

Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadoras

Licentiate Degree Program in Computer Engineering



Efecto de estimación de fondo monocular en mejoras de imagen en condiciones adversas

(Effect of monocular depth estimation on image enhancement in adverse conditions)

Informe de trabajo de graduación para optar por el título de ingeniero en Computadores con grado académico de Licenciatura

Report of Graduation Work in fulfillment of the requirements for the degree of Licentiate in Computer Engineering

Fabián Ramírez Arrieta

Cartago, junio de 2024

TEC – Escuela de Ingeniería en Computadores (CES)

Acta de Aprobación de Trabajo de Graduación

Con fundamento en lo que establece el "Reglamento de Trabajos Finales de Graduación del Instituto Tecnológico de Costa Rica", el Tribunal Examinador del Trabajo Final de Graduación, nombrado con el propósito de evaluar el proyecto:

Efecto de estimación de fondo monocular en mejoras de imagen en condiciones adversas

Habiendo analizado el resultado general del trabajo presentado por el estudiante

Primer Apellido	Segundo Apellido	Nombre	No. de carné
Ramírez	Arrieta	Fabián	2018099536

Emite el siguiente dictamen:

APROBADO 100 CALIFICACION: _____ puntos.	<input type="radio"/> REPROBADO <input type="radio"/> SE RECOMIENDA <input type="radio"/> NO SE RECOMIENDA Brindarle una nueva oportunidad para la DEFENSA PUBLICA de su Trabajo Final NUEVA FECHA: _____
--	--

Dando fe de lo aquí expuesto firmamos



MÁSTER PEDRO GUTIÉRREZ GARCÍA

Profesor Asesor



Ing. Santiago Gamboa Ramírez

Lector



Ing. Gabriel Sánchez Carvajal

Lector

Dedicatoria

A mis padres, Carolina y Juan Carlos, por ese apoyo incondicional desde que tengo uso de razón.

Agradecimientos

A mi familia, mis padres Carolina y Juan Carlos y hermano Esteban, porque gracias a su soporte e incontables esfuerzos realizados durante toda mi vida como estudiante, es posible estar escribiendo estas palabras.

A Dani, mi compañera de vida y de sueños. Gracias por estar a mi lado en cada paso. Este logro es uno más de los muchos que hemos compartido y de los que vienen.

A Yayita, cuyo interés constante en mi bienestar ha sido una fuente de motivación y a mis primos y sus parejas, por ser amigos excepcionales. Gracias por todo su afecto y ánimo.

Quiero agradecer a mis amigos y compañeros de carrera. Tomás, Wajib, Mario, Carmen, José, Alejandro, Mariana y Jessica, porque, además de ser excelentes amigos, con ustedes comprendí lo valioso que es tener un excelente equipo de trabajo.

Quiero agradecer de la manera más sincera al Instituto Tecnológico de Costa Rica, especialmente a todos aquellos profesores que han contribuido a mi formación como profesional durante todo el proceso.

Por último, agradezco a la Escuela de Ingeniería en Computadores y, especialmente, al máster Luis Alberto Chavarría por darme la oportunidad y la confianza para realizar esta investigación y el continuo apoyo durante todo el proceso. Agradezco también al máster Pedro Gutiérrez por todo el consejo brindado durante la redacción de este documento.

A todos los que han sido parte de este camino, mi más profundo agradecimiento. Sin ustedes, este logro no habría sido posible.

Resumen

Este proyecto de graduación se centra en el desarrollo de un sistema de estimación de fondo monocular en condiciones no ideales, utilizando algoritmos avanzados de procesamiento de imágenes y técnicas de inteligencia artificial. El objetivo principal es mejorar la precisión de los mapas de profundidad que se generan en entornos desafiantes, como aquellos con baja iluminación, neblina o lluvia. Para lograr esto, se implementó un diseño modular que permitió crear componentes independientes, cada uno especializado en abordar diferentes condiciones adversas, que luego se incorporaron para formar una solución integral. Además, se hizo hincapié en la cuantificación estandarizada de los resultados para evaluar objetivamente el rendimiento del sistema. Los resultados demostraron mejoras significativas en la calidad de los mapas de profundidad, validando la eficacia de los preprocesamientos inteligentes aplicados antes de ingresar los datos a la red generadora. Este trabajo no solo destaca la importancia de la calidad de los datos y el diseño modular en proyectos de IA, sino que también sugiere futuras direcciones de investigación y desarrollo para abordar nuevas condiciones adversas y optimizar continuamente la exactitud y robustez del sistema.

