UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN
Resolución: RPC-SO-09-No.265-2021

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGÍSTER

Título del proyecto:
SIMULACIÓN DE SISTEMAS DE CONDUCCIÓN AUTÓNOMO PARA LA CIUDAD DE MACAS MEDIANTE DRIVING SCENARIO DESIGN APP DE MATLAB
Línea de Investigación:
Automatización
Campo amplio de conocimiento:
Electrónica y Simulación
Autor/a:
Ing. Juan Ennis Espinoza Gonzalez MBA.
Tutor/a:
Mag. René Ernesto Cortijo Leyva

Quito – Ecuador

2023

APROBACIÓN DEL TUTOR

Yo, **Rene Ernesto Cortijo Leyva** con C.I: **1719010108** en mi calidad de Tutor del proyecto de investigación titulado: **SIMULACIÓN DE SISTEMAS DE CONDUCCIÓN AUTÓNOMO PARA LA CIUDAD DE MACAS MEDIANTE DRIVING SCENARIO DESIGN APP DE MATLAB.**

Elaborado por: **Juan Ennis Espinoza Gonzalez**, de C.I: **1400799852**, estudiante de la Maestría: en **ELECTRÓNICA Y AUTOMATIZACIÓN**, resolución: ***RPC-SO-09-No.265-2021***, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., 15 de marzo de 2023

RENE ERNESTO CORTIJO LEYVA

Firmado digitalmente por RENE ERNESTO CORTIJO LEYVA
Fecha: 2023.03.15 19:05:33 -05'00'

_____

Firma

DECLARACIÓN DE AUTORIZACIÓN POR PARTE DEL ESTUDIANTE

Yo, **Juan Ennis Espinoza González** con C.I: **1400799852**, autor del proyecto de titulación denominado: **SIMULACIÓN DE SISTEMAS DE CONDUCCIÓN AUTÓNOMO PARA LA CIUDAD DE MACAS MEDIANTE DRIVING SCENARIO DESIGN APP DE MATLAB**. Previo a la obtención del título de Magister en **ELECTRÓNICA Y AUTOMATIZACIÓN**, resolución: ***RPC-SO-09-No.265-2021***.

1. Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar el respectivo trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Manifiesto mi voluntad de ceder a la Universidad Tecnológica Israel los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor@ del trabajo de titulación, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital como parte del acervo bibliográfico de la Universidad Tecnológica Israel.
3. Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de prosperidad intelectual vigentes.

Quito D.M., 16 de marzo de 2023

Firmado electrónicamente por:
JUAN ENNIS ESPINOZA GONZALEZ

Firma

Tabla de contenidos

Índice de tablas

Índice de Ilustraciones

Índice de Gráficos

Índice de Imágenes

INFORMACIÓN GENERAL

Contextualización del tema

Las máquinas que funcionan con vapor y carbón han existido desde la industria 1.0, alrededor de 1784, siendo los autos de gran importancia debido a que evolucionan el transporte común, con el transcurso de los años con la industrialización del automóvil en la industria 2.0, en los años 1900, en donde se incorporó la producción en masa del ensamblaje de los autos debido a la incorporación de las líneas eléctricas de energía. Para los años 2000, se incorpora la automatización, los computadores y la electrónica en estado sólido, en donde la industria nuevamente cambió a industria 3.0 y para esta época los automóviles tenían varias funciones que aportan a los conductores su bienestar, como por ejemplo elementos electrónicos en el cerebro del automóvil para el control general de la reacción del motor y demás. Para los años 2017 a la fecha, y con la aceleración e integración tecnológica del internet se dio apertura a la industria 4.0 en donde se produce el auge de IoT, sistemas ciberfísicos, la inteligencia artificial y demás, que facilitan y ayudan a mejorar los instrumentos de un automóvil (Ferrás & López-lópez, 2021).

Según (INEC, 2022) en su anuario de estadísticas de transporte, 2021 en Ecuador se registraron 2.361 millones de vehículos matriculados para el año 2020, para el año 2021 se registraron 2.536 millones de vehículos matriculados, representando un incremento del 7.4%, y sus provincias con mayor cantidad de vehículos matriculados es Guayas con 552.569 y Pichincha con 534.300. De acuerdo con el (INEC, 2023)en su visualizador digital de estadísticas de transporte (ESTRA), la provincia de Morona Santiago cuenta con 15.095 vehículos matriculados para el año 2021, superior al año 2020 en donde se matricularon 14.669 vehículos, un incremento significativo cada año.

De igual manera según (AEADE, 2022), existe un incremento en Ecuador en la venta de vehículos eléctricos en donde para el año 2022 se tuvo 395 autos vendidos, en comparación para el año 2021 que es de 348 autos vendidos, es un claro incremento de venta de autos eléctricos, además para enero del 2023 se vendieron 42 autos eléctricos en comparación de enero 2022 en donde se tuvo 5 autos vendidos, esta tendencia se produce para años anteriores. Además este boletín técnico informa que los tipos de vehículos eléctricos más vendidos son los tipos SUV seguido de los automóviles y camiones, este sector de nuevas tecnologías en el área de transporte tiene una participación del 7 % del total de ventas. Además según (León & Salinas, 2018) en su tesis titulada: Introducción y uso de vehículos eléctricos para el transporte público

y su impacto en la red eléctrica de la ciudad de Loja, manifiesta que ya se están implementando electrolineras de carga rápida y carga lenta en las principales ciudades del ecuador como es en Cuenca, Quito y Guayaquil, esta iniciativa la está impulsando importantes universidades en convenios con las empresas distribuidoras. Para promover el uso de esta nueva tecnología en movilidad, el gobierno ecuatoriano con la respectiva ley de régimen orgánica del servicio público de energía eléctrica realiza la implementación de incentivos y políticas al uso del servicio del respectivo almacenamiento de energía y carga de autos eléctricos, de la misma manera ya se encuentra las directrices en el reglamento a la ley de servicio de energía eléctrica el servicio de carga a vehículos eléctricos.

Según (INEC, 2023)en Ecuador los siniestros de tránsito son un problema de mortalidad muy común, en donde para el año 2021 se registró 21.352 siniestros de tránsito, para el año 2020 se registró 17.000 siniestros de tránsito en donde el porcentaje de vehículos involucrados son los automóviles son del 37.2% seguido de la motocicleta con un 22%, en donde la clase de siniestro que prima son los choques y pérdida de pista. Los siniestros se producen en su gran mayoría los cuatro últimos meses del año, teniendo un pico de accidentes en el mes de octubre del año respectivo.

En referencia al (INEC, 2023), en su visualizador digital de estadísticas de transporte (ESTRA), en las estadísticas de siniestros de tránsito, en la provincia de Morona Santiago se registraron 136 accidentes de tránsito para el año 2021, para el año 2020 se registraron 121 accidentes, las causas de los accidentes es principalmente debido a la imprudencia del conductor y a la imprudencia del peatón, además del porcentaje de vehículos involucrados en estos siniestros son los automóviles y motocicletas con un 22.2% seguido de las camionetas con un 21.7%. Para la ciudad de Macas se registraron 51 siniestros para el año 2021 y para el año 2020 se registraron 56 siniestros. Además se registran fallecidos para el año 2021 un total de 13 personas y para el año 2020 de 6 fallecidos, la clase de siniestro que se registra son atropellamientos y estrellamientos, considerando el porcentaje de vehículo involucrado es la motocicleta con el 26.1% seguido del automóvil un 23.2%.

Según (Alban, 2016), Macas es conocida como la esmeralda oriental, es la principal ciudad del cantón Morona y la capital de la Provincia de Morona Santiago, así como la capital más grande y poblada de la región Sur de la Amazonía Ecuatoriana, se localiza entre los extremos la cordillera de los Andes entre los ríos Jurumbaino y Upano, con un clima tropical lluvioso de 19 °C promedio a una altitud de 1030 msnm su población es de 18.984 habitantes con una densidad de 355,97 hab/km2 y su superficie es de 53,33 km2. En cuanto a transporte en la ciudad existen distintas áreas que a continuación se detallan: 1) Es el transporte Aéreo con su

aeropuerto Edmundo Carvajal, en donde se puede viajar a las distintas ciudades del país, 2) Es el transporte público en donde se tiene el transporte intracantonal, intraparroquial e interprovinciales, y finalmente 3) El transporte privado, estos vehículos se trasladan avenidas de importancia para la ciudad, que se mencionan: Av. Soasti, Av. 9 de Octubre, Av, 24 de Mayo, Av. Jaime Roldós, Av. Amazonas, Av. 13 de abril, Av. Domingo Comín y Av. Simón Bolívar.

Problema de investigación

Los accidentes de tránsito se incrementan a medida que pasan los años, y paralelamente aumentan las muertes de personas en vehículos, debido a obstáculos estáticos y dinámicos, según (Stadler et al., 2022) en su artículo: Un enfoque de Evaluación de la credibilidad para pruebas virtuales basadas en escenarios de funciones de conducción automatizada, manifiesta que los obstáculos dinámicos son los más significativos para accidentes provocando siniestros fuertes y desfavorables para el conductor. Además, en (Navin Kumar et al., 2021) en su artículo sistema inteligente de caracterización de impactos de vehículos y notificaciones de accidentes, manifiesta que los accidentes en vehículos son producidos principalmente a problemas o inadecuaciones de las vías y carreteras por las ciudades y países.

Según datos expuestos (ECU 911, 2023) en su plataforma Virtual de power BI, en la ciudad de Macas se reportaron alrededor de 21.552 emergencias para el año 2021, para el año 2022 se produjeron 21.000 emergencias, en dicho año en Tránsito y Movilidad se reportaron el 7.7% del total. Además el (ECU 911, 2023) en Macas se reportaron 102 accidentes de tránsito para el año 2021, para el año 2022 se produjeron 103 accidentes de tránsito, y la zona centro en donde se producen más accidentes de tránsito, con 14 accidentes de tránsito es la zona en la Avenida 29 de Mayo y Gavino Rivadeneira, provocándose accidentes sin heridos. De acuerdo a los datos antes citados se puede observar que existe un problema en movilidad en la ciudad, es por eso que se centrara principal atención a las Av. 29 de Mayo y Av. Gavino Rivadeneira.

Según el (AEADE, 2022), en la provincia de Morona Santiago no se tiene datos de ventas de vehículos eléctricos, vehículos híbridos y vehículos autónomos (vehículos con radares, sensores y lidar), es decir no se comercializan en la región amazónica, por falta de conocimiento de las nuevas tecnologías en el transporte particular, comercial y público. Por lo expuesto anteriormente en dicha región, se genera un problema de no dar uso a nuevas tecnologías en la ciudad de Macas. Es por eso por lo que surge la necesidad de realizar un escenario para resolver el desconocimiento de nuevas tecnologías y de accidentes de tránsito y movilidad.

Objetivo general

Desarrollar la simulación de un escenario para la detección de objetos de un sistema de conducción autónoma de la ciudad de Macas mediante Driving Scenario Designer APP de Matlab.

Objetivos específicos

- Contextualizar el fundamento teórico sobre los sistemas de conducción automática, para el escenario de la ciudad de Macas.
- Aplicar el Driving Scenario Designer App de Matlab, para la intersección entre Av. 29 de mayo y Gavino Rivadeneira de la ciudad de Macas.
- Analizar los resultados de simulación del escenario para la detección de objetos en la ciudad de macas.

Vinculación con la sociedad y beneficiarios directos:

Ya que se desarrollarán simulaciones de escenarios para la detección de objetos de una conducción autónoma, este tiene la implicación directa con la sociedad, debido a que se plantea la aplicación a la ciudad de Macas para la Av. 29 de mayo y Av. Gavino Rivadeneira en donde se interactúa con actores; como personas, autos y objetos. De esta manera se probará el comportamiento para la detección de objetos ante escenarios al interactuar con humanos, y poder observar si los sensores funcionan adecuadamente, esto para poder evitar problemas de movilidad.

De acuerdo con (León & Salinas, 2018) el uso de vehículos autónomos, usan alimentación eléctrica para el funcionamiento de sus sistemas autónomos, debido a que todo el automóvil tiene que funcionar con un mismo sistema. Los vehículos eléctricos y autónomos producen emisiones de carbono mucho más bajos que los vehículos con combustibles fósiles, según Transport & Environment afirma que los vehículos de gasolina y diésel emiten tres veces más CO2 que un vehículo eléctrico. Por tanto, la implementación de la simulación de un escenario para la detección de objetos para la conducción autónoma tiene un gran impacto medioambiental debido a la tendencia hacia lo sostenible.

En (León & Salinas, 2018) una investigación de la universidad de Cambridge Michael He y Nicholas Hydmar, expresan que los vehículos autónomos que trabajan de forma colectiva reducirán el tráfico en al menos un 35%, es decir mejora la movilidad vehicular, debido a que se plantea una simulación para detectar objetos para una conducción autónoma, este tendrá un impacto en la movilidad y tránsito vehicular respectivamente.

Este proyecto tiene varios beneficiarios, el primero es el conductor de los vehículos, mejorando su confort y estado de ánimo para su respectivo trabajo y accionar del día. Además esta simulación beneficiará a el GAD Municipal y Provincial de Morona debido a que se le obtendrá resultados de un escenario real de la detección de objetos y su respectiva implementación de los mismo en autos eléctricos para la conducción autónoma. Finalmente los beneficiarios indirectos son los concesionarios y locales de venta de vehículos debido incentivaran el buen uso de estas tecnologías.

Este proyecto tendrá una implicación económica, es decir que tendrá un aporte económico indirecto debido a los resultados expuestos de la evaluación de los vehículos autónomos, se tendrá resultados en donde los entes reguladores de las vías de tránsito tendrán una variación de su costo de implementación. Además la población tendrá una variación significativa en cuanto a costos de mantenimiento, movilidad y siniestros de tránsito, obteniéndose beneficios para los usuarios de los vehículos autónomos.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1.1. Contextualización general del estado del arte

La conducción autónoma es un asunto de gran interés en la presente realidad, y se ha vuelto muy importante para muchas empresas del sector automotriz. La simulación de sistemas de conducción autónoma es un área en constante evolución, que busca desarrollar y probar algoritmos y tecnologías que permitan a los vehículos moverse de manera autónoma y segura en diferentes entornos y situaciones. El Automated Driving Toolbox de Matlab es una herramienta de simulación avanzada que permite a los ingenieros y diseñadores de sistemas de conducción autónoma desarrollar, probar y validar algoritmos y sistemas de control. Esta herramienta ofrece una amplia variedad de algoritmos, modelos y bloques permitiendo simular diferentes escenarios de conducción autónoma en entornos virtuales. Además, se están combinando nuevas tecnologías como el aprendizaje automático y la inteligencia artificial, para mejorar la toma de decisiones y la adaptabilidad de los sistemas de conducción autónoma.

En su trabajo (Estaña, 2017) titulado, SAEJ3016 Atand Deployment for automotive Autopilot and Automation in Road Traffic, mencionan que el nivel del sistema de conducción automatizada es: Nivel 0; Sin automatización en la conducción, lo que significa que todas las acciones las realiza solo el conductor, Nivel 1; Asistencia al conductor, en donde el automotor tiene implementado un sistema de ayuda a la conducción, Nivel 2; Automatización parcial, el automóvil tiene incorporado el respectivo control de movimiento tanto longitudinal como lateral para detectar y responder a objetos, Nivel 3; Automatización condicional, además de tener un control de movimiento contará con un sistemas de frenado para evitar colisionar con otro vehículo, pero sin desvirtuar la interacción humana, Nivel 4; Automatización Elevada, se podrá interactuar y colocar instrucciones u órdenes sin necesidad de manipular el volante ni pedales y demás, Nivel 5; Automatización Completa; además de realizar instrucciones de manera autónoma, cuenta un sistema de fallos automatizado, en donde interactúa y aprende ante el ambiente a su alrededor.

En la actualidad existen diferentes empresas que aplican el nivel 2 de automatización en los sistemas vehiculares, mientras que el nivel 3 se encuentra en desarrollo e implementando de manera paulatina en la automatización vehicular. Es decir, se aplica la automatización parcial y condicionada, enfocándose a la detección de objetos estáticos y dinámicos de manera longitudinal como lateral (Miranda, 2021).

De acuerdo con (Zambrano, 2022), en su tesis de posgrado titulada: Sistema de monitoreo autónomo inteligente vehicular combinando tecnología IoT y Redes Neuronales, realiza un algoritmo mediante redes neuronales para ser implementada en un vehículo y puede detectar

si se sale de su área de recorrido normal, enviando una alarma al dueño del automotor. En (Miura & Azumi, 2020), en su artículo titulado: Marco de escenario de conducción de conversión para probar sistemas de conducción autónoma, en donde presenta una conversión de escenario desde Matlab/Simulink hacia el simulador LGSVL, que es un vehículo autónomo simulador operado con Autoware que es un sistema de conducción autónomo de código abierto. En (Herrero, 2020) en su trabajo de Magister, de la Universidad Politécnica de Madrid denominado: Desarrollo e implementación de algoritmo de conducción autónoma en simuladores, presentando un manual de uso del software con CARLA y un análisis de ejemplos de uso del Automated Driving Toolbox de Matlab, además se presentó tres algoritmos, para la simulación de la trayectoria del vehículo autónomo, para la planificación de la ruta óptima dado un mapa respectivo y el tema de un filtro bayesiano que permita el manejar eventos no deterministas, para ejecutar movimientos para su trayectoria autónoma.

En referencia a (Giurgica & Florescu, 2020), en su artículo llamado: Un caso de estudio para modelar sistemas de conducción autónoma, en donde modelan un prototipo de vehículo autónomo nivel 2, utilizando para el modelado y simulación el automated driving toolbox, encontraron que es de gran utilidad el toolbox de matlab debido a que crearon los escenarios a detalle y se simularon el comportamiento del auto con sensores respectivos, además implementan el algoritmo para un robot en donde probaron el código. Con (Miranda, 2021), en su trabajo de titulación que se llama Fusión sensorial cámara y LIDAR para el seguimiento de múltiples objetivos. Aplicación a vehículos de conducción autónoma, en donde indican el estado de arte en referencia a la conducción autónoma y la fusión de los sensores en escenarios creados por el automated driving toolbox de Matlab, para que sea punto de partida para proyectos a futuros, encontrándose como conclusión que la mejor y eficaz alternativa es la fusión de sensores, obteniéndose mejores resultados en cuanto a detección y visión artificial. Para (Niu, 2018), en su documento de posgrado, llamado: Detección y seguimiento de objetos para la conducción autónoma mediante Automated Driving Toolbox de Matlab, muestra el procesamiento de imágenes, con el seguimiento de líneas para las vías, mediante el Automated Driving Toolbox de Matlab, este algoritmo de seguimiento y detección de líneas de las vías, de vehículos y peatones muestra grandes resultados. Finalmente (Escobar et al., 2021), en su artículo científico de detección de objetos en un entorno de simulación virtual con automated driving toolbox, menciona las simulaciones virtuales de varios escenarios relacionados con la conducción autónoma, los sensores detectan objetos en movimiento y estáticos obteniéndose el mejor desempeño con dos sensores laterales y uno frontal.

1.2. Proceso investigativo metodológico

Para empezar con la definición del tipo de investigación en este documento se va a definir qué es investigación, De acuerdo con (Sampieri et al., 2010) en su libro Metodología de la investigación en su capítulo 1 expresa que Investigación es un grupo de pasos sistemáticos, críticos y empíricos que se pueden ajustar e implementar a un proyecto o estudio. De acuerdo con el autor antes el enfoque de la investigación cuantitativa es secuencial y probatorio con un orden adecuado de pasos a seguir, cada etapa debe continuar sin antes terminar anterior y no se puede evitar pasos, es netamente estricto, recolectando datos para llevar a su procesamiento estadístico respectivo. Por lo tanto el enfoque de investigación es de tipo cuantitativa, debido a que se generará resultados numéricos de la simulación respectiva en donde se realizará un análisis de este descriptivo mostrando la mejor opción de simulación respectiva. El tipo de investigación es de tipo no experimental, transversal debido a que no se va a probar en la vida real un auto autónomo con el escenario respectivo y transversal debido a que se realizará el estudio en este momento del tiempo.

Según (Sampieri et al., 2010) en su libro de Metodología de Investigación Quinta Edición, explica de qué se trata las investigaciones de alcance descriptivos, mencionando la investigación descriptiva es especificar una persona, grupo, objeto o cualquier otro fenómeno, es decir únicamente se centra es recopilas información de manera independiente o conjunta. Debido al antecedente anterior, la investigación tiene un alcance descriptivo, debido a que se describirá el porqué del escenario virtual con sensores para una conducción autónoma, cuáles son los objetos detallados y las características de los sensores a detectar, obteniendo información del funcionamiento de la detección de objetos, posteriormente se realizará un análisis de esa información.

En los procesos de una investigación es indispensable el método científico en donde (Ramirez & Zwerg, 2012) al método científico como conjunto de hipótesis de crucial importancia en el proceso de investigación; debido a normas, reglas para resolver investigaciones y preguntas de investigación; han sido institucionalizadas por una comunidad académica reconocida. Es decir que la ciencia clasifica en algunos métodos básicos que a continuación se describen: a) Método Inductivo, b) Método Deductivo, c) Método Analítico, e) Método histórico y Método Experimental. En esta investigación tiene como método el deductivo debido a que parte de conclusiones generales a conclusiones particulares, utilizando el análisis de leyes y principios comprobados de los sistemas de automatización y el software MATLAB, para probar un hecho en particular en este caso para probar y detectar objetos respectivamente.

Finalmente, en referencia a (Concepto, 2023), en su información virtual, dentro del área de conocimiento, manifiesta que una técnica de investigación, son un grupo de herramientas, procesos y todo instrumento capaz de recopilar información para obtener conocimiento. Por lo tanto, la técnica de investigación que se utilizará en este documento es Driving Designer App de Matlab, en donde se genera el escenario real de la ciudad de Macas con sus respectivos enfoques en donde se aplican la teoría de los sistemas de asistencia avanzados al conductor, con los respectivos algoritmos adecuados para generar dichos escenarios y cargar los sensores respectivos para que puedan detectar los respectivos datos.

CAPÍTULO II: PROPUESTA

2.1. Fundamentos teóricos aplicados

2.1.1. Importancia de los sistemas SAAC o ADAS (por sus siglas en inglés)

De acuerdo a (Pico, 2019), en su trabajo de titulación llamado: Sistema Avanzado de Asistencia al conductor empleando Visión Artificial en Vehículos de transporte Público, menciona que un Sistema Avanzado de Asistencia a la Conducción (SAAC), es un procedimiento sistemático para poder aumentar la seguridad al manejar un automóvil y reducir la probabilidad de accidentes en las carreteras oh choques con otros autos.

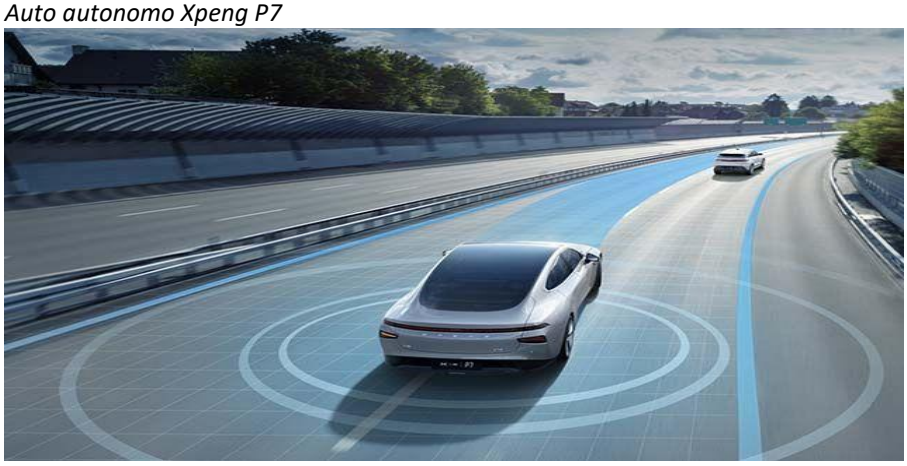
En donde para (MathWorks, 2023), los SAAC con sus elementos de hardware y software se proponen ha automatizar las actividades del conductor. Los ejemplos claros para los vehículos son: Control crucero adaptativo , la detección del punto ciego y finalmente el frenado automático de emergencia.

Las ventajas de un sistema SAAC para un vehículo, es que cuenta con una detección de línea de la calzada o vía, intentando producir una trayectoria adecuada sin salirse del carril correspondiente, además es posible incorporar y configurar un sistema anti-trayectorias para que pueda notificar la misma de manera oportuna mediante una vibración al asiento, aviso acústico y un aviso luminoso. Finalmente cuenta con un sistema completo de sensores capaces de detectar obstáculos, autos, peatones e incluso señales o avisos de tránsito.

