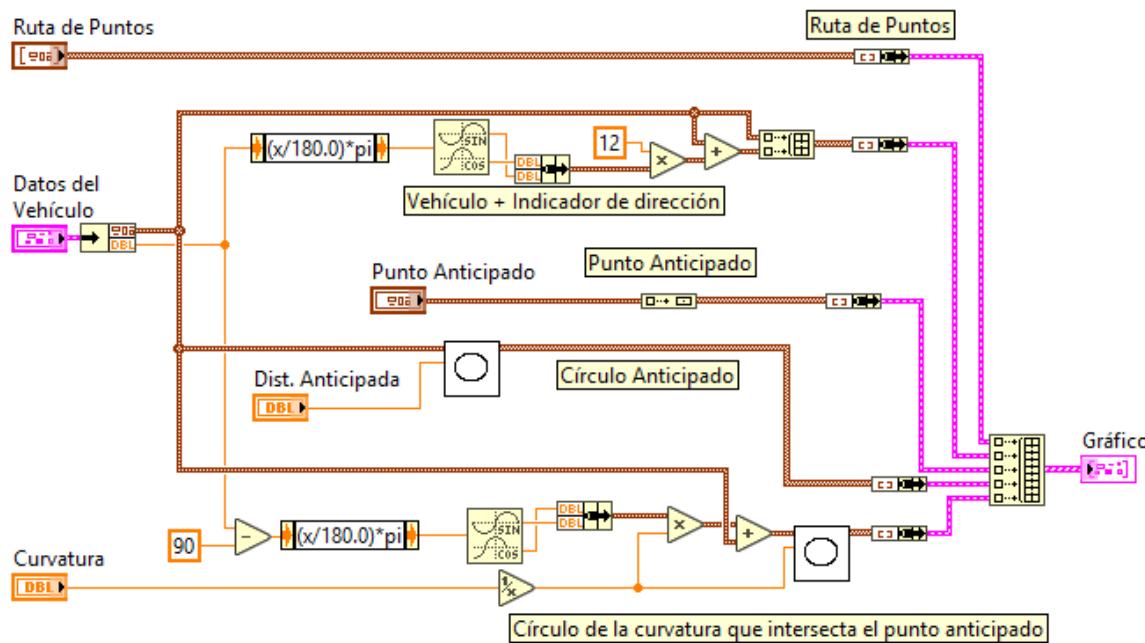


Fuente: Franco, Galeano, & Reckzeigel, (2018).

B.3.4. SubVI para la visualización del recorrido

El código que se muestra en la Figura 57, fue desarrollado por el Team 1712 con el fin de poder visualizar el recorrido del vehículo o robot. En él, se pueden observar: la ruta de puntos, el vehículo moviéndose, el círculo que tiene como radio la distancia anticipada moviéndose junto con él y los puntos anticipados, todo ello en tiempo real.

Figura 57 Código fuente de la visualización del recorrido



Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.3.5. Parte 2: Fragmento del código del subVI Persecución Pura

En la Figura 58 se presenta la segunda parte del código del subVI Persecución Pura. Las dos partes que se han mostrado en este apéndice son los utilizados para este proyecto. En la

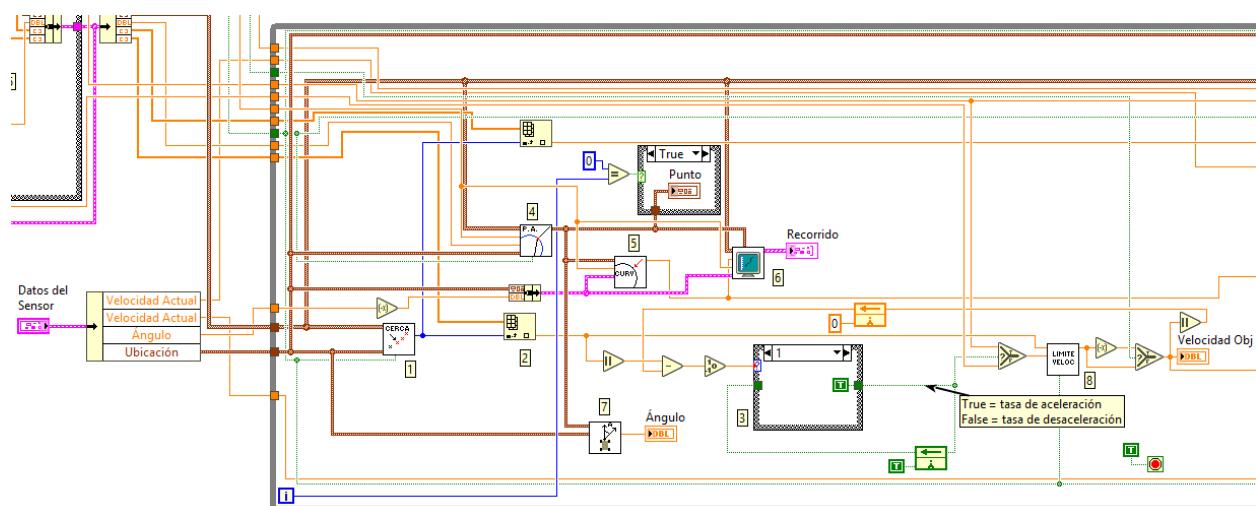
implementación no fue necesario el cálculo de velocidades diferentes para las ruedas, así como no es del alcance de este proyecto realizar el control PID de la velocidad y la dirección.

En este fragmento, se pueden observar los subVIs que: calcula el punto más cercano (1) y el punto anticipado (4); el subVI que permite la visualización del recorrido (6), el que corrige el ángulo de dirección (7) y el que limita el valor de la velocidad (8).

En el punto (2), el algoritmo busca la velocidad objetivo para ese punto cercano y en la estructura de caso (3), determina si se debe utilizar la tasa de aceleración o desaceleración en función de si la velocidad objetivo es mayor o menor que el actual (Team 1712, 2018).

En el punto (5) se tiene el subVI que calcula la curvatura al punto anticipado, dicho subVI es utilizado para calcular la velocidad que necesitan las ruedas izquierda y derecha. Al ser irrelevante para este proyecto, no fue explicado en el Capítulo 6. Solo es utilizado para la visualización del recorrido.

Figura 58 Parte 2: Fragmento del código fuente del subVI Persecución Pura



Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

B.4. Proyecto Algoritmo Persecución Pura

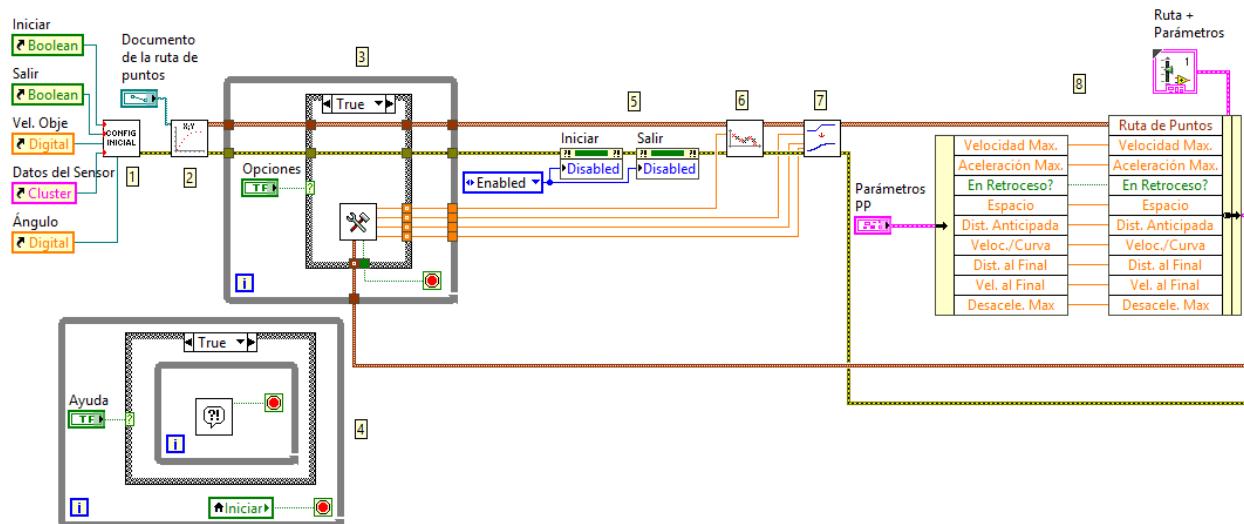
En la Figura 59 y Figura 60 se muestra el código fuente del proyecto del algoritmo de navegación. A continuación, se explica cada parte del mismo:

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.