Palabras clave: estimación de fondo monocular, procesamiento de imágenes, inteligencia artificial, condiciones adversas, diseño modular, cuantificación de resultados.

Abstract

This graduation project focuses on the development of a monocular depth estimation system under non-ideal conditions, using advanced image processing algorithms and artificial intelligence techniques. The main objective is to improve the accuracy of depth maps generated in challenging environments, such as those with low lighting, fog, or rain. To achieve this, a modular design was implemented, allowing the creation of independent components, each specialized in addressing different adverse conditions, which were then integrated to form a comprehensive solution. Additionally, emphasis was placed on standardized quantification of results to objectively evaluate the system's performance. The results demonstrated significant improvements in the quality of depth maps, validating the effectiveness of intelligent preprocessing applied before feeding data into the generative network. This work not only highlights the importance of data quality and modular design in artificial intelligence projects but also suggests future research and development directions to address new adverse conditions and continually improve the system's accuracy and robustness.

Keywords: Monocular depth estimation, image processing, artificial intelligence, adverse conditions, modular design, results quantification.

Índice general

Índice general.....	7
Índice de tablas.....	9
Índice de figuras.....	10
Siglas y acrónimos.....	10
Capítulo 1: Introducción.....	12
1.1 Antecedentes del proyecto.....	12
1.1.1 Descripción de la entidad.....	12
1.1.2 Descripción del área de conocimiento del proyecto.....	12
1.1.3 Trabajos similares encontrados.....	13
1.2 Planteamiento del problema.....	14
1.2.1 Contexto del problema.....	14
1.2.2 Justificación del problema.....	14
1.2.3 Enunciado del problema.....	15
1.3 Objetivos del proyecto.....	15
1.3.1 Objetivo general.....	15
1.3.2 Objetivos específicos.....	15
1.4 Alcances, entregables y limitaciones del proyecto.....	16
Capítulo 2: Marco de referencia teórico.....	18
2.1 Mapa conceptual.....	18
2.2 Explicación y discusión de los principales elementos.....	18
2.2.1 Estimación de fondo monocular.....	18
2.2.2 Visión por computadora.....	19
2.2.2.1 Reconocimiento de patrones.....	19
2.2.2.2 Extracción de características.....	20
2.2.2.3 Visión estéreo.....	20
2.2.2.3 LiDAR (Light Detection and Ranging).....	20
2.2.3 Procesamiento de imágenes digitales.....	21
2.2.3.2 Filtrado de imágenes.....	21
2.2.4 Inteligencia artificial para procesamiento de imágenes.....	22
2.2.4.1 Entrenamiento.....	22
2.2.4.3 Aprendizaje profundo.....	23
2.2.4.2 Funciones de activación.....	23
2.2.4.4 Redes neuronales convolucionales (CNN).....	23
2.2.4.5 Redes U-Net.....	24
2.2.4.6 Redes generativas antagónicas.....	24
Capítulo 3: Marco metodológico.....	26
3.1 Explicación y justificación de la estrategia elegida.....	26
3.2 Diagrama de la secuencia del proceso seguido.....	27
3.3 Tabla de desglose del marco metodológico.....	28
Capítulo 4: Descripción del trabajo realizado.....	29
4.1 Descripción del proceso de solución.....	29
4.1.1 Procesado inicial de las grabaciones.....	29
4.1.1.1 Renombramiento de los videos.....	29