Por lo tanto, en (MathWorks, 2023), los sistemas SAAC son importantes, debido a que reducen considerablemente los errores de personas y promueven para que las vías sean muy tranquilas, llegando a incentivar en gran medida la conducción autónoma debido a que estos sistemas alertan de situaciones peligrosas, por ejemplo el sistema autónomo de frenado de emergencia debido a un obstáculo en la vía evitando accidentes y colisiones. Según el estudio de Boston Consulting Group, los sistemas SAAC previene el 28% de siniestros en las vías y 9.900 fallecimientos en cada año en los Estados Unidos de Norteamérica.

Imagen *1*

Auto autonomo Xpeng P7

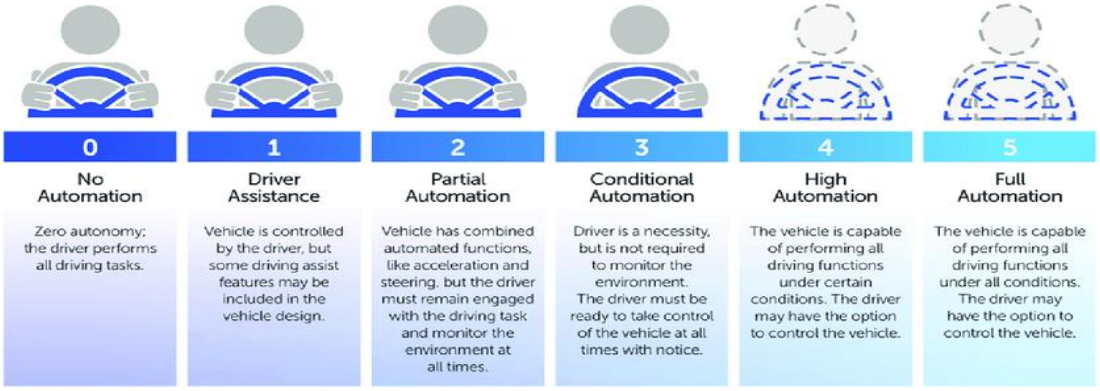


Fuente: Electric Cars Report

De acuerdo con la Sociedad de Ingeniero de Automatización (SAE), hay cinco niveles de conducción autónoma, actualmente existen automóviles que tienen implementado en sus sistemas características de nivel 0 a nivel 3, y las empresas a la vanguardia en automatización en autos eléctricos pretenden llegar al nivel 4 y 5.

Imagen *2*

Niveles de Automatización en la Conducción



Fuente: SAE J3016 Standard

De acuerdo a los niveles de automatización en el mercado internacional existen un sin número de empresas que están incursionando en la conducción autónoma, en américa del norte existen empresas importantes como es WAYMO, en donde desde el año 2009 comienza con su vehículo autónomo de Google, capaz de conducir 10 rutas de 160 kilómetros ininterrumpidos con los vehículos modelo Toyota piscis, posteriormente en 2015 salió al mercado un auto autónomo en donde se realizó la primera prueba por las calles de Austin, Texas, con una persona no vidente. Para el año 2018 inició el lanzamiento de Waymo One el primer servicio comercial de viajes autónomos a pedido en el mundo en Phoenix, posteriormente en el año 2020 sale al

mercado el Waymo Driver el primer transporte autónomo de bienes de forma seguro y eficiente a ciudades lejanas. Finalmente para el año 2021 se realiza una investigación de verificación de confianza para el uso de los vehículos autónomos además de ofrecerle una aplicación para poder solicitar un servicio de transporte autónomo. Los Vehículos de Waymo cuentan con 5 unidades de sensores de última generación tipo Lidar Perimetrales, con 29 cámaras y 4 unidades de radar, con esta tecnología la empresa asegura cubrir hasta unos 500 metros de longitud.

Imagen 3
Vehículos Autónomos

Fuente: *WAYMO/Home*

De la misma manera una de las empresas en innovación de Norteamérica es Tesla, en donde con su flota de vehículos eléctricos de distintos modelos futuristas desde el modelo y hasta el último modelo S, en donde dicho modelo ya incorpora autonomía en su conducción, con su elección en el menú del auto de Autopiloto, en donde su conducción es de nivel 3, necesitando una supervisión activa del conducto que no permite que el vehículo sea autónomo totalmente. Este autopiloto permite que el vehículo esquive, incremente su velocidad y aplique los frenos de forma autónoma estando en la vía. Estos vehículos cuentan con seis sensores de cámara, un sensor ultrasónico y un sensor de radar, cubriendo una distancia total de 250 m, en donde se aplican técnicas de procesamiento de visión, construidas sobre una red neuronal profunda deconstruyendo el entorno real con muy buenos niveles de viabilidad.

Imagen 4

Vehículo Modelo S

Fuente: TESLA

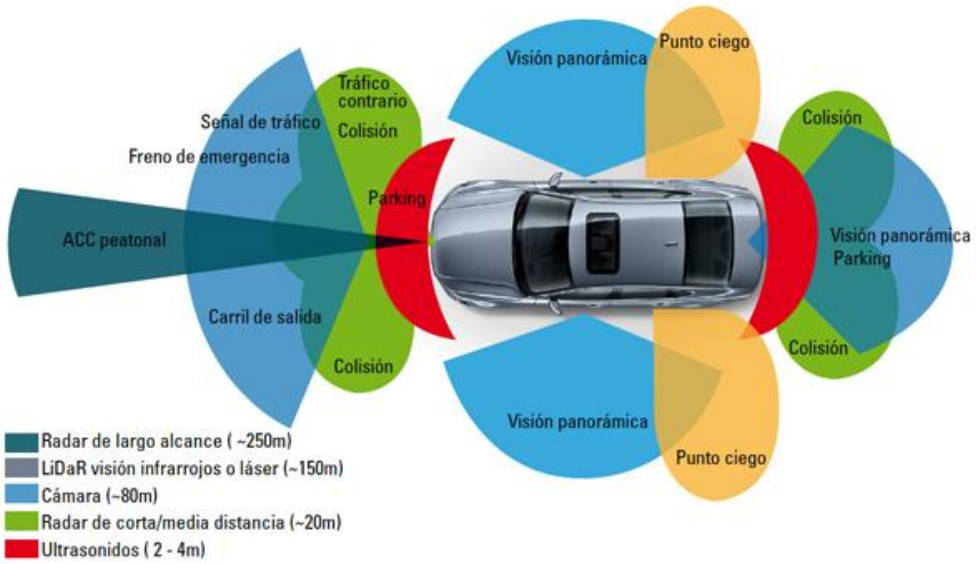
2.1.2. Diseño de funcionalidades de SAAC o ADAS (por sus siglas en inglés)

De acuerdo con (Laencina, 2016), en su trabajo de posgrado titulado Diseño de un sistema avanzado de asistencia al conductor basado en Visión Artificial para la predicción del tiempo de Colisión("TTC"), menciona que los sistemas SAAC están compuestos mediante sensores físicos como: cámara, radar, LiDAR, sensor ultrasónico, dotando al vehículo con la capacidad de percibir y monitorear en cualquier dirección estén lejos o cerca. Además es necesario la implementación de algoritmos de fusión de sensores y procesamiento de los datos respectivos que actúen en tiempo real a través de notificaciones y avisos para poder realizar una acción de emergencia. Como se puede observar en la imagen 5 un arreglo de sensores en un sistema SAAC para poder obtener el funcionamiento del vehículo.

Los desafíos son varios para el desarrollo, implementación y diseño para poner en marcha el funcionamiento de un sistema SAAC, es decir se debe recopilar información del entorno y ambiente que le rodea, además del procesamiento rápido de estos datos, para poder, predecir acciones o eventos en donde el vehículo se encuentra implicado, reaccionando de manera inmediata en tiempo real. Este funcionamiento es importante que sea fornido, confiable y presente mínimas acciones de equivocación.

Imagen 5

Esquema de Ubicación de los sensores y su alcance



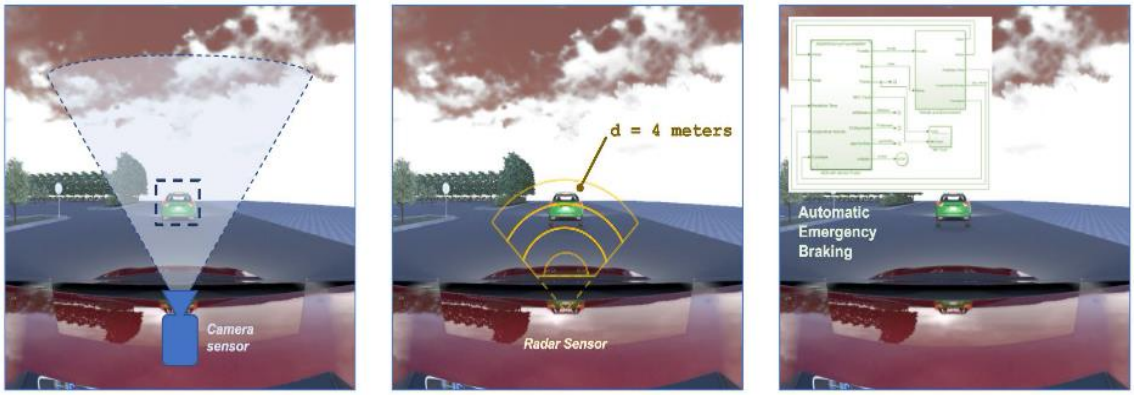
Fuente: Revistaautocrash

Para un mejor entendimiento en (MathWorks, 2023) se usa un ejemplo llamado control de crucero adaptativo para expresar cómo está diseñada la funcionalidad SAAC, el automóvil reducirá la velocidad cuando se acerque al vehículo de enfrente y acelerará a velocidad de crucero cuando el vehículo se aleje una distancia considerable. El primer paso para diseñar un control de crucero adaptativo (CCA) es recopilar datos de los sensores instalados en el automóvil mediante cámaras y sensores de radar. La cámara detecta los objetos de la imagen (árboles, peatones, vehículos, etc.) y el radar calcula la distancia entre el vehículo y el objeto correspondiente. Después de recopilar los datos de los dispositivos SAAC, el siguiente paso es diseñar el algoritmo SAAC.

El respectivo control de crucero adaptativo se divide en tres pasos importantes en donde se muestran en la imagen 6, en donde el 1) Se debe realizar un algoritmo de percepción para la detectar si hay un vehículo situado delante, 2) Un algoritmo de radar para calcular la distancia entre los dos vehículos u objetos y 3) Un algoritmo de control para ajustar la velocidad del vehículo en función de la distancia medida.

Imagen 6

Pasos de un Sistema CCA



Fuente: MathWorks

Para (Pico, 2019), en su trabajo de titulación llamado: Sistema Avanzado de asistencia al conductor empleando visión artificial en vehículos de transporte público, en donde manifiesta la importancia de los sensores, indicando que se tiene para una funcionalidad de SAAC de cuatro tipos que son cámaras, radar, sensor ultrasónico, y LiDAR.

Cámaras

Los sensores tipo cámaras son aquellos en los que se utilizan generalmente para la detección dentro de los sistemas SAAC. Las cámaras se ubican de manera lateral que sirven para detectar en los puntos ciegos. Las cámaras además se instalan en la parte frontal para poder detectar vehículos, carriles, señales, peatones y ciclistas. De la misma manera ocurre con la ubicación de cámaras en la parte de atrás para poder dar la marcha en reversa servirá para detección de varios objetos, peatones y vehículos. Los algoritmos de detección para sistemas SAAC se deben a la creación de algoritmos convencionales de visión artificial y Deep Learning. Las cámaras muestran ciertas condiciones de virtud que a continuación se mencionan:

a) Entregan información fiable para la detección de objetos
b) En el mercado comercial las cámaras son de costos reducidos permitiendo la prueba de muchos tipos de cámaras.
c) Existe un sinnúmero de variedades de cámaras entre las más destacables tales como: Ojo de pez, estenopeicas y Ojo de pez.
d) Las cámaras son del tipo sensor con mayor cantidad de investigaciones debido a ser las más antiguas.

La desventaja más grande de las cámaras es que son menos adecuadas para detectar el alcance de un objeto comparado con otros sensores.

Imagen 7
Cámara de sensor lineal

Fuente: Miranda, 2021

Radar

Los sensores de radar son aquellos que emiten ondas de frecuencia grande y registran el momento que la onda regresa debido a un rebote del entorno u objeto. Estos datos o registros permiten que se pueda calcular la distancia de un objeto de manera adecuada, además en los sistemas SAAC estos sensores se ubican en la posición delantera y trasera del vehículo. Este sensor es fiable debido a que funciona en entornos de situación climáticas críticas es por eso por lo que en los SAAC se utilizan principalmente para el frenado de emergencia automática y para el control de crucero adaptativo. Los sensores tipo radas si bien son adecuados para la detección de distancia de objetos, para la clasificación de objetos detectados resultan inútiles, por esa razón se suele utilizar una combinación de radares con cámaras.

Imagen 8
Sensor tipo Radar para aparcamiento

*Fuente: Miranda, 202*1

Sensor ultrasónico

Los sensores tipo ultrasónicos son aquellos aparatos electrónicos que emiten ondas sonoras fuera del espectro auditivo en donde dichas ondas son impactadas con los objetos del medio y recopiladas por un receptor. Con este tipo de sensor es posible estimar de forma precisa la distancia de un objeto debido a los tiempos entre onda emitida y recibida. Estos sensores son adecuados a bajas distancias debido a que alcanzan su máxima eficacia, además funcionan de manera adecuada con obstáculos cercanos. Estos tipos de sensores son de bajo costo, en comparación con los antes mencionados, actualmente son utilizados para aparcamientos automáticos y detección de obstáculos en ruta debido a su precisión, además de ser robustos y seguros. La desventaja de este tipo de sensor es que es sensible a cambios de temperatura en el ambiente por lo que requiere un tratamiento adicional para solventar este tipo de problema.

Imagen 9
Sensores ultrasónicos de automóvil

Fuente: Miranda, 2021

LiDAR

Los sensores tipo LiDAR también conocidos por distancia de luz y detección, emite un láser al ambiente o entorno y reconoce la señal cuando es regresada. Estos a su vez crean o construyen una nube de puntos en 3D mostrando así el entorno circundante del LiDAR. Los elementos e información de un LiDAR se pueden usar para el cálculo respectivo de distancia hacia el objeto en la nube de puntos en 3D.

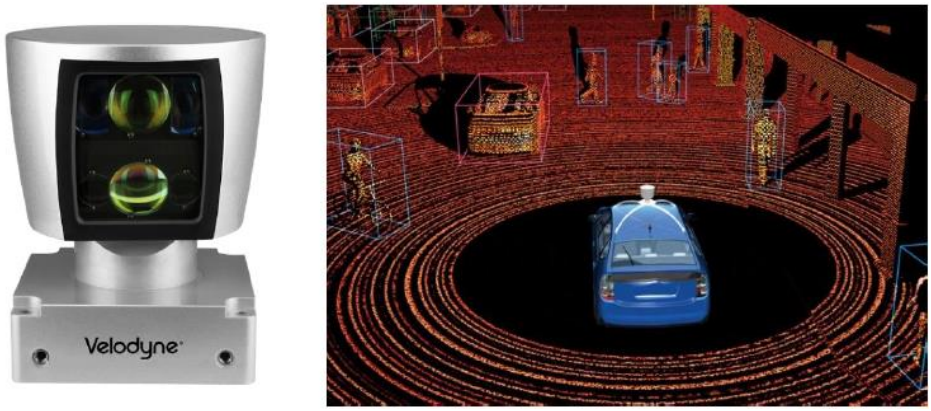
En las aplicaciones de SAAC se tiene los siguientes tipos:

a) LiDAR electromecánico(giratorio): Está instalado en la parte superior del vehículo en donde la misma gira para poder crear un mapa en 3D del entorno o ambiente exterior.
b) LiDAR de estado sólido: el LiDAR de estado sólido, no gira como el electromecánico y es más rápido, preciso y económico.

Los sensores LiDAR son utilizados para la detección de distancia además de la detección de objetos, pero necesita de una gran capacidad de procesamiento de datos y cálculos.

Imagen 10

Sensores lidar y nube de puntos generada con dicho sensor



Fuente: Miranda, 2021

2.1.3. Automated Driving Toolbox de Matlab

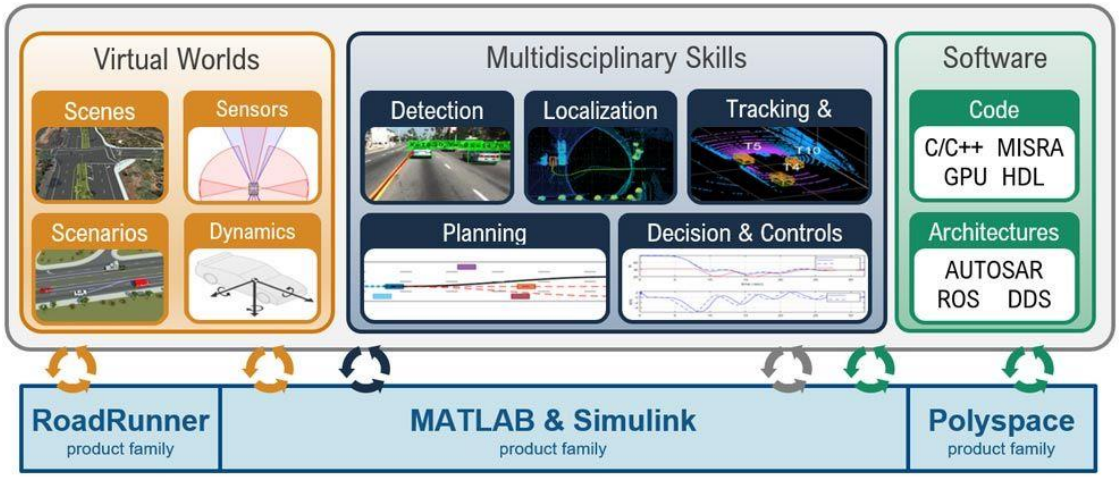
Según (MathWorks, 2023), el Automated Driving Toolbox, es un paquete de herramientas pertenecientes al entorno de MATLAB, ofrece algoritmos para diseñar, simular y probar sistemas SAAC y de conducción autónoma. En esta herramienta se puede diseñar y probar sistemas de visión artificial y sistemas LiDAR, así como también sistemas de combinación de sensores y preparación de rutas para vehículos. Dentro de este toolbox permite importar datos mediante el HERE HD Live Map, y red de carreteras de OpenDRIVE.

Dentro de esta herramienta existen aplicaciones en MATLAB como Ground Truth Laber, donde permite etiquetar objetos o datos en múltiples vídeos, proceso continuo de imágenes y nube de puntos de un sensor LiDAR. Además se puede generar y simular escenario de conducción de pruebas Hardware-in-the-Loop (HIL), en donde son simulaciones en tiempo real que permiten probar código sin necesidad de un hardware de sistema, permitiendo que las simulaciones se prueban en condiciones anormales y de fallo sin dañar el hardware respectivo. Básicamente se centra en las pruebas de percepción, de fusión de sensores y planificación de rutas junto con el respectivo control. Finalmente este toolbox permite realizar simulaciones de sensores de cámara, sensores de radar y Lidar encontrando su respectiva detección de objeto y distancia en un entorno fotorrealista 3D y 2.5D.

El presente toolbox además se encuentra asociado una aplicación fundamental (APP Driving Escenario Design) en donde sirve para la creación de entornos de vías con autos, peatones y objetos en donde es posible crear una trayectoria específica junto con sensores para poder simular y obtener un algoritmo respectivo. Además en este medio existen opciones para trabajar

e importar con información de HERE HD Live Map y redes de Carreteras llamado OneDrive, facilitando y adecuando a la realidad cada simulación.

Imagen 11
Sistemas de conducción autónoma



Fuente: MathWorks, 2023

En (MathWorks, 2023), se presenta la explicación de un sistema completo de simulación que se muestra en la Imagen 11, en dónde está compuesta de tres etapas: 1) Se trata del entorno virtual, en donde se compone de escenarios, escenas, sensores y autos dinámicos, en esta etapa existe el producto RoadRunner para la creación específica de escenarios virtuales, además se tiene la aplicación Driving Scenario Design, y por último se puede crear una escena con la herramienta Unreal Engine, la siguiente etapa 2) Comprende la Planificación, Control, Localización, seguimiento, detección y la toma de decisiones, y finalmente la tercera etapa 3) Está compuesto del Software en donde se tiene una familia de productos poli-espacio combinando un sinnúmero de código y arquitectura.

2.1.4. Aplicaciones del Automated Driving Toolbox

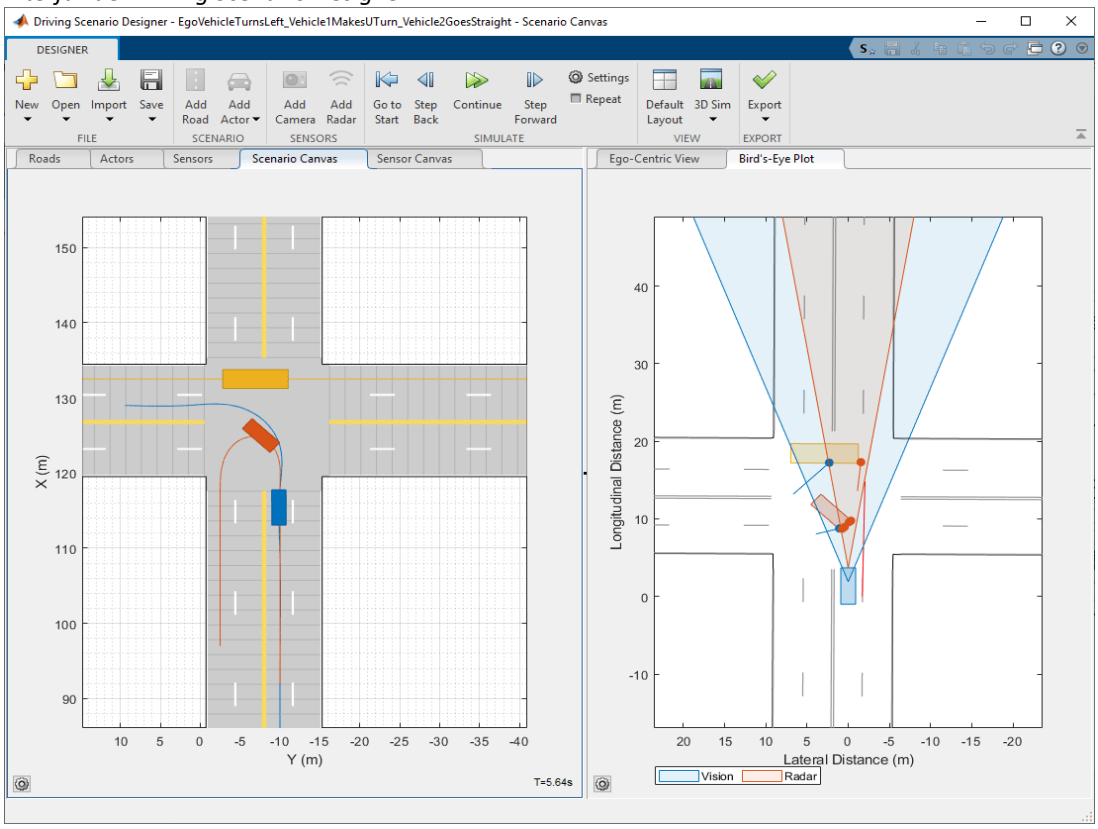
Driving Scenario Designer App de Matlab

Es una aplicación en donde se puede realizar escenarios para una conducción autónoma sintética, además de configurar e incorporar sensores, para la detección respectiva de objetos, todo esto permite generar datos sintéticos respectivos para su análisis y respectivas incorporaciones para control, seguimiento, localización, planificación y toma de decisiones de un auto de conducción autónoma.

Driving Scenario Designer (DSD), es un planificador de escenarios de conducción e incorporado por MATLAB, en donde permite la configuración de autos autónomos, personas, bicicletas, barreras y otros objetos. Esta permite la creación de vías de un carril doble, y demás, en donde se puede incorporar sus líneas de división y demarcación de carril, pasos cebra y demás elementos importantes para la creación de una vía.

Imagen 12

Interfaz del Driving Scenario Designer



Fuente: MathWorks, 2023

Con esta aplicación es posible de realizar lo siguiente:

- Diseñar una vía o camino de acuerdo con cualquier modelo propuesto por el diseñador, además de incorporar actores, todo esto se puede realizar usando una interfaz de arrastrar y soltar.
- Configurar e incorporar sensores del tipo Radar, de Visión, LiDAR, INS y ultrasonidos, en donde se les puede montar en el vehículo tipo ego de Matlab. Estos sensores pueden generar detecciones de objetos, detección de límite de carril, de actores y generar datos para una nube de puntos de un tipo LiDAR.
- Es posible también la importación de información, en este caso se puede importar carreteras y carriles mediante ASAM OpenDRIVE, en donde se

importará dicha información georreferenciada con su respectiva altitud, además con todas las características de tránsito de las mismas.

