ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA INGENIERÍA AUTOMOTRIZ

DESARROLLO DE SISTEMA PARA LA DETECCIÓN DE BACHES DE LA VÍA RIOBAMBA – EL ARENAL MEDIANTE LA APLICACIÓN DE UNA CÁMARA ESTEREOSCÓPICA Y VISIÓN POR COMPUTADORA PARA REDUCIR LA PROBABILIDAD DE CAÍDAS BRUSCAS O MANIOBRAS DE ESQUIVA PELIGROSAS.

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO AUTOMOTRIZ

AUTOR:

EDWIN ALEXIS GUNSHA ALLAUCA

Riobamba – Ecuador

2022

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA INGENIERÍA AUTOMOTRIZ

DESARROLLO DE SISTEMA PARA LA DETECCIÓN DE BACHES DE LA VÍA RIOBAMBA – EL ARENAL MEDIANTE LA APLICACIÓN DE UNA CÁMARA ESTEREOSCÓPICA Y VISIÓN POR COMPUTADORA PARA REDUCIR LA PROBABILIDAD DE CAÍDAS BRUSCAS O MANIOBRAS DE ESQUIVA PELIGROSAS.

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO AUTOMOTRIZ

AUTOR: EDWIN ALEXIS GUNSHA ALLAUCA

DIRECTOR: Dr. MARIO EFRAÍN AUDELO GUEVARA

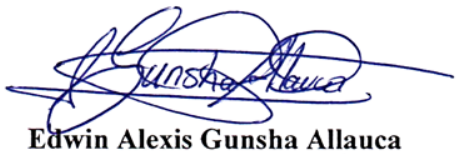
Riobamba – Ecuador

2022

Yo, Edwin Alexis Gunsha Allauca, declaro que el presente Trabajo de Integración Curricular es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 13 de diciembre de 2022



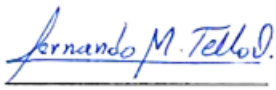
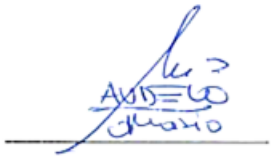
Edwin Alexis Gunsha Allauca
0603954777

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA INGENIERÍA AUTOMOTRIZ

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular, Tipo: Proyecto Técnico, **DESARROLLO DE SISTEMA PARA LA DETECCIÓN DE BACHES DE LA VÍA RIOBAMBA – EL ARENAL MEDIANTE LA APLICACIÓN DE UNA CÁMARA ESTEREOSCÓPICA Y VISIÓN POR COMPUTADORA PARA REDUCIR LA PROBABILIDAD DE CAÍDAS BRUSCAS O MANIOBRAS DE ESQUIVA PELIGROSAS**, realizado por el señor: **EDWIN ALEXIS GUNSHA ALLAUCA**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Fernando Mauricio Tello Oquendo Ph.D **PRESIDENTE DEL TRIBUNAL**		2022 - 12 - 13
Dr. Mario Efraín Audelo Guevara **DIRECTOR DE TRABAJO DE INTEGRACIÓN CURRICULAR**		2022 - 12 - 13
Ing. Javier José Gavilanes Carrión **ASESOR DE TRABAJO DE INTEGRACIÓN CURRICULAR**		2022 - 12 - 13

ÍNDICE DE CONTENIDO

CAPÍTULO IV

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

RESUMEN

El presente proyecto técnico tuvo como objetivo el desarrollo de un sistema para la detección de baches de la vía Riobamba – El Arenal, aplicando visión por computadora y una cámara estereoscópica, para reducir la probabilidad de caídas bruscas o maniobras de esquiva peligrosas. Con la información obtenida de la revisión bibliográfica realizada sobre soluciones para la detección de baches, se escogió al modelo de detección de objetos YOLOR-P6 y al dispositivo Luxonis OAK-D Pro de enfoque fijo para desarrollar dicho sistema. La base de datos fue elaborada mediante la captura de imágenes a color de 1920x1080 píxeles recorriendo 41,27% de la ruta de análisis, sobre las que se etiquetaron las clases bache y parche, donde esta última se incluyó como filtro para evitar detecciones erróneas; obteniéndose 630 y 70 imágenes para el entrenamiento y evaluación de la red neuronal respectivamente. Este entrenamiento y evaluación se ejecutó en una tarjeta gráfica NVIDIA GeForce RTX 2070, y con los pesos resultantes se ejecutaron inferencias utilizando dicha tarjeta gráfica y la unidad de procesamiento de visión Myriad-X, para evaluar su rendimiento. Además, se analizaron los registros de los acelerómetros en los ejes X, Y y Z, generados en una conducción con y sin el detector de baches. De esta manera, la red neuronal entrenada, obtuvo una precisión de 49,39% y exhaustividad de 80,09% al evaluarla para ambas clases, mientras que en su validación para la clase bache, se obtuvo una precisión de 87,84% y exhaustividad de 67,24% con el crossover Renault Stepway, y 83,73% y 69,87% con el SUV Toyota Corolla Cross. Se concluye que la utilización del sistema permite reducir la intensidad de la caída en un bache, frenada y maniobras laterales de esquiva. Se recomienda robustecer la base de datos con la inclusión de nuevas rutas de estudio y condiciones climáticas.

Palabras clave: <SISTEMA DE DETECCIÓN DE BACHES> <RED NEURONAL> <VISIÓN POR COMPUTADORA> <VISIÓN ESTÉREO ACTIVA> <ACELERÓMETRO>.

SUMMARY

This technical project's objective was to develop a system for detecting potholes on the Riobamba - El Arenal road, applying computer vision and a stereoscopic camera, to reduce the probability of sudden falls or dangerous avoidance maneuvers. With the information obtained from the bibliographical review carried out on pothole detection solutions, the YOLOR-P6 object detection model and the fixed-focus Luxonis OAK-D Pro device was chosen to develop said system. The database was created by capturing color images of 1920x1080 pixels covering 41.27% of the analysis route, on which the pothole and patch classes were labeled, where the latter was included as a filter to avoid erroneous detections, obtaining 630 and 70 images for the training and evaluation of the neural network respectively. This training and evaluation were run on an NVIDIA GeForce RTX 2070 graphics card, and with the resulting weights inferences were run using said graphics card and the Myriad-X mink processing unit, to evaluate its performance. In addition, the records of the accelerometers in the X, Y, and Z axes, generated while driving with and without the pothole detector, were analyzed. In this way, the trained neural network obtained a precision of 49.39% and completeness of 80.09% when evaluating it for both classes, while in its validation for the bump class, a precision of 87.84% was obtained and 67.24% completeness with the Renault Stepway crossover, and 83.73% and 69.87% with the Toyota Corolla Cross SUV. It is concluded that the use of the system allows for reducing the intensity of the fall in a pothole, braking, and lateral dodging maneuvers. It is recommended to strengthen the database with the inclusion of new study routes and climatic conditions.

Keywords: <POTHOLE> <NEURAL NETWORK> <COMPUTER VISION> <ACTIVE STEREO VISION> <ACCELEROMETER>

Licda. Mónica A. Castelo R. Mgs.

C.I: 060453982-5

INTRODUCCIÓN

Los baches existentes en las vías son uno de los principales factores que afectan al desarrollo de una conducción normal, pues en función de su gravedad ocasionarán un menor o mayor daño, ya sea a los componentes automotrices más próximos a la calzada o a los propios ocupantes de un automotor (Reza Kashyzadeh y Amiri, 2022, pp.593-604). Si bien una de las alternativas para solucionar tales deformaciones sería la ejecución de mantenimientos oportunos de la capa asfáltica, esta estrategia no resulta muy efectiva en Ecuador, debido a su dependencia en la gestión de recursos que realizan los organismos responsables de este tipo de actividades.

Por otro lado, el auge de los sistemas de asistencia a la conducción que se encuentran enfocados en reducir los accidentes de tránsito, mediante la incorporación de estrategias que permitan evitar la sucesión de estos o en dado caso minimizar los daños que puedan causar, ha motivado que ciertas regiones del mundo como Europa, incluyan en su legislación la obligatoriedad de incorporar este tipo de sistemas en vehículos que se comercialicen a partir del segundo semestre de 2022 (Hoyo, 2021).

En este sentido, se pretende tratar dicha problemática desde una perspectiva proactiva, mediante el desarrollo de un sistema que permita la detección temprana de un bache y alerte al conductor de forma sonora, para que este pueda tomar las medidas correspondientes a la hora de afrontar dicha deformación de una manera segura. Mismo que según el estándar SAE J3016_202104, pertenece al nivel 0 de conducción autónoma, ya que brinda una alerta, pero no interviene directamente en las maniobras del conductor.

Para la identificación de baches se ha optado por utilizar la visión por computadora en conjunto con una cámara estereoscópica, capaz de otorgar datos de profundidad, ya que con dicho dispositivo se pretende minimizar la detección errónea de baches en comparación con los datos provistos por las cámaras convencionales (Manzanares, 2019, pp.3-55). Además, debe indicarse que el procesamiento de datos se llevará a cabo en cuatro etapas principales: recolección de imágenes, entrenamiento de red neuronal, evaluación de entrenamiento, y evaluación en prueba de ruta.

De esta manera, en el capítulo I relacionado con los antecedentes, problemática, justificación y objetivos del proyecto, se abordarán temáticas relacionadas con la gestión vial existente en Ecuador, situación de la vía que conecta al cantón Riobamba con la Reserva de Producción Faunística Chimborazo, efectos ocasionados por los baches sobre conductores y pasajeros de los vehículos, así como sobre los componentes mecánicos; además de especificar los objetivos a cumplir para alcanzar la finalidad principal del proyecto.

En el capítulo II, relacionado con las referencias teóricas se presentan las definiciones elementales relacionadas con las deformaciones superficiales en las vías asfaltadas, redes neuronales, visón por computadora y visión estéreo, de manera que se facilite el entendimiento de la terminología utilizada en el presente proyecto técnico, así como de los trabajos similares, cuya revisión se presenta al final de este capítulo, donde se destacan dos enfoques adoptados para este tipo de soluciones, como son el monitoreo de vibraciones mediante el uso de acelerómetros, sensores piezoeléctricos y ultrasónicos, y la visión por computadora en dos y tres dimensiones.

En el capítulo III enfocado en la metodología utilizada para la elaboración del sistema de detección de baches, se especifica la ruta de análisis con sus respectivas zonas de recolección de datos, vehículos de prueba, modelo de detección de objetos seleccionado para aplicaciones en tiempo real, así como la cámara estereoscópica con sus principales características. De igual manera, se indican los procedimientos realizados para la elaboración de la base de datos, entrenamiento y evaluación de la red neuronal, ejecución del modelo de detección de baches en tarjeta gráfica y en unidad de procesamiento de visión, validación del detector de baches en prueba de ruta, registro e interpretación de datos de acelerómetros en los ejes X, Y y Z durante conducción con y sin el sistema de detección de baches.

En dicho capítulo, a su vez se van explicando las razones que motivaron el establecimiento de ciertas configuraciones diferentes a las establecidas por defecto, especialmente en la ejecución de los algoritmos utilizados, que por lo general se hallan asociadas a problemas en la gestión de la memoria y compatibilidad.

En el capítulo IV relacionado con los resultados, se muestran las especificaciones de la base de datos, métricas generadas durante el entrenamiento y evaluación de la red neuronal, funcionamiento del modelo de detección de baches ejecutado en tarjeta gráfica y unidad de procesamiento de visión, precisión y exhaustividad obtenidas en la validación del sistema en la prueba de ruta, y la comparación de las aceleraciones vertical, lateral y longitudinal entre una conducción realizada con y sin el detector de baches.

Finalmente, en la última parte del documento se aportan las conclusiones y recomendaciones a las que se llegó, en base a los resultados obtenidos e inconvenientes que se presentaron durante la elaboración del sistema de detección de baches, de manera que pueda valorarse su utilidad para la conducción en carretera.

CAPÍTULO I

1. DIAGNÓSTICO DEL PROBLEMA

1.1. Antecedentes

En la Norma Ecuatoriana Vial NEVI-12 – MTOP relacionada con los estudios y diseños viales, se específica que, si bien el tiempo de vida útil de una carretera resulta complicado de establecer debido a diferentes factores como las condiciones climáticas, volumen de tránsito, material y proceso de fabricación, frecuencia y calidad de mantenimiento, entre otros, el pavimento debería presentar entre 10 y 30 años de utilidad bajo condiciones de un mantenimiento adecuado.

Sin embargo, en el caso de las vías estatales, especialmente aquellas cuyo mantenimiento no presentan concesión alguna a empresas privadas, sino que esta tarea recae directamente sobre el Ministerio de Transporte y Obras Públicas (MTOP), se ha visto un acelerado deterioro que con dificultad permite alcanzar 10 años bajo condiciones de transitabilidad óptimas, especialmente debido a un escaso mantenimiento sobre estas, además, una retrasada planificación relacionada con la renovación de las vías estatales, ha ocasionada que varías de estas a pesar de haber cumplido con su vida útil no puedan experimentar la sustitución de su capa asfáltica, sino por el contrario se ha optado por soluciones rápidas para tapar los baches generados, que duran escasos meses en el mejor de los casos.

Por otro lado, algunas de estas vías no se consideran como parte de la red estatal ocasionando inconvenientes al momento de su intervención, especialmente por los problemas de concesión generados con los gobiernos autónomos descentralizados provinciales, como es el caso de la carretera que enlaza Riobamba con la Reserva de Producción Faunística Chimborazo, sector comúnmente conocido como El Arenal (El Comercio, 2020), donde la confusión de responsabilidades entre el MTOP y el Gobierno Autónomo Descentralizado de la Provincia de Chimborazo, ha ocasionado que esta vía se vea sometida únicamente a trabajos de bacheo durante más de una década. Situación que se ha replicado para 2022, pues de acuerdo con el MTOP, gracias al trabajo conjunto con la Unión Cementera y el Gobierno Autónomo Descentralizado Parroquial de San Juan, se ha podido efectuar solamente el bacheo de esta vía al finalizar el primer trimestre del presente año (Molina, 2022), y no se ha optado por la sustitución de la capa asfáltica.

1.2. Planteamiento del problema

Las grietas, discontinuidad de pavimento, ondulaciones, depresiones, elevaciones, exudación de ligante y baches son algunas de las evidencias visuales de una carretera que ha llegado al fin de su vida utilidad (Manzanares, 2019, pp.3-55), precisamente los baches son una constante en la carretera

Riobamba – El Arenal, pues a pesar de las intervenciones que tratan de subsanar estas formaciones, después de cuatro o cinco meses vuelven a surgir, incluso con un tamaño superior a su predecesor (El Universo, 2007), es así que en las cercanías del ingreso a la Reserva de Producción Faunística Chimborazo, se tienen baches que cubren alrededor del 60% del ancho disponible para la circulación.

Los baches en función de su área y profundidad pueden generar desde simples molestias de confortabilidad durante la conducción, hasta constituirse en causales de accidentes de tránsito, ya sea por la ejecución de maniobras peligrosas para evitarlos ante una detección tardía de estos, o por la ruptura o deformación de algún componente automotriz que pueda inestabilizar al vehículo especialmente si la velocidad de circulación resulta elevada.

En este sentido los principales elementos afectados suelen ser los neumáticos, que pueden presentar pinchazos; aros, sean de acero o aleación, siendo propensos a deformaciones o rupturas respectivamente; amortiguadores y resortes, reduciendo su vida útil y en el peor de los casos ocasionando rupturas; terminales de dirección, que además de poder fracturarse transmiten esfuerzos adicionales a la caja de dirección, acortando su vida útil; de igual manera cuando las características de los baches resultan extremas se corre el riesgo de incluso propiciar golpes contra los componentes que se hallan en la zona baja de la carrocería, especialmente cuando de vehículos con poco despeje del suelo se trata (Rojas, 2018, pp.16-20).

1.3. Justificación

La presencia de baches en las carreteras puede ocasionar graves inconvenientes a corto y largo plazo en materia de componentes automotrices, especialmente aquellos que se hallan más próximos al contacto directo con la calzada; por otro lado el rango de velocidad al que circulan los vehículos resulta elevado, pues bajo la propia norma de tránsito vigente, los vehículos livianos pueden alcanzar los 100 km/h y los pesados, 70 km/h (Panavial, 2022), por lo que la ejecución de maniobras de esquiva para evitar precisamente inconvenientes mecánicos en el automotor, resulta peligrosa si se combina una velocidad de circulación en el límite o superior, con una detección tardía del bache, sea por distracción, condiciones de escasa visibilidad, o confusión con una mancha en el asfalto.

Es así como Mutlu Aydin y Topal (2016, pp.192-201) han detectado que la invasión de los carriles aledaños por parte de los conductores a causa de una deformación en la calzada depende en gran medida de su percepción sobre esta, siendo la profundidad, tipo y altura los aspectos que se toman como referente para decidir si se continúa o no en el carril de circulación normal.

Bajo esta consideración, se pretende desarrollar un sistema capaz de generar una detección oportuna de los baches en tiempo real durante la conducción, de manera que pueda alertar al conductor con anterioridad de la presencia de baches para que este pueda tomar las medidas correspondientes, como la reducción de la velocidad del vehículo, e inspección de su entorno, verificando así la posibilidad de maniobras de esquiva o en caso de que estas no puedan ejecutarse, afrontar el bache a una velocidad que no afecte la integridad del automotor ni de sus ocupantes.

Pues según Reza Kashyzadeh y Amiri (2022, pp.593-604) la velocidad de circulación incide directamente sobre la comodidad de los ocupantes de un vehículo, en especial cuando se atraviesan irregularidades propias de las carreteras, es así como, de su análisis a velocidades constantes de 50, 40 y 30 km/h, obtuvieron que la velocidad máxima de circulación que resulta de agrado para los ocupantes es de 67 km/h, por un periodo no superior a los 178 minutos.

De esta manera, se quiere contribuir al desarrollo de soluciones enfocadas en la asistencia a la conducción, pero enmarcadas en las condiciones viales que presenta Ecuador, de manera que su aplicación no se vea limitada a carreteras que presentan condiciones y señalética adecuadas; además, se pretende motivar a los entes que integran la ingeniería automotriz a la creación de soluciones de este tipo cada vez más robustas.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un sistema para la detección de baches de la vía Riobamba – El Arenal mediante la aplicación de una cámara estereoscópica y visión por computadora para reducir la probabilidad de caídas bruscas o maniobras de esquiva peligrosas.

1.4.2. Objetivos específicos

- Realizar una revisión bibliográfica de soluciones desarrolladas con visión por computadora para identificar el hardware y software necesarios para el sistema de detección de baches.
- Recolectar imágenes a color con la cámara estereoscópica para generar una base de datos con etiquetas de baches y parches.
- Entrenar y evaluar el modelo de detección de objetos seleccionado para analizar su precisión, exhaustividad, mAP 50%-95% y mAP 50%.
- Probar el sistema de detección de baches en un recorrido por la carretera Riobamba – El Arenal para su validación.

CAPÍTULO II

2. MARCO TEÓRICO

Una vez que se han indicado los antecedentes, justificación y problemática que motivaron el desarrollo de un sistema detección de baches, y se definieron los objetivos a cumplir; en este capítulo se presentarán definiciones básicas relacionadas con deformaciones superficiales en vías asfaltadas, redes neuronales y visión por computadora, para facilitar la comprensión de la terminología utilizada en el presente proyecto y en los trabajos similares, revisados al final de este capítulo.

2.1. Referencias teóricas

La visión por computadora en conjunto con las redes neuronales tiene un campo de conocimiento muy amplio, que a su vez se mantiene en una constante actualización, sin embargo, existen definiciones elementales como las que se presentan en esta sección, que otorgan una perspectiva generalizada de esta área de estudio. Por otro lado, también se tratarán las deformaciones en vías asfaltadas para que puedan distinguirse las características de los baches, que constituyen el objeto de análisis para el presente proyecto.

2.1.1. Deformaciones en vías asfaltadas

De la guía presentada por Koid Teng et al. (2019, pp.3-21), se pueden destacar las siguientes deformaciones en una superficie asfaltada.

2.1.1.1. Empujones

Este tipo de deformación se caracteriza por presentar abultamientos paralelos a la dirección del tráfico o al desplazamiento horizontal de los materiales que componen la superficie, en el primer caso suelen darse en zonas de frenado o aceleración, donde los neumáticos empujan con mayor fuerza sobre el pavimento; mientras en el segundo caso, los empujones transversales suelen darse en los lugares donde se producen movimientos de giro, como rotondas, por parte del tráfico.

Además de las causas anteriores relacionadas con el tráfico, los empujones también pueden darse por una mezcla que presente una reducida estabilidad, unión pobre entre la capa de asfalto y la capa subyacente por un exceso de adherentes, y un espesor escaso del pavimento. En la Ilustración 1-2 se pueden apreciar tres tipos de empujones de acuerdo con su nivel de gravedad, como son bajo, donde su efecto sobre la conducción es reducido, moderado, donde la conducción resulta difícil, y alto, donde el vehículo corre el riesgo de perder el control.

a b c

Ilustración 1-2: Empujones de nivel (a) bajo, (b) moderado y (c) alto en vías asfaltadas

Fuente: (Koid Teng et al., 2019)

2.1.1.2. Grietas

Las grietas son deformaciones que se generan por una falta de soporte en la estructura, que permita tolerar las cargas a las que se somete la capa asfáltica, esto a su vez se intensifica bajo condiciones de humedad, debido al ablandamiento de material que permite a las grietas formadas en la subbase extenderse rápidamente, provocando el desplazamiento y ruptura de capas superficiales.

Se pueden distinguir tres niveles de gravedad asociadas a estas deformaciones, como son el nivel bajo, con grietas sin desprendimiento de asfalto y un ancho menor o igual a 3mm; moderado, grietas con un desprendimiento leve de la capa asfáltica y un ancho mayor a 3mm; y alto, con grietas cuyo desprendimiento superficial resulta muy severo. Para una mayor comprensión en la Ilustración 2-2 se muestra un ejemplo de cada nivel.

a b c

Ilustración 2-2: Grietas de nivel (a) bajo, (b) moderado y (c) alto en capa asfáltica

Fuente: (Koid Teng et al., 2019)

De igual manera, en función de la distribución que presenten estas grietas se las clasifica en grietas de cocodrilo, en bloque, longitudinales, transversales, de borde y media luna, de las cuales, estas últimas se diferencian del resto debido a que presentan un desplazamiento de la capa superficial denominado “empujón”, que genera su característica forma de medialuna observada en la Ilustración 3-2, de modo que su nivel de gravedad depende de la intensidad de los “empujones” y rupturas que estas presenten.

Ilustración 3-2: Grieta en forma de medialuna con presencia de ruptura

Fuente: (Koid Teng et al., 2019)

2.1.1.3. Rodaje

Este tipo de deformación se caracteriza por la presencia de un surco paralelo al eje de la calzada, en una o ambas trayectorias de circulación de los vehículos, generalmente se produce por un espesor inadecuado de la capa asfáltica, falta de compactación, inestabilidad en las mezclas bituminosas, sobrecarga en el subsuelo e inestabilidad en las bases o subbases granuladas.

De acuerdo con la profundidad de los surcos se tienen tres niveles de gravedad: bajo, con una profundidad menor a 12mm; moderado, entre 12mm y 25mm; y alto, mayor a 25mm y la presencia de algún tipo de grietas. Dichos niveles se pueden apreciar en la Ilustración 4-2.

a b c

Ilustración 4-2: Presencia de surcos de nivel (a) bajo, (b) moderado y (c) alto en una superficie asfáltica

Fuente: (Koid Teng et al., 2019)

2.1.1.4. Corrugaciones

Las corrugaciones son ondulaciones que se presentan en el eje transversal de la calzada, donde el espacio entre las crestas suele ser inferior a 2m. La falta de estabilidad en la superficie bituminosa, compactación ondulada de la base, alta densidad de tráfico en zonas de pendientes pronunciadas, con algunas de las principales causas de esta deformación.

En la Ilustración 5-2 se pueden distinguir tres tipos de corrugaciones en función de su efecto en la calidad de conducción, como las de nivel bajo donde su efecto resulta notable, nivel moderado correspondiente a un camino difícil, y de nivel alto que puede ocasionar la pérdida de control del vehículo.

a b c

Ilustración 5-2: Corrugaciones de nivel (a) bajo, (b) moderado y (c) alto en una superficie asfáltica

Fuente: (Koid Teng et al., 2019)

2.1.1.5. Parches

Los parches son secciones donde se ha removido y reparado una parte del pavimento, ya sea por la pérdida de su capacidad de servicio, o de su capacidad estructural. Si bien los parches están pensados para dar solución a deformaciones donde se haya afectado la uniformidad del nivel de una vía, pueden convertirse en deformaciones adicionales, cuando la estructura final del parche se halla sobre o debajo del nivel superficial del pavimento.

De igual manera, a través de la extensión y frecuencia de los parches, se puede tener un indicador sobre la necesidad de una modificación en la estructura del pavimento. En la Ilustración 6-2 se pueden apreciar dos tipos de parches, uno que sobresale se la calzada y otro que se halla debajo del nivel de esta.

a b

Ilustración 6-2: Paches (a) sobre y (b) debajo del nivel de una superficie asfáltica

Fuente: (Koid Teng et al., 2019)

2.1.1.6. Baches

Los baches constituyen una de las deformaciones más comunes que pueden presentar las vías, estos tienden a generarse ante la presencia de grietas en la capa asfáltica, que permiten el ingreso de humedad o incluso la penetración horizontal de agua debajo del pavimento, ocasionando que la zona afectada se ablande y su remoción resulte fácil por efecto del tráfico, dando así origen a un bache.

Por tal razón los baches generalmente aparecen luego de una condición climática lluviosa, como se observa en la Ilustración 7-2, donde a su vez se pueden distinguir tres tipos de baches de acuerdo con su nivel de severidad, baja, con áreas menores a 0,3m^2 y profundidades menores a 25mm; moderada, con áreas iguales a la anterior, pero con profundidades entre 25mm y 50mm; y alta, con áreas superiores a 0,3m^2 y profundidades mayores a 50mm.

a b c

Ilustración 7-2: Baches de severidad (a) baja, (b) moderada y (c) alta en una vía asfaltada

Fuente: (Koid Teng et al., 2019)

2.1.2. Visión por computadora

La visión por computadora constituye una disciplina en la que se pretende proporcionar a una máquina, la capacidad de extraer las propiedades materiales y geométricas de objetos tridimensionales, a partir de imágenes en dos dimensiones. Es así como actualmente se disponen de técnicas para calcular con precisión un modelo 3D en base a miles de imágenes superpuestas, crear superficies 3D precisas y densas mediante emparejamiento estéreo, distinguir objetos y personas en una fotografía.

Sin embargo, pese a todos los avances que se han realizado en este campo, todavía no se consigue otorgar a las máquinas una capacidad de interpretación autónoma, que les permita desempeñarse en cualquier entorno una vez que han sido entrenadas. Además, debe recordarse que el campo de la visión computadora, tiene un enfoque de aprendizaje distinto al que tradicionalmente se emplea en otras áreas, ya que se proporciona un conjunto de las respuestas que se pretende obtener para que las máquinas se encarguen de reconstruir las propiedades de tales entradas, siendo un factor determinante la calidad de tales datos iniciales.

