GRUPO BLR

VISIÓN ARTIFICIAL: HACIA EL ESTACIONAMIENTO INTELIGENTE

ISBN: 978-9907-802-80-1

Jonathan Víctor Aguilar Alvarado

2026

VISIÓN ARTIFICIAL: HACIA EL ESTACIONAMIENTO INTELIGENTE

Jonathan Víctor Aguilar Alvarado

AUTOR



Este libro ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad científica.

©Grupo Editorial BLR
Universidad Técnica de Machala
Riobamba – Ecuador
Correo: publicaciones@grupoblr.com
https://grupoblr.com/libros-investigo
REPOSITORIO



Aguilar, J. (2026) Visión artificial: Hacia el estacionamiento inteligente. Grupo Editorial BLR.

©Jonathan Víctor Aguilar Alvarado

Visión artificial: Hacia el estacionamiento inteligente.

ISBN: 978-9907-802-80-1

El copyright promueve la libertad de expresión, protege la diversidad de ideas y conocimiento, además apoya la libre expresión. Se prohíbe de manera rigurosa la producción o el almacenamiento de esta publicación, ya sea en su totalidad o en parte, está estrictamente prohibido por ley, incluyendo el diseño de la portada, así como su difusión a través de cualquiera de sus medios, ya sean electrónicos, mecánicos, ópticos, de grabación o incluso de fotocopia, sin permiso de los propietarios de los derechos de autor.

FILIACIÓN DEL AUTOR

Jonathan Víctor Aguilar Alvarado

Universidad Técnica de Machala

Correo Electrónico: jaguilar@utmachala.edu.ec

ORCID: https://orcid.org/0000-0003-0137-2618

PREFACIO

INTRODUCCIÓN DEL LIBRO

La creciente demanda de plazas de estacionamiento en entornos académicos y urbanos ha revelado la urgencia de adoptar soluciones tecnológicas que mejoren la eficiencia en la búsqueda y asignación de espacios. Dentro de este contexto, la visión artificial se consolida como una herramienta versátil para el monitoreo, conteo y localización de vehículos, ofreciendo perspectivas novedosas para reducir el congestionamiento y agilizar el tránsito.

Diversos estudios han abordado la aplicación del procesamiento de imágenes en espacios abiertos, enfatizando la detección de objetos y la optimización del flujo vehicular. Sin embargo, se ha identificado la necesidad de profundizar en la integración de estas técnicas con sistemas de control específicos para la gestión de estacionamientos, así como de disponer de recursos pedagógicos que faciliten la comprensión y adopción de este tipo de tecnologías en el ámbito académico.

Por ello, el texto se plantea como objetivo principal dar a conocer las bases conceptuales y prácticas de la visión artificial enfocada en el control de estacionamientos, garantizando una aproximación didáctica y aplicable en entornos formativos. Para lograrlo, se emplearon metodologías de aprendizaje profundo y procedimientos de validación experimental, con el fin de demostrar la eficacia de la detección, el conteo y la localización vehicular mediante algoritmos de procesamiento de imágenes.

El propósito de esta obra radica en dotar a estudiantes y docentes de un texto de asignatura que facilite el entendimiento de los fundamentos y la implementación de sistemas de visión artificial para la movilidad. Este recurso se inserta en áreas como informática, electrónica, sistemas y tecnologías de la información, impulsando el desarrollo de competencias orientadas a la innovación y la resolución de problemáticas contemporáneas.

Se abordan desde los principios teóricos de la visión por computadora hasta la puesta en marcha de proyectos piloto, pasando por la configuración de infraestructura y la evaluación de resultados en escenarios reales. Esta aproximación permite que el lector comprenda de forma sencilla el potencial de la visión artificial y los desafíos que implica su adopción.

Este libro se estructura en capítulos que exponen los fundamentos conceptuales, describen las herramientas y técnicas de procesamiento de imágenes, y muestran ejemplos de aplicación con énfasis en la planificación, el diseño y la validación de sistemas inteligentes de estacionamiento. A lo largo del texto, se pretende dotar a la comunidad académica de una base sólida para el aprendizaje y la práctica de la visión artificial, promoviendo así la formación de profesionales capaces de aportar soluciones efectivas a la creciente problemática de la movilidad en espacios educativos y urbanos.

OBJETIVO GENERAL DEL LIBRO

Desarrollar una guía teórico-práctica sobre visión artificial en la gestión de estacionamiento vehicular, dirigida a estudiantes y docentes, con miras a su aplicación en entornos reales.

ESTRUCTURA DEL LIBRO

El libro se compone de dos capítulos principales:

Capítulo 1: Fundamentos de Visión Artificial.

Capítulo 2: Implementación de un sistema de visión artificial para control de estacionamiento.

Al final de cada capítulo se presentan las referencias bibliográficas correspondientes, de acuerdo con la normativa APA 7ª edición. Además, el texto se ajusta a los estándares y requisitos contemplados para la publicación digital dentro de la Universidad Técnica de Machala.

CARACTERÍSTICAS PEDAGÓGICAS DEL LIBRO

Este libro ha sido elaborado con un enfoque teórico-práctico, con el propósito de facilitar la comprensión de los fundamentos de la visión artificial y su aplicación en situaciones reales. Los contenidos se presentan de manera progresiva, iniciando con los conceptos básicos y avanzando hacia la implementación de un sistema de visión artificial aplicado al control de estacionamientos.

Para apoyar el aprendizaje, se incorporan explicaciones conceptuales, tablas, gráficos, ejemplos de procesamiento de imágenes y fragmentos de código que permiten observar de forma práctica cómo se aplican las diferentes técnicas estudiadas. Estos recursos buscan que el lector no solo conozca los conceptos, sino que pueda relacionarlos con su aplicación dentro de un sistema computacional.

Además, los ejemplos utilizados a lo largo del libro están vinculados con escenarios de estacionamiento vehicular, lo que permite mantener una relación directa entre los fundamentos de la visión artificial y el problema práctico que se desarrolla en el libro. De esta manera, el texto puede ser utilizado como material de apoyo por estudiantes, docentes y personas interesadas en iniciarse en el estudio de la visión artificial.

ÍNDICE

PREFACIO ..... **i**

INTRODUCCIÓN DEL LIBRO ..... **i**

OBJETIVO GENERAL DEL LIBRO ..... **iii**

ESTRUCTURA DEL LIBRO ..... **iv**

CARACTERÍSTICAS PEDAGÓGICAS DEL LIBRO ..... **v**

ÍNDICE ..... **vi**

ÍNDICE DE TABLAS ..... **viii**

ÍNDICE DE FIGURAS ..... **ix**

CAPÍTULO I **13**

1 FUNDAMENTOS DE VISIÓN ARTIFICIAL **13**

1.1 Introducción 13

1.2 Objetivo 14

1.3 Historia 14

1.4 Visión artificial 16

1.4.1 Etapas del proceso de visión artificial 17

1.4.2 Adquisición 18

1.4.3 Digitalización 20

1.4.4 Preprocesamiento 21

1.4.5 Operaciones elementales 23

1.4.6 Operaciones Geométricas 33

1.4.7 Filtros Lineales 43

1.4.8 Filtro de derivadas (Gradientes) 48
1.4.9 Filtros no lineales 56
1.4.10 Operadores Lógicos o Booleanos 66
1.5 Operaciones Morfológicas 71
1.6 Procesamiento y Análisis 79
1.7 Interpretación 84
1.8 Resumen 86
CAPÍTULO II 88
2 IMPLEMENTACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA CONTROL DE ESTACIONAMIENTO 88
2.1 Introducción 88
2.2 Objetivo 90
2.3 Arquitectura del sistema 90
2.4 Adquisición de datos 91
2.5 Preprocesamiento de las imágenes 96
2.6 Entrenamiento del modelo 99
2.7 Validación y visualización de resultados 107
2.8 Resumen 109
BIBLIOGRAFÍA 111

ÍNDICE DE TABLAS

Tabla 1. Técnicas de adquisición de imágenes. 19

Tabla 2. Componentes de la digitalización. 21

Tabla 3. Operaciones fundamentales en el preprocesamiento. 22

Tabla 4. Procesamiento y análisis de imágenes. 80

Tabla 5. Interpretación de los resultados. 84

ÍNDICE DE FIGURAS

Figura 1. Hitos de la visión artificial. 14

Figura 2. Esquema de visión artificial. 16

Figura 3. Las etapas del proceso de visión artificial. 18

Figura 4. Adquisición de información digital. 18

Figura 5. Proceso de digitalización. 20

Figura 6. Imagen con ajuste de brillo. 24

Figura 7. Imagen con ajuste de contraste. 26

Figura 8. Imagen con negativo. 27

Figura 9. Imagen con transformación logarítmica. 30

Figura 10. Imagen con corrección gamma. 32

Figura 11. Imagen con traslación. 34

Figura 12. Imagen con rotación. 36

Figura 13. Imagen con escalado. 37

Figura 14. Imagen con reflexión horizontal. 39

Figura 15. Imagen con sesgado. 41

Figura 16. Transformaciones afines. 43

Figura 17. Filtro promedio. 45

Figura 18. Filtro Gaussiano. 47

Figura 19. Filtro Sobel. 50

Figura 20. Filtro Prewitt. 51

Figura 21. Filtro Roberts. 53

Figura 22. Laplaciano (2ª derivada). 54

Figura 23. Filtro Canny. 56

Figura 24. Filtro de Mediana. 58

Figura 25. Filtro de Mínimo. 60

Figura 26. Filtro de Máximo. 62

Figura 27. Filtro de Rango. 63

Figura 28. Filtro Adaptativo. 65

Figura 29. Imagen con Operación AND. 67

Figura 30. Imagen con Operación OR. 69

Figura 31. Imagen con Operación XOR. 70

Figura 32. Imagen con Erosión. 73

Figura 33. Imagen con Dilatación. 75

Figura 34. Imagen con Apertura. 76

Figura 35. Imagen con Cierre. 78

Figura 36. Procesamiento y análisis de las imágenes. 79

Figura 37. Técnicas de segmentación. 81

Figura 38. Extracción de característica. 82

Figura 39. Clasificación/Reconocimiento de imágenes. 83

Figura 40. Interpretación de imágenes. 85

Figura 41. Esquema de adquisición de datos. 91

Figura 42. Descarga de datos externos CNRPark-Patches. 93

Figura 43. Conteo de imágenes CNRPark-Patches. 94

Figura 44. Organización de las imágenes CNRPark-Patches. 95

Figura 45. Normalización de píxeles. 97

Figura 46. Mejora de contraste (CLAHE). 98

Figura 47. Reducción de ruido. 98

Figura 48. Aumento de datos (Data Augmentation). 99

Figura 49. División de datos (entrenamiento, validación y prueba). 100

Figura 50. Construcción de DataLoaders. 102

Figura 51. Modelo CNN. 104

Figura 52. Entrenamiento de la CNN. 106

Figura 53. Validación y visualización de resultados. 108

CAPÍTULO I

1 FUNDAMENTOS DE VISIÓN ARTIFICIAL

1.1 Introducción

La visión artificial permite que un sistema computacional obtenga información a partir de imágenes y videos para posteriormente procesarla e interpretarla. Su desarrollo ha permitido abordar problemas cada vez más complejos, desde el reconocimiento de objetos hasta la automatización de procesos en áreas como la industria, la salud, la seguridad y la movilidad.

Para que una imagen pueda ser utilizada dentro de un sistema de visión artificial debe pasar por diferentes etapas. El proceso comienza con la adquisición y digitalización de la información visual y continúa con tareas de preprocesamiento destinadas a mejorar la calidad de las imágenes. A partir de allí pueden aplicarse transformaciones geométricas, filtros y operaciones morfológicas que facilitan la identificación de características relevantes para su posterior análisis e interpretación.

En este capítulo se presentan los fundamentos que permiten comprender este proceso de manera progresiva. Se parte de una breve revisión de la evolución de la visión artificial y posteriormente se estudian sus principales etapas y técnicas de procesamiento de imágenes. Los ejemplos utilizados se relacionan principalmente con imágenes de vehículos y espacios de estacionamiento, con el propósito de vincular

los conceptos estudiados con la aplicación práctica que se desarrolla posteriormente en el libro.

1.2 Objetivo

- Analizar los fundamentos y las principales etapas de la visión artificial mediante el estudio de la adquisición, digitalización, preprocesamiento, procesamiento e interpretación de imágenes, relacionando estas técnicas con su aplicación en sistemas inteligentes de estacionamiento vehicular.

1.3 Historia

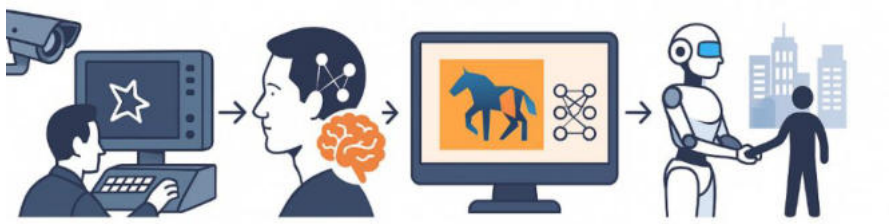


Figura 1. Hitos de la visión artificial.

Fuente: Autor.

La visión artificial nació a mediados del siglo XX, específicamente en la década de 1960, como resultado del interés científico por dotar a las máquinas de la capacidad de interpretar imágenes, imitando de alguna forma la percepción visual humana. Inicialmente, la visión artificial se enfocó en tareas básicas de reconocimiento de patrones y procesamiento elemental de imágenes, con técnicas limitadas debido a las restricciones tecnológicas de la época (Dobson, 2023). Sus primeras aplicaciones se centraron en tareas relativamente simples, como reconocer patrones

básicos en entornos controlados. Estas técnicas iniciales fueron perfeccionadas y ampliadas, impulsadas por la creciente disponibilidad de datos digitales y la evolución del poder computacional. Así surgieron nuevas técnicas, como el procesamiento digital avanzado, la segmentación y análisis de imágenes, que rápidamente encontraron utilidad en áreas industriales, médicas y de seguridad.

Durante las décadas posteriores, particularmente en los años ochenta y noventa, el avance en la capacidad de procesamiento computacional permitió el desarrollo de algoritmos más sofisticados. Esto impulsó mejoras significativas en la segmentación de imágenes, la identificación precisa de objetos y la implementación en tiempo real, favoreciendo la automatización industrial y los primeros usos médicos avanzados (Szeliski, 2021).

Sin embargo, fue con la llegada del aprendizaje profundo (deep learning) cuando la visión artificial logró una verdadera revolución, aumentando considerablemente la precisión y versatilidad en tareas complejas como la detección, segmentación y reconocimiento automático de objetos (Goodfellow et al., 2016).

Uno de los hitos más importantes ha sido la incorporación de los modelos basados en Transformers en el ámbito visual. Estas arquitecturas, desarrolladas inicialmente para el procesamiento del lenguaje natural, se mostraron efectivas para tareas visuales, permitiendo mejoras significativas frente a métodos tradicionales como las redes neuronales convolucionales (CNN) (Carion Nicolasand Massa, 2020). Así como también la introducción del Vision Transformer (ViT),

que impulsó avances importantes en el reconocimiento visual a gran escala y permitió un rendimiento superior en diversas aplicaciones (Dosovitskiy et al., 2020).

En los próximos años, la visión artificial promete seguir evolucionando gracias al desarrollo de tecnologías aún más avanzadas, como la integración plena con modelos multimodales capaces de interpretar múltiples fuentes de información simultáneamente. Estas innovaciones ampliarán significativamente la capacidad de los sistemas inteligentes para comprender contextos complejos y tomar decisiones autónomas en tiempo real. Se espera que la visión artificial continúe extendiéndose hacia ámbitos hasta ahora poco explorados, como la interacción social entre humanos y robots, ciudades inteligentes totalmente automatizadas y sistemas avanzados de realidad aumentada, consolidándose como una tecnología esencial para enfrentar los desafíos emergentes en la sociedad del futuro cercano.

1.4 Visión artificial

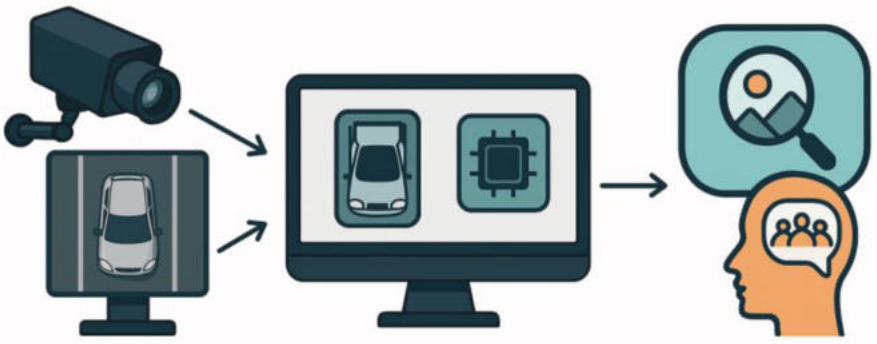


Figura 2. Esquema de visión artificial.

Fuente: Autor.

La visión artificial es un campo interdisciplinario de la inteligencia artificial que desarrolla métodos para adquirir, analizar, procesar y comprender imágenes digitales y videos (**¡Error! No se encuentra el origen de la referencia.**), permitiendo a las máquinas extraer información significativa del entorno visual, con el fin de automatizar tareas relacionadas con el reconocimiento, clasificación y seguimiento de objetos o patrones visuales.

1.4.1 Etapas del proceso de visión artificial

En un sistema de visión artificial, como se muestra en **¡Error! No se encuentra el origen de la referencia.**, el procesamiento de datos visuales se organiza en una secuencia de fases coherentes y complementarias: la adquisición captura imágenes o secuencias de video mediante sensores o cámaras, asegurando que se recoja la información visual del entorno; la digitalización convierte esas señales ópticas en datos numéricos estructurados (matrices de píxeles) que pueden ser procesados por algoritmos computacionales; el preprocesamiento aplica transformaciones matemáticas y estadísticas —como filtrado de ruido, normalización, corrección de iluminación y realce de contraste— para mejorar la calidad de las imágenes y facilitar la extracción de características significativas; el procesamiento y análisis emplea técnicas de visión por computador y aprendizaje profundo para extraer información relevante, como la detección o clasificación de objetos y patrones en las escenas; y finalmente, la interpretación traduce estas salidas cuantitativas en información semántica útil para la toma de decisiones, visualización o acciones

automatizadas en aplicaciones reales como el control de parqueo inteligente o la navegación robótica (Liu, 2024).

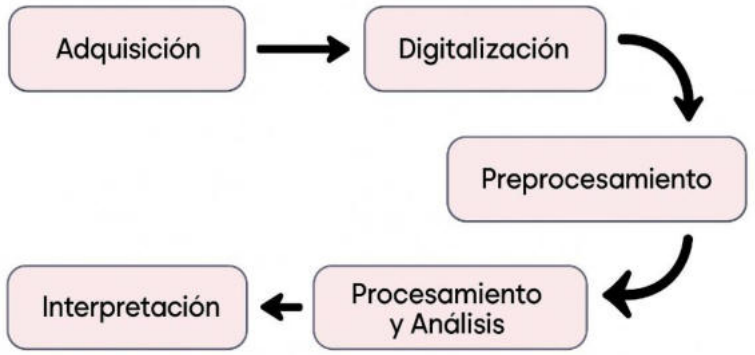


Figura 3. Las etapas del proceso de visión artificial.