- También se podrá exportar datos de carretera por medio de OpenStreetMap, HERE HD Live Map o con Zerin Japan Map Api 3.0, que son servicios web para escenario de conducción autónoma.
- Esta también permite exportar la red de carreteras en el escenario de conducción creado por el diseño todo esto en formato ASAM OpenDrive. Además se podrá exportar las respectivas trayectorias, los actores de un escenario de conducción en Archivo ASAM OpenSCENARIO.
- Importante también los datos generados por sensores, se podrá exportar las detecciones de sensores sintéticos a MATLAB.
- Genera código MATLAB del escenario y los sensores para poder luego modificarlo e importarlo hacia una nueva aplicación para simulación adicional.
- Finalmente genera un modelo en Simulink a partir del escenario creado junto con los sensores y actores, estos modelos se podrán utilizar para probar la fusión de sensores o algoritmos de control.

2.2. Descripción de la propuesta

Para realizar la simulación se debe considerar lo siguiente:

- Tener un computador con las siguientes características mínimas, 16 RAM de memoria, una tarjeta de video de 16GB y con espacio de disco duro de 1TB.
- Instalado el Software Matlab 2022b, deberá tener instalado el Automated Driving Toolbox y tener la App Driving Scenario Design.
- Georeferenciar el espacio o vía en donde se va a aplicar la simulación.

Para la diseñar la simulación y realizar la prueba respectiva se tendrá presente:

- Cargar los puntos georreferenciados o cargar el mapa mediante OpenStreetMap de la intersección Av. 29 de Mayo y Av. Gavino Rivadeneira. Esto se realiza gracias a las opciones de importar los datos.
- Crear los escenarios en donde intervienen autos, tráiler, peatones y ciclistas.
- Se deberá implementar la trayectoria respectiva a cada actor en este caso a nuestro auto.
- Implementará los sensores de acuerdo con configuraciones que se ofrezcan en el mercado.

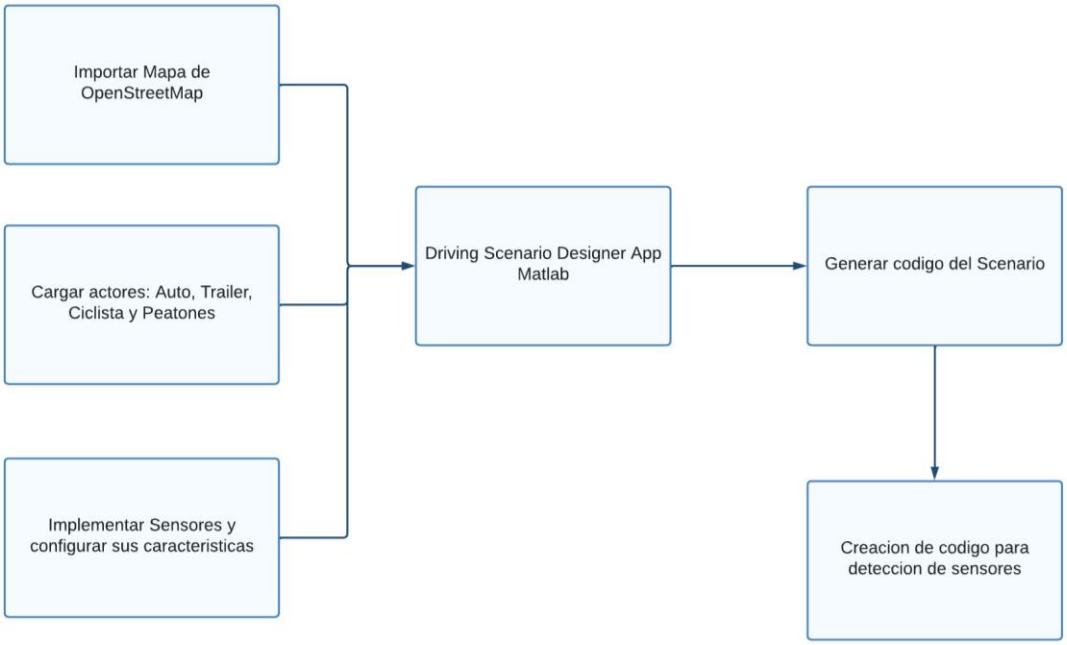
- Generar la prueba corriendo la simulación respectiva encontrando algún error o defecto en el mismo.
- Finalmente se exporta la programación de cada simulación
- Se analizan los resultados del algoritmo de simulación, obteniéndose datos y gráficos respectivos para obtener resultados respectivos.

a. Estructura general

Para un mejor entendimiento del procedimiento empleado se elaboró un diagrama de bloques que a continuación se muestra en la Ilustración 1. Este está compuesto por 6 bloques en donde el 1) Bloque de Importar el Mapa de OpenStreetMaps, es donde se importa las vías de interés y se carga al DSD, 2) Bloque el de crear los actores, se realiza la inserción de los peatones, autos, ciclistas y trailers en donde se ubicar de acuerdo a el escenario respectivo, 3) Bloque Implementar Sensores y configuración, es en donde se cargan los sensores o grupo de ellos para su respectiva detección, 4) Bloque Driving Scenario Designer App, es en donde se realiza la prueba de respectiva del escenario junto con los sensores en donde se corrobora errores y demás, 5) Bloque Generar código del Escenario, es aquel en donde se exporta el código del escenario respectivo y en donde se crear Script general, para su respectiva prueba, de los resultados y 6) Bloque Creación de código para detección de sensores, en este bloque se realiza codificación respectiva para la extracción de los resultados del número de objetos detectados y de las medidas obtenidas.

Ilustración 1

Diagrama de Bloques, de simulación de Escenario para Macas



Fuente: Intertigador

b. Explicación del Aporte

La simulación definirá de la siguiente estructura que se emplea a continuación:

Definición de escenarios

Para la creación de los escenarios respectivos se importó desde el OpenStreetMap, en donde se tiene georreferenciado las Avenidas Cargándose hacia mi aplicación, como se observa en la ilustración 2 respectiva. Una vez ubicada las avenidas se procede a seleccionar las para poder guardar en un archivo. osm.

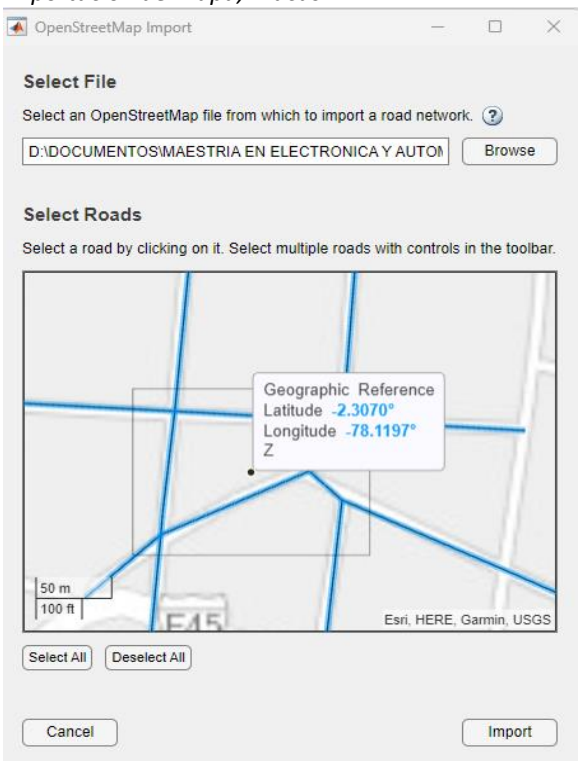
Ilustración 2
Av. 29 de Mayo y Av. Gavino Rivadeneira, Macas

Fuente: www.openstreetmap.org

Para cargar el mapa, se procedió a ingresar, en la pestaña de importar, posteriormente a cargar el archivo punto oms, en donde está predeterminado las avenidas de interés de la ciudad de macas respectivamente georreferenciadas con su altitud. En la Ilustración 3, se observa que está cargando el mapa de interés de la ciudad de macas.

Ilustración 3

Importación de Mapa, Macas



Fuente: Investigador

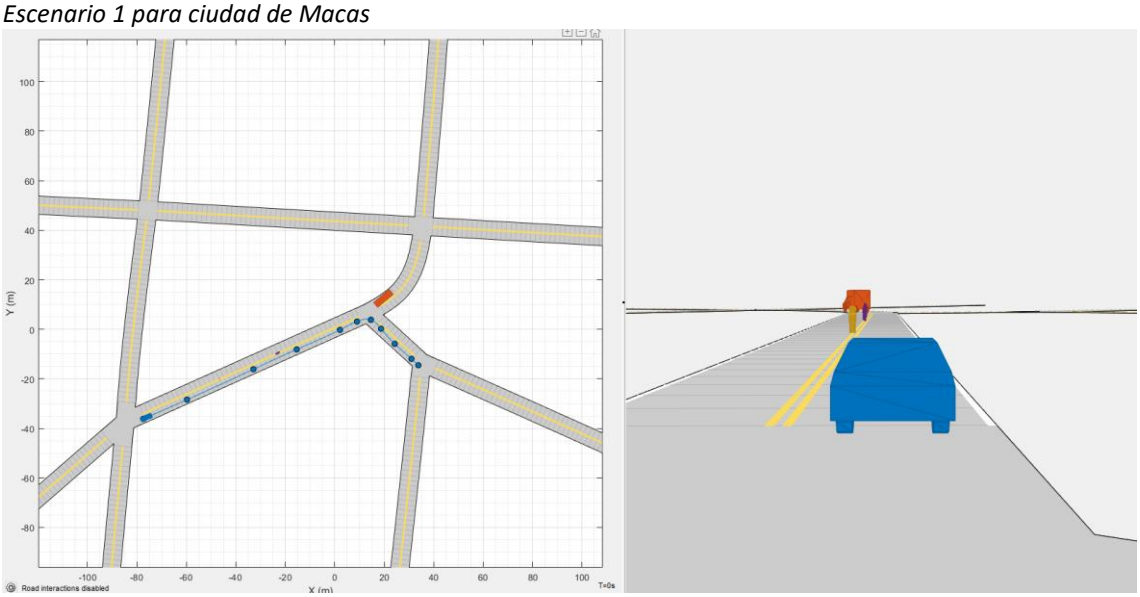
Para continuar con la simulación respectiva se plantea tres escenarios en donde se explican a continuación:

Escenario 1

Para el escenario 1 se plantea, los siguientes actores, un auto tipo ego, un peatón, un ciclista y un trailer. Se le brinda de trayectoria al auto ego, que se muestra en la ilustración 4 de color azul. Se observa que los puntos de color azul son por donde el auto se va a mover automáticamente que se encuentra dentro del carril respectivo, al encontrarse con la intersección el auto da un giro de inmediato, debido a que en la parte de adelante se encuentra un tráiler de manera estática. Además en el transcurso de la trayectoria del auto, se observa un peatón y ciclista en media vía de manera inmóvil.

Ilustración 4

Escenario 1 para ciudad de Macas



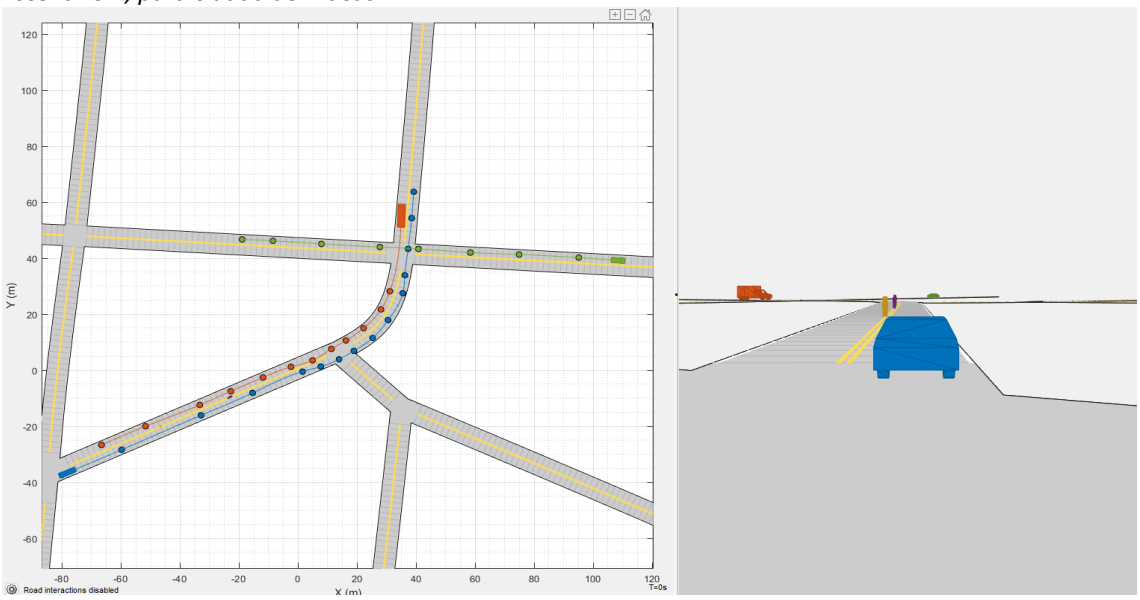
Fuente: Investigador

Escenario 2

Para el escenario 2, se incorporó, los siguientes actores, dos autos ego, un tráiler, un peatón y un ciclista. En este escenario se brindó de trayectoria y movimiento a tres actores los dos autos ego y el tráiler respectivamente y para el peatón y ciclista se mantuvieron estáticos en media vía. En la Ilustración 5 se puede observar el auto de color azul que es de nuestro análisis se mueve por la av.29 de mayo hasta llegar a la intersección de la Av. Tarqui, en esta avenida se observa un auto de color verde en donde está circulando de manera perpendicular la av. 29 de mayo. Además se observa que el tráiler o camión se mueve de igual manera desde la Soasti-Amazonas cruza la Av. Tarqui hasta el final de la Av. 29 de Mayo. En este escenario se observa que el Tráiler se sale de su respectivo carril en dirección del auto ego azul, este auto evita dicha colisión de manera que sigue con su trayectoria. De igual manera el auto azul evita una colisión con el auto verde frenando su velocidad hasta evitar al mismo.

Ilustración 5

Escenario 2, para ciudad de Macas



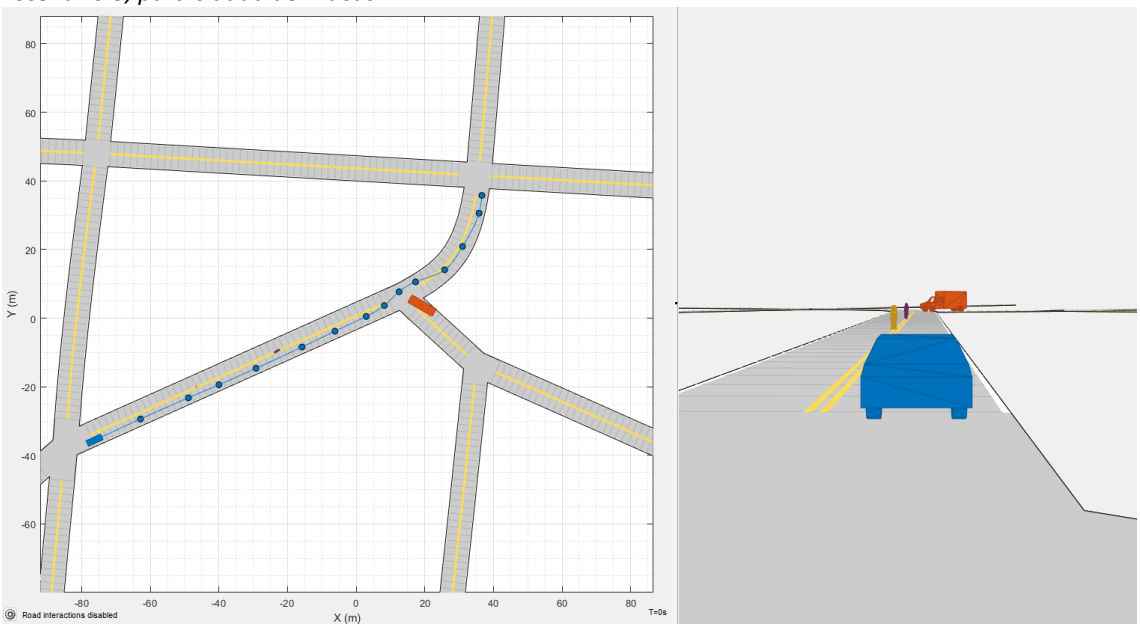
Fuente: Investigador

Escenario 3

Este último escenario, se configuró con los siguientes actores, un auto ego, un tráiler, un peatón y un ciclista. Este escenario cuenta únicamente con el movimiento y trayectoria del auto ego color azul, en donde se observa su trayectoria con los puntos respectivos en toda la Av. 29 de Mayo. El auto azul está por colocar en la intersección este por lo tanto esquiva al tráiler que se encuentra inmóvil. De Igual manera que en los escenarios anteriores están colocados el peatón y el ciclista en medio de los carriles, provocando una situación para que sean detectados respectivamente.

Ilustración 6

Escenario 3, para ciudad de Macas



Fuente: Investigador

Configuración de los sensores

Para completar con la configuración y diseño de los escenarios, se procede a incorporar los sensores en el respectivo auto ego de color azul de la siguiente manera:

Sensores para escenario 1

Para el Escenario 1, se propuso un automóvil WAYMO ONE, con su configuración base de los tres tipos de sensores, en donde están compuestos por 4 sensores Cámara, que están ubicados 2 adelante y dos en los laterales de la parte posterior del auto, además se propuso un sensor tipo radar en la parte frontal, con un alcance de 75m, y finalmente un sensor ultrasónico, con un alcance de 12m.

Tabla 1

Configuración de Sensores Escenario 1, Macas

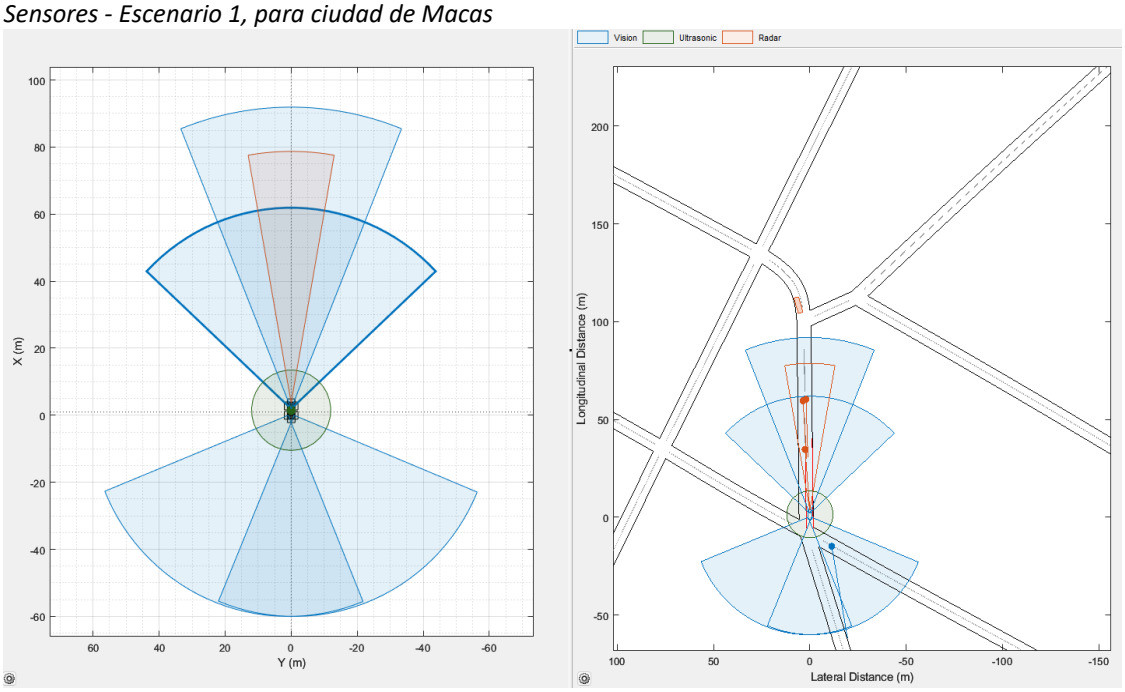
Sensores	X(m)	Y(m)	Altura(m)	Longitud Focal(m) o campo visual	Rango Máximo alcance (m)
Cámara Frontal 1	1.9	0	2	x=800 y=800	90
Cámara frontal 2	1.90	0	2	x=300 y=800	60
Cámara Posterior Derecha	0	-0.9	2	x=320 y=320	60
Cámara Posterior Izquierda	0	0.9	2	x=320 y=320	60
Radar Frontal	3.70	0	0.80	25 grados	75
Ultrasónico	1.50	0	0.80	360	12

Fuente: Investigador

En la Ilustración 7 se puede observar la configuración de los respectivos sensores para el escenario 1.

Ilustración 7

Sensores - Escenario 1, para ciudad de Macas



Fuente: Investigador

Sensores para Escenario 2

En el presente escenario, se propuso un automóvil eléctrico tipo TESLA Modelo S, en donde se tomó como base la configuración de los sensores, que están compuestos de 4 tipos de sensores, estos son LiDAR, Cámara, Radar y Ultrasónico. Se implementó 4 sensores tipo cámara, 2 sensores tipo radar, un sensor tipo ultrasónico y un solo sensor tipo LiDAR. Tanto para los sensores tipo cámara como para los sensores tipo radas se implementó un alcance de 25m, con una longitud focal de (320m, 320m) para los dos sensores laterales del auto y para los sensores frontal y posterior de (800m, 800m). Para el sensor Ultrasónico el alcance que tienes es de 8m, y su campo de visión AZimut es de 70 grados, para el sensor tipo LiDAR es de 20m, y su ángulo de visión es de 360 grados. En la tabla 2 se puede observar la configuración de los sensores para la Escena 2.

Tabla 2

Configuración de Sensores Escenario 2, Macas

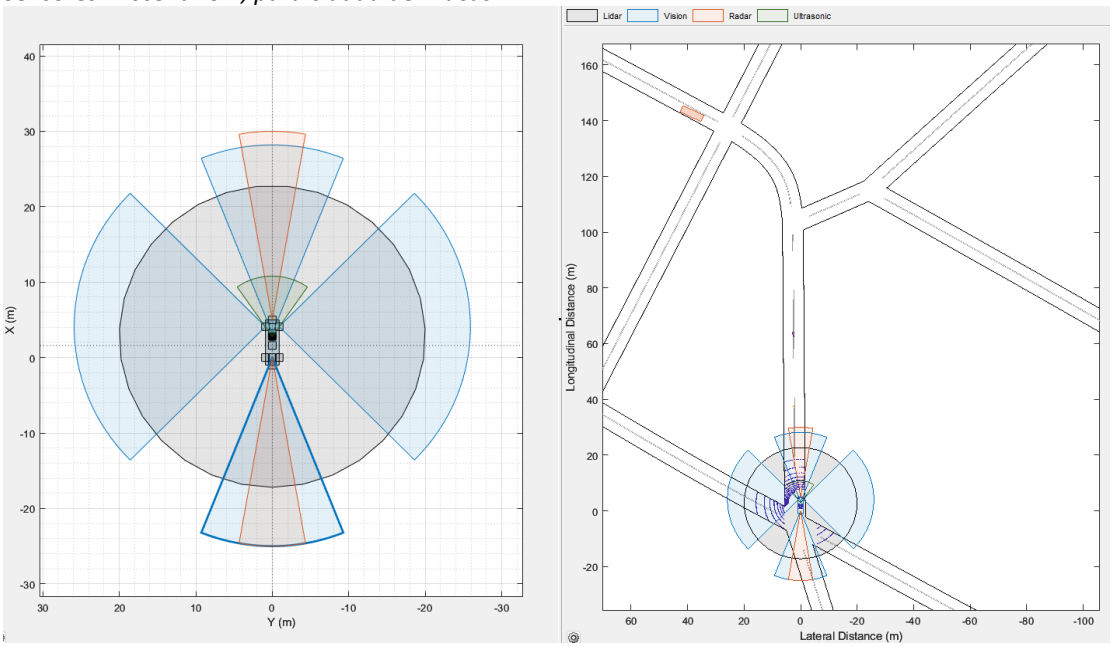
Sensores	X(m)	Y(m)	Altura(m)	Longitud Focal(m) o campo visual	Rango Máximo alcance (m)
Cámara Frontal	3.20	0	1.50	x=800 y=800	25
Cámara Posterior	0	0	1.50	x=800 y=800	25
Cámara Lateral Derecha	4.1	-0.9	1.50	x=320 y=320	25
Cámara Lateral Izquierda	4.1	0.9	1.50	x=320 y=320	25
Radar Frontal	5	0	0.50	30 grados	25
Radar Posterior	0	0	0.50	30 grados	25
Ultrasónico	2.8	0	0.50	70 grados	8
LiDAR	2.8	0	1.60	360	20

Fuente: Investigador

En la Ilustración 8 se puede observar la configuración de este tipo de auto con sus respectivos sensores, mostrando los cuatro tipos, respectivamente.