Es así como dentro de esta área se recurren a técnicas probabilísticas basadas en física, o de aprendizaje automático, para la generación de modelos que permiten identificar el movimiento y animación de los objetos, a través del reflejo de la luz en las superficies, su dispersión en la atmósfera, la refracción en los lentes de una cámara, su proyección en un plano, y demás fenómenos relacionados con radiometría y óptica, para mejorar la interpretación del mundo real (Szeliski, 2022, pp.1-26).

2.1.2.1. Componentes básicos de un sistema de visión por computadora

La estructura de un sistema dedicado a la visión por computadora, si bien dependerá de la técnica empleada para la recolección e interpretación de los objetos del entorno real, según González et al. (2006, pp.15-17) debe integrar los siguientes componentes básicos.

- **Sensor óptico. -** Encargado de la captura de imágenes generalmente bidimensionales. Se suelen emplear cámaras a color o monocromo.
- **Capturador de fotogramas. -** Digitaliza la información generada por el sensor óptico.
- **Unidad de almacenamiento y procesamiento. -** Guarda la imagen digitaliza, para la ejecución de un postproceso que permita extraer determinadas características.
- **Monitor. -** Muestra las imágenes recolectadas y el resultado del postproceso.

2.1.2.2. Etapas de visión por computadora

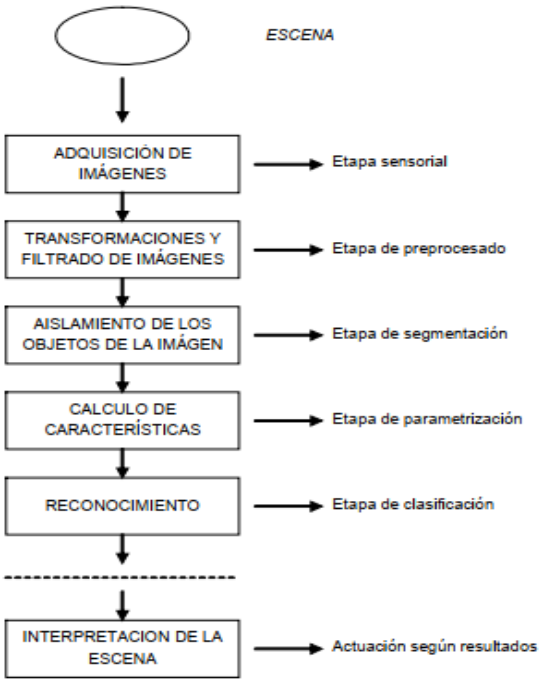


Ilustración 8-2: Etapas para un sistema de visión por computadora

Fuente: (González et al., 2006)

En el marco de los pasos fundamentales que deben ejecutarse al desarrollar un sistema de visión por computadora, González et al. (2006, pp.15-17) consideran las etapas que se muestran en la Ilustración 8-2.

De manera que se inicia con la adquisición de la imagen mediante el sensor óptico, para su posterior digitalización. A continuación, se somete a la imagen a un preproceso donde se busca mejorar la información que esta presenta, a fin de aumentar las posibilidades de éxito. Se sigue con la segmentación, donde lo que se busca identificar las partes u objetos que constituyen la imagen.

Dicha etapa presenta gran importancia, ya que de esta depende en gran medida la minimización de errores que puedan presentarse en el sistema. Por lo general se pueden optar por dos enfoques, representación por frontera, ideal para identificar características externas del objeto, como concavidades y convexidades; y representación por regiones, dedicada a características internas, como la textura.

Luego se ingresa a la etapa de parametrización, donde se incluye información cuantitativa que ayude en la distinción entre un objeto u otro. Finalmente, se llega a las etapas de reconocimiento e interpretación, donde en la primera se proporcionan las etiquetas a los objetos detectados, y en la segunda se establece un significado o acción para tales objetos.

2.1.3. Redes neuronales

Según Haykin (2009, pp.21-24) una red neuronal se puede considerar como un procesador que presenta una distribución en paralelo, y se halla compuesto por varias unidades de procesamiento simples, capaces de almacenar conocimiento a través de la experiencia y disponer de este cuando sea necesario.

Además, establece que este tipo de estructuras guardan similitud con el cerebro humano en dos aspectos, el primero relacionado con la adquisición de conocimiento del entorno, la cual se realiza mediante un proceso de aprendizaje; y el segundo, la necesidad de fuerzas de conexión de las interneuronas o también denominadas pesos sinápticos, para el almacenamiento del conocimiento.

De esta manera, al procedimiento aplicado para el aprendizaje se lo conoce como algoritmo de aprendizaje, cuya finalidad es la modificación ordenada de los pesos sinápticos para alcanzar un diseño determinado. Sin embargo, también resulta posible que estas redes modifiquen la topología

que las componen, creando nuevas conexiones sinápticas, similar a lo que ocurre con las neuronas humanas cuando estas mueren.

Para Haykin (2009, pp.21-24) la estructuración de las neuronas que componen una red se halla profundamente relacionada con el algoritmo de aprendizaje, sin embargo, se pueden distinguir tres arquitecturas elementales.

- **Redes de capa única. -** Presenta la forma más simple de una red, ya que se tiene una capa de entrada (nodos de origen) proyectada sobre una capa de salida (nodos de cálculo), pero no en sentido inverso, como se aprecia en la Ilustración 9-2.

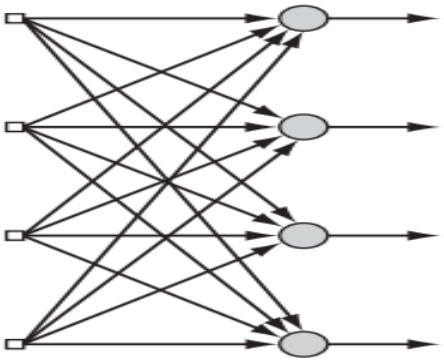


Ilustración 9-2: Red neuronal de capa única

Fuente: (Haykin, 2009)

- **Redes multicapa. -** Presenta una o más capas ocultas, las cuales no pueden observarse directamente desde la entrada o salida de la red, pero que permiten a la red extraer una perspectiva global pese a su conectividad local. Las señales de salida en capa final corresponden a la respuesta general de la red ante la activación proporcionada en los nodos de la capa inicial, donde cada capa intermedia se alimentado con las señales de salida de su capa anterior, tal como se muestra en la Ilustración 10-2.

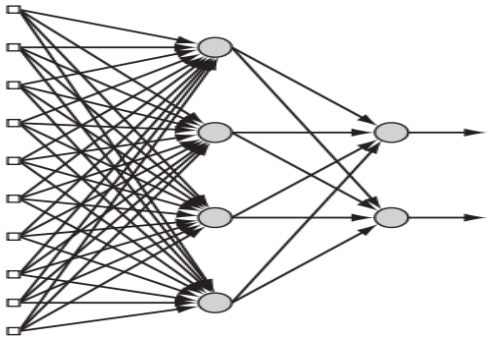
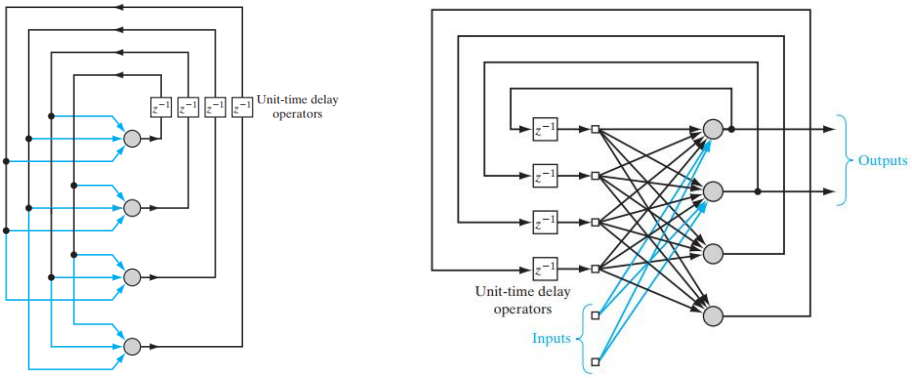


Ilustración 10-2: Red neuronal multicapa con una capa oculta

Fuente: (Haykin, 2009)

- **Redes recurrentes.** - Se diferencian de las anteriores dado que disponen de al menos un circuito de retroalimentación, lo cual incide sobre el rendimiento y capacidad de la red para el aprendizaje; ya que se aplican rapas particulares formadas por elementos de retardo (z-1) que permiten un comportamiento dinámico no lineal. Este tipo de redes se pueden usar tanto para reestructurar redes de capa única, representada en la Ilustración 11-2a, como redes multicapa, presentada en la Ilustración 11-2b.



a b

Ilustración 11-2: Red neuronal recurrente de (a) capa única y (b) multicapa

Fuente: (Haykin, 2009)

2.1.4. Detección de objetos

De acuerdo con Szeliski (2022, pp.295-301), la detección de objetos mediante la visión por computadora empezó con la detección de rostros y peatones, sin embargo, el reto actual se ha centrado en la detección y etiquetado general de objetos en conjunto con la clasificación de imágenes, y no solo de un limitado número de clases.

Es así como, a fin de motivar la creación de nuevas soluciones, han surgido ciertas competencias, como el desafío "PASCAL Visual Object Classes (VOC) Challenge", que presentaba veinte clases y veinte mil imágenes, donde se tenían retos tanto de detección como clasificación de objetos. O el "ImageNet Large Scale Visual Recognition Challenge (ILSVRC)" lanzado en 2010, que incrementó las clases e imágenes a mil y 1,4 millones, respectivamente, y si viene tenía retos de detección de objetos como el anterior, presentaba una mayor cantidad de imágenes desafiantes para alcanzar esta tarea.

Por otra parte, las bases de datos usadas para el entrenamiento de los modelos para la detección de objetos también han ido evolucionando; actualmente la base de datos COCO (Common Objects in Context) generada por Microsoft, es una de las más aplicadas para esta tarea, pues presenta una mayor cantidad de objetos por cada imagen y segmentaciones con precisión de pixel de múltiples objetos, que facilitan la detección de objetos, clasificación de imágenes y segmentación, mediante

la aplicación de redes profundas que es precisamente el enfoque sobre el cual han volcado la mayoría de soluciones modernas. Aunque también se tienen otras bases de datos como las mostradas en la Tabla 1-2.

Tabla 1-2: Bases de datos empleadas para clasificación, detección y localización de objetos

Base de datos	Tipo de datos	Contenido / Referencia
Reconocimiento de objetos		
INRIA holidays	Escenas de vacaciones	1491 imágenes
Oxford buildings	Imágenes de edificaciones	5062 imágenes
PASCAL	Segmentaciones, cajas	11000 imágenes (2900 con segmentación)
Fashion MNIST	Imágenes completas	70000 productos de moda
ImageNet	Imágenes completas	21000 clases (WorldNet), 14 millones de imágenes
Detección de objetos y segmentación		
Caltech pedestrian	Cuadros delimitadores	Peatones
MSR Cambridge	Segmentaciones por píxel	23 clases
LabelMe	Límites poligonales	Más de 500 categorías
Microsoft COCO	Segmentaciones, cajas	330000 imágenes
Cityscapes	Límites poligonales	30 clases, 25000 imágenes
Broden	Máscaras de segmentación	Variedad de conceptos visuales
Broden+	Máscaras de segmentación	Variedad de conceptos visuales
LVIS	Segmentación de instancias	1000 categorías, 2,2 millones de imágenes
Open Images	Segmentaciones y relaciones	478000 imágenes, 3 millones de relaciones

Fuente: (Szeliski, 2022)

Si bien existen diversos modelos destinados a la detección de objetos, Gandhi (2018) destaca los siguientes algoritmos.

- **R-CNN (Region Based Convolutional Neural Networks). -** Realiza una búsqueda selectiva para extraer 2000 regiones de una imagen, que se deforman en un cuadrado, para alimentar una red neuronal convolucional que genera un vector de características de salida de 4096 dimensiones, de manera que la capa de salida resulta densa, pues incluso se añaden cuatro valores de desplazamiento para mejorar la precisión del cuadro delimitador; por lo que este modelo no puede aplicarse en detecciones en tiempo real ya que el procesamiento de cada imagen toma alrededor de 47 segundos.

- **Fast R-CNN. -** Tiene un enfoque similar al anterior modelo, aunque en lugar de enviar las 2000 regiones a la red neuronal convolucional de manera repetitiva, se lo hace una sola vez por cada imagen, generando el mapa de características a partir de este proceso. Sin embargo, las propuestas de regiones producen cuellos de botella en este modelo afectando así su rendimiento.

- **Faster R-CNN. -** En este modelo se elimina la búsqueda selectiva que implementaban los modelos anteriores, y en su lugar se opta por una red adicional para la predecir las propuestas de región, que luego se remodelan mediante una capa de agrupación, para finalmente clasificar la imagen y predecir los valores de compensación de los cuadros delimitadores. Debido a que su el tiempo de procesamiento resulta menor a los anteriores modelos, este ya puede aplicarse para la detección de objetos en tiempo real.

- **YOLO (You Only Look Once). -** Este método presenta un enfoque diferente a los anteriores, pues una sola red convolucional se encarga de predecir tanto los cuadros delimitadores, como las probabilidades de cada clase sobre estos cuadros, tomando a la imagen completa para este análisis. Para ello divide a la imagen en una cuadrícula de igual número de filas y columnas, dentro de cada cuadrícula generan un determinado número de cuadros delimitadores, y a continuación, la red va estableciendo una probabilidad de clase y valores de compensación para cada cuadro delimitador, para ubicar al objeto dentro de la imagen. Este método tiene una velocidad de detección superior a la mayoría de los modelos desarrollados para la detección de objetos, permitiendo su uso en aplicaciones de tiempo real, aunque suele tener ciertas dificultades con objetos pequeños que puedan incluirse en la imagen por las propias limitaciones espaciales del algoritmo.

2.1.5. Cámaras estereoscópicas

De acuerdo con Ko (2020), las cámaras estereoscópicas basan su funcionamiento en la coincidencia píxeles de las imágenes tomadas por cada cámara (generalmente dos monocromo), para de esta manera triangular la profundidad de pixeles a partir de la línea base, que corresponde a la distancia existente entre las cámaras, como se observa en la Ilustración 12-2.

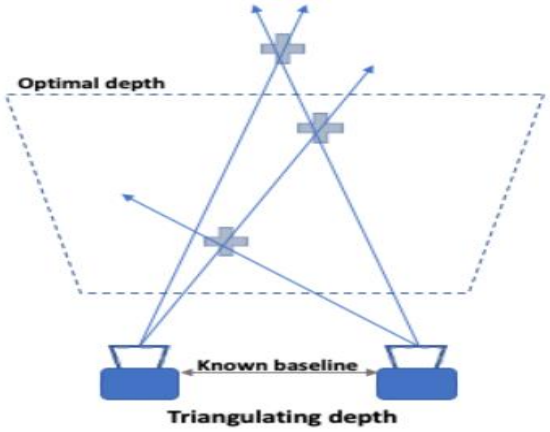


Ilustración 12-2: Esquema de funcionamiento de una cámara estereoscópica

Fuente: (Ko, 2020)

Además, destaca que las principales ventajas de este tipo de cámaras constituyen el costo de los componentes y rendimiento en ambientes exteriores iluminados, aunque en contraparte, el costo de tipo algorítmico para estimar la profundidad resulta elevado, y en ambientes con poca luz el rendimiento puede resultar afectado. De igual manera, indica que la percepción de profundidad depende de la distancia de referencia, resolución y distancia focal del objetivo; de manera que con una distancia focal y línea base amplias se puede conseguir una mayor precisión en el rango, pero a su vez se incrementa la distancia mínima para determinar la profundidad.

Ko (2020) a su vez menciona que se puede agregar un proyector de puntos láser a los sistemas convencionales de cámaras estéreo, dando lugar así al estéreo activo, que utiliza dicha proyección para simplificar la coincidencia entre las imágenes de las cámaras, de manera que la profundidad pueda ser medida de una manera más directa. Este método resulta adecuado bajo condiciones donde se tenga escasa luz, de manera que cuando se experimentan en entornos de gran iluminación, como en un día soleado, su ventaja no se suele percibir.

2.2. Antecedentes de investigación

En el marco de la detección de baches en vías asfaltadas, se han podido distinguir dos enfoques, uno relacionado con el monitoreo de las vibraciones causadas por la caída de un vehículo en un bache, y otro encaminado en la utilización de la visión por computadora; en la Ilustración 13-2 se han resumido las estrategias adoptadas en tales enfoques.

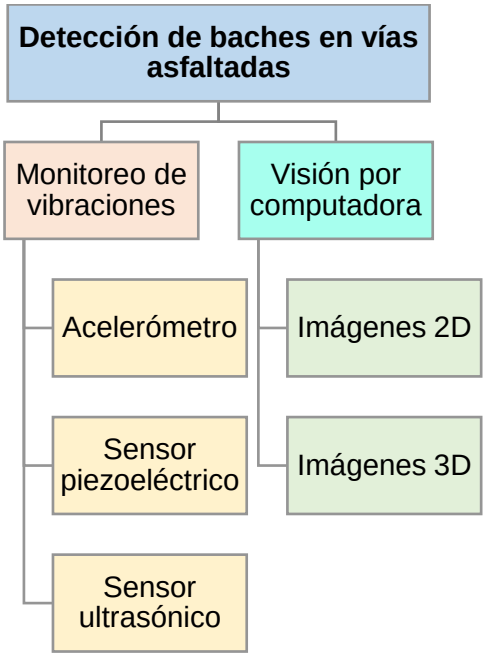


Ilustración 13-2: Clasificación de estrategias encontradas para la detección de baches en vías asfaltadas

Realizado por: Gunsha, E. 2022

2.2.1. Enfoque basado en el monitoreo de vibraciones

El enfoque basado en el monitoreo de las vibraciones generadas cuando el vehículo cae en un bache mide tales perturbaciones, generalmente mediante acelerómetros o sensores piezoeléctricos, y en función de los parámetros propios de cada solución propuesta, solicita las coordenadas a un módulo GPS (Global Positioning System), para establecer la ubicación aproximada del bache y luego subir esta información a una base de datos que pueda ser utilizada por otros conductores. Sin embargo, la principal limitación de esta estrategia radica en la actualización de la información.

A nivel de Ecuador, los dos trabajos que se han detectado en relación con la detección de baches, han optado por esta estrategia; es así que Rojas (2018, pp.51-74) ha desarrollado un sistema que mediante un sensor piezoeléctrico detecta el impacto del vehículo con un bache, a continuación solicita las coordenadas al módulo GPS, esta información se procesa en un microcontrolador para enviarse a una base de datos, que es utilizada por una aplicación para teléfonos inteligentes encargada de alertar a los conductores de forma sonora la aproximación a un bache, y a su vez permite alimentar dicha base de datos con entradas de los propios usuarios.

Con respecto al otro trabajo, Pezo y Barzola (2017, pp.8-19) plantean una solución similar, aunque en lugar de utilizar un sensor piezoeléctrico, emplean un acelerómetro para detectar el paso del vehículo por un bache, esta información se complementa con los datos de un módulo GPS y se envía a una base de datos, a la que se podrá acceder a través de una aplicación móvil o una página web, de igual manera se permite la contribución generada por los reportes de sus usuarios.

Fuera del ámbito nacional, se han encontrado alternativas como la propuesta por Anand et al. (2020, pp.1-5), donde se plantea la integración de un limitador de velocidad activado en función de la velocidad máxima de circulación detectada en la señalética de tránsito, con un detector de baches que utiliza el acelerómetro integrado en un teléfono inteligente para solicitar las coordenadas a un módulo GPS, que a su vez se suben a una base de datos que sirve para alertar a futuros conductores que transiten por la zona de detección.

Con una estrategia semejante, Wu et al. (2020, pp.1-23) han empleado el acelerómetro propio de un teléfono inteligente, pero en lugar de recurrir a un módulo GPS externo, han usado el que viene integrado en dicho dispositivo, todo esto en conjunto con una aplicación móvil encargada de administrar la información recolectada, para establecer la ubicación de los baches. Sin embargo, para mejorar el procesamiento de datos han establecido diferentes dominios de frecuencia y probado varios clasificadores disponibles para el aprendizaje automático; obteniendo así que el

método Random Forest presenta el mejor rendimiento en la clasificación de baches, ya que alcanza un precisión y recuperación del 88,5% y 75%, respectivamente.

Este enfoque de utilizar los datos registrados por un acelerómetro y módulo GPS, en conjunto con un modelo de red neuronal también lo han usado Kempaiah, Mampilli y Goutham (2022, pp.73-85), aunque en este caso se ha recurrido a módulos externos administrados por una Raspberry-Pi, y se ha adicionado la detección de reductores de velocidad. La validación de este sistema se ha llevado a cabo en un recorrido de 20km, obteniéndose una precisión promedio de 80%.

Combinando la aplicación del acelerómetro y GPS de los teléfonos inteligentes con el entrenamiento de una red neuronal, Pawar, Jagtap y Bhoir (2020, pp.3-13) han desarrollado un sistema capaz de diferenciar entre un bache y otro tipo de perturbación, el cual ha logrado una precisión y recuperación del 94,78% y 81%, respectivamente.

Mandal et al. (2022, pp.515-524) han seguido la línea de usar los datos provenientes del acelerómetro y GPS con los que cuentan los teléfonos inteligentes, aunque en este caso la información no se administra en una aplicación móvil, sino que pasa por un proceso de filtrado en el que se emplea un graficador, para determinar si el estado de la vía es bueno o malo y con ello establecer un índice de superficie, utilizable para establecer rangos de velocidad en función de las condiciones del camino, para aplicaciones de conducción autónoma.

Khaneja et al. (2020, pp.1-8) por su parte han presentado una solución que, si bien adopta estrategias anteriores, usando un acelerómetro, módulo GPS y añadiendo un sensor ultrasónico, para la detección y ubicación de baches, conjuga esta información con otras bases datos relacionadas con el tráfico e intensidad de lluvia, para entrenar modelos de aprendizaje automático que permitan obtener predicciones sobre el grado de intensidad que podrían presentar los baches en zonas geográficas específicas. Para la visualización de dicha predicción han optado por una aplicación similar a un mapa de calor, y esta ha alcanzado una precisión del 35,5%.

2.2.2. Enfoque basado en la visión por computadora

Este enfoque a diferencia del anterior no requiere que el vehículo caiga en el bache para registrar una determinada señal, por el contrario, basa su funcionamiento en la utilización de una o varias cámaras para la captura de imágenes que serán procesadas por un modelo de aprendizaje automático para reconocer la presencia de baches, de manera que la detección puede realizarse con anterioridad. A continuación, se mencionan algunos trabajos realizados bajo esta perspectiva. Rasyid et al. (2019, pp.672-675) han propuesto la creación de un sistema para la detección baches aplicando visión por computadora, donde se ha utilizado una cámara inalámbrica que se ha

ubicado en la zona cercana a la placa del vehículo, sensor combinado GPS e IMU (Inertial Measurement Unit), y una computadora portátil como unidad de procesamiento. Además, han empleado una base de datos conformada por 1000 imágenes (700 para entrenamiento y 300 para evaluación), así como el entorno de trabajo de Tensorflow y el modelo de aprendizaje automático faster R-CNN inception V2. De este modo, el sistema detecta el bache y una vez que el vehículo lo atraviesa, el sensor combinado GPS e IMU permite establecer la ubicación de este.

De manera similar al caso anterior, Kumar et al. (2020, pp.1-5) han propuesto una metodología de detección de baches aplicando el mismo modelo faster R-CNN inception V2, bajo la justificación del ahorro de recursos hasta en un 33%, en comparación con los modelos que usan convulsiones 3x3; y de igual manera han utilizado una computadora portátil como unidad de procesamiento. Aunque en este caso la base de datos utilizada se ha incrementado a 1500 imágenes, y si bien han indicado que una vez detectado el bache esta información se encontraría disponible en una aplicación móvil, no se han especificado los dispositivos destinados a esta actividad.

Garcillanosa et al. (2018, pp.191-195) por su parte han desarrollado un sistema que utiliza un módulo de cámara ubicado en la zona interna central del parabrisas, Raspberry-Pi y módulo GPS; en este sentido a diferencia de las propuestas anteriores, se ha optado por un diseño más compacto al utilizar una Raspberry-Pi y no un ordenador. En esta propuesta también se ha evaluado la calidad de envió de informes a una base de datos en la nube, con la ubicación, imagen, fecha y hora de la detección, para que esta información pueda ser utilizada por los conductores a través de un sitio web. Con este sistema ellos han obtenido una precisión y exhaustividad de 85,71% y 93,22% respectivamente; mientras que en relación con el éxito con el que se envían los reportes a la base datos, ha sido del 100%.

Con un enfoque parecido, en lo que se refiere al tamaño del hardware utilizado para el procesamiento de datos, Anand et al. (2019, pp.1-6) han propuesto una alternativa para la detección de baches y grietas en las carreteras, utilizando una cámara IDS uEye Le y una Nvidia Jetson TX-2, cuya capacidad de procesamiento supera a la Raspberry-Pi usada en el caso anterior. La red neuronal empleada ha sido la SqueezeNet, y se han evaluado el sistema de detección usando dos bases de datos, GAPs con 1969 imágenes, e ICIP con 700 imágenes, aunque dicha evaluación se ha llevado a cabo en una estación de trabajo y no en la Nvidia Jetson TX-2. De esto, han obtenido una exactitud de 98,93% y 92,37%, para las bases de datos GAPs e ICIP, respectivamente, que a su vez han sido los valores más elevados en comparación con otros modelos probados. Sin embargo, los falsos positivos generados por el sistema propuesto se han centrado en baches y grietas ya reparadas, cuya coloración difiere significativamente del resto de la capa asfáltica.

En el caso de Shah y Deshmukh (2019, pp.1-4) también utilizan modelos de aprendizaje automático para la detección de baches y reductores de velocidad, pero a diferencia de los casos anteriores, no presentan una propuesta en materia de hardware, sino únicamente de software, pues su base de datos de 5283 imágenes ha sido elaborada a partir de imágenes disponibles en internet. Inicialmente han utilizado ResNet50, con la que han obtenido una precisión de 88,90%, sin embargo, para mejorar la detección de reductores de velocidad han reemplazado las capas finales de dicho modelo, por algunas capas del modelo YOLO, aunque no se especifica en que porcentaje se benefició el reconocimiento de estos objetos.

Anandhalli et al. (2022, pp.1-11) por su lado si muestran la diferencia porcentual en materia de precisión entre dos modelos de aprendizaje automático, aunque en esta ocasión se comparan los modelos CNN y YOLO V3, pero únicamente para la detección de baches, obteniéndose una precisión de 60% y 83%, y una exhaustividad de 75% y 67% respectivamente. Esto podría resultar contrario a lo que se planteaba en la propuesta anterior, pero ha de recordarse que no se especificaba la versión del modelo YOLO y su enfoque se centraba en mejorar la detección de los reductores de velocidad.

Hasta este punto, las soluciones indicadas han utilizado únicamente imágenes en dos dimensiones, sin embargo, existen otras propuestas en las que se aplican imágenes en tres dimensiones, generalmente obtenidas por cámaras estereoscópicas, que capturan un mismo objetivo desde dos puntos diferentes, de manera que pueda generar una percepción de profundidad.