Fuente: Autor

1.4.2 Adquisición

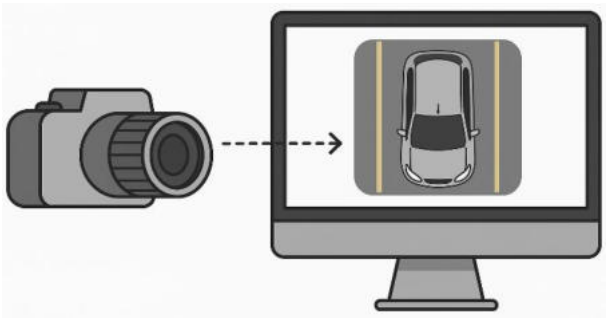


Figura 4. Adquisición de información digital.

Fuente: Autor

La adquisición es la fase inicial de cualquier sistema de visión artificial. Consiste en capturar una imagen del entorno mediante dispositivos

ópticos, como cámaras digitales, industriales o sensores especializados (**¡Error! No se encuentra el origen de la referencia.**). Esta etapa es crítica, ya que la calidad de la imagen obtenida afecta directamente el rendimiento de las fases posteriores. Se debe considerar el tipo de lente, la iluminación y la posición del sensor, ya que todo influye en la nitidez y contraste. Una adquisición deficiente puede limitar el análisis visual posterior y generar errores de interpretación (Szeliski, 2021). Algunas técnicas de adquisición se detallan a continuación:

Tabla 1. Técnicas de adquisición de imágenes.

Tipo de Adquisición	Dispositivo	Componentes
Captura en color Imágenes en tres canales: rojo, verde y azul (RGB).	Cámara digital	– Sensor CCD (Charge-Coupled Device) o CMOS (Complementary Metal-Oxide-Semiconductor). – Lente estándar. – Iluminación natural o artificial.
Captura monocromática Imágenes en escala de grises, centradas en la intensidad de luz.	Cámara monocromática	– Sensor CCD/CMOS monocromático. – Lente fija. – Filtro de paso bajo opcional.
Captura térmica infrarroja Adquisición con variaciones de temperatura mediante radiación infrarroja (IR).	Cámara térmica	– Sensor bolométrico o IR no refrigerado. – Óptica térmica. – Aislante térmico externo opcional.
Captura tridimensional (3D)	Cámara de profundidad	– Sensor Time-of-Flight (ToF), Light Detection

Tipo de Adquisición	Dispositivo	Componentes
Obtiene profundidad y forma de objetos.		and Ranging (LiDAR) o estéreo. – Proyector de luz estructurada. – Lente de precisión.
Captura lineal (1D) Adquisición línea por línea para inspección continua.	Escáner de línea	– Sensor CCD/CMOS lineal. – Óptica adaptada a desplazamiento. – Sistema de arrastre o sincronización.
Captura aérea o dinámica Imágenes en movimiento o desde el aire, usando múltiples sensores.	Dron o sistema móvil	– Cámaras RGB, IR o multiespectrales. – Sistema Global Positioning System (GPS). – Controlador de navegación autónoma.

Fuente: Autor.

1.4.3 Digitalización

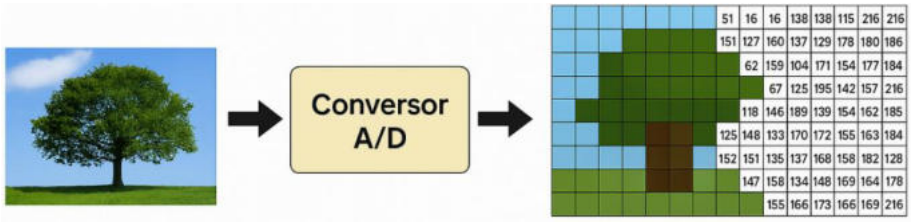


Figura 5. Proceso de digitalización.

Fuente: Autor

Como se muestra en la **¡Error! No se encuentra el origen de la referencia.**, la digitalización convierte la imagen adquirida, que puede ser analógica, en una matriz de valores numéricos que representan información visual digital. Esto se logra mediante conversores analógico-digital (A/D) que traducen la señal óptica en datos binarios. Cada píxel contiene información sobre la intensidad de luz o color. Se definen propiedades como la resolución espacial, profundidad de bits y espacio de color. Esta etapa es esencial para que la imagen pueda ser manipulada computacionalmente por algoritmos posteriores (Burger & Burge, 2022).

Tabla 2. Componentes de la digitalización.

Elemento	Descripción
¿Qué es?	Es el proceso que convierte una imagen analógica en una representación digital numérica.
Objetivo	Permitir que una imagen sea entendida y procesada por un sistema computacional.
Dispositivo clave	Convertidor Analógico-Digital (A/D): transforma señales ópticas continuas en valores binarios.
Unidad de imagen	Píxel: la unidad mínima de una imagen digital; representa intensidad de luz o color.
Formato resultante	Matriz de píxeles: cada celda representa un valor numérico que compone la imagen.
Resolución	Número total de píxeles de una imagen (por ejemplo: 1280x720); define el nivel de detalle.
Profundidad de bits	Número de niveles que puede representar un píxel (por ejemplo: 8 bits = 256 tonos).
Espacio de color	Modelo para representar el color (por ejemplo: RGB, Hue-Saturation-Value (HSV), escala de grises).

Fuente: Autor.

1.4.4 Preprocesamiento

El preprocesamiento mejora la calidad de la imagen digital antes de que sea analizada (Tabla 3***Tabla 1***). Se aplican filtros para reducir el ruido, mejorar bordes, normalizar la iluminación y resaltar detalles importantes. También puede implicar convertir la imagen a escala de grises o aplicar umbrales binarios. El objetivo es optimizar la imagen para que el sistema de análisis pueda identificar con mayor precisión los objetos o regiones de interés. Esta etapa actúa como una "limpieza" inicial de datos visuales (Goodfellow et al., 2016).

Tabla 3. Operaciones fundamentales en el preprocesamiento.

Preprocesamiento	Descripción
Reducción de ruido	Atenúa variaciones aleatorias en la intensidad de píxeles utilizando filtros espaciales.
Mejoramiento de bordes	Realza discontinuidades de intensidad para resaltar límites entre regiones.
Normalización de iluminación	Ajusta la distribución de brillo para corregir variaciones en las condiciones de captura.
Realce de detalles	Incrementa el contraste local para destacar estructuras finas en la imagen.
Conversión a escala de grises	Elimina la información de color conservando la luminancia para reducir la dimensionalidad.
Umbralización binaria	Asigna valores binarios a los píxeles según un umbral definido para segmentación básica.
Optimización general	Conjunto de transformaciones que mejoran la calidad de la imagen para su análisis posterior.

Fuente: Autor

Las técnicas de preprocesamiento no solo facilitan el trabajo de los algoritmos posteriores, como la segmentación o de reconocimiento, sino que también permiten aumentar la precisión general del sistema. Se detallan algunas técnicas utilizadas:

1.4.5 Operaciones elementales

Las operaciones punto a punto son transformaciones básicas que se aplican directamente sobre los valores de los píxeles de una imagen, sin tener en cuenta su vecindario. Estas técnicas resultan especialmente útiles en las primeras etapas del preprocesamiento, ya que permiten modificar características fundamentales como el brillo, el contraste o la intensidad general de la imagen de manera rápida y efectiva.

Ajuste de brillo: Es una operación punto a punto que modifica la luminosidad global de una imagen al sumar o restar un valor constante a cada uno de sus píxeles (**¡Error! No se encuentra el origen de la referencia.**). Este procedimiento no depende del vecindario del píxel, por lo que su implementación es rápida y directa. Su objetivo principal es aclarar u oscurecer la imagen, mejorando así la visibilidad de los elementos visuales en condiciones de iluminación no óptimas (Zhang et al., 2024).

Fórmula aplicada:

$$I^{'}(x, y) = I(x, y) + \beta$$

Donde:

- x: coordenada horizontal del píxel en la imagen.
- y: coordenada vertical del píxel en la imagen.
- $I(x, y)$: valor de intensidad del píxel original en la posición (x, y)
- β: valor constante que se suma a cada píxel
    - $\beta > 0$: aclara la imagen
    - $\beta < 0$: oscurece la imagen
- $I'(x, y)$: valor de intensidad del píxel modificado

Nota: Los valores resultantes deben limitarse al rango de la imagen, por ejemplo, de 0 a 255 en imágenes de 8 bits.



Figura 6. Imagen con ajuste de brillo.

Fuente: Autor.

En la **¡Error! No se encuentra el origen de la referencia.** se compara el efecto del ajuste de brillo en una fotografía tomada desde una vista cenital de un vehículo estacionado. A la izquierda se muestra la imagen original, mientras que a la derecha se presenta la misma imagen tras

aplicar un aumento de brillo con un valor de β=+0.5, esta operación punto a punto consiste en incrementar la intensidad de cada píxel mediante la suma de un valor constante, sin afectar la relación entre colores. Su principal objetivo es mejorar la visibilidad en condiciones de baja iluminación, facilitando la detección de objetos y contornos por parte de sistemas de visión artificial.

Ajuste de Contraste: El ajuste de contraste es una operación punto a punto que modifica la diferencia entre los valores de intensidad más claros y oscuros de una imagen (**¡Error! No se encuentra el origen de la referencia.**). Su propósito es acentuar los detalles visuales al expandir o comprimir el rango dinámico de los píxeles.

Una imagen con bajo contraste puede lucir opaca o "lavada", dificultando la identificación de objetos. Al aumentar el contraste, se resalta la estructura interna de la imagen, lo cual es especialmente útil en sistemas de visión artificial para la segmentación, detección y reconocimiento de objetos(Rivera-Aguilar et al., 2024).

Fórmula aplicada:

$$I^{'}(x,y) = \propto * \big(I(x,y) - 128\big) + 128$$

Donde:

- x: coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $I(x,y)$: valor de intensidad del píxel original en la posición (x,y)
- $\propto$: factor de contraste

- $\propto > 1$: aumenta el contraste
- $0 < \propto < 1$: reduce el contraste

- $I'(x, y)$: valor de intensidad del píxel modificado

Nota: Esta fórmula ajusta el contraste respecto al punto medio (128) del rango de intensidad (0–255), expandiendo o comprimiendo los tonos alrededor de ese punto.

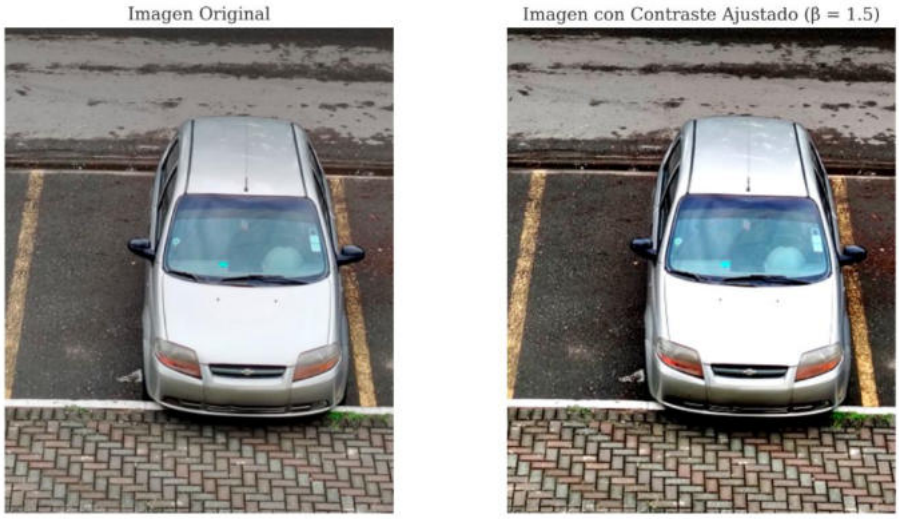


Figura 7. Imagen con ajuste de contraste.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra una imagen original (izquierda) y su versión modificada mediante la técnica de ajuste de contraste (derecha). En la imagen ajustada se aplicó un factor de contraste $\propto$=1.5, la cual expande el rango dinámico alrededor del punto medio de intensidad. Como resultado, se acentúan las diferencias entre las regiones claras y oscuras, mejorando la percepción

de bordes y detalles estructurales. Este tipo de procesamiento es fundamental en visión artificial para optimizar la segmentación y el reconocimiento de objetos en escenas con bajo contraste.

Negativo de imagen: La operación de negativo es una técnica punto a punto utilizada en procesamiento digital de imágenes, que consiste en invertir la intensidad de cada píxel respecto al valor máximo posible (**¡Error! No se encuentra el origen de la referencia.**). En imágenes en escala de grises o en color, esto genera una representación visual inversa, donde los valores claros se vuelven oscuros y viceversa (Ramadhan et al., 2025).

Esta técnica resulta especialmente útil en aplicaciones médicas y científicas, donde los detalles que no son perceptibles en la imagen original pueden volverse más evidentes en su negativo. También se emplea para realce de bordes, análisis de texturas y procesamiento de imágenes de bajo contraste.

Fórmula aplicada:

$$I^{'}(x,y) = 255 - I(x,y)$$

Donde:

- x: coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $I(x,y)$: valor de intensidad del píxel original en la posición (x,y)
- $I'(x,y)$: valor de intensidad del píxel modificado

Nota: Esta fórmula invierte el valor de cada píxel dentro del rango [0,255]. Por ejemplo, un píxel con intensidad 100 será transformado a 155, ya que 255 -100=155.



Figura 8. Imagen con negativo.

Fuente: Autor.

En la **¡Error! No se encuentra el origen de la referencia.** se observa a la izquierda, la imagen original capturada desde una vista superior, y a la derecha, su versión modificada mediante la operación de negativo. Esta técnica consiste en invertir los valores de intensidad de cada píxel, transformando los tonos claros en oscuros y viceversa. El resultado es

una imagen en la que se invierte el contraste global, lo cual puede ser útil en sistemas de visión artificial para resaltar bordes, detalles sutiles o estructuras internas que no son fácilmente perceptibles en la imagen original.

Transformación logarítmica: La transformación logarítmica es una técnica de procesamiento punto a punto que mejora el contraste de las regiones oscuras de una imagen (**¡Error! No se encuentra el origen de la referencia.**). Se basa en aplicar una función logarítmica a los valores de intensidad de los píxeles, comprimiendo los valores altos y expandiendo los bajos, lo cual permite resaltar detalles ocultos en sombras.

Esta transformación es especialmente útil cuando se trabaja con imágenes con iluminación desigual o con datos científicos donde la mayoría de los valores útiles se encuentran en niveles bajos de intensidad. Es común en imágenes astronómicas, médicas y microscopía digital (Bae et al., 2025).

Fórmula aplicada:

$$I^{'}(x,y) = \ c * \ \log\Big(1 + I(x,y)\Big)$$

Donde:

- x : coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $I(x,y)$: valor de intensidad del píxel original en la posición (x,y)
- $I'(x,y)$: valor de intensidad del píxel modificado

- c: constante de escala para normalizar al rango [0,255]

Nota: Se suma 1 para evitar logaritmo de cero y se usa logaritmo natural o base 10, según la implementación.

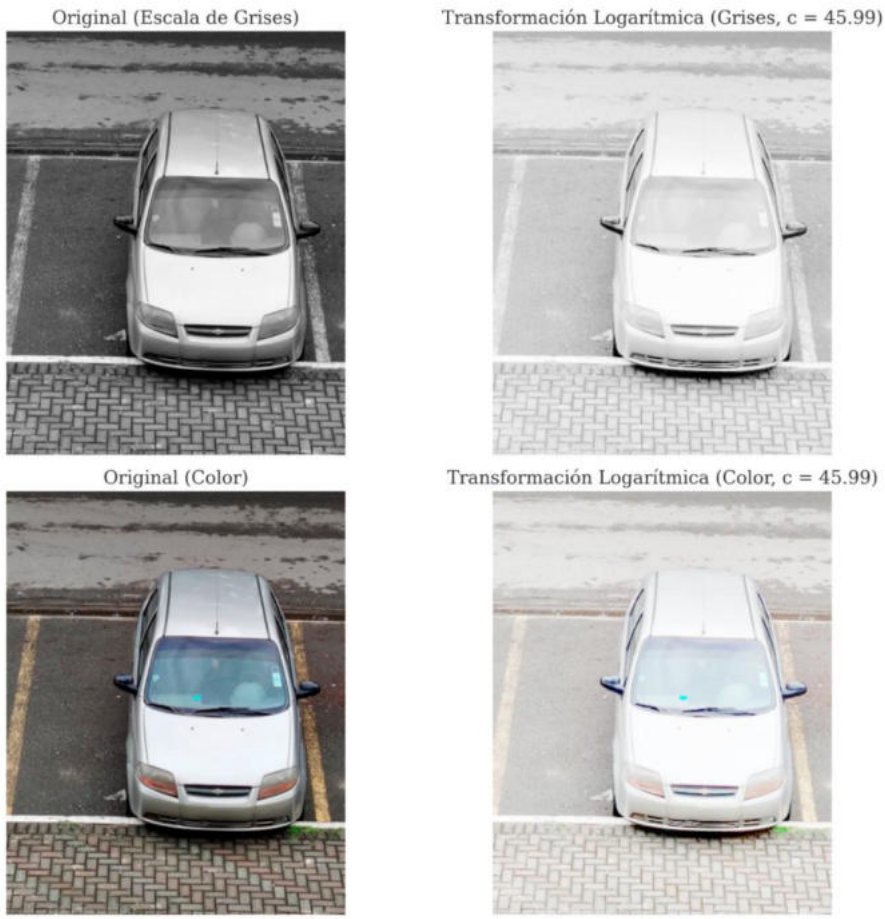


Figura 9. Imagen con transformación logarítmica.

Fuente: Autor.

En la **¡Error! No se encuentra el origen de la referencia.**, en la parte superior, la imagen original en escala de grises (izquierda) y su transformación logarítmica (derecha). En la parte inferior, se observa la misma comparación, pero aplicada a la imagen en color. En ambos casos, se utilizó la fórmula con un valor de $c \approx 45.99$, que permite realzar los niveles bajos de intensidad sin saturar los altos. Esta técnica es útil en visión artificial para resaltar detalles en zonas oscuras, mejorando así la visibilidad de estructuras que podrían pasar desapercibidas en la imagen original.

Corrección Gamma: La corrección gamma es una técnica de preprocesamiento punto a punto que ajusta la luminancia de una imagen, permitiendo mejorar la visualización de detalles en zonas oscuras o muy claras (**¡Error! No se encuentra el origen de la referencia.**). Esta transformación simula la percepción del ojo humano, que no responde linealmente a la luz, y permite controlar el brillo de una imagen sin perder información en los extremos de intensidad (Soni et al., 2024).

En sistemas de visión artificial, la corrección gamma es esencial para adaptar las imágenes a las características del sensor o del dispositivo de visualización. También se utiliza para compensar condiciones de iluminación no ideales y lograr una distribución más uniforme de los tonos.