4.1.1.2 Extracción de fotogramas.....	30
4.1.1.3 Etiquetado de las imágenes.....	30
4.1.2 Modelos de procesado.....	31
4.1.2.1 Estructura común de los modelos.....	32
4.1.2.2 Tagger: modelo etiquetador de imágenes.....	32
4.1.2.2.1 Arquitectura del modelo.....	33
4.1.2.2.2 Data loader del modelo.....	35
4.1.2.2.3 Entrenamiento del modelo.....	36
4.1.2.3 Algoritmo para el mejoramiento de imágenes nocturnas.....	37
4.1.2.4 Dehazing: modelo eliminador de neblina.....	39
4.1.2.4.1 Datos que se utilizan para el entrenamiento.....	39
4.1.2.4.2 Arquitectura del modelo.....	41
4.1.2.4.3 Data loader del modelo.....	45
4.1.2.4.4 Entrenamiento del modelo.....	46
4.1.2.5 Deraining: modelo eliminador de lluvia.....	47
4.1.2.5.1 Datos que se utilizan para el entrenamiento.....	47
4.1.2.5.2 Arquitectura del modelo.....	48
4.1.2.5.3 Data loader del modelo.....	50
4.1.2.5.4 Entrenamiento del modelo.....	51
4.1.2.6 Interfaz gráfica.....	51
4.2 Análisis de los resultados.....	52
4.2.1 Tagger: modelo etiquetador de imágenes.....	54
4.2.2 Algoritmo para el mejoramiento de imágenes nocturnas.....	56
4.2.3 Dehazing: modelo eliminador de neblina.....	58
4.2.4 Deraining: modelo eliminador de lluvia.....	60
4.2.5 Interfaz gráfica.....	63
Capítulo 5: Conclusiones.....	66
5.1 Recomendaciones.....	67
Apéndices.....	69
Apéndices 1: implicaciones ambientales y de desarrollo sostenible.....	69
Apéndices 2: algunos ejemplos de procesamiento de imágenes nocturnas.....	72
Apéndices 3: algunos ejemplos de procesamiento de imágenes con neblina.....	73
Apéndices 4: algunos ejemplos de procesamiento de imágenes con lluvia.....	74
Bibliografía.....	75

Índice de tablas

3.1 Desglose para cada objetivo específico.....	26
4.1 Cantidad total de frames por condición en la base de datos.....	29
4.2 Cantidad total de frames por condición del conjunto de pruebas.....	
50	
4.3 Pruebas realizadas al tagger.....	52
4.4 Densidad media de bordes en imágenes nocturnas.....	55
4.5 Densidad media de bordes en imágenes con neblina.....	57
4.6 Densidad media de bordes en imágenes con lluvia.....	59

Índice de figuras

1.1	Alcance del proyecto.....	15
1.2	Alcance del proyecto y bloques de procesamiento.....	15
2.1	Mapa conceptual.....	17
2.2	Ejemplo de mapas de profundidad monocular.....	18
2.3	Point <i>cloud</i> generado por un sensor LiDAR.....	19
2.4	Aplicación de filtro Gaussiano y filtro de mediana a una imagen.....	21
2.5	Composición de una red neuronal.....	22
2.6	Ejemplo de CNN utilizada para identificar animales.....	23
3.1	Cronograma del proyecto.....	25
3.2	Diagrama de secuencia de la metodología.....	25
4.1	Secciones de la entrada Dual del Tagger.....	31
4.2	Diagrama de flujo del tagger.....	32
4.3	Ejemplo de imagen nocturna.....	35
4.4	Ejemplo de par de imágenes del dataset LOL.....	35
4.5	Diagrama de flujo del algoritmo de mejoramiento de imágenes nocturnas...	35
4.6	Ejemplo de par de imágenes del dataset O-Haze.....	38
4.7	Ejemplo de par de imágenes del dataset Foggy Cityscapes.....	38
4.8	Diagrama de arquitectura de la Dehazing Net.....	40
4.9	Diagrama de arquitectura del Down Block.....	40
4.10	Diagrama de arquitectura del Up Block.....	41
4.11	Visualización de una imagen a través de la red Dehazing.....	43
4.12	Ejemplo de par de imágenes del dataset Raindrop.....	45
4.13	Diagrama de arquitectura de la Deraining Net.....	46
4.14	Diagrama de arquitectura de los bloques de atención.....	47
4.15	Visualización de una imagen a través de la red Deraining.....	48
4.16	Diseño de la interfaz gráfica de usuario en Figma.....	50
4.17	Ejemplo de algoritmo de detección de bordes.....	51
4.18	Ejemplo de imagen que no fue identificada por el tagger como neblina.....	53
4.19	Ejemplo de imagen que fue identificada por el tagger como neblina.....	53
4.20	Resultados del algoritmo de mejoramiento de imágenes nocturnas.....	54
4.21	Resultados del dehazing con neblina artificial.....	56
4.22	Resultados del dehazing con neblina real.....	56
4.23	Resultados del deraining con imagen del Dataset Raindrop.....	58
4.24	Resultados del deraining con imagen del Dataset RainDs.....	59
4.25	Interfaz gráfica de usuario desarrollada en Tkinter.....	60
4.26	Interfaz gráfica de usuario luego de procesar la imagen.....	62
5.2.1	Ejemplos del mejoramiento nocturno.	68
5.3.1	Ejemplos de procesamiento con la dehazing Net.....	69
5.4.1	Ejemplos de procesamiento con la deraining Net.....	70