En función de aquello, Dhiman y Klette (2020, pp.3536-3550) han generado un análisis entre dos métodos de aprendizaje automático, R-CNN y YOLO V2, y dos métodos aplicando visión estereoscópica, visión estéreo de cuadro único y fusión de múltiples cuadros; donde se han obtenido precisiones de 89,8%, 69%, 45,8% y 67,4%, respectivamente. De este estudio se ha desprendido que el modelo R-CNN puede detectar baches bajo condiciones climáticas desfavorables con buena precisión, aunque el modelo YOLO V2 se destaca por su identificación en tiempo real; mientras que, de los modelos de visión estereoscópica, la fusión de múltiples cuadros presenta una precisión elevada y su implementación resulta fácil de ejecutar.

Otro estudio, aunque en esta ocasión enfocado netamente en la visión estereoscópica, ha sido el de Fan et al. (2020, pp.897-908), donde se ha generado un sistema de detección de baches con 98,7% de precisión; el cual basa su funcionamiento en la creación de mapas de disparidad correspondientes a una carretera en buen estado, de manera que cuando se compara con los mapas de disparidad obtenido en condiciones reales, los baches pueden detectarse con facilidad, al generarse una nube de puntos en la sección de esta deformación.

CAPÍTULO III

3. MARCO METODOLÓGICO

En este capítulo se abordarán los materiales y procedimientos empleados para el desarrollo del sistema de detección de baches, así como las razones que motivaron su selección para el caso de los materiales, y los parámetros de ajuste con su respectiva justificación para el caso de los procedimientos.

3.1. Enfoque de la investigación

La investigación tendrá un enfoque cuantitativo, pues durante el entrenamiento y prueba del modelo de detección de baches, se tomará como parámetro de selección del mejor modelo generado, los valores de pérdida de caja, pérdida de objetividad, pérdida de clasificación, precisión, precisión promedio principal y recuperación, donde la estabilización de estos valores marcará la elección del modelo.

Además, para la validación del modelo en tiempo real se analizarán los valores de precisión y recuperación, para definir si el sistema de detección de baches tiene un desempeño adecuado bajo las condiciones de velocidad y clima definidos.

De igual manera, para la comparación entre un vehículo que incorpora y carece del sistema de detección de baches, se usarán los datos registrados por los acelerómetros del propio automotor, de manera que se pueda identificar si existe una modificación en el intervalo de aceleración con mayor frecuencia entre un caso y otro.

3.2. Diseño de la investigación

Para el presente proyecto técnico se desarrollará una investigación de tipo aplicada, ya que se pretende emplear la visión por computadora, utilizando datos acordes a la realidad de las carreteras de Ecuador, tomando como base la vía Riobamba – El Arenal, para el desarrollo de un sistema que sea capaz de la detección de baches en tiempo real.

Por otra parte, la investigación también será de tipo transversal, puesto que la toma de datos y la validación del sistema de detección baches entrenado, se las realizarán en la vía de interés únicamente dentro del intervalo de duración del proyecto, es decir, no se considerará la evolución del estado de la vía a largo plazo, ya sea que este empeore o mejore.

Y también, se generará una investigación de tipo experimental, puesto que para la toma de datos y la validación del sistema se han establecido restricciones sobre las condiciones climáticas en las cuales se podían ejecutar estas actividades, además para todas las pruebas planteadas se han establecido límites de velocidad de circulación, de manera que este factor no afecte a los resultados que se obtengan en cada una de estas.

3.2.1. Ruta de análisis

La ruta seleccionada para desarrollar el sistema de detección de baches ha sido la vía Riobamba – El Arenal, la cual constituye una de rutas de ingreso a la Reserva de Producción Faunística Chimborazo, misma que recibe a una gran cantidad de turistas como lo indica el Ministerio del Ambiente, Agua y Transición Ecológica (2015), para el feriado de Carnaval del año 2015 acudieron 5228 turistas, mientras que, para el feriado de Semana Santa de ese mismo año, lo hicieron 3662 personas.

De esta manera, esta vía resulta de interés por el tráfico que presenta especialmente durante los días de feriado nacional, además de ser una ruta alterna para trasladarse entre las provincias de Chimborazo y Bolívar, a pesar de la existencia de la vía Guaranda – Las Herrerías – San Juan, que presenta un mejor estado de la capa asfáltica, pero en contraparte se caracteriza por ser angosta y tener una gran cantidad de curvas cerradas.

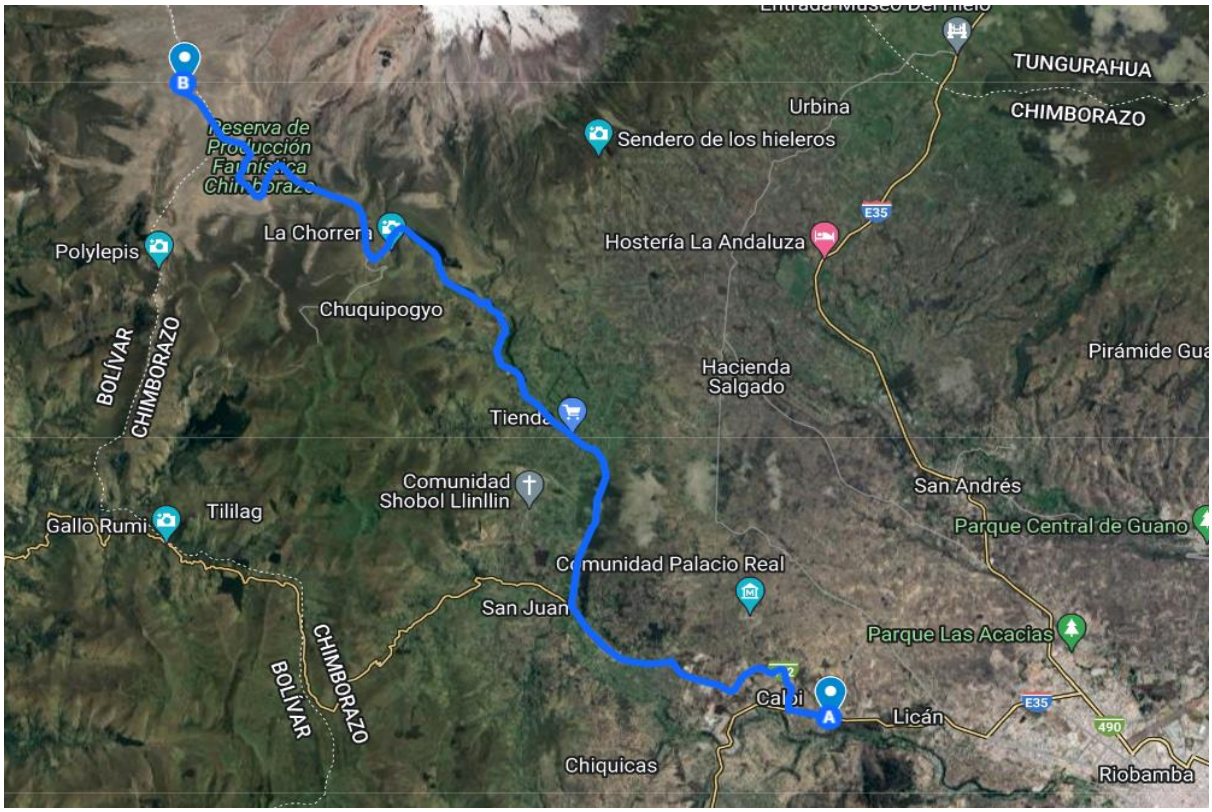


Ilustración 1-3: Recorrido de análisis extraído de la vía Riobamba – El Arenal

Realizado por: Gunsha, E. 2022

Bajo esta perspectiva, se ha considerado el recorrido de 31,5km que se puede observar en la Ilustración 1-3, desde el kilómetro 10 de la vía Troncal de la Sierra (E35), comprendido entre las parroquias rurales Licán y Calpi, pertenecientes al cantón Riobamba, hasta el kilómetro 31 de la vía San Juan - El Arenal.

Dicha selección no incluye la sección anterior al kilómetro 10 de la vía Troncal de la Sierra, dado que esta presenta un estado adecuado sin la existencia de baches, lo cual no aportaría a la detección y captura de baches; mientras que el límite final del recorrido se ha determinado por su proximidad al ingreso de la Reserva de Producción Faunística Chimborazo.

Se han definido tres zonas de interés, las cuales se caracterizan por presentar una cantidad de baches elevada. La primera se compone desde la intersección de la vía colectora Guaranda – Chimborazo (E492) con la vía E35, hasta el kilómetro 7 de la vía E492. La segunda zona se ha considerado entre el kilómetro 16 y 17 de la vía que conecta San Juan con El Arenal, y la tercera, entre los kilómetros 26 y 31 de esta misma vía.

Ilustración 2-3: Zonas de interés para el desarrollo del sistema detector de baches

Realizado por: Gunsha, E. 2022

De esta manera, las zonas de interés suman 13km de recorrido que representa el 41,27% del recorrido considerado para el desarrollo del sistema detector de baches. En la Ilustración 2-3 se pueden apreciar las tres zonas de interés mencionadas.

3.2.2. Selección de modelo de detección de objetos

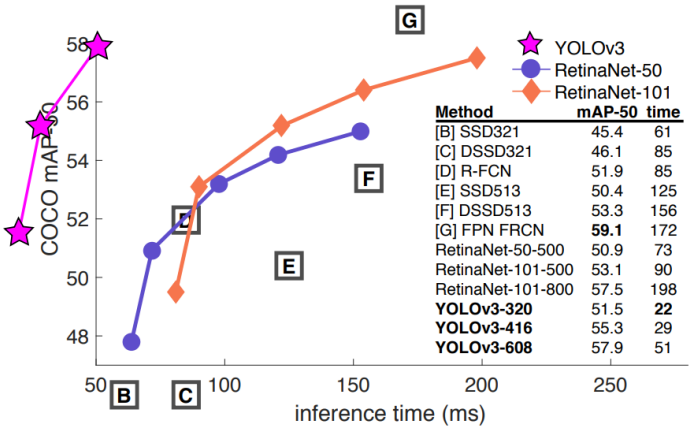


Ilustración 3-3: Comparación de tiempo de inferencia de varios modelos de detección de objetos

Fuente: (Redmon y Farhadi, 2018)

Con la revisión bibliográfica desarrollada, en primera instancia se decidió por la familia de modelos YOLO, debido a su tiempo reducido de inferencia (en su versión 3) en comparación con otros modelos dedicados a la detección de objetos, como se observa Ilustración 3-3; donde a su vez se puede apreciar que no se sacrifica precisión, pues al evaluar los modelos utilizando la base de datos COCO, su precisión promedio media (mAP) alcanza 57,9%, mientras que el valor más alto de la comparación es de 59,1% (Redmon y Farhadi, 2018, pp.1-6).

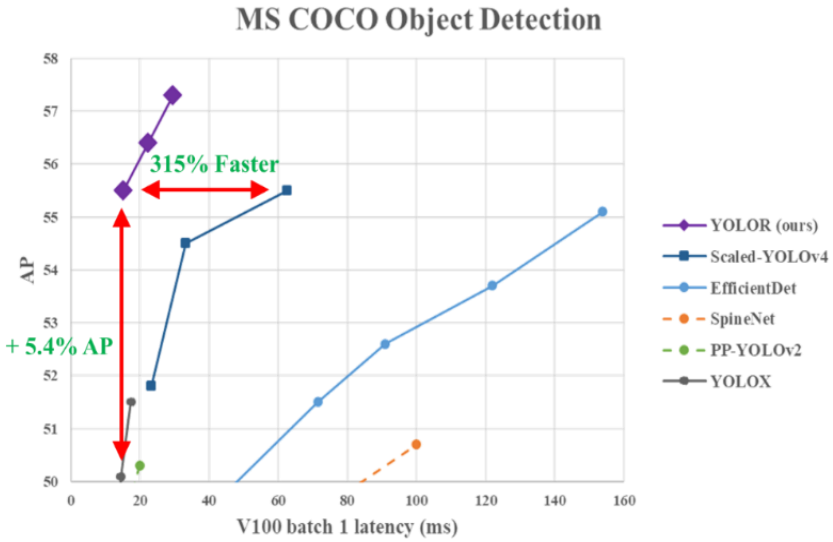


Ilustración 4-3: Comparación de YOLOR con otros modelos de detección de objetos

Fuente: (Wang, Yeh y Liao, 2021)

A continuación, se realizó una revisión dentro de la familia de modelos YOLO, donde se seleccionó al modelo YOLOR, el cual presenta una precisión media entre 55,5% y 57,5%, y un tiempo de latencia entre 10 y 30ms, al evaluarlo con la base de datos COCO; de manera que al compararlo con los demás modelos de la familia, se aprecia que resulta más rápido y conserva una adecuada precisión en la detección de objetos (Wang, Yeh y Liao, 2021, pp.1-11), como se observa en la Ilustración 4-3.

3.2.3. Vehículos de prueba

El vehículo principal seleccionado para la recolección de datos para el desarrollo y validación del sistema de detección de baches, ha sido el crossover Renault Stepway modelo 2023, dado que este tipo de vehículos sirven como nexo entre los segmentos de automóviles y SUV que tiene una alta representación en el mercado ecuatoriano; pues si bien según datos de la Asociación de Empresas Automotrices del Ecuador (AEADE), desde 2020 los SUV lideran las ventas en el país, hasta 2019 la representación de los automóviles resultaba ligeramente superior a dicho segmento, de manera que con el vehículo seleccionado se pretende cubrir a estas dos clases de vehículos. Para una mejor visualización de la participación en ventas de cada segmento se presenta la Ilustración 5-3.

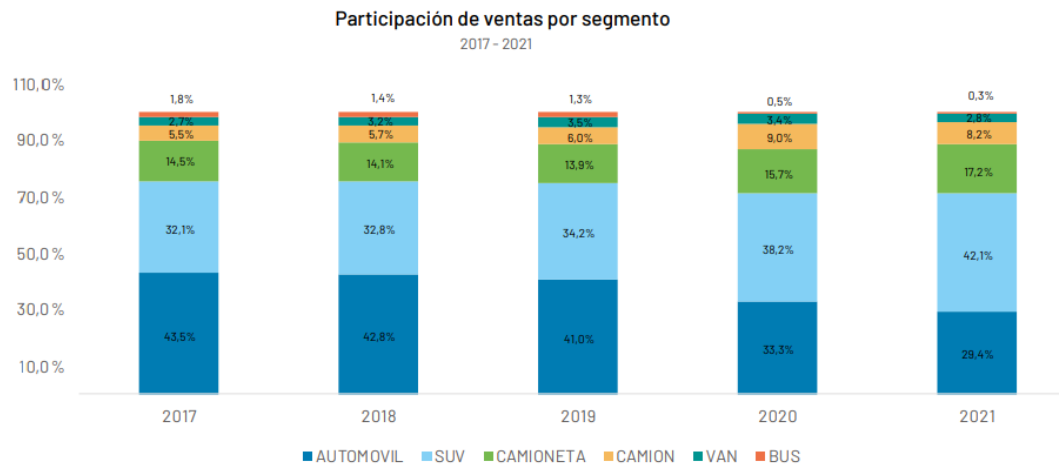


Ilustración 5-3: Participación de ventas por segmento de vehículos en el mercado ecuatoriano entre 2017 y 2021

Fuente: (AEADE, 2021)

Además, se ha incluido otro vehículo destinado únicamente para una validación adicional, el cual corresponde a un Toyota Corolla Cross modelo 2023, que pertenece al segmento de los SUV, de manera que se pueda analizar el comportamiento del sistema que ha sido entrenado desde una perspectiva distinta a la del vehículo usado para el entrenamiento.

3.3. Recolección de datos

En esta sección se indicarán los instrumentos y procedimientos utilizados para la recolección de imágenes necesarias para el entrenamiento y evaluación del modelo de detección de objetos; así como de los valores de aceleración registrados en los ejes X, Y y Z durante la prueba del detector de baches en tiempo real.

3.3.1. Instrumentos de recolección de datos

3.3.1.1. Luxonis OAK-D Pro de enfoque fijo

OAK-D Pro es un dispositivo de la marca Luxonis que integra tres cámaras, dos de las cuales se destinan a la visión estéreo, y la restante a la captura de videos en RGB. Además, dispone de un proyector de puntos laser infrarrojos, que le otorga una visión estéreo activa, y un LED de iluminación infrarroja, destinado a la visión nocturna, aunque para su utilización se requiere de un adaptador "Y" que permita tanto el flujo de información de las cámaras como la alimentación para estos dos componentes; así como de la unidad de procesamiento de visión (VPU) Intel Movidius Myriad-X (Luxonis, 2021). En la Tabla 1-3 se indican sus principales características.

Tabla 1-3: Especificaciones y peso del dispositivo Luxonis OAK-D Pro

Especificaciones de Luxonis OAK-D Pro		
Especificaciones de la cámara	Cámara a color	Par estéreo
Sensor	IMX378	OV9282
Obturador	laminación	global
Campo de visión (D/H/V)	78°/66°/54°	89,5°/80°/55°
Resolución	12MP (4032 x 3040)	1MP (1280 x 800)
Enfoque	Enfoque automático: 8cm - ∞	FF: 19,6cm - ∞
Frecuencia de imagen máxima	60 FPS	120 FPS
Tamaño de píxel	1,55µm x 1,55µm	3µm x 3µm
Tamaño de la lente	1/2,3 pulgadas	1/4 pulgada
Especificaciones del proyector	Valor	
Proyector de puntos	Iluminador infrarrojo de patrón de puntos AMS Belago 1.1	
Número de puntos	4700	
HFOI 50%	78 ± 7%	
VFOI 50%	61 ± 7%	
Longitud de onda VSCEL	940nm	
Temperatura de funcionamiento	10°C a ~60°C	
Límites absolutos de temperatura	0°C a ~80°C	
Percepción de profundidad	Valor	
Distancia mínima perceptible	~20cm (400P, extendida), ~35cm (400P u 800P, extendida), ~70cm (800P)	
Distancia máxima perceptible	~35m	
Peso	91g	

Fuente: (Luxonis, 2021)

Se debe indicar que se ha optado por la variante de enfoque fijo, ya que esta resulta de utilidad para aplicaciones donde se presentan vibraciones elevadas, como las generadas por el tránsito en una vía con baches. De igual manera este dispositivo presenta un diseño compacto, como se aprecia en la Ilustración 6-3, lo cual favorece su transporte e instalación en el vehículo.

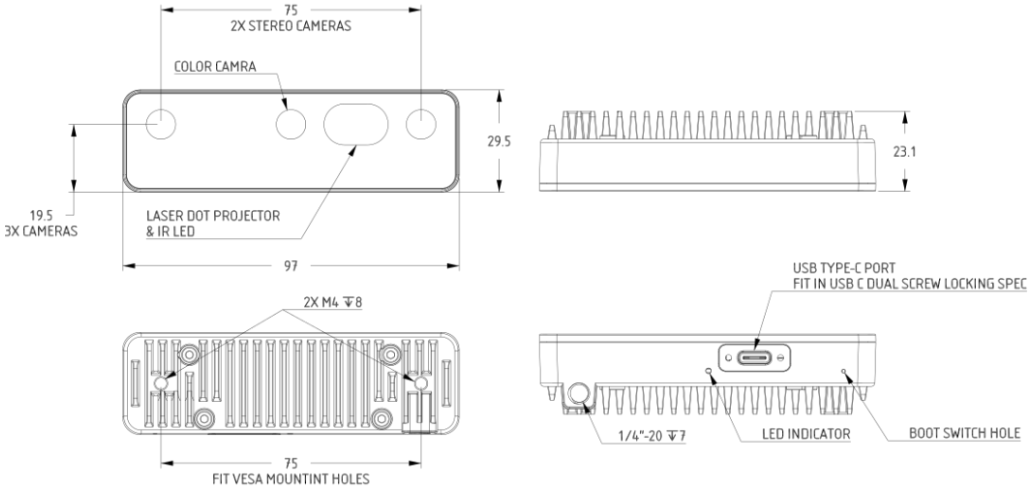


Ilustración 6-3: Dimensiones externas de dispositivo Luxonis OAK-D Pro

Fuente: (Luxonis, 2021)

Para dicha instalación se ha recurrido al soporte de ventosa Delkin Fat Gecko Stealth, que puede soportar una carga de hasta 4lb y tiene un cabezal pivotante de 140° (Delkin, 2022); en conjunto con un mecanismo amortiguador, que permita reducir la transferencia de vibraciones generadas por las irregularidades del recorrido hacia el dispositivo, tomando de base el soporte para cámaras de drones 4SCYU8WGZ2 de la cámara Dilwe; formando así la combinación que se observa en la Ilustración 7-3.

Ilustración 7-3: Conjunto formado por soporte de ventosa, amortiguador y Luxonis OAK-D Pro

Realizado por: Gunsha, E. 2022

3.3.1.2. ELM327 y aplicación Torque Pro

Se ha seleccionado el dispositivo ELM327 debido a su gran compatibilidad con la mayoría de los fabricantes automotrices, especialmente en lo relacionado con el monitoreo de los parámetros principales del vehículo; para lo cual a su vez se ha optado por la aplicación Torque Pro, que permite la visualización de tales datos, así como el registro de estos. Además, se debe indicar que este dispositivo no está pensado para la modificación de parámetros de las centralitas a las que se conecta, por lo que uso no representa un riesgo para la lectura de sus datos (OBD2 ELM327, 2022).

3.3.2. Técnicas de recolección de datos

3.3.2.1. Instalación de dispositivo Luxonis OAK-D Pro en vehículos de prueba

La instalación del dispositivo Luxonis OAK-D Pro en los vehículos de prueba, se lo ha realizado en el parabrisas en la zona superior central, de manera que las cámaras puedan captar desde una perspectiva central el recorrido de la vía. Además, se ha tomado en consideración evitar que el dispositivo pueda obstruir instrumentos como los limpiaparabrisas, así como sensores de lluvia y luz. En la Ilustración 8-3 se puede observar la ubicación del dispositivo mencionado en el automotor Renault Stepway.

Ilustración 8-3: Ubicación de dispositivo Luxonis OAK-D Pro en Renault Stepway

Realizado por: Gunsha, E. 2022

3.3.2.2. Base de datos para entrenamiento y prueba de modelo detección de baches

Para la recolección de las imágenes que conforman la base de datos se ha optado por emplear el código “RGB full resolution saver” disponible en la documentación de Luxonis (2022a), el cual permite la captura de imágenes en intervalos de tiempo muy pequeños, desde la cámara a color en formato .jpeg.

Sin embargo, dicho código captura imágenes a una resolución de 3840x2160, mientras que para la base de datos se requería de una resolución de 1920x1080, ya que la validación del modelo será en tiempo real, y un tamaño de imagen muy elevado perjudicaría el procesamiento de la

información, para ello se ha modificado la resolución, cambiando de 4K a 1080P, en siguiente línea del código disponible en el anexo 1.

```
camRgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_1080_P)
```

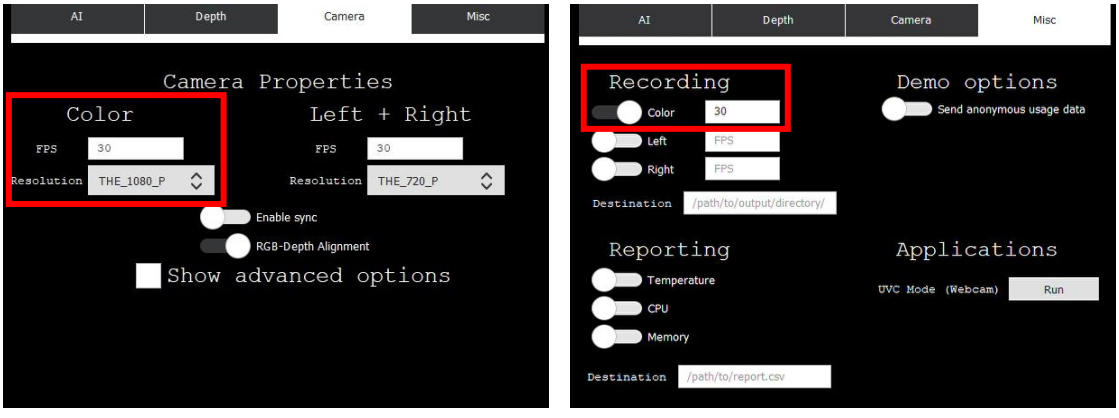
De igual manera, dado que el código en mención está diseñado para capturar imágenes cada milisegundo, se ha optado por aumentar el intervalo entre cada captura, a fin de evitar la acumulación de imágenes con información repetitiva, por lo que se ha modificado el valor de 1000 por un valor de 10 en la siguiente línea del código, de modo que las imágenes se capturen cada décima de segundo.

```
with open(f"{dirName}/{int(time.time() * 10)}.jpeg", "wb") as f:
```

Además, debe indicarse que esta recolección de imágenes se ha realizado con un clima parcialmente nublado, de manera que la visualización de las imperfecciones de la carretera sea la adecuada, y el recorrido se lo ha efectuado a una velocidad no superior a los 60km/h para capturar la mayor cantidad de información disponible.

3.3.2.3. Video de recorrido de ruta de análisis y mapa de profundidad

Debido a que para la validación del modelo de detección de baches utilizando la ruta de análisis, se requiere realizar un análisis utilizando la matriz de confusión (como se verá en la sección de técnica de procesamiento de datos), se optó por la grabación de dicho recorrido con un clima parcialmente nublado y a una velocidad no superior a los 80km/h, utilizando ambos vehículos de prueba. Para ello se ha utilizado el script de demostración "depthai_demo.py" desarrollado por Luxonis (2019), disponible en su repositorio de GitHub.



a b

Ilustración 9-3: Panel de configuración de (a) propiedades de las cámaras y (b) grabado de vídeo con dispositivo Luxonis OAK-D Pro

Realizado por: Gunsha, E. 2022

Una vez ejecutado dicho script, en la ventana emergente se ha seleccionado la pestaña “Camera” para establecer la resolución (THE_1080_P) y velocidad de video (30fps) de la cámara a color del dispositivo Luxonis OAK-D Pro, como se observa en la Ilustración 9-3a.

Mientras que en la pestaña “Misc” se ha activado la opción de grabado de vídeo únicamente para la cámara a color, como se observa en la Ilustración 9-3b, estableciendo la misma velocidad de video de la pestaña anterior (30fps); con respecto a esta grabación debe mencionarse que el formato de salida es .h264 por lo que tuvo que transformarse a formato .mp4, para poder utilizarlo en la inferencia de la red neuronal entrenada.