Fórmula aplicada:

$$I^{'}(x,y) = 255 * \left(\frac{I(x,y)}{255}\right)^{\gamma}$$

Donde:

- x: coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $I(x,y)$: valor de intensidad del píxel original en la posición (x,y)
- γ: parámetro de corrección gamma
    - $\gamma < 1$: aclara la imagen
    - $\gamma > 1$: oscurece la imagen
- $I'(x,y)$: valor de intensidad del píxel modificado

Nota: La imagen debe normalizarse al rango [0, 1] antes de aplicar la potencia y luego volver a escalarse a [0, 255].



Figura 10. Imagen con corrección gamma.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** presenta una comparación entre una imagen original en color (izquierda) y su versión modificada mediante la corrección gamma con un valor de γ=0.5 (derecha). Cuando γ<1, la transformación aclara la imagen, realzando las regiones oscuras sin saturar las zonas más brillantes. Esta operación es especialmente útil en visión artificial para mejorar la percepción de detalles ocultos en sombras o áreas subexpuestas.

1.4.6 Operaciones Geométricas

Las operaciones geométricas en visión artificial son transformaciones aplicadas a las imágenes con el objetivo de modificar su forma, orientación o posición en el espacio, sin alterar su contenido visual. Su aplicación es fundamental en tareas como la alineación de imágenes, corrección de perspectiva, normalización de objetos, seguimiento de movimiento y reconstrucción de escenas. Mediante estas técnicas, se puede garantizar la coherencia espacial de los datos visuales, mejorando así la precisión en la segmentación, el reconocimiento de patrones y la interpretación de estructuras visuales complejas.

Traslación: La traslación es una operación geométrica que desplaza todos los píxeles de una imagen en una dirección específica sobre el plano cartesiano. Se utiliza comúnmente para reubicar objetos dentro de la imagen, alinear imágenes, o preparar un conjunto para operaciones más complejas como registro o detección de movimiento(Awaluddin et al., 2023).

En visión artificial, la traslación es útil para ajustar la posición de los objetos detectados en relación con un punto de referencia, o para simular desplazamientos en datos de entrenamiento (aumentación de datos).

Fórmula aplicada:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Donde:

- x: coordenada horizontal original del punto.
- y: coordenada vertical original del punto.
- x': coordenada horizontal después de la traslación.
- y': coordenada vertical después de la traslación.
- t_x: desplazamiento aplicado en el eje horizontal.
- t_y: desplazamiento aplicado en el eje vertical.



Figura 11. Imagen con traslación.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se muestra el efecto de una transformación geométrica de traslación aplicada a una imagen en color. A la izquierda se encuentra la imagen original y a la derecha su versión trasladada, desplazada 50 píxeles hacia la derecha y 30 píxeles hacia abajo. Esta operación se realiza modificando las coordenadas de cada píxel, sin alterar el contenido interno de la imagen. En visión artificial, la traslación es útil para simular movimientos, realizar aumentos de datos (data augmentation) o ajustar el alineamiento entre imágenes.

Rotación: La rotación es una transformación geométrica que gira todos los píxeles de una imagen alrededor de un punto fijo, generalmente el centro de la imagen. Esta técnica permite cambiar la orientación de los objetos dentro de una imagen y es útil para normalizar la posición, hacer análisis multivista o aplicar aumentación de datos en entrenamiento de modelos (Awaluddin et al., 2023).

La rotación también es esencial para alinear imágenes que fueron capturadas desde distintos ángulos o para simular movimientos en entornos sintéticos.

Fórmula aplicada:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

Donde:

- x: coordenada horizontal original.
- y: coordenada vertical original.
- x': coordenada horizontal después de la rotación.
- y': coordenada vertical después de la rotación.
- θ: ángulo de rotación en sentido antihorario (grados o radianes)



Figura 12. Imagen con rotación.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se ilustra el efecto de una transformación geométrica de rotación aplicada a una imagen en color. La imagen de la derecha ha sido rotada 45 grados respecto a su centro, mientras que la imagen de la izquierda permanece sin modificaciones. En visión artificial, la rotación se emplea en el

preprocesamiento de imágenes, aumento de datos y reconocimiento de objetos desde distintas orientaciones.

Escalado: El escalado es una transformación geométrica que permite cambiar el tamaño de una imagen sin alterar su contenido. Puede realizarse de forma proporcional (manteniendo la relación de aspecto) o no proporcional (modificando alto y ancho de manera independiente) (Awaluddin et al., 2023).

El escalado se utiliza comúnmente en:

- Normalizar resoluciones de imágenes.
- Optimización de imágenes para visualización.
- Generación de imágenes miniatura (thumbnails).
- Aumentación de datos en Aprendizaje Profundo (Deep learning).

Fórmula aplicada:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

Donde:

- x: coordenada horizontal original.
- y: coordenada vertical original.
- x': coordenada horizontal después del escalado.
- y': coordenada vertical después del escalado.
- s_x: factor de escalado aplicado en el eje horizontal.
- s_y: factor de escalado aplicado en el eje vertical.



Figura 13. Imagen con escalado.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se presenta, a la izquierda la imagen original y a la derecha, su versión escalada con un factor de 0.5. Para evidenciar visualmente la reducción de tamaño, la imagen escalada fue centrada sobre un fondo negro del mismo tamaño que la original. El escalado permite modificar las dimensiones de una imagen, siendo una técnica esencial en visión artificial para normalizar entradas a modelos, ajustar resoluciones o generar variaciones en los datos de entrenamiento.

Reflexión (Espejado): La reflexión es una transformación geométrica que invierte una imagen respecto a un eje. También se conoce como espejado, ya que genera una imagen simétrica similar a la vista frente a un espejo (Awaluddin et al., 2023). Puede realizarse en dos formas:

- Reflexión horizontal: invierte la imagen de izquierda a derecha.
- Reflexión vertical: invierte la imagen de arriba hacia abajo.

Esta técnica es útil en tareas como aumentación de datos, análisis simétrico, y verificación visual de patrones. En aprendizaje profundo, se emplea para generar variaciones adicionales de las imágenes originales y así mejorar la robustez del modelo.

Fórmula aplicada:

- Para una reflexión horizontal: $x' = -x$.
- Para una reflexión vertical: $y' = -y$.

Donde:

- x: coordenada horizontal original.
- y: coordenada vertical original.
- x': coordenada horizontal resultante después de la reflexión.

y': coordenada vertical resultante después de la reflexión.



Figura 14. Imagen con reflexión horizontal.

Fuente: Autor

En la

se muestra una imagen original (izquierda) y su versión reflejada horizontalmente (derecha). Esta transformación invierte la imagen respecto al eje vertical, generando un efecto de espejo. En visión artificial, la reflexión es utilizada para aumentar la variabilidad de los datos de entrenamiento, mejorar la robustez de los modelos ante simetrías y simular diferentes perspectivas de un mismo objeto.

Sesgado (Shear): El sesgado es una transformación geométrica que desplaza las coordenadas de los píxeles en una dirección, proporcional a su posición en el eje opuesto. Este efecto genera una imagen con apariencia de estar inclinada o deformada en forma de paralelogramo (Hwang et al., 2025).

Existen dos tipos de sesgado:

- Sesgado horizontal: los puntos se desplazan lateralmente en función de su posición vertical.
- Sesgado vertical: los puntos se desplazan hacia arriba o abajo en función de su posición horizontal.

El sesgado es útil para simular perspectiva o deformaciones angulares en objetos, y también se utiliza como técnica de aumentación en aprendizaje automático.

Fórmula aplicada:

- Para sesgado horizontal:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

- Para sesgado vertical:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$$

Donde:

- x: coordenada horizontal original.
- y: coordenada vertical original.
- x': coordenada horizontal después del sesgado.
- y': coordenada vertical después del sesgado.
- k: coeficiente de sesgado que define la inclinación

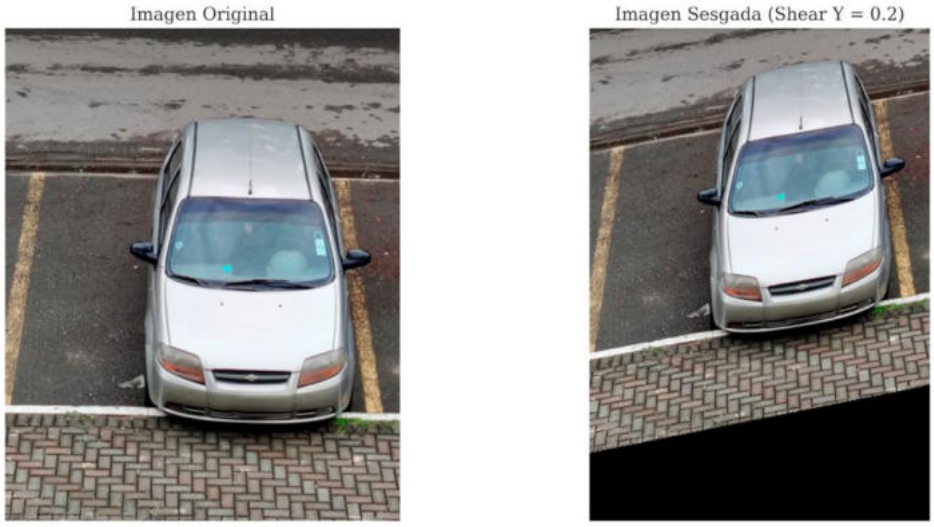


Figura 15. Imagen con sesgado.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se muestra una imagen original (izquierda) y su versión transformada mediante un sesgado en el eje vertical (derecha), utilizando un factor de shear k=0.2. Esta transformación afín modifica las coordenadas verticales de los píxeles proporcionalmente a su posición horizontal, generando una inclinación en la imagen

Transformaciones afines: Las transformaciones afines son operaciones geométricas que permiten modificar una imagen conservando la colinealidad (los puntos que estaban en línea recta permanecen en línea) y las relaciones proporcionales entre distancias (Torpey & Klein, 2024).

Una transformación afín permite combinar varias operaciones geométricas, como traslación, rotación, escalado, reflexión y sesgado.(shear).

Estas transformaciones se aplican comúnmente en tareas de registro de imágenes, alineamiento, perspectiva controlada, y aumentación de datos para aprendizaje profundo.

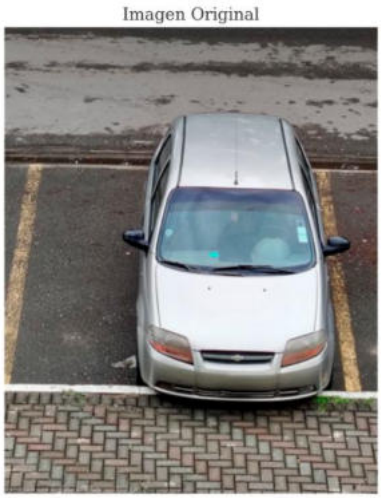
Fórmula aplicada:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Donde:

- $\boldsymbol{x}$: coordenada horizontal original.

- y: coordenada vertical original.
- x': coordenada horizontal resultante de la transformación afín.
- y': coordenada vertical resultante de la transformación afín.
- t_x: componente de traslación horizontal.
- t_y: componente de traslación vertical.
- a, b, c, d: combinación de rotación, escalado y sesgado



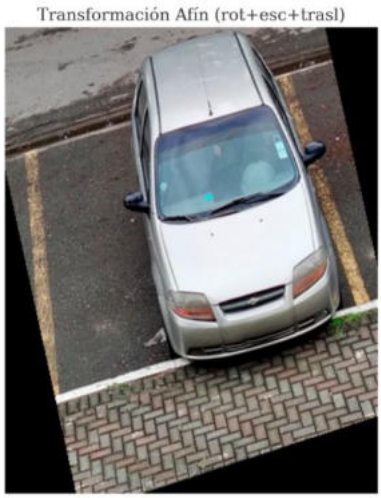


Figura 16. Transformaciones afines.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se presenta la imagen con una transformación afín combinada (imagen derecha), que incluye rotación (15°), escalado (x=1.1, y=0.9) y traslación (x=+30, y=+20). Los parámetros controlan la rotación del contenido, el cambio de tamaño (escalado) y el desplazamiento espacial (traslación). En visión artificial, estas transformaciones son fundamentales para simular cambios de perspectiva, normalizar imágenes y generar variaciones realistas durante el preprocesamiento o la expansión de datos.

1.4.7 Filtros Lineales

Los filtros lineales son operadores fundamentales en el procesamiento digital de imágenes, utilizados para modificar la intensidad de los píxeles mediante combinaciones lineales con sus vecinos. Su funcionamiento se basa en la convolución de la imagen con una máscara o kernel, que define el comportamiento del filtro. Son ampliamente utilizados por su simplicidad matemática, eficiencia computacional y aplicabilidad en etapas iniciales del análisis de imágenes, permitiendo preparar los datos visuales para procesos posteriores como la segmentación o el reconocimiento.

Filtrado Promedio: El filtro promedio es un tipo de filtro lineal que se utiliza comúnmente en procesamiento de imágenes para reducir el ruido y lograr un efecto de suavizado. Este filtro actúa sobre una región (vecindario) de la imagen y reemplaza el valor del píxel central con el

promedio de los valores de los píxeles vecinos, incluidos él mismo (Nguyen Cong et al., 2025).

Es especialmente útil para suavizar transiciones suaves y reducir ruido de alta frecuencia (como el ruido "sal y pimienta").

Filtro promedio 3x3:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Fórmula aplicada:

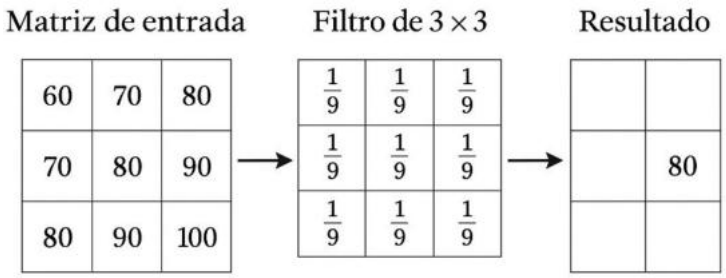
$$I^{'}(x,y) = \frac{1}{mn}\sum_{i=-a}^{a}\sum_{j=-b}^{b} I(x+i, y+j)$$

Donde:

- x: coordenada horizontal del píxel central.
- y: coordenada vertical del píxel central.
- i: índice utilizado para recorrer una dimensión del vecindario.
- j: índice utilizado para recorrer la otra dimensión del vecindario.
- $I(x,y)$: valor de intensidad del píxel original en la posición.
- $I'(x,y)$: valor del píxel tras aplicar el filtro.
- $m\ x\ n$: es el tamaño del Kernel (3x3).

- $a = \frac{m-1}{2}, b = \frac{n-1}{2}$: son los márgenes del vecindario asociados a m y n.

Se suman los valores del vecindario y se dividen por el total de elementos



$$\frac{(60+70+80+70+80+90+80+90+100)}{9} = 80$$



Figura 17. Filtro promedio.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra la imagen original y su versión suavizada mediante la aplicación de un filtro promedio 3×3 (derecha). Este tipo de filtro espacial lineal actúa calculando la media de los valores de intensidad de un píxel y sus vecinos en una ventana de 3×3, sustituyendo el valor central por ese promedio. El objetivo principal es reducir la variación local (ruido) y suavizar la imagen, mejorando la visibilidad de estructuras sin aplicar transformaciones complejas. Aunque este filtro es eficaz para eliminar ruido aleatorio, puede provocar pérdida de nitidez en los bordes de la imagen.

Filtro Gaussiano: El filtro gaussiano es un tipo de filtro lineal que suaviza una imagen aplicando una función gaussiana (en forma de campana) sobre el vecindario de cada píxel. Este filtro da más peso a los

píxeles cercanos al centro del vecindario, lo que lo hace más efectivo que el filtro promedio para reducir ruido sin perder tantos detalles en los bordes (Lindeberg, 2024).

Fórmula aplicada:

$$G(x,y) = \frac{1}{2\,\pi\,\sigma^2} * \; e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Donde:

- G(x,y): valor de la función gaussiana en la posición (x,y).
- x, y: coordenadas relativas al centro del kernel.
- (x, y): valor de la función gaussiana en la posición.
- σ: desviación estándar (controla la intensidad del suavizado)



Figura 18. Filtro Gaussiano.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra a la izquierda una imagen original y a la derecha la misma imagen tras aplicar un filtro gaussiano. Este filtro suaviza la imagen utilizando una máscara basada en la distribución normal (gaussiana), otorgando mayor peso a los píxeles cercanos al centro del kernel y menor peso a los más lejanos. A diferencia del filtro promedio, el filtro gaussiano reduce el ruido preservando mejor los bordes y las estructuras importantes. Es ampliamente utilizado en visión artificial como paso previo a técnicas de detección de bordes, segmentación o reducción de ruido.

1.4.8 Filtro de derivadas (Gradientes)

Los filtros de derivadas se utilizan para detectar bordes en imágenes. Un borde es una zona donde hay un cambio abrupto en la intensidad de los píxeles, y estos filtros calculan precisamente esa variación, conocida como gradiente. Los gradientes se pueden calcular en dirección horizontal (eje X) o dirección vertical (eje Y).

Fórmula aplicada:

- Gradiente horizontal: $G_x = \frac{\partial I}{\partial x}$
- Gradiente vertical: $G_y = \frac{\partial I}{\partial y}$

Luego se calcula la magnitud del gradiente total:

$$G = \sqrt{G_x^2 + G_y^2}$$

Donde:

- *I*: intensidad de la imagen.
- x: dirección horizontal de la imagen.
- y: dirección vertical de la imagen.
- Gx: gradiente de intensidad calculado en la dirección horizontal.
- Gy: gradiente de intensidad calculado en la dirección vertical.
- G: magnitud total del gradiente.

Previo a la aplicación práctica de los filtros de derivadas, resulta esencial comprender las diferentes máscaras o kernels que se utilizan para estimar los gradientes de intensidad en una imagen digital. Estos kernels permiten calcular aproximaciones de la primera o segunda derivada en direcciones específicas, siendo herramientas clave en la detección de bordes y contornos. A continuación, se detallan los kernels más representativos empleados en el análisis de gradientes.

Filtro Sobel: es uno de los operadores de derivadas más utilizados en procesamiento de imágenes para la detección de bordes. Se basa en el cálculo de gradientes de intensidad en direcciones específicas (horizontal y vertical), permitiendo resaltar zonas donde ocurren transiciones abruptas de tonalidad o contraste (Onyedinma Ebele G. et al., 2025).