Siglas y acrónimos

En esta sección se presenta un glosario con las siglas y acrónimos más relevantes que se utilizan en el informe para facilitar la comprensión del contenido:

CLAHE: ecualización del histograma limitada por contraste (por sus siglas en inglés, Contrast Limited Adaptive Histogram Equalization).

CNN: redes neuronales convolucionales (por sus siglas en inglés, Convolutional Neural Networks).

HDR: alto rango dinámico (por sus siglas en inglés, High Dynamic Range).

IA: inteligencia artificial.

ITCR: Instituto Tecnológico de Costa Rica.

LiDAR: detección y rango de luz (por sus siglas en inglés, Light Detection and Ranging).

ReLU: unidad lineal rectificada (por sus siglas en inglés, Rectified Linear Unit).

U-Net: red neuronal U-Net.

Estas siglas y acrónimos son esenciales para entender los conceptos y técnicas abordados en el proyecto, lo que facilita la lectura y el análisis del informe completo.

Capítulo 1: Introducción

1.1 Antecedentes del proyecto

El presente proyecto surge como respuesta a un problema cada vez más evidente en el campo de la visión por computadora y el procesamiento de imágenes. Con el avance tecnológico y la creciente demanda de aplicaciones inteligentes con base en la percepción visual ha surgido la necesidad de mejorar la precisión y eficacia de la estimación de fondo monocular en condiciones adversas. En un entorno donde la visibilidad puede verse comprometida por factores como la neblina, la lluvia o la iluminación deficiente, la capacidad de generar mapas de fondo precisos es esencial para garantizar la seguridad y eficiencia de sistemas autónomos, como vehículos autónomos, sistemas de vigilancia y asistentes de visión por computadora. Es en este contexto que el trabajo realizado encuentra su justificación y relevancia, al abordar un problema crítico que afecta diversos aspectos de la vida moderna y que representa un gran desafío tecnológico.

En este apartado se plantea el contexto y los antecedentes que rodean la iniciativa, detallando la entidad en la que se realizó el trabajo y una descripción de las áreas de conocimiento involucradas. Por último, un repaso de las investigaciones previas encontradas con sus respectivas soluciones.

1.1.1 Descripción de la entidad

El proyecto se desarrolla en el Instituto Tecnológico de Costa Rica (ITCR), una institución educativa de renombre en el país. La entidad es reconocida por su excelencia en la formación de profesionales altamente capacitados en diversas disciplinas tecnológicas y científicas.

La iniciativa se realiza en la Escuela de Ingeniería en Computadores. Esta escuela cuenta con profesores con amplia experiencia, quienes poseen un destacado historial de éxitos en investigaciones similares, lo que proporciona el entorno propicio y los recursos necesarios para llevar a cabo este proyecto.

1.1.2 Descripción del área de conocimiento del proyecto

La iniciativa se enmarca en tres áreas clave de conocimiento definidas por la Escuela de Ingeniería en Computadores:

- **Reconocimiento y procesamiento de imágenes:** esta área se centra en el análisis y manipulación de imágenes digitales para extraer información valiosa. Durante el desarrollo de la iniciativa, se aplican técnicas avanzadas de procesamiento de imágenes para mejorar la estimación de fondo monocular en condiciones adversas.
- **Inteligencia artificial:** la IA (inteligencia artificial) es un campo de estudio que busca desarrollar algoritmos y sistemas capaces de aprender y tomar decisiones de manera autónoma. En este trabajo, se utilizaron técnicas de IA

para diseñar y aplicar algoritmos que optimizan la estimación de fondo monocular.

- **Aplicaciones de alta complejidad algorítmica:** esta área se enfoca en el diseño y desarrollo de algoritmos complejos para resolver problemas específicos. Aquí se trabaja en integrar diversos algoritmos para abordar la reducción de neblina, eliminación de lluvia y mejorar la visión nocturna, con el fin de optimizar la estimación de fondo monocular en condiciones desafiantes.

