

Detección y Clasificación de Daños en Infraestructuras Viales mediante Visión por Computador

Gabriel Rios Sanchez, Zakaria Boudich Makran,
Sebastián Malbaceda Leyva

Grupo 24

Resumen: En este artículo se ha desarrollado un sistema de detección automática de daños en infraestructuras viales mediante técnicas de visión por computador. Se comparan dos enfoques: uno clásico, que utiliza segmentación de color y detección de bordes, y otro moderno, basado en técnicas de deep learning con el modelo YOLOv8s. Para su desarrollo se utiliza el conjunto de datos RDD2022, seleccionando las clases de daño más representativas. Mientras que el enfoque clásico plantea una clasificación binaria (daño/no daño), el modelo YOLO permite detectar y clasificar múltiples tipos de daños.

Ambos métodos se han evaluado en precisión y eficiencia. Los resultados muestran que el enfoque basado en deep learning obtiene mejores resultados, tanto en capacidad de generalización como en velocidad de inferencia, aunque con un mayor coste de entrenamiento. Se concluye que esta solución es viable para futuras implementaciones en dispositivos móviles, y se propone como trabajo futuro su validación en tiempo real.

Palabras clave: Visión por computador, detección automática de daños, deep learning, YOLOv8, segmentación de color, eficiencia computacional, Road Damage Dataset.

1. INTRODUCCIÓN

El mantenimiento de las infraestructuras viales es un componente clave para garantizar una movilidad segura y eficiente. Factores como el uso continuo, las condiciones meteorológicas o una mala construcción provocan daños como la aparición de grietas o baches que comprometen directamente a la seguridad vial. Detectarlos a tiempo permite planificar reparaciones y reducir costes.

Tradicionalmente, esta detección de daños se hacía de forma manual con ayuda de operarios o por las quejas de los propios ciudadanos, lo que suponía altos costes, mucho tiempo y puede llegar a ser peligroso en vías de difícil acceso. Como alternativa, se presenta una solución mediante visión por computador, lo que permite identificar los daños con alta precisión.

El objetivo principal es desarrollar un sistema de visión por computador capaz de detectar y clasificar automáticamente distintos tipos de daños en la superficie de carreteras. Para ello, se exploran y comparan enfoques clásicos de procesamiento de imágenes y técnicas modernas basadas en deep learning. Además de analizar la precisión, se evalúa sobre su aplicabilidad en condiciones reales y su potencial para una futura implementación en tiempo real.

2. ESTADO DEL ARTE

2.1. Datasets y competiciones públicas

En los últimos años, la inspección automatizada de carreteras ha sido impulsada por diversas competiciones públicas. Uno de los conjuntos de datos más importantes en este ámbito es el **Road Damage Dataset (RDD)**, introducido inicialmente por Hiroya Maeda y su equipo en el año 2018, compuesto únicamente por imágenes tomadas en Japón. En 2020, el dataset se amplió incorporando nuevos países como India y la República Checa. Posteriormente, en 2022, se añadieron también Noruega, Estados Unidos y China, alcanzando un total de **47.420 imágenes** [1].

Finalmente, en 2024, se presentó una nueva versión del conjunto de datos que mantiene las imágenes de los países incorporados anteriormente, pero introduce una clasificación más detallada, ampliando el número de clases a diez. Además de la precisión en la detección, también se valora la optimización de recursos para facilitar su implementación en tiempo real [2].

2.2. Métodos clásicos y modernos de visión por computador

Como primeras propuestas para resolver el problema, se utilizaron diferentes **métodos clásicos** para la detección de grietas o baches. Estas soluciones se basaban en aplicar **filtros de detección de bordes** como *Canny* o *Sobel*. Además, una buena técnica para detectar grietas lineales (longitudinales y transversales) es el uso de la **transformada de Hough** [3]. Otra técnica utilizada, es la que se usa para caracterizar texturas irregulares asociadas a daños, llamado **filtros de Gabor** [4]. El uso de estos métodos clásicos ofrecían unos resultados aceptables en determinadas condiciones, aunque mostraban una notable sensibilidad al momento de que había un cambio de iluminación o ruido.

Con el tiempo, las **redes neuronales convolucionales (CNNs)** se han convertido en una de las principales herramientas para abordar problemas complejos de visión por computador. A diferencia de las técnicas clásicas, el uso de *deep learning* ha mostrado un aprendizaje superior frente a grandes cantidades de datos, extrayendo patrones de una manera robusta. Modelos como **You Only Look Once (YOLO)**, han demostrado ser capaces de detectar y clasificar múltiples tipos de daños viales a partir de imágenes capturadas en tiempo real.