Ilustración 8

Sensores - Escenario 2, para ciudad de Macas



Fuente: Investigador

Sensores para Escenario 3

Para este último escenario se propuso el automóvil Lexus LS 500, en donde de la misma forma se ha tomado la configuración base de los sensores implementando tres tipos de sensores, de tipo cámara, de tipo radar y de tipo LiDAR. Se colocaron 4 sensores de tipo cámara, dos laterales y dos en la parte frontal y parte posterior del auto, los sensores de tipo radar se colocaron tres uno en la parte frontal, y dos en la parte lateral del auto. Su respectiva posición altura campo visual y rango de alcance viene dada en la tabla 3 que se muestra a continuación.

Tabla 3

Configuración de Sensores Escenario 3, Macas

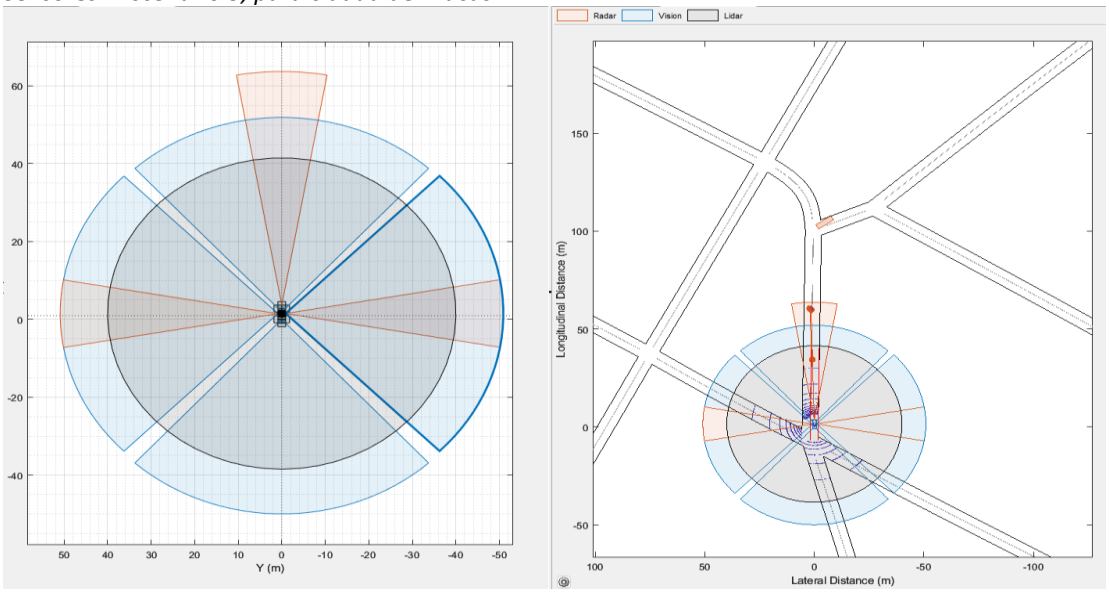
Sensores	X(m)	Y(m)	Altura(m)	Longitud Focal(m) o campo visual	Rango Máximo alcance (m)
Cámara Frontal	1.9	0	1.1	x=350 y=350	50
Cámara Posterior	0	0	1.1	x=350 y=8000	50
Cámara Lateral Derecha	1.52	-0.9	1.1	x=320 y=320	50
Cámara Lateral Izquierda	1.47	0.9	1.1	x=320 y=320	50
Radar Frontal	3.70	0	0.20	20 grados	60
Radar Lateral Derecho	1.51	-0.90	0.20	20	50
Radar Lateral Izquierdo	1.46	0.9	0.20	20	50
LiDAR	1.5	0	2	360	40

Fuente: Investigador

En la Ilustración 9 se puede observar la configuración de este tipo de auto con sus respectivos sensores, mostrando los tres tipos, respectivamente.

Ilustración 9

Sensores - Escenario 3, para ciudad de Macas



Fuente: Investigador

Simulación de cada escenario

A continuación se mostrará la simulación de cada uno de los escenarios respectivos, para su posterior análisis respectivo.

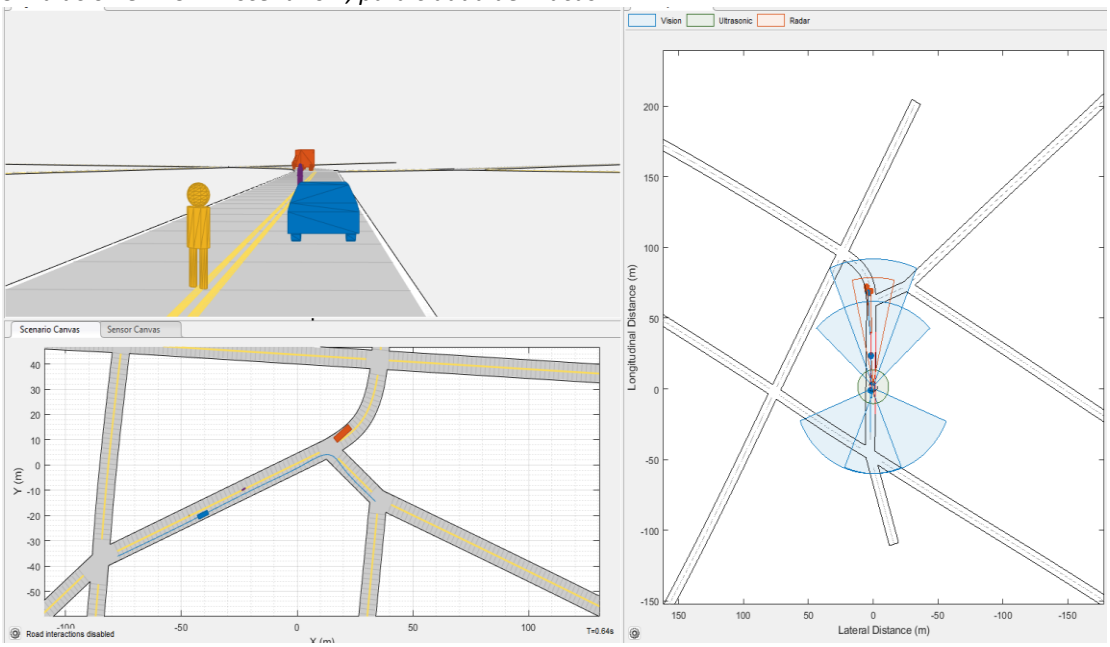
Simulación escenario 1

Para la simulación del escenario 1 se procede a realizar la siguiente simulación en donde se tiene los siguientes resultados con el Driving Scenario Designer (DSD) y con el Unreal Engine. Como se puede observar en la simulación en DSD, existen tres visiones respectivas la primera es la visión egocéntrica, en la que se muestra la simulación en forma de cubos de los actores

respectivos. La visión de ojo de pájaro es aquella que se muestra los sensores con su respectiva detección y finalmente se muestra la simulación del escenario en donde se encuentra la vía principal. Esta simulación se la puede encontrar en la ilustración 10 con su respectiva detección de objetos.

Ilustración 10

Simulación en DSD - Escenario 1, para ciudad de Macas

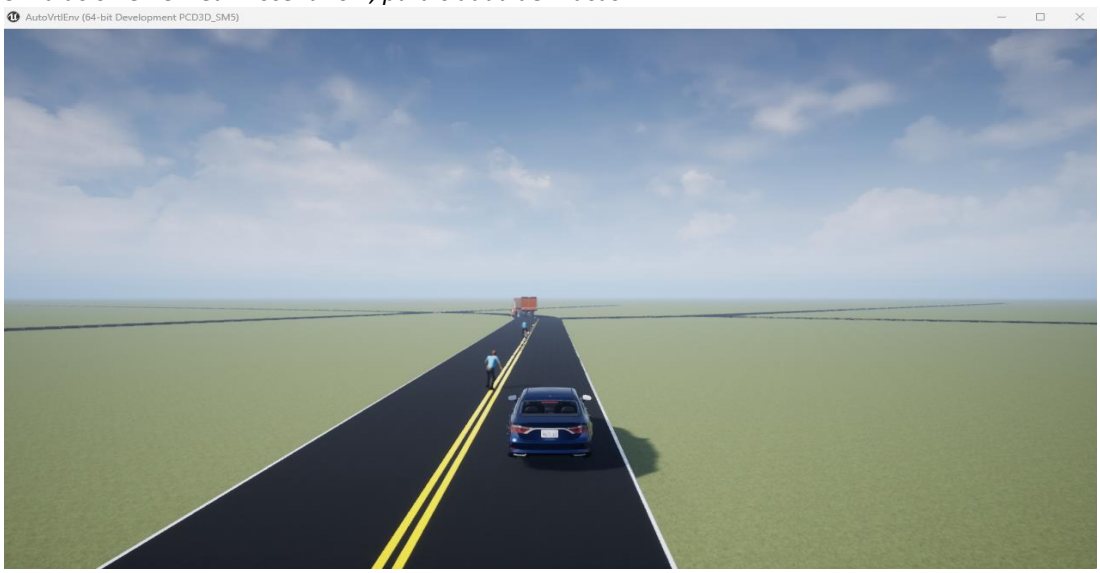


Fuente: Investigador

En la ilustración 11 se muestra la simulación en 3D mediante Unreal Engine, que es una ilustración en tres dimensiones incorporada en MATLAB.

Ilustración 11

Simulación en Unreal- Escenario 1, para ciudad de Macas



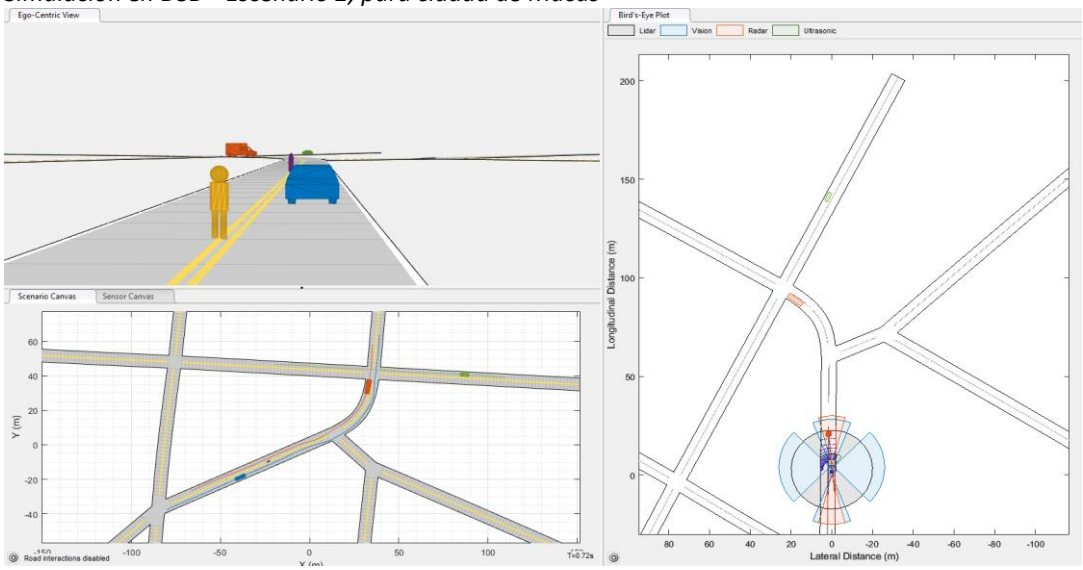
Fuente: Investigador

Simulación escenario 2

Para el escenario 2, de igual forma se procede a realizar la simulación, con dos resultados en DSD, encontrándose tres visiones distintas, en donde se observa una configuración de los actores en forma de cubos en una vista superior observándose la trayectoria y posición de los actores principalmente los vehículos que se mueven y una vista de del auto en donde se tiene la configuración de los sensores, ahí se observa las detecciones mediante puntos respectivos. Esta Simulación se encuentra en la ilustración 12, en donde se puede observar la trayectoria y las detecciones de los sensores respectivamente.

Ilustración 12

Simulación en DSD - Escenario 2, para ciudad de Macas

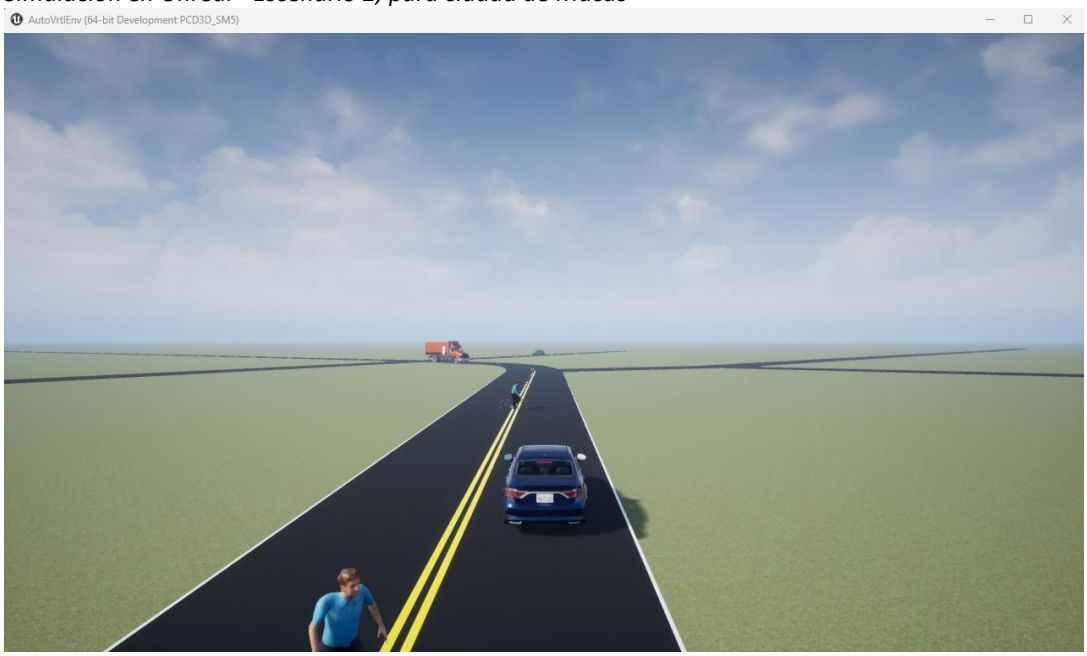


Fuente: Investigador

En la ilustración 13 se muestra la simulación en 3D mediante Unreal Engine, que es una ilustración en tres dimensiones incorporada en MATLAB.

Ilustración 13

Simulación en Unreal - Escenario 2, para ciudad de Macas



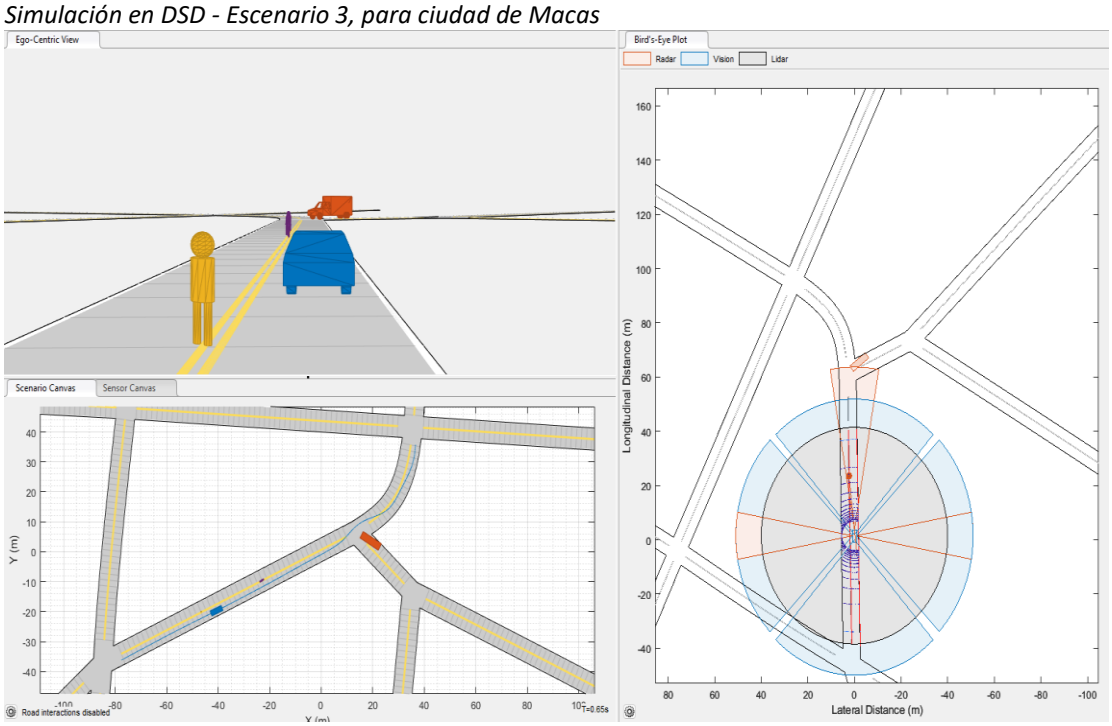
Fuente: Investigador

Simulación escenario 3

Finalmente para el Escenario 3 se obtuvieron los siguientes resultados en DSD, en donde de igual forma se obtuvo una visita de cubos, seguido de una vista superior en donde se observa cual es la trayectoria del vehículo y finalmente la vista de ojos de pájaro en donde se observa la configuración de los sensores y su respectiva detección mediante puntos respectivamente. En la Ilustración 14 se puede observar la simulación en DSD con su respectiva trayectoria, movimiento para los actores y su detección de objetos de los sensores respectivos.

Ilustración 14

Simulación en DSD - Escenario 3, para ciudad de Macas



Fuente: Investigador

En la ilustración 15 se muestra la simulación en 3D mediante Unreal Engine, que es una ilustración en tres dimensiones incorporada en MATLAB.

Ilustración 15

Simulación en Unreal - Escenario 3, para ciudad de Macas

Fuente: Investigador

Codificación generada

Una vez generada cada una de las simulaciones respectivas, con el comando MATLAB function, se procede a exportar y generar una función para el escenario de manejo autónomo y el comportamiento de sensores. Esta generación de algoritmos procede a realizar los tres escenarios respectivos. A continuación se muestra el Script general del primer escenario.

Ilustración 16
Función General- Escenario 1, para ciudad de Macas

```
function [allData, scenario, sensors] = EscenarioDeSimulacionMaacas1()
%EscenarioDeSimulacionMaacas1 - Returns sensor detections
%    allData = EscenarioDeSimulacionMaacas1 returns sensor detections in a structure
%    with time for an internally defined scenario and sensor suite.
%
%    [allData, scenario, sensors] = EscenarioDeSimulacionMaacas1 optionally returns
%    the drivingScenario and detection generator objects.

% Generated by MATLAB(R) 9.13 (R2022b) and Automated Driving Toolbox 3.6 (R2022b).
% Generated on: 14-Mar-2023 00:11:55

% Create the drivingScenario object and ego car
[scenario, egoVehicle] = createDrivingScenario;

% Create all the sensors
[sensors, numSensors] = createSensors(scenario);

allData = struct('Time', {}, 'ActorPoses', {}, 'ObjectDetections', {}, 'LaneDetections', {}, 'PointClo
running = true;
while running

    % Generate the target poses of all actors relative to the ego vehicle
    poses = targetPoses(egoVehicle);
    time  = scenario.SimulationTime;

    objectDetections = {};
    laneDetections   = [];
    ptClouds = {};
```

Fuente: Investigador

En la Ilustración 17 se observa las respuestas en forma de estructura en donde sus campos son los siguientes, tiempo, posición de los actores, Objetos detectados y detección de carril. Dentro de cada uno de los campos se encuentran estructuras y celdas. Para la posición de los objetos se observa que existe la velocidad respectiva con su posición. Para los objetos detectados se observa que existen un número de objetos detectados y su respectiva medida para cada uno de un objeto detectado.

Ilustración 17

Estructura con 6 Campos- Escenario 1, para ciudad de Macas

1x22 struct with 6 fields

Fields	Time	ActorPoses	ObjectDetections	LaneDetections	PointClouds	INSMeasurements
1	0	4x1 struct	3x1 cell	1x4 struct	[]	[]
2	0.1000	4x1 struct	6x1 cell	1x4 struct	[]	[]
3	0.2000	4x1 struct	3x1 cell	1x4 struct	[]	[]
4	0.3000	4x1 struct	7x1 cell	1x4 struct	[]	[]
5	0.4000	4x1 struct	4x1 cell	1x4 struct	[]	[]
6	0.5000	4x1 struct	7x1 cell	1x4 struct	[]	[]
7	0.6000	4x1 struct	8x1 cell	1x4 struct	[]	[]
8	0.7000	4x1 struct	10x1 cell	1x4 struct	[]	[]
9	0.8000	4x1 struct	13x1 cell	1x4 struct	[]	[]
10	0.9000	4x1 struct	9x1 cell	1x4 struct	[]	[]
11	1.0000	4x1 struct	9x1 cell	1x4 struct	[]	[]
12	1.1000	4x1 struct	10x1 cell	1x4 struct	[]	[]
13	1.2000	4x1 struct	9x1 cell	1x4 struct	[]	[]
14	1.3000	4x1 struct	7x1 cell	1x4 struct	[]	[]
15	1.4000	4x1 struct	4x1 cell	1x4 struct	[]	[]
16	1.5000	4x1 struct	5x1 cell	1x4 struct	[]	[]
17	1.6000	4x1 struct	3x1 cell	1x4 struct	[]	[]
18	1.7000	4x1 struct	2x1 cell	1x4 struct	[]	[]
19	1.8000	4x1 struct	2x1 cell	1x4 struct	[]	[]
20	1.9000	4x1 struct	1x1 cell	1x4 struct	[]	[]
21	2.0000	4x1 struct	4x1 cell	1x4 struct	[]	[]
22	2.1000	4x1 struct	1x1 cell	1x4 struct	[]	[]

Fuente: Investigador

2.3. Validación de la propuesta

En este documento se realizó la selección de los expertos, de acuerdo con los siguientes lineamientos: Formación profesional, trayectoria académica, experiencia en su campo laboral y su respectiva motivación a formar parte de esta validación. La siguiente tabla muestra los detalles de los participantes seleccionados para la validación del modelo.

Tabla 4

Descripciones profesionales para validar

Nombres y Apellidos	Años de Experiencia	Titulación Académica	Cargo
Edgar Emanuel Gonzalez Malla	9	Magister en Sistemas y Redes de Comunicación Ingeniero Electrónica y Telecomunicaciones	Docente
Francisco Valverde Alulema	26	Doctor en Informática	Docente
Jhonatan Omar Baca Villavicencio	7	Máster en Modelamiento BIM	Director de Planificación del GADMCS (S)

Fuente: El investigador

Los fines de la aprobación son los siguientes:

- Aprobar los métodos de ejecución utilizados en el transcurso de la investigación.

- Confirmar los resultados obtenidos, conclusiones y recomendaciones.
- Reorientar (si es necesario) los elementos formulados en la propuesta, teniendo en cuenta la experiencia de los expertos.
- Confirmar la probabilidad de aplicar la propuesta de gestión.

Instrumento para validar

Para la presente validación de los criterios respectivos se procedió a realizar una escala desde en total desacuerdo hasta totalmente de acuerdo siendo de 5, esta validación será de importancia y representación. A Continuación se muestra en la tabla 6 las preguntas para la validación respectiva.

Tabla 5

Preguntas Instrumentos de Validación

Criterios	Descripción
Impacto	¿Cree que el modelo de gestión propuesto tendrá un impacto significativo en la creación de valor público?
Aplicabilidad	¿Es válido el contenido sugerido?
Conceptualización	¿Los componentes de la propuesta se basan en conceptos y teorías de los resultados de búsqueda?
Actualidad	¿El contenido de la propuesta tiene en cuenta los procedimientos actuales y posibles nuevos cambios?
Calidad Técnica	Facilita el cumplimiento del modelo ¿Analizar los protocolos de atención desde una perspectiva tecnocientífica?
Factibilidad	¿Es posible incluir en el modelo de control con el resultado está en el capítulo?
Pertinencia	¿El contenido de la propuesta proporciona una solución a la pregunta planteada?

Fuente: UISRAEL

2.4. Matriz de articulación de la propuesta

En la siguiente tabla se muestran ejes del proyecto realizado con los sustentos teóricos, metodológicos, técnicos y tecnológicos empleados.