Kernel Sobel Gx (Horizontal)

-1	0	1
-2	0	2
-1	0	1

Kernel Sobel Gy (Vertical)

-1	-2	-1
0	0	0
1	2	1

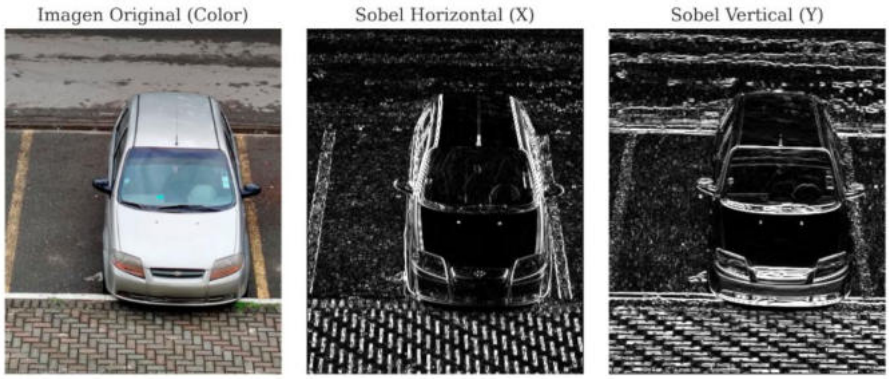


Figura 19. Filtro Sobel.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** presenta, de izquierda a derecha, la imagen original en color, el resultado del operador Sobel aplicado en la dirección horizontal (Sobel X) y el resultado del operador Sobel en la dirección vertical (Sobel Y). El operador Sobel es un filtro derivativo de primer orden que calcula aproximaciones discretas de las derivadas parciales de la imagen. En Sobel X, se utiliza una convolución con un kernel que enfatiza los cambios de intensidad en el eje X, permitiendo la detección de bordes verticales. En Sobel Y, el kernel resalta los cambios en el eje Y, lo que permite identificar bordes horizontales. Ambos operadores son sensibles a las transiciones abruptas de intensidad y son fundamentales para el análisis morfológico y estructural de la imagen, sirviendo como etapa inicial en algoritmos de segmentación, extracción de contornos y reconocimiento de formas en sistemas de visión artificial.

Filtro Prewitt: es un operador de derivadas utilizado para detectar bordes en imágenes digitales, mediante la estimación de los gradientes

en las direcciones horizontal y vertical. Emplea pesos constantes en sus máscaras, lo que lo hace más sencillo de implementar, aunque algo más sensible al ruido. Este filtro es útil en aplicaciones donde se requiere una detección de bordes rápida y de bajo costo computacional (Said & Rawahi, n.d.).

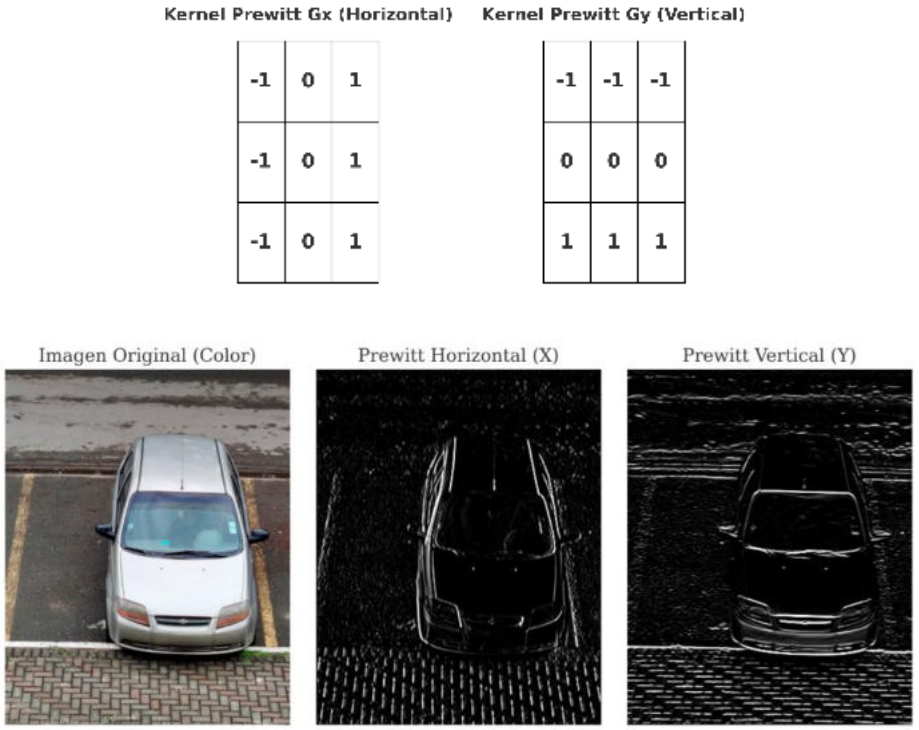


Figura 20. Filtro Prewitt.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se presenta una comparación entre la imagen original a color y los resultados obtenidos mediante el operador de Prewitt en sus componentes horizontal (X) y vertical (Y). Este operador utiliza máscaras

convolucionales discretas de 3×3 para aproximar las derivadas parciales de la intensidad de la imagen. En la dirección horizontal, se aplica un kernel que detecta cambios de intensidad a lo largo del eje X, permitiendo la identificación de bordes verticales. En la dirección vertical se utiliza un kernel que resalta cambios en el eje Y, detectando bordes horizontales. El operador de Prewitt es menos sensible al ruido que otros operadores simples como Roberts, y proporciona una estimación robusta del gradiente para análisis morfológicos y extracción de contornos en sistemas de visión artificial.

Filtro Roberts: es uno de los operadores de derivadas más simples utilizados para la detección de bordes. Calcula una aproximación a la derivada de la imagen utilizando un par de máscaras 2×2, que permiten detectar bordes diagonales. Este operador se basa en evaluar las diferencias entre píxeles adyacentes, lo cual lo hace muy sensible a los cambios rápidos de intensidad, pero también lo hace vulnerable al ruido. Por esta razón, se utiliza principalmente en imágenes con buen contraste y bajo nivel de ruido (Burnham et al., n.d.).

Kernel Roberts Gx

1	0
0	-1

Kernel Roberts Gy

0	1
-1	0

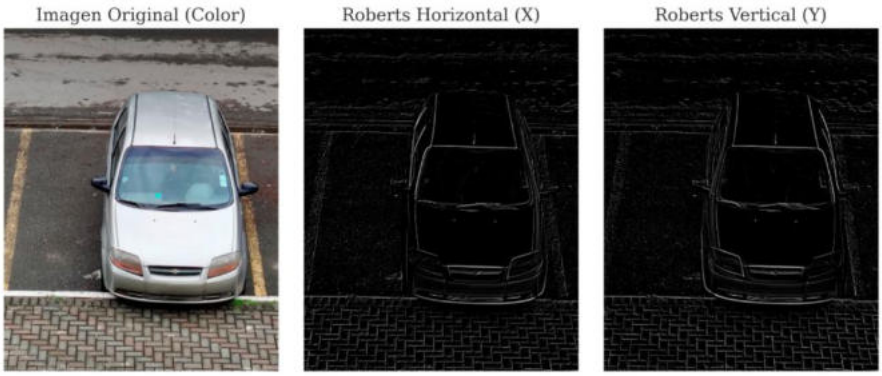


Figura 21. Filtro Roberts.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra el resultado de aplicar el operador Roberts en la detección de bordes en dos direcciones. El kernel Roberts en la dirección horizontal (X) detecta los bordes verticales, mientras que en la dirección vertical (Y), se identifican los bordes horizontales. A diferencia de otros operadores, como Sobel y Prewitt, el operador Roberts utiliza un kernel de 2×2, lo que lo hace especialmente sensible a cambios abruptos de intensidad en las imágenes. Este operador es útil en tareas de detección de bordes en imágenes de alta resolución donde se buscan transiciones precisas a pequeña escala.

Filtro Laplaciano (2ª derivada): es un operador de derivadas de segundo orden utilizado en procesamiento de imágenes para detectar bordes. A diferencia de los operadores de primer orden (como Sobel o Prewitt), el Laplaciano identifica zonas de cambio rápido mediante el cálculo de la

segunda derivada, lo que le permite resaltar contornos en todas las direcciones simultáneamente. Este filtro no distingue entre orientación vertical u horizontal, ya que es isotrópico. Generalmente se aplica sobre una imagen previamente suavizada (por ejemplo, con un filtro gaussiano) para evitar amplificar el ruido (Singh & Walingo, 2024).

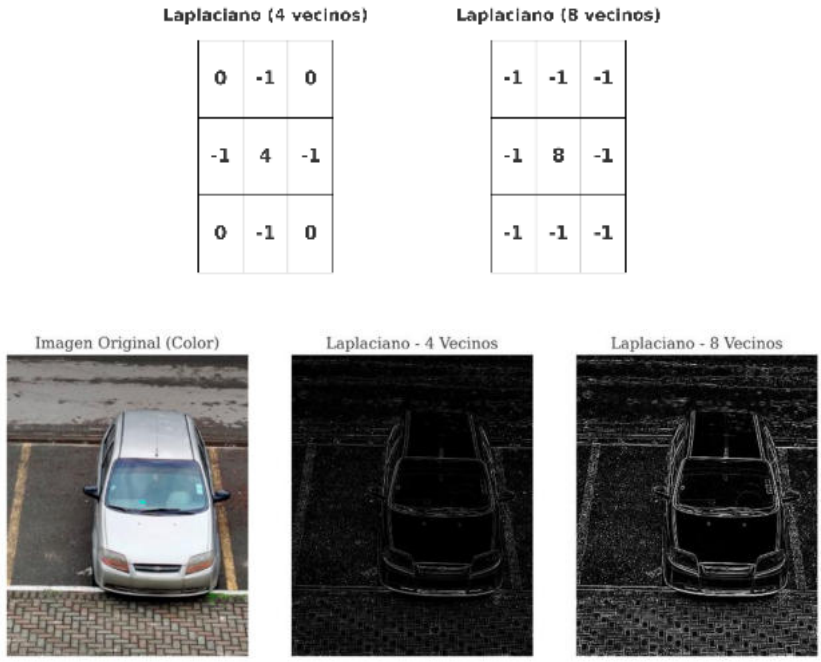


Figura 22. Laplaciano (2ª derivada).

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra dos variantes del operador Laplaciano: de4 y 8 vecinos. El operador Laplaciano con 4 vecinos utiliza un Kernel que considera las derivadas de segundo orden únicamente en las direcciones ortogonales (arriba, abajo, izquierda y derecha), mientras que el de 8 vecinos incorpora también las diagonales, generando una respuesta más sensible a bordes

en todas las direcciones. Ambos operadores realzan zonas con cambios abruptos de intensidad, pero la versión de 8 vecinos produce un mayor contraste en las estructuras, lo que lo hace más eficaz para detectar detalles finos en tareas de segmentación y análisis morfológico.

Filtro Canny: es un operador de detección de bordes que utiliza derivadas de primer orden para identificar transiciones abruptas de intensidad en una imagen. Su funcionamiento se basa en una secuencia de etapas: primero, aplica un suavizado con un filtro gaussiano para reducir el ruido; luego, calcula el gradiente de intensidad mediante derivadas parciales en las direcciones horizontal y vertical; a continuación, utiliza la magnitud del gradiente para detectar posibles bordes, afinando los resultados mediante una supresión de no-máximos que conserva únicamente los máximos locales; finalmente, aplica una umbralización con histéresis que clasifica los bordes en fuertes y débiles, conservando solo aquellos que estén conectados a estructuras relevantes. Gracias a esta combinación de etapas, el filtro Canny logra una detección precisa, continua y robusta de bordes, siendo ampliamente utilizado en aplicaciones de visión artificial, segmentación y análisis estructural de imágenes (Burnham et al., n.d.).

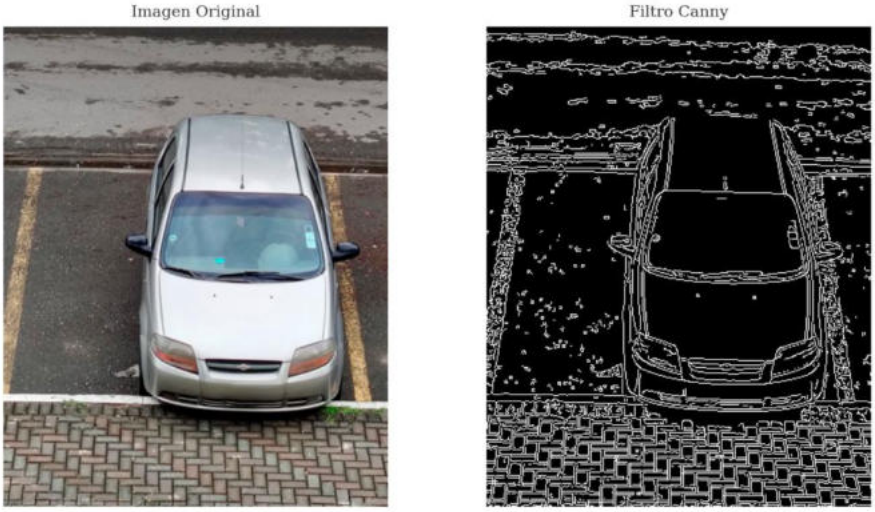


Figura 23. Filtro Canny.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra el filtro Canny aplicado sobre una imagen original en color. Este método utiliza una secuencia de etapas que incluyen suavizado con filtro Gaussiano, cálculo del gradiente de intensidad, supresión de no-máximos y umbralización con histéresis. En este caso, se utilizaron umbrales de 100 (mínimo) y 200 (máximo) para controlar la sensibilidad del detector. El filtro Canny ofrece alta precisión en la localización de bordes y minimiza la detección de falsos positivos, siendo una técnica robusta y ampliamente utilizada en sistemas de visión artificial para segmentación, reconocimiento de contornos y extracción de características.

1.4.9 Filtros no lineales

Son técnicas de procesamiento de imágenes que modifican los valores de los píxeles sin utilizar operaciones matemáticas lineales (como sumas o multiplicaciones directas). En lugar de eso, toman decisiones basadas en el orden, la lógica o la estadística de los píxeles en un vecindario.

A diferencia de los filtros lineales que pueden suavizar, pero también generar desenfoque, los filtros no lineales son especialmente útiles para preservar bordes y eliminar ruido impulsivo (como el "sal y pimienta").

Filtro de Mediana: El filtro de mediana es una operación no lineal utilizada para reducir el ruido en una imagen, especialmente el ruido impulsivo como el ruido sal y pimienta. A diferencia de los filtros lineales que promedian los valores del vecindario, este filtro reemplaza el valor del píxel central por el valor mediano de los píxeles dentro de un vecindario definido (generalmente 3×3, 5×5, etc.).

Esta técnica es muy eficaz para conservar los bordes nítidos, ya que no introduce valores nuevos por interpolación, sino que selecciona uno existente que mejor representa la región. Se aplica ampliamente en procesamiento de imágenes médicas, visión artificial y preprocesamiento de imágenes antes del análisis de características (de Mooij & Martinez, 2024).

Fórmula aplicada:

Dado un vecindario V de píxeles, el nuevo valor del píxel central $I'(x,y)$ se calcula como:

$$I^{'}(x,y) = mediana\big(V(x,y)\big)$$

Donde:

- x: coordenada horizontal del píxel procesado.
- y: coordenada vertical del píxel procesado.
- $I'(x,y)$: valor del nuevo píxel en la posición (x,y).
- V: conjunto de valores de pixeles dentro del vecindario (por ejemplo, 3x3).
- La mediana es el valor central tras ordenar los valores de V de menor a mayor.

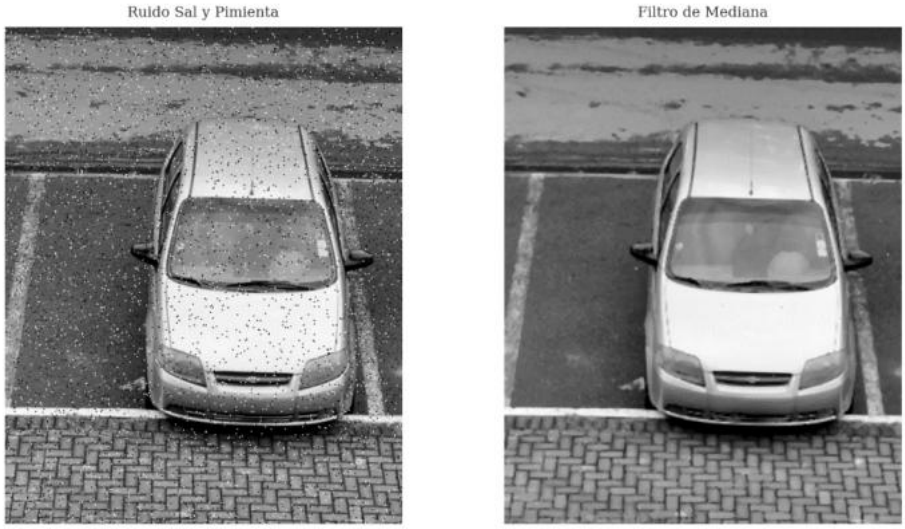


Figura 24. Filtro de Mediana.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra el efecto del filtro de mediana aplicado a una imagen afectada por ruido

impulsivo tipo sal y pimienta. Este tipo de ruido se caracteriza por la aparición de píxeles blancos (sal) y negros (pimienta) de forma aleatoria, lo que afecta la calidad de la imagen. El filtro de mediana, al ser un operador no lineal, reemplaza el valor de cada píxel por la mediana de su vecindario, eliminando eficazmente valores extremos sin distorsionar los bordes. Esta técnica resulta especialmente efectiva frente a este tipo de ruido, a diferencia de filtros lineales como el promedio que tienden a difuminar los contornos.

Filtro de Mínimo: Es una técnica no lineal que reemplaza el valor de un píxel por el menor valor dentro de su vecindario. Es particularmente útil para reducir el ruido tipo "sal", es decir, los puntos blancos que aparecen en imágenes ruidosas (Ohtani et al., 2024).

Fórmula aplicada:

$$I_{salida}(x, y) = \min\{I(i, j) | (i, j) \in vecindario\ (x, y)\}$$

Donde:

- i: coordenada o índice utilizado para recorrer el vecindario.
- j: coordenada o índice utilizado para recorrer el vecindario.
- x: coordenada horizontal del píxel central.
- y: coordenada vertical del píxel central.
- $\mathrm{I_{salida}(x, y)}$: valor resultante del píxel después de aplicar el filtro mínimo.
- $I(i, j)$: valor de píxel en la posición (i, j)
- $vecindario\ (x, y)$: conjunto de píxeles alrededor de (x, y)



Figura 25. Filtro de Mínimo.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra a la izquierda una imagen en escala de grises contaminada con ruido impulsivo tipo sal, caracterizado por la aparición aleatoria de píxeles blancos de alta intensidad. A la derecha se presenta el resultado del procesamiento con un filtro mínimo, el cual reemplaza cada píxel por el valor mínimo de su vecindario. Este operador no lineal es especialmente eficaz para eliminar puntos blancos aislados, preservando las regiones

oscuras y suavizando el efecto del ruido sin distorsionar significativamente los contornos originales de la imagen.

Filtro de Máximo: El filtro máximo es una técnica no lineal de procesamiento de imágenes que reemplaza el valor de cada píxel por el valor máximo dentro de su vecindario definido. Este filtro se utiliza comúnmente para eliminar el ruido de tipo pimienta (píxeles negros), ya que elimina los valores extremos bajos del vecindario. También tiene el efecto de resaltar bordes brillantes y aumentar el brillo en ciertas zonas (Archana & Jeevaraj, 2024).

Fórmula aplicada:

$$I_{salida}(i,j) = \max\{I(x,y)|(x,y) \in W\}$$

Donde:

- $I_{salida}(i,j)$: valor resultante del píxel después de aplicar el filtro máximo.
- i,j: posición del píxel de salida.
- x,y: coordenadas de los píxeles considerados dentro del vecindario.
- W: vecindario considerado para calcular el valor máximo.
- $I(x,y)$: intensidad de los píxeles vecinos.
- max: operador máximo aplicado sobre todos los valores del vecindario.