La combinación de estas áreas de conocimiento proporciona un enfoque integral para abordar el desafío que se planteó en el proyecto, lo que permite desarrollar soluciones innovadoras y efectivas para mejorar la estimación de fondo monocular en condiciones adversas. Con este sólido fundamento interdisciplinario se espera que el proyecto pueda ofrecer contribuciones significativas, tanto en el ámbito académico como en el desarrollo tecnológico, abriendo nuevas oportunidades para la investigación y la aplicación práctica de la visión por computadora en entornos desafiantes.

1.1.3 Trabajos similares encontrados

En el estado del arte de la *estimación de fondo monocular en entornos desafiantes* se encontraron varios trabajos publicados por distintos investigadores. Estos estudios suelen centrarse en una condición específica mientras que este trabajo se propone como una solución adaptable y flexible ante diversas situaciones del entorno, como variables climáticas, diferentes niveles de iluminación, entre otros factores

Por ejemplo, con respecto a la mejora de imágenes nocturnas, Yu *et al.* [4] proponen un método basado en aprendizaje profundo para mejorar imágenes capturadas en condiciones de baja luz. En este caso se aborda simultáneamente el aumento del contraste y la reducción del ruido sin causar desenfoque.

En el caso de eliminación de lluvia en distintas imágenes, Qian *et al.* [5] abordan el problema de eliminar visualmente gotas de lluvia adheridas a una ventana de vidrio o lente de cámara, transformando una imagen degradada por gotas de lluvia en una imagen limpia. Esto lo logran al utilizar una red generativa antagónica. Refiérase al apartado 2.2.4.6 para más información sobre este tipo de redes.

Otro trabajo enfocado en el procesamiento de imágenes es el que plantearon Bijelic *et al.* [6], que presenta un enfoque novedoso para la detección de objetos en vehículos autónomos en condiciones climáticas adversas, donde las fuentes de datos habituales como cámaras, LiDAR y radares pueden fallar debido a distorsiones asimétricas. Los autores abordan este desafío mediante la introducción de un conjunto de datos multimodal adquirido durante más de 10,000 km de conducción en el norte de Europa, único por su inclusión de condiciones climáticas severas.

1.2 Planteamiento del problema

En este apartado se explora la complejidad del problema en cuestión y se analiza la relevancia de buscar una solución efectiva. Además, se destaca la importancia de comprender en profundidad los desafíos asociados con la estimación de fondo monocular en las condiciones adversas mencionadas. Esto sirve como fundamento para presentar el enunciado del problema y la discusión sobre cómo abordarlo de manera eficaz.

1.2.1 Contexto del problema

La generación de fondo monocular es un área crucial en visión por computadora que busca estimar la profundidad y la estructura tridimensional de una escena a partir de una sola imagen. Sin embargo, este proceso se ve desafiado por condiciones adversas como la presencia de neblina, lluvia o iluminación deficiente, que pueden distorsionar la calidad de las imágenes y dificultar la generación precisa de mapas de fondo. En este contexto, el presente proyecto surge como una respuesta a la necesidad de mejorar la estimación de fondo monocular en condiciones no ideales.

Este proyecto se enmarca en un contexto más amplio de avances tecnológicos en visión por computadora, donde el desarrollo de algoritmos y técnicas de mejora de imagen ha ganado relevancia en la búsqueda de soluciones efectivas para problemas prácticos. La capacidad de mejorar la percepción visual en condiciones adversas no solo tiene aplicaciones en campos como la seguridad vial y la monitorización ambiental, sino que también abre nuevas oportunidades para la investigación y el desarrollo tecnológico.

1.2.2 Justificación del problema

La necesidad de mejorar la estimación de fondo monocular en condiciones adversas es evidente en diversos contextos, donde la visibilidad reducida puede comprometer la seguridad y la eficiencia de las aplicaciones con base en visión por computadora. La presencia de neblina, lluvia o iluminación deficiente puede distorsionar significativamente la calidad de las imágenes, lo que a la vez afecta generar mapas de fondo precisos.

La importancia de abordar este problema radica en su relevancia para múltiples aplicaciones prácticas, incluidas la seguridad vial, la monitorización ambiental y la vigilancia de seguridad. En entornos donde la claridad visual es crucial, como carreteras, aeropuertos o zonas urbanas, la capacidad de obtener mapas de fondo precisos en condiciones adversas puede marcar la diferencia entre la seguridad y el riesgo.