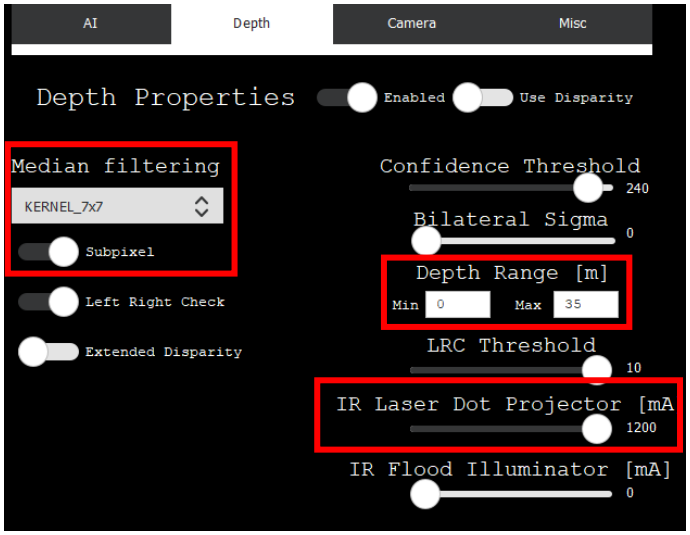


Ilustración 10-3: Panel de configuración de propiedades de profundidad de dispositivo Luxonis OAK-D Pro

Realizado por: Gunsha, E. 2022

Para obtener el mapa de profundidad se ha realizado su configuración inicial en la pestaña “Depth” mostrada en la Ilustración 10-3, donde se han seleccionado los filtros “KERNEL_7x7” y “subpíxel” que se enfocan en la percepción de profundidad para rangos amplios. El rango de profundidad se ha establecido en el valor máximo permisible por el dispositivo, que corresponde a 35m; mientras que la intensidad del proyector de puntos láser, utilizado para la visión estéreo activa, se ha configurado en 1200mA que corresponde a su valor máximo.

3.3.2.4. Datos de acelerómetros generados en conducción con y sin detector de baches

La recolección de los valores generados por los acelerómetros de los ejes X, Y y Z, a través del dispositivo ELM327 y la aplicación Torque Pro, se la ha realizado recorriendo toda la ruta de análisis a una velocidad no superior a los 80 km/h y con un clima parcialmente nublado, conduciendo únicamente el vehículo Renault Stepway.

De esta manera, para la conducción sin el detector de baches se ha conectado el dispositivo ELM327 al puerto OBDII del vehículo, y en la aplicación Torque Pro se han seleccionado los PID: Acceleration Sensor (X axis), Acceleration Sensor (Y axis) y Acceleration Sensor (Z axis).

Mientras que, para la conducción con el detector de baches, además del procedimiento anterior, se ha ejecutado en el computador la red neuronal ya entrenada (como se indica en la sección de técnicas de procesamiento de datos), para que pueda generarse la alerta sonora ante una detección.

3.4. Procesamiento de datos

En esta sección se indicarán las herramientas informáticas y procedimientos ejecutados para el procesamiento de los datos generados en el etiquetado de imágenes de la base de datos, entrenamiento y evaluación del modelo de detección de objetos, y su respectiva validación en tiempo real.

3.4.1. Herramientas para el procesamiento de datos

3.4.1.1. MakeSense

MakeSense es un sitio web que permite el etiquetado de imágenes para el entrenamiento de modelos de detección de objetos, de modo que se lo ha seleccionado porque no requiere la instalación de un programa como tal, sino que su ejecución se la hace directamente en un navegador, además la creación de clases resulta intuitiva, así como la colocación de etiquetas y su respectiva exportación en diferentes formatos, incluyendo YOLO (Alpha, 2022).

3.4.1.2. Microsoft Excel

Microsoft Excel es un programa dedicado al análisis de datos y distribuido dentro del paquete informático de Office 365; básicamente se constituye de una hoja de cálculo organizada en filas y columnas, donde los datos ingresados en cada celda pueden manipularse mediane el uso de fórmulas y funciones que pueden ser de índole financiera, lógica, matemática, trigonométrica, estadística, entre otras; además de permitir la visualización de dichos datos mediante diferentes tipos de gráficos, sobre los cuales resulta posible la configuración en parámetros de escala y apariencia (Apen, 2022). La interfaz intuitiva y diversidad de funciones fueron las razones que motivaron su selección para el presente proyecto técnico.

3.4.1.3. Anaconda Navigator

Anaconda Navigator es una interfaz gráfica que permite la ejecución de aplicaciones, así como la administración de entornos, paquetes, bibliotecas y canales de Conda sin recurrir a la utilización de líneas de código (Anaconda, 2022); esto facilita la utilización de entornos virtuales en los que se pueden instalar distribuciones específicas de Python en conjunto con determinadas bibliotecas,

de manera que en caso de ocurrir algún error, únicamente se procede con la verificación del entorno o en su defecto este sea eliminado, y se cree uno nuevo con las mismos requerimientos.

3.4.2. Técnicas de procesamiento de datos

3.4.2.1. Etiquetado de datos

Para el etiquetado de datos se ha utilizado el sitio web MakeSense, donde inicialmente se han establecido dos clases: baches y parches, esto con la finalidad de que el sistema pueda diferenciar entre estas dos deformaciones, de manera que en lo posible evite confundir un parche reciente con un bache, a causa de su coloración.

Una vez que se han colocado las etiquetas en todas las imágenes de la base de datos, se procede con la exportación de las etiquetas en formato YOLO, el cual se caracteriza por generar archivos de texto (.txt) con los nombres de las imágenes correspondientes, en los que se registran las coordenadas de las etiquetas que se han colocado y el identificador de cada clase, en este caso "0" para "bache" y "1" para "parche". Adicional a ello se ha creado un archivo de texto denominado "labels.txt" donde se especifica el identificador y nombre de las dos clases utilizadas.

3.4.2.2. Entrenamiento y evaluación de modelo de detección de baches

Para el entrenamiento del modelo de detección de baches inicialmente se ha generado la distribución aleatoria de imágenes de la base de datos, de manera que una parte se destine al entrenamiento y otra a la evaluación, para ello se ha empleado el algoritmo desarrollado por Singla (2022) disponible en el ANEXO B, generándose dos archivos de texto denominados "train.txt" y "test.txt" que contienen el listado de las imágenes que se ha repartido para cada grupo.

A continuación, se ha accedido al repositorio de YOLOR en GitHub perteneciente Wang, Yeh y Liao (2021, pp.1-11), para clonarlo en una capeta local dentro del computador utilizado para el presente proyecto, y se ha creado un entorno virtual con Python 3.8 utilizando Anaconda Navigator, donde se han instalado los requerimientos para poder ejecutar YOLOR.

En este punto debe aclararse que, a pesar de contar con un archivo de texto en el propio repositorio que contiene tales requerimientos para realizar una instalación directa, se ha optado por realizar una instalación individual de los paquetes, ya que en el caso de pycocotools al no ser directamente compatible con Windows 10 (sistema operativo del computador utilizado para este proyecto) se ha tenido que utilizar un código de instalación alterno, y en el caso PyTorch se ha buscado una versión compatible con CUDA Toolkit 11.3, que corresponde a la versión instalada en el computador utilizado. De esta manera, la instalación de los requerimientos se efectuó utilizando los siguientes códigos.

conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch
pip install cython
pip install "git+https://github.com/philferriere/cocoapi.git#egg=pycocotools&subdirectory=PythonAPI"
pip install opencv-python
pip install PyYAML
pip install scipy

Luego se ha accedido accede a la carpeta “cfg” contenida en la carpeta local de YOLOR, para la creación de una copia del archivo “yolor_p6.cfg” renombrada como “yolor_p6_custom.cfg”, en la que se han modificado los parámetros indicados en la Tabla 2-3.

Tabla 2-3: Modificaciones en archivo yolor_p6_custom.cfg para entrenamiento de modelo para la detección de baches

Ancho y alto de imagen	Número de clases	Filtros
width=1920 height=1080	[yolo] mask = 0,1,2 anchors = 19,27, 44,40, 38,94, 96,68, 86,152, 180,137, 140,301, 303,264, 238,542, 436,615, 739,380, 925,792 classes=2 num=12 jitter=.3	# 207 [implicit_mul] filters=21 # 208 [implicit_mul] filters=21 # 209 [implicit_mul] filters=21 # 210 [implicit_mul] filters=21

Fuente: (Singla, 2022)

En el caso de los filtros debe indicarse que su valor depende del número de clases, y se han determinado con la expresión: (número de clases + 5) x 3 (Singla, 2022), que ha otorgado el valor de 21 mostrado en la tabla anterior.

Luego se ha accedido a la carpeta “data” para crear un archivo denominado “custom.yaml”, en el que se ha colocado la siguiente configuración.

train: ../bache_dataset/entrenamiento/

```
val: ../bache_dataset/prueba/
nc: 2
names: ['bache','parche']
```

Finalmente, en la consola de Anaconda Prompt con el entorno virtual de YOLOR activado, se ha ejecutado el siguiente código, que se encarga del entrenamiento y evaluación de la red neuronal para la detección de baches y parches.

```
python train.py --batch-size 1 --img 1920 1080 --data custom.yaml --cfg cfg/yolor_p6_custom.cfg --weights '' --device 0 --name yolor_p6 --hyp hyp.scratch.1280.yaml --epochs 300
```

3.4.2.3. Conversión de modelo de detección de baches de formato .pt a .blob

Una vez que la red neuronal ya se ha entrenado y evaluado, se obtuvieron los pesos de la misma en formato de PyTorch (.pt), sin embargo, debido a que se pretendía ejecutar el modelo de detección de baches dentro del dispositivo Luxonis OAK-D Pro utilizando la VPU Myriad-X que este integra, se tuvieron que convertir dichos pesos al formato .blob que es compatible con la VPU mencionada.

La primera conversión realizada fue de .pt a formato ONNX (Open Neural Network Exchange), para lo cual se utilizó el algoritmo “convert-to-onnx.py” disponible en el repositorio en GitHub de Jing (2021), que fue ejecutado en un nuevo entorno virtual con Python 3.8, en el que además de instalar los paquetes utilizados en el entorno virtual de YOLOR, se incluyeron los paquetes indicados a continuación.

```
pip install onnxruntime
pip install seaborn thop
pip install nvidia-pyindex
pip install onnx-graphsurgeon
```

Con el entorno virtual configurado y activado, se utilizó el siguiente código para efectuar la conversión mencionada.

```
python convert_to_onnx.py --weights yolor_p6_custom.pt --cfg cfg/yolor_p6_custom.cfg --max_size 384 --output yolor_p6_custom.onnx
```

Con relación al código anterior debe indicarse que el parámetro “max_zise” únicamente aceptaba un valor de entrada, de manera que solo podían establecerse tamaños de imágenes cuadradas, debido a esto inicialmente se optó por un tamaño de 1920x1920 píxeles, ya que las imágenes de la base de datos tenían un ancho de 1920x1080 píxeles, de manera que la diferencia de tamaños hubiese sido rellenada con píxeles negros en la prueba en tiempo real del sistema.

Sin embargo, en el proceso de compilación del modelo de OpenVino IR (proceso que se explicara al final de esta sección) se generó un error causado por el peso de la red neuronal, ya que superara el límite de 200MB admitido por la VPU Myriad-X.

A causa de esto se optó por mantener la configuración por defecto, que establecía imágenes de entrada de 1280x1280 píxeles, con la que no se tuvo inconvenientes en el proceso de compilación del modelo de OpenVino IR (Representación Intermedia), pero cuando se intentó ejecutar la red neuronal en el dispositivo Luxonis OAK-D Pro, se generó un error debido a que la configuración de la cámara a color otorgaba imágenes de 1920x1080 píxeles, pero la red neuronal esperaba recibir imágenes de 1280x1280 píxeles.

Debido a este comportamiento también se intentó establecer imágenes de entrada de 1080x1080 píxeles, pero esto no fue posible dado que no era un tamaño aceptado por el algoritmo “convert-to-onnx.py”, por lo que finalmente, se escogió el tamaño de 384x384 píxeles que era compatible con dicho algoritmo y resultaba cercano al tamaño de 400x400 píxeles utilizado por YOLO V3 y V4 que venían configurados por defecto en el algoritmo para ejecutar la red neuronal en el dispositivo Luxonis OAK-D Pro.

Con el modelo ya en formato .onnx, se utilizó la solución en línea “ONNX Simplifier” desarrollada por Xian (2019), para eliminar los operadores redundantes de la red neuronal y de este modo simplificar el modelo en uno menos pesado.

Luego se procedió a generar otro entorno virtual que incluía Python 3.8, ONNX 1.8 y PyTorch 1.7, además se descargó el kit de herramientas de OpenVino, en su versión 2022.1, para ejecutar el siguiente código proporcionado por Chang (2021), que permite convertir el modelo en formato .onnx simplificado a un modelo de OpenVino IR.

mo --input_model yolor_p6_custom.onnx -s 255 --reverse_input_channels --output Conv_557,Conv_931,Conv_1305,Conv_1679

Los valores establecidos en el parámetro "—output" corresponden a los nodos de convolución de salida de la red neuronal ya entrenada, que deben especificarse para poder realizar esta conversión, de manera que tuvieron que extraerse los nombres de dichos nodos del propio modelo, como se observa en la Ilustración 11-3.

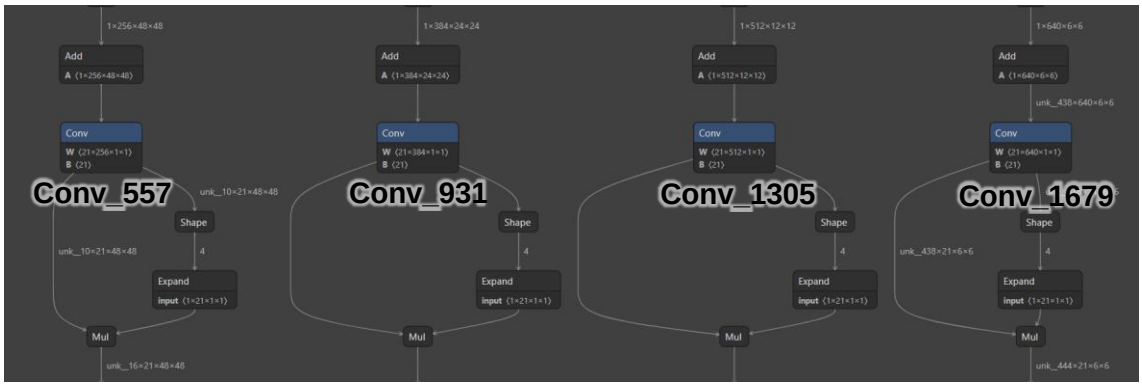


Ilustración 11-3: Nodos de convolución de salida de la red neuronal entrenada

Realizado por: Gunsha, E. 2022

Finalmente, dentro del mismo entorno virtual se procedió con la compilación del modelo OpenVino IR para convertir al modelo a formato .blob, utilizando el siguiente código.

compile_tool -m D:\blob\yolor_p6_custom.xml -d MYRIAD -ip U8 -VPU_NUMBER_OF_SHAVES 4 -VPU_NUMBER_OF_CMX_SLICES 4 -o D:\blob\yolor_p6_custom.blob

3.4.2.4. Ejecución de modelo de detección de baches en dispositivo Luxonis OAK-D Pro

Con los pesos de la red neuronal entrenada y evaluada para la detección de las clases bache y parche, ya convertidos a formato .blob, se ha tomado de base el código "RGB & TinyYolo with spatial data", extraído de la documentación de Luxonis (2022b), que puede apreciarse en el anexo C, para ejecutar la red neuronal en la VPU del dispositivo Luxonis OAK-D Pro.

Para ello se ha reemplazado el modelo de detección de objetos, que por defecto viene integrado con tiny YOLO en su versión 4, por el modelo de detección de baches ya entrenado, como se muestra en el siguiente código.

nnBlobPath=str((Path(__file__).parent/Path('../models/detector_baches.blob')).resolve().absolute())

De igual manera se cambiaron las 80 clases que vienen establecidas por defecto, para incluir únicamente las dos del modelo entrenado, bache y parche.

```
labelMap = [
    "bache", "parche"
]
spatialDetectionNetwork.setNumClasses(2)
```

La resolución de la cámara de color se mantuvo en 1080P, pero se modificó el tamaño de la imagen de vista previa a 384x384 píxeles, pues debido a las conversiones de formato realizadas sobre la red neuronal, las imágenes de entrada debían ser de este tamaño.

```
camRgb.setPreviewSize(384, 384)
camRgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_1080_P)
```

3.4.2.5. Ejecución de modelo de detección de baches en computadora

Para ejecutar la red neuronal en una computadora inicialmente tuvo que utilizarse el script de demostración "depthai_demo.py" desarrollado por Luxonis (2019), para activar el modo UVC (cámara virtual USB) en la pestaña "Misc", mostrado en la Ilustración 12-3, que permite al computador reconocer la cámara a color del dispositivo Luxonis OAK-D Pro como una webcam.

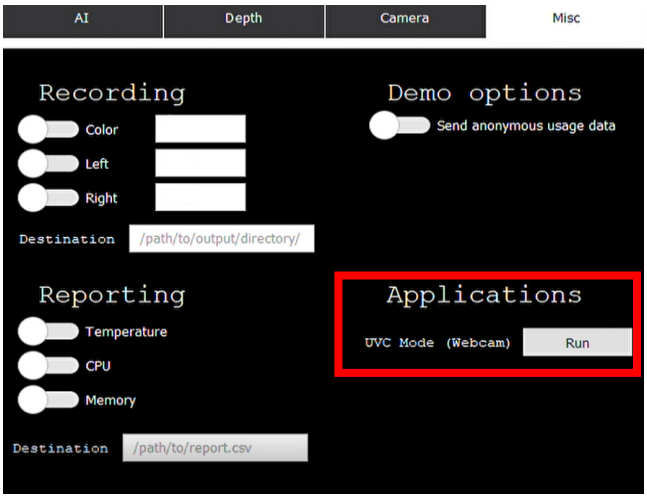


Ilustración 12-3: Activación de modo UVC de dispositivo Luxonis OAK-D Pro

Realizado por: Gunsha, E. 2022

A continuación, se ha utilizado el código "detect.py" disponible en el repositorio de GitHub de Wang, Yeh y Liao (2021, pp.1-11), desarrolladores de YOLOR, al cual se le han incorporado las siguientes líneas de código, que permiten la reproducción de un sonido de alerta de 1 segundo, cada vez que un bache es detectado por la red neuronal.

```
import pygame
```

```
pygame.init()
alerta = pygame.mixer.Sound('D:\yolorf\sonido.mp3')
if int(cls) == 0:
    alerta.play()
```

La ejecución del código “detect.py” se lo realizó a través del siguiente comando, en el que se especifica como recurso el valor 0, que indica que las imágenes de entrada para la red serán proporcionadas por una webcam; el parámetro “device” también se lo ha establecido en 0 que indica que se va a utilizar la tarjeta gráfica del computador, al igual que “classes” para indicar que únicamente se desean detectar a los baches (clase 0). Además, se ha mantenido el valor de confianza de 0,25 (25%) para las detecciones, y el tamaño de imágenes se lo ha establecido en 1920x1920 píxeles, ya que esta configuración solo acepta imágenes cuadradas.

```
python detect.py --source 0 --cfg cfg/yolor_p6_custom.cfg --weights yolor_p6_custom.pt --names data/custom.names --conf 0.25 --img-size 1920 --device 0 --classes 0
```

3.4.2.6. Validación del modelo de detección de baches en prueba de ruta completa

Una vez que se generó el video del recorrido de la ruta de análisis completa a través del procedimiento indicado en la sección de técnicas de recolección de datos, se procedió a utilizar el código “detect.py” perteneciente a Wang, Yeh y Liao (2021, pp.1-11), creadores de YOLOR, con una configuración similar a la indicada en la sección anterior, salvo por el parámetro “source” que en este caso se ha establecido como el archivo de video resultante de la grabación, como se aprecia en el siguiente comando.

```
python detect.py --source inference/images/ruta_completa.mp4 --cfg cfg/yolor_p6_custom.cfg --weights yolor_p6_custom.pt --names data/custom.names --conf 0.25 --img-size 1920 --device 0 --classes 0
```

Al finalizar la inferencia generada por este comando, se ha obtenido un archivo de video con las respectivas detecciones de baches realizadas por la red neuronal, para cada recorrido realizado por ambos vehículos de prueba. Con esta información se ha procedido a clasificar las detecciones realizadas por el detector de baches cada 10 segundos, en función del acierto o equivocación de sus predicciones siguiendo la matriz de confusión presentada en la Tabla 3-3.

Tabla 3-3: Matriz de confusión para validación de sistema de detección de baches.

		Realidad	
		Positivo	Negativo
Predicción	Positivo	Verdadero positivo	Falso positivo
	Negativo	Falso negativo	Verdadero negativo

Fuente: (Yanfeng, 2018)

Para luego calcular la precisión y recuperación del modelo mediante las siguientes expresiones.

$$\text{Precisión} = \frac{VP}{VP + FP} \qquad \text{Recuperación} = \frac{VP}{VP + FN}$$

Donde:

VP: verdadero positivo

FP: falso positivo

FN: falso negativo

3.4.2.7. Comparación de datos de acelerómetros generados durante conducción con y sin el sistema de detección de baches

Para la interpretación de los datos registrados por los acelerómetros en los ejes X, Y y Z, se ha utilizado la configuración de ejes mostrada en la Ilustración 13-3, establecida una vez que se ha analizado el registro de los datos al caer en un bache y realizar maniobras de frenado y giro. De esta manera, en los desplazamientos longitudinales se ha asignado a la aceleración del vehículo un registro positivo, y a la frenada, uno negativo.

En el caso de los desplazamientos laterales en cambio se han definido a los valores positivos como los giros hacia la izquierda y a los valores negativos, hacia la derecha; y en los desplazamientos verticales, a los valores positivos se han asignado los movimientos hacia abajo, como la caída en un bache, y a los valores negativos, los movimientos hacia arriba.

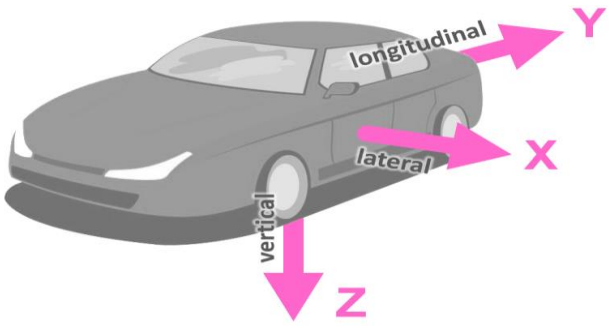


Ilustración 13-3: Ejes establecidos para análisis de registro de acelerómetros

Realizado por: Gunsha, E. 2022

CAPÍTULO IV

4. RESULTADOS

En este capítulo se mostrará la constitución de la base de datos utilizada, métricas de precisión, exhaustividad, mAP 50%-95 y mAP 50% generadas en el entrenamiento y evaluación de la red neuronal, comportamiento del modelo detección de baches al ejecutarse en la VPU Myriad-X y en la tarjeta gráfica NVIDIA GeForce RTX 2070 Max-Q Design 8GB, así como la validación del sistema a través de la precisión y exhaustividad obtenidas al analizar el recorrido de toda la ruta de análisis, y la diferencia entre los registros de los acelerómetros en los ejes X, Y y Z de una conducción con y sin el detector de baches.

4.1. Base de datos para entrenamiento y prueba de modelo detección de baches

De la recolección de imágenes realizada en las tres zonas de interés se han seleccionado 700 imágenes que incluyen baches y parches, de las cuales 630 se destinaron para el entrenamiento del modelo de detección de objetos, y 70 para su evaluación, para generar esta división se utilizó el algoritmo indicado en el ANEXO B. Algunas de las imágenes de la base de datos se presentan en la Ilustración 1-4.

Ilustración 1-4: Imágenes de la base de datos para el sistema de detección de baches

Realizado por: Gunsha, E. 2022

Con relación al número de imágenes establecido, debe indicarse que si bien Wang, Yeh y Liao (2021, pp.1-11) como creadores de YOLOR, emplearon la base de datos Microsoft COCO para

evaluar el modelo de detección de objetos, que como se indicó en la Tabla 1-2 comprende de 330000 imágenes distribuidas en 80 clases de objetos diferentes; en el presente estudio se optó por desarrollar una base de datos propia, a fin de capturar imágenes que contengan características de la zona analizada.

Es así como de los 13km que comprendían las tres zonas de interés seleccionadas para la recolección de datos, a pesar de haberse un recorrido de ida y retorno, solo se obtuvieron las 700 imágenes mencionadas, pues incluso para la selección de estas se evitó incluir imágenes con tiempos de captura similares para no repetir un mismo objeto en dos o más imágenes.

Bajo esta perspectiva, no se buscó incrementar la extensión de las zonas de interés como medida para robustecer la base de datos, pues esto hubiese dejado un porcentaje reducido de la ruta total disponible para la ejecución de la prueba en tiempo real del modelo ya entrenado.

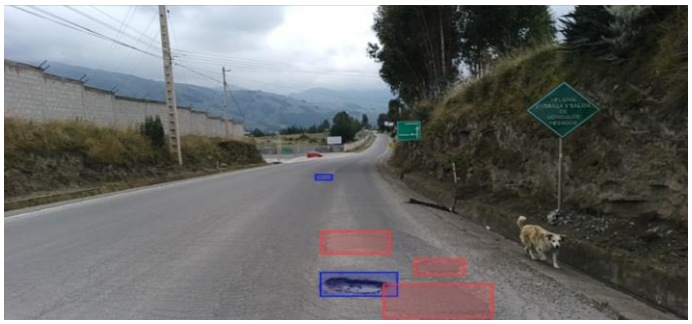


Ilustración 2-4: Etiquetado de imágenes de la base de datos para el sistema de detección de baches

Realizado por: Gunsha, E. 2022

Con respecto al etiquetado de las imágenes, una vez que se definieron las clases: bache y parche, se procedió a su ubicación en forma de cajas rectangulares. En la Ilustración 2-4, se puede apreciar la colocación de dichas etiquetas, donde las de color rojo corresponden a los parches y las de color azul a los baches.

Ilustración 3-4: Presencia de parches con tonalidad oscura en las cercanías de un bache

Realizado por: Gunsha, E. 2022

A pesar de que el sistema final solo detecta baches, se incluyó la clase parche con la finalidad de que esta actúe como un filtro y evite que el modelo confunda a los baches con los parches, ya que, en ciertas zonas de la ruta de análisis, como la mostrada en la Ilustración 3-4, por la coloración de ambas deformaciones de carretera se podría propiciar dicha confusión.

De modo que, a diferencia de los trabajos revisados, como Anand et al. (2019, pp.1-6) y Anandhalli et al. (2022, pp.1-11) donde las clases grietas y reductores de velocidad, respectivamente, también formaron parte de la evaluación sobre el rendimiento del sistema; en el presente estudio, la clase parche no ha sido evaluada dentro de la prueba en tiempo real.

Por otro lado, se buscó que la cantidad de etiquetas totales para ambas clases sean similares, de manera que durante el entrenamiento y evaluación del modelo de detección de objetos no exista una preferencia para una de las clases; obteniéndose así 1750 etiquetas para la clase bache y 2000 para la clase parche.

4.2. Entrenamiento y evaluación de modelo de detección de baches

Para el entrenamiento y evaluación del modelo se utilizaron las 300 épocas recomendadas por los creadores del modelo YOLOR, Wang, Yeh y Liao (2021, pp.1-11), aunque el tamaño del lote tuvo que reducirse a 1 en lugar de 8, valor sugerido por los creadores, ya que al dividir la cantidad de imágenes de entrenamiento y evaluación para 8 no se obtuvo un valor entero de iteraciones.