En el año 2018, Maeda et al. propusieron una solución basada en *deep learning* para poder detectar los daños en carreteras [5]. Su metodología destacó por su capacidad de procesar imágenes de manera eficiente, lo que permitía una implementación en tiempo real.

Más recientemente, en el año 2023, Zhang et al. exploraron modelos basados en **arquitecturas Transformer** para la resolución de este problema [6]. Sus resultados fueron prometedores, ya que mostraron una mejora respecto a los métodos anteriores. Sin embargo, esta solución presentaba un **mayor consumo de memoria** y, en consecuencia, una **menor eficiencia computacional** en el procesamiento de imágenes. Este tipo de enfoque se aleja del objetivo a resolver, ya que se busca poder desarrollar e implementar una solución en tiempo real para poder optimizar los costes operativos.

3. METODOLOGÍA

3.1. Datos utilizados

Para el desarrollo del sistema se utilizará el *Road Damage Dataset* 2022 (RDD2022). Este conjunto de datos está compuesto por **47.420 imágenes** capturadas en seis países: Japón, India, República Checa, Noruega, Estados Unidos y China.

Cada imagen está asociada a un archivo de anotaciones en formato **Pascal VOC** (.xml), que contiene una o varias cajas delimitadoras que indican tanto la ubicación como el tipo de daño en la imagen. En total, se identifican **nueve clases distintas** descritas en la Tabla 1.

Código	Tipo de daño
D00	Grieta longitudinal
D01	Falla lineal (unión longitudinal)
D10	Grieta transversal
D11	Falla lineal (unión transversal)
D20	Grieta tipo piel de cocodrilo (Alligator Crack)
D40	Bache (Pothole)
D43	Difuminado en paso de peatones (Crosswalk Blur)
D44	Difuminado en línea blanca (White Line Blur)
D50	Otros daños en carretera

Tabla 1: Tipos de daños presentes en el dataset RDD2022.

Con el objetivo de alinearnos con la competición oficial y centrarnos en los daños más representativos, en este trabajo se seleccionan únicamente las clases D00, D10, D20 y D40.

Para el diseño del sistema de **visión clásica**, utilizamos únicamente las imágenes de Noruega, ya que supone un reto adicional al tener presencia de nieve, hielo y sombras muy marcadas. Además, la textura y composición del pavimento resulta que es muy similar a los tramos viales que nos podemos encontrar en España, aportando relevancia práctica.

Cabe mencionar que, en este sistema clásico, se ha simplificado el problema original transformándolo en uno de **clasificación binaria**, donde el objetivo es únicamente detectar si una imagen contiene algún tipo de daño o no. Esta decisión responde tanto a la naturaleza de las técnicas empleadas como a la necesidad de reducir la complejidad del problema, permitiendo un análisis más robusto sobre un subconjunto específico del dataset.

Aunque el dataset ya viene dividido en `train` y `test`, se aplicará una nueva **división estratificada** para obtener un subconjunto de validación, siguiendo una proporción 80-20. Esto garantiza que la distribución original de clases sea coherente con la del conjunto original.

Gracias a esta partición, se permitirá realizar la búsqueda de los mejores hiperparámetros mediante *Random Search*, mejorando la generalización del modelo y aprovechando al máximo los recursos computacionales disponibles.

3.2. Preprocesado

El conjunto de datos original contiene imágenes de diferentes países, lo que resulta beneficioso al tener datos tan diversos. Sin embargo, las imágenes contienen hasta **cuatro resoluciones distintas** [1], lo que dificulta un tratamiento uniforme.

En general, la mayoría de las imágenes son cuadradas, lo cual facilita su procesamiento, salvo las de **Noruega**, que se capturaron con dos cámaras simultáneamente. Para unificar los datos, se

aplicó un preprocesado que normalizó las imágenes en una resolución de **512x512 píxeles**, ya que se trataba de la resolución más pequeña que había en el dataset y que conservaba una calidad visual aceptable.

Para el enfoque clásico, se recortaron las imágenes de Noruega en dos partes horizontales, eliminando la parte derecha de la imagen, ya que haciendo un análisis, no contenía partes de carretera. Un ejemplo de este tipo de imágenes se muestra en la Figura 1. Después de este proceso, todas las imágenes fueron redimensionadas a la resolución que se iba a trabajar.