Además, desde una perspectiva tecnológica, resolver este problema representa un avance significativo en el campo de la visión por computadora y el procesamiento de imágenes. El perfeccionamiento de los algoritmos de mejora de imagen y la integración sinérgica con técnicas de estimación de fondo monocular

puede abrir nuevas posibilidades en la investigación y desarrollo de sistemas inteligentes capaces de operar en entornos desafiantes.

Por lo tanto, la justificación de este trabajo se basa en las amplias repercusiones que una estimación precisa de fondo monocular tiene en aspectos críticos como la seguridad, la eficiencia operativa y el avance de las tecnologías emergentes. En el ámbito de la seguridad, por ejemplo, una mejora en la claridad visual durante condiciones climáticas adversas puede ser determinante para la navegación segura de vehículos autónomos, lo que reduce el riesgo de accidentes.

En términos de eficacia, sistemas de vigilancia y monitoreo ambiental que dependen de una visión clara pueden operar con mayor precisión, lo que evita falsos positivos y optimiza el uso de recursos. Debido a lo anterior, este proyecto no solo busca solucionar una necesidad técnica, sino que también tiene el potencial de inspirar aplicaciones innovadoras y robustas en campos que van desde la robótica autónoma hasta los sistemas de seguridad inteligente.

1.2.3 Enunciado del problema

¿Cómo se optimiza la calidad de las entradas de un generador de mapas de profundidad existente para mejorar la estimación de fondo monocular en condiciones adversas, como neblina, lluvia y baja iluminación?

1.3 Objetivos del proyecto

1.3.1 Objetivo general

- Desarrollar un sistema de estimación de fondo monocular en condiciones no ideales, mediante el procesamiento previo de las imágenes de entrada, aumentando la precisión del proceso en entornos desafiantes.

1.3.2 Objetivos específicos

1. Diseñar algoritmos para la mejora de la calidad de las imágenes con condiciones adversas, mediante la identificación de dichas condiciones, optimizando la claridad de los elementos presentes en las imágenes.
2. Implementar los algoritmos de mejora de imagen en el sistema de estimación de fondo monocular ya existente, mediante la integración modular de cada algoritmo, garantizando la precisión de los mapas generados en condiciones adversas.
3. Validar el desempeño y la eficacia del sistema desarrollado mediante un análisis de resultados en escenarios, tanto en condiciones óptimas como en condiciones adversas, garantizando la función y utilidad en entornos diversos y desafiantes.

1.4 Alcances, entregables y limitaciones del proyecto

Este proyecto se basa en cómo procesar los distintos escenarios que se puede encontrar un vehículo en carretera y, de esta manera, obtener un mejor resultado en el momento de calcular su estimación de fondo monocular. Para entender mejor el alcance del proyecto ver la [Figura 1.1](#).

Es importante destacar que el proyecto no desarrolló un sistema propio para la estimación directa de fondo monocular; en cambio, se basó en el algoritmo AdaBins, desarrollado por Farooq Bhat *et al.* [7]. El enfoque principal estuvo en la creación de algoritmos de preprocesamiento que mejoran la calidad de las entradas para este generador de mapas de fondo monocular existente.

AdaBins es un algoritmo para la estimación de profundidad monocular que se utilizó como una "caja negra" en el presente proyecto. Fue entrenado principalmente en ambientes de conducción con condiciones mayormente soleadas y con elementos claramente visibles, debido al uso del Dataset Kitti [40] como datos de entrenamiento. Garantiza su desempeño en escenarios con buena iluminación y visibilidad. Sin embargo, cuando las imágenes de entrada presentan calidad deteriorada debido a condiciones adversas como lluvia, neblina o falta de luz, el algoritmo tiende a mostrar un rendimiento inferior.