Figura 26. Filtro de Máximo.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** muestra una imagen en escala de grises contaminada con ruido impulsivo tipo pimienta, caracterizado por la presencia aleatoria de píxeles negros de baja intensidad. A la derecha se observa el resultado tras aplicar un filtro máximo, el cual reemplaza cada píxel por el valor máximo dentro de su vecindario. Esta operación no lineal, equivalente a la dilatación

morfológica, es particularmente eficaz para suprimir puntos oscuros aislados sin afectar las regiones brillantes de la imagen. Su uso es común en tareas de preprocesamiento donde se requiere restaurar información degradada por valores mínimos.

Filtro de Rango: El filtro de rango reemplaza cada píxel por la diferencia entre el valor más alto y el más bajo en su vecindario. Es útil para destacar transiciones fuertes en la imagen, como contornos o bordes, ya que estos generan una gran diferencia local (Umbaugh, 2022).

Fórmula aplicada:

$$I'_{(i,j)} = \max(V) - \min(V)$$

Donde:

- i: coordenada o índice correspondiente a la posición del píxel procesado.
- j: coordenada o índice correspondiente a la posición del píxel procesado.
- V: conjunto de valores de intensidad pertenecientes al vecindario considerado.
- $I'_{(i,j)}$: nuevo valor del píxel después de aplicar filtro
- $max(V)$: valor máximo del vecindario
- $min(V)$: valor mínimo del vecindario



Figura 27. Filtro de Rango.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** presenta una comparación entre la imagen original y el resultado del filtro de rango, un operador no lineal que calcula la diferencia entre el valor máximo y el mínimo dentro de un vecindario (en este caso, de 3×3 píxeles). Este filtro es sensible a la variación local de intensidad y permite resaltar regiones con alto contraste, como bordes o texturas, mientras produce valores bajos en áreas homogéneas. Es útil para análisis de textura, segmentación basada en contraste y como paso previo en algoritmos de detección de patrones visuales.

Filtro Adaptativo: El filtro adaptativo es una técnica de procesamiento de imágenes cuyo comportamiento se ajusta de acuerdo con las características estadísticas locales de la imagen. En este ejemplo, se aplica sobre una imagen en escala de grises a la que se ha incorporado ruido impulsivo de tipo sal y pimienta, con el propósito de observar el

efecto del filtrado sobre las alteraciones introducidas. El procedimiento utiliza información del vecindario de cada píxel, particularmente la media y la varianza local, junto con una estimación de la varianza del ruido, para calcular el nuevo valor del píxel. De esta manera, el nivel de corrección se adapta a la variabilidad presente en cada región de la imagen (Guan et al., 2024).

Fórmula aplicada:

$$I^{'}(x,y) = u(x,y) + \frac{\sigma^2(x,y) - n^2}{\sigma^2(x,y)} * [I(x,y) - u(x,y)]$$

Donde:

- x: coordenada horizontal del píxel procesado.
- y: coordenada vertical del píxel procesado.
- $I(x,y)$: valor original del píxel en la posición (x, y)de la imagen con ruido.
- $u(x,y)$: media local en la ventana vecina (por ejemplo, 5x5)
- $\sigma^2(x,y)$: varianza local en la misma ventana
- n^2: varianza estimada del ruido
- $I'(x,y)$: valor filtrado del píxel



Figura 28. Filtro Adaptativo.

Fuente: Autor

La **¡Error! No se encuentra el origen de la referencia.** presenta, a la izquierda, una imagen en escala de grises a la que se ha incorporado ruido impulsivo de tipo sal y pimienta. A la derecha se muestra el resultado obtenido después de aplicar el filtro adaptativo. Este procedimiento utiliza información estadística del vecindario de cada píxel, como la media y la varianza local, junto con una estimación de la varianza del ruido, para ajustar el proceso de filtrado. Como resultado, se observa una reducción de las alteraciones producidas por el ruido y una mejora visual de la imagen procesada, procurando conservar sus principales características estructurales.

1.4.10 Operadores Lógicos o Booleanos

Las operaciones lógicas o booleanas son herramientas fundamentales en el procesamiento de imágenes digitales, especialmente en sistemas de

visión artificial. Estas operaciones permiten combinar o modificar imágenes binarias (con valores de píxeles de 0 o 1) aplicando principios de la lógica booleana (L. Wang et al., 2023). Son esenciales para tareas como:

- Combinación de máscaras de segmentación
- Detección de regiones comunes o exclusivas entre objetos
- Refinamiento de resultados tras umbralización o filtrado

Operación AND: es una de las más utilizadas en visión artificial para combinar dos imágenes binarias. El resultado de esta operación es una nueva imagen en la que los píxeles solo son activados (valor 1) si ambos píxeles correspondientes en las imágenes originales también están activados.

Sea A (x, y) y B(x,y) dos imágenes binarias (valores 0 o 1), la operación AND se define como:

$$R(x, y) = A(x, y) \wedge B(x, y)$$

Donde:

- x: coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $R(x, y)$: es la imagen resultante.
- $A(x, y)$: valor del píxel de la primera imagen binaria en la posición (x, y).
- $B(x, y)$: valor del píxel de la segunda imagen binaria en la posición (x, y).El valor del píxel será 1 (o 255 en imágenes

binarias) si ambos píxeles $A(x, y)$ y $B(x, y)$ están activados. En los demás casos, el resultado será 0..

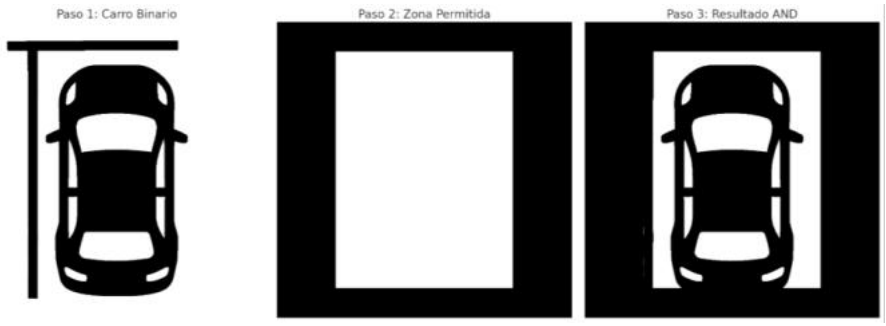


Figura 29. Imagen con Operación AND.

Fuente: Autor.

La **¡Error! No se encuentra el origen de la referencia.** muestra una imagen binaria del vehículo, donde su silueta aparece en negro sobre un fondo blanco, representando la detección del auto por parte del sistema. En el segundo paso se muestra la zona permitida de parqueo, también como una imagen binaria, en la que el área blanca indica el espacio válido para estacionar. Finalmente, en el tercer paso se presenta el resultado de aplicar la operación AND entre ambas imágenes: se conservan únicamente las áreas donde coinciden los píxeles blancos, es decir, donde el vehículo se encuentra dentro del espacio autorizado. Las zonas del carro que exceden los límites permitidos desaparecen del resultado, lo cual permite identificar visualmente si el parqueo es correcto o si existe alguna infracción. Esta técnica resulta clave para sistemas inteligentes de control de parqueo, ya que automatiza la verificación de la ubicación adecuada del vehículo.

Operación OR: El operador lógico OR (O inclusiva) es una función booleana que se utiliza ampliamente en el procesamiento digital de imágenes, especialmente cuando se trabaja con imágenes binarias (compuestas solo por píxeles de valor 0 o 255).

Cuando se aplica entre dos imágenes, evalúa cada píxel de forma individual y devuelve un valor blanco (255) si al menos uno de los píxeles en esa posición es blanco. Si ambos píxeles son negros (0), entonces el resultado es negro.

Sea A (x, y) y B(x,y) dos imágenes binarias (valores 0 o 1), la operación OR se define como:

$$R(x, y) = A(x, y) \vee B(x, y)$$

Donde:

- x: coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $A(x, y)$: valor del píxel de la primera imagen binaria.
- $B(x, y)$: valor del píxel de la segunda imagen binaria.
- $R(x, y)$: es la imagen resultante.
- El valor del píxel será 1 (o 255 en imágenes binarias), si al menos uno de los píxeles $A(x, y)$ o $B(x, y)$ está activado. Si ambos son 0, el resultado será 0..



Figura 30. Imagen con Operación OR.

Fuente: Autor.

En la **¡Error! No se encuentra el origen de la referencia.** se muestra a la izquierda una detección parcial del vehículo mediante una cámara de vigilancia, que capta la silueta blanca del cuerpo del auto sobre un fondo negro. En el centro, se presenta la detección realizada por sensores de presión ubicados en el suelo, los cuales identifican únicamente las ruedas del vehículo como dos áreas blancas separadas. Finalmente, a la derecha, el resultado de aplicar OR combina ambas fuentes de información, generando una imagen binaria que confirma la ocupación del espacio de parqueo, al mostrar en blanco todas las áreas donde al menos uno de los sistemas detectó la presencia del automóvil. Esta fusión asegura una mayor precisión en la detección y reduce los errores en la gestión automatizada del parqueo.

Operación XOR: se utiliza en visión artificial para identificar las diferencias entre dos imágenes binarias. La operación devuelve un píxel blanco (255) únicamente cuando los valores de entrada son diferentes. Es decir, si un sensor detecta un objeto y el otro no, esa diferencia será resaltada.

Esto permite detectar inconsistencias entre sistemas de detección, como cuando un sensor de presión indica la presencia de un vehículo, pero la cámara no lo registra (o viceversa), lo que puede alertar sobre fallas, obstrucciones visuales, o la necesidad de recalibración del sistema.

Sea A (x, y) y B(x,y) dos imágenes binarias (valores 0 o 1), la operación XOR se define como:

$$R(x,y) = A(x,y) \oplus B(x,y)$$

Donde:

- x: coordenada horizontal del píxel.
- y: coordenada vertical del píxel.
- $A(x, y)$: valor del píxel de la primera imagen binaria.
- $B(x, y)$: valor del píxel de la segunda imagen binaria.
- $R(x,y)$: es la imagen resultante.
- El valor del píxel será 1 (o 255 en imágenes binarias), solo si $A(x,y)$ o B (x,y) es igual a 1 (o 255) pero no ambos al mismo tiempo.
- En todos los demás casos, es decir, si ambos píxeles son 0 o ambos son 1, el resultado será 0.

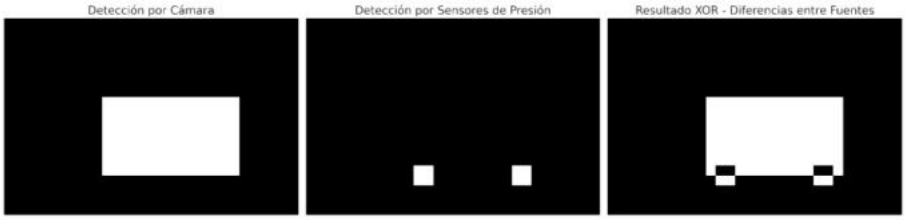


Figura 31. Imagen con Operación XOR.

Fuente: Autor

En la **¡Error! No se encuentra el origen de la referencia.** se observa a la izquierda la detección del vehículo mediante una cámara, que capta gran parte del cuerpo del automóvil. En el centro, los sensores de presión identifican solamente las ruedas, representadas como dos áreas blancas. A la derecha, el resultado del operador XOR revela únicamente las regiones donde las detecciones no coinciden: se destacan en blanco las zonas detectadas por la cámara que no fueron confirmadas por los sensores, y viceversa. Esta imagen permite visualizar las diferencias entre ambas fuentes, siendo útil para detectar inconsistencias o errores de calibración entre sensores en un sistema de parqueo automatizado.

1.5 Operaciones Morfológicas

Los filtros morfológicos son operadores no lineales aplicados principalmente a imágenes binarias o en escala de grises que utilizan un elemento estructurante para explorar la forma, estructura o geometría de los objetos presentes en una imagen. A diferencia de los filtros lineales que operan sobre intensidades, los filtros morfológicos analizan las relaciones espaciales entre los píxeles, permitiendo modificar las formas según su vecindario (Jiang & Scott, 2020).

Erosión: Es un operador morfológico fundamental que consiste en reducir las regiones blancas (foreground) de una imagen binaria. Su principio es aplicar un elemento estructurante (por ejemplo, un cuadrado 3×3) y conservar un píxel solo si todos los píxeles del vecindario definido por el elemento estructurante también están activados (valor 1 o 255).

Fórmula aplicada

$$A \ominus B = \{z \in E \mid B_z \subseteq A\}$$

Donde:

- z: punto al cual se traslada el elemento estructurante.
- A: imagen binaria original
- B: elemento estructurante
- B_z: traslación del punto B al z
- $A \ominus B$: imagen resultante de la erosión

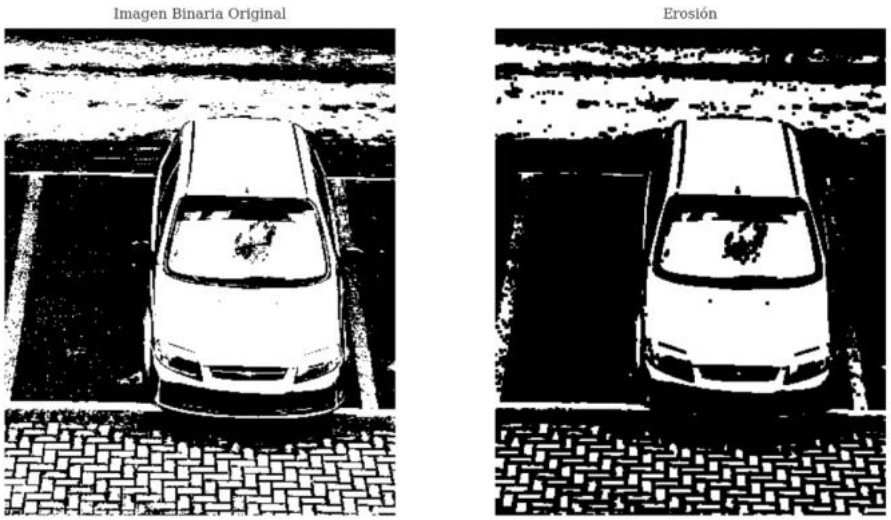


Figura 32. Imagen con Erosión.

Fuente: Autor.

La **¡Error! No se encuentra el origen de la referencia.** presenta una comparación entre una imagen binaria original (izquierda) y su versión erosionada (derecha), obtenida mediante la aplicación de un operador morfológico con un elemento estructurante rectangular de 3×3 píxeles. En esta operación, se eliminan los píxeles blancos (valor 255) que no están completamente rodeados por vecinos también blancos, lo que provoca un adelgazamiento de las estructuras, contracción de los bordes y desaparición de detalles finos o aislados. Esta transformación es útil para eliminar ruido blanco disperso, separar objetos conectados por puntos estrechos o preparar la imagen para etapas posteriores como la apertura o la segmentación precisa de objetos.

Dilatación: expande las regiones blancas (objetos) en una imagen binaria. Se aplica un elemento estructurante sobre la imagen y se activa

un píxel si al menos uno de los píxeles vecinos bajo la forma del elemento estructurante está activado (blanco). Este operador aumenta el tamaño de los objetos, conecta regiones cercanas y rellena pequeños huecos o interrupciones.

Fórmula aplicada

$$A \oplus B = \left\{ z \in E \,|\left(\hat{B}\right)_z \cap A \neq 0 \right\}$$

Donde:

- z: punto al cual se traslada el elemento estructurante.
- A: imagen binaria original
- B: elemento estructurante
- $\hat{B}$: reflexión de B
- $\left(\hat{B}\right)_z$: traslación del elemento estructurante al punto z
- $A \oplus B$: imagen resultante de la dilatación

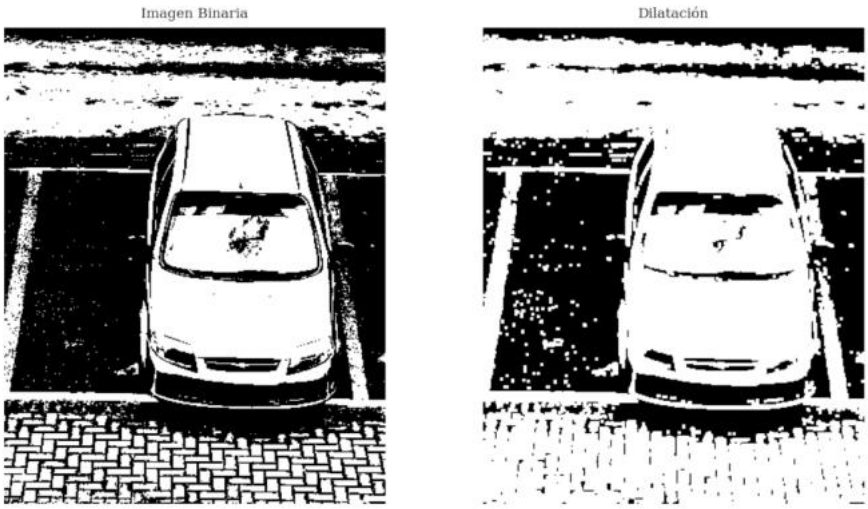


Figura 33. Imagen con Dilatación.

Fuente: Autor.

La**¡Error! No se encuentra el origen de la referencia.** presenta el efecto del operador de dilatación aplicado sobre una imagen binaria. A la izquierda se encuentra la imagen original y a la derecha, su transformación. Se ha utilizado un elemento estructurante rectangular de 3×3 píxeles, lo que permite expandir las regiones blancas al incorporar los píxeles vecinos. Como resultado, los objetos se agrandan, los bordes se desplazan hacia fuera, se rellenan huecos internos pequeños y se conectan regiones que estaban cercanas, pero no unidas. Esta operación es especialmente útil en la corrección de cortes finos, el fortalecimiento de líneas o la preparación para segmentaciones más limpias.

Apertura: es una operación morfológica compuesta que se define como una erosión seguida de una dilatación. Su objetivo principal es eliminar pequeños objetos o ruido blanco sin afectar significativamente la forma general de los objetos más grandes.

Fórmula aplicada

$$A \circ B = (A \ominus B) \oplus B$$

Donde:

- A: imagen binaria original
- B: elemento estructurante
- $\ominus$: erosión
- $\oplus$: dilatación
- $A \circ B$: resultado de la apertura

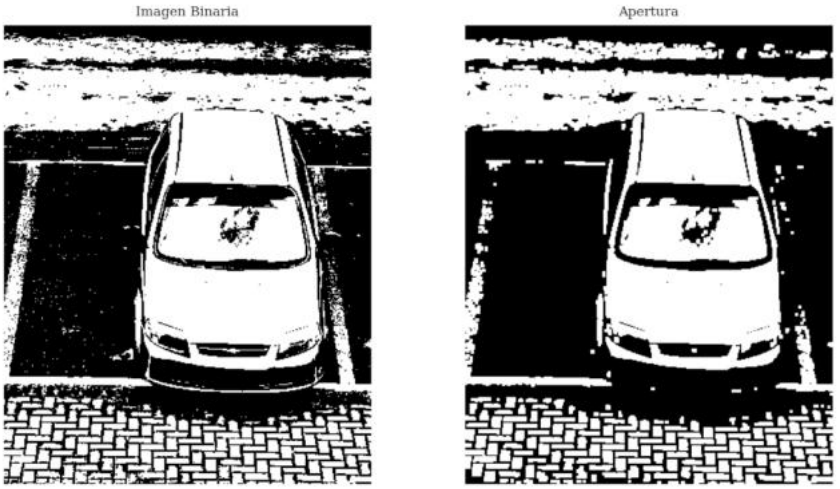


Figura 34. Imagen con Apertura.