Además, a pesar de que se intentó con tamaños de 7, 5 y 2 que otorgaban valores enteros de iteraciones, en todos los casos surgió el error “sin memoria” en la tarjeta gráfica utilizada, específicamente la NVIDIA GeForce RTX 2070 Max-Q Design 8GB; para este caso, dicho error se puede atribuir a los datos de entrada, ya que al tener 3 canales, RGB, y un tamaño de 1920 x 1080, resulta sencillo llegar a saturar los 8GB de la tarjeta gráfica, a pesar de que la base solo contenga 700 imágenes; de este modo el tamaño de lote de 1 fue el que se aplicó, obteniéndose un tiempo de 24 horas con 19 minutos para el entrenamiento y evaluación del modelo de detección de objetos.

En la Ilustración 4-4a se pueden ver algunas detecciones realizadas durante el entrenamiento del modelo de detección de objetos, considerando que la clase 0 corresponde a los baches y la clase 1 a los parches; mientras que en la Ilustración 4-4b se muestran las detecciones realizadas durante la evaluación, en este caso ya se presentan las clases con sus respectivas denominaciones y a primera vista se aprecian detecciones adecuadas, aunque esto se comprobará con las métricas que se analizarán en las secciones siguientes.

a b

Ilustración 4-4: Detecciones generadas durante el (a) entrenamiento y (b) evaluación del modelo de detección de objetos

Realizado por: Gunsha, E. 2022

4.2.1. Precisión generada en entrenamiento y evaluación de 300 épocas

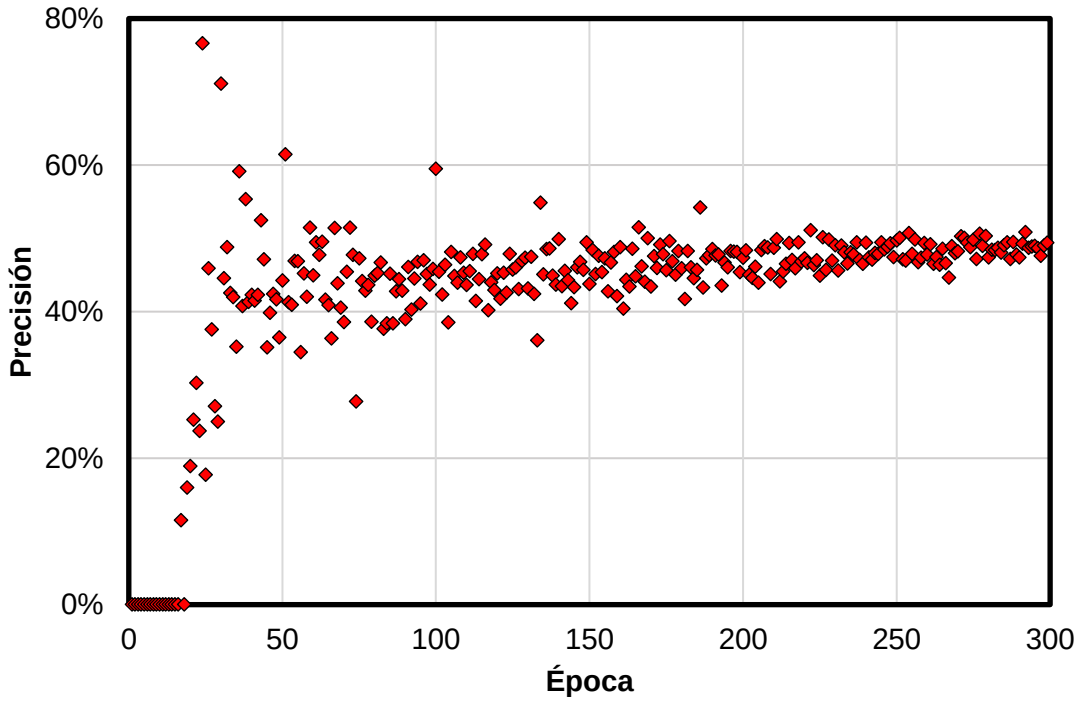


Ilustración 5-4: Precisión generada en entrenamiento y evaluación de 300 épocas

Realizado por: Gunsha, E. 2022

El comportamiento de la precisión del modelo de detección de objetos se muestra en la Ilustración 5-4, donde se aprecia que hasta la época 18 se tiene un valor de precisión muy reducido que tiende a cero, pues constituyen las primeras épocas del entrenamiento y evaluación, aunque en la época 17 este valor asciende a 11,53%, lo cual proporciona un indicio de que la red ya está encontrado pesos funcionales para las clases de objetos definidas (bache y parche).

A partir de la época 19 empieza el ascenso gradual de la precisión, aunque en la época 24 y 30 se alcanzar valores muy elevados de 76,60% y 71,09% respectivamente, lo cual nos demuestra que en estas primeras épocas existió un riesgo de sobreajuste por parte del modelo causado por un descenso en la cantidad de falsos positivos detectados por el modelo; de haberse mantenido este comportamiento se corría el riesgo de que el modelo llegase a considerar como válidas únicamente las imágenes usadas para el entrenamiento y evaluación.

En el resto de las épocas se observa que a medida que avanza el entrenamiento y evaluación, se van ajustando los pesos para alcanzar precisiones que se sitúan entre 40% y 55%, en conjunto con la estabilización de esta métrica, que partió con una desviación estándar de 18,41% en las 100 épocas iniciales, pasando por 2,91% entre las épocas 101 y 200, hasta alcanzar el valor de 1,59% entre las épocas 201 y 300.

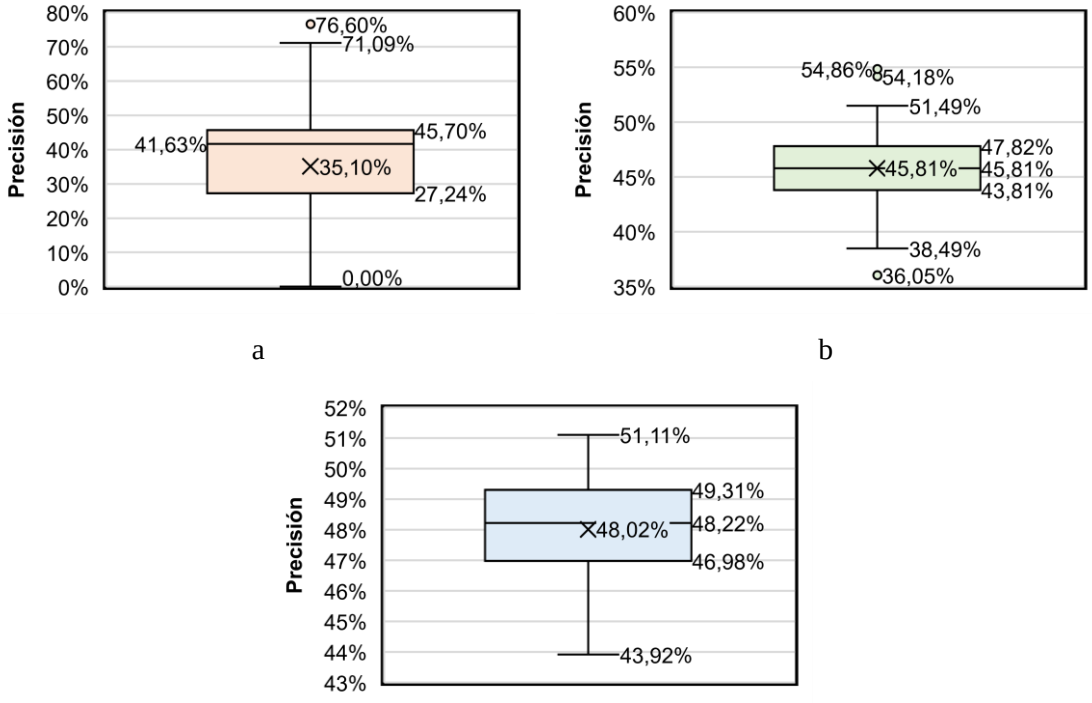


Ilustración 6-4: Diagramas de caja y bigotes sobre la precisión en secciones de (a) 100, (b) 200 y (c) 300 épocas

Realizado por: Gunsha, E. 2022

Para observar de mejor manera este comportamiento de la precisión, se ha optado por generar diagramas de caja y bigotes cada centenar de épocas, donde para la sección de 100 épocas mostrada en la Ilustración 6-4a se aprecia claramente una distribución asimétrica negativa, pues los datos más dispersos se encuentran debajo del segundo cuartil correspondiente a 41,63%, dado que en las primeras épocas de entrenamiento y evaluación se tienen valores inferiores al 30%, comportamiento que a su vez termina afectando a la precisión promedio de esta sección que alcanza los 35,10%; de igual manera se observan en la zona del bigote superior, que el límite superior y el valor atípico que allí se ubica corresponden a los valores más altos de precisión generados en las épocas 24 y 30.

En la Ilustración 6-4b por su parte se puede ver el comportamiento en la sección entre la época 101 y 200, donde a diferencia de la anterior se tiene prácticamente una distribución normal, pues el valor del segundo cuartil y la media presentan el mismo valor que corresponde a 45,81%, además el rango intercuartílico se ha reducido de 18,46% presente en la sección anterior a 4,01%, lo cual permite comprobar que el modelo está generando predicciones con una precisión cada vez más estable, aunque no completamente, pues a pesar de tener una distribución más compacta alrededor de la media, se tiene dos valores atípicos de precisión de 54,86% y 54,18%.

En la tercera sección que figura en la Ilustración 6-4c y corresponde a las épocas finales de entrenamiento y evaluación del modelo, se aprecia que nuevamente se tiene una distribución asimétrica negativa dado que los valores de precisión entre las épocas 201 y 250 resultan más dispersos si se los compara con aquellos que se obtienen a partir de la época 251, sin embargo, esta distribución no presenta una asimetría pronunciada como en la primera sección, pues el segundo cuartil y la media presentan valores cercanos, 48,22% y 48,02% respectivamente. De igual manera el rango intercuartílico se ha reducido 4,01% a 2,33%, y en esta sección ya no se tiene valores atípicos, de modo que las predicciones ya presentan valores de precisión estables.

Al comparar el valor de precisión promedio de 48,02% generado en esta última sección, con el valor publicado para YOLOR-P6 por Wang, Yeh y Liao (2021, pp.1-11), que corresponde a 55,7%, se evidencia que existe una diferencia de 7,68% en materia de precisión, aunque debe recordarse que los creadores de YOLOR han utilizado la base de datos de Microsoft COCO y el tamaño de las imágenes aplicado ha sido de 1280, mientras que para el presente proyecto se han usado imágenes de 1920x1080, de modo que el tamaño de las imágenes, su etiquetado o la robustez de la base de datos desarrollada podría estar afectando a esta métrica.

4.2.2. Exhaustividad generada en entrenamiento y evaluación de 300 épocas

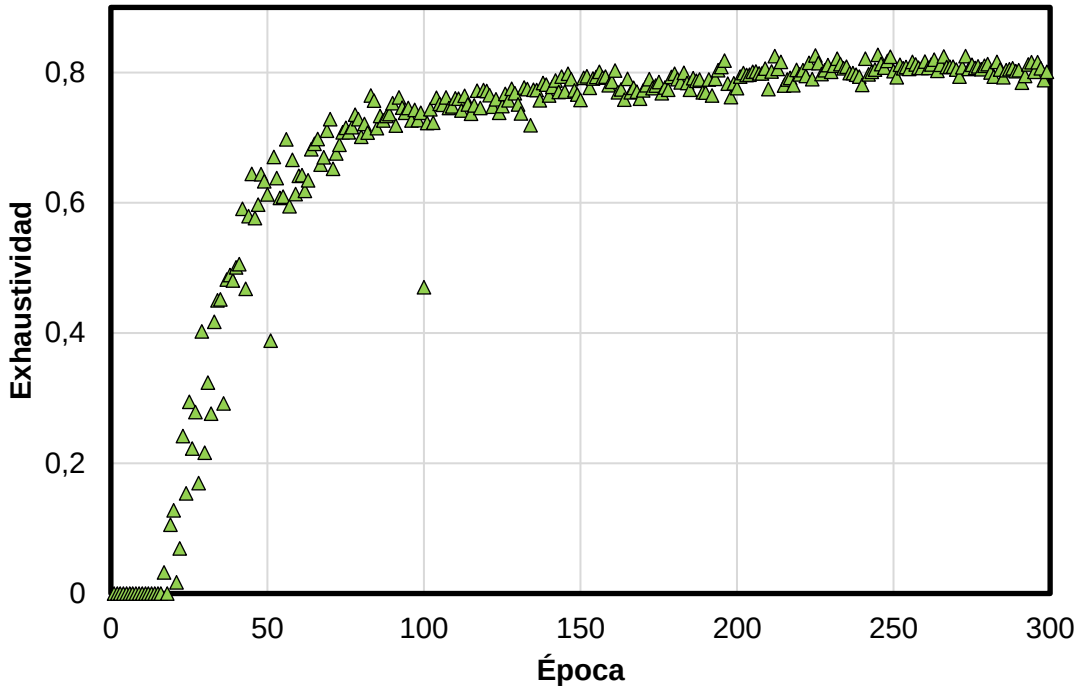


Ilustración 7-4: Exhaustividad generada en entrenamiento y evaluación de 300 épocas

Realizado por: Gunsha, E. 2022

Otra métrica obtenida durante el entrenamiento y evaluación ha sido la exhaustividad o también denominada "recall", la cual nos permite identificar en que porcentaje se han logrado identificar las clases bache y parche; el comportamiento de esta se puede visualizar en la Ilustración 7-4 donde al igual que en la precisión, hasta la época 18 se tienen valores que tienden a cero, aunque en la época anterior a esta ya se presenta un valor de 3,25% que da un indicio del acierto en la búsqueda de pesos por parte del modelo de detección de objetos, que a partir de la época 19 permite obtener niveles de exhaustividad crecientes, salvo en las épocas 51 y 100 que presentan los valores más alejados del resto de datos, causados posiblemente por un incremento de los falsos negativos durante esas épocas.

A diferencia del comportamiento de la presición donde en las primeras épocas se vio un peligro de sobreajuste en las predicciones del modelo, en el caso de la exhasutividad se observa que no existió un descenso pronunciado de los falsos negativos en ninguna de las épocas, como si lo hubo de falsos positivos que terminaron afectando a la precisión; aunque si se produjeron ascensos pronunciados de los falsos negativos, pero únicamente en las dos épocas mencionadas en el párrafo anterior, permitiendole así a la exhasutividad tener un crecimiento más gradual y no tan desordenado como el de la precisión.

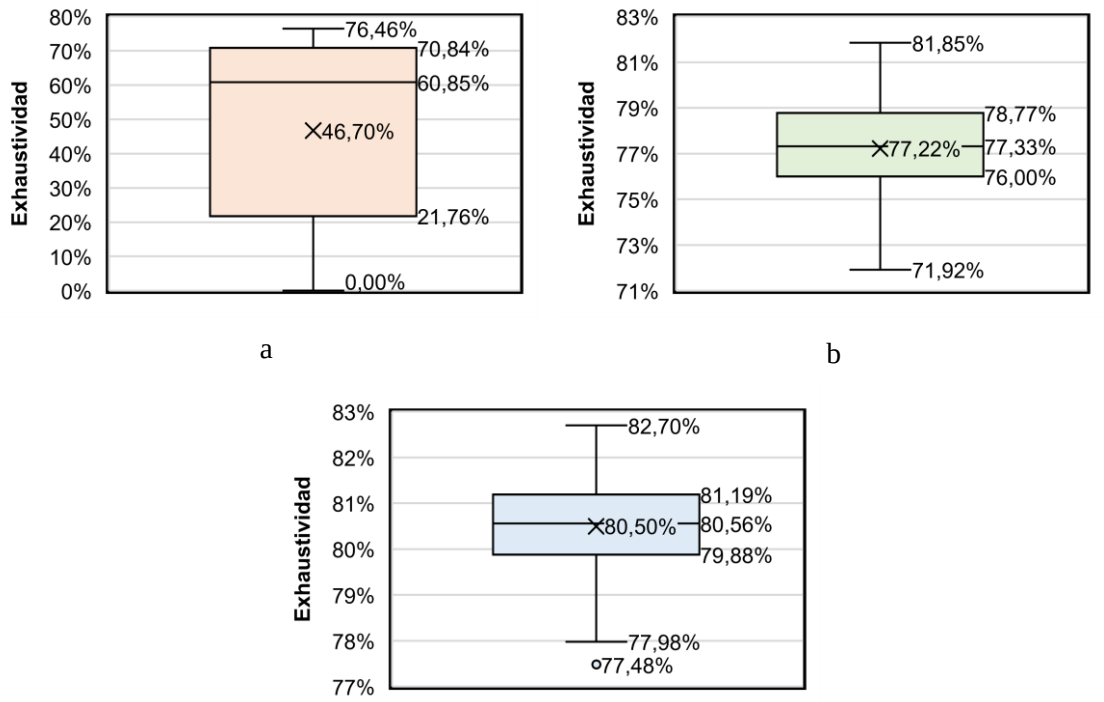


Ilustración 8-4: Diagramas de caja y bigotes sobre la exhaustividad en secciones de (a) 100, (b) 200 y (c) 300 épocas

Realizado por: Gunsha, E. 2022

Para la exhaustividad también se han generado diagramas de cajas y bigotes para las tres secciones definidas cada 100 épocas, donde en la Ilustración 8-4a correspondiente a la primera sección se tiene una distribución asimétrica negativa en la que los valores más dispersos se encuentran por debajo de 60,85% que corresponde al segundo cuartil, esto debido que durante las primeras 50 épocas se observa un crecimiento más pronunciado en los porcentajes de exhaustividad. Es así que la desviación estándar presente para esta sección alcanza un 28,03%, que su vez afecta a la exhaustividad promedio que llega a 46,70% a pesar de que entre las épocas 52 y 99 se obtienen valores entre 60% y 80%, además de carecer de valores atípicos.

Para la segunda sección que se puede ver en la Ilustración 8-4b, se observa que a pesar de tener una distribución más uniforme, no se alcanza la normalidad, ya que se tiene una diferencia de 0,11% entre el segundo cuartil y la media, de manera que la distribución es de tipo asimétrica negativa; sin embargo se logra reducir el rango intercuartílico de 49,08% de la primera sección a 2,77%, así como la desviación estándar a 1,92%, lo cual nos muestra que el modelo esta manteniendo niveles más estables de exhautividad entre las épocas 101 y 200.

Para la última sección que se muestra en la Ilustración 8-4c, se puede ver un comportamiento similar al de la sección anterior, con una distribución asimétrica negativa bajo una diferencia reducida entre el segundo cuartil y la media, igual a 0,06%. Se sigue reduciendo el rango intercuartílico pasando de 2,77% a 1,31%, y la desviación estándar hasta alcanzar el 1,05%, demostrando mayor estabilidad en el ajuste de pesos del modelo, además, para esta sección la exhasutividad alcanza un valor promedio de 80,50%, que es el más elevado de las tres secciones.

Sin embargo, este diagrama es el único que presenta un valor atípico, aunque al estar este dentro de las primeras épocas de la tercera sección, específicamente la épcoca 210, y dado que el rango intercuartílico es reducido lo cual ocasiona que los límites situados en los bigotes también lo sean, no resulta preocupante dentro de esta sección, pues incluso la diferencia entre el límite inferior y el valor atípico es de 0,5%.

En el caso de esta métrica, no se tiene un valor refencial, otorgado por los creadores del modelo YOLOR, con la que se pueda comparar, de igual manera con respecto al resto de trabajos revisados que se presentaron en la revisión de literatura, dado que emplearon otros modelos para el entrenamiento y evaluación, no se puede desarrollar tal actividad dentro de la presente fase de desarrollo del sistema de detección de baches, sino que se lo hará en la fase de evaluación en tiempo real.

4.2.3. mAP 50% - 95% generada en entrenamiento y evaluación de 300 épocas

Otra métrica corresponde a la precisión promedio media con un umbral de IoU (Intersection over Union) entre 50% y 95%, que de forma simplificada se expresa como mAP (mean Average Precision) 50%-95%, la cual permite evaluar la precisión promedio para las dos clases utilizadas, bache y parche, considerando una superposición entre el 50% y 95% de los cuadros delimitadores predichos por el modelo sobre los establecidos en la base de datos.

Es así que en la Ilustración 9-4 se observa su respectivo comportamiento, el cual presenta similitudes con la gráfica de exhaustividad, pues de igual manera se aprecia que en la época 51 y 100 se tienen valores inferiores alejados del resto de datos. En las épocas iniciales de igual manera se tiene valores que tienden a cero, aunque en este caso el comportamiento se mantiene hasta la época 15, es decir, tres épocas menos que en la exhaustividad. Sin embargo, debe recordarse que sobre esta métrica influyen tanto precisión como exhaustividad, de modo que aunque a primera vista parezca que la exhasutividad ha sido la única que ha incidido, la precisión también ha afectado a los datos de la métrica mAP 50%-95%, pues se observa como entre las épocas 50 y 150 se tiene una diferencia aproximada de 30%, mientras que para la exhasutividad dentro de estas mismas épocas, la diferencia aproximada es de 20%.

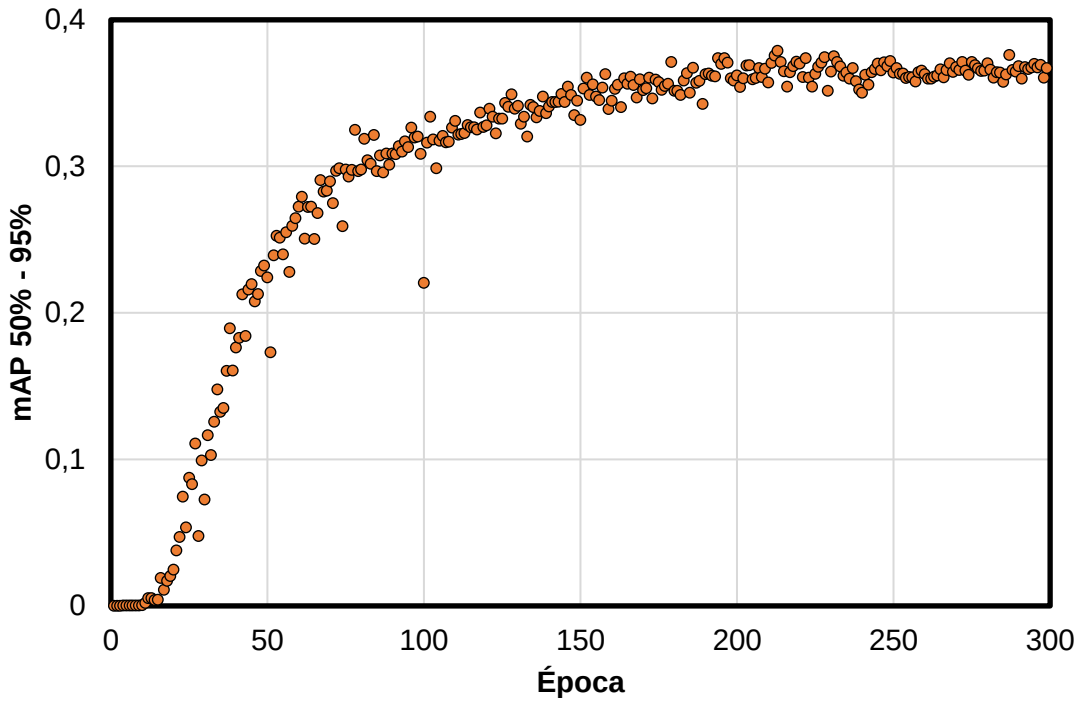


Ilustración 9-4: mAP 50%-95% generada en entrenamiento y evaluación de 300 épocas

Realizado por: Gunsha, E. 2022

Como en las métricas anteriores se han generado tres diagramas de caja y bigotes para cada intervalo de 100 épocas; donde para la primera sección, mostrada en la Ilustración 10-4a, se aprecia una distribución asimétrica negativa dado que el valor de la media, 18,62%, está por debajo del segundo cuartil, 22,59%, de manera que los datos más dispersos se hallan por debajo de este último valor, lo cual resulta evidente en la Ilustración 9-4 al observar el crecimiento pronunciado de mAP 50%-95% hasta la zona de la época 50.

Por otra parte en esta sección se tiene una desviación estádar de 11,65% acompañada de un rango intercuartílico de 22,33%, lo cual guarda relación con los comportamientos de las métricas anteriores, pues dado que en las primeras épocas todavía no se generan predicciones estables, la superposición de los cuadros delimitadores generados por estas sobre los establecidos en la base de datos, todavía resultará menor al 50% requerido para ingresar en esta métrica.

Para la segunda sección indicada en la Ilustración 10-4b, se mantiene una distribución asimétrica negativa aunque con una diferencia menor entre el segundo cuartil y la media, específicamente de 0,1%; de igual manera se tiene un descenso en el rango intercuartílico de 22,33% a 2,33%, así como en la desviación estándar, donde se llega a 1,54%, de manera que entre las épocas 101 y 200 se están generando predicciones más estables alrededor del 34,46% del total de predicciones, con una superposición de cuadros delimitadores entre el 50% y 95%.

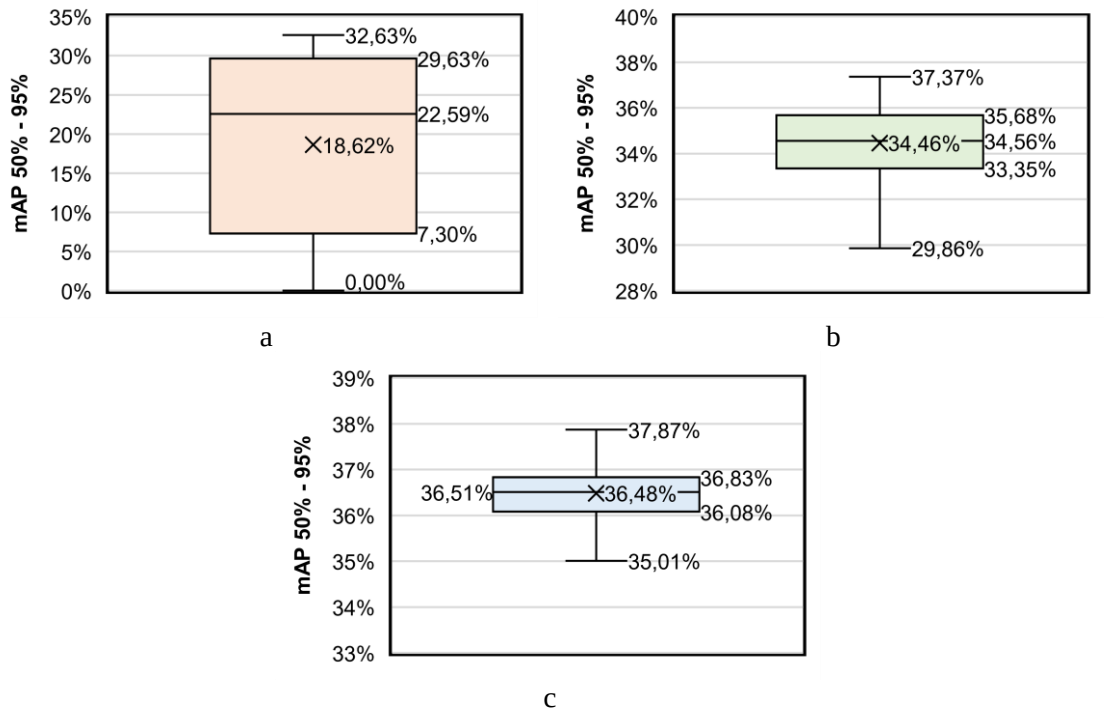


Ilustración 10-4: Diagramas de caja y bigotes sobre mAP 50%-95% en secciones de (a) 100, (b) 200 y (c) 300 épocas

Realizado por: Gunsha, E. 2022

En la tercera sección correpondiente a la Ilustración 10-4c, se continúa con una distribución asimétrica negativa con una diferencia entre el segundo cuartil y la media de 0,03%, de igual manera se sigue reduciendo el rango intercuartílico a un valor de 0,75% y la deviación estándar a 0,05%, de modo que el modelo mejora su estabilidad y a su vez incrementa el promedio de predicciones que generan cuadros delimitadores con superposiciones entre 50% y 95% en un 2,02%, dado que pasa de 34,46% de la sección anterior a 36,48% en la correspondiente a las épocas finales. Por otra parte, debe indicarse que en esta sección y en las anteriores no se han presentado valores atípicos, a pesar de los resultados de las épocas 51 y 100 que en la Ilustración 9-4 se mostraban alejados del resto de datos en las épocas siguientes y antecesoras a estas.