Tabla 6
Matriz de articulación

Ejes o partes principales del proyecto		Breve descripción de los resultados de cada parte	Sustento teórico que se aplicó en la construcción del proyecto	Metodologías, herramientas técnicas y tecnológicas que se emplearon
1	**Conducción Autónoma de autos Eléctricos**	Se contextualizo el fundamento teórico para cada uno de los sistemas de conducción automática para plantear escenarios específicos. Se recabo información para los distintos autos autónomos en el mercado.	Inteligencia Artificial Internet de las Cosas Matemáticas Aplicadas Comunicaciones Inalámbricas Visión por Computador Machine Learning	Matwoks Automated Driving Toolbox WAYMO ONE TESLA LEXUS
2	**Simulación en Driving Scenario Designer App de Matlab**	Parala simulación respectiva se planteó tres escenarios en base a un contexto base de configuración de sensores y autos de distintos fabricantes.	MatWorks MATLAB Automated Driving Toolbox OpenStreetMap	App Driving Scenario Designer
3	**Implementación de Sensores en el Auto.**	Se configuro cada sensor con su posición, alcance y altura respectiva. Una ves simulado se obtuvo la velocidad, aceleración del auto y el número de detecciones de los sensores con sus medidas respectivas.	MatWorks MATLAB Automated Driving Toolbox OpenStreetMap	App Driving Scenario Designer

Fuente: UISRAEL

2.5. Análisis de resultados. Presentación y discusión.

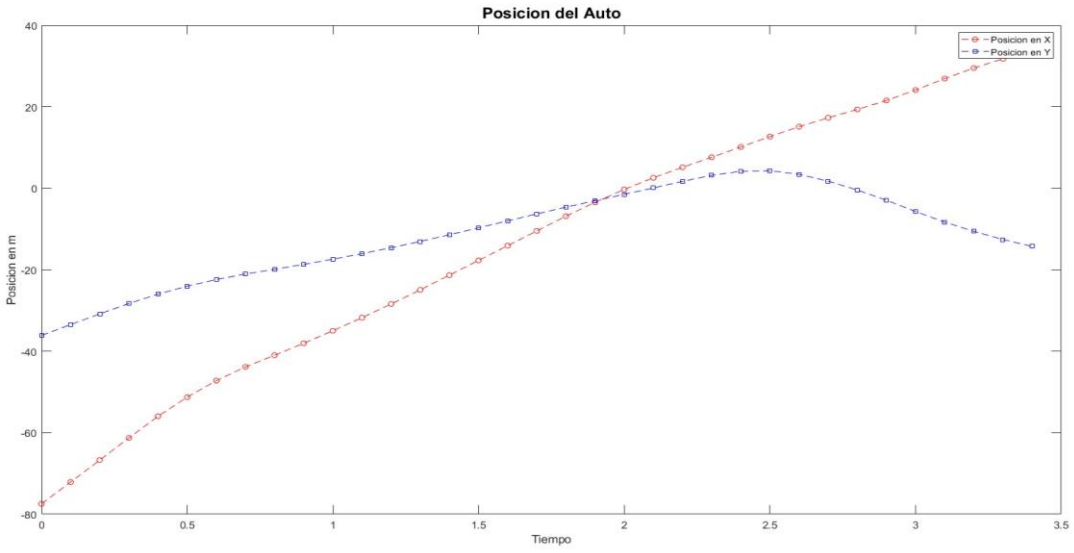
En este punto se mostrará los resultados del análisis de la función creada en DSD, se trabajó en tres scripts, en donde contiene código y muestra la velocidad, la aceleración, las medidas de la detección y el número de detecciones. Entonces a continuación se mostrará por Escenario los resultados respectivos.

Resultados para Escenario 1

Para el escenario se puede observar las siguientes figuras, en donde se muestra la posición en el eje de las X y , de igual manera la velocidad y aceleración de vehículo en los respectivos ejes. Además se muestra dos gráficos del número de detecciones con respecto al tiempo, de igual manera se muestra las medidas de los sensores con respecto al tiempo. En este escenario se observa que existen 12 objetos detectados entre los tiempos 0.5 al 2.5. A Continuación se pueden ver en los gráficos 1,2,3,4,5 y 6. E

Gráfico 1

Posición Auto - Escenario 1, para ciudad de Macas



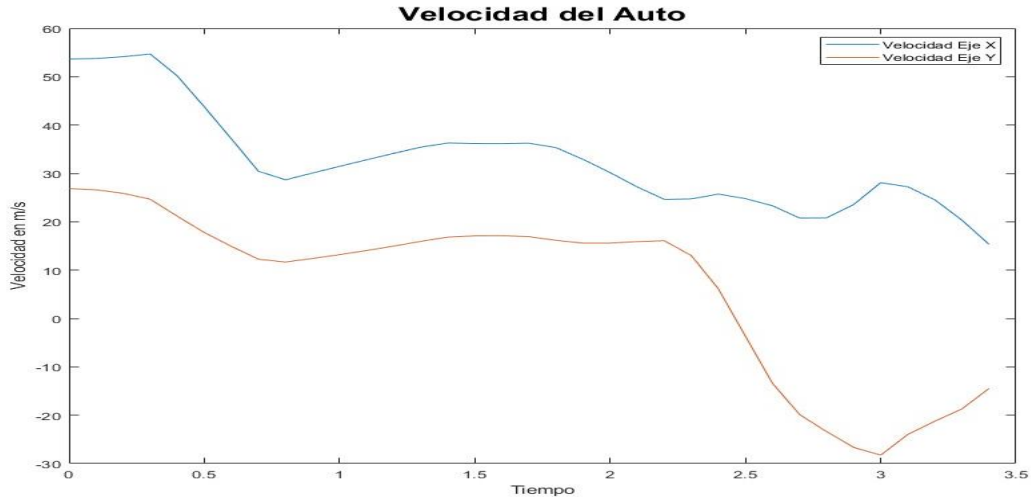
Fuente: Investigador

Para este gráfico se observa la posición del vehículo, en cada uno de sus ejes tanto en X y en Y, se puede observar que para el eje X, su posición comienza desde el punto (0, -80), hasta el punto final de (3.5, 40), en donde se puede observar que crece de manera proporcional. Para el Eje Y se observa que tiene puntos negativos con su punto inicial de (0, -40); y para el tiempo (2.5

, 0), y un punto final es de (3.4, -21). Se observa que para el eje Y tiene puntos negativos y con una cresta respectiva.

Gráfico 2

Velocidad del Auto - Escenario 1, para ciudad de Macas

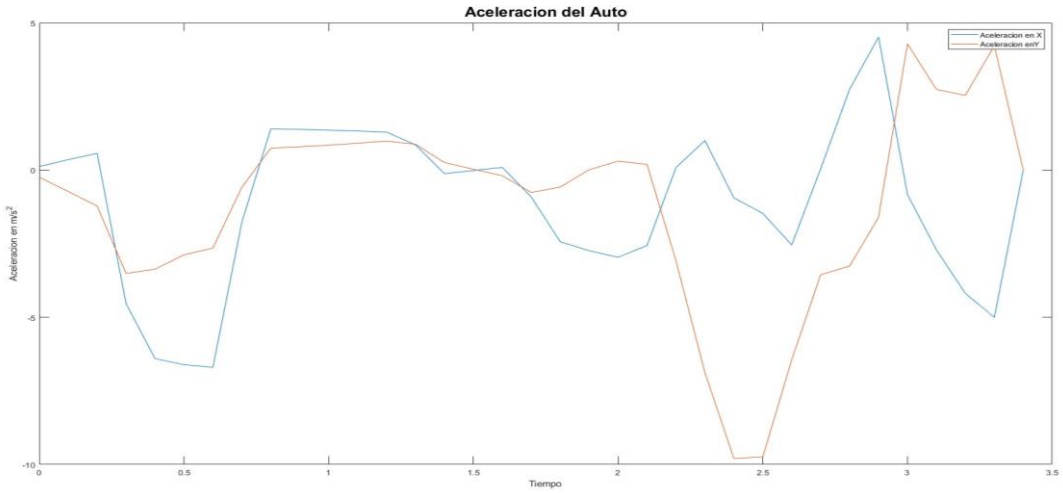


Fuente: Investigador

Para el gráfico 2, se observa la velocidad en cada uno de los ejes en X y en Y. Se observa que para el vehículo para el eje X comienza con una velocidad de 55 m/s y termina con una velocidad de 18m/s, se observa que tiene crecimientos y decrecimientos a medida que aumenta el tiempo. Para el eje Y se observa que la velocidad comienza en 28 m/s, y termina con una velocidad negativa de -15 m/s, esto significa que el desplazamiento es negativo. Se observa subida y bajadas en el eje de las Y, teniendo un punto mínimo de (3,-29).

Gráfico 3

Aceleración Auto - Escenario 1, para ciudad de Macas

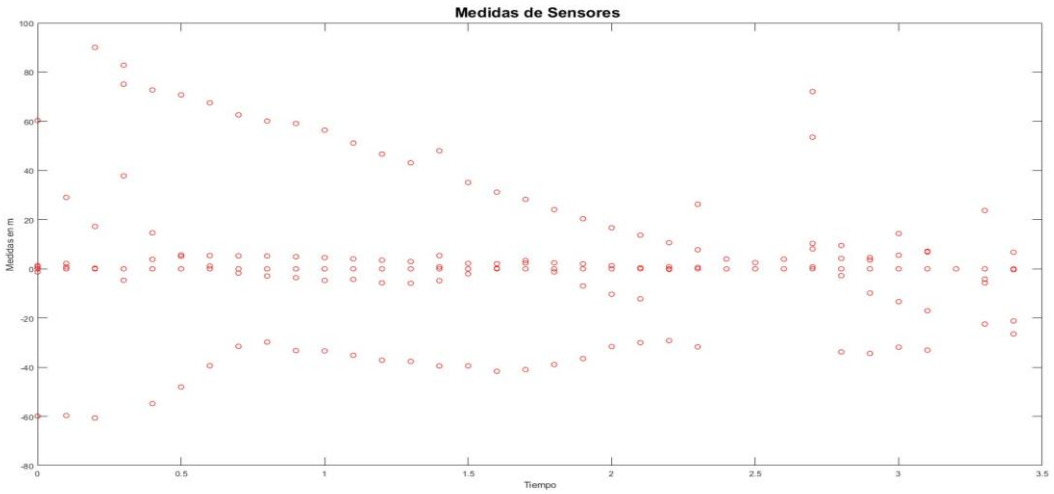


Fuente: Investigador

En el gráfico anterior se presenta la aceleración respecto del auto observándose para los ejes X y Y, se muestra que para el tiempo 0.5, 2.5, una desaceleración de -5.4 m/s2 violenta y así mismo existe una aceleración pico para el punto de tiempo 2.8 con 4 m/s2.

Gráfico 4

Medidas Sensor - Escenario 1, para ciudad de Macas

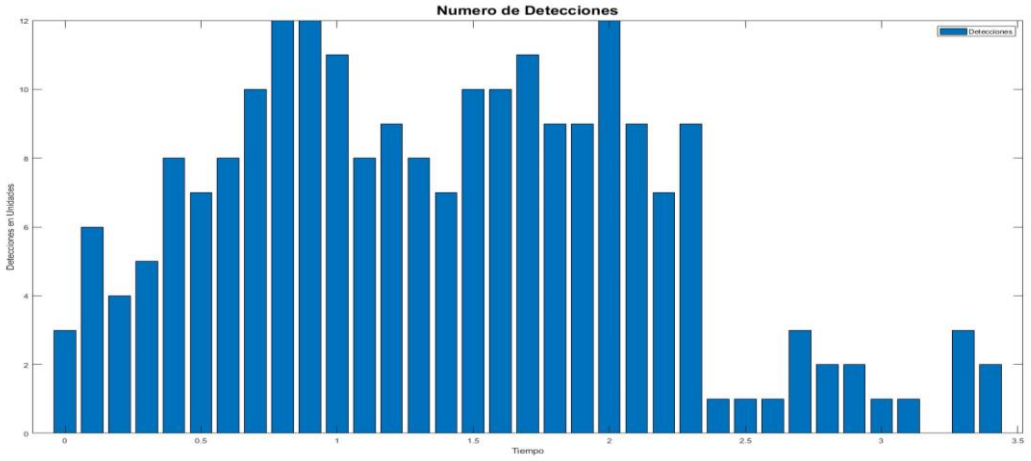


Fuente: Investigador

En este gráfico se muestra cuando el tiempo transcurre detectando las medidas de los sensores, cabe señalar que cada sensor detecta 6 puntos para cada uno de los tiempos, por lo que muestra una medida en m. Se observa que la medida más grande que se detecta espera el tiempo 0.3, con una medida de 90 m. Se observa una tendencia decreciente para las medidas positivas y un creciente senoidal con los puntos negativos. Estos puntos son las distancias desde el auto hasta cada objeto detectado.

Gráfico 5

Número Detecciones - Escenario 1, para ciudad de Macas

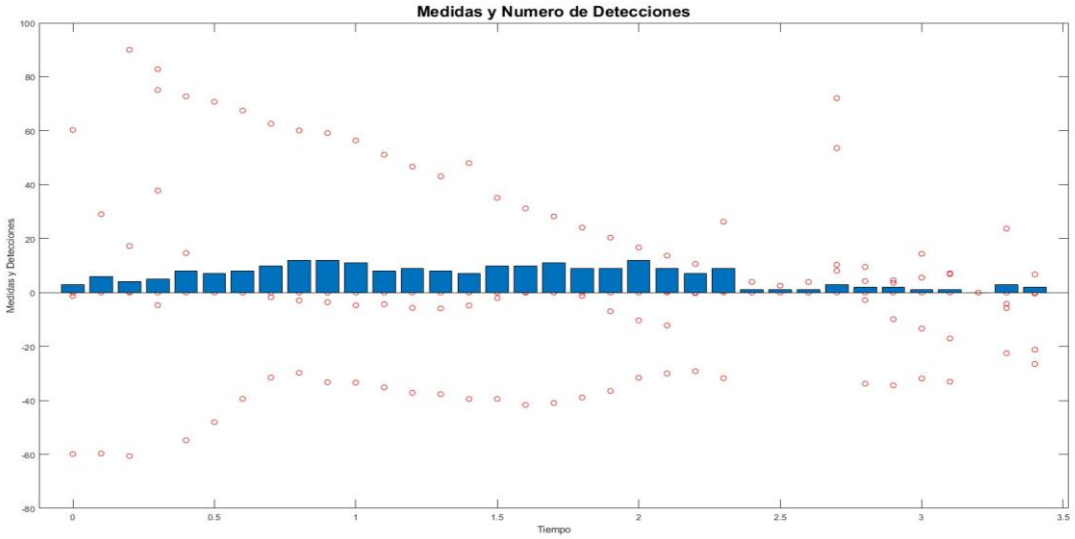


Fuente: Investigador

En el gráfico 5 de barras se muestra el número de detecciones respectivas, para cada uno de los tiempos respectivamente, y se observa dos crestas pronunciadas entre 0.6 hasta 1.1 en tiempo, con su pico de 12 objetos detectados en el tiempo de 0.9. Para la segunda cresta se muestra desde 1.6 hasta 2.2 presentando su pico de 12 objetos para el tiempo 2.

Gráfico 6

Medidas y Detecciones - Escenario 1, para ciudad de Macas



Fuente: Investigador

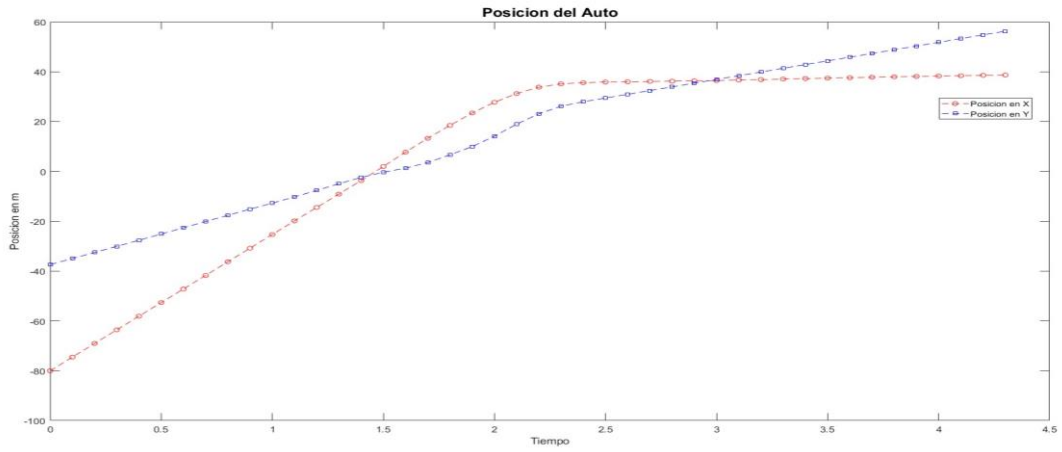
En este gráfico se observa el número de detecciones y las medidas detectadas, en donde se muestra detecciones en un periodo de tiempo amplio desde 0.1 hasta 2.3 con un total de 12 objetos detectados, Sus medidas se muestran de igual forma las detecciones de la distancia con una dispersión de dichas para los tiempos de 2.4 hasta 3.5.

Resultados para Escenario 2

Para El Escenario 2, de igual manera se observa los 6 gráficos de posición, de velocidad, de aceleración, de medidas de los sensores, de número de detecciones y de medidas y detecciones. Con esta configuración se observa que existen 8 objetos detectados entre el tiempo 1.5 y el tiempo 2. Se puede observar a continuación el comportamiento de este escenario en los siguientes gráficos 7,8 ,9 ,10 ,11 y 12.

Gráfico 7

Posición Auto - Escenario 2, para ciudad de Macas

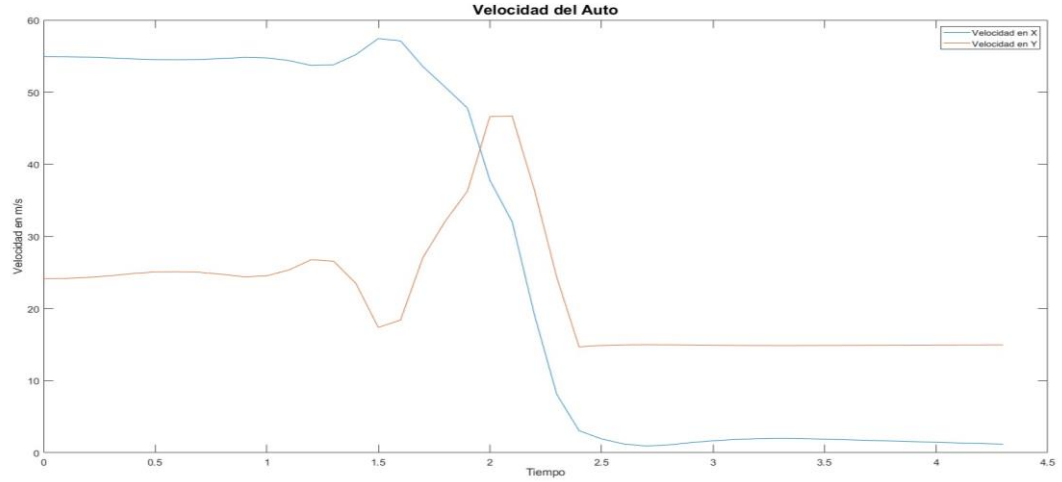


Fuente: *Investigador*

Para la posición del auto en el eje de las X y Y. Para su posición en el eje de las X, muestra una creciente continúa y se estabilizador en el tiempo 2.4, con su posición de 30m. De igual manera para el eje de las Y, se muestra un crecimiento continuo, con un ligero alteración para el intervalo de tiempo de 1.5 a 3.

Gráfico 8

Velocidad Auto - Escenario 2, para ciudad de Macas

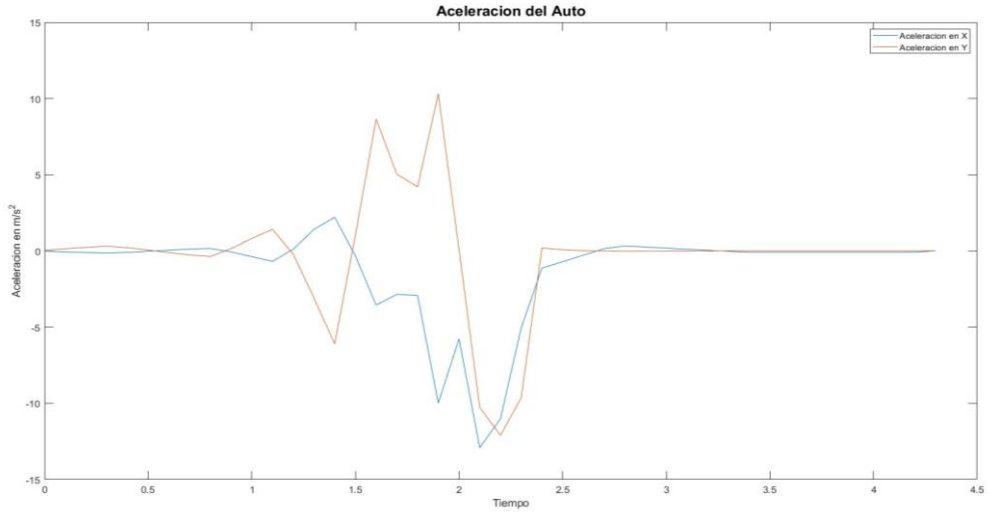


Fuente: Investigador

En este gráfico se observa la velocidad en cada uno de los ejes respectivos. Para el eje X se observa una velocidad inicial de 55 m/s, y con su cresta de 60 m/s en el tiempo 1.5 respectivo, posteriormente se observa un decrecimiento pronunciado en donde a partir del tiempo 2.5 la velocidad es cero. Para el eje en Y, el inicio es de 35 m/s en el tiempo cero, se muestra una cresta de 48 m/s en el tiempo de 2.5 y una caída de hasta el tiempo de 2.4, llegando la velocidad hasta 15 m/s, posteriormente se mantiene estable.

Gráfico 9

Aceleración Auto - Escenario 2, para ciudad de Macas

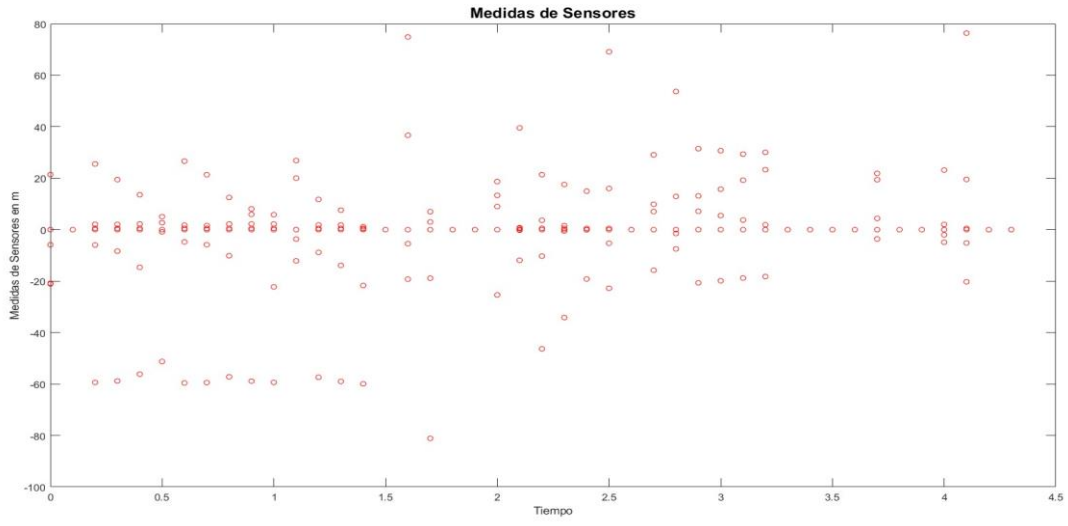


Fuente: Investigador

La aceleración se muestra que es casi cero con ligeros incrementos en eje de las X y Y, excepto en un intervalo de tiempo de 1.1 hasta 2.5. En donde para el eje de las X se muestra un decrecimiento con su punto mínimo de -10.5 m/s2 en el tiempo de 2.1. Para el eje de las Y se observa que dos crestas y dos puntos mínimos, en donde su punto máximo es de 10 m/s2, en el tiempo 1.8, y su punto mínimo es de -12 m/s2 en el tiempo de 2.4.