Fuente: Autor.

La**¡Error! No se encuentra el origen de la referencia.** presenta el efecto de la operación de apertura sobre una imagen binaria. A la izquierda se muestra la imagen original, mientras que a la derecha se observa el resultado tras aplicar una erosión seguida de una dilatación

con un elemento estructurante de 3×3. Esta secuencia permite eliminar estructuras blancas pequeñas (ruido) y suavizar los contornos, sin expandir significativamente las regiones. La apertura es útil en tareas de preprocesamiento donde se requiere depurar la imagen de elementos espurios preservando la forma principal de los objetos.

Cierre: es una operación morfológica compuesta que consiste en una dilatación seguida de una erosión. Su objetivo es cerrar pequeños huecos o espacios negros dentro de objetos blancos, unir regiones separadas por líneas finas oscuras y suavizar contornos sin reducir el tamaño general del objeto.

Fórmula aplicada

$$A \cdot B = (A \oplus B) \ominus B$$

Donde:

- A: imagen binaria original
- B: elemento estructurante
- $\oplus$: dilatación
- $\ominus$: erosión
- $A \cdot B$: resultado del cierre

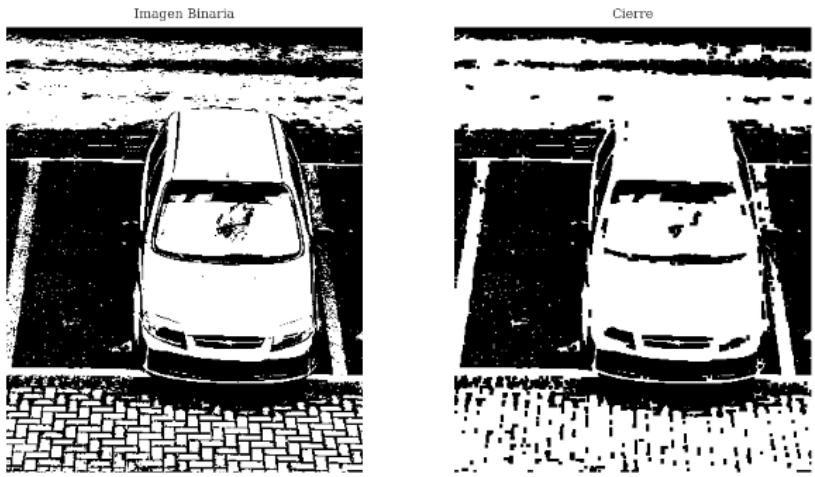


Figura 35. Imagen con Cierre.

Fuente: Autor.

La **¡Error! No se encuentra el origen de la referencia.** muestra una imagen binaria original (izquierda) y su versión procesada mediante la operación de cierre (derecha), utilizando un elemento estructurante de 3×3 píxeles. El cierre permite rellenar huecos pequeños y eliminar líneas negras angostas en el interior de los objetos blancos. Esta operación morfológica es especialmente útil para conectar elementos fragmentados y mejorar la continuidad de los objetos segmentados, preservando su forma general y sin eliminar sus estructuras principales.

1.6 Procesamiento y Análisis

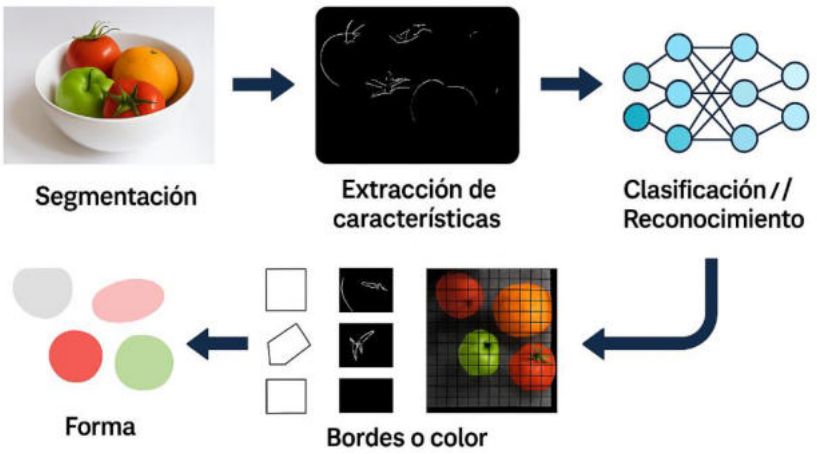


Figura 36. Procesamiento y análisis de las imágenes.

Esta etapa permite al sistema extraer información útil de la imagen (**¡Error! No se encuentra el origen de la referencia.**). Comienza con la segmentación, que separa los objetos del fondo. Luego se realiza la extracción de características, que pueden ser geométricas (forma), estructurales (bordes) o estadísticas (textura, color). Finalmente, se ejecuta un proceso de clasificación o reconocimiento, donde se determina qué representa cada objeto mediante algoritmos de aprendizaje supervisado o redes neuronales. Este procesamiento constituye el núcleo de la visión artificial, donde se simula la capacidad de entender el contenido visual (Dosovitskiy et al., 2020).

Tabla 4. Procesamiento y análisis de imágenes.

Componente	Descripción
Segmentación	Separa objetos o regiones relevantes del fondo, dividiendo la imagen en áreas homogéneas mediante técnicas de bordes, regiones o umbrales.
Extracción de características	Obtiene atributos cuantificables de las regiones segmentadas. Pueden ser geométricas (forma), estructurales (bordes) o estadísticas (textura, color).
Clasificación / Reconocimiento	Asocia las características extraídas a una categoría específica utilizando algoritmos como redes neuronales o aprendizaje supervisado.

Fuente: Autor.

Segmentación: tiene como objetivo dividir una imagen en regiones homogéneas que representen objetos, formas o zonas de interés. En términos simples, consiste en separar el fondo de los elementos relevantes, permitiendo que el sistema de visión artificial centre su atención en las partes útiles de la imagen.

Existen diversas técnicas de segmentación, que pueden clasificarse en tres grupos principales:

- Basadas en intensidad o umbralización: convierten una imagen en blanco y negro según un valor límite. Son simples y útiles cuando hay buen contraste.
- Basadas en bordes: detectan discontinuidades abruptas de intensidad para trazar contornos, utilizando operadores como Sobel o Canny.

- Basadas en regiones: agrupan píxeles que comparten características similares, como textura o color, usando algoritmos como crecimiento de regiones o clustering.



Figura 37. Técnicas de segmentación.

Fuente: Autor.

En la**¡Error! No se encuentra el origen de la referencia.** se presenta una comparación visual de cuatro técnicas de segmentación aplicadas a una fotografía aérea de un vehículo estacionado. En la primera sección, correspondiente a la umbralización, se observa una separación clara entre el automóvil y el fondo mediante el uso de blanco y negro, facilitando su detección. En la segunda técnica, la detección de bordes, se delinean con precisión los contornos del vehículo, resaltando sus límites estructurales. La tercera imagen muestra la segmentación basada en el crecimiento de regiones, donde se agrupan los píxeles similares en torno a una semilla inicial, conservando detalles y formas reales. Finalmente, en la técnica de K-means, se visualiza una agrupación por colores que divide la escena en regiones homogéneas, destacando el vehículo frente a su entorno inmediato. Esta representación permite

entender cómo cada método segmenta una misma escena desde diferentes enfoques computacionales.

Extracción de características: permite convertir regiones segmentadas en información cuantificable y significativa. Consiste en obtener atributos específicos de las áreas de interés para facilitar su análisis, clasificación o reconocimiento posterior por parte del sistema de visión artificial. Estas características pueden ser de distintos tipos y su elección depende del objetivo del análisis y del tipo de objeto en estudio.



Figura 38. Extracción de característica.

Fuente: Autor.

La**¡Error! No se encuentra el origen de la referencia.** ilustra el proceso de extracción de características aplicado a un vehículo estacionado. En ella se presentan tres etapas clave dispuestas de forma horizontal sobre fondo blanco. A la izquierda, se observa la imagen original, mostrando un automóvil plateado ubicado en un espacio de parqueo. En el centro, se representa la forma segmentada del vehículo, aislada del entorno. Finalmente, a la derecha se muestra una versión esquemática del auto. Esta secuencia gráfica permite comprender cómo

un sistema inteligente transforma una imagen en datos para analizar y reconocer objetos dentro de un entorno estructurado.

Clasificación/Reconocimiento: el sistema asigna una etiqueta o categoría específica a los objetos detectados en la imagen. Utiliza los datos obtenidos en las fases anteriores (segmentación y extracción de características) para determinar qué representa cada objeto y a qué clase pertenece.

Este proceso puede realizarse mediante técnicas tradicionales como máquinas de soporte vectorial (SVM), árboles de decisión, o mediante algoritmos más avanzados como las redes neuronales convolucionales (CNN), que han demostrado alta precisión en tareas visuales complejas.

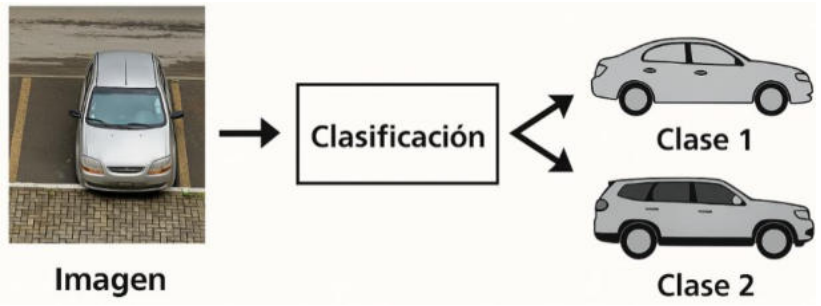


Figura 39. Clasificación/Reconocimiento de imágenes.

Fuente: Autor.

La **¡Error! No se encuentra el origen de la referencia.** ilustra el flujo del proceso de clasificación dentro de un sistema de visión artificial, utilizando como entrada una imagen aérea de un vehículo estacionado. Esta imagen es procesada a través de un módulo de clasificación que actúa como unidad de decisión, en la cual se analizan las características

previamente extraídas del objeto segmentado. A partir de este análisis, el sistema genera una salida que corresponde a una de dos clases predefinidas: “Clase 1” o “Clase 2”.

Análisis del contenido: en la que el sistema interpreta los datos obtenidos del reconocimiento y los convierte en acciones, decisiones o respuestas automatizadas. A diferencia de la clasificación, que simplemente asigna una etiqueta, esta fase busca darle sentido al objeto detectado en su contexto, simulando una forma básica de comprensión similar a la percepción humana.

1.7 Interpretación

La interpretación traduce los resultados del análisis en una acción concreta. En esta fase, el sistema decide qué hacer con la información visual obtenida: activar una alarma, registrar un evento, guiar un robot, autorizar acceso, entre otras funciones. Esta etapa también puede incluir retroalimentación, en la que el sistema ajusta parámetros automáticamente en tiempo real para mejorar su rendimiento. La interpretación marca el paso de la percepción visual artificial a la acción inteligente (Kirillov et al., 2023).

Tabla 5. Interpretación de los resultados.

Elemento	Descripción
¿Qué es?	Es el momento en que el sistema decide qué hacer con la información que analizó.
¿Qué hace?	Toma una acción concreta basada en lo que "entendió" de la imagen.

Elemento	Descripción
Ejemplos	– Activar una alarma – Guiar un robot – Autorizar el acceso – Guardar un registro
¿Por qué es útil?	Porque convierte la visión artificial en algo práctico: actúa, no solo analiza.
¿Ajusta su comportamiento?	Sí. Puede mejorarse solo en tiempo real gracias a la retroalimentación.

Fuente: Autor.

En un sistema de visión artificial es el momento en que el sistema define qué debe hacer con la información visual que ha analizado. Una vez que el procesamiento y el análisis de imagen han extraído características relevantes (como la detección de objetos, la clasificación o la identificación de patrones), el sistema usa esa información para generar una respuesta concreta que puede integrarse con otros sistemas o mecanismos de control.

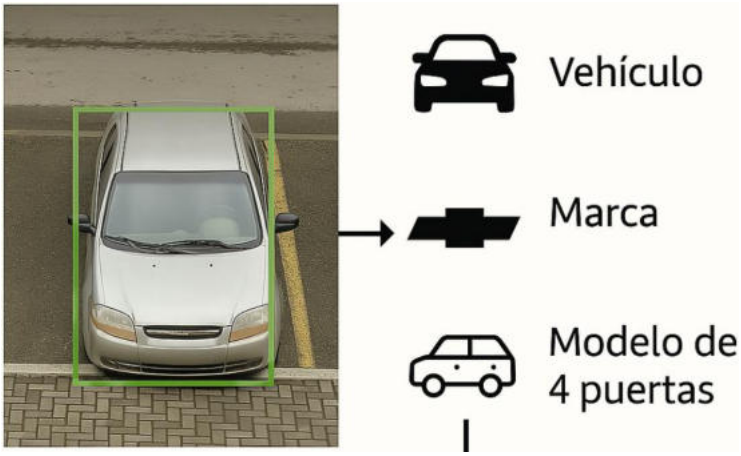


Figura 40. Interpretación de imágenes.

Fuente: Autor.

La **¡Error! No se encuentra el origen de la referencia.** presenta la interpretación en un sistema de visión artificial aplicado a un entorno de parqueo. Se muestra a la izquierda una imagen aérea de un vehículo identificado por un recuadro verde. A partir de esta detección, se despliegan tres niveles de análisis: la categoría general del objeto (Vehículo), un campo de metadato etiquetado como Marca, y una característica morfológica asociada al diseño del automóvil (Modelo de 4 puertas). Esta disposición refleja cómo, tras el reconocimiento visual, el sistema interpreta semánticamente la información obtenida para clasificar el objeto dentro de una estructura jerárquica de conocimiento, esencial para tareas de análisis avanzado y toma de decisiones automatizadas.

1.8 Resumen

En el Capítulo I de Fundamentos de Visión Artificial introduce los conceptos esenciales que permiten comprender cómo los sistemas computacionales adquieren, procesan e interpretan información visual del entorno. Se define la visión artificial como una disciplina interdisciplinaria que integra principios de la inteligencia artificial, el procesamiento digital de imágenes y la computación, con el objetivo de emular la capacidad humana de percepción visual mediante algoritmos y modelos matemáticos.

Asimismo, se describen los componentes básicos de un sistema de visión artificial, explicando el rol de los sensores de imagen, el proceso de adquisición y la transformación de la información visual en datos

digitales. Se enfatiza la importancia de estas etapas iniciales, ya que la calidad de la imagen capturada influye directamente en el rendimiento de los procesos posteriores de análisis y toma de decisiones.

El texto aborda los principios fundamentales del procesamiento digital de imágenes, incluyendo la representación de imágenes mediante píxeles, los diferentes tipos de imágenes digitales y los conceptos asociados a resolución e intensidad. Estos fundamentos permiten comprender cómo se aplican operaciones básicas para mejorar, analizar o segmentar imágenes, sentando las bases técnicas necesarias para el uso de algoritmos más avanzados.

Finalmente, el apartado establece el marco conceptual que sustenta el desarrollo de aplicaciones prácticas de la visión artificial. Al consolidar los fundamentos teóricos y técnicos, se prepara al lector para comprender la aplicación de estas tecnologías en sistemas inteligentes, facilitando la transición desde los conceptos básicos hacia soluciones reales basadas en visión artificial.

CAPÍTULO II

2 IMPLEMENTACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA CONTROL DE ESTACIONAMIENTO

2.1 Introducción

La gestión inteligente de espacios de estacionamiento representa un reto creciente debido al aumento constante del parque automotor y la limitada disponibilidad de áreas de parqueo. En muchos contextos, la búsqueda de un espacio disponible contribuye significativamente a la congestión vehicular, el desperdicio de tiempo y el incremento de emisiones contaminantes, lo que afecta tanto la movilidad como la calidad de vida de la población. La literatura contemporánea en ingeniería y ciencias computacionales indica que los sistemas de visión artificial tienen un potencial significativo para abordar estas problemáticas mediante la automatización del monitoreo de ocupación de espacios, superando las limitaciones de métodos tradicionales basados en sensores o conteo manual (Encío et al., 2023).

La visión artificial aplicada al control de estacionamientos vehiculares se fundamenta en técnicas de análisis de imágenes digitales y modelos de aprendizaje profundo que permiten identificar objetos y patrones dentro de escenas complejas. En particular, las arquitecturas neuronales profundas han demostrado una elevada precisión en tareas de clasificación y detección visual, ofreciendo resultados robustos incluso en condiciones lumínicas adversas, variaciones de perspectiva y oclusiones parciales de objetos (Encío et al., 2023).

Un enfoque destacado en la literatura científica reciente propone sistemas que utilizan redes neuronales convolucionales profundas (CNN) combinadas con métodos de detección de objetos como YOLO (You Only Look Once) para identificar la presencia de vehículos y determinar, de forma automática, el estado de ocupación de cada espacio disponible en un estacionamiento en tiempo real. Estos sistemas no solo ofrecen buenos resultados frente a enfoques basados en sensores físicos, sino que también abren la posibilidad de aprovechar cámaras ya instaladas en infraestructuras urbanas, reduciendo costos de implementación y mantenimiento (Noura Alshareef et al., 2025).

El uso de Datasets públicos y propios, que contienen decenas de miles de imágenes etiquetadas de espacios de estacionamiento en múltiples condiciones ambientales y angulaciones de cámara, ha sido clave para evaluar y comparar el desempeño de diferentes modelos de clasificación de ocupación vehicular. Dichos conjuntos de datos permiten explorar la generalización de los algoritmos y su capacidad para asumir variabilidad real en la escena de parqueo, lo cual es fundamental para construir sistemas robustos y aplicables fuera de entornos controlados (Yuldashev et al., 2023).

Este segundo capítulo se centra en la implementación práctica de un sistema de visión artificial para el control de estacionamientos, guiando al lector desde la descripción y preparación del conjunto de datos, pasando por la selección y entrenamiento de modelos, hasta la evaluación y análisis de resultados. El contenido está concebido con un enfoque metodológico claro y replicable, que integra evidencia científica actual con pasos prácticos y explicaciones detalladas

adecuadas para estudiantes y profesionales interesados en sistemas inteligentes basados en visión por computador.

2.2 Objetivo

- Desarrollar un sistema inteligente de control de estacionamientos vehiculares mediante técnicas de aprendizaje profundo, empleando conjuntos de datos públicos y propios bajo distintas condiciones ambientales, para la generación de un modelo robusto aplicable en escenarios reales fuera de entornos controlados.