Fig. 1: Ejemplo de imagen original de Noruega donde se aprecia que gran parte de la zona derecha no corresponde a la carretera.

Para la metodología basada en *deep learning*, se realizó un proceso de *letterbox* para cada imagen, adaptándolas al tamaño objetivo sin deformar su contenido y así optimizar su uso en el entrenamiento de redes neuronales.

En cuanto a las etiquetas del conjunto de `train`, y con el objetivo de facilitar su uso en ambos enfoques, se hizo una conversión de **Pascal VOC** (.xml) a archivos .txt.

En el caso de la metodología clásica, las etiquetas se adaptaron al formato (`clase xmin ymin xmax ymax`), conservando las coordenadas absolutas. Durante el proceso, también se ajustaron las posiciones en función de los recortes aplicados a las imágenes de Noruega.

Para la metodología de *deep learning*, se convirtió a formato **YOLO**, con la estructura (`clase x_center y_center width height`), normalizando todas las coordenadas respecto al tamaño de cada imagen.

3.3. Metodología clásica

En el enfoque clásico, se realiza una primera etapa de procesamiento que combina **segmentación de color** y **detección de bordes**. En primer lugar, se suavizan las imágenes con filtrado por *mean shift*, con el objetivo de reducir el ruido y preservar los bordes. A continuación, se aplica una segmentación adicional mediante **Quickshift**, lo que permite una representación más uniforme de regiones similares.

Posteriormente, se filtran regiones que corresponden al tono gris característico del pavimento. Esto se logra transformando la imagen al espacio **HSV** y aplicando **umbrales de color**, combinados con operaciones morfológicas para eliminar ruido.

Finalmente, sobre la región segmentada, se aplica el detector de bordes **Canny**, ajustado para captar detalles finos. Los contornos detectados se analizan según su área, priorizando aquellos más propensos a corresponder a grietas.

3.4. Metodología de deep learning

Para el enfoque basado en *deep learning* se utilizará el modelo **YOLOv8s**. Esta versión ofrece un equilibrio entre **precisión** y **velocidad**, diseñada para tareas de detección rápida y eficiente en tiempo real [7].

El modelo será entrenado sobre las imágenes procesadas mencionadas anteriormente, aplicando técnicas de ***data augmentation***, donde se incluirán *volteos horizontales*, *escalados* y *modificaciones de brillo* y *contraste* para simular diferentes condiciones de mala iluminación. Esto con el objetivo de mejorar la **capacidad de generalización** y **robustez del sistema**.

3.5. Postprocesado

Tras la detección, será necesario un ajuste de las coordenadas de salida para cada metodología. Este paso es fundamental para garantizar la coherencia espacial entre las predicciones generadas por ambos métodos y las coordenadas originales del dataset.

En la visión clásica, se compensará el recorte aplicado a las imágenes de Noruega y el redimensionamiento posterior, restaurando las coordenadas a su posición inicial.

En *deep learning*, se corregirá el efecto del *letterbox*, eliminando el *padding* y reescalando las predicciones a la resolución original.

3.6. Métricas de evaluación

Para la validación, se utilizará la misma métrica utilizada en la competición RDD2022: el *F1-Score* a un umbral de solape IoU=0.5 [8].

De este modo, se asegura la coherencia entre las métricas de validación internas y el criterio final de evaluación aplicado sobre el conjunto de test, donde se enviarán las predicciones para determinar el rendimiento real del sistema para cada metodología.

4. EXPERIMENTOS Y RESULTADOS

4.1. Entorno experimental

Para realizar los experimentos se ha utilizado un equipo con una **GPU NVIDIA RTX 2080**, 16 GB de RAM y un procesador Intel i7-8700K. Además, el sistema se estructuró en forma de *pipeline*, donde cada etapa está organizada de manera modular y permite la ejecución controlada de los experimentos.

4.2. Parámetros de entrenamiento

En el enfoque de *deep learning*, el entrenamiento se ha realizado ajustando los siguientes hiperparámetros relacionados con la estrategia de *data augmentation* y el proceso de entrenamiento:

- **fliplr**: probabilidad de aplicar un volteo horizontal a la imagen.
- **hsv_h, hsv_s, hsv_v**: factores de aleatoriedad aplicados sobre el matiz (hue), la saturación y el valor (brillo), respectivamente, en el espacio de color HSV.
- **scale**: rango de escalado aleatorio de la imagen.
- **batch**: tamaño del lote de imágenes utilizado en cada iteración del entrenamiento.
- **epochs**: número total de pasadas completas sobre el conjunto de entrenamiento.