Gráfico 10

Medidas Sensores - Escenario 2, para ciudad de Macas

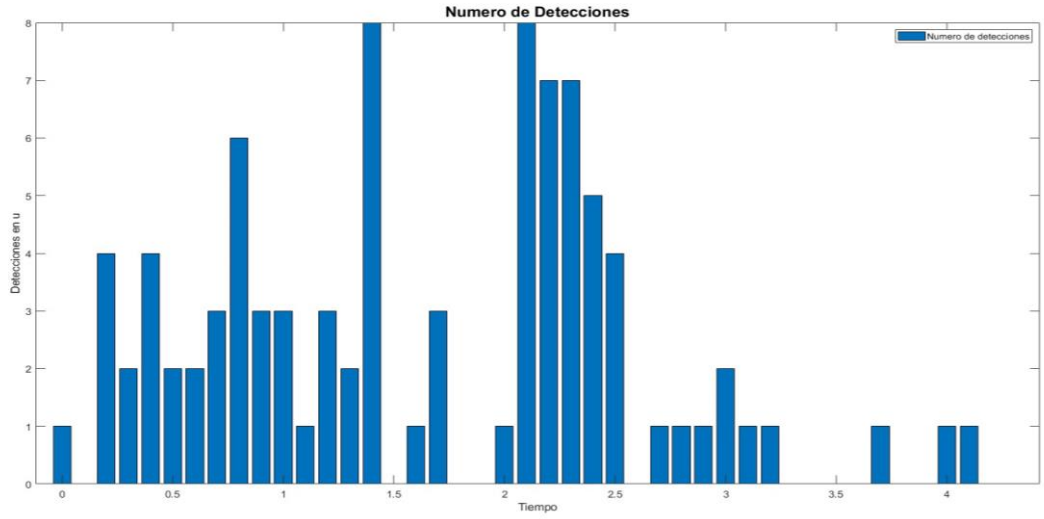


Fuente: Investigador

En la gráfica 10 se muestran las medidas de los sensores detectados, observándose que existe una medida máxima de 70.6m y una medida mínima negativa de -80m. Se observa que las medidas positivas en mayor proporción están entre los tiempos de 1 y 3.1.

Gráfico 11

Número Detecciones - Escenario 2, para ciudad de Macas

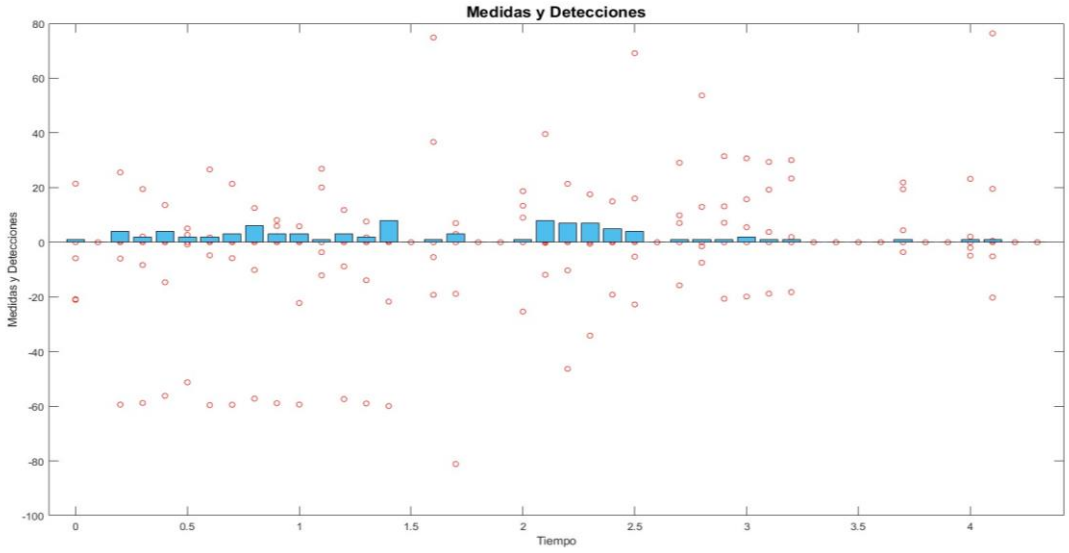


Fuente: Investigador

En esta gráfica se observa el número de detecciones respectivas. Se muestra que se tiene un total de 8 detecciones respectivas para los tiempos de 1.4 y 2.1 respectivamente. En la gráfica se muestra en mayor proporción las detecciones de tiempo desde 0.1 hasta 2.5.

Gráfico 12

Medidas y Detecciones - Escenario 2, para ciudad de Macas



Fuente: Investigador

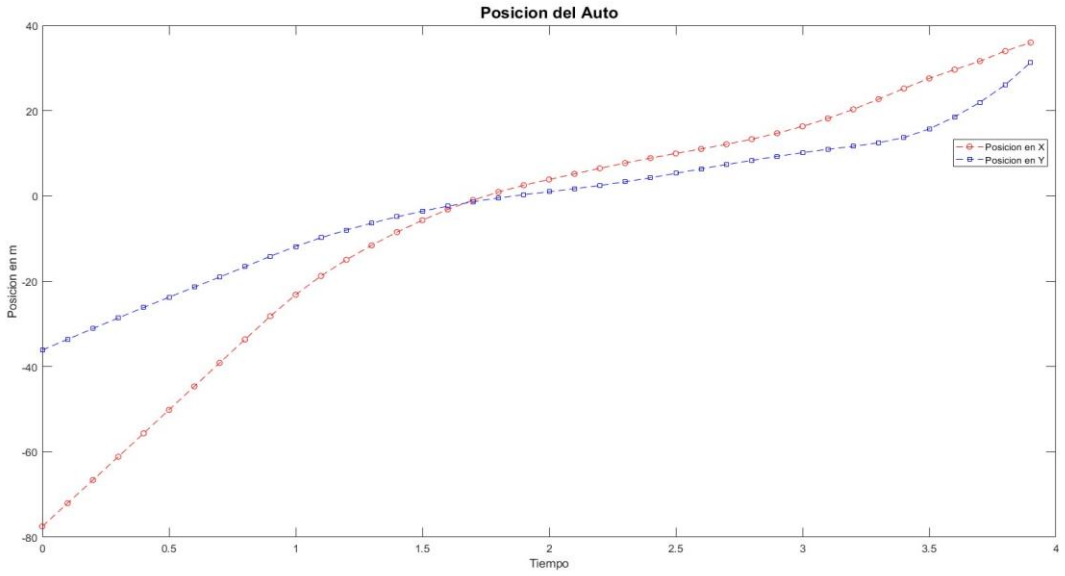
En esta gráfica se muestra la detección de objetos con las medidas respectivas, se muestra que en un intervalo de mayor proporción de detecciones es desde el tiempo 2 hasta 2.5. En el gráfico se observa que el tiempo crece con tendencia positiva, las medidas son producidas de acuerdo con el número de detecciones respectivas.

Resultados para Escenario 3

De la misma manera que en los escenarios anteriores para este escenario se procedió a obtener los 6 gráficos respectivos de posición, de velocidad, de aceleración, de medidas de los sensores, de número de detecciones y de medidas y detecciones. Con esta configuración de sensores se observa que existen 12 objetos detectados en el tiempo 1 y 5 en el tiempo 3. Se puede observar a continuación el comportamiento de este escenario en los siguientes gráficos 13 ,14 ,15 ,16 ,17 y 18.

Gráfico 13

Posición Auto - Escenario 3, para ciudad de Macas

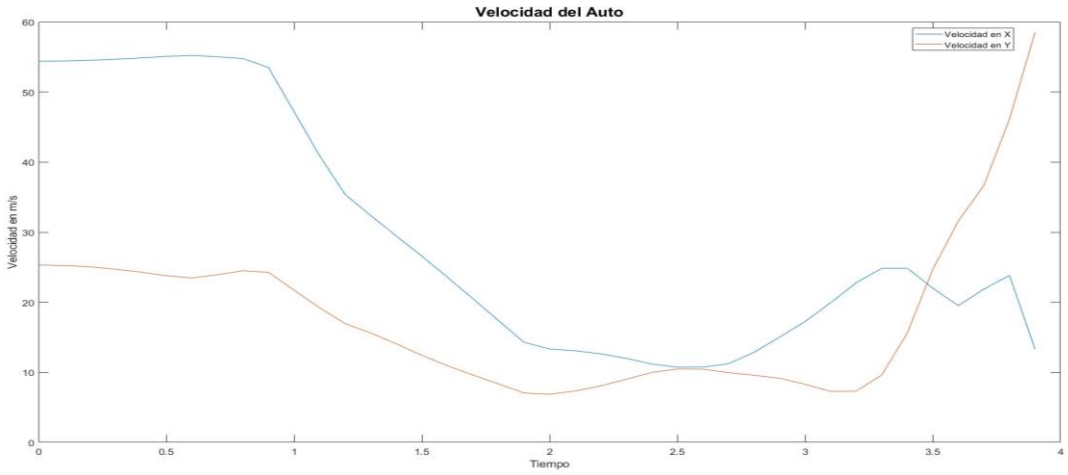


Fuente: Investigador

En este primer gráfico se observa de igual manera la posición en el eje X y Y a medida que incrementa el tiempo. Para el Eje en X se muestra un incremento continuo hasta el tiempo 1.5, posteriormente ese incremento se aplana hasta el tiempo 3. Para el eje Y se observa una creciente de la posición muy vaga, en hasta el tiempo 3.5, en donde se crece de inmediato.

Gráfico 14

Velocidad Auto - Escenario 3, para ciudad de Macas

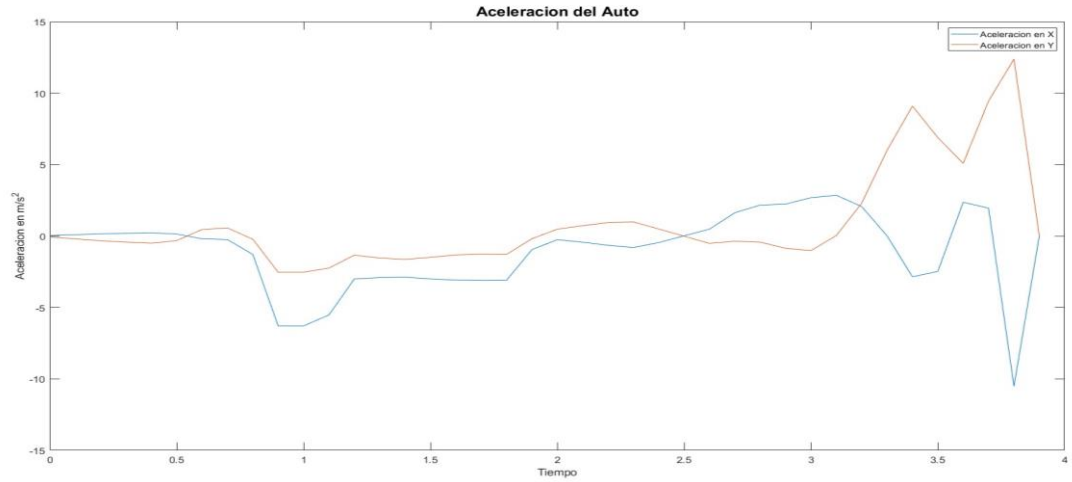


Fuente: Investigador

En la gráfica antes mencionada, se muestra la velocidad para los ejes X y Y. En donde para el eje en X se muestra una caída desde tiempo 1 hasta 3 respectivamente con un punto mínimo de 10 m/s. Para el eje de las Y, se muestra una leve estabilidad hasta el punto 3.2 en donde a partir de ese punto crece de manera inmediata.

Gráfico 15

Aceleración Auto - Escenario 3, para ciudad de Macas

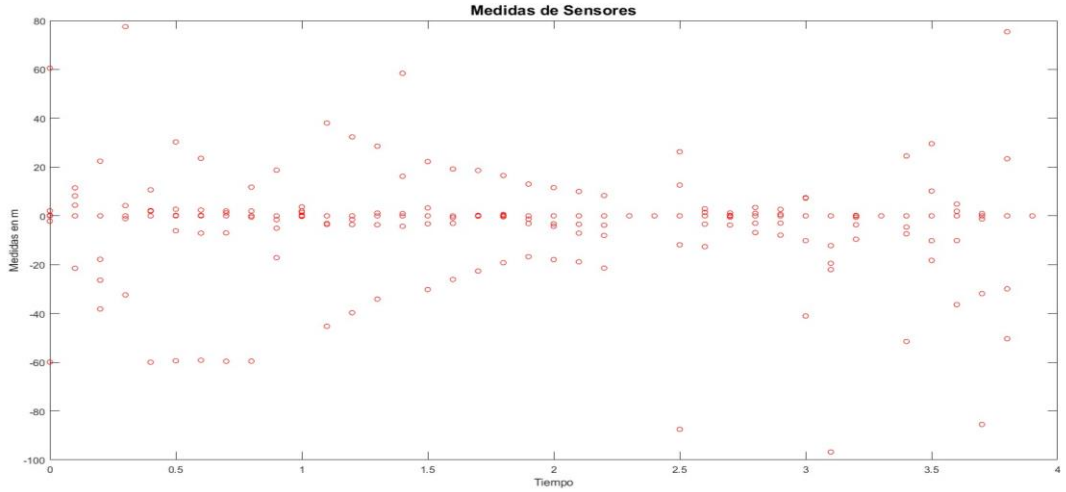


Fuente: Investigador

En el gráfico anterior se muestra la aceleración respectiva en donde se tiene una alteración significativa en la última etapa de tiempo esto es desde el tiempo 3 hasta 4, y una caída de esta en los periodos de tiempo desde 0.8 hasta 1.2. Se muestra de manera general qué aceleración tanto en el eje de las X y Y en donde son muy parecidas.

Gráfico 16

Medidas Sensor - Escenario 3, para ciudad de Macas

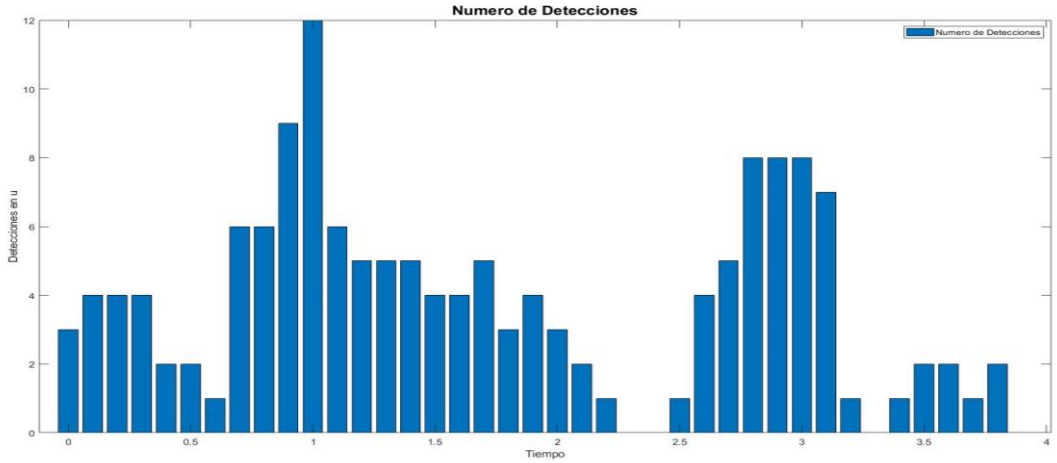


Fuente: Investigador

Los datos de lectura de los sensores son mostrados en el gráfico 16 en donde, se muestra la medida más pico que es de 80 m desde el tiempo 0.4, seguida de 78m en el tiempo de 3.8. Se observa en la gráfica que claramente existe una tendencia de detección de medidas desde el tiempo 0 hasta 2.3.

Gráfico 17

Número Detecciones - Escenario 3, para ciudad de Macas



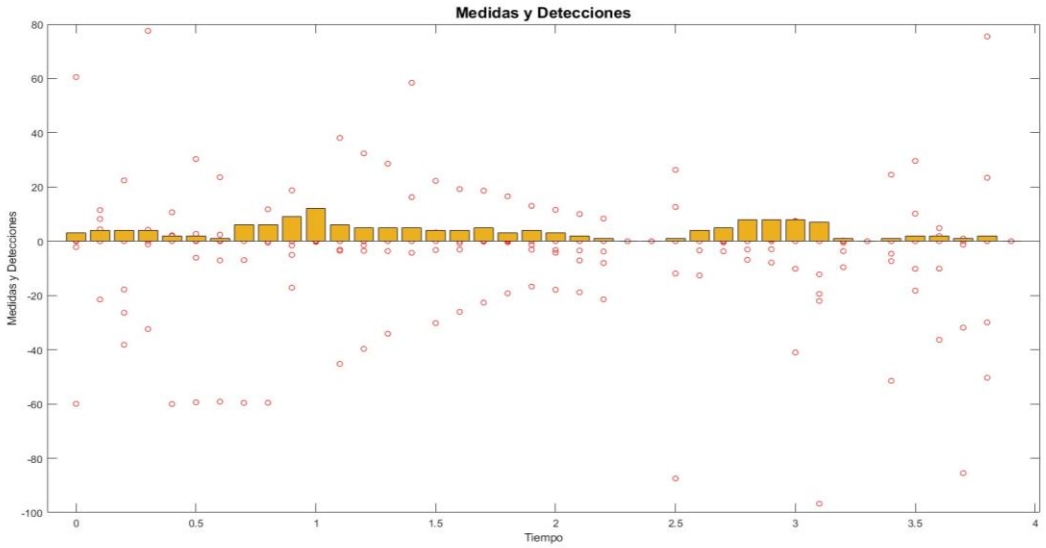
Fuente: Investigador

En el gráfico anterior se muestran el número de detecciones respectivas, para la configuración de sensores respectivos. Se tiene que en el tiempo 1 se detectan la mayor cantidad de objetos que son de 12. Además se observa claramente que en los primeros instantes de

tiempos existen una mayor detección de los objetos estos es desde el tiempo 0 hasta el tiempo 2.2.

Gráfico 18

Medidas y Detecciones - Escenario 3, para ciudad de Macas



Fuente: Investigador

En este gráfico se observa las medidas de los sensores y el número de detecciones, observándose que existen medidas que las medidas con mayor distancia concuerdan con el mayor número de objetos detectados, excepto 3 medidas en los tiempos 0.3, 1.4 y 3.3 respectivamente.

CONCLUSIONES

De acuerdo con las simulaciones de los tres escenarios y de acuerdo los gráficos de resultados mostrando la posición, la aceleración y velocidad en el eje de las X y Y, posteriormente las medidas de los sensores y el número de objetos detectados, a continuación se concluye los siguientes:

- De acuerdo a datos presentados por el ECU 911 en la ciudad de Macas se a reportado 102 accidentes de tránsito, en función de su página virtual de ECU 911 se producen 14 accidentes de tránsito en la zona de la Av.29 de Mayo y Gavino Rivadeneira, es por eso que se pudo realizar una explicación del fundamento teórico para un sistema de conducción autónoma de manera general y en su escenario de la Ciudad de Macas, planteando escenario como autos, trailers, peatones y ciclistas respectivamente. Recreando un escenario real explicando su fundamento respectivo.
- Para la simulación se plantearon 3 escenarios respectivamente, en base a una configuración de sensores base de tres fabricantes de autos eléctricos autónomos como son WAYMO, TESLA y LEXUS, en donde se tomaron sus ubicaciones respectivas, cambiando en cada escenario su configuración de altura, ángulo de apertura y alcance del sensor respectivamente. Una vez configurado estos escenarios se aplicó el Driving Scenario Designer App de MATLAB, para poder observar el comportamiento de los sensores y sus medidas, además de las respectivas velocidades y aceleración.
- Para cada escenario se aplicó las respectivas simulaciones encontrándose con la posición y velocidad respectivamente en cada uno de los ejes X y Y. Se muestra que para el escenario 1 la velocidad y aceleración tiene picos considerables e inestables en el tiempo, comparado con los escenarios 2 y 3, debido a la trayectoria del vehículo, además de su detección de objetos, en donde se produce su frenado respectivo para cambiar de dirección y evitar la colisión con el trailer. Además la configuración de sensores que mayor detección se produce es para el escenario 1 en donde se configura un escenario base de los sensores del auto WAYMO ONE, colocados a una altura de 2m las cámaras frontales a 0.80m el radar y el sensor ultrasónico. El alcance que se dio a esta configuración es de 90 m, 60 m para las cámaras, 75m para el radar y 12 metros para el sensor ultrasónico.
- Además se muestra que en el escenario 3 se obtienen mayores resultados en cuanto a detección en comparación con el escenario 2, esto debido a la configuración de los sensores y su ubicación. Para el escenario 3 se tiene una configuración base del auto Lexus, en donde se colocó las cámaras a una altura de 1.10, los sensores tipo radar a

0.20 m, y el sensor tipo LiDAR a una altura de 2 m. Cada uno de estos tiene un alcance de 50 m respectivamente.

RECOMENDACIONES

De acuerdo con la simulación plateada y de sus gráficos respectivos se encontraron con las siguientes recomendaciones que describen a continuación:

- Se recomienda para el fundamento técnico teórico, profundizar, revisar el funcionamiento de un auto eléctrico, el esquema eléctrico de los sensores y su funcionamiento de movilidad, para poder realizar la implementación respectiva en la ciudad de macas, esto debido a la intervención del clima y de la topografía del terreno.
- Para la simulación se recomienda tomar como objeto de escenario a los trailer automáticos con su respectiva configuración base de los sensores, para su implementación en la ciudad de Macas. Además se recomienda la reubicación en otro lugar de la ciudad de macas, en donde los datos de accidentes sean altos para poder obtener el comportamiento de este trailer eléctricos autónomos.
- En cuanto a los datos de los sensores adquiridos se recomienda cambiar la ubicación y características de estos con su respectivo ángulo vertical, y su altura para poder observar el comportamiento de la adquisición de las detecciones. Además se recomienda aplicar estos sensores a un auto eléctrico que exista en el mercado para poder hacer un acoplamiento y poder observar dicho comportamiento y poder usar esos datos.
- Se recomienda realizar la planificación y control, para los escenarios ya planteados en la ciudad de Macas, en los cuales se deberá modelar y realizar el tema de control para poder automatizar adecuadamente en el sitio deseado.

BIBLIOGRAFÍA

AEADE. (2022). *Secotr Automotor en Cifras*. *77*.

Alban. (2016). *Influencia de la Jerarquizacion y senalizacion en la seguridad vial del casco central de la ciudad de Macas, Provincia Morona Santiago, Periodo 2015*. 140. http://dspace.espoch.edu.ec/handle/123456789/5016

Concepto. (2023). *Tecnicas de Investigación*.

ECU 911. (2023). *Informacion Estadistica SIS ECU 911*. https://www.ecu911.gob.ec/

Escobar, P., Gonzalez, E., Miño, E., & Morales, F. (2021). Object Detection in a Virtual Simulation Environment with Automated Driving Toolbox. *Iberian Conference on Information Systems and Technologies, CISTI*, *June*, 23–26. https://doi.org/10.23919/CISTI52073.2021.9476414

Estaña, G. (2017). El estándar SAE J3016: Despliegue del piloto automático en los automóviles y automatización del tráfico rodado. *Zaguan.Unizar.Es*, 186.

Ferrás, C., & López-lópez, P. C. (2021). *Information Technology and Systems* (Vol. 2).

Giurgica, G., & Florescu, R. D. (2020). A case study for modeling autonomous driving systems. *2020 24th International Conference on System Theory, Control and Computing, ICSTCC 2020 - Proceedings*, 745–750. https://doi.org/10.1109/ICSTCC50638.2020.9259650

Herrero, J. (2020). *Desarrollo E Implementacion De Algoritmos De Conduccion Autonoma En Simuladores*.

INEC. (2022). *Nota Técnica-Metodológica Siniestros de tránsito Estadísticas de Transporte-ESTRA*.

INEC. (2023). *Estadisticas de Transporte*. https://app.powerbi.com/view?r=eyJrIjoiN2FhOTIxNTktNDA1My00OTQ2LWI2YTItYTIyY2E3NjFiMGY0IiwidCI6ImYxNThhMmU4LWNhZWMtNDQwNi1iMGFiLWY1ZTI1OWJkYTExMiJ9

Laencina, R. (2016). *Diseño de un sistema avanzado de asistencia al conductor (ADAS) basado en visión artificial para predicción del tiempo-de-colisión (TTC)*. http://repositorio.upct.es/handle/10317/6129

León, F. L., & Salinas, M. (2018). *Implementación y uso de autos eléctricos en el transporte público y su impacto en la red de distribución*. 5–130.

MathWorks. (2023). *Sistemas Avanzados de Asistencia al Conductor*. https://la.mathworks.com/discovery/adas.html#:~:text=Los sistemas avanzados de asistencia,las tareas de un conductor.

Miranda, F. (2021). *Fusión sensorial Cámara y LiDAR para el seguimiento de múltiples objetivos. Aplicación a vehículos de conducción autónoma*. http://hdl.handle.net/10317/9388

Miura, K., & Azumi, T. (2020). Converting driving scenario framework for testing self-driving systems. *Proceedings - 2020 IEEE 18th International Conference on Embedded and Ubiquitous Computing, EUC 2020*, 56–63. https://doi.org/10.1109/EUC50751.2020.00015

Navin Kumar, M., Pravin Kumar, S., Premkumar, R., & Navaneethakrishnan, L. (2021). Smart Characterization of Vehicle Impact and Accident Reporting System. *2021 7th International Conference on Advanced Computing and Communication Systems, ICACCS 2021*, 964–968. https://doi.org/10.1109/ICACCS51430.2021.9441797

Niu, M. (2018). Object Detection and Tracking for Autonomous Driving by MATLAB toolbox.