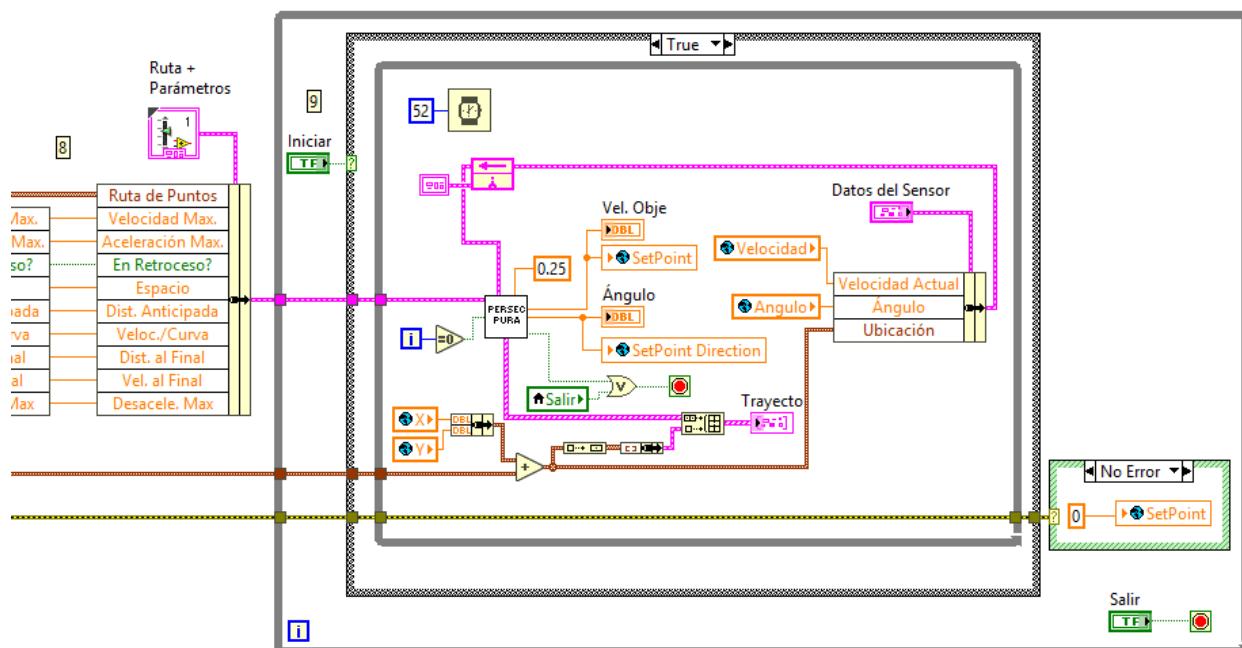
El valor 0.25 que se muestra encima del subVI es el ancho entre las ruedas del vehículo, dicha información es utilizado para el cálculo de las velocidades de la rueda izquierda y derecha, sin embargo, como esta implementación no es necesario el cálculo de las mismas, ese dato puede ser omitido.

Figura 59 Código fuente del proyecto Algoritmo Persecución Pura. Parte 1



Fuente: Franco, Galeano, Reckzeigl, & Jara (2018).

Figura 60 Código fuente del proyecto Algoritmo Persecución Pura. Parte 2



Fuente: Franco, Galeano, Reckzeigel, & Jara (2018).

Apéndice C

Resultados del concurso del Technical Description Paper

3/29/2019

Gmail - Anuncio dos vencedores de melhor TDP do RoboCar Race 2018



Micaela Jara <micajara10@gmail.com>

Anuncio dos vencedores de melhor TDP do RoboCar Race 2018

1 mensaje

RoboCar Race <robocar.race@gmail.com>
Cco: micajara10@gmail.com

6 de diciembre de 2018, 22:09

Boa noite a Todos.

Desculpem-nos pela longa demora no retorno a vocês após o evento. Estamos entrando em contato para anunciar os vencedores do melhor TDP.

1a - Equipe Semear - USP São Carlos

2a - Equipe A2G - Universidad Católica Nuestra Señora de la Asunción

3a - Equipe Car-not-found - UFABC

Os vencedores terão o TDP publicado na revista JPAUT - <https://jpaut.com.br> na 3a Edição do 1a trimestre de 2019.

Solicitamos que os vencedores entrem em contatos conosco para receber maiores informações sobre o formato, correções e extensão do paper.

Agradecemos a participação de todos e contamos com a vossa presença no próximo evento de 2019. Manteremos vocês informados sobre as novidades.

Abraços

—

Prof. Dr. Edson Kitani (FATEC Santo André)

Prof. Dr. Luiz Celiberto Jr. (UFABC)

Coordenadores

Apéndice D

Technical Description Paper

D.1. TDP presentado en el concurso

Desarrollo de un vehículo autónomo a escala 1:8

Gregorio Ariel Guerrero Moral
 Centro de Tecnologías de la
 Información y Comunicación
 Parque Tecnológico de Itaipu -
 Paraguay
 Hernandarias, Paraguay
 ariel.guerrero@pti.org.py

Ariel David Bogado Arce
 Centro de Investigación en Ciencias,
 Tecnología e Innovación Avanzada
 Universidad Católica "Nuestra Señora
 de la Asunción"
 Hernandarias, Paraguay
 ariel.bogado@uc.edu.py

Micaela Carolina Jara Ten Kathen
 Centro de Investigación en Ciencias,
 Tecnología e Innovación Avanzada
 Universidad Católica "Nuestra Señora
 de la Asunción"
 Hernandarias, Paraguay
 micaela.jara@uc.edu.py

Jesús María Franco Santacruz
 Centro de Investigación en Ciencias,
 Tecnología e Innovación Avanzada
 Universidad Católica "Nuestra Señora
 de la Asunción"
 Hernandarias, Paraguay
 jesus.francos@uc.edu.py

Erid Eulogio Pacheco Viana
 Centro de Investigación en Ciencias,
 Tecnología e Innovación Avanzada
 Universidad Católica "Nuestra Señora
 de la Asunción"
 Hernandarias, Paraguay
 erid.pacheco@uc.edu.py

Resumen— El desarrollo de un vehículo autónomo es objeto de amplio estudio por muchos investigadores [1]. Uno de los enfoques de estudio consiste en el empleo de modelos físicos a escala cuya construcción es el objeto de este artículo. Para este propósito un auto eléctrico a escala 1:8 remotamente controlado es modificado. Se utiliza un myRIO para las tareas de adquisición de datos (odómetro, acelerómetro, magnetómetro, giroscopio, gps), y las de actuación. Otra tarea que realiza es fusionar los datos de los sensores de manera tal a que la información de posición y orientación pueda ser utilizada para la toma de decisión de navegación. Estos datos pueden ser remitidos a una notebook ejecutando una aplicación en LabVIEW. Esta estación de trabajo provee una interfaz hombre máquina al operador para visualizar los parámetros de navegación.

Palabras clave—*vehículo autónomo, IMU, GPS, algoritmo de navegación, kalman*

I. INTRODUCCIÓN

Los vehículos autónomos han atraído una gran cantidad de interés en investigación en los últimos años, así como importantes esfuerzos de desarrollo de la industria. En 2007, DARPA ha ejecutado Urban Grand Challenge [2], con las entradas de varias universidades documentadas en muchas publicaciones diferentes, por ejemplo, [3, 4].

Muchas empresas del sector automotriz poseen su propia división dedicadas a la investigación de vehículos autónomos y recientemente otras empresas que no son del sector tales como NVIDIA, APPLE, GOOGLE, YANDEX, BIADU han comenzado a desarrollar su propio vehículo autónomo. Proyectos que han sido ampliamente reportado en las noticias [5, 6]. Si bien se han realizado importantes esfuerzos en este campo, aún quedan muchos problemas por resolver, entre ellos, los problemas de detección, los diferentes tipos y niveles de control y la interacción de los vehículos autónomos con su entorno.

Un vehículo autónomo, es un vehículo capaz de imitar las capacidades humanas de manejo y control. El conductor

podrá elegir el destino, pero no se le requiere para activar ninguna operación mecánica del vehículo.

Los vehículos autónomos perciben el entorno mediante sensores tales como láser, radar, lidar, sistema de posicionamiento global y visión computarizada. Los sistemas avanzados de control interpretan la información para identificar la ruta apropiada, así como los obstáculos y la señalización relevante. Los vehículos autónomos generalmente son capaces de recorrer carreteras previamente programadas y requieren una reproducción cartográfica del terreno, con lo cual si una ruta no está recogida por el sistema se puede dar el caso que no pueda avanzar de forma coherente y normal.

II. ARQUITECTURA HARDWARE DEL SISTEMA

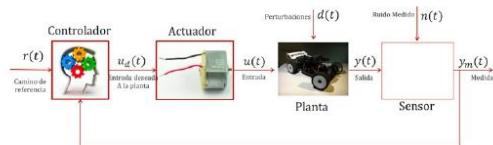


Figura 1 - Modelo matemático de un vehículo autónomo

En la literatura consultada [7] se ha verificado la necesidad de comprender cómo el modelo de la planta, así como sus restricciones, así como la compresión de los marcos referenciales que utilizados por un observador para medir la posición y otras medidas físicas de un sistema físico. Se ha realizado la identificación de la planta utilizando una operación *off-line* (acoplamiento indirecto), en el cual se almacenan los datos adquiridos y posteriormente se transfieren al ordenador para ser evaluados y procesados.

El sistema a ser implementado consta de:

- Vehículo Autónomo a escala 1:8: Chasis modificado de un auto eléctrico miniaturizado en el cual se adapta un myRIO 1900 así como los sensores y actuadores requeridos para navegación

inercial (IMU+GPS). En esta plataforma se ejecutarán los algoritmos que permitan determinar la información de posición y orientación con base a los sensores. Así mismo permitirá la adquisición de datos de los sensores, ejecutar el algoritmo de navegación seleccionado y determinar las señales de control a los actuadores.

- b. Estación Base: Para configuración de los parámetros de navegación (waypoints) y visualización de estado de los sensores.

La plataforma hardware consta de los siguientes elementos:

A. Crius Crius AIOB v2.1

Esta placa electrónica tipo MARG (Magnetic, Angular rate and Gravity) tiene incorporado varios sensores como un giroscopio/accelerómetro MPU6050 de 6 ejes, un altímetro de alta precisión MS5611-01BA01, y un magnetómetro HMC5883L de 3 ejes. El microcontrolador integrado es un ATMEGA 2560 de 8 bits, 16 MHz y se comunica con los dispositivos externos mediante los pines y puertos seriales.