Para evaluar el rendimiento y seleccionar la mejor combinación de hiperparámetros, además de prevenir el **overfitting**, se utilizó un conjunto de validación mediante la técnica de *Random Search*.

En la Tabla 2 se muestran las diferentes combinaciones de hiperparámetros evaluadas, junto con el *F1-score* obtenido para cada una de ellas.

fliplr	hsv_h	hsv_s	hsv_v	scale	batch	epochs	f1-score
0.5	0.01	0.6	0.3	0.3	8	30	0.5371
0.5	0.01	0.6	0.4	0.3	8	50	0.5459
0.5	0.015	0.7	0.3	0.5	16	50	0.5564
0.3	0.01	0.6	0.4	0.3	16	30	0.5486
0.3	0.015	0.7	0.4	0.5	8	30	0.5406
0.5	0.015	0.7	0.4	0.5	16	50	0.5597

Tabla 2: Parámetros utilizados en cada validación y resultados obtenidos. En negrita, la mejor configuración.

En el sistema de visión clásica, para definir el rango de color en el espacio **HSV**, se seleccionaron aleatoriamente 100 imágenes del conjunto de entrenamiento. En cada una de ellas se identificó manualmente una región representativa de la superficie de la carretera, a partir de la cual se extrajo el **valor medio de color** en dicho espacio.

A continuación, se calculó una **media global** a partir de los 100 vectores HSV obtenidos. Con base en este valor central, se estableció un rango de detección aplicando un margen de **±15 unidades** en cada uno de los *tres canales* (*H*, *S* y *V*). Este intervalo permitió capturar de forma robusta el tono grisáceo característico del asfalto bajo diferentes condiciones de iluminación, sin incorporar colores ajenos de la carretera.

4.3. Evaluación cuantitativa

El sistema basado en *deep learning* fue evaluado inicialmente sobre el conjunto completo de test, compuesto por las imágenes de los seis países. El modelo alcanzó un **F1-Score global de 0.67**, indicando la buena capacidad de generalización ante distintas condiciones geográficas, tipos de daños y variaciones de iluminación.

Sin embargo, para poder realizar una comparación justa con el sistema de visión clásica, que fue entrenado y ajustado con imágenes de Noruega, se evaluaron ambos métodos sobre el conjunto de prueba de dicho país.

La Tabla 3 muestra los resultados obtenidos por ambas metodologías en el conjunto de prueba de Noruega. Aunque el sistema de visión clásica aborda el problema como una clasificación binaria (daño / no daño), mientras que el enfoque basado en *deep learning* detecta y clasifica distintos tipos de daños, el modelo YOLOv8s presenta una ventaja significativa en términos de precisión y eficiencia computacional, a pesar de la mayor complejidad de la tarea.

Método	F1-Score	Tiempo de inferencia (ms/img)
Visión clásica	0.21	1537.2
YOLOv8s	0.54	32.6

Tabla 3: Comparativa de resultados entre visión por computador clásica y YOLOv8s sobre el conjunto de prueba de Noruega.

4.4. Ejemplo visual de detección

Con el objetivo de evaluar la aplicabilidad del sistema en condiciones reales, se ha evaluado el modelo sobre una imagen propia capturada con un dispositivo móvil, ajena al conjunto de datos original. La Figura 2 muestra la detección realizada por el modelo YOLOv8s, donde se observan correctamente identificados varios tipos de daños presentes en el entorno urbano.

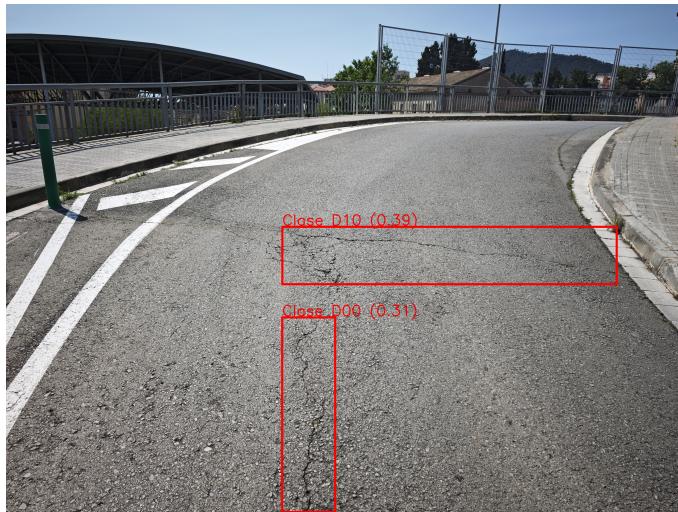


Fig. 2: Predicción del modelo sobre una imagen capturada en un entorno real. Se detecta una grieta longitudinal (clase D00) y una grieta transversal (clase D10)