Con respecto a esta métrica, Wang, Yeh y Liao (2021, pp.1-11) no presentan un valor directo para mAP 50%-95% sino que optan por una mAP 75%, en la que se evalúa un porcentaje de superposición de los cuadros delimitadores del 75%, misma que para YOLOR-P6 logra un valor de 61%, valor que resulta alejado del 36,48% de mAP50%-95% obtenida en el presente proyecto, sin embargo, al no ser completamente la misma métrica, no resulta posible generar una comparación adecuada entre estos dos valores; además, debe recordarse que a medida que se incrementa el porcentaje de superposición se reduce la cantidad de predicciones capaces de alcanzar este umbral, lo cual afecta directamente a esta métrica.

4.2.4. mAP 50% generada en entrenamiento y evaluación de 300 épocas

De manera similar a la métrica anterior, se tiene la precisión promedio media con una IoU de 50% que como su nombre lo indica, evalúa la precisión promedio de las clases bache y parche considerando una superposición del 50% entre los cuadros delimitadores predichos por el modelo de detección de objetos y los indicados dentro de la base de datos.

En la Ilustración 11-4 se puede apreciar el comportamiento de la mAP 50% a lo largo de las 300 épocas de entrenamiento y evaluación, mismo que presenta similitud en cuanto a la forma de las gráficas obtenidas para la exhaustividad y mAP 50%-95%, conservando un descenso de su valor en las épocas 51 y 100; sin embargo, al comparar el rango en el que la mAP 50% se estailiza con el correpondiente a la mAP 50%-95%, se observa que el preteneciente a esta última es menor, lo que resulta coherente ya que en el caso de la mAP 50%-95% se analizan porcentajes de superposición más elevados que resultan más complicados de obtener por parte del modelo de detección de objetos.

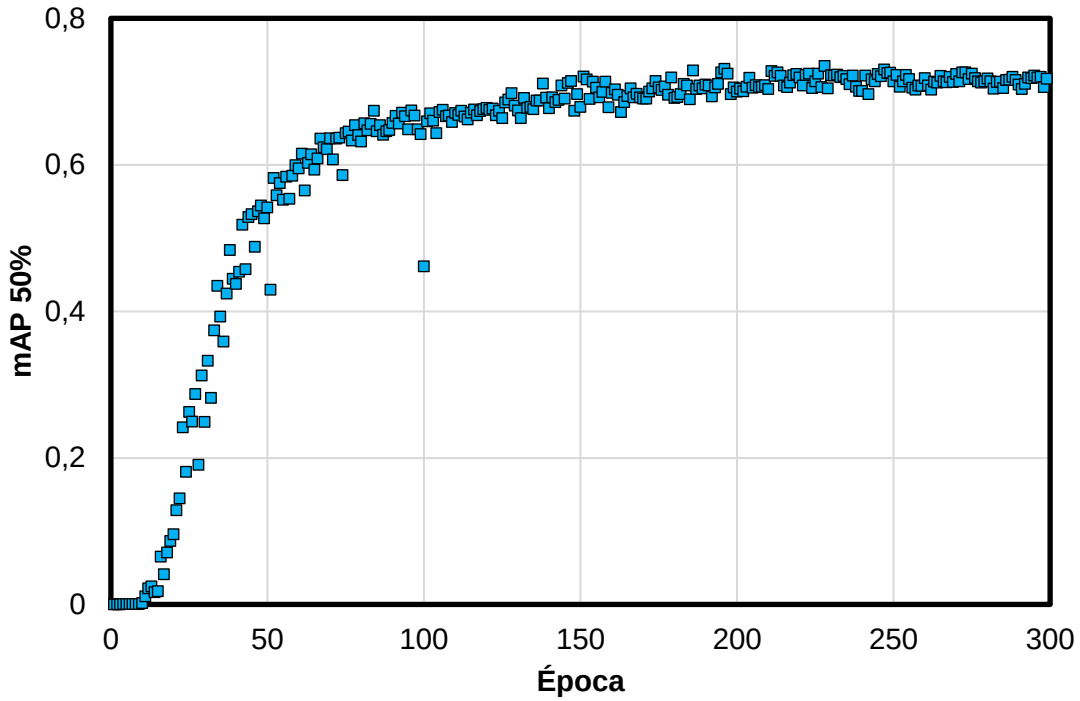


Ilustración 11-4: mAP 50%-95% generada en entrenamiento y evaluación de 300 épocas

Realizado por: Gunsha, E. 2022

Para este caso también se han generado diagramas de cajas y bigotes para las tres secciones comprendidas cada 100 épocas; en la primera sección mostrada en la Ilustración 12-4a, se tiene una distribución asimétrica negativa, con una media de 42,78% menor al segundo cuartil igual a 53,92%, causada porque en estas primeras épocas se tiene un crecimiento pronunciado hasta la época 50, de manera similar a lo manifestado en la métrica de mAP 50%-95%; esto a su vez se

acompaña de un rango intercuartílico de 39,24% y una desviación estándar de 24,13%. Por otro lado, no se observan valores atípicos y en el caso del bigote superior se aprecia una cercanía con los valores del segundo y tercer cuartil, explicado por el comportamiento de transición hacia la estabilización presente entre las épocas 50 y 100 alrededor del 50% y 70%.

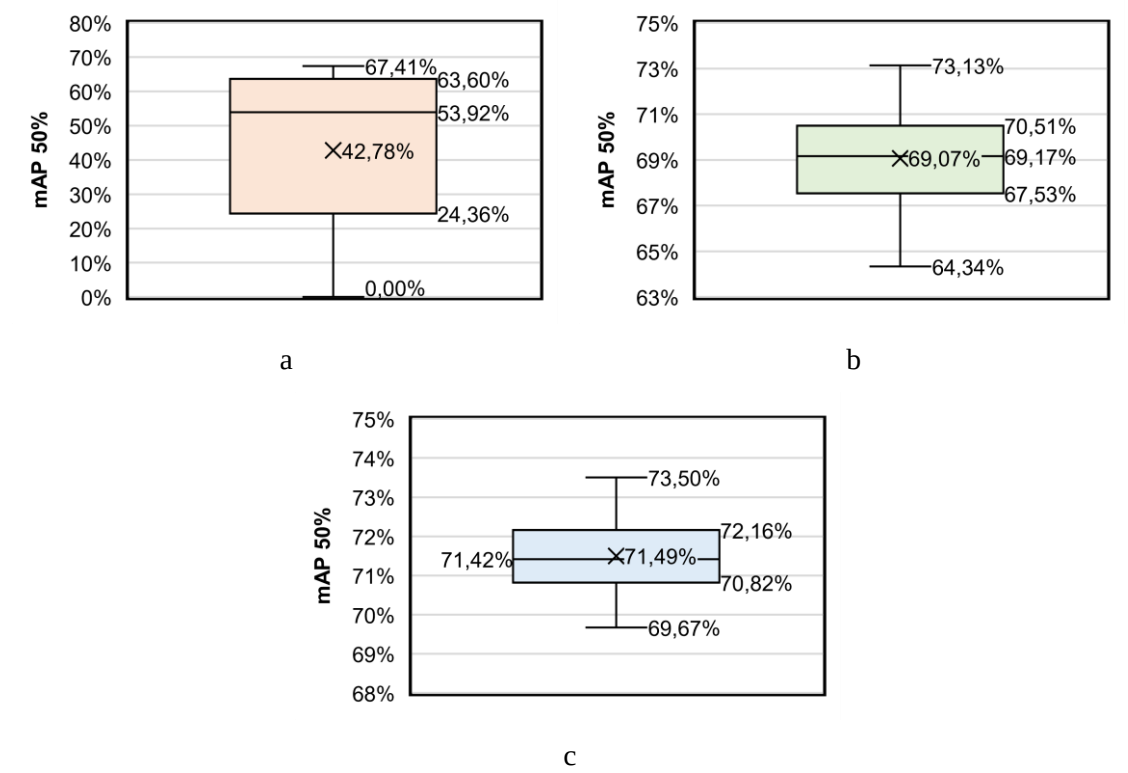


Ilustración 12-4: Diagramas de caja y bigotes sobre mAP 50% en secciones de (a) 100, (b) 200 y (c) 300 épocas

Realizado por: Gunsha, E. 2022

En la sección de 200 épocas mostrada en la Ilustración 12-4b, se mantiene la distribución asimétrica negativa, aunque con una diferencia entre la media y el segundo cuartil de 0,1%, que resulta inferior a la presentada en la sección anterior e igual a la existente en la segunda sección de la gráfica de mAP 50%-95%; dicha reducción se replica para el rango intercuartílico y la desviación estándar, alcanzado valores de 2,98% y 1,81% respectivamente, lo cual denota un comportamiento más estable entre las épocas 101 y 200, que a su vez carece de valores atípicos.

Para la tercera sección indicada en la Ilustración 12-4c se tiene un comportamiento aún más estable, con un rango intercuartílico de 1,34% y una desviación estándar de 0,79%, aunque se conserva la distribución asimétrica negativa, pero con una diferencia entre la media y el segundo cuartil de 0,07%; para esta sección tampoco se tienen valores atípicos y se logra incrementar el valor promedio de esta métrica, alcanzando un 71,49% para predicciones con una superposición

de 50% de cuadros delimitadores, que resulta superior en 2,42% a lo generado en la sección de 200 épocas y en 28,71% para la sección de las 100 épocas.

En el caso de Wang, Yeh y Liao (2021, pp.1-11) han obtenido una mAP 50% de 73,3% para su modelo YOLOR P6, entrenado y evaluado con la base de datos COCO de Microsoft; al comparar este valor con el generado en la sección de 300 épocas se tiene una diferencia de 1,81%, lo que demuestra que para detecciones con el porcentaje de superposición considerado para esta métrica, se tiene un valor más cercano al planteado por los autores del modelo de detección de objetos, a pesar de que durante la evaluación de la precisión general se obtuvo una diferencia mayor, de manera que para las clases utilizadas en el presente trabajo y su correspondiente base de datos, se podría inferir que a medida que se evalúan valores de superposición superiores al 50% se va reduciendo el valor de precisión en comparación con el modelo original.

Una vez que se han analizado las métricas anteriores, se han seleccionado los pesos del modelo de detección de objetos con los que se van a trabajar en las secciones siguientes, siendo estos los generados en la última época, dado que como se vio en el comportamiento de la precisión, esta métrica empezó a tener sus datos más estables a partir de la época 250, mientras que el algoritmo de evaluación y entrenamiento del modelo sugería como mejores pesos a configuraciones obtenidas antes de esta época, por lo que se optó por utilizar una configuración estable a nivel todas las métricas analizadas, que se conseguía en las últimas épocas. De acuerdo con lo manifestado, los pesos seleccionados presentan los valores mostrados en la Tabla 1-4.

Tabla 1-4: Precisión, exhaustividad, mAP 50%-95% y mAP 50% de pesos seleccionados

Precisión	Exhaustividad	mAP 50%-95%	mAP 50%
49,39%	80,09%	36,70%	71,77%

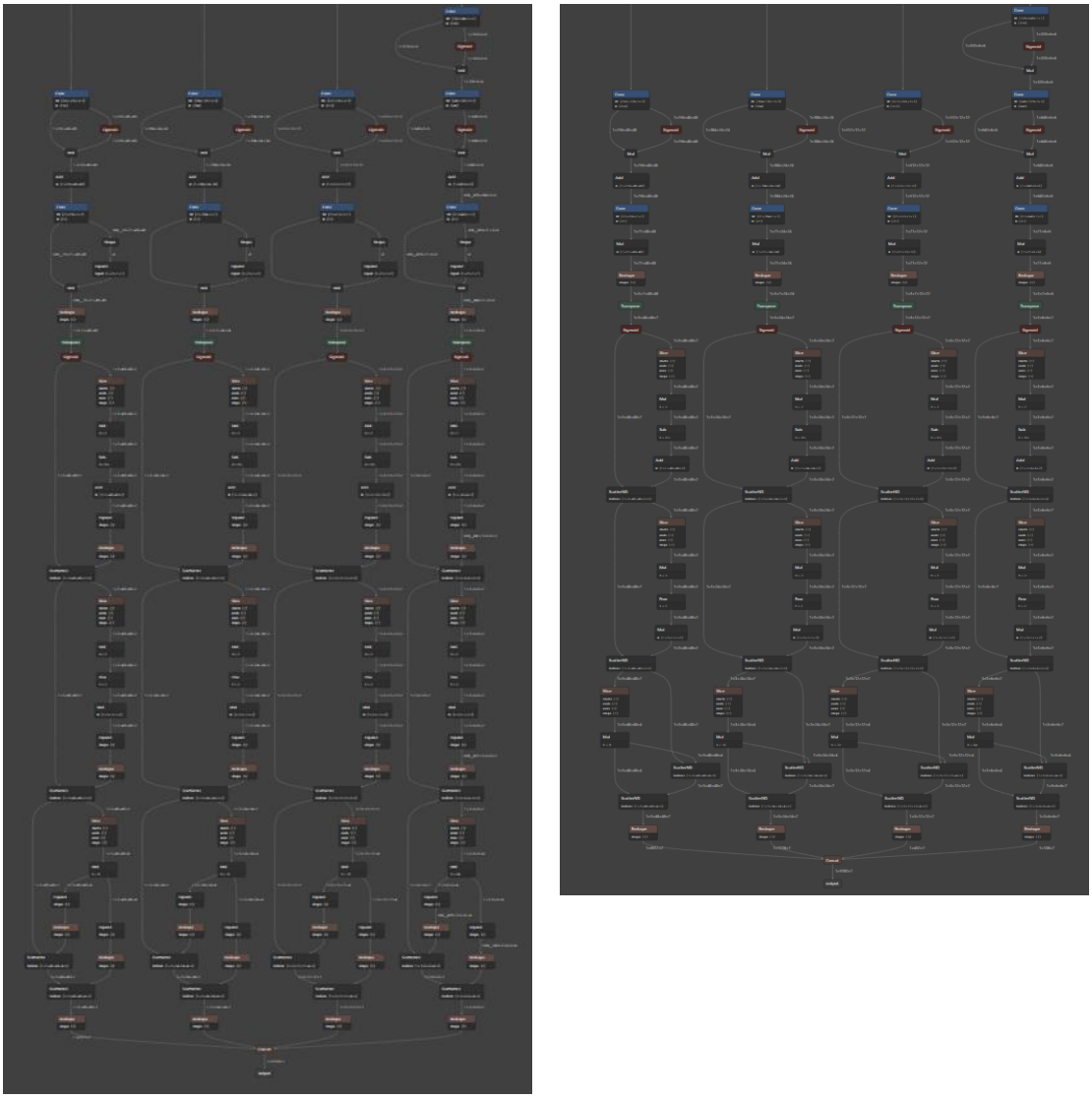
Realizado por: Gunsha, E. 2022

4.3. Implementación del modelo de detección de baches en Luxonis OAK-D Pro

Para este apartado debe recordarse que el dispositivo Luxonis OAK-D Pro cuenta con una unidad de procesamiento de visión o VPU de Intel Movidius, específicamente Myriad-X, de modo que resulta posible ejecutar el modelo de detección de baches ya entrenado en la propia cámara estereoscópica, sin embargo, como se había visto en el marco metodológico, es necesario convertir el modelo que originalmente se obtiene en formato .pt de PyTorch a .blob, que es el formato compatible con la biblioteca DepthAi que administra el dispositivo de Luxonis.

4.3.1. Conversión de modelo de detección de baches de formato .pt a .blob

El modelo de detección de baches en formato .pt generado luego del entrenamiento y evaluación, pesaba 288,57MB, pero una vez convertido a formato .onnx se redujo a 151,76MB, sin embargo, esto se logró debido a la restricción del tamaño de imágenes de entrada (explicada en el marco metodológico) establecido en 384x384. Con la simplificación del modelo en formato .onnx, se pretendía seguir disminuyendo su peso, pero dicho proceso solo logró una reducción en 1,2MB, debido a que únicamente logró eliminar ciertos operadores redundantes en los últimos nodos de la red neuronal, como se aprecia en la Ilustración 13-4.



a b

Ilustración 13-4: Últimos nodos de la red neuronal (a) original y (b) simplificada para la detección de baches

Realizado por: Gunsha, E. 2022

Por su parte, en la transformación del modelo simplificado al formato OpenVINO IR (Representación Intermedia) se logró que este se redujera a 147,34MB, y en la última conversión

a formato .blob se llegó a 73,94MB, sin embargo, al comparar este valor con el peso de los modelos YOLO V3 y V4 en su configuración "tiny", que Luxonis pone a disposición, correspondiente a 17,34MB y 11,88MB respectivamente, se notó que el modelo entrenado en el presente proyecto técnico resultaba claramente más pesado, lo cual fue perjudicial para su ejecución utilizando la VPU de la cámara estereoscópica.

4.3.2. Ejecución del modelo de detección de baches en Luxonis OAK-D Pro

Una vez que se hicieron las configuraciones indicadas en el marco metodológico sobre el algoritmo del anexo C, que permite la ejecución del modelo de detección de baches utilizando la VPU Myriad-X incorporada en el dispositivo Luxonis OAK-D Pro, se procedió con su ejecución, obteniéndose una velocidad de funcionamiento muy lenta, incapaz de superar los 5fps como se observa en la Ilustración 14-4, lo cual se atribuye al tamaño del modelo, que como se indicó en la sección anterior, es hasta cuatro veces superior al presentado por los modelos que vienen configurados por defecto.

Ilustración 14-4: Ejecución del modelo de detección de baches utilizando VPU Myriad-X

Realizado por: Gunsha, E. 2022

Sin embargo, a pesar de dicha lentitud se esperaba que pudieran generarse detecciones de baches, pero esto no sucedió, debido a la limitación de tamaño de 384x384 píxeles en las imágenes de entrada, que fue establecida por la incompatibilidad con los tamaños de 1920x1920, 1280x1280 y 1080x1080 pixeles presentada en los algoritmos utilizados para convertir el modelo de formato .pt a .blob (revisar marco metodológico); ya que el modelo originalmente se entrenó con imágenes de 1920x1080 píxeles, de manera que la información de entrada de la red neuronal tenía menor resolución a la que esta esperaba.

Debido a este comportamiento erróneo se optó por la utilización del algoritmo de inferencia proporcionado por los desarrolladores de YOLOR, con la modificación indicada en el marco metodológico, ya que con este resultaba posible ejecutar el modelo de detección de baches con imágenes de entrada de 1920x1080 píxeles a 30fps, y teniendo tiempos de inferencia alrededor de 0,05s, que permiten su aplicación en tiempo real.

4.4. Validación del modelo de detección de baches en prueba de ruta completa

Para la validación del modelo de detección de baches se ha utilizado toda la ruta de análisis para aplicar la matriz de confusión, que permitió obtener las métricas de precisión y exhaustividad; además, debe recordarse que esta prueba es la única en la que participaron dos vehículos, Renault Stepway y Toyota Corolla Cross, ya que se pretendía verificar si las proporciones entre un vehículo tipo crossover y SUV afectaban a la detección de los baches, que para este caso en concreto se tenía una diferencia de 4mm en el alto y 64mm en el ancho.

Ilustración 15-4: Falso positivo generado para la clase parche

Realizado por: Gunsha, E. 2022

De igual manera, para esta prueba únicamente se ha evaluado la detección de baches, puesto que como se había mencionado, la clase parche se ha utilizado como un filtro para reducir las confusiones del modelo entre ambas clases, por la coloración similar que se tenía en ciertos tramos del recorrido, pero a su vez este filtro evitó la generación de falsos positivos como el mostrado en la Ilustración 15-4, ya que estos terminaron afectando a la clase parche y no a la clase bache.

Tabla 2-4: Resultados de matriz de confusión generada en la prueba de ruta completa

Vehículo	Tiempo de recorrido	Verdaderos positivos	Falsos positivos	Falsos negativos
Renault Stepway	55min 40s	195	27	95
Toyota Corrolla Cross	54min 30s	211	41	91

Realizado por: Gunsha, E. 2022

Bajo esta perspectiva, en la Tabla 2-4 se muestran los valores de verdaderos positivos (VP), falsos positivos (FP) y falsos negativos (FN) correspondientes a los datos obtenidos con cada vehículo, así como sus respectivos tiempos de recorrido, que tienen una diferencia de 1min 10s a pesar de que durante la ejecución de la prueba se procuró tardar el mismo tiempo para ambos vehículos, pero debido a las condiciones de tráfico esto no se alcanzó en su totalidad.

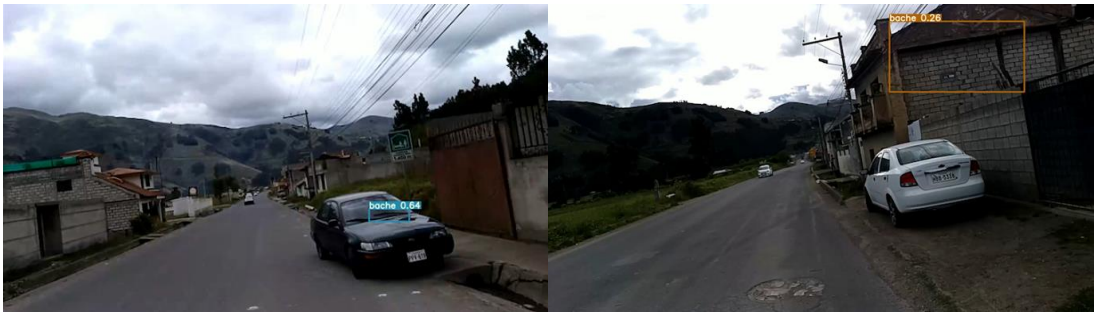
Ilustración 16-4: Verdaderos positivos generados en prueba de ruta completa

Realizado por: Gunsha, E. 2022

Dicha diferencia de tiempos terminó afectando ligeramente a la selección de fotogramas para su evaluación entre un vehículo y otro, lo cual explicaría la diferencia entre la cantidad de verdaderos positivos. Algunas detecciones consideradas como VP se muestran en la Ilustración 16-4, donde

aquellas que presentan recuadros azules fueron realizadas durante la conducción del vehículo Renault Stepway, y las de color naranja, con el automotor Toyota Corrolla Cross.

Con respecto a los falsos positivos, se pudo determinar que su generación además de ser causada por una concepción errónea del propio modelo, como se muestra en la Ilustración 17-4a; también se veía influenciada por las condiciones climáticas, ya que a pesar de ejecutar esta prueba bajo un clima parcialmente nublado, en ciertas zonas donde se tuvo mayor presencia de luz solar se originaron sombras de alto contraste con respecto a la capa asfáltica, que fueron detectadas como parches por el modelo, como se aprecia en la Ilustración 17-4b.



a

b

c

Ilustración 17-4: Falsos positivos generados por (a) concepción errónea del modelo, (b) condiciones ambientales y (c) características de la carretera

Realizado por: Gunsha, E. 2022

De igual manera, las características nuevas de la vía incidieron sobre la generación de FP, ya que cuando se realizó el recorrido con el vehículo Renault Stepway, una de las secciones de la ruta de

análisis había sido sometida a trabajos de bacheo recientes, por lo que los parches resultantes tenían un tono más oscuro a los que se tomaron originalmente para la base de datos, esto produjo ciertas detecciones incorrectas sobre estos parches más oscuros, como se observa en la Ilustración 17-4c. Mientras que la diferencia de perspectivas entre ambos vehículos, al ser prácticamente imperceptible, no terminó contribuyendo a la aparición de FP.

En el caso de los falsos negativos no se evidenció una notable afectación por parte de las condiciones climáticas, condiciones de la carretera o diferencia de perspectivas entre los vehículos, de manera que su ocurrencia obedecía a las características propias del modelo, aunque dado que las imágenes utilizadas provienen del sensor RGB de la cámara estereoscópica que carece de visión nocturna, bajo condiciones de escasa luz lógicamente se incrementaría la cantidad de FN, por la poca información que la cámara podría llegar a captar. En la Ilustración 18-4 se aprecia un FN obtenido para esta prueba.

Ilustración 18-4: Falso negativo generado para la clase bache

Realizado por: Gunsha, E. 2022

Como se había indicado en el marco metodológico, para las métricas de precisión y exhaustividad no se requiere de los verdaderos negativos, de modo que estos no han sido considerados para esta prueba; además por la propia concepción de dichos datos, que se encargan de evaluar aquellas escenas que carecen de la clase de objetos buscada, no resultaba necesaria su tabulación, ya que en el tipo de escenas analizadas en esta prueba, un verdadero negativo podría ser cualquier objeto distinto de un parche o un bache.

De esta manera, en la Tabla 3-4 se registran los valores de precisión y exhaustividad generados en esta prueba para cada vehículo, donde se aprecia que en comparación con los valores obtenidos para estas mismas métricas durante el entrenamiento y evaluación del modelo, existe un crecimiento de 38,45% y 34,34% en la precisión, y un decremento de 12,85% y 10,22% en la

exhaustividad, pero en el caso de esta prueba de ruta completa ha de recordarse que únicamente se están evaluando los baches.

Tabla 3-4: Precisión y exhaustividad obtenidas en la prueba de ruta completa

Vehículo	Precisión	Exhaustividad
Renault Stepway	87,84%	67,24%
Toyota Corrolla Cross	83,73%	69,87%

Realizado por: Gunsha, E. 2022

Al comparar estos valores con los obtenidos por Garcillanosa et al. (2018, pp.191-195), 85,71% de precisión y 93,22% de exhaustividad; Shah y Deshmukh (2019, pp.1-4), 88,90% de precisión con ResNet50; y Anandhalli et al. (2022, pp.1-11), 83% de precisión y 67% de exhaustividad con YOLO V3; se evidencia que, en materia de precisión el sistema propuesto en el presente trabajo tiene un desempeño similar al expuesto por las tres soluciones indicadas. En el caso de la exhaustividad, esto se replica con respecto a Anandhalli et al. (2022, pp.1-11), aunque al compararlo con el dato de Garcillanosa et al. (2018, pp.191-195), se tiene un decremento alrededor de un 20%.