B. Sensor: MPU6050 : Acelerómetro y giroscopio

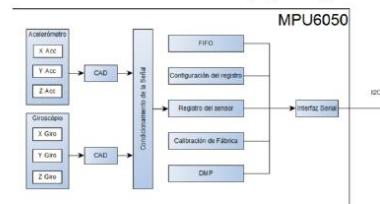


Figura 2 - Arquitectura del MPU 6050 [8]

En la Figura 2 se presenta la arquitectura del MPU6050 [9], el sensor posee conversores analógicos digitales para cada uno de los ejes y así obtener los valores en simultáneo con un rango de hasta $\pm 2000^\circ$ por segundo en el caso del giroscopio y de $\pm 16g$ para el acelerómetro. A continuación, los datos se filtran de acuerdo a la configuración preestablecida y la calibración de fábrica, para pasar luego al registro del sensor, los datos pueden ser accedidos por el DMP o por el usuario. El DMP actualiza los datos del FIFO leídos a una frecuencia determinada con el fin de evitar el desbordamiento. La interfaz serial de comunicación del MPU6050 es el protocolo de comunicación I2C.

C. Sensor: HMC5883L: Magnetómetro integrado

El magnetómetro integrado en el controlador es el HMC5883L de Honeywell, el sensor triaxial tiene un campo de operación de -8 a +8 gauss lo que de esta manera se obtiene la dirección real del norte geográfico considerando la inclinación de 15 grados que hay en la zona. La interfaz serial del HMC5883L cumple el protocolo de comunicación I2C a 400 kHz.

D. Sensor de velocidad: Encoder FC03

Voltaje de Operación: 3.3V - 5V DC
Salidas: Analogica y Digital TTL
Sensor: MOCH22A
Modelo Placa: FC-03/FZ0888
Tipo de emisor: Fotodiodo IR

Tipo de detector: Fototransistor

Longitud de onda del emisor: 950 nm (infrarrojo)

Peso: 8 g

Dimensiones: 32*14*7 mm

Ranura de 5 mm

Comparador Opamp: LMS393

Led indicador de alimentación

Led indicador de pulso

Salida TTL ON: sensor bloqueado

Salida TTL OFF: sensor desbloqueado

Conocer la posición o velocidad de un motor es muy importante en robótica, para lo cual existen diversas alternativas, siendo una de las más comunes el uso de encoders de tipo óptico. Los encoders incrementales ópticos realizan la medición de movimiento con el uso de un haz de luz infrarrojo que se ve interrumpido por las ranuras de un disco acoplado al eje. La cantidad de ranuras por vuelta determinará la precisión del encoder, en este caso de 4 pulsos por vuelta.

E. Actuador: Servo digital para dirección HB-5514 14kg



Figura 3 - Servo digital para dirección

Es un dispositivo actuador que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y de mantenerse estable en dicha posición.

F. Actuador: Motor BLDC 2200KV (rmp/V)

Son motores síncronos alimentados por CC través de un inversor o fuente de alimentación de commutación que produce una corriente eléctrica de CA para controlar cada fase del motor a través de un controlador de circuito cerrado. El controlador proporciona pulsos de corriente a los devanados del motor que controlan la velocidad y el par del motor.

G. Actuador: Control de velocidad ESC 100A

Un control de velocidad electrónico o ESC es un circuito electrónico que controla y regula la velocidad de un motor eléctrico. También puede proporcionar la inversión de giro del motor y el frenado dinámico. Los controles electrónicos de velocidad en miniatura se utilizan en modelos controlados por radio y con alimentación eléctrica. Los vehículos eléctricos de tamaño completo también tienen sistemas para controlar la velocidad de sus motores de accionamiento.

H. Planta: Chasis Haboo Hyper VS 1/8



Figura 4 - Chasis

Dimensión: 460 mm. x 306 mm. x 140 mm.
 Distancia entre ejes: 322 mm.
 Peso: 4720 g
 Batería para myRIO: LI-PO 2S 3000 mAh 7.4 v
 Batería para motor: LI-PO 4S 5400 mAh 14.8 v
 Chasis de aluminio anodizado.
 Torreta de aluminio delantera 4mm y trasera 3mm
 Soporte de suspensión reforzado
 Soporte de baterías con velcro
 Amortiguadores Big Bore de 17mm

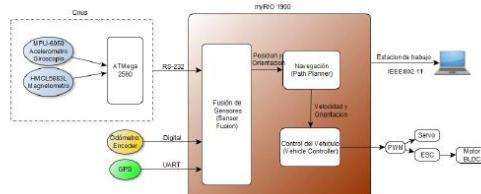


Figura 5 - Arquitectura del Hardware

III. ARQUITECTURA SOFTWARE

En esta sección se enfocará más en el tratamiento de los datos de los sensores para obtener los datos de posición y orientación, y la utilización de los mismos para el cálculo de la trayectoria de recorrido.

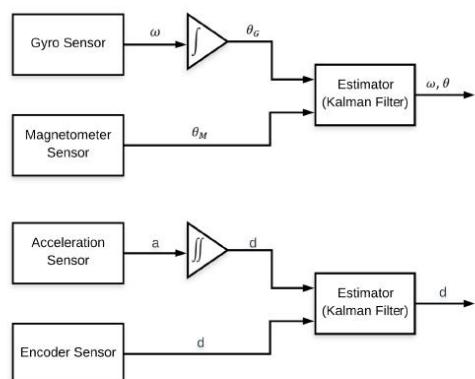


Figura 6 - Visión general del Sistema de fusión

A. Sensor Fusión

Se utilizaron señales de los sensores para mejorar y corregir la medición de la posición propia del vehículo

autónomo de 4 ruedas para obtener una estimación de posición más confiable. A partir de esto, calculamos la estimación de la posición y redujimos los errores sistemáticos y no sistemáticos durante las pruebas y tuvimos éxito en estimar la desviación del sesgo del giro. La herramienta básica aquí es un filtro de Kalman.

En la Figura 6, ω es el dato de la velocidad angular proveniente del giroscopio, a es la aceleración del acelerómetro y θ es el ángulo referente al norte magnético obtenido del magnetómetro.

B. Path Planner

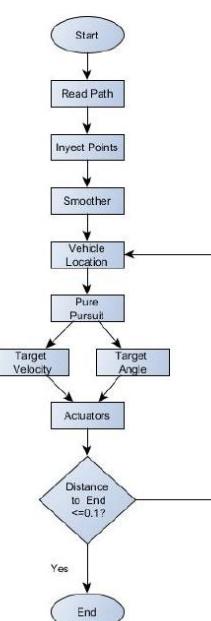


Figura 7 - Planificador de ruta

Este módulo se encarga de llevar al vehículo autónomo desde una posición inicial a una final siguiendo una trayectoria. El algoritmo utilizado es el denominado de persecución pura (pure pursuit algorithm [14]). A efectos prácticos se ha recurrido a la implementación del "Team 1712" [15] y modificada según el requerimiento.

Con este algoritmo se consigue determinar la velocidad objetivo del vehículo autónomo dependiendo de la curvatura del segmento de la trayectoria en el cual se encuentra el vehículo autónomo, como también, establecer la dirección a la cual debe dirigirse conociendo su posición actual y un punto objetivo llamado "Look Ahead Point".

Para obtener mayor precisión en el cálculo de la velocidad y la curvatura, se injectaron puntos a la trayectoria original obteniendo de esta manera puntos más

cercanos, para luego pasarlos por una etapa de suavizado y lograr la continuidad en el trayecto.

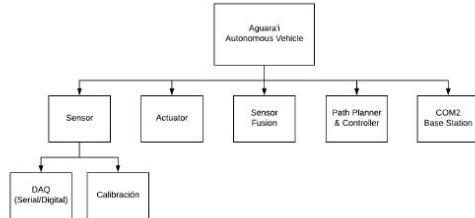


Figura 8 - Arquitectura del Sistema

IV. PRUEBAS REALIZADAS

A. FUSIÓN DE SENSORES

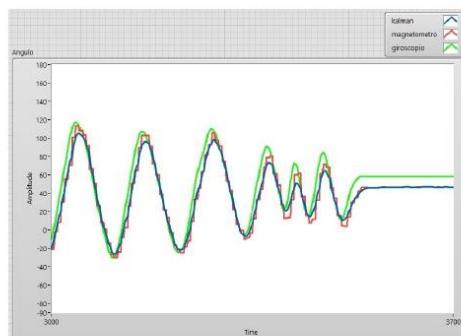


Figura 9 - Filtro de Kalman para la orientación

En la Figura 9, se muestra el resultado de la fusión del giroscopio y el magnetómetro. En ella, se puede observar que el magnetómetro no posee buena respuesta a frecuencias altas, en cambio, a baja frecuencia, la respuesta es buena. Por otro lado, el comportamiento de la curva del giroscopio es suave, pero el error acumulativo aumenta y no hay manera de corregirlo sin realizar la fusión de los sensores.

B. PATH PLANNER

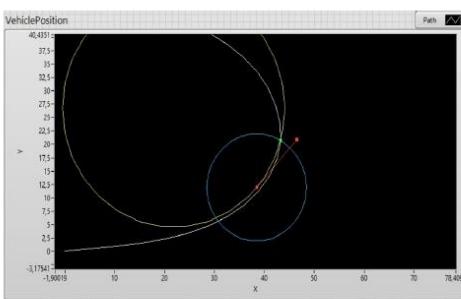


Figura 10 - Simulación del algoritmo persecución pura

Durante las pruebas se evidenció que el algoritmo de persecución pura arroja buenos resultados en la determinación de seguimiento de la trayectoria

establecida. Debido a su robustez, esta permite que se tengan algunos errores en la adquisición de los datos de los sensores o en la sintonización del control de velocidad y dirección, sin embargo como tiene en cuenta una realimentación de un estado anterior, permite actualizar los actuadores con valores coherentes por lo cual se van atenuando los pequeños errores de orientación y posición.