2.3 Arquitectura del sistema

La arquitectura para el control inteligente de estacionamientos vehiculares basada en visión artificial se estructura a partir de un conjunto de etapas que corresponden al proceso de desarrollo de un modelo. Dichas etapas incluyen la adquisición de datos visuales, preprocesamiento de las imágenes, el entrenamiento del modelo, la evaluación de su desempeño y visualización de los resultados obtenidos (Verma et al., 2025). Esta organización se fundamenta en enfoques adoptados en estudios recientes de visión artificial aplicados a estacionamientos, donde la separación clara de cada fase del desarrollo permite comprender el funcionamiento del sistema, facilitar su implementación y garantizar un comportamiento escalable y robusto en escenarios reales.

2.4 Adquisición de datos

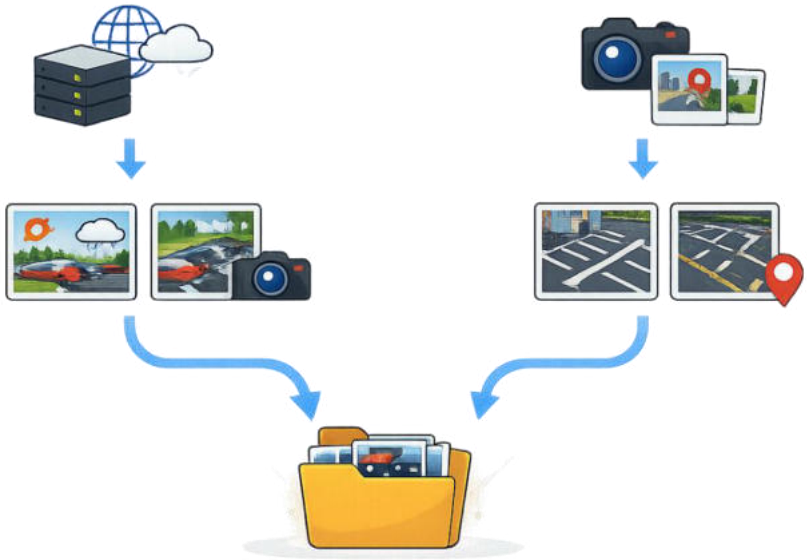


Figura 41. Esquema de adquisición de datos.

Fuente: Autor.

La adquisición de datos constituye la primera y una de las etapas más determinantes en el desarrollo de sistemas de visión artificial basados en aprendizaje profundo. En aplicaciones de control inteligente de estacionamientos vehiculares, esta etapa se orienta a recopilar imágenes representativas de escenarios reales, considerando variaciones en las condiciones ambientales, diferentes perspectivas de cámara y diversos niveles de ocupación de los espacios. Estudios recientes destacan que la diversidad y calidad de los datos de entrada influyen de manera directa en la capacidad del modelo para generalizar su comportamiento en entornos no controlados (Verma et al., 2025).

Se adopta una estrategia de adquisición basada en el uso combinado de Datasets públicos y propios, con el objetivo de disponer de un conjunto de datos amplio, heterogéneo y representativo. Los Datasets públicos permiten aprovechar bases de datos consolidadas y validadas en la literatura científica, mientras que los Datasets propios facilitan la adaptación del modelo a condiciones específicas del entorno donde se prevé su aplicación(Wei et al., 2025).

Entre los Datasets públicos empleados se considera CNRPark+EXT, el cual contiene decenas de miles de imágenes etiquetadas de espacios de estacionamiento capturadas mediante cámaras fijas en escenarios reales. Este conjunto de datos incorpora variaciones significativas de iluminación, condiciones climáticas y ángulos de captura, lo que lo convierte en una referencia adecuada para evaluar modelos de clasificación de ocupación vehicular en condiciones cercanas a la realidad (Grbić & Koch, 2023).

Asimismo, los Datasets propios se generan mediante la captura de imágenes en entornos específicos de estacionamiento, permitiendo incorporar características particulares como la disposición de los espacios, el tipo de señalización horizontal y la configuración de las cámaras. La inclusión de estos datos contribuye a mejorar la robustez del modelo y a reducir el sesgo asociado al entrenamiento exclusivo con datos externos.

En conjunto, la estrategia de adquisición de datos planteada en este capítulo busca sentar una base sólida para el desarrollo del sistema de visión artificial, garantizando que el modelo sea entrenado y evaluado

sobre información diversa y realista, condición indispensable para su correcta aplicación en escenarios reales fuera de entornos controlados.

```
# ======================================
# 1) INSTALAR DEPENDENCIAS
# ======================================
!pip install -q pillow tqdm pandas imagehash albumentations opencv-python torch torchvision scikit-learn

# ======================================
# 2) DESCARGAR Y DESCOMPRIMIR ZIP
# ======================================
url = "https://github.com/fabiocarrara/deep-parking/releases/download/archive/CNRPark-Patches-150x150.zip"
print("📥 Descargando dataset...")
!wget -q --show-progress {url} -O CNRPark-Patches-150x150.zip

print("📦 Descomprimiendo...")
!unzip -q CNRPark-Patches-150x150.zip -d CNRPark_Patches

dataset_root = "/content/CNRPark_Patches/CNRPark-Patches-150x150"
print("Dataset raíz:", dataset_root)
```

Figura 42. Descarga de datos externos CNRPark-Patches.

Fuente: Autor.

Para operacionalizar la adquisición de datos con Datasets públicos, se descargó el archivo comprimido CNRPark-Patches-150x150.zip, que constituye una porción segmentada del conjunto de datos *CNRPark+EXT*. Este archivo contiene imágenes de plazas de estacionamiento etiquetadas manualmente como vacías (“free”) u ocupadas (“busy”) organizadas en carpetas por cámara y clase, las cuales se descargan automáticamente dentro del entorno de procesamiento para su posterior preprocesamiento y análisis. La estructura de las imágenes extraídas permite identificar con precisión el estado de cada espacio de estacionamiento a partir de las etiquetas presentes en la ruta de cada archivo, lo que facilita la construcción de un Dataset etiquetado listo para tareas de clasificación binaria mediante aprendizaje profundo y modelos de visión por computador

(CNRPark+EXT es un subconjunto extendido del Dataset original *CNRPark* disponible públicamente) (Amato et al., 2017).

CNRPark+EXT, desarrollado por Amato y colaboradores, es un Dataset ampliamente citado en la literatura para la detección de ocupación de estacionamientos mediante CNN, que extiende la colección original de *CNRPark* (≈12 000 imágenes) capturados bajo diversas condiciones de iluminación, clima y perspectiva de cámara, lo que favorece entrenar modelos con mayor capacidad de generalización en escenarios reales y no controlados.

```
import os
from pathlib import Path


# Ajusta esta ruta si cambiaste donde están tus imágenes
dataset_root = Path("/content/CNRPark_Patches")

valid_ext = {".jpg", ".jpeg", ".png", ".bmp"}
all_files = list(dataset_root.rglob("*"))
images = [p for p in all_files if p.suffix.lower() in valid_ext]

print("[redacted] Total imágenes:", len(images))
```

Figura 43. Conteo de imágenes CNRPark-Patches.

Fuente: Autor.

Se establece una ruta base (dataset_root) que apunta al directorio donde se ha descomprimido el Dataset *CNRPark-Patches-150x150*. A continuación, se define un conjunto de extensiones de archivo (valid_ext) que corresponden a los formatos de imagen que se considerarán válidos (por ejemplo, .jpg, .png). Utilizando el método rglob("*"), se recorre de forma recursiva todo el árbol de directorios desde la ruta raíz hasta los niveles más profundos, y se filtran únicamente aquellos archivos cuya

extensión coincide con las incluidas en valid_ext. El resultado de esta operación es una lista de rutas a imágenes, la cual se utiliza para obtener estadísticas básicas, como el número total de imágenes disponibles para su uso en el preprocesamiento y entrenamiento de modelos.

Este procedimiento es crucial para garantizar que el conjunto de datos se haya descargado y descomprimido correctamente y para confirmar que existen suficientes imágenes que cumplan con los criterios esperados, evitando errores posteriores en las fases de procesamiento. La inspección de la estructura y cantidad de imágenes antes de iniciar la preparación del Dataset es una práctica recomendada en tareas de visión por computador para asegurar la calidad y consistencia de los datos de entrada.

```
import os
import pandas as pd

data = []
label_map = {"free": 0, "busy": 1}
label_name = {0: "Libre", 1: "Ocupado"}

for escenario in os.listdir(dataset_root):
    esc_path = os.path.join(dataset_root, escenario)
    if os.path.isdir(esc_path):
        for clase in ["free", "busy"]:
            class_path = os.path.join(esc_path, clase)
            for img in os.listdir(class_path):
                data.append({
                    "path": os.path.join(class_path, img),
                    "label": label_map[clase],
                    "estado": label_name[label_map[clase]],
                    "escenario": escenario
                })

df = pd.DataFrame(data)
df.head()
```

Figura 44. Organización de las imágenes CNRPark-Patches.

Fuente: Autor.

2.5 Preprocesamiento de las imágenes

El preprocesamiento de datos constituye una de las fases más relevantes en el desarrollo de sistemas de visión artificial basados en aprendizaje profundo, debido a que la calidad y consistencia de los datos de entrada influyen directamente en el desempeño y la capacidad de generalización del modelo. En aplicaciones de control inteligente de estacionamientos vehiculares, las imágenes suelen presentar variaciones significativas en iluminación, presencia de sombras, condiciones climáticas adversas y diferencias en la perspectiva de captura, lo que hace indispensable una etapa de preparación previa antes del entrenamiento del modelo.

Estudios recientes destacan que un preprocesamiento adecuado contribuye a reducir el ruido visual, mejorar la estabilidad del proceso de aprendizaje y evitar problemas como el sobreajuste, especialmente cuando se trabaja con Datasets capturados en entornos reales y no controlados (Martynova et al., 2024; Yuldashev et al., 2023). En este contexto, el preprocesamiento se estructura en cuatro etapas principales: limpieza del conjunto de datos, estandarización del tamaño de las imágenes, normalización y aumento de datos, y división del Datasets para entrenamiento y evaluación.

La conversión de las imágenes a un formato estándar y la normalización de los valores de los píxeles constituyen el primer paso del preprocesamiento en sistemas de visión por computador. Este procedimiento consiste en transformar las imágenes al espacio de color RGB y escalar los valores de intensidad al rango [0,1] . En modelos de aprendizaje profundo, esta estandarización es fundamental, ya que evita

que valores elevados de píxeles dominen el proceso de aprendizaje y mejora la estabilidad numérica durante la optimización. En aplicaciones de parqueo inteligente, la normalización garantiza que imágenes capturadas bajo distintas condiciones de iluminación sean tratadas de forma consistente por la red neuronal (Stojnev D., 2020).

```python
import cv2
import numpy as np

def preprocesar_imagen(img_path, size=(150,150)):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, size)
    img = img.astype("float32") / 255.0
    return img
```

Figura 45. Normalización de píxeles.

Fuente: Autor.

La variación de iluminación puede dificultar la extracción de características relevantes de una imagen. La técnica de ecualización de histograma adaptativo con límite de contraste (CLAHE) mejora localmente el contraste de la imagen sin amplificar excesivamente el ruido. CLAHE calcula histogramas adaptativos en regiones separadas de la imagen, lo que permite resaltar detalles locales y ayuda a los modelos de visión por computador a identificar formas y bordes presentes en espacios de estacionamiento en condiciones heterogéneas de luz (Wang et al., 2023).

```python
def clahe_contraste(img):
    img_uint8 = (img * 255).astype("uint8")
    img_lab = cv2.cvtColor(img_uint8, cv2.COLOR_RGB2LAB)
    l, a, b = cv2.split(img_lab)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    l_clahe = clahe.apply(l)

    img_lab = cv2.merge((l_clahe, a, b))
    img_contraste = cv2.cvtColor(img_lab, cv2.COLOR_LAB2RGB)
    return img_contraste.astype("float32") / 255.0
```

Figura 46. Mejora de contraste (CLAHE).

Fuente: Autor.

El ruido visual en imágenes puede provenir de las limitaciones del sensor, condiciones ambientales u otros factores externos. La reducción de ruido mediante filtros (como el filtro gaussiano) suaviza las variaciones aleatorias mientras preserva los contornos más relevantes. Este paso es útil en sistemas de parqueo inteligente para evitar que artefactos no informativos dificulten el proceso de extracción de características por parte del modelo de clasificación.

```python
def reducir_ruido(img):
    img_uint8 = (img * 255).astype("uint8")
    img_suave = cv2.GaussianBlur(img_uint8, (5,5), 0)
    return img_suave.astype("float32") / 255.0
```

Figura 47. Reducción de ruido.

Fuente: Autor.

El aumento de datos (data augmentation) se refiere a la generación de nuevas imágenes artificiales a través de transformaciones geométricas y fotométricas (rotaciones, volteos, cambios de brillo, etc.). Estas

transformaciones preservan las etiquetas originales y ayudan a expandir la diversidad del conjunto de entrenamiento, lo cual es crucial cuando los datos son limitados. El uso de técnicas de aumento mejora la generalización del modelo frente a variaciones reales en el entorno, reduciendo la probabilidad de sobreajuste (Alomar et al., 2023).

```python
import albumentations as A

augmentations = A.Compose([
  A.HorizontalFlip(p=0.5),
  A.RandomBrightnessContrast(p=0.4),
  A.ShiftScaleRotate(shift_limit=0.05, scale_limit=0.1, rotate_limit=10, p=0.5)
])

def aplicar_augmentaciones(img):
    augmented = augmentations(image=(img*255).astype("uint8"))
    return augmented["image"].astype("float32") / 255.0
```

Figura 48. Aumento de datos (Data Augmentation).

Fuente: Autor.

2.6 Entrenamiento del modelo

Antes de entrenar un modelo de aprendizaje profundo, es fundamental dividir el Dataset en tres subconjuntos independientes: entrenamiento (training), validación (validation) y prueba (test). El conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo, el de validación para ajustar hiperparámetros y evitar el sobreajuste (overfitting), y el de prueba para evaluar la capacidad de generalización final del modelo en datos no vistos durante el entrenamiento.

```python
from sklearn.model_selection import train_test_split

# Dividir primero el dataset en entrenamiento+validación vs. prueba
train_val_df, test_df = train_test_split(
    df, test_size=0.2, random_state=42, stratify=df["label"]
)

# Dividir entrenamiento+validación en entrenamiento y validación
train_df, val_df = train_test_split(
  train_val_df, test_size=0.25, random_state=42, stratify=train_val_df["label"]
)
# 0.25 de 0.8 = 0.2 → 60% train, 20% val, 20% test

print("Entrenamiento:", len(train_df))
print("Validación:", len(val_df))
print("Prueba:", len(test_df))
```

Figura 49. División de datos (entrenamiento, validación y prueba).

Fuente: Autor.

La clase ParkingDataset es una implementación personalizada de un objeto Dataset en PyTorch, diseñada para encapsular la lógica de acceso y preprocesamiento de las imágenes en tu proyecto de parqueo inteligente. En aprendizaje profundo con PyTorch, un Dataset representa una colección de datos que puede ser indexada; cada vez que el DataLoader solicita un lote (batch), el Dataset debe devolver una muestra (input) y su etiqueta (target) correspondiente.

Esta clase extiende la clase base Dataset y define tres métodos fundamentales:

- __init__: inicializa el objeto con un DataFrame que contiene rutas y etiquetas de las imágenes, y un parámetro opcional augment que indica si se desea aplicar aumento de datos. Además, se llama a reset_index(drop=True) para asegurar que los índices estén

correctamente alineados (sin huecos) después de cualquier partición del Dataset.

- __len__: devuelve la cantidad de muestras disponibles en el Dataset. Esto es requerido por PyTorch para saber cuántos elementos hay, lo que permite calcular cuántos lotes producir por época.
- __getitem__: es el método principal del Dataset. Recibe un índice (idx) y debe retornar la muestra correspondiente. En este caso:

Se obtiene la ruta de la imagen desde el DataFrame.

Se llama a la función pipeline_preprocesamiento(...) para aplicar todas las transformaciones como normalización, corrección de contraste, reducción de ruido y opcionalmente data augmentation.

La imagen preprocesada se reordena con np.transpose(img, (2,0,1)) para pasar del formato HWC (alto-ancho-canales) al formato CHW (canales-alto-ancho) que espera PyTorch.

Finalmente se convierten tanto la imagen como la etiqueta en tensores de PyTorch de tipo float32 y Long, respectivamente, lo cual permite que puedan ser procesados por la red neuronal durante el entrenamiento.

Este diseño modular no solo mejora la legibilidad del código, sino que también permite reutilizar la lógica de preprocesamiento en diferentes fases (entrenamiento, validación y prueba) de forma consistente. En contextos de visión por computador, implementar un Dataset personalizado es una práctica estándar, ya que facilita manejar transformaciones complejas y cargas de datos de manera eficiente.

```python
import torch
from torch.utils.data import Dataset, DataLoader

class ParkingDataset(Dataset):
    def __init__(self, dataframe, augment=False):
        self.df = dataframe.reset_index(drop=True)
        self.augment = augment

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        ruta = self.df.loc[idx, "path"]
        img = pipeline_preprocesamiento(ruta, augment=self.augment)
        img = np.transpose(img, (2,0,1))
        img = torch.tensor(img, dtype=torch.float32)
        label = torch.tensor(self.df.loc[idx, "label"], dtype=torch.long)
        return img, label

train_dataset = ParkingDataset(train_df, augment=True)
val_dataset   = ParkingDataset(val_df, augment=False)
test_dataset  = ParkingDataset(test_df, augment=False)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader   = DataLoader(val_dataset, batch_size=32, shuffle=False)
test_loader  = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

Figura 50. Construcción de DataLoaders.

Fuente: Autor.

La clase ParkingCNN implementa una red neuronal convolucional (CNN), que es un tipo de modelo de aprendizaje profundo especialmente eficaz para tareas de clasificación de imágenes. Las CNN explotan la información espacial local de las imágenes mediante la aplicación de filtros (kernels) que aprenden automáticamente a detectar características discriminativas como bordes, texturas y formas, eliminando la necesidad de extracción manual de características (Zhao et al., 2024).

La arquitectura se compone de dos bloques de extracción de características (capas convolucionales seguidas de funciones de

activación y de pooling) y de capas totalmente conectadas que realizan la etapa de clasificación final:

- Capas convolucionales (nn.Conv2d): estas capas aplican filtros 2D sobre la imagen de entrada para extraer patrones locales. Cada filtro aprende un conjunto de pesos que responden a características visuales específicas, por ejemplo, bordes horizontales o verticales.
- Funciones de activación no lineal (F.relu): introducen no linealidad, permitiendo que la red aprenda relaciones complejas entre las características extraídas.
- Max-Pooling (nn.MaxPool2d): reduce la dimensionalidad espacial de los mapas de características, disminuyendo la cantidad de parámetros y agregando invarianza a pequeñas traslaciones en la imagen.
- Capas totalmente conectadas (nn.Linear): finalmente, las representaciones aprendidas se aplanan y se pasan a capas densas que combinan estas características de alto nivel para producir la salida de clasificación entre dos clases (Libre y Ocupado).