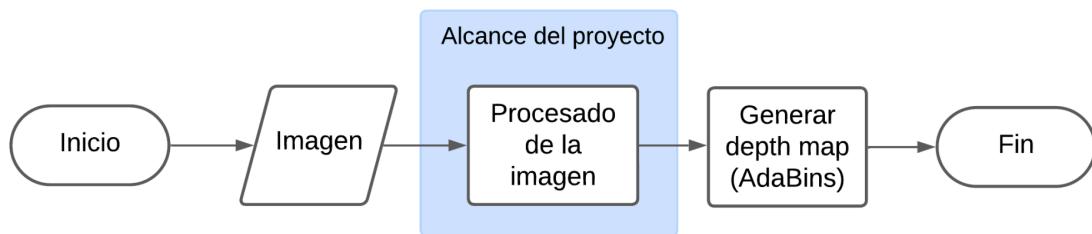


Figura 1.1: Alcance del proyecto

Con respecto a los entregables del proyecto, se sigue una metodología modular para desarrollar algoritmos independientes especializados en una tarea específica para luego utilizarlos en una herramienta central que une todos los módulos de una manera sinérgica. En la [Figura 1.2](#) se muestra una continuación de la [Figura 1.1](#), pero con mayor detalle acerca del procesado de la imagen. Dicha figura es de gran utilidad para comprender cada uno de los entregables.

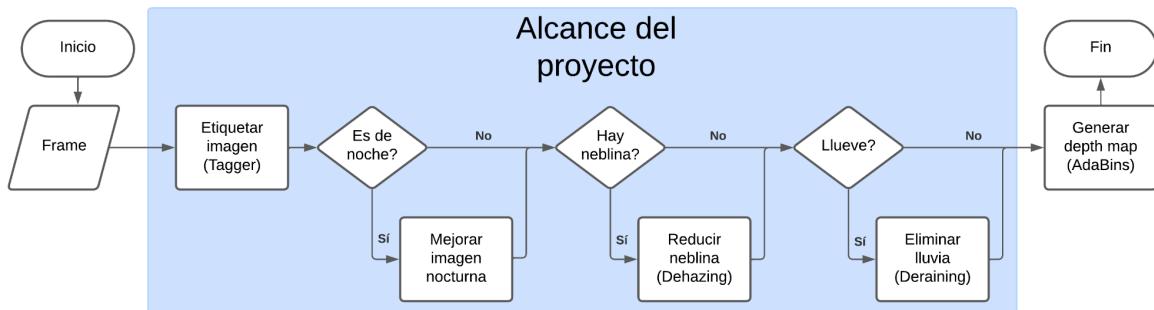


Figura 1.2: Alcance del proyecto y bloques de procesamiento

Por lo tanto, los entregables se pueden desglosar de la siguiente manera:

1. Código fuente del algoritmo módulo etiquetador (*tagger*), encargado de identificar cuáles situaciones se presentan en la imagen de entrada.
2. Código fuente del algoritmo reductor de lluvia (*deraining*), encargado de identificar y reducir la lluvia de una imagen en caso de que sea necesario.
3. Código fuente del algoritmo reductor de neblina (*dehazing*), encargado de identificar y reducir la neblina de una imagen en caso de que sea necesario.
4. Código fuente del algoritmo mejorador de condiciones luminosas (*light enhancer*), encargado de mejorar las condiciones lumínicas de una imagen según sea necesario.
5. Código fuente de la herramienta centralizada que implementa los algoritmos anteriores de manera sinérgica para una interacción amigable con la persona usuaria.
6. Resultados de pruebas extensivas realizadas al sistema para evidenciar el funcionamiento de la solución que se planteó.
7. Informe final, en el que se evidencian los resultados al finalizar el proyecto.

Este trabajo presenta la limitación de requerir un sistema con tarjeta gráfica Nvidia para su ejecución, ya que se desarrolló utilizando PyTorch, una biblioteca de aprendizaje profundo [9], que aprovecha la tecnología CUDA de Nvidia para realizar cálculos en la GPU de manera eficiente [8]. Aunque el sistema operativo no es tan restrictivo, la dependencia de PyTorch y CUDA limita la ejecución del proyecto a sistemas compatibles con *hardware* Nvidia para garantizar un rendimiento óptimo.

Capítulo 2: Marco de referencia teórico

En este capítulo se revisan los conceptos fundamentales para la comprensión del proyecto. Comienza con la sección 2.1, que presenta un mapa conceptual para una visión integral de los temas abordados. Luego, la sección 2.2 profundiza en cada concepto relacionado con la estimación de fondo monocular, explorándolos con detalle y elevando gradualmente el nivel de profundidad a lo largo del capítulo.

2.1 Mapa conceptual

La Figura 2.1 muestra un mapa conceptual que ilustra los conceptos fundamentales necesarios para facilitar la comprensión de las secciones subsiguientes del proyecto. Las explicaciones detalladas de estos conceptos se encuentran en la sección 2.2.