Journal of Controlled Release, *11*(2), 430–439.

Pico, G. (2019). *Sistema avanzado de asistencia al conductor empleando visión artificial en vehículos de transporte público*. 128. https://repositorio.uta.edu.ec/bitstream/123456789/29951/1/Tesis_1600ec.PDF

Ramirez, F., & Zwerg, A. (2012). Metodologia de la Investigacion: Mas que una receta. *AD-Minister*, *20*, 91–111.

Sampieri, R., Collado, C., & Lucio, P. (2010). Metodologia de la Investigacion. In *Nucl. Phys.* (Vol. 13, Issue 1).

Stadler, C., Montanari, F., Baron, W., Sippl, C., & Djanatliev, A. (2022). A Credibility Assessment Approach for Scenario-Based Virtual Testing of Automated Driving Functions. *IEEE Open Journal of Intelligent Transportation Systems*, *3*(January), 45–60. https://doi.org/10.1109/ojits.2022.3140493

Zambrano, E. (2022). *Sistema De Monitorización Y Control Inteligente Autónomo Para Vehículos Combinando Tecnología Iot Y Redes Neuronales*.

ANEXOS

Escala de evaluación de criterios del Mag. Edgar Emanuel Gonzalez Malla

CRITERIOS	EVALUACIÓN SEGÚN IMPORTANCIA Y REPRESENTATIVIDAD				
	En Total Desacuerdo	En Desacuerdo	Ni de Acuerdo Ni en Desacuerdo	De Acuerdo	Totalmente Acuerdo
Impacto					X
Aplicabilidad					X
Conceptualización					X
Actualidad					X
Calidad Técnica					X
Factibilidad					X
Pertinencia				X	

Elaborado por: El investigador

Escala de evaluación de criterios del Mag. Omar Baca Villavicencio

CRITERIOS	EVALUACIÓN SEGÚN IMPORTANCIA Y REPRESENTATIVIDAD				
	En Total Desacuerdo	En Desacuerdo	Ni de Acuerdo Ni en Desacuerdo	De Acuerdo	Totalmente Acuerdo
Impacto					X
Aplicabilidad					X
Conceptualización					X
Actualidad					X
Calidad Técnica					X
Factibilidad				X	
Pertinencia					X

Elaborado por: El investigador

Escala de evaluación de criterios del PhD. Francisco Valverde Alulema

CRITERIOS	EVALUACIÓN SEGÚN IMPORTANCIA Y REPRESENTATIVIDAD				
	En Total Desacuerdo	En Desacuerdo	Ni de Acuerdo Ni en Desacuerdo	De Acuerdo	Totalmente Acuerdo
Impacto					X
Aplicabilidad					X
Conceptualización					X
Actualidad				X	
Calidad Técnica					X
Factibilidad					X
Pertinencia					X

Elaborado por: El investigador

Yo, **Edgar Emanuel Gonzalez Malla** C.I: **1104934755** en mi calidad de validador de la propuesta del proyecto titulado: **SIMULACIÓN DE SISTEMAS DE CONDUCCIÓN AUTÓNOMO PARA LA CIUDAD DE MACAS MEDIANTE DRIVING SCENARIO DESIGN APP DE MATLAB**.

Elaborado por el Ing. **Juan Ennis Espinoza González**, de C.I: **1400799852**, estudiante de la Maestría: **ELECTRÓNICA Y AUTOMATIZACIÓN**, resolución: RPC-SO-09-No.265-2021, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL),** como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar haber revisado el trabajo y realizado la evaluación de criterios

Quito D.M., 16 de marzo del 2023

Firmado electrónicamente por:
EDGAR EMANUEL GONZALEZ MALLA

Edgar Emanuel Gonzalez Malla

C.I: 1104934755

Registro SENECYT: 7241133197

Yo, **Jhonatan Omar Baca Villavicencio** C.I: **1400515324** en mi calidad de validador de la propuesta del proyecto titulado: **SIMULACIÓN DE SISTEMAS DE CONDUCCIÓN AUTÓNOMO PARA LA CIUDAD DE MACAS MEDIANTE DRIVING SCENARIO DESIGN APP DE MATLAB**.

Elaborado por el Ing. **Juan Ennis Espinoza González**, de C.I: **1400799852**, estudiante de la Maestría: **ELECTRÓNICA Y AUTOMATIZACIÓN**, resolución: RPC-SO-09-No.265-2021, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL),** como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar haber revisado el trabajo y realizado la evaluación de criterios

Quito D.M., 16 de marzo del 2023

Firmado electrónicamente por:
JHONATAN OMAR BACA VILLAVICENCIO

Jhonatan Omar Baca Villavicencio

C.I: 1400515324

Registro SENECYT: 1029-16-1452644

Yo, **Francisco Xavier Valverde Alulema** C.I: **1712156684** en mi calidad de validador de la propuesta del proyecto titulado: **SIMULACIÓN DE SISTEMAS DE CONDUCCIÓN AUTÓNOMO PARA LA CIUDAD DE MACAS MEDIANTE DRIVING SCENARIO DESIGN APP DE MATLAB**.

Elaborado por el Ing. **Juan Ennis Espinoza González**, de C.I: **1400799852**, estudiante de la Maestría: **ELECTRÓNICA Y AUTOMATIZACIÓN**, resolución: RPC-SO-09-No.265-2021, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL),** como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar haber revisado el trabajo y realizado la evaluación de criterios

Quito D.M., 16 de marzo del 2023

FRANCISCO XAVIER VALVERDE ALULEMA

Firmado digitalmente por FRANCISCO XAVIER VALVERDE ALULEMA
Fecha: 2023.03.16 23:06:10 -05'00'

Francisco Xavier Valverde Alulema

C.I: 1712156684

Registro SENECYT: 7526R-12-4071

ANEXO 1

SCRIPT FUNCTION ESCENARIO 1

```
function [allData, scenario, sensors] = EscenarioDeSimulacionMaacas1()
%EscenarioDeSimulacionMaacas1 - Returns sensor detections
%    allData = EscenarioDeSimulacionMaacas1 returns sensor detections in a structure
%    with time for an internally defined scenario and sensor suite.
%
%    [allData, scenario, sensors] = EscenarioDeSimulacionMaacas1 optionally returns
%    the drivingScenario and detection generator objects.
% Generated by MATLAB(R) 9.13 (R2022b) and Automated Driving Toolbox 3.6 (R2022b).
% Generated on: 14-Mar-2023 09:38:42
% Create the drivingScenario object and ego car
[scenario, egoVehicle] = createDrivingScenario;
% Create all the sensors
[sensors, numSensors] = createSensors(scenario);
allData = struct('Time', {}, 'ActorPoses', {}, 'ObjectDetections', {}, 'LaneDetections',
{}, 'PointClouds', {}, 'INSMeasurements', {});
running = true;
while running
   % Generate the target poses of all actors relative to the ego vehicle
   poses = targetPoses(egoVehicle);
   time  = scenario.SimulationTime;
   objectDetections = {};
   laneDetections   = [];
   ptClouds = {};
   insMeas = {};
   isValidTime = false(1, numSensors);
   isValidLaneTime = false(1, numSensors);
   isValidPointCloudTime = false(1, numSensors);
   isValidINSTime = false(1, numSensors);
   % Generate detections for each sensor
   for sensorIndex = 1:numSensors
       sensor = sensors{sensorIndex};
       % Generate the ego vehicle lane boundaries
       if isa(sensor, 'visionDetectionGenerator')
           maxLaneDetectionRange = min(500,sensor.MaxRange);
           lanes = laneBoundaries(egoVehicle, 'XDistance', linspace(-
maxLaneDetectionRange, maxLaneDetectionRange, 101));
       end
       type = getDetectorOutput(sensor);
```

```
        if strcmp(type, 'Objects only')
            if isa(sensor,'ultrasonicDetectionGenerator')
                [objectDets, isValidTime(sensorIndex)] = sensor(poses, time);
                numObjects = length(objectDets);
            else
                [objectDets, numObjects, isValidTime(sensorIndex)] = sensor(poses, time);
            end
            objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes only')
            [laneDets, ~, isValidTime(sensorIndex)] = sensor(lanes, time);
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes and objects')
            [objectDets, numObjects, isValidTime(sensorIndex), laneDets, ~,
isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);
            objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes with occlusion')
            [laneDets, ~, isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'PointCloud')
            if sensor.HasRoadsInputPort
                rdmesh = roadMesh(egoVehicle,min(500,sensor.MaxRange));
                [ptCloud, isValidPointCloudTime(sensorIndex)] = sensor(poses, rdmesh,
time);
            else
                [ptCloud, isValidPointCloudTime(sensorIndex)] = sensor(poses, time);
            end
            ptClouds = [ptClouds; ptCloud]; %#ok<AGROW>
        elseif strcmp(type, 'INSMeasurement')
            insMeasCurrent = sensor(actorState, time);
            insMeas = [insMeas; insMeasCurrent]; %#ok<AGROW>
            isValidINSTime(sensorIndex) = true;
        end
    end
    % Aggregate all detections into a structure for later use
    if any(isValidTime) || any(isValidLaneTime) || any(isValidPointCloudTime) ||
any(isValidINSTime)
        allData(end + 1) = struct( ...
            'Time',       scenario.SimulationTime, ...
            'ActorPoses', actorPoses(scenario), ...
            'ObjectDetections', {objectDetections}, ...
            'LaneDetections', {laneDetections}, ...
```

```
            'PointClouds',   {ptClouds}, ... %#ok<AGROW>
            'INSMeasurements',   {insMeas}); %#ok<AGROW>
    end
    % Advance the scenario one time step and exit the loop if the scenario is complete
    running = advance(scenario);
end
% Restart the driving scenario to return the actors to their initial positions.
restart(scenario);
% Release all the sensor objects so they can be used again.
for sensorIndex = 1:numSensors
    release(sensors{sensorIndex});
end
%%%%%%%%%%%%%%%%%%%%
% Helper functions %
%%%%%%%%%%%%%%%%%%%%
% Units used in createSensors and createDrivingScenario
% Distance/Position - meters
% Speed             - meters/second
% Angles            - degrees
% RCS Pattern       - dBsm
function [sensors, numSensors] = createSensors(scenario)
% createSensors Returns all sensor objects to generate detections
% Assign into each sensor the physical and radar profiles for all actors
profiles = actorProfiles(scenario);
sensors{1} = visionDetectionGenerator('SensorIndex', 1, ...
    'SensorLocation', [1.9 0], ...
    'Height', 2, ...
    'MaxRange', 60, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([300 799.999999999999],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{2} = visionDetectionGenerator('SensorIndex', 2, ...
    'SensorLocation', [0 0.9], ...
    'Height', 2, ...
    'Yaw', 157.195459528679, ...
    'MaxRange', 60, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([320 320],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{3} = visionDetectionGenerator('SensorIndex', 3, ...
    'SensorLocation', [0 -0.9], ...
```

```
    'Height', 2, ...
    'Yaw', -157.449124518333, ...
    'MaxRange', 60, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([320 320],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{4} = ultrasonicDetectionGenerator('SensorIndex', 4, ...
    'MountingLocation', [1.5 0 0.8], ...
    'FieldOfView', [360 35], ...
    'DetectionRange', [0.03 0.15 12], ...
    'Profiles', profiles);
sensors{5} = drivingRadarDataGenerator('SensorIndex', 5, ...
    'MountingLocation', [3.7 0 0.8], ...
    'RangeLimits', [0 75], ...
    'TargetReportFormat', 'Detections', ...
    'FieldOfView', [25 5], ...
    'Profiles', profiles);
sensors{6} = visionDetectionGenerator('SensorIndex', 6, ...
    'SensorLocation', [1.9 0], ...
    'Height', 2, ...
    'MaxRange', 90, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([800 799.999999999999],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
numSensors = 6;
function [scenario, egoVehicle] = createDrivingScenario
% createDrivingScenario Returns the drivingScenario defined in the Designer
% Construct a drivingScenario object.
scenario = drivingScenario('GeographicReference', [-2.307025 -78.11947 0], ...
    'VerticalAxis', 'Y');
% Add all road segments
roadCenters = [35.693673371007 41.60977290687 -0.00023651501808897;
    45.559802662809 172.27751645913 -0.0025050227472985;
    50.364966164054 260.57250844299 -0.0055573552922823;
    54.302526096433 307.85483507256 -0.0077107563859666;
    61.588151364517 449.15999164263 -0.016219010941253;
    70.508854052171 543.99002359338 -0.023744094140969;
    79.006898523718 651.53630150687 -0.033990770226897;
    86.225807043454 763.01908586944 -0.046529823661082;
    100.30768554232 936.74512872215 -0.070040243597852;
    106.3253324 1049.7096273986 -0.08784724364137;
```

```
   115.67991438924 1171.177420254 -0.10929995861997;
   123.81096895222 1288.5538971948 -0.13223792477039;
   128.86093954244 1404.2606756719 -0.15692755451018;
   140.87397740664 1524.1693356968 -0.1848938123399;
   145.60135408195 1591.9966803706 -0.20168057660022;
   143.63813943677 1605.680468207 -0.20510438590259;
   142.3652443002 1610.6114700209 -0.20632652556605;
   140.50977353566 1611.7206876261 -0.20656913525404;
   138.07172714315 1609.0081210226 -0.20583221496657;
   135.05110512267 1602.4737702104 -0.20411576470364;
   128.42728545375 1585.5832843774 -0.19970333420233;
   113.62239810247 1523.3621524318 -0.1841559926118;
   51.421986328948 1294.2043736456 -0.13239526412319;
   29.909932667657 1176.2529006908 -0.10926133115107;
   17.407589960763 1056.6096082081 -0.088131753063372;
   6.6960778973904 946.67489059663 -0.070730955683224;
   -9.8327683750502 777.63726462475 -0.047731963827541;
   -19.409693237493 666.66312574617 -0.035104646004203;
   -28.89762203987 564.15911124548 -0.025183716797986;
   -38.174202058467 465.16035610583 -0.017190454129686;
   -51.243701617595 316.45765437837 -0.0081093154970056;
   -56.193424675794 271.7075168516 -0.0060737957705257;
   -63.089652529937 182.12983850454 -0.0029298979611658;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Soasti-Amazonas');
roadCenters = [-290.93295269834 62.784827845533 -0.0069463826167149;
   -188.99072991804 54.182163783948 -0.0030316656470699;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [-75.625208809638 48.023171184327 -0.00063034586777322;
   35.693673371007 41.60977290687 -0.00023651501808897];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [35.693673371007 41.60977290687 -0.00023651501808897;
   156.43371779338 34.720810202964 -0.0020135204139162];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [306.01535042747 -132.78004778433 -0.008732485265539;
   223.22707896946 -98.567671288529 -0.0046730700310524;
```

```
   145.77780432921 -62.984199715448 -0.0019790062146257;
   35.148634687547 -13.335478564037 -0.0001108815331694];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Gavino Rivadeneira');
roadCenters = [35.148634687547 -13.335478564037 -0.0001108815331694;
   13.325338500533 6.6124487244011 -1.7370602418965e-05];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Gavino Rivadeneira');
roadCenters = [-12.780275372348 -448.38599729265 -0.015879638199628;
   5.0498253749432 -330.2465126303 -0.0086092155698818;
   11.323183821902 -231.08188539781 -0.0042242778801;
   21.678682189488 -136.4508792722 -0.0015062378264741;
   35.148634687547 -13.335478564037 -0.0001108815331694];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Soasti');
roadCenters = [-111.58557350856 -323.32448929601 -0.0092262690265841;
   -99.817550605671 -207.87199767089 -0.0041912532709922;
   -90.151752452812 -94.66420408555 -0.0013443462114835;
   -84.834998769236 -37.330505754843 -0.00067416797188802];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Amazonas');
roadCenters = [-84.834998769236 -37.330505754843 -0.00067416797188802;
   -83.700462115227 -22.74552079618 -0.00059002831725374;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Amazonas');
roadCenters = [-134.95512742174 -80.875399226003 -0.0019439520979732;
   -84.834998769236 -37.330505754843 -0.00067416797188802];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
roadCenters = [-84.834998769236 -37.330505754843 -0.00067416797188802;
   13.325338500533 6.6124487244011 -1.7370602418965e-05];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
roadCenters = [13.325338500533 6.6124487244011 -1.7370602418965e-05;
   26.90650734894 16.298911159009 -7.771785477928e-05;
   35.693673371007 41.60977290687 -0.00023651501808897];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
% Add the ego vehicle
egoVehicle = vehicle(scenario, ...
```

```
    'ClassID', 1, ...
    'Position', [-77.4853823403963 -36.1373201554646 0.01], ...
    'Mesh', driving.scenario.carMesh, ...
    'Name', 'Car');
waypoints = [-77.4853823403963 -36.1373201554646 0.01;
    -60.03 -27.74 0.01;
    -42.71 -20.6 0.01;
    -22.34 -11.88 0.01;
    -8.55 -5.41 0.01;
    4.18 1.05 0.01;
    12.6 4.3 0;
    18.7 0.26 0.01;
    24.21 -5.83 0.01;
    31 -11.9 0;
    33.8 -14.5 0];
speed = [60;60;30;40;40;30;25;30;40;30;20];
waittime = [0;0;0;0;0;0;0;0;0;0;0];
trajectory(egoVehicle, waypoints, speed, waittime);
% Add the non-ego actors
vehicle(scenario, ...
    'ClassID', 2, ...
    'Length', 8.2, ...
    'Width', 2.5, ...
    'Height', 3.5, ...
    'Position', [19.9 12 0], ...
    'Yaw', -140, ...
    'Mesh', driving.scenario.truckMesh, ...
    'Name', 'Truck');
actor(scenario, ...
    'ClassID', 4, ...
    'Length', 0.24, ...
    'Width', 0.45, ...
    'Height', 1.7, ...
    'Position', [-46.6 -20 0], ...
    'RCSPattern', [-8 -8;-8 -8], ...
    'Mesh', driving.scenario.pedestrianMesh, ...
    'Name', 'Pedestrian');
actor(scenario, ...
    'ClassID', 3, ...
    'Length', 1.7, ...
    'Width', 0.45, ...
```

```
    'Height', 1.7, ...
    'Position', [-23.1 -9.6 0], ...
    'Yaw', 30, ...
    'Mesh', driving.scenario.bicycleMesh, ...
    'Name', 'Bicycle');
function output = getDetectorOutput(sensor)
if isa(sensor, 'visionDetectionGenerator')
    output = sensor.DetectorOutput;
elseif isa(sensor, 'lidarPointCloudGenerator')
    output = 'PointCloud';
elseif isa(sensor, 'insSensor')
    output = 'INSMeasurement';
else
    output = 'Objects only';
end
```