CONCLUSIONES

Se ha modificado el chasis de un vehículo eléctrico a escala 1:8 e implementado un algoritmo de navegación basado en la estrategia denominada de persecución pura. La plataforma utilizada para estos trabajos a sido el myRIO 1900 con el entorno de desarrollo de LabVIEW. El trabajo se ha implementado en dos meses y medio, utilizando ejemplos de aplicaciones de National Instruments en conjunto con librerías del “Team 1712”, lo cual nos ha permitido realizar las primeras pruebas operativas del prototipo, con un error aceptable.

El proyecto recalca el valor pedagógico de la enseñanza basada en retos, permitiendo a los alumnos adquirir la experiencia de trabajo en equipo, en un entorno multidisciplinario así como la experiencia de reutilización de código de terceros. También ha permitido que los alumnos comparen los conceptos desarrollados en distintas materias de la universidad al calibrar los distintos sensores, a mejorar la lectura minimizando los errores mediante la implementación del filtro de kalman.

La plataforma de trabajo seleccionada también permitió que alumnos de diferentes años puedan acceder al mismo conocimiento, desde el punto de vista de implementación de un algoritmo, al permitirles obtener la certificación CLAD, con lo cual no solo han aprendido el lenguaje de programación, también se han enfocado en la metodología de trabajo en un proyecto específico.

RECONOCIMIENTOS

Se agradece al apoyo de las siguientes instituciones: Universidad Católica “Nuestra Señora de la Asunción”, Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada, Fundación Parque Tecnológico Itaipú – Paraguay.

REFERENCIAS

- [1] Iyengar, D., & Peters, D. L. (2015, October). Development of a miniaturized autonomous vehicle: Modification of a 1:18 scale rc car for autonomous operation. In ASME 2015 Dynamic Systems and Control Conference (pp. V003T50A008-V003T50A008). American Society of Mechanical Engineers.
- [2] McBride, J. (2007). Darpa urban challenge.
- [3] Umnson, C., Bagnell, J. A., Baker, C. R., Hebert, M., Kelly, A., Rajkumar, R., & Team, D. U. C. (2007). Tartan racing: A multi-modal approach to the DARPA
- [4] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Etinger, S., & Thrun, S. (2008). Junior: The Stanford entry in the Urban Challenge. Journal of field Robotics, 25(9), 569-597
- [5] The Economist. (2013). Look, no hands. Extraído de <http://www.economist.com/news/special-report/21576224-one-day-every-car-may-come-invisible-chauffeur-look-no-hands>

- [6] Dockterman, E. (2015). Google's self-driving car may come with airbags on the outside. Time Magazine. Extraido de <http://time.com/3758446/googles-self-driving-car-may-come-with-airbags-on-the-outside/>
- [7] The DuckieTown project. (2017). Extraido de <https://www.duckietown.org/>
- [8] Benítez, W., & Bogado, Y. (2015). Desarrollo de un prototipo de VANT (Vehículo Aéreo No Tripulado) para inspección visual de líneas eléctricas aéreas (Tesis de Grado). Universidad Católica "Nuestra Señora de la Asunción" Campus Alto Paraná Paraguay.
- [9] MPU-6000 and MPU-6050 Product Specification Revision 3.4. Sunnyvale, California, Estados Unidos. Extraido de https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf
- [10] EureCar, KAIST Self-Driving car. Extraido de <https://forums.ni.com/t5/Projects-Products/EureCar-KAIST-Self-Driving-car/ta-p/3517884>
- [11] Kok, M., Hol, J. D., & Schön, T. B. (2017). Using inertial sensors for position and orientation estimation. *arXiv preprint arXiv:1704.06053*.
- [12] Ozyagcilar, T. (2012). Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. *Freescale semiconductor, AN, 4248*.
- [13] Zunaidi, I., Kato, N., Nomura, Y., & Matsui, H. (2006). Positioning system for 4-wheel mobile robot: encoder, gyro and accelerometer data fusion with error model method. *CMU. Journal*, 5(1).
- [14] Conlter, R. C. (1992). Implementation of the Pure Pursuit Path Tracking Algorithm.
- [15] Implementation of adaptative pure persuit controller. Extraido de <https://www.chiefdelphi.com/media/papers/3488>.

D.2. TDP en proceso de evaluación para ser publicado en el JPAUT



Development of an autonomous vehicle at a 1:8 scale

Ariel Guerrero¹, Micaela Jara², Erid Pacheco², Ariel Bogado², Jesús Franco²

¹Parque Tecnológico de Itaipu – Paraguay, ²Universidad Católica “Nuestra Señora de la Asunción”

ABSTRACT

The development of an autonomous vehicle is the subject of extensive study by many researchers [1]. One of the approaches of study consists of the use of physical models at scale whose construction is the object of this article. For this purpose, a RC (Remotely Controlled) electric car with a 1: 8 scale is modified. A myRIO is used for data acquisition (odometer, accelerometer, magnetometer, gyroscope) and operation tasks. Another task is to merge the data of the sensors in such a way that the position and orientation information can be used for the navigation decision making. This data can be sent to a workstation, running an application in LabVIEW. This workstation provides a human-machine interface for the operator to display the navigation parameters.

Keywords: Autonomous Vehicle, IMU, Navigation Algorithm, Kalman.

INTRODUCTION

Autonomous vehicles have attracted a great deal of interest in research in recent years, as well as important industry development efforts. In 2007, DARPA has executed the Urban Grand Challenge [2], with entries from several universities documented in many different publications, for example, [3, 4].

Many companies in the automotive sector have their own division dedicated to the investigation of autonomous vehicles and recently other companies that are not of the sector such as NVIDIA, APPLE, GOOGLE, YANDEX, BIADU have started to develop their own autonomous vehicle. Projects that have been widely reported in the news [5, 6]. Although important efforts have been made in this field, there are still many problems to be solved, among them, the problems of detection, the different types and levels of control and the interaction of autonomous vehicles with their environment.

An autonomous vehicle is a vehicle capable of imitating the human capacities of management and control. The driver may choose the destination but is not required to activate any mechanical operation of the vehicle. Autonomous vehicles perceive the environment through sensors such as laser, radar, lidar, global positioning system and computer vision. Advanced control systems interpret information to identify the appropriate route, as well as obstacles and relevant signage. Autonomous vehicles are generally capable to travel previously programmed roads and require a cartographic reproduction of the terrain, so if a route is not picked up by the system it is possible that it cannot advance coherently and normally.

METHODOLOGY

The methodology used in this project was the traditional design. In Figure 1, the flow diagram of the tasks performed is presented. The main problem was how to develop, in a short period of time, an autonomous scaled electric car. Based on this approach, possible solutions were analyzed, carrying out the necessary studies and evaluations. Once the objectives to be met were set, the hardware was prepared, in this case the adaptation of the components in the scaled electric car, and the design of the software. The software and hardware implementations were carried out. Subsequently, the necessary tests were carried out in order to verify problems, and if necessary, implement modifications and improvements to the system, whether they were in the software or hardware.

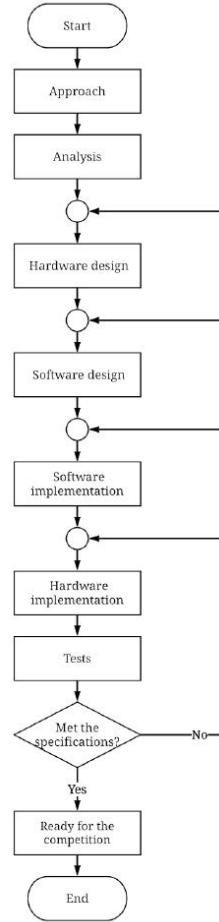


Figure 1 - Project methodology flowchart

DEVELOPMENT

I. SYSTEM HARDWARE ARCHITECTURE

Following the consulted literature [7] we needed to have a mathematical model that describes the behavior of the plant. As it would take more time to reach an accurate model taking into account the short time available to complete the project and the fact that we needed to have the control of both the traction and the direction of the plant, we proceeded to obtain the mathematical model through a transfer function using the "black box" method, which consists in the study of an element from its output behavior for a given entry without covering its internal functioning. The diagram can be seen in figure 1. Knowing the input and output, the transfer function of the "black box" was identified using the Matlab System Identification tool.

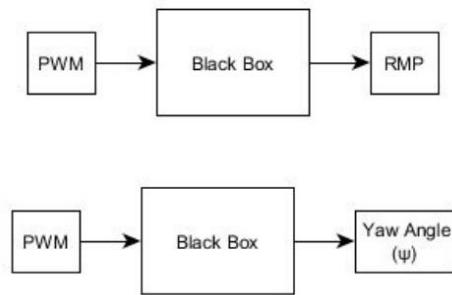


Figure 2 -Powertrain and direction schematic

In the "black box" of the powertrain, there is a brushless motor (BLDC) powered by an electronic speed controller (ESC), which receives a PWM signal to control the speed of the motor. On the other hand, in the "black box" of the steering, there is a servo motor, which receives another PWM signal to control the Yaw Angle of the front wheels, and consequently control the direction of the car. The result obtained through this method had an approximation of 70% for the powertrain and 87% for the direction, which was enough to be able to control the plant.

The system to be implemented consists of:

- Autonomous Vehicle at 1: 8 scale: Modified chassis of a miniaturized electric car in which a myRIO 1900 is adapted as well as the sensors and actuators required for inertial navigation (IMU + GPS). In this platform the algorithms that allow determining the position and orientation information based on the sensors will be executed. It will also allow the acquisition of data from the sensors, execute the selected navigation algorithm and determine the control signals to the actuators.
- Base Station: For configuration of the navigation parameters (waypoints) and visualization of the status of the sensors.