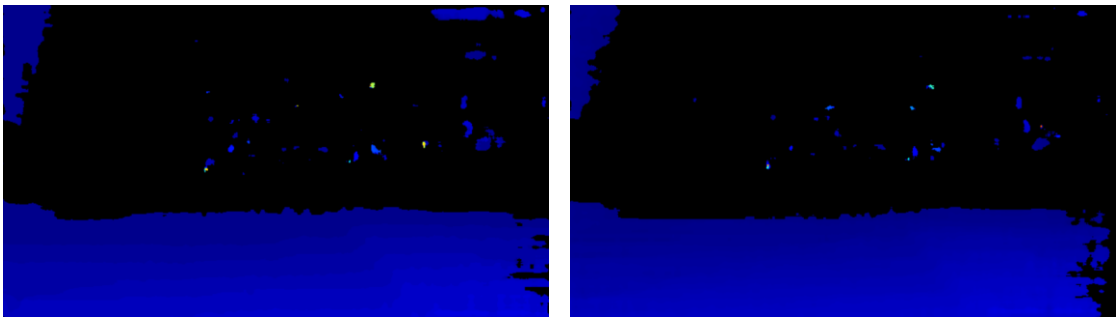
Este decremento de la exhaustividad con respecto a la solución planteada por Garcillanosa et al. (2018, pp.191-195), se podría compensar con el tiempo de procesamiento de la información, ya que en ese trabajo puede ascender a 1 segundo, mientras que en la solución planteada en el presente proyecto puede llegar a 0,05 segundos, haciéndola adecuada para su utilización en tiempo real.

Con respecto al trabajo presentado por Anand et al. (2019, pp.1-5), en la que ha empleado la red neuronal denominada SqueezeNet, se tiene una mayor diferencia en las métricas de precisión y exhaustividad, ya que en dicha solución se alcanzan 98,93% y 92,37% respectivamente, sin embargo, no se especifican los tiempos de procesamiento de la información, que constituye un factor determinante a la hora de verificar su compatibilidad con aplicaciones en tiempo real, y que podría compensar tal diferencia.

Además, un aspecto a destacar de lo planteado por Anand et al. (2019, pp.1-5), constituye la causa de la mayoría de sus falsos positivos, que corresponde a baches y grietas que han sido reparados, ocasionando que la zona donde estos se encontraban adquiera una coloración significativamente distinta al resto de la capa asfáltica; ya que dicho aspecto también se detectó en el presente proyecto técnico.

Por otro lado, durante esta prueba también se capturó el mapa de profundidad generado por la cámara estereoscópica, configurado en un rango de 0 a 35m, con el que se pretendía verificar si los baches podían identificarse con facilidad dentro de este mapa. Para ello inicialmente se

comparó la información obtenida con la visión estéreo normal y la visión estéreo activa, donde como se observa en la Ilustración 19-4, no existió una diferencia notable debido a que se tenía un entorno iluminado, donde por lo general la ventaja de la visión estéreo activa se reduce (Ko, 2020); a pesar de ello se decidió mantener la visión estéreo activa.



a b

Ilustración 19-4: Mapa de profundidad generado con visión estéreo (a) normal y (b) activa

Realizado por: Gunsha, E. 2022

Sin embargo, se notó que existía una pérdida de la información de profundidad a pesar de utilizar el modo "subpíxel", recomendado por el fabricante de la cámara estereoscópica para mediciones de rangos amplios, ocasionando que esta percepción de profundidad únicamente alcance alrededor de 15m y no los 35m que se establecieron en la configuración.

En el caso de la visualización de baches mediante el mapa de profundidad, realmente no se aprecia una diferencia entre la medición de profundidad realizada sobre la capa asfáltica y los baches, tal como se observa en la Ilustración 20-4, de manera que la utilización de este modo de visualización no aportó a la identificación de baches.

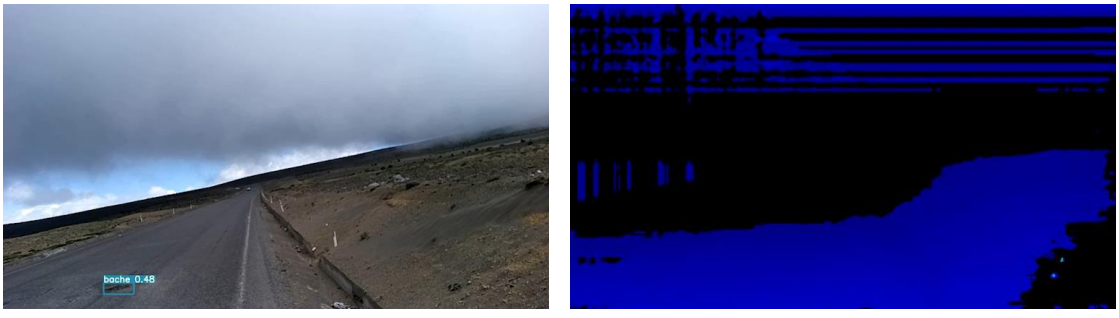


Ilustración 20-4: Zona con presencia de bache capturada con cámara a color (izq.) y visión estéreo activa

Realizado por: Gunsha, E. 2022

4.5. Comparación de datos de acelerómetros generados durante conducción con y sin el sistema de detección de baches

En esta prueba se ha comparado la ejecución de maniobras de esquiva, frenadas y desplazamientos verticales, desarrollados durante una conducción en la que se ha utilizado y no el sistema de detección de baches, que se encarga de lanzar una alerta sonora de 1 segundo cada vez que se detecta dicha deformación superficial.

Por otro lado, si bien el sistema advierte al conductor sobre la presencia de un bache, no interviene directamente en la conducción del vehículo, de manera que esta responsabilidad continúa siendo del propio conductor, quien puede decidir entre obedecer o no a la alerta lanzada por el sistema; de este modo los resultados que se presentan en esta sección no deben ser generalizados para cualquier estilo de conducción que se desee aplicar para replicar esta prueba.

Para obtener una visualización enfocada en los desplazamientos realizados en maniobras de esquiva, frenadas pronunciadas y caídas en baches, se han tomado los datos de los acelerómetros en los ejes X, Y y Z registrados en el primer kilómetro de recorrido de la ruta de análisis, que se caracteriza por tener una capa asfáltica en buen estado y carecer de curvas, para utilizarlos como un filtro de las aceleraciones registradas en los tres ejes en el resto de la ruta.

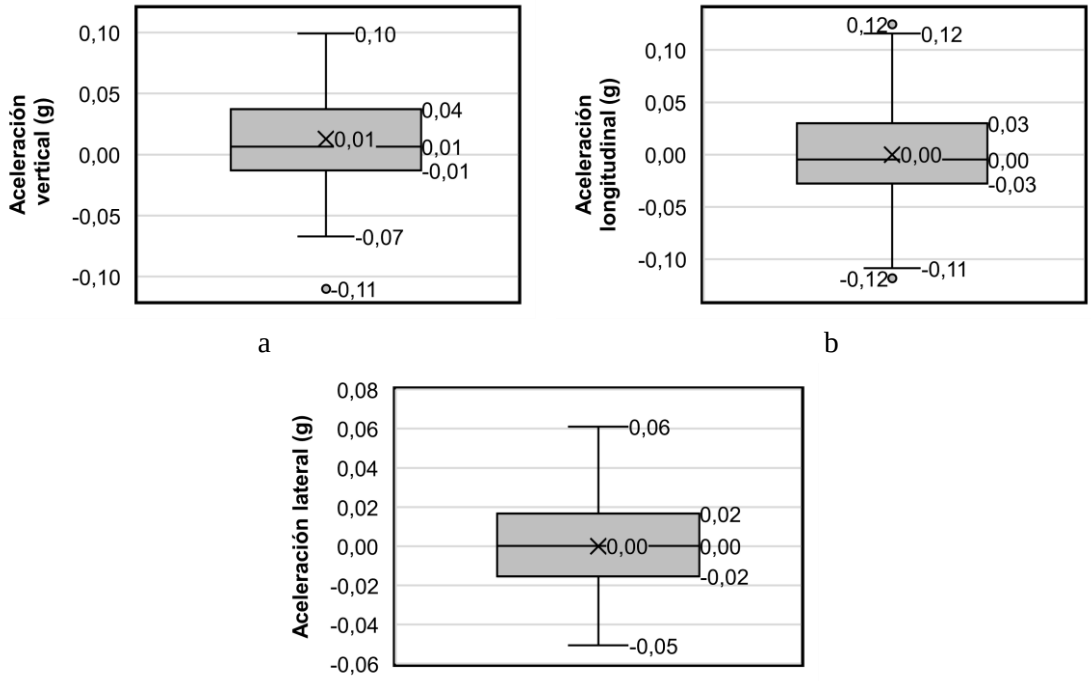


Ilustración 21-4: Diagramas de caja y bigotes de registro de aceleración (a) vertical, (b) longitudinal y (c) lateral en primer kilómetro de ruta de análisis

Realizado por: Gunsha, E. 2022

De esta manera, en la Ilustración 21-4 se muestran los diagramas de cajas y bigotes correspondientes a los registros de aceleración en los tres ejes, de los cuales se han considerado a los límites establecidos en los bigotes como los valores para la aplicación del filtro. En el caso de la aceleración vertical, debido a que la caída en un bache genera un registro positivo, se ha tomado al valor de 0,10g (fuerza g) como valor mínimo dentro del gráfico de aceleración en este eje.

Para la aceleración longitudinal, al contrario del caso anterior, la frenada del vehículo genera registros negativos, de manera que se ha tomado el valor de -0,11g para la aplicación del filtro. Y para la aceleración lateral, que proporciona a información sobre las maniobras de esquiva hacia la izquierda (+) o derecha (-), se ha establecido el valor máximo absoluto entre los dos bigotes como valor de filtrado, que corresponde a 0,06g, de modo que la región de filtrado es la misma para ambos lados.

Además, para las tres aceleraciones analizadas durante la conducción con y sin el detector de baches, se tiene una duración mayor al realizar el recorrido con la utilización de dicho detector, específicamente 6 minutos con 25 segundos, sin embargo, esta diferencia se explica por las condiciones de tráfico y maniobras anticipadas desarrollas ante la alerta sonora emitida por el sistema de detección de baches.

4.5.1. Aceleración vertical generada durante conducción con y sin el sistema de detección de baches

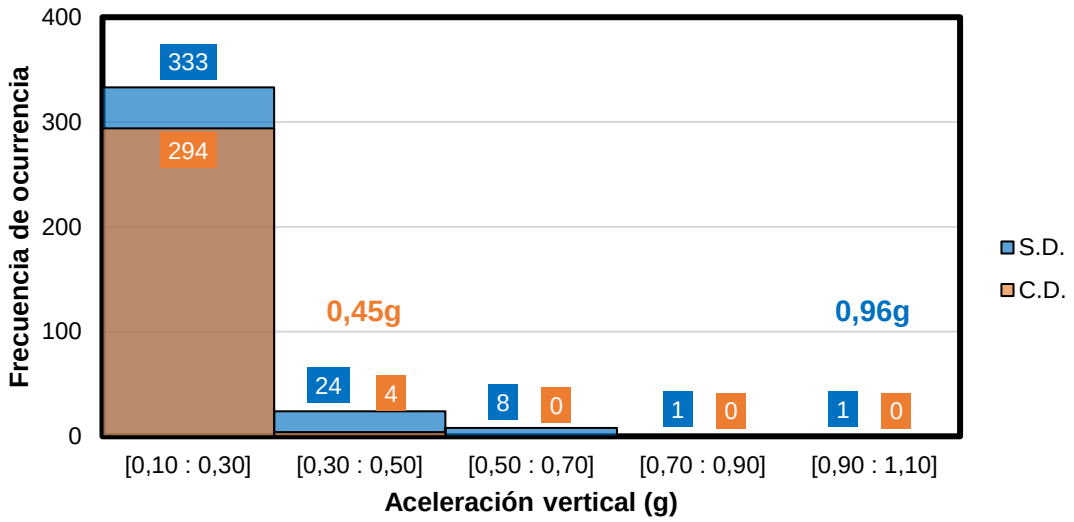


Ilustración 22-4: Histograma de aceleración vertical registrada en conducción sin (azul) y con detector de baches

Realizado por: Gunsha, E. 2022

En la Ilustración 22-4 se aprecia el registro de las aceleraciones causadas por las caídas del vehículo en baches y otras depresiones presentes en la capa asfáltica, donde para la conducción sin el detector se llegan a generar aceleraciones verticales entre 0,10g y 1,10g, y cuando se incluye el detector, este rango logra descender a 0,10g y 0,50g. Además, se tiene un valor máximo, 0,96g y 0,45g para la conducción sin y con el detector de baches respectivamente, donde de igual manera se percibe que la alerta otorgada por el sistema permite afrontar de mejor manera los baches.

Sin embargo, cuando se analizan dichos valores máximos debe indicarse que estos no se producen en una misma zona del recorrido, lo que corrobora la influencia de las decisiones del conductor para afrontar los baches sobre estos resultados.

Por otro lado, tomando como referencia la conducción sin el detector de baches, se observa una reducción en la frecuencia de ocurrencia de las aceleraciones verticales al implementar el detector, pasando así de 367 casos a 298, logrando así un descenso sobre esta métrica de 18,80%.

4.5.2. Aceleración longitudinal generada durante conducción con y sin el sistema de detección de baches

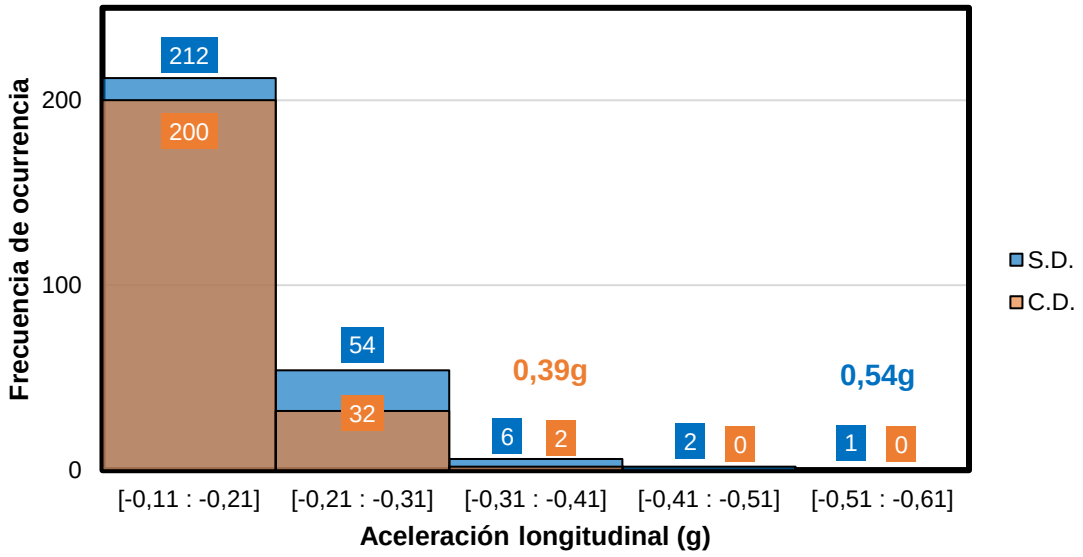


Ilustración 23-4: Histograma de aceleración longitudinal registrada en conducción sin (azul) y con detector de baches

Realizado por: Gunsha, E. 2022

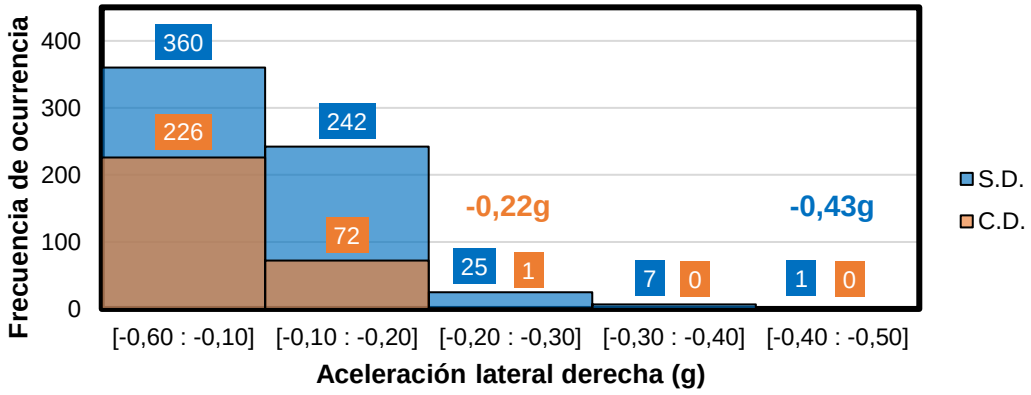
En la ilustración 23-4 se aprecian las aceleraciones longitudinales generadas por la ocurrencia de frenadas durante una conducción sin y con el detector de baches, donde para el primer caso las frenadas más pronunciadas se hallan dentro de -0,11g y -0,61g, mientras que, en el segundo caso este rango se reduce a valores entre -0,11g y -0,41g. Esto se replica en los valores máximos

registrados, donde para la conducción sin el detector de baches se alcanza -0,54g, y al incluir el detector se tiene -0,39g; de manera que la alerta otorgada por el sistema posibilita la anticipación del uso del freno para evitar su aplicación repentina.

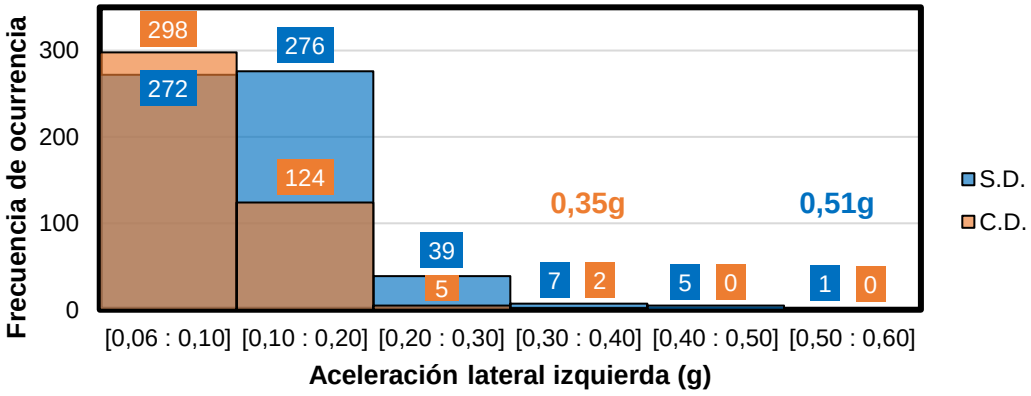
Con relación a la frecuencia de ocurrencia también se observa una disminución en la sucesión de frenadas bruscas, al pasar de 275 casos durante la conducción sin el detector, a 234 casos, logrando así una reducción equivalente al 14,91%.

4.5.3. Aceleración lateral generada durante conducción con y sin el sistema de detección de baches

En la Ilustración 24-4 se observan las aceleraciones laterales generadas durante la conducción con y sin el sistema de detección de baches, donde los valores positivos corresponden a los desplazamientos hacia la izquierda, y los negativos hacia la derecha.



a



b

Ilustración 24-4: Histograma de aceleración lateral derecha (a) e izquierda (b) registrada en conducción sin (azul) y con detector de baches

Realizado por: Gunsha, E. 2022

De esta manera, para la conducción realizada con el uso del detector se han alcanzado valores máximos de 0,35g y -0,22, que se han visto superados por los registrados en la conducción sin el detector, donde se han obtenido 0,51g y -0,43g; además para el primer caso los datos se encuentran entre 0,06g y 0,40g para la aceleración lateral izquierda, y entre -0,06g y –0,30g para la aceleración lateral derecha, mientras que el intervalo se incrementa para la conducción sin el detector de baches al encontrarse entre 0,06g y 0,60g para la aceleración lateral izquierda, y entre -0,06g y -0,50 para la aceleración lateral derecha; esto demuestra que durante la conducción en la que no se han recibido alertas, se han generado maniobras de esquiva con una intensidad ligeramente superior.

Por otro lado, en lo relacionado con la frecuencia de ocurrencia se mantiene esta disminución al comparar la conducción sin el detector con una conducción utilizando el detector de baches, pasando de 1235 casos a 728, existiendo así una reducción del 41,05%, que en comparación con las dos aceleraciones anteriores el valor sobre esta métrica es superior, lo cual podría significar una mayor utilidad del sistema para situaciones de esquiva.

Con este último apartado se ha finalizado el análisis sobre el rendimiento del sistema de detección de baches, que ha incluido desde su entrenamiento y evaluación, hasta su validación en la ruta completa de estudio, de manera que resulte posible evidenciar la utilidad de dicho sistema, así como sus fortalezas y debilidades en comparación con otras soluciones relacionadas con la visión por computadora.

CONCLUSIONES

De acuerdo con la revisión bibliográfica desarrollada, se optó por la utilización de la cámara estereoscópica Luxonis OAK-D Pro de enfoque fijo en conjunto con un mecanismo amortiguador y una ventosa, que permitió la captura de imagen y video sin pérdidas de enfoque a pesar de las vibraciones experimentadas en la sección de análisis de la vía Riobamba – El Arenal. Además, se escogió el modelo de detección de objetos YOLOR-P6, debido a su precisión y velocidad de detección superior a los modelos YOLO v4 y YOLOX, también utilizados para este tipo de aplicaciones.

Con la información recolectada en el 41,27% de la ruta de análisis, se elaboró una base de datos constituida por 700 imágenes a color de 1920x1080 píxeles, en las que se registraron 1750 etiquetas para la clase bache, y 2000 para la clase parche, donde esta última clase fue incorporada para utilizarse como filtro en la validación del modelo de detección de baches, a fin de evitar detecciones erróneas sobre parches recientes. Además, de la base de datos se destinaron de forma aleatoria 630 imágenes para el entrenamiento de la red neuronal, y las 70 restantes, para su evaluación.

El entrenamiento y evaluación de la red neuronal se ejecutó por 24 horas y 19 minutos, utilizando la tarjeta gráfica NVIDIA GeForce RTX 2070 Max-Q Design 8GB, bajo una configuración de 300 épocas y un tamaño de lote de 1; lo que permitió obtener un modelo de detección de baches y parches con una precisión de 49,39%, exhaustividad de 80,09%, mAP 50%-95% de 36,70% y mAP 50% de 71,77%, correspondiente a los pesos generados en la última época, cuya selección obedeció a la estabilización de la curva de precisión a partir de la época 250.

Al ejecutar el modelo de detección de baches en la VPU Myriad-X del dispositivo Luxonis OAK-D Pro, se obtuvo una velocidad de funcionamiento inferior a los 5fps debido al elevado peso informático de la red neuronal resultante en comparación con los modelos probados por el fabricante del dispositivo; además, no se obtuvieron detecciones de baches, a causa de la modificación del tamaño de las imágenes de entrada a 384x384 píxeles que se realizó por los inconvenientes de compatibilidad en los procesos de conversión de formato .pt a .blob, ya que la red neuronal fue entrenada con imágenes de 1920x1080 píxeles.

Para la validación del modelo de detección baches en toda la ruta de análisis, se utilizó la misma tarjeta gráfica empleada en el entrenamiento y evaluación, lo que permitió obtener una velocidad de funcionamiento de 30fps y un tiempo de inferencia menor a 0,05 segundos; así como ambos vehículos de prueba, obteniéndose una precisión de 87,84% y exhaustividad de 67,24% con el

crossover Renault Stepway, y 83,73% y 69,87%, con el SUV Toyota Corrolla Cross; donde se pudo determinar que la diferencia de dimensiones entre ambos vehículos no afecta a las métricas mencionadas, sino que estas se ven perjudicadas directamente por las condiciones climáticas y la aparición de parches recientes, cuya coloración es más oscura que la presentada por los parches registrados para la base de datos. Además, el mapa de profundidad generado en esta prueba no contribuyó a la identificación de baches, ya que la percepción de profundidad registrada para la capa asfáltica y la deformación mencionada era prácticamente la misma, e incluso existía una pérdida de información a partir de los 15 metros.

En la comparación de los datos registrados por los acelerómetros en los ejes X, Y y Z durante una conducción con y sin el detector de baches, se obtuvo que la alerta generada por el sistema permitió reducir en 18,80% la ocurrencia de aceleraciones en el eje vertical registradas ante la caída en un bache; en 14,91%, las aceleraciones ocurridas sobre el eje longitudinal producto de una frenada; y en 41,05%, las aceleraciones laterales registradas durante maniobras de esquiva hacia la izquierda o hacia la derecha. Aunque para esta prueba debe recordarse la influencia que representa el estilo de conducción, ya que el sistema únicamente lanza una alerta ante la detección de un bache, pero es el conductor el que termina obedeciendo o no a la advertencia.

RECOMENDACIONES

De acuerdo con los resultados obtenidos, se recomienda incluir otras rutas de análisis para robustecer la base de datos sin necesidad de ampliar el porcentaje de las zonas de recolección, y de este modo mejorar los valores de precisión, exhaustividad, mAP 50%-95% y mAP 50%, obtenidos en el entrenamiento y evaluación de la red neuronal.

Debido a la incidencia de las condiciones climáticas sobre la precisión del modelo entrenado, se recomienda experimentar con una base de datos generada bajo climas diferentes al utilizado en el presente proyecto técnico, de manera que puedan realizarse comparaciones para verificar el crecimiento o descenso de esta métrica.

A pesar de que en el presente proyecto técnico se utilizaron imágenes a color, con la finalidad de que la diferencia en la coloración aporte a la detección de baches, podrían experimentarse con imágenes monocromáticas, para evaluar el rendimiento del sistema y posibilitar su implementación con cámaras de visión nocturna.

Dado que el mapa de profundidad generado por visión estéreo activa no aportó significativamente para la identificación de baches, se recomienda utilizar el enfoque de tiempo de vuelo para la generación de nubes de puntos, a fin de mejorar la precisión de la percepción de profundidad.