SCRIPT FUNCTION ESCENARIO 2

```
function [allData, scenario, sensors] = EscenarioDeSimulacionMaacas2()
%EscenarioDeSimulacionMaacas2 - Returns sensor detections
%    allData = EscenarioDeSimulacionMaacas2 returns sensor detections in a structure
%    with time for an internally defined scenario and sensor suite.
%
%    [allData, scenario, sensors] = EscenarioDeSimulacionMaacas2 optionally returns
%    the drivingScenario and detection generator objects.
% Generated by MATLAB(R) 9.13 (R2022b) and Automated Driving Toolbox 3.6 (R2022b).
% Generated on: 14-Mar-2023 10:15:46
% Create the drivingScenario object and ego car
[scenario, egoVehicle] = createDrivingScenario;
% Create all the sensors
[sensors, numSensors] = createSensors(scenario);
allData = struct('Time', {}, 'ActorPoses', {}, 'ObjectDetections', {}, 'LaneDetections',
{}, 'PointClouds', {}, 'INSMeasurements', {});
running = true;
while running
   % Generate the target poses of all actors relative to the ego vehicle
   poses = targetPoses(egoVehicle);
   time  = scenario.SimulationTime;
   objectDetections = {};
   laneDetections   = [];
   ptClouds = {};
   insMeas = {};
   isValidTime = false(1, numSensors);
   isValidLaneTime = false(1, numSensors);
   isValidPointCloudTime = false(1, numSensors);
   isValidINSTime = false(1, numSensors);
   % Generate detections for each sensor
   for sensorIndex = 1:numSensors
       sensor = sensors{sensorIndex};
       % Generate the ego vehicle lane boundaries
       if isa(sensor, 'visionDetectionGenerator')
           maxLaneDetectionRange = min(500,sensor.MaxRange);
           lanes = laneBoundaries(egoVehicle, 'XDistance', linspace(-
maxLaneDetectionRange, maxLaneDetectionRange, 101));
       end
       type = getDetectorOutput(sensor);
       if strcmp(type, 'Objects only')
           if isa(sensor,'ultrasonicDetectionGenerator')
               [objectDets, isValidTime(sensorIndex)] = sensor(poses, time);
```

```
                numObjects = length(objectDets);
            else
                [objectDets, numObjects, isValidTime(sensorIndex)] = sensor(poses, time);
            end
            objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes only')
            [laneDets, ~, isValidTime(sensorIndex)] = sensor(lanes, time);
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes and objects')
            [objectDets, numObjects, isValidTime(sensorIndex), laneDets, ~,
isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);
            objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes with occlusion')
            [laneDets, ~, isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'PointCloud')
            if sensor.HasRoadsInputPort
                rdmesh = roadMesh(egoVehicle,min(500,sensor.MaxRange));
                [ptCloud, isValidPointCloudTime(sensorIndex)] = sensor(poses, rdmesh,
time);
            else
                [ptCloud, isValidPointCloudTime(sensorIndex)] = sensor(poses, time);
            end
            ptClouds = [ptClouds; ptCloud]; %#ok<AGROW>
        elseif strcmp(type, 'INSMeasurement')
            insMeasCurrent = sensor(actorState, time);
            insMeas = [insMeas; insMeasCurrent]; %#ok<AGROW>
            isValidINSTime(sensorIndex) = true;
        end
    end
    % Aggregate all detections into a structure for later use
    if any(isValidTime) || any(isValidLaneTime) || any(isValidPointCloudTime) ||
any(isValidINSTime)
        allData(end + 1) = struct( ...
            'Time',       scenario.SimulationTime, ...
            'ActorPoses', actorPoses(scenario), ...
            'ObjectDetections', {objectDetections}, ...
            'LaneDetections', {laneDetections}, ...
            'PointClouds',   {ptClouds}, ... %#ok<AGROW>
            'INSMeasurements',   {insMeas}); %#ok<AGROW>
    end
```

```
    % Advance the scenario one time step and exit the loop if the scenario is complete
    running = advance(scenario);
end
% Restart the driving scenario to return the actors to their initial positions.
restart(scenario);
% Release all the sensor objects so they can be used again.
for sensorIndex = 1:numSensors
    release(sensors{sensorIndex});
end
%%%%%%%%%%%%%%%%%%%%
% Helper functions %
%%%%%%%%%%%%%%%%%%%%
% Units used in createSensors and createDrivingScenario
% Distance/Position - meters
% Speed             - meters/second
% Angles            - degrees
% RCS Pattern       - dBsm
function [sensors, numSensors] = createSensors(scenario)
% createSensors Returns all sensor objects to generate detections
% Assign into each sensor the physical and radar profiles for all actors
profiles = actorProfiles(scenario);
sensors{1} = lidarPointCloudGenerator('SensorIndex', 1, ...
    'SensorLocation', [2.8 0], ...
    'MaxRange', 20, ...
    'ActorProfiles', profiles);
sensors{2} = visionDetectionGenerator('SensorIndex', 2, ...
    'SensorLocation', [3.2 0], ...
    'Height', 1.5, ...
    'MaxRange', 25, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([800 799.999999999999],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{3} = visionDetectionGenerator('SensorIndex', 3, ...
    'SensorLocation', [4.1 -0.9], ...
    'Height', 1.5, ...
    'Yaw', -90, ...
    'MaxRange', 25, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([320 320],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{4} = visionDetectionGenerator('SensorIndex', 4, ...
```

```
    'SensorLocation', [4.1 0.9], ...
    'Height', 1.5, ...
    'Yaw', 90, ...
    'MaxRange', 25, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([320 320],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{5} = visionDetectionGenerator('SensorIndex', 5, ...
    'SensorLocation', [0 0], ...
    'Height', 1.5, ...
    'Yaw', -180, ...
    'MaxRange', 25, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([800 799.999999999999],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{6} = drivingRadarDataGenerator('SensorIndex', 6, ...
    'MountingLocation', [5 0 0.5], ...
    'RangeLimits', [0 25], ...
    'TargetReportFormat', 'Detections', ...
    'FieldOfView', [30 5], ...
    'Profiles', profiles);
sensors{7} = drivingRadarDataGenerator('SensorIndex', 7, ...
    'MountingLocation', [0 0 0.5], ...
    'MountingAngles', [-180 0 0], ...
    'RangeLimits', [0 25], ...
    'TargetReportFormat', 'Detections', ...
    'FieldOfView', [30 5], ...
    'Profiles', profiles);
sensors{8} = ultrasonicDetectionGenerator('SensorIndex', 8, ...
    'MountingLocation', [2.8 0 0.2], ...
    'DetectionRange', [0.03 0.15 8], ...
    'Profiles', profiles);
numSensors = 8;
function [scenario, egoVehicle] = createDrivingScenario
% createDrivingScenario Returns the drivingScenario defined in the Designer
% Construct a drivingScenario object.
scenario = drivingScenario('GeographicReference', [-2.307025 -78.11947 0], ...
    'VerticalAxis', 'Y');
% Add all road segments
roadCenters = [35.693673371007 41.60977290687 -0.00023651501808897;
    45.559802662809 172.27751645913 -0.0025050227472985;
```

```
   50.364966164054 260.57250844299 -0.0055573552922823;
   54.302526096433 307.85483507256 -0.0077107563859666;
   61.588151364517 449.15999164263 -0.016219010941253;
   70.508854052171 543.99002359338 -0.023744094140969;
   79.006898523718 651.53630150687 -0.033990770226897;
   86.225807043454 763.01908586944 -0.046529823661082;
   100.30768554232 936.74512872215 -0.070040243597852;
   106.3253324 1049.7096273986 -0.08784724364137;
   115.67991438924 1171.177420254 -0.10929995861997;
   123.81096895222 1288.5538971948 -0.13223792477039;
   128.86093954244 1404.2606756719 -0.15692755451018;
   140.87397740664 1524.1693356968 -0.1848938123399;
   145.60135408195 1591.9966803706 -0.20168057660022;
   143.63813943677 1605.680468207 -0.20510438590259;
   142.3652443002 1610.6114700209 -0.20632652556605;
   140.50977353566 1611.7206876261 -0.20656913525404;
   138.07172714315 1609.0081210226 -0.20583221496657;
   135.05110512267 1602.4737702104 -0.20411576470364;
   128.42728545375 1585.5832843774 -0.19970333420233;
   113.62239810247 1523.3621524318 -0.1841559926118;
   51.421986328948 1294.2043736456 -0.13239526412319;
   29.909932667657 1176.2529006908 -0.10926133115107;
   17.407589960763 1056.6096082081 -0.088131753063372;
   6.6960778973904 946.67489059663 -0.070730955683224;
   -9.8327683750502 777.63726462475 -0.047731963827541;
   -19.409693237493 666.66312574617 -0.035104646004203;
   -28.89762203987 564.15911124548 -0.025183716797986;
   -38.174202058467 465.16035610583 -0.017190454129686;
   -51.243701617595 316.45765437837 -0.0081093154970056;
   -56.193424675794 271.7075168516 -0.0060737957705257;
   -63.089652529937 182.12983850454 -0.0029298979611658;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Soasti-Amazonas');
roadCenters = [-290.93295269834 62.784827845533 -0.0069463826167149;
   -188.99072991804 54.182163783948 -0.0030316656470699;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [-75.625208809638 48.023171184327 -0.00063034586777322;
   35.693673371007 41.60977290687 -0.00023651501808897];
```

```
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [35.693673371007 41.60977290687 -0.00023651501808897;
   156.43371779338 34.720810202964 -0.0020135204139162];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [306.01535042747 -132.78004778433 -0.008732485265539;
   223.22707896946 -98.567671288529 -0.0046730700310524;
   145.77780432921 -62.984199715448 -0.0019790062146257;
   35.148634687547 -13.335478564037 -0.0001108815331694];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Gavino Rivadeneira');
roadCenters = [35.148634687547 -13.335478564037 -0.0001108815331694;
   13.325338500533 6.6124487244011 -1.7370602418965e-05];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Gavino Rivadeneira');
roadCenters = [-12.780275372348 -448.38599729265 -0.015879638199628;
   5.0498253749432 -330.2465126303 -0.0086092155698818;
   11.323183821902 -231.08188539781 -0.0042242778801;
   21.678682189488 -136.4508792722 -0.0015062378264741;
   35.148634687547 -13.335478564037 -0.0001108815331694];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Soasti');
roadCenters = [-111.58557350856 -323.32448929601 -0.0092262690265841;
   -99.817550605671 -207.87199767089 -0.0041912532709922;
   -90.151752452812 -94.66420408555 -0.0013443462114835;
   -84.834998769236 -37.330505754843 -0.00067416797188802];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Amazonas');
roadCenters = [-84.834998769236 -37.330505754843 -0.00067416797188802;
   -83.700462115227 -22.74552079618 -0.00059002831725374;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Amazonas');
roadCenters = [-134.95512742174 -80.875399226003 -0.0019439520979732;
   -84.834998769236 -37.330505754843 -0.00067416797188802];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
roadCenters = [-84.834998769236 -37.330505754843 -0.00067416797188802;
   13.325338500533 6.6124487244011 -1.7370602418965e-05];
laneSpecification = lanespec([1 1]);
```

```
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
roadCenters = [13.325338500533 6.6124487244011 -1.7370602418965e-05;
   26.90650734894 16.298911159009 -7.771785477928e-05;
   35.693673371007 41.60977290687 -0.00023651501808897];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
% Add the ego vehicle
egoVehicle = vehicle(scenario, ...
   'ClassID', 1, ...
   'Length', 6, ...
   'Position', [-80.0380653357798 -37.380043230025 0], ...
   'Mesh', driving.scenario.carMesh, ...
   'Name', 'Car');
waypoints = [-80.0380653357798 -37.380043230025 0;
   -59.8 -28.4 0;
   -32.9 -16.1 0;
   -15.4 -8.1 0;
   1.5 -0.5 0;
   7.7 1.3 0;
   13.9 3.9 0;
   18.9 6.9 0;
   25.3 11.5 0;
   30.5 17.9 0;
   35.5 27.5 0;
   36.2 33.9 0;
   37.3 43.3 0;
   38.5 54.3 0;
   39.2 63.7 0];
speed = [60;60;60;60;60;60;60;60;60;60;15;15;15;15;15];
waittime = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
trajectory(egoVehicle, waypoints, speed, waittime);
% Add the non-ego actors
truck = vehicle(scenario, ...
   'ClassID', 2, ...
   'Length', 8.2, ...
   'Width', 2.5, ...
   'Height', 3.5, ...
   'Position', [35.1093097258125 58.2214206195563 0], ...
   'Mesh', driving.scenario.truckMesh, ...
   'Name', 'Truck');
waypoints = [35.1093097258125 58.2214206195563 0;
```

```
    31.1 28.2 0;
    28.1 21.7 0;
    22.2 15 0;
    16.2 10.6 0;
    11.3 7.6 0;
    4.9 3.5 0;
    -2.4 1.2 0;
    -11.9 -2.6 0;
    -22.8 -7.5 0;
    -33.3 -12.4 0;
    -51.7 -20 0;
    -66.6 -26.7 0];
speed = [30;30;30;30;30;30;30;30;30;30;30;30;30];
trajectory(truck, waypoints, speed);
actor(scenario, ...
    'ClassID', 4, ...
    'Length', 0.24, ...
    'Width', 0.45, ...
    'Height', 1.7, ...
    'Position', [-46.6 -20 0], ...
    'RCSPattern', [-8 -8;-8 -8], ...
    'Mesh', driving.scenario.pedestrianMesh, ...
    'Name', 'Pedestrian');
actor(scenario, ...
    'ClassID', 3, ...
    'Length', 1.7, ...
    'Width', 0.45, ...
    'Height', 1.7, ...
    'Position', [-23.1 -9.6 0], ...
    'Yaw', 30, ...
    'Mesh', driving.scenario.bicycleMesh, ...
    'Name', 'Bicycle');
car1 = vehicle(scenario, ...
    'ClassID', 1, ...
    'Position', [109.8 39.1 0], ...
    'Mesh', driving.scenario.carMesh, ...
    'Name', 'Car1');
waypoints = [109.8 39.1 0;
    95.1 40.2 0;
    74.9 41.3 0;
    58.4 42 0;
```

```
    40.8 43.3 0;
    27.7 44 0;
    7.9 45.1 0;
    -8.5 46.2 0;
    -19 46.7 0];
speed = [30;30;30;30;30;30;30;30;30];
trajectory(car1, waypoints, speed);
function output = getDetectorOutput(sensor)
if isa(sensor, 'visionDetectionGenerator')
    output = sensor.DetectorOutput;
elseif isa(sensor, 'lidarPointCloudGenerator')
    output = 'PointCloud';
elseif isa(sensor, 'insSensor')
    output = 'INSMeasurement';
else
    output = 'Objects only';
end
```

SCRIPT FUNCTION ESCENARIO 3

```
function [allData, scenario, sensors] = EscenarioDeSimulacionMaacas3()
%EscenarioDeSimulacionMaacas3 - Returns sensor detections
%    allData = EscenarioDeSimulacionMaacas3 returns sensor detections in a structure
%    with time for an internally defined scenario and sensor suite.
%
%    [allData, scenario, sensors] = EscenarioDeSimulacionMaacas3 optionally returns
%    the drivingScenario and detection generator objects.
% Generated by MATLAB(R) 9.13 (R2022b) and Automated Driving Toolbox 3.6 (R2022b).
% Generated on: 14-Mar-2023 10:46:58
% Create the drivingScenario object and ego car
[scenario, egoVehicle] = createDrivingScenario;
% Create all the sensors
[sensors, numSensors] = createSensors(scenario);
allData = struct('Time', {}, 'ActorPoses', {}, 'ObjectDetections', {}, 'LaneDetections',
{}, 'PointClouds', {}, 'INSMeasurements', {});
running = true;
while running
   % Generate the target poses of all actors relative to the ego vehicle
   poses = targetPoses(egoVehicle);
   time  = scenario.SimulationTime;
   objectDetections = {};
   laneDetections   = [];
   ptClouds = {};
   insMeas = {};
   isValidTime = false(1, numSensors);
   isValidLaneTime = false(1, numSensors);
   isValidPointCloudTime = false(1, numSensors);
   isValidINSTime = false(1, numSensors);
   % Generate detections for each sensor
   for sensorIndex = 1:numSensors
       sensor = sensors{sensorIndex};
       % Generate the ego vehicle lane boundaries
       if isa(sensor, 'visionDetectionGenerator')
           maxLaneDetectionRange = min(500,sensor.MaxRange);
           lanes = laneBoundaries(egoVehicle, 'XDistance', linspace(-
maxLaneDetectionRange, maxLaneDetectionRange, 101));
       end
       type = getDetectorOutput(sensor);
       if strcmp(type, 'Objects only')
           if isa(sensor,'ultrasonicDetectionGenerator')
               [objectDets, isValidTime(sensorIndex)] = sensor(poses, time);
```

```
                numObjects = length(objectDets);
            else
                [objectDets, numObjects, isValidTime(sensorIndex)] = sensor(poses, time);
            end
            objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes only')
            [laneDets, ~, isValidTime(sensorIndex)] = sensor(lanes, time);
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes and objects')
            [objectDets, numObjects, isValidTime(sensorIndex), laneDets, ~,
isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);
            objectDetections = [objectDetections; objectDets(1:numObjects)]; %#ok<AGROW>
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'Lanes with occlusion')
            [laneDets, ~, isValidLaneTime(sensorIndex)] = sensor(poses, lanes, time);
            laneDetections   = [laneDetections laneDets]; %#ok<AGROW>
        elseif strcmp(type, 'PointCloud')
            if sensor.HasRoadsInputPort
                rdmesh = roadMesh(egoVehicle,min(500,sensor.MaxRange));
                [ptCloud, isValidPointCloudTime(sensorIndex)] = sensor(poses, rdmesh,
time);
            else
                [ptCloud, isValidPointCloudTime(sensorIndex)] = sensor(poses, time);
            end
            ptClouds = [ptClouds; ptCloud]; %#ok<AGROW>
        elseif strcmp(type, 'INSMeasurement')
            insMeasCurrent = sensor(actorState, time);
            insMeas = [insMeas; insMeasCurrent]; %#ok<AGROW>
            isValidINSTime(sensorIndex) = true;
        end
    end
    % Aggregate all detections into a structure for later use
    if any(isValidTime) || any(isValidLaneTime) || any(isValidPointCloudTime) ||
any(isValidINSTime)
        allData(end + 1) = struct( ...
            'Time',       scenario.SimulationTime, ...
            'ActorPoses', actorPoses(scenario), ...
            'ObjectDetections', {objectDetections}, ...
            'LaneDetections', {laneDetections}, ...
            'PointClouds',   {ptClouds}, ... %#ok<AGROW>
            'INSMeasurements',   {insMeas}); %#ok<AGROW>
    end
```

```
    % Advance the scenario one time step and exit the loop if the scenario is complete
    running = advance(scenario);
end
% Restart the driving scenario to return the actors to their initial positions.
restart(scenario);
% Release all the sensor objects so they can be used again.
for sensorIndex = 1:numSensors
    release(sensors{sensorIndex});
end
%%%%%%%%%%%%%%%%%%%%
% Helper functions %
%%%%%%%%%%%%%%%%%%%%
% Units used in createSensors and createDrivingScenario
% Distance/Position - meters
% Speed             - meters/second
% Angles            - degrees
% RCS Pattern       - dBsm
function [sensors, numSensors] = createSensors(scenario)
% createSensors Returns all sensor objects to generate detections
% Assign into each sensor the physical and radar profiles for all actors
profiles = actorProfiles(scenario);
sensors{1} = drivingRadarDataGenerator('SensorIndex', 1, ...
    'MountingLocation', [3.7 0 0.2], ...
    'RangeLimits', [0 60], ...
    'TargetReportFormat', 'Detections', ...
    'Profiles', profiles);
sensors{2} = visionDetectionGenerator('SensorIndex', 2, ...
    'SensorLocation', [0 0], ...
    'Yaw', -180, ...
    'MaxRange', 50, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([350 800],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{3} = visionDetectionGenerator('SensorIndex', 3, ...
    'SensorLocation', [1.47 0.9], ...
    'Yaw', 90, ...
    'MaxRange', 50, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([320 320],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{4} = visionDetectionGenerator('SensorIndex', 4, ...
```

```
    'SensorLocation', [1.9 0], ...
    'MaxRange', 50, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([350 800],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{5} = drivingRadarDataGenerator('SensorIndex', 5, ...
    'MountingLocation', [1.46 0.9 0.2], ...
    'MountingAngles', [90 0 0], ...
    'RangeLimits', [0 50], ...
    'TargetReportFormat', 'Detections', ...
    'Profiles', profiles);
sensors{6} = drivingRadarDataGenerator('SensorIndex', 6, ...
    'MountingLocation', [1.51 -0.9 0.2], ...
    'MountingAngles', [-90 0 0], ...
    'RangeLimits', [0 50], ...
    'TargetReportFormat', 'Detections', ...
    'Profiles', profiles);
sensors{7} = visionDetectionGenerator('SensorIndex', 7, ...
    'SensorLocation', [1.52 -0.9], ...
    'Yaw', -90, ...
    'MaxRange', 50, ...
    'DetectorOutput', 'Lanes and objects', ...
    'Intrinsics', cameraIntrinsics([320 320],[320 240],[480 640]), ...
    'ActorProfiles', profiles);
sensors{8} = lidarPointCloudGenerator('SensorIndex', 8, ...
    'SensorLocation', [1.5 0], ...
    'Height', 2, ...
    'MaxRange', 40, ...
    'ActorProfiles', profiles);
numSensors = 8;
function [scenario, egoVehicle] = createDrivingScenario
% createDrivingScenario Returns the drivingScenario defined in the Designer
% Construct a drivingScenario object.
scenario = drivingScenario('GeographicReference', [-2.307025 -78.11947 0], ...
    'VerticalAxis', 'Y');
% Add all road segments
roadCenters = [35.693673371007 41.60977290687 -0.00023651501808897;
    45.559802662809 172.27751645913 -0.0025050227472985;
    50.364966164054 260.57250844299 -0.0055573552922823;
    54.302526096433 307.85483507256 -0.0077107563859666;
    61.588151364517 449.15999164263 -0.016219010941253;
```

```
   70.508854052171 543.99002359338 -0.023744094140969;
   79.006898523718 651.53630150687 -0.033990770226897;
   86.225807043454 763.01908586944 -0.046529823661082;
   100.30768554232 936.74512872215 -0.070040243597852;
   106.3253324 1049.7096273986 -0.08784724364137;
   115.67991438924 1171.177420254 -0.10929995861997;
   123.81096895222 1288.5538971948 -0.13223792477039;
   128.86093954244 1404.2606756719 -0.15692755451018;
   140.87397740664 1524.1693356968 -0.1848938123399;
   145.60135408195 1591.9966803706 -0.20168057660022;
   143.63813943677 1605.680468207 -0.20510438590259;
   142.3652443002 1610.6114700209 -0.20632652556605;
   140.50977353566 1611.7206876261 -0.20656913525404;
   138.07172714315 1609.0081210226 -0.20583221496657;
   135.05110512267 1602.4737702104 -0.20411576470364;
   128.42728545375 1585.5832843774 -0.19970333420233;
   113.62239810247 1523.3621524318 -0.1841559926118;
   51.421986328948 1294.2043736456 -0.13239526412319;
   29.909932667657 1176.2529006908 -0.10926133115107;
   17.407589960763 1056.6096082081 -0.088131753063372;
   6.6960778973904 946.67489059663 -0.070730955683224;
   -9.8327683750502 777.63726462475 -0.047731963827541;
   -19.409693237493 666.66312574617 -0.035104646004203;
   -28.89762203987 564.15911124548 -0.025183716797986;
   -38.174202058467 465.16035610583 -0.017190454129686;
   -51.243701617595 316.45765437837 -0.0081093154970056;
   -56.193424675794 271.7075168516 -0.0060737957705257;
   -63.089652529937 182.12983850454 -0.0029298979611658;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Soasti-Amazonas');
roadCenters = [-290.93295269834 62.784827845533 -0.0069463826167149;
   -188.99072991804 54.182163783948 -0.0030316656470699;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [-75.625208809638 48.023171184327 -0.00063034586777322;
   35.693673371007 41.60977290687 -0.00023651501808897];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [35.693673371007 41.60977290687 -0.00023651501808897;
```

```
   156.43371779338 34.720810202964 -0.0020135204139162];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Tarqui');
roadCenters = [306.01535042747 -132.78004778433 -0.008732485265539;
   223.22707896946 -98.567671288529 -0.0046730700310524;
   145.77780432921 -62.984199715448 -0.0019790062146257;
   35.148634687547 -13.335478564037 -0.0001108815331694];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Gavino Rivadeneira');
roadCenters = [35.148634687547 -13.335478564037 -0.0001108815331694;
   13.325338500533 6.6124487244011 -1.7370602418965e-05];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Gavino Rivadeneira');
roadCenters = [-12.780275372348 -448.38599729265 -0.015879638199628;
   5.0498253749432 -330.2465126303 -0.0086092155698818;
   11.323183821902 -231.08188539781 -0.0042242778801;
   21.678682189488 -136.4508792722 -0.0015062378264741;
   35.148634687547 -13.335478564037 -0.0001108815331694];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Soasti');
roadCenters = [-111.58557350856 -323.32448929601 -0.0092262690265841;
   -99.817550605671 -207.87199767089 -0.0041912532709922;
   -90.151752452812 -94.66420408555 -0.0013443462114835;
   -84.834998769236 -37.330505754843 -0.00067416797188802];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Amazonas');
roadCenters = [-84.834998769236 -37.330505754843 -0.00067416797188802;
   -83.700462115227 -22.74552079618 -0.00059002831725374;
   -75.625208809638 48.023171184327 -0.00063034586777322];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', 'Amazonas');
roadCenters = [-134.95512742174 -80.875399226003 -0.0019439520979732;
   -84.834998769236 -37.330505754843 -0.00067416797188802];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
roadCenters = [-84.834998769236 -37.330505754843 -0.00067416797188802;
   13.325338500533 6.6124487244011 -1.7370602418965e-05];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
roadCenters = [13.325338500533 6.6124487244011 -1.7370602418965e-05;
   26.90650734894 16.298911159009 -7.771785477928e-05;
```

```
   35.693673371007 41.60977290687 -0.00023651501808897];
laneSpecification = lanespec([1 1]);
road(scenario, roadCenters, 'Lanes', laneSpecification, 'Name', '29 de Mayo');
% Add the ego vehicle
egoVehicle = vehicle(scenario, ...
   'ClassID', 1, ...
   'Position', [-77.4853823403963 -36.1373201554646 0.01], ...
   'Mesh', driving.scenario.carMesh, ...
   'Name', 'Car');
waypoints = [-77.4853823403963 -36.1373201554646 0.01;
   -62.9 -29.4 0;
   -48.9 -23.2 0;
   -40 -19.4 0;
   -29.2 -14.6 0;
   -15.8 -8.4 0;
   -6.2 -3.8 0;
   2.9 0.5 0;
   8.2 3.7 0;
   12.5 7.7 0;
   17.3 10.6 0;
   25.8 14.1 0;
   31 20.9 0;
   35.8 30.6 0;
   36.64 35.77 0.01];
speed = [60;60;60;60;60;40;30;15;15;15;20;30;40;60;60];
waittime = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
trajectory(egoVehicle, waypoints, speed, waittime);
% Add the non-ego actors
vehicle(scenario, ...
   'ClassID', 2, ...
   'Length', 8.2, ...
   'Width', 2.5, ...
   'Height', 3.5, ...
   'Position', [21.9 2.1 0], ...
   'Yaw', 150, ...
   'Mesh', driving.scenario.truckMesh, ...
   'Name', 'Truck');
actor(scenario, ...
   'ClassID', 4, ...
   'Length', 0.24, ...
   'Width', 0.45, ...
```

```
    'Height', 1.7, ...
    'Position', [-46.6 -20 0], ...
    'RCSPattern', [-8 -8;-8 -8], ...
    'Mesh', driving.scenario.pedestrianMesh, ...
    'Name', 'Pedestrian');
actor(scenario, ...
    'ClassID', 3, ...
    'Length', 1.7, ...
    'Width', 0.45, ...
    'Height', 1.7, ...
    'Position', [-23.1 -9.6 0], ...
    'Yaw', 30, ...
    'Mesh', driving.scenario.bicycleMesh, ...
    'Name', 'Bicycle');
function output = getDetectorOutput(sensor)
if isa(sensor, 'visionDetectionGenerator')
    output = sensor.DetectorOutput;
elseif isa(sensor, 'lidarPointCloudGenerator')
    output = 'PointCloud';
elseif isa(sensor, 'insSensor')
    output = 'INSMeasurement';
else
    output = 'Objects only';
end
```

SCRIPT POSICIÓN

```
% Datos Seleccionados para la posicion
for i=1:length(data)
        Positionx(i)=data(i).ActorPoses(1).Position(1);
        Positiony(i)=data(i).ActorPoses(1).Position(2);
        time(i)=data(i).Time;
end
%plot Posicion
figure;
plot(time,Positionx,'--ro' )
hold on
plot(time,Positiony,'--bsquare' )
hold off
```

SCRIPT VELOCIDAD Y ACELERACIÓN

```
% Selecciona datos para velocidad
for i=1:length(data)
        velocityx(i)=data(i).ActorPoses(1).Velocity(1);
        velocityy(i)=data(i).ActorPoses(1).Velocity(2);
        time(i)=data(i).Time;
end
% Deriva para encontrar la aceleracion
accx=[diff(velocityx) 0];
accy=[diff(velocityy) 0];
% plot Velocidad
figure;
plot(time,velocityx)
hold on
plot(time,velocityy)
hold off
%  plot Aceleracion
figure;
plot(time,accx)
title('accelerrazione lungo x e y')
xlabel('time')
ylabel('m/s^2')
hold on
plot(time,accy)
legend('acc X', 'acc Y')
hold off
% plot Aceleracion en X vs en Aceleracion en Y
figure;
plot(accx, accy)
title('GG plot')
```

SCRIPT MEDIDAS Y NÚMEROS DETECCIONES SENSORES

```
% Selecciona datos para deteccion de medidas
clc
m=0;
Detection=[];
for i=1:length(data)
   Detection(i,1)=m;
    for n=1:length(data(i).ObjectDetections)
       for k=1:length(data(i).ObjectDetections{n, 1}.Measurement)
       %Detection(i)=data(i).ObjectDetections{n, 1}.Measurement(k)
       Detection(i,k+1)=data(i).ObjectDetections{n, 1}.Measurement(k);

       end
    end

    m=m+0.1;
end
%Plot las medidas de detección
[p,q]=size(Detection)
for b=1:p
for c=2:q
   plot(Detection(b,1),Detection(b,c),'ro')
   %bar(Detection(b,1),Detection(b,c))
   hold on
end
end
% Crea una matriz con los valores detectados
NumDetection=[];
for i=1:length(data)
       NumDetection(i)=length(data(i).ObjectDetections);
       time(i)=data(i).Time
end
%time=0:0.1:(length(data)-1)/10;
%Plot de Nuemero de Detecciones
bar(time,NumDetection)
```