The hardware platform consists of the following elements:

A. Crius AIOP v2.1

This electronic board type MARG (Magnetic, Angular rate and Gravity) has several built-in sensors such as a gyroscope /accelerometer MPU6050 6-axis, a high-precision altimeter MS5611-01BA01, and a magnetometer HMC5883L 3-axis. The integrated microcontroller is an ATMEGA 2560 8-bit, 16 MHz and communicates with external devices through pins and serial ports.

1) Sensor: MPU6050 : Accelerometer and gyroscope

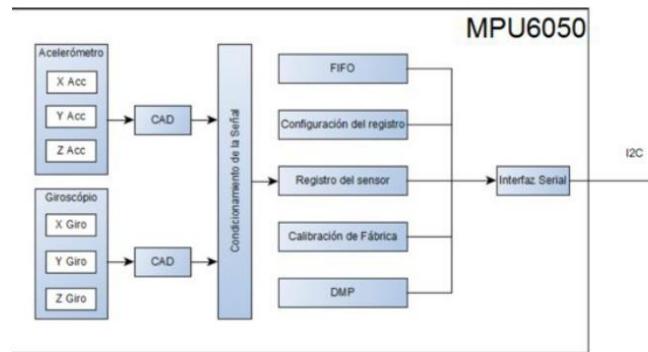


Figure 3 - Architecture of the MPU 6050 [8]

Figure 3 shows the architecture of the MPU6050 [9], the sensor has digital analog converters for each of the axes and thus obtains the values simultaneously with a range of up to ± 2000 per second in the case of the gyroscope and $\pm 16g$ for the accelerometer. Then, the data is filtered according to the preset configuration and the factory calibration, then it goes to the sensor register, the data can be accessed by the DMP or by the user. The DMP updates the FIFO data read at a certain frequency in order to avoid overflow. The communication serial interface of the MPU6050 is the I2C communication protocol.

2) Sensor: HMCL5883L: Integrated magnetometer

The magnetometer integrated in the controller is the Honeywell HMC5883L, the triaxial sensor has an operation field of -8 to +8 gauss, which in this way gives the real direction of the geographic north considering the 15 degree inclination that exists in the zone. The serial interface of the HMC5883L complies with the I2C communication protocol at 400 kHz.

B. Odometer and Speed Sensor Encoder FC03

Operating voltage: 3.3V - 5V DC

Outputs: Analogica y Digital TTL

Sensor: MOCH2A

Board model: FC-03/FZ0888

Type of emitter: Photodiode IR

Detector type: Phototransistor

Wavelength of the emitter: 950 nm (infrared)

Weight: 8 g

Dimensions: 32*14*7 mm

Slot: 5 mm

Opamp comparator: LMS393

Power indicator LED

Pulse indicator LED

Output TTL ON: blocked sensor

Output TTL OFF: unlocked sensor

Knowing the position or speed of an engine is very important in robotics, for which there are several alternatives, one of the most common being the use of optical type encoders. The incremental optical encoders perform the measurement of movement with the use of an infrared beam that is interrupted by the slots of a disk coupled to the shaft. The number of slots per revolution will determine the encoder's accuracy, in this case 4 pulses per revolution.

C. Actuator: Digital Servo HB-5514 14kg



Figure 4 - Digital servo for direction

It is an actuator device that has the ability to be located in any position within its operating range, and to remain stable in that position.

D. Actuator: Motor BLDC 2200KV (rmp/V)

They are synchronous motors fed by DC through an inverter or switching power supply that produces an AC electric current to control each phase of the motor through a closed circuit controller. The controller provides pulses of current to the motor windings that control the speed and torque of the motor.

E. Actuator: Electronic Speed Controller ESC WP-8BL100, 100A

An electronic speed controller or ESC is an electronic circuit that controls and regulates the speed of an electric motor. It can also provide reversing of the motor and dynamic braking. ESCs are often used on motors essentially providing an electronically-generated three-phase electric power low voltage source of energy for the motor.

F. Mechanical Plant: Chasis Haboo Hyper VS 1/8



Figure 5 - Chassis

Dimension: 460 mm. x 306 mm. x 140 mm.

Distance between axis: 322 mm.

Weight: 4720 g

Battery for myRIO: LI-PO 2S 3000 mAh 7.4 v

Motor battery: LI-PO 4S 5400 mAh 14.8 v

Anodized aluminum chassis.

Front aluminum turret 4mm and rear 3mm

Reinforced suspension support

Battery holder with velcro

Big Bore 17mm shock absorbers

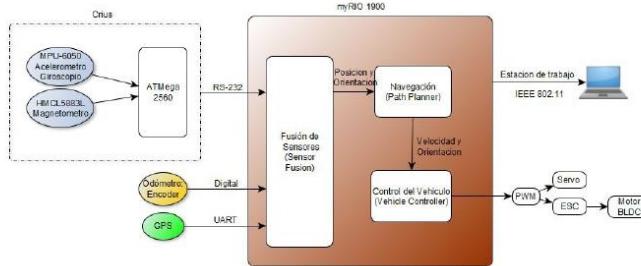


Figure 6 - Hardware Architecture

II. SOFTWARE ARCHITECTURE

In this section, we will focus more on the processing of sensor data to obtain the position and orientation of the vehicle, and the use of them for the calculation of the path of travel. Before the fusion of sensors, the collected data go through a calibration process, where offset and gain errors are eliminated, this process can be found in [8], then the data goes through a process of changing the reference, passing from a fixed frame of reference to the autonomous vehicle, to which the inertial sensors belong, to a frame of reference fixed to the ground, known as the navigation reference frame.

A. Sensor Fusion

Sensor signals were used to improve and correct the position measurement of the 4-wheel autonomous vehicle to obtain a more reliable position estimate. From this, we calculated the estimation of the position and reduced the systematic and non-systematic errors during the tests and we succeeded in estimating the deviation of the turn bias. The basic tool here is a Kalman filter.

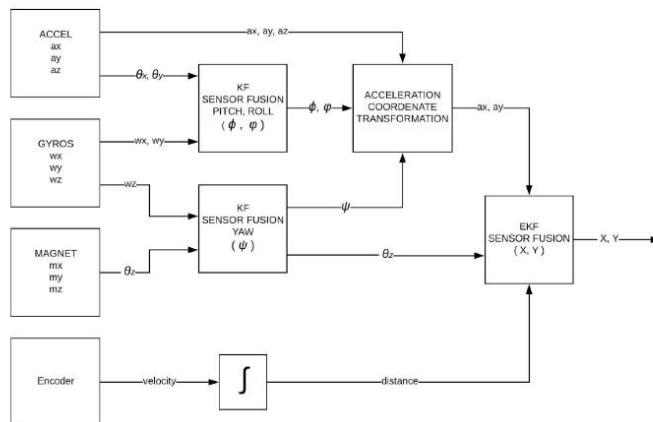


Figure 7 - Overview of the fusion system

As shown in figure 7; initially, a linear Kalman filter (KF) is used to merge the data of the accelerometer and the gyroscope, with this we obtain the pitch (ϕ) and roll (ψ) angles, the yaw angle (ψ) is obtained from the fusion of the gyroscope with the magnetometer, also through a linear Kalman filter (KF) [14].

In Figure 8, the result of the fusion of the gyroscope and the magnetometer is shown. In the figure, it can be seen that the magnetometer does not have good response at high frequencies, however, at low frequency, the response is good. On the other hand, the behavior of the gyroscope curve is smooth, but the cumulative error increases and there is no way to correct it without performing the fusion of the sensors.

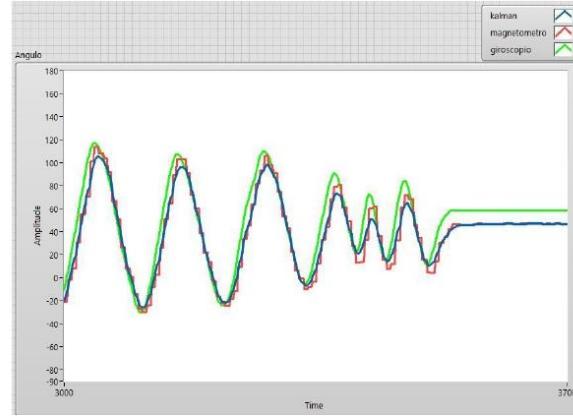


Figure 8 - Kalman filter for orientation

Finally, the encoder is added to the system, in order to have a better estimate of the distance traveled, an extended Kalman filter (EKF) is used, taking into account the non-linearity of the estimation by means of the odometry model [15].

This model is subject to cumulative errors that increase with time, because there is no external reference. These errors can be minimized by integrating a GPS (Global Position Systems) into the system, but this is beyond the scope of this work [16].

B. Path Planner

This module is responsible for taking the autonomous vehicle from an initial position to a final, following a trajectory. The algorithm used is the so-called pure pursuit algorithm [17]. For practical purposes, the implementation of "Team 1712" [18] has been used and modified according to the requirement.

With this algorithm it is possible to determine the target speed of the autonomous vehicle depending on the curvature of the segment of the trajectory in which the autonomous vehicle is located, as well as to establish the direction to which it should go knowing its current position and a target point called "Look Ahead Point" (Figure 10).

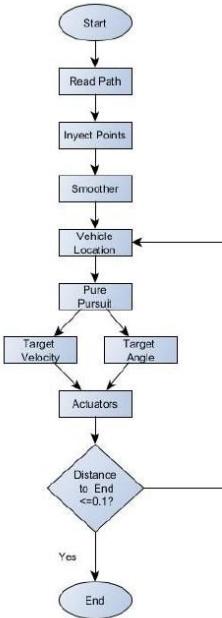


Figure 9 - Route planner

The "Look Ahead Point" is a fundamental parameter in the application of this algorithm, since by varying its value it is possible to vary: the response of the car to deviations from the wanted trajectory, and the stability in which the car follows its trajectory preventing oscillations. Its value can be static or dynamic, that is, static if its value is predetermined by the programmer before the car starts its trajectory, or dynamic when its value depending on characteristics such as the speed of the car and the curvature of the trajectory make its value to get the best response. In this project the static was applied due to its simplicity and rapid implementation.