4.5. Discusión de resultados

Los resultados obtenidos muestran que el enfoque basado en *deep learning* supera claramente a la metodología clásica, tanto en precisión como en tiempo de inferencia. Aunque el método clásico resuelve una versión simplificada del problema, puede resultar útil en entornos específicos y controlados, como es el caso de las imágenes capturadas en Noruega. Sin embargo, sus resultados se ven muy afectados cuando aparecen cambios de iluminación, ruido o variaciones en la textura del pavimento.

Por otro lado, el modelo YOLO ha mostrado una mayor capacidad de generalización ante distintos escenarios y tipos de daño. Sin embargo, esta robustez viene acompañada de un mayor coste computacional durante la fase de entrenamiento y validación, lo que puede ser un factor a considerar en entornos con recursos limitados.

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este trabajo se ha desarrollado y comparado un sistema de detección de daños en carreteras mediante visión por computador, utilizando dos enfoques: el clásico, basado en segmentación y detección de bordes, y otro moderno basado en *deep learning* utilizando el modelo YOLOv8s.

Los resultados obtenidos permiten afirmar que se ha cumplido con los objetivos planteados: se ha implementado un sistema capaz de detectar y clasificar automáticamente distintos tipos de daños, evaluando su precisión y eficiencia en condiciones diversas, incluidas situaciones reales. Aunque el sistema clásico resuelve una versión más sencilla del problema, el modelo basado en *deep learning* ofrece un rendimiento superior en precisión y eficiencia durante la inferencia, mostrando una mayor capacidad

de generalización ante distintas condiciones geográficas y tipos de daño.

Sin embargo, esta mejora viene acompañada de un mayor coste computacional en la fase de entrenamiento, lo que puede suponer una limitación en entornos con recursos limitados. Por otro lado, el sistema clásico, aunque sea menos preciso y con un mayor tiempo de inferencia, puede ser útil en entornos específicos donde se priorice soluciones simples.

Como trabajos futuros, se plantea la integración del sistema en plataformas móviles para su validación en tiempo real, contribuyendo así a cumplir uno de los objetivos principales del trabajo: su aplicación en escenarios reales.

REFERENCIAS

- [1] D. Arya, H. Maeda, S. Ghosh, D. Toshniwal y Y. Sekimoto, “RDD2022: A multi-national image dataset for automatic Road Damage Detection,” arXiv:2209.08538, Sep. 2022. Disponible en: <https://doi.org/10.48550/arXiv.2209.08538>.
- [2] Sekilab. “Open Road Damage Detection Challenge 2024 (ORDDC2024).” Disponible en: <https://orddc2024.sekilab.global/>. [Accedido: 20-Abr-2025].
- [3] S. Matarneh, F. Elghaish, A. Al-Ghraibah, E. Abdellatef y D.J. Edwards, “An automatic image processing based on Hough transform algorithm for pavement crack detection and classification,” *Smart and Sustainable Built Environment*, vol. 14, no. 1, pp. 1–22, 2025. Disponible en: <https://doi.org/10.1108/SASBE-01-2023-0004>.
- [4] E. Zalama, J. Gómez-García-Bermejo, R. Medina y J. Llamas, “Road Crack Detection Using Visual Features Extracted by Gabor Filters,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 5, pp. 342–358, 2014. Disponible en: <https://doi.org/10.1111/mice.12042>.
- [5] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama y H. Omata, “Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, 2018. Disponible en: <https://doi.org/10.1111/mice.12387>.
- [6] Y. Zhang y L. Zhang, “Detection of Pavement Cracks by Deep Learning Models of Transformer and UNet,” *arXiv preprint arXiv:2304.12596*, 2023. Disponible en: <https://arxiv.org/abs/2304.12596>.
- [7] Ultralytics, “YOLOv8 Documentation,” 2023. Disponible en: <https://docs.ultralytics.com>. [Accedido: 24-Abr-2025].
- [8] Sekilab, “CRDDC2022: Challenge on Road Damage Detection and Classification 2022.” Disponible en: <https://crddc2022.sekilab.global/data/>. [Accedido: 25-Abr-2025].