```python
import torch.nn as nn
import torch.nn.functional as F

class ParkingCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.pool = nn.MaxPool2d(2,2)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)

        self.fc1 = nn.Linear(64 * 37 * 37, 256)
        self.fc2 = nn.Linear(256, 2)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        return self.fc2(x)

model = ParkingCNN()
```

Figura 51. Modelo CNN.

Fuente: Autor.

En el **¡Error! No se encuentra el origen de la referencia.** se muestra el proceso completo de entrenamiento y validación de una red neuronal convolucional utilizando la biblioteca PyTorch, siguiendo el esquema clásico de aprendizaje supervisado. En primer lugar, se define el dispositivo de ejecución, priorizando el uso de GPU (CUDA) cuando está disponible, lo que permite acelerar significativamente el cálculo durante el entrenamiento. A continuación, se establece la función de pérdida CrossEntropyLoss, adecuada para problemas de clasificación multiclase, y el optimizador Adam, que ajusta automáticamente la tasa de aprendizaje de los parámetros del modelo para lograr una convergencia más eficiente. El entrenamiento se desarrolla a lo largo de

un número fijo de épocas (epochs), donde en cada iteración el modelo entra en modo entrenamiento (model.train()), procesa los datos provenientes del conjunto de entrenamiento en forma de lotes, calcula la pérdida entre las predicciones y las etiquetas reales, y actualiza los pesos mediante retropropagación del error. Posteriormente, el modelo se evalúa sobre el conjunto de validación (model.eval()), desactivando el cálculo de gradientes para evitar modificaciones en los parámetros aprendidos. Durante este proceso se registran las pérdidas promedio de entrenamiento y validación, lo que permite monitorear la evolución del aprendizaje, identificar posibles problemas de sobreajuste y analizar la capacidad de generalización del modelo antes de su evaluación final con datos de prueba.

```
import torch.nn as nn
import torch.optim as optim

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

num_epochs = 10
loss_train_history = []
loss_val_history   = []
```

```
for epoch in range(num_epochs):
    model.train()
    running_loss_train = 0.0

    for imgs, labels in train_loader:
        imgs, labels = imgs.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(imgs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss_train += loss.item()

    avg_loss_train = running_loss_train / len(train_loader)
    loss_train_history.append(avg_loss_train)


  model.eval()
  running_loss_val = 0.0
  with torch.no_grad():
      for imgs, labels in val_loader:
          imgs, labels = imgs.to(device), labels.to(device)
          outputs = model(imgs)
          loss = criterion(outputs, labels)
          running_loss_val += loss.item()

  avg_loss_val = running_loss_val / len(val_loader)
  loss_val_history.append(avg_loss_val)

  print(f"Epoch {epoch+1}/{num_epochs}")
  print(f"Loss Train: {avg_loss_train:.4f}")
  print(f" Loss Val: {avg_loss_val:.4f}")
```

Figura 52. Entrenamiento de la CNN.

Fuente: Autor.

2.7 Validación y visualización de resultados

El código presentado corresponde a la fase de validación del modelo de clasificación, cuyo objetivo es evaluar el desempeño de la red neuronal utilizando un conjunto de datos independiente del entrenamiento, sin modificar los pesos aprendidos. Para ello, el modelo se ejecuta en modo inferencia y, mediante el uso del contexto torch.no_grad(), se desactiva el cálculo de gradientes, optimizando el consumo de memoria y tiempo de cómputo. Durante esta fase, las imágenes y etiquetas provenientes del val_loader son procesadas en lotes, generándose predicciones que se comparan con las etiquetas reales para construir dos vectores: uno de valores verdaderos y otro de valores predichos. A partir de estos resultados se calculan métricas fundamentales de evaluación como accuracy, que mide la proporción global de clasificaciones correctas; precision, que cuantifica la fiabilidad de las predicciones positivas; recall, que evalúa la capacidad del modelo para identificar correctamente las instancias positivas; y F1-score, que equilibra precisión y recuperación en una sola medida. Además, se genera un reporte de clasificación detallado por clase y una matriz de confusión, cuya representación gráfica permite visualizar de forma intuitiva los aciertos y errores del modelo para cada categoría (Libre y Ocupado), facilitando el análisis del comportamiento del sistema y la detección de posibles sesgos o patrones de error antes de la fase final de prueba.

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score
from sklearn.metrics import recall_score, f1_score, classification_report
import matplotlib.pyplot as plt

# Listas para etiquetas reales y predichas
val_true = []
val_pred = []

with torch.no_grad():
    for imgs, labels in val_loader:
        imgs, labels = imgs.to(device), labels.to(device)
        outputs = model(imgs)
        _, preds = torch.max(outputs, 1)

        val_true.extend(labels.cpu().numpy())
        val_pred.extend(preds.cpu().numpy())

# Calcular métricas
val_acc  = accuracy_score(val_true, val_pred)
val_prec = precision_score(val_true, val_pred)
val_rec  = recall_score(val_true, val_pred)
val_f1   = f1_score(val_true, val_pred)
val_cm   = confusion_matrix(val_true, val_pred)
print(f"📊 Validación - Accuracy: {val_acc:.4f}")
print(f"🔎 Validación - Precision: {val_prec:.4f}")
print(f"♻️ Validación - Recall: {val_rec:.4f}")
print(f"📈 Validación - F1-Score: {val_f1:.4f}\n")

print("📋 Reporte de clasificación (Validación):")
print(classification_report(val_true, val_pred,
                            target_names=["Libre","Ocupado"], digits=4))
 # Matriz de confusión visual
 plt.figure(figsize=(6,5))
 plt.imshow(val_cm, interpolation="nearest", cmap=plt.cm.Blues)
 plt.title("Matriz de Confusión - Validación")
 plt.colorbar()
 plt.xticks([0,1], ["Libre","Ocupado"])
 plt.yticks([0,1], ["Libre","Ocupado"])
 plt.ylabel("Clase Real")
 plt.xlabel("Clase Predicha")

 thresh = val_cm.max()/2
 for i in range(val_cm.shape[0]):
     for j in range(val_cm.shape[1]):
         plt.text(j, i, val_cm[i,j],
                  horizontalalignment="center",
                  color="white" if val_cm[i,j] > thresh else "black")
 plt.tight_layout()
 plt.show()
```

Figura 53. Validación y visualización de resultados.

Fuente: Autor.

2.8 Resumen

El Capítulo 2 desarrolla la implementación de un sistema de visión artificial aplicado al control inteligente de estacionamientos vehiculares, abordando la problemática de la gestión ineficiente de espacios de parqueo en contextos urbanos. El capítulo se enfoca en el uso de técnicas de procesamiento digital de imágenes y aprendizaje profundo para automatizar la detección de ocupación vehicular, evidenciando el potencial de la visión artificial como una alternativa eficiente frente a soluciones tradicionales basadas en sensores físicos.

A lo largo del capítulo se presenta la arquitectura general del sistema, estructurada en etapas claramente definidas que permiten comprender el flujo completo del procesamiento visual. Se describe la fase de adquisición de datos, en la cual se emplean conjuntos de datos públicos reconocidos y datos propios, con el propósito de disponer de imágenes representativas de diferentes escenarios, condiciones de iluminación y niveles de ocupación, fortaleciendo la capacidad de generalización del sistema.

El capítulo también aborda el preprocesamiento de las imágenes como una etapa fundamental para garantizar la calidad de los datos de entrada. En esta fase se aplican técnicas orientadas a la reducción de ruido, normalización y adecuación visual de las imágenes, facilitando la posterior extracción de características relevantes mediante modelos de aprendizaje profundo. Sobre esta base, se expone el proceso de entrenamiento de modelos de visión artificial, priorizando el uso de

redes neuronales convolucionales para la detección y clasificación de vehículos.

Finalmente, se define el enfoque de evaluación del sistema propuesto, orientado a verificar su desempeño en escenarios cercanos a condiciones reales de operación. El capítulo integra fundamentos teóricos con procedimientos prácticos y replicables, permitiendo comprender el proceso completo de diseño, implementación y validación de un sistema de visión artificial aplicado al estacionamiento inteligente, consolidando así la transición de la teoría a una solución funcional.

BIBLIOGRAFÍA

Alomar, K., Aysel, H. I., & Cai, X. (2023). Data Augmentation in Classification and Segmentation: A Survey and New Strategies. Journal of Imaging, 9(2). https://doi.org/10.3390/jimaging9020046

Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., & Vairo, C. (2017). Deep learning for decentralized parking lot occupancy detection. Expert Systems with Applications, 72, 327–334. https://doi.org/https://doi.org/10.1016/j.eswa.2016.10.055

Archana, R., & Jeevaraj, P. S. E. (2024). Deep learning models for digital image processing: a review. Artificial Intelligence Review, 57(1). https://doi.org/10.1007/s10462-023-10631-z

Awaluddin, B. A., Chao, C. T., & Chiou, J. S. (2023). Investigating Effective Geometric Transformation for Image Augmentation to Improve Static Hand Gestures with a Pre-Trained Convolutional Neural Network. Mathematics, 11(23). https://doi.org/10.3390/math11234783

Bae, Y. Y., Cho, D. J., & Jung, K. H. (2025). A New Log-Transform Histogram Equalization Technique for Deep Learning-Based Document Forgery Detection. Symmetry, 17(3). https://doi.org/10.3390/sym17030395

Burger, W., & Burge, M. J. (2022). Digital image processing: An algorithmic introduction. Springer Nature.

Burnham, J., Hardy, J., & Meadors, K. (n.d.). Comparison of the Roberts, Sobel, Robinson, Canny, and Hough Image Detection Algorithms.

Carion Nicolas and Massa, F. and S. G. and U. N. and K. A. and Z. S. (2020). End-to-End Object Detection with Transformers. In H. and B. T. and F. J.-M. Vedaldi Andrea and Bischof (Ed.), Computer Vision – ECCV 2020 (pp. 213–229). Springer International Publishing.

de Mooij, C., & Martinez, M. (2024). A Critical Comparison of Shape Sensing Algorithms: The Calibration Matrix Method versus iFEM. Sensors, 24(11). https://doi.org/10.3390/s24113562

Dobson, J. E. (2023). The birth of computer vision. U of Minnesota Press.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. http://arxiv.org/abs/2010.11929

Encío, L., Díaz, C., del-Blanco, C. R., Jaureguizar, F., & García, N. (2023). Visual Parking Occupancy Detection Using Extended Contextual Image Information via a Multi-Branch Output ConvNeXt Network. Sensors, 23(6). https://doi.org/10.3390/s23063329

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Grbić, R., & Koch, B. (2023). Automatic vision-based parking slot detection and occupancy classification. Expert Systems with Applications, 225, 120147. https://doi.org/https://doi.org/10.1016/j.eswa.2023.120147

Guan, S., Liu, B., Chen, S., Wu, Y., Wang, F., Liu, X., & Wei, R. (2024). Adaptive median filter salt and pepper noise suppression approach for common path coherent dispersion spectrometer. Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-66649-y

Hwang, D., Kim, J. J., Moon, S., & Wang, S. (2025). Image Augmentation Approaches for Building Dimension Estimation in Street View Images Using Object Detection and Instance Segmentation Based on Deep Learning. Applied Sciences (Switzerland), 15(5). https://doi.org/10.3390/app15052525

Jiang, X. J., & Scott, P. J. (2020). Fundaments for free-form surfaces. Advanced Metrology, 11–34. https://doi.org/10.1016/B978-0-12-821815-0.00002-2

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., & Girshick, R. (2023). Segment Anything. http://arxiv.org/abs/2304.02643

Lindeberg, T. (2024). Discrete Approximations of Gaussian Smoothing and Gaussian Derivatives. Journal of Mathematical Imaging and Vision, 66(5), 759–800. https://doi.org/10.1007/s10851-024-01196-9

Liu, C. (2024). A Review of Digital Image Processing Techniques and Future Prospects. International Journal of Computer Science and Information Technology, 4(3), 223–233. https://doi.org/10.62051/ijcsit.v4n3.22

Martynova, A., Kuznetsov, M., Porvatov, V., Tishin, V., Kuznetsov, A., Semenova, N., & Kuznetsova, K. (2024). Revising deep learning methods in parking lot occupancy detection. http://arxiv.org/abs/2306.04288

Nguyen Cong, Q. H., Nguyen, K. S., Nguyen, M. D., Luu, H. N., Nguyen, H. T., Quach, C. H., & Le, V. H. (2025). Smoothing filter impact on the classification accuracy of EEG signals. Results in Engineering, 27. https://doi.org/10.1016/j.rineng.2025.106243

Noura Alshareef, Asma Abdaljalil, Reema Abobaker, Sumia Albera, & Saja Alsunosi. (2025). Deep Learning-Based Vehicle Parking Occupancy Detection. AlQalam Journal of Medical and Applied Sciences, 1190–1196. https://doi.org/10.54361/ajmas.258299

Ohtani, T., Yamamoto, R., & Ohzahata, S. (2024). IDAC: Federated Learning-Based Intrusion Detection Using Autonomously Extracted Anomalies in IoT. Sensors, 24(10). https://doi.org/10.3390/s24103218

Onyedinma Ebele G., Asogwa Doris C., & Onwumbiko Joy N. (2025). Exploring the Effectiveness of Sobel, Canny, and Prewitt Edge Detection Algorithms on Digital Images. World Journal of Advanced Engineering Technology and Sciences, 15(1), 1722–1730. https://doi.org/10.30574/wjaets.2025.15.1.0346

Ramadhan, F., Stirman, A. N. R., Dharmawan, F., & Astharini, D. (2025). Real-Time Video Filtering Using Python: A Study on Dynamic Video Enhancement Techniques. EXSACT-A, 2(1), 84. https://doi.org/10.36722/exc.v2i1.2774

Rivera-Aguilar, B. A., Cuevas, E., Luque-Chang, A., López, J., & Pérez-Cisneros, M. (2024). Pixel Interaction Model for Contrast Enhancement: Bridging Social Science and Image Processing. Applied Sciences (Switzerland), 14(23). https://doi.org/10.3390/app142310887

Said, S., & Rawahi, A. (n.d.). A Comparison of Sobel and Prewitt Edge Detection Operators. Retrieved https://doi.org/

Singh, Y., & Walingo, T. (2024). Smart Water Quality Monitoring with IoT Wireless Sensor Networks. Sensors, 24(9). https://doi.org/10.3390/s24092871

Soni, S., Singh, P., & Waoo, A. A. (2024). REVIEW OF GAMMA CORRECTION TECHNIQUES IN DIGITAL IMAGING. ShodhKosh: Journal of Visual and Performing Arts, 5(5). https://doi.org/10.29121/shodhkosh.v5.i5.2024.1902

Stojnev D., S. I. A. (2020). Preprocessing Image Data for Deep Learning. Sinteza 2020 - International Scientific Conference on Information Technology and Data Related Research, 312–317. https://doi.org/10.15308/Sinteza-2020-312-317

Szeliski, R. (2021). Computer Vision: Algorithms and Applications 2nd Edition. https://szeliski.org/Book,

Torpey, D., & Klein, R. (2024). Affine transformation estimation improves visual self-supervised learning. http://arxiv.org/abs/2402.09071

Umbaugh, S. E. (2022). Digital image processing and analysis: digital image enhancement, restoration and compression. Crc Press.

Verma, S., Singh, A., Sunil, M. P., Islam, S. M. N., Satish Kumar, D., & Ebenezar Jebarani, M. R. (2025). Deep learning enabled real time parking monitoring using YOLOv7 for intelligent and secure critical infrastructure. Discover Computing, 28(1). https://doi.org/10.1007/s10791-025-09789-7

Wang, L., Meng, Q., Wang, H., Jiang, J., Wan, X., Liu, X., Lian, X., & Cai, Z. (2023). Digital image processing realized by memristor-based technologies. In Discover Nano (Vol. 18, Number 1). Springer. https://doi.org/10.1186/s11671-023-03901-w

Wang, T. S., Kim, G. T., Kim, M., & Jang, J. (2023). Contrast Enhancement-Based Preprocessing Process to Improve Deep

Learning Object Task Performance and Results. Applied Sciences (Switzerland), 13(19). https://doi.org/10.3390/app131910760

Wei, W., Chen, H., Gong, J., Che, K., Ren, W., & Zhang, B. (2025). Real-Time Parking Space Detection Based on Deep Learning and Panoramic Images. Sensors, 25(20). https://doi.org/10.3390/s25206449

Yuldashev, Y., Mukhiddinov, M., Abdusalomov, A. B., Nasimov, R., & Cho, J. (2023). Parking Lot Occupancy Detection with Improved MobileNetV3. Sensors, 23(17). https://doi.org/10.3390/s23177642

Zhang, N., Han, X., Liu, C., Gang, R., Ma, S., & Cao, Y. (2024). Joint Luminance Adjustment and Color Correction for Low-Light Image Enhancement Network. Applied Sciences (Switzerland), 14(14). https://doi.org/10.3390/app14146320

Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57(4). https://doi.org/10.1007/s10462-024-10721-6

Visión artificial: Hacia el estacionamiento inteligente, se publicó en el mes de noviembre de 2026.

ISBN: 978-9907-802-80-1

Grupo Editorial BLR
Ecuador
Cel: +593 98 320 4362
https://grupoblr.com/
publicaciones@grupoblr.com

BIOGRAFÍA DEL AUTOR

Jonathan Víctor Aguilar Alvarado:
Doctorando en Tecnologías de la Información y las Comunicaciones. Subdecano de la Facultad de Ciencias Químicas y de la Salud de la Universidad Técnica de Machala. Investigador en inteligencia artificial y visión por computador para la resolución de problemas reales en movilidad, entornos inteligentes y sistemas académicos, complementada con experiencia en desarrollo de software y en la integración de tecnologías emergentes en procesos de enseñanza-aprendizaje, lo que se refleja en el enfoque didáctico y aplicado de la presente obra.

VISIÓN ARTIFICIAL: HACIA EL ESTACIONAMIENTO INTELIGENTE

Estimado lector, la creciente demanda de plazas de aparcamiento en entornos urbanos y académicos exige soluciones tecnológicas eficientes para optimizar la asignación de espacios y mitigar el congestionamiento. En este marco, la visión artificial y el procesamiento de imágenes emergen como herramientas clave para el monitoreo, conteo y localización vehicular automatizada, superando las limitaciones de los sistemas tradicionales mediante modelos de aprendizaje profundo y validación experimental.

Esta obra se concibe como un texto de asignatura orientado a estudiantes y docentes de carreras afines a la informática, electrónica y tecnologías de la información. Su objetivo primordial es articular los fundamentos teóricos de la visión por computadora con la implementación práctica de sistemas de control de estacionamientos, promoviendo el desarrollo de competencias aplicadas para resolver problemas contemporáneos de movilidad.

A lo largo de sus capítulos, el libro abarca desde los conceptos basales de procesamiento de imágenes y algoritmos de detección hasta el diseño, configuración de infraestructura y evaluación de proyectos piloto en escenarios reales. Con un enfoque didáctico y riguroso, la publicación proporciona a la comunidad académica una base metodológica sólida para concebir y validar sistemas inteligentes de estacionamiento, consolidando la formación de profesionales capaces de innovar en el ámbito de la gestión vehicular sostenible.

GRUPO BLR

Grupo Editorial BLR
Ecuador
Cel: +593 98 320 4362
https://grupoblr.com/
publicaciones@grupoblr.com

ISBN: 978-9907-802-80-1

9 789907 802801

UNIVERSIDAD TÉCNICA DE MACHALA
1969