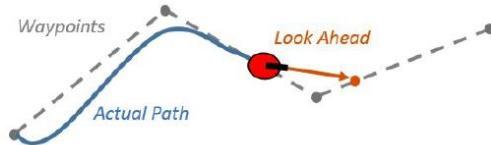


Figure 10 - Look Ahead Distance [19]

With regard to its value, the choice of a small number will cause the vehicle to quickly seek to approach the desired trajectory, however, as a consequence, the car begins to oscillate in search of the trajectory as shown in Figure 11.

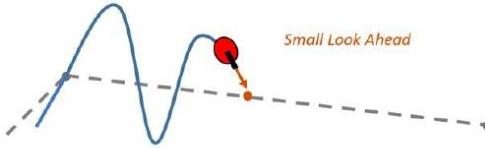


Figure 11 - Small Look Ahead [19]

On the other hand, a choice of a large value will cause the car to stop oscillating, however the response to sudden variations in the trajectory becomes very slow, as does the curvature in which the car follows its trajectory (Figure 12).

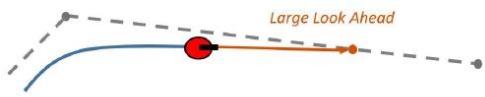


Figure 12 - Large Look Ahead [19]

To obtain greater precision in the calculation of the speed and curvature, points were injected into the original trajectory obtaining in this way closer points, then, to pass them through a smoothing stage and achieve continuity in the path.

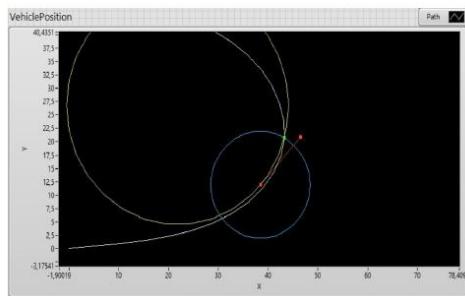


Figure 13 - Simulation of Pure Pursuit Algorithm

FINDINGS

During the tests it was evidenced that the pure pursuit algorithm gives good results in the determination of follow-up of the established trajectory. Due to its robustness, this allows some errors in the acquisition of sensor data or in the tuning of the speed and direction control, however, as it takes into account feedback from a previous state, it allows to update the actuators with consistent values so that small errors of orientation and position are attenuated.

The choice of the best value for the Look Ahead Point was made through experimentation finding that for the speed 2 m/s, a value of Look Ahead Point equal to 1.8m, prevents oscillations and a good response is obtained for more curves closed.

CONCLUSIONS

Autonomous navigation was validated in the tests, demonstrating its effectiveness in tracking a defined trajectory. A set of tests was performed varying the initial position and the orientation to verify that, despite the different initial conditions, in the same way, the car manages to follow a predefined trajectory, as expected by the results of the simulations. Tests in a controlled environment have obtained satisfactory results for the desired purposes, taking into account that the scaled vehicle was used in a race of autonomous scaled cars. For this purpose, the cumulative errors inherent in the system were reduced so that the influence on the result was negligible. In addition, the inertial navigation algorithm turned out to be very effective compared to other types of navigation used for this purpose, achieving a higher response speed due to its low computational requirement.

However, in uncontrolled environments, it has not been very effective, due to the inability of the linear Kalman filter to eliminate electromagnetic distortions that affect the readings of the magnetometer, these readings with distortion considerably affect the calculation of the yaw angle, and consequently, they produce errors in the calculation of the x and y coordinates.

Bearing in mind that the duration of the race does not generate significant cumulative errors, the use of a magnetometer could be eliminated, and thus make the system less sensitive to disturbances in the magnetic field (with the cost that this entails in the absence of an absolute orientation, which is what the magnetometer offered).

ACKNOWLEDGEMENTS

We thank the support of the following institutions: Universidad Católica "Nuestra Señora de la Asunción", Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada (CICTIA), Fundación Parque Tecnológico Itaipu – Paraguay and National Instruments Brazil.

REFERENCES

- [1] IYENGAR, D., & PETERS, D. L. (2015, October). Development of a miniaturized autonomous vehicle: Modification of a 1: 18 scale rc car for autonomous operation. In ASME 2015 Dynamic Systems and Control Conference (pp. V003T50A008-V003T50A008). American Society of Mechanical Engineers.
- [2] MCBRIDE, J. (2007). Darpa urban challenge.
- [3] URMSON, C., BAGNELL, J. A., BAKER, C. R., HEBERT, M., KELLY, A., RAJKUMAR, R., & TEAM, D. U. C. (2007). Tartan racing: A multi-modal approach to the DARPA
- [4] MONTEMERLO, M., BECKER, J., BHAT, S., DAHLKAMP, H., DOLGOV, D., ETTINGER, S., & THRUN, S. (2008). Junior: The Stanford entry in the Urban Challenge. Journal of field Robotics, 25(9), 569-597
- [5] THE ECONOMIST. (2013). Look, no hands. Recovered from The economist: <http://www.economist.com/news/special-report/21576224-one-day-every-car-may-come-invisible-chauffeur-look-no-hands>
- [6] DOCKTERMAN, E. (2015). Google's self-driving car may come with airbags on the outside. Time Magazine. Recovered from Time: <http://time.com/3758446/googles-self-driving-car-may-come-with-airbags-on-the-outside/>
- [7] THE DUCKETOWN FOUNDATION. (2017). The Duckietown Project. Recovered from Duckietown: <https://www.duckietown.org/>

- [8] BENÍTEZ, W., & BOGADO, Y. (2015). Desarrollo de un prototipo de VANT (Vehículo Aéreo No Tripulado) para inspección visual de líneas eléctricas aéreas (Tesis de Grado). Universidad Católica "Nuestra Señora de la Asunción" Campus Alto Paraná. Paraguay.
- [9] INVENSENSE. (2013). MPU-6000 and MPU-6050 Product Specification Revision 3.4. Sunnyvale, California, United States of America. Recovered from Invensense: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf
- [10] LEE, U., OH, J., SHIN, S., SHIM, I., CHOI, J., JUNG, Y., PARK, K., KIM, M., & JUNG, J. (2014). EureCar, KAIST Self-Driving car. Recovered from National Instruments: <https://forums.ni.com/t5/Projects-Products/EureCar-KAIST-Self-Driving-car/ta-p/351784>
- [11] KOK, M., HOL, J. D., & SCHÖN, T. B. (2017). Using inertial sensors for position and orientation estimation. *arXiv preprint arXiv:1704.06053*.
- [12] OZYAGCILAR, T. (2012). Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. *Freescale semiconductor, AN, 4248*.
- [13] ZUNAIDI, I., KATO, N., NOMURA, Y., & MATSUI, H. (2006). Positioning system for 4-wheel mobile robot: encoder, gyro and accelerometer data fusion with error model method. *CMU. Journal, 5*(1).
- [14] VIGOUROUX CAVOLINA, D. P. (2010). Implementación de unidad de mediciones iniciales (IMU) para robótica utilizando filtro de Kalman. Sartenejas, Venezuela.
- [15] FISCHER, T., NITSCHE, M. A., & PEDRE, S. (2014). Fusión de encoders de cuadratura, sensores iniciales y magnéticos para la localización de robots móviles. Facultad de Ciencias Exactas y Naturales - UBA, Buenos Aires.
- [16] MOHINDER S., G., & ANGUS P., A. (2008). Kalman Filtering: Theory and Practice Using MATLAB (Third ed.). Hoboken, New Jersey: JOHN WILEY & SONS, INC.
- [17] Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm (No. CMU-RI-TR-92-01). Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.
- [18] FRC TEAM 1712 (2018). Implementation of adaptive pure pursuit controller. Recovered from chiefdelphi: <https://www.chiefdelphi.com/media/papers/3488>.
- [19] MATHWORKS (w/d). Pure Pursuit Controller. Recovered from MathWorks: <https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html>

CONTACT INFORMATION

Gregorio Ariel Guerrero Moral (corresponding author)
ariel.guerrero@pti.org.py

Micaela Carolina Jara Ten Kathen
micaela.jara@uc.edu.py

Erid Eulogio Pacheco Viana
erid.pacheco@uc.edu.py

Ariel David Bogado Arce
ariel.bogado@uc.edu.py

Jesús María Franco Santacruz
jesus.francosantacruz@uc.edu.py

Apéndice E

Proyecto presentado en el Labview Student Design Competition

Contact Information:

Country: Paraguay

Year Submitted: 2019

University: Universidad Católica "Nuestra Señora de la Asunción"

List of Team Members (with year of graduation):

- Micaela Carolina Jara Ten Kathon (2019)
- Ariel David Bogado Arce (2019)
- Erid Eulogio Pacheco Viana (2019)
- Jesús María Franco Santacruz (2020)

Faculty Advisers: Prof. Gregorio Ariel Guerrero Moral

Main Contact Email Address: ariel.guerrero@uc.edu.py

Project Information:

Title: Development of an autonomous vehicle at a 1:8 scale

Description:

The challenge of the Robocar Race 2018 competition held in San Paulo-Brazil is to design an autonomous scale vehicle that can compete in 2023 against a radio-controlled car piloted by a man. Our team of students of the Universidad Católica “Nuestra Señora de la Asunción” accepted the challenge and presented a solution based on inertial navigation using the NI myRIO and LabVIEW, being the first students of Paraguay to accept the challenge of building an autonomous car.