BIBLIOGRAFÍA

AEADE. *Anuario 2021* [en línea]. 2021 [Consulta: 6 junio 2022]. Disponible en: https://www.aeade.net/wp-content/uploads/2022/03/Anuario-Aeade-2021.pdf

ALPHA. *Make Sense* [en línea]. 2022. [Consulta: 6 junio 2022]. Disponible en: https://www.makesense.ai/

ANACONDA. Anaconda Navigator [en línea]. 2022. [Consulta: 6 junio 2022]. Disponible en: https://anaconda.org/anaconda/anaconda-navigator

ANAND, Aditya; et al. "Intelligent Vehicle Speed Controlling and Pothole Detection System". *E3S Web of Conferences* [en línea], 2020, 170, pp. 1–5. [Consulta: 23 abril 2022]. ISSN 2267-1242. Disponible en: https://www.e3s-conferences.org/articles/e3sconf/abs/2020/30/e3sconf_evf2020_02010/e3sconf_evf2020_02010.html

ANAND, Sukhad; et al. "Crack-pot: Autonomous Road Crack and Pothole Detection". *International Conference on Digital Image Computing: Techniques and Applications, DICTA 2018* [en línea], 2019, pp. 1–6. [Consulta: 10 mayo 2022]. Disponible en: https://ieeexplore.ieee.org/document/8615819

ANANDHALLI, Mallikarjun; et al. "Indian pothole detection based on CNN and anchor-based deep learning method". *International Journal of Information Technology (Singapore)* [en línea], 2022, pp. 1–11. [Consulta: 11 mayo 2022]. ISSN 2511-2112. Disponible en: https://link.springer.com/article/10.1007/s41870-022-00881-5

APEN. *¿Qué es Microsoft Excel?* [en línea]. 2022. [Consulta: 6 junio 2022]. Disponible en: https://apen.es/glosario-de-informatica/microsoft-excel/

CHANG, Chen-Ming. *pytorch_YOLO_OpenVINO_demo* [en línea]. 23 febrero 2021. [Consulta: 8 junio 2022]. Disponible en: https://github.com/Chen-MingChang/pytorch_YOLO_OpenVINO_demo

DELKIN. *Fat Gecko Archives* [en línea]. 2022. [Consulta: 5 junio 2022]. Disponible en: https://www.delkindevices.com/products/accessories/fat-gecko/

DHIMAN, Amita; & KLETTE, Reinhard. "Pothole Detection Using Computer Vision and Learning". *IEEE Transactions on Intelligent Transportation Systems* [en línea], 2020, 21 (8), pp. 3536–3550. [Consulta: 6 abril 2022]. ISSN 1558-0016. Disponible en: https://ieeexplore.ieee.org/document/8788687

EL COMERCIO. *Conexión vial entre 3 provincias tiene varios daños en la calzada* [en línea]. 23 diciembre 2020. [Consulta: 6 abril 2022]. Disponible en: https://www.pressreader.com/ecuador/el-comercio-ecuador/20201123/281651077656228

EL UNIVERSO. *Llegar a Riobamba y Guaranda por carretera es toda una odisea* [en línea]. 28 octubre 2007. [Consulta: 11 abril 2022]. Disponible en: https://www.eluniverso.com/2007/10/28/0001/12/741A29A867E64923AD71B2AFC1C07210.html/

FAN, Rui; et al. "Pothole Detection Based on Disparity Transformation and Road Surface Modeling". *IEEE Transactions on Image Processing* [en línea], 2020, 29, pp. 897–908. [Consulta: 6 mayo 2022]. ISSN 1941-0042. Disponible en: https://ieeexplore.ieee.org/document/8809907

GANDHI, Rohith. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms* [en línea]. 9 julio 2018. [Consulta: 26 mayo 2022]. Disponible en: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

GARCILLANOSA, Mae M.; et al. "Smart Detection and Reporting of Potholes via Image Processing using Raspberry-Pi Microcontroller". *10th International Conference on Knowledge and Smart Technology: Cybernetics in the Next Decades, KST 2018* [en línea], 2018, pp. 191–195. [Consulta: 6 mayo 2022]. Disponible en: https://ieeexplore.ieee.org/document/8426203

GONZÁLEZ, Ana; et al. *Técnicas y algoritmos básicos de visión artificial*. La Rioja: Universidad de La Rioja, 2006. ISBN 84-689-9345-X, pp. 15-17.

HAYKIN, Simon. *Neural Networks and Learning Machines*. 3ª ed. New Jersey: Pearson Education, 2009. ISBN 978-0-13-147139-9, pp. 21-24.

HOYO, Rubén. *¿Qué asistencias a la conducción son obligatorias en Europa a partir de 2022?* [en línea]. 15 junio 2021. [Consulta: 24 abril 2022]. Disponible en: https://noticias.espanol.autocosmos.com/2021/06/15/que-asistencias-a-la-conduccion-sonobligatorias-en-europa-a-partir-de-2022

JING, Xu. *Modelo de entrenamiento de conjunto de datos personalizado de YOLOR* [en línea]. 28 agosto 2021. [Consulta: 8 junio 2022]. Disponible en: https://github.com/DataXujing/YOLOR

KEMPAIAH, Bharani Ujjaini; et al. "A Deep Learning Approach for Speed Bump and Pothole Detection Using Sensor Data". *Lecture Notes in Electrical Engineering* [en línea], 2022, pp. 73–85. [Consulta: 14 mayo 2022]. ISSN 1876-1119. Disponible en: https://link.springer.com/chapter/10.1007/978-981-16-1338-8_7

KHANEJA, Nikhil K.; et al. "Pothole detection and prediction using sensors and machine learning". *11th International Conference on Advances in Computing, Control, and Telecommunication Technologies, ACT 2020,* 2020, pp. 1–8.

KO, Daiu. *Depth cameras and RGB-D camera SLAM* [en línea]. 11 junio 2020. [Consulta: 25 mayo 2022]. Disponible en: https://www.kudan.io/archives/517

KOID TENG, Hye; et al. *A Guide To The Visual Assessment of Flexible Pavement Surface Conditions* [en línea]. Malasia: Departamento de Obras Públicas de Malasia, 2019. [Consulta: 29 abril 2022]. Disponible en: https://cupdf.com/document/a-guide-to-the-visualassessement-of-flexible-pavement-surface-conditions-jkr-20709-2060-92.html?page=23

KUMAR, Abhishek; et al. "A modern pothole detection technique using deep learning".*2nd International Conference on Data, Engineering and Applications, IDEA 2020* [en línea], 2020, pp. 1–5. [Consulta: 10 mayo 2022]. Disponible en: https://ieeexplore.ieee.org/document/9170705

LUXONIS. *DepthAI Python API utilities, examples, and tutorials* [en línea]. 27 octubre 2019. [Consulta: 8 junio 2022]. Disponible en: https://github.com/luxonis/depthai

LUXONIS. *OAK-D Pro* [en línea]. 2021. [Consulta: 5 junio 2022]. Disponible en: https://docs.luxonis.com/projects/hardware/en/latest/pages/DM9098pro.html

LUXONIS. *RGB Full Resolution Saver* [en línea]. 2022a. [Consulta: 6 junio 2022]. Disponible en: https://docs.luxonis.com/projects/api/en/latest/samples/VideoEncoder/rgb_full_resolution_saver/

LUXONIS. *RGB & TinyYolo with spatial data* [en línea]. 2022b. [Consulta: 6 junio 2022]. Disponible en: https://docs.luxonis.com/projects/api/en/latest/samples/SpatialDetection/spatial_tiny_yolo/

MANDAL, Ratna; et al. "City Traffic Speed Characterization Based on City Road Surface Quality". *Lecture Notes in Networks and Systems* [en línea], 2022, 291, pp. 515–524. [Consulta: 23 abril 2022]. ISSN 2367-3389. Disponible en: https://link.springer.com/chapter/ 10.1007/ 978-981-16-4284-5_45

MANZANARES, Albert. Detector de baches con Deep Learning (Trabajo de Fin de Grado). [en línea] Universitat Pompeu Fabra, Ingeniería Audiovisual. Barcelona-España. 2019. pp. 3-55 [Consulta: 12 abril 2022]. Disponible en: https://repositori.upf.edu/bitstream/handle/10230/ 42402/Manzanares_2019.pdf?sequence=1&isAllowed=y

MINISTERIO DEL AMBIENTE AGUA Y TRANSICIÓN ECOLÓGICA. *6 245 turistas visitaron áreas protegidas de Chimborazo* [en línea]. 27 febrero 2015. [Consulta: 11 mayo 2022]. Disponible en: https://www.ambiente.gob.ec/6-245-turistas-visitaron-areas-protegidas-dechimborazo/

MINISTERIO DEL AMBIENTE AGUA Y TRANSICIÓN ECOLÓGICA. *Reserva de Producción de Fauna Chimborazo, recibió a cientos de turistas por feriado de Semana Santa 2015* [en línea]. 9 abril 2015. [Consulta: 11 mayo 2022]. Disponible en: https://www.ambiente.gob.ec/reserva-de-produccion-de-fauna-chimborazo-recibio-a-cientos-deturistas-por-feriado-de-semana-santa-2015/

MOLINA, Boanerges. *El trabajo interinstitucional permite el mejoramiento de la vía San Juan – El Arenal que conecta a Riobamba y Guaranda* [en línea]. 21 febrero 2022. [Consulta: 10 abril 2022]. Disponible en: https://www.obraspublicas.gob.ec/el-trabajointerinstitucional-permite-el-mejoramiento-de-la-via-san-juan-el-arenal-que-conecta-ariobamba-y-guaranda/

MUTLU AYDIN, Metin; & TOPAL, Ali. "Effect of road surface deformations on lateral lane utilization and longitudinal driving behaviours". *Transport* [en línea], 2016, 31(2), pp. 192–201. [Consulta: 24 abril 2022]. ISSN 1648-3480. Disponible en: https://journals.vilniustech.lt/ index.php/Transport/article/view/1455

NEVI-12 – MTOP. *Norma Ecuatoriana Vial. Volumen N°2 – Libro A. Norma para estudios viales.*

OBD2 ELM327. *Información sobre los dispositivos ELM327* [en línea]. 2022. [Consulta: 6 junio 2022]. Disponible en: https://obd2-elm327.es/elm327-informaciondispositivos

PANAVIAL. *Los límites de velocidad en Ecuador* [en línea]. 2022. [Consulta: 11 abril 2022]. Disponible en: https://www.panavial.com/limites-velocidad-ecuador/

PAWAR, Kshitij; et al. "Efficient pothole detection using smartphone sensors". *ITM Web of Conferences* [en línea], 2020, 32, pp. 3–13. [Consulta: 26 abril 2022]. Disponible en: https://www.itm-conferences.org/articles/itmconf/pdf/2020/02/itmconf_icacc2020_03013.pdf

PEZO, Diego; & BARZOLA, Wladimir. Desarrollo de un sistema de monitoreo de la calidad de superficie de carreteras usando la red de usuarios móviles (Informe de materia integradora). [en línea] Escuela Superior Politécnica del Litoral, Ingeniería en Telemática. Guayaquil-Ecuador. 2017. pp. 8-19 [Consulta: 11 abril 2022]. Disponible en: https://www.dspace.espol.edu.ec/xmlui/bitstream/handle/123456789/44475/D-106436.pdf?sequence=1&isAllowed=y

RASYID, Alfandino; et al. "Pothole Visual Detection using Machine Learning Method integrated with Internet of Thing Video Streaming Platform". *IES 2019 - International Electronics Symposium: The Role of Techno-Intelligence in Creating an Open Energy System Towards Energy Democracy, Proceedings* [en línea], 2019, pp. 672–675. [Consulta: 2 mayo 2022]. Disponible en: https://ieeexplore.ieee.org/document/8901626

REDMON, Joseph; & FARHADI, Ali. "YOLOv3: An Incremental Improvement". [en línea], 2018, pp. 1–6. [Consulta: 6 junio 2022]. Disponible en: https://arxiv.org/abs/1804.02767v1

REZA KASHYZADEH, Kazem; & AMIRI, Nima. "Experimental Study of the Effect of Vehicle Velocity on the Ride Comfort of a Car on a Road with Different Types of Roughness". *Lecture Notes in Electrical Engineering* [en línea], 2022, pp. 593–604. [Consulta: 30 abril 2022]. ISSN 1876-1119. Disponible en: https://link.springer.com/chapter/10.1007/978-981-33-4597-3_54

ROJAS, Víctor. Diseño de un sistema electrónico de detección de baches para asistencia vehicular (Trabajo de Titulación). [en línea] Universidad de Guayaquil, Ingeniería en Teleinformática. Guayaquil-Ecuador. 2018. pp. 16-20, 51-74 [Consulta: 12 abril 2022]. Disponible en: http://repositorio.ug.edu.ec/bitstream/redug/36304/1/Tesis_PDF.pdf

SAE J3016_202104. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.*

SHAH, Sandeep; & DESHMUKH, Chandrakant. "Pothole and Bump detection using Convolution Neural Networks". *IEEE Transportation Electrification Conference, ITEC-India 2019* [en línea], 2019, pp. 1–4. [Consulta: 24 mayo 2022]. Disponible en: https://ieeexplore.ieee.org/document/9080761

SINGLA, Aarohi. *YOLOR on a Custom Dataset | Object detection using YOLOR* [en línea]. 18 marzo 2022. [Consulta: 9 mayo 2022]. Disponible en: https://www.youtube.com/watch?v=h_Ookqy0K2Y

SZELISKI, Richard. *Computer Vision. Algorithms and Applications*. 2a ed. Cham: Springer Nature Switzerland AG, 2022. ISBN 978-3-030-34372-9, pp. 1-26, 295-301.

WANG, Chien-Yao; et al. "You Only Learn One Representation: Unified Network for Multiple Tasks". [en línea], 2021, pp. 1–11. [Consulta: 6 junio 2022]. Disponible en: https://arxiv.org/abs/2105.04206v1

WU, Chao; et al. "An automated machine-learning approach for road pothole detection using smartphone sensor data". *Sensors (Switzerland)* [en línea], 2020, 20(19), pp. 1–23. [Consulta: 24 abril 2022]. ISSN 1424-8220. Disponible en: https://www.mdpi.com/1424-8220/20/19/5564/htm

XIAN, Daque. *onnx-simplifier: Simplify your onnx model* [en línea]. 1 abril 2019. [Consulta: 9 junio 2022]. Disponible en: https://github.com/daquexian/onnx-simplifier

YANFENG, Liu. *The Confusing Metrics of AP and mAP for Object Detection / Instance Segmentation* [en línea]. 25 octubre 2018. [Consulta: 6 junio 2022]. Disponible en: https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef

ANEXOS

ANEXO A: ALGORITMO DE CAPTURA DE IMÁGENES PARA LA BASE DE DATOS

```
#!/usr/bin/env python3
import time
from pathlib import Path
import cv2
import depthai as dai

# Create pipeline
pipeline = dai.Pipeline()

# Define sources and outputs
camRgb = pipeline.create(dai.node.ColorCamera)
videoEnc = pipeline.create(dai.node.VideoEncoder)
xoutJpeg = pipeline.create(dai.node.XLinkOut)
xoutRgb = pipeline.create(dai.node.XLinkOut)
xoutJpeg.setStreamName("jpeg")
xoutRgb.setStreamName("rgb")

# Properties
camRgb.setBoardSocket(dai.CameraBoardSocket.RGB)
camRgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_4_K)
videoEnc.setDefaultProfilePreset(camRgb.getFps(),
dai.VideoEncoderProperties.Profile.MJPEG)

# Linking
camRgb.video.link(xoutRgb.input)
camRgb.video.link(videoEnc.input)
videoEnc.bitstream.link(xoutJpeg.input)

# Connect to device and start pipeline
with dai.Device(pipeline) as device:

    # Output queue will be used to get the rgb frames from the output defined above
    qRgb = device.getOutputQueue(name="rgb", maxSize=30, blocking=False)
    qJpeg = device.getOutputQueue(name="jpeg", maxSize=30, blocking=True)

    # Make sure the destination path is present before starting to store the examples
    dirName = "rgb_data"
    Path(dirName).mkdir(parents=True, exist_ok=True)
    while True:
        inRgb = qRgb.tryGet()  # Non-blocking call, will return a new data that has arrived or
None otherwise

        if inRgb is not None:
            cv2.imshow("rgb", inRgb.getCvFrame())
        for encFrame in qJpeg.tryGetAll():
            with open(f"{dirName}/{int(time.time() * 1000)}.jpeg", "wb") as f:
                f.write(bytearray(encFrame.getData()))

        if cv2.waitKey(1) == ord('q'):
            break
```

ANEXO B: ALGORITMO PARA DISTRIBUCIÓN ALEATORIA DE IMÁGENES PARA ENTRENAMIENTO Y PRUEBA DEL MODELO DE DETECCIÓN DE OBJETOS

```
import glob
import os
import numpy as np
import sys

current_dir = "D:\yolor\dataset\original_dataset"
split_pct = 10;

file_train = open("dataset/train.txt", "w")
file_val = open("dataset/test.txt", "w")
counter = 1
index_test = round(100 / split_pct)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpeg")):
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))
    if counter == index_test:
        counter = 1
        file_val.write(current_dir + "/" + title + '.jpeg' + "\n")
    else:
        file_train.write(current_dir + "/" + title + '.jpeg' + "\n")
        counter = counter + 1
file_train.close()
file_val.close()
```

ANEXO C: ALGORITMO PARA EJECUCIÓN DE RED NEURONAL CON DATOS ESPACIALES EN DISPOSITIVO LUXONIS APLICABLE PARA MODELOS YOLO

```
#!/usr/bin/env python3

from pathlib import Path
import sys
import cv2
import depthai as dai
import numpy as np
import time

'''
Spatial Tiny-yolo example
  Performs inference on RGB camera and retrieves spatial location coordinates: x,y,z relative to the center of depth map.
  Can be used for tiny-yolo-v3 or tiny-yolo-v4 networks
'''

# Get argument first
nnBlobPath = str((Path(__file__).parent / Path('../models/yolo-v4-tiny-tf_openvino_2021.4_6shave.blob')).resolve().absolute())
if 1 < len(sys.argv):
    arg = sys.argv[1]
    if arg == "yolo3":
        nnBlobPath = str((Path(__file__).parent / Path('../models/yolo-v3-tiny-tf_openvino_2021.4_6shave.blob')).resolve().absolute())
    elif arg == "yolo4":
        nnBlobPath = str((Path(__file__).parent / Path('../models/yolo-v4-tiny-tf_openvino_2021.4_6shave.blob')).resolve().absolute())
    else:
        nnBlobPath = arg
else:
    print("Using Tiny YoloV4 model. If you wish to use Tiny YOLOv3, call 'tiny_yolo.py yolo3'")

if not Path(nnBlobPath).exists():
    import sys
    raise FileNotFoundError(f'Required file/s not found, please run "{sys.executable} install_requirements.py"')

# Tiny yolo v3/4 label texts
labelMap = [
    "person",        "bicycle",    "car",           "motorbike",     "aeroplane",   "bus",           "train",
    "truck",         "boat",       "traffic light", "fire hydrant",  "stop sign",   "parking meter", "bench",
    "bird",          "cat",        "dog",           "horse",         "sheep",       "cow",           "elephant",
    "bear",          "zebra",      "giraffe",       "backpack",      "umbrella",    "handbag",       "tie",
    "suitcase",      "frisbee",    "skis",          "snowboard",     "sports ball", "kite",          "baseball bat",
    "baseball glove", "skateboard", "surfboard",     "tennis racket", "bottle",      "wine glass",    "cup",
    "fork",          "knife",      "spoon",         "bowl",          "banana",      "apple",         "sandwich",
    "orange",        "broccoli",   "carrot",        "hot dog",       "pizza",       "donut",         "cake",
    "chair",         "sofa",       "pottedplant",   "bed",           "diningtable", "toilet",        "tvmonitor",
```

```
    "laptop",       "mouse",     "remote",       "keyboard",     "cell phone",  "microwave",
"oven",
    "toaster",      "sink",      "refrigerator",  "book",        "clock",      "vase",        "scissors",
    "teddy bear",     "hair drier", "toothbrush"
]

syncNN = True

# Create pipeline
pipeline = dai.Pipeline()

# Define sources and outputs
camRgb = pipeline.create(dai.node.ColorCamera)
spatialDetectionNetwork = pipeline.create(dai.node.YoloSpatialDetectionNetwork)
monoLeft = pipeline.create(dai.node.MonoCamera)
monoRight = pipeline.create(dai.node.MonoCamera)
stereo = pipeline.create(dai.node.StereoDepth)
nnNetworkOut = pipeline.create(dai.node.XLinkOut)
xoutRgb = pipeline.create(dai.node.XLinkOut)
xoutNN = pipeline.create(dai.node.XLinkOut)
xoutBoundingBoxDepthMapping = pipeline.create(dai.node.XLinkOut)
xoutDepth = pipeline.create(dai.node.XLinkOut)
xoutRgb.setStreamName("rgb")
xoutNN.setStreamName("detections")
xoutBoundingBoxDepthMapping.setStreamName("boundingBoxDepthMapping")
xoutDepth.setStreamName("depth")
nnNetworkOut.setStreamName("nnNetwork")

# Properties
camRgb.setPreviewSize(416, 416)
camRgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_1080_P)
camRgb.setInterleaved(False)
camRgb.setColorOrder(dai.ColorCameraProperties.ColorOrder.BGR)

monoLeft.setResolution(dai.MonoCameraProperties.SensorResolution.THE_400_P)
monoLeft.setBoardSocket(dai.CameraBoardSocket.LEFT)
monoRight.setResolution(dai.MonoCameraProperties.SensorResolution.THE_400_P)
monoRight.setBoardSocket(dai.CameraBoardSocket.RIGHT)

# setting node configs
stereo.setDefaultProfilePreset(dai.node.StereoDepth.PresetMode.HIGH_DENSITY)
# Align depth map to the perspective of RGB camera, on which inference is done
stereo.setDepthAlign(dai.CameraBoardSocket.RGB)
stereo.setOutputSize(monoLeft.getResolutionWidth(), monoLeft.getResolutionHeight())

spatialDetectionNetwork.setBlobPath(nnBlobPath)
spatialDetectionNetwork.setConfidenceThreshold(0.5)
spatialDetectionNetwork.input.setBlocking(False)
spatialDetectionNetwork.setBoundingBoxScaleFactor(0.5)
spatialDetectionNetwork.setDepthLowerThreshold(100)
spatialDetectionNetwork.setDepthUpperThreshold(5000)

# Yolo specific parameters
spatialDetectionNetwork.setNumClasses(80)
spatialDetectionNetwork.setCoordinateSize(4)
```

```
spatialDetectionNetwork.setAnchors([10,14, 23,27, 37,58, 81,82, 135,169, 344,319])
spatialDetectionNetwork.setAnchorMasks({ "side26": [1,2,3], "side13": [3,4,5] })
spatialDetectionNetwork.setIouThreshold(0.5)

# Linking
monoLeft.out.link(stereo.left)
monoRight.out.link(stereo.right)

camRgb.preview.link(spatialDetectionNetwork.input)
if syncNN:
    spatialDetectionNetwork.passthrough.link(xoutRgb.input)
else:
    camRgb.preview.link(xoutRgb.input)

spatialDetectionNetwork.out.link(xoutNN.input)
spatialDetectionNetwork.boundingBoxMapping.link(xoutBoundingBoxDepthMapping.input)
stereo.depth.link(spatialDetectionNetwork.inputDepth)
spatialDetectionNetwork.passthroughDepth.link(xoutDepth.input)
spatialDetectionNetwork.outNetwork.link(nnNetworkOut.input)

# Connect to device and start pipeline
with dai.Device(pipeline) as device:

    # Output queues will be used to get the rgb frames and nn data from the outputs defined
above
    previewQueue = device.getOutputQueue(name="rgb", maxSize=4, blocking=False)
    detectionNNQueue = device.getOutputQueue(name="detections", maxSize=4,
blocking=False)
    xoutBoundingBoxDepthMappingQueue =
device.getOutputQueue(name="boundingBoxDepthMapping", maxSize=4, blocking=False)
    depthQueue = device.getOutputQueue(name="depth", maxSize=4, blocking=False)
    networkQueue = device.getOutputQueue(name="nnNetwork", maxSize=4, blocking=False);

    startTime = time.monotonic()
    counter = 0
    fps = 0
    color = (255, 255, 255)
    printOutputLayersOnce = True

    while True:
        inPreview = previewQueue.get()
        inDet = detectionNNQueue.get()
        depth = depthQueue.get()
        inNN = networkQueue.get()

        if printOutputLayersOnce:
            toPrint = 'Output layer names:'
            for ten in inNN.getAllLayerNames():
                toPrint = f'{toPrint} {ten},'
            print(toPrint)
            printOutputLayersOnce = False;

        frame = inPreview.getCvFrame()
        depthFrame = depth.getFrame() # depthFrame values are in millimeters
```

```
        depthFrameColor = cv2.normalize(depthFrame, None, 255, 0, cv2.NORM_INF,
cv2.CV_8UC1)
        depthFrameColor = cv2.equalizeHist(depthFrameColor)
        depthFrameColor = cv2.applyColorMap(depthFrameColor, cv2.COLORMAP_HOT)

        counter+=1
        current_time = time.monotonic()
        if (current_time - startTime) > 1 :
            fps = counter / (current_time - startTime)
            counter = 0
            startTime = current_time

        detections = inDet.detections
        if len(detections) != 0:
            boundingBoxMapping = xoutBoundingBoxDepthMappingQueue.get()
            roiDatas = boundingBoxMapping.getConfigData()

            for roiData in roiDatas:
                roi = roiData.roi
                roi = roi.denormalize(depthFrameColor.shape[1], depthFrameColor.shape[0])
                topLeft = roi.topLeft()
                bottomRight = roi.bottomRight()
                xmin = int(topLeft.x)
                ymin = int(topLeft.y)
                xmax = int(bottomRight.x)
                ymax = int(bottomRight.y)

                cv2.rectangle(depthFrameColor, (xmin, ymin), (xmax, ymax), color,
cv2.FONT_HERSHEY_SCRIPT_SIMPLEX)


        # If the frame is available, draw bounding boxes on it and show the frame
        height = frame.shape[0]
        width  = frame.shape[1]
        for detection in detections:
            # Denormalize bounding box
            x1 = int(detection.xmin * width)
            x2 = int(detection.xmax * width)
            y1 = int(detection.ymin * height)
            y2 = int(detection.ymax * height)
            try:
                label = labelMap[detection.label]
            except:
                label = detection.label
            cv2.putText(frame, str(label), (x1 + 10, y1 + 20), cv2.FONT_HERSHEY_TRIPLEX,
0.5, 255)
            cv2.putText(frame, "{:.2f}".format(detection.confidence*100), (x1 + 10, y1 + 35),
cv2.FONT_HERSHEY_TRIPLEX, 0.5, 255)
            cv2.putText(frame, f"X: {int(detection.spatialCoordinates.x)} mm", (x1 + 10, y1 + 50),
cv2.FONT_HERSHEY_TRIPLEX, 0.5, 255)
            cv2.putText(frame, f"Y: {int(detection.spatialCoordinates.y)} mm", (x1 + 10, y1 + 65),
cv2.FONT_HERSHEY_TRIPLEX, 0.5, 255)
            cv2.putText(frame, f"Z: {int(detection.spatialCoordinates.z)} mm", (x1 + 10, y1 + 80),
cv2.FONT_HERSHEY_TRIPLEX, 0.5, 255)
```

```
            cv2.rectangle(frame, (x1, y1), (x2, y2), color, cv2.FONT_HERSHEY_SIMPLEX)

        cv2.putText(frame, "NN fps: {:.2f}".format(fps), (2, frame.shape[0] - 4),
cv2.FONT_HERSHEY_TRIPLEX, 0.4, color)
        cv2.imshow("depth", depthFrameColor)
        cv2.imshow("rgb", frame)

        if cv2.waitKey(1) == ord('q'):
            break
```