Products: NI myRIO 1900 and LabVIEW 2017.

The Challenge:

Brazilian branch of National Instruments challenged LabVIEW Student Ambassadors (LSAs) to design and build an autonomous scale vehicle to compete at the Robocar Race, in return, they would give them a NI myRIO controller. Our ambassador for PTI-UCA accepted the challenge and assembled the first team from Paraguay to develop an autonomous scaled vehicle. Our goal was to win the competition that consisted in running the track in the shortest time. We entered the race with our car called Aguara'i (little fox in Guarani), a Radio-Controlled RC car that we modified to be autonomous. Although we finished in 4th place, our work inspired a lot of engineering students from our country to take part of this exciting and promising field.



Members of the team with Robocar Race 2018 juices.

The Solution:



*Project Overview - The resulting performance of Aguara'i **

System Configuration

Autonomous Vehicle at 1: 8 scale: Modified chassis of a miniaturized electric RC car in which a myRIO 1900 is adapted as well as the sensors and actuators required for inertial navigation (IMU). In this platform the algorithms that allow determining the position and orientation information based on the sensors will be executed. It will also allow the acquisition of data from the sensors, execute the selected navigation algorithm and determine the control signals to the actuators.

Base Station: For configuration of the navigation parameters (waypoints) and visualization of the status of the sensors. Human-machine interface with LabVIEW.



"Aguara'i" Autonomous Car

Using LabVIEW and myRIO

The first step is to acquire the signals of the sensors. The encoder was connected to one of the FPGA's (Field-Programmable Gate Array) inputs, and the IMU sensors was connected through a serial connection from an arduino microcontroller (ATMega2560) to the myRIO CPU.

In order to control the speed and direction of the car, we made use of the parallelism of the loops in the FPGA, these PWMs (Pulse Width Modulation) values to control the direction and the speed came from constantly updated values in the myRIO. To communicate this data from the RT-target to FPGA-target we use global variables, since we needed that several processes running in parallel in LabVIEW, can communicate their data and in this way obtain the processing and control of the state variables of the car in real time.

Next, the configuration of the system can be seen in Figure 1.

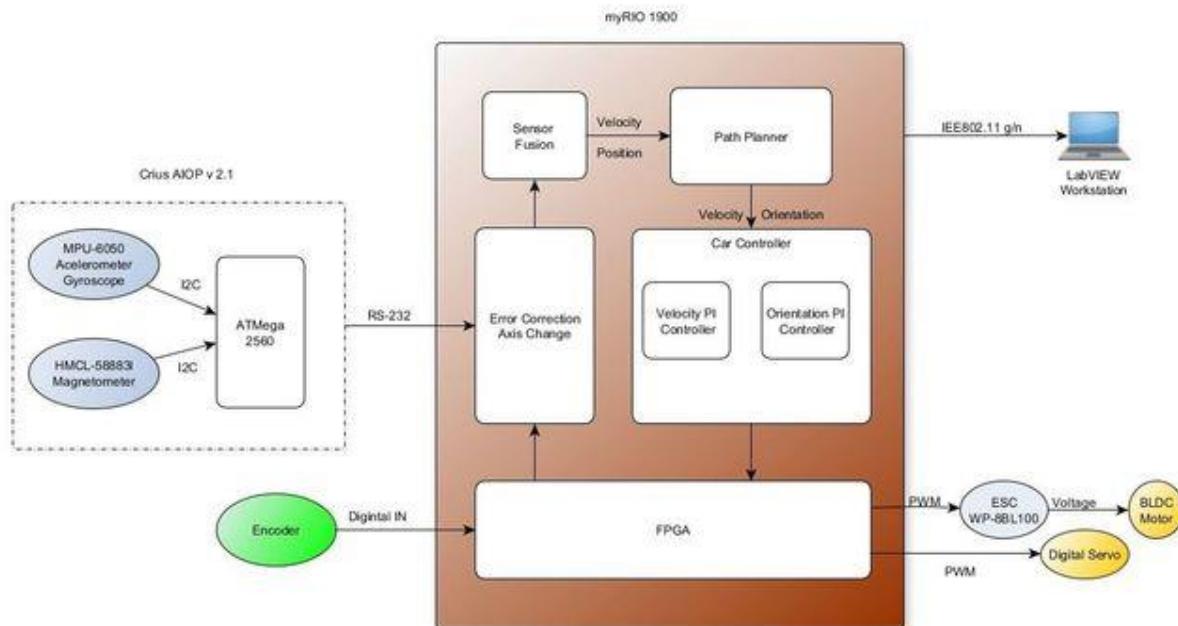


Figure 1 - Aguara'i Architecture

Sensor Fusion

Sensor signals were used to improve and correct the position measurement of the 4-wheel autonomous vehicle to obtain a more reliable position estimate. From this, we calculated the estimation of the position and reduced the systematic and non-systematic errors during the tests and we succeeded in estimating the deviation of the turn bias. The basic tool here is a Kalman filter.

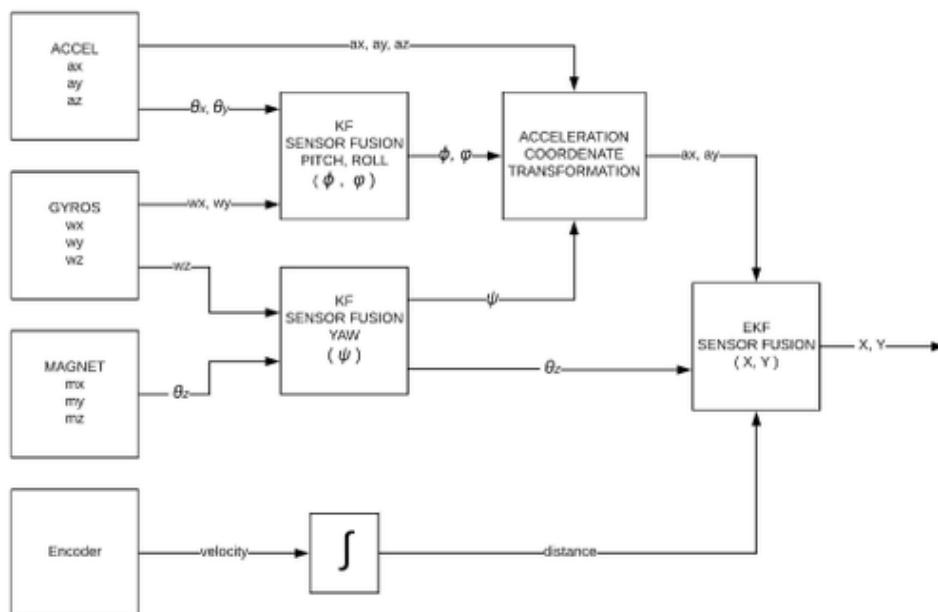


Figure 2 - Overview of the Fusion System

As shown in figure 2; Initially, a linear Kalman filter (KF) is used to merge the data of the accelerometer and the gyroscope, with this we obtain the pitch (ϕ) and roll (θ) angles, the yaw angle (ψ) is obtained from the fusion of the gyroscope with the magnetometer, also through a linear Kalman filter (KF).

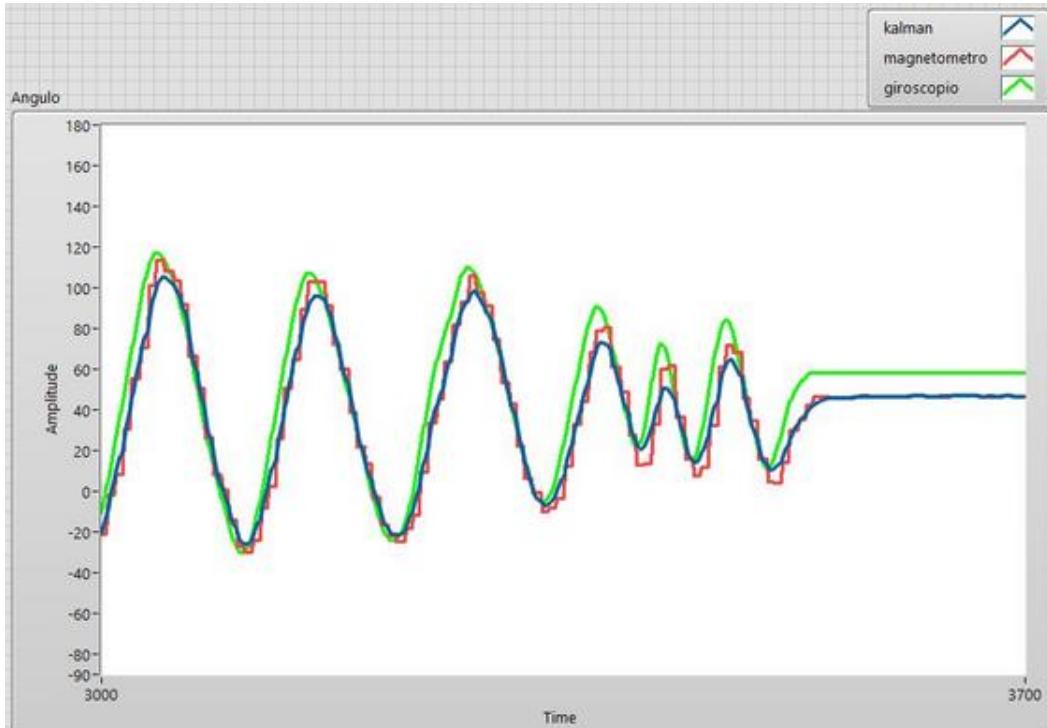


Figure 3 - Kalman filter for orientation

In Figure 3, the result of the fusion of the gyroscope and the magnetometer is shown. In it, the magnetometer does not have good response at high frequencies, however, at low frequency, the response is good. On the other hand, the behavior of the gyroscope curve is smooth, but the cumulative error increases and there is no way to correct it without performing the fusion of the sensors.

Finally, the encoder is added to the system, in order to have a better estimate of the distance traveled, an extended Kalman filter (EKF) is used, considering the non-linearity of the estimation by means of the odometry model.

This model is subject to cumulative errors that increase with time, because there is no external reference. These errors can be minimized by integrating a GPS (Global Position Systems) into the system, but this is beyond the scope of this work.

Path Planner

This module is responsible for taking the autonomous vehicle from an initial position to a final, following a trajectory. The algorithm used is the so-called pure pursuit algorithm. For practical purposes, the implementation of "Team 1712" has been used and modified according to the requirement.

With this algorithm it is possible to determine the target speed of the autonomous vehicle depending on the curvature of the segment of the trajectory in which the autonomous vehicle is

located, as well as to establish the direction to which it should go knowing its current position and a target point called "Look Ahead Point".

To obtain greater precision in the calculation of the speed and curvature, points were injected into the original trajectory obtaining in this way closer points, to then passing through a smoothing stage and achieve continuity in the path.

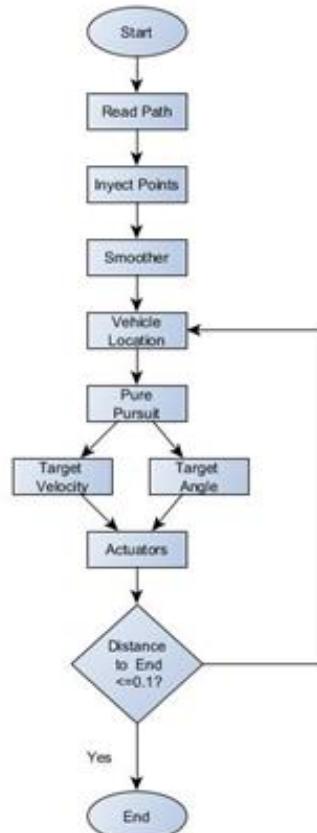


Figure 4 - Path Planner

The Conclusion:

If it wasn't for the NI platform, which made us achieve a very high level of development without entering into hardware programming and allowed us to make use of modularity even using third-party code, we wouldn't have been able to develop the car in time for the competition.

This project was not carried out only by five people, this project involved the entire university community, students of electromechanical engineering, architecture and teachers of the university. In the Aguara'i, the effort of each one of them, colleagues who worked with the members of the team during the morning, afternoon and night is put. This experience not only brought intellectual growth but also growth in teamwork. We learned that there are always people willing to help for the simple fact of being necessary. With the project it was possible to inspire the students of the inferior courses of the career and even to high school students, in the exciting area of the autonomous cars.

The Technical Description Paper (TDP), presented in Robocar Race 2018 competition, was selected, along with two others TDP, to be published in a journal of São Paulo, Journal of Production and Automation, besides, the paper achieved the first place in the Research Day

organized by the Center for Research in Sciences, Technology and Advanced Innovation (CICTIA) belonging to the institution. This project allowed to start a new line of research.

Time to Build:

This project was carried out in two months. The period in which it was made is September 2018 - October 2018.

*El video puede ser encontrado en el link que se encuentra en la referencia Guerrero, Jara, Bogado, Pacheco, & Franco, (2019).

Apéndice F

Invitación al evento NIWeek 2019



Dear Micaela Jara,

It is with great satisfaction we would like to invite you for attending **NIWeek 2019**, our annual user conference in Austin, Texas from May 20-23. Given your engagement and accomplishments using our technology along with the fact you are enrolled in an academic institution makes you a valuable participant and contributor for the conference, it will be a pleasure to have your presence participating and contributing to the event through discussions with peers from other universities and industry from all parts of the world.

A Full Conference Pass for NI Week 2019 costs US\$ 1,095.00 if purchased before May 19th, due to your commitment, outstanding accomplishments and positive impact at the Robocar Race in 2018, National Instruments is glad to offer you a free Full Conference Pass for NI Week this year.

Overall, the NIWeek conference provides:

- More than 200 hours of technical, case study, and panel sessions
- 40+ hands-on product sessions
- 50+ training, certifications, and badging opportunities
- Over 100 exhibitors featuring the latest test and measurement advancements
- Keynote presentations from industry thought leaders

I thought you might find these sessions in the Academic track valuable:

- Using Record Players and LabVIEW to Teach PID Control
- Industry-University Partnerships
- Increasing Undergraduate Exposure to Interdisciplinary Design

See the attached flyer for an overview of all sessions and presenters.

At the NIWeek Expo, I think you should visit our Academic pavilion to see these demos:

- Oklahoma State University's Endeavor Labs
- New Solutions for Teaching Circuits and Measurements
- 16x1 Massive MIMO

If needed I would be happy to set up a quick 10 min follow-up call with a NI team member or even an in-person meeting at an appropriate time if you have any questions on the event.

Looking forward to your participation in this year NI Week edition.

Best Regards,


André Oliveira
Regional Sales Manager
Brazil and Cono Sur

Apéndice G

Nota de Agradecimiento

Hernandarias, 23 de noviembre de 2018

Ing. Juan Carlos Ocampos Núñez
Decano
Facultad de Ciencias y Tecnología
Universidad Católica Nuestra Señora de la Asunción
Campus Alto Paraná

MICHAELA JARA, ERID PACHECO, ARIEL BOGADO y JESÚS FRANCO, alumnos de la Facultad de Ciencias y Tecnología, de la carrera Ingeniería Electromecánica, nos dirigimos a usted y por su intermedio a donde corresponda para manifestar cuanto sigue:

Antes que todo, nuestro más profundo agradecimiento por haber apoyado en todos los aspectos la participación en la corrida de autos autónomos a escala denominado ROBOCAR RACE en la cual participamos como representantes de esta casa de estudios y que fue llevado a cabo en la ciudad de São Paulo – Brasil.

Nuestra participación no hubiera sido posible sin el apoyo de la Universidad por su intermedio, así como del estamento estudiantil y de la sociedad en general inclusive, quienes nos ayudaron a cubrir los gastos que generó nuestra participación en dicho evento, y que fue motivado principalmente por el respaldo de la Universidad.

Agradecemos igualmente a la Universidad Católica “Nuestra Señora de la Asunción” campus Alto Paraná y al director general de campus, el Ing. Ladislao Aranda, por su intermedio, pues como ya hemos manifestado, no hubiera sido posible nuestra participación en dicha competencia, sin su apoyo.

Le comentamos, que igualmente hemos tenido el apoyo del Centro de Estudiantes de la Facultad de Ciencias y Tecnología (CEFACYT), quienes nos apoyaron de forma económica y logística en la organización de una fiesta con el fin de recaudar fondos para el viaje. Esto demuestra el compañerismo y el compromiso con esta casa de estudios, lo que nos ha sorprendido gratamente y queremos poner a su conocimiento.

Además, agradecemos el apoyo incondicional del Lic. Ariel Guerrero, profesor de la Universidad quien nos asesoró durante todo el proyecto. Por otra parte, agradecemos de igual manera el apoyo moral de los profesores y alumnos de nuestra facultad, en especial de un grupo

de alumnos del tercer año de nuestra carrera, quienes incansablemente nos han ayudado en todo lo que refiere a la parte técnica de la construcción del auto autónomo al que le llamamos AGUARA'I. También tuvimos la ayuda de una alumna la carrera de Arquitectura y de dos ingenieros egresados de nuestra carrera a quienes agradecemos.

Adjuntamos a la presente nota, los detalles técnicos de la construcción del auto autónomo y con el cual participamos en la competencia mencionada, de lo cual podemos referir que ha sido un éxito, a pesar de que no pudimos traer los primeros lugares de la competencia, pudimos demostrar que nuestra Universidad tiene el mismo nivel que Universidades del extranjero y los alumnos la misma capacidad.

En la página del Facebook del evento (<https://www.facebook.com/robocarrace/>), se pueden observar fotografías y videos de la corrida, donde estuvimos en representación del Paraguay, por la Universidad Católica Nuestra Señora de la Asunción.

Le comentamos, que deseamos realizar una demostración del auto autónomo que hemos construido para todos los integrantes de esta comunidad educativa y para el efecto le solicitamos nos indique una fecha y se puedan cursar las invitaciones correspondientes para que la sociedad en general incluso pueda conocer sobre el trabajo realizado en esta Universidad.

Nos despedimos nuevamente agradeciendo y con este pensamiento: *¡Con el apoyo de todos los integrantes de esta casa de estudios y de la sociedad, pudimos!*

P.D. Se adjunta la lista de nombres de las personas e instituciones que ayudaron para el proyecto.

Micaela Jara	Erid Pacheco	Ariel Bogado	Jesús Franco
Mat.: 45.581	Mat.: 45.990	Mar.: 45.302	Mat.: 46.204

Lista de las personas e instituciones que ayudaron para que la participación en la competencia sea exitosa:

- Ing. Walter Benítez
- Ing. Yessica Bogado
- Ing. Gabriela Cáceres
- Alfredo Galeano
- Alejandro Báez
- Jorge Bareiro
- María Belén Álvarez
- Oscar Martínez
- Sebastian Reckzeigel
- Paloma Frutos
- Paulo Melgarejo
- Ana Rodríguez
- Centro de Estudiantes FACYT
- Parque Tecnológico Itaipú (PTI-PY)
- National Instruments
- Universidad Católica “Nuestra Señora de la Asunción”
- Centro de Investigación en Ciencias, Tecnología e Innovación Avanzada (CICTIA)