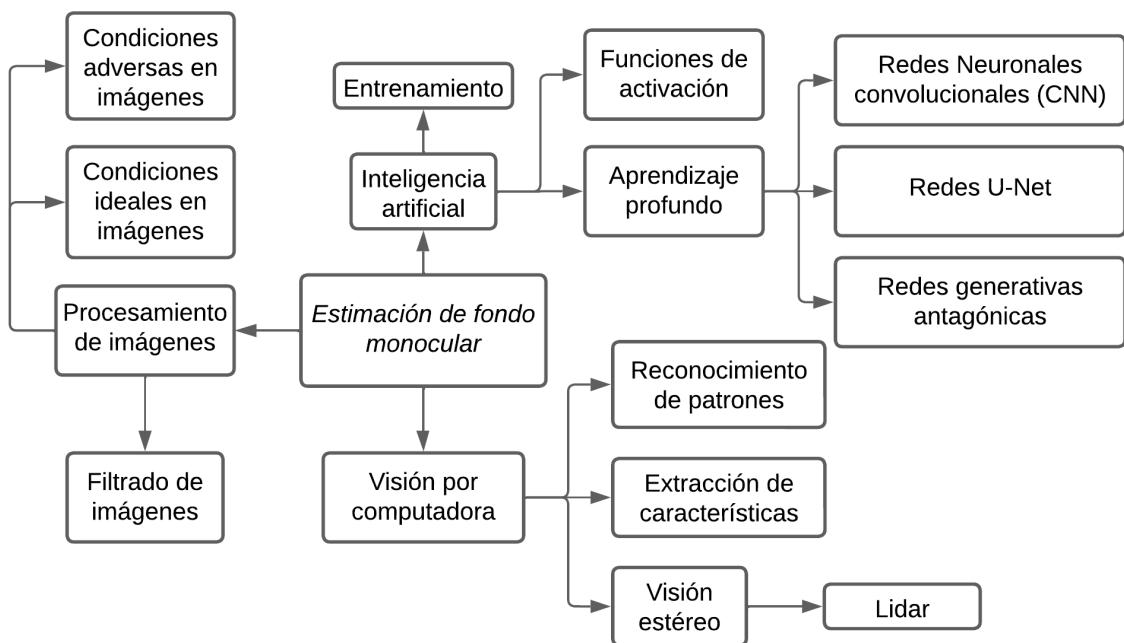


Figura 2.1: Mapa conceptual

2.2 Explicación y discusión de los principales elementos

2.2.1 Estimación de fondo monocular

Como explica Szeliski [1] la estimación de profundidad monocular es el proceso de generar mapas de profundidad a partir de imágenes individuales, lo que permite aplicaciones creativas como visualización en 3D, efectos de enfoque suave y ayuda en la comprensión de escenas. Esta tecnología también es útil en robótica, como en la navegación autónoma, ya que les permite a los sistemas comprender a qué distancia se encuentran los objetos a su alrededor con respecto a su propio

punto de vista. Aunque tradicionalmente se usaban múltiples cámaras o sensores de distancia, investigaciones recientes han avanzado en el uso de diversas técnicas utilizando inteligencia artificial para inferir la profundidad con base en una sola imagen.

Un ejemplo de un mapa de profundidad se puede observar en la Figura 2.2. Dicha figura presenta los resultados de la solución que se planteó para generar mapas de profundidad monocular por Casser *et al.* [3] utilizando datos del *dataset* CityScapes [2].

2.2.2 Visión por computadora

Como explica Szeliski en su libro, *Computer Vision: Algorithms and Applications* [10], la visión por computadora es un campo de la ciencia y la ingeniería que se centra en la interpretación automática de imágenes y vídeos. Además, busca replicar la complejidad de la percepción visual humana, lo que permite que las máquinas detecten y comprendan el contenido visual para realizar tareas como el reconocimiento de objetos, la navegación y la interacción con el entorno.

La visión por computadora tiene aplicaciones prácticas en múltiples sectores, desde la automatización industrial hasta la asistencia médica y la interacción del consumidor. Asimismo, incluye tecnologías como el reconocimiento óptico de caracteres, la inspección de máquinas, la logística de almacenes, el diagnóstico por imágenes médicas, los vehículos autónomos y la creación de modelos 3D, así como aplicaciones más cotidianas como la autenticación visual y la detección de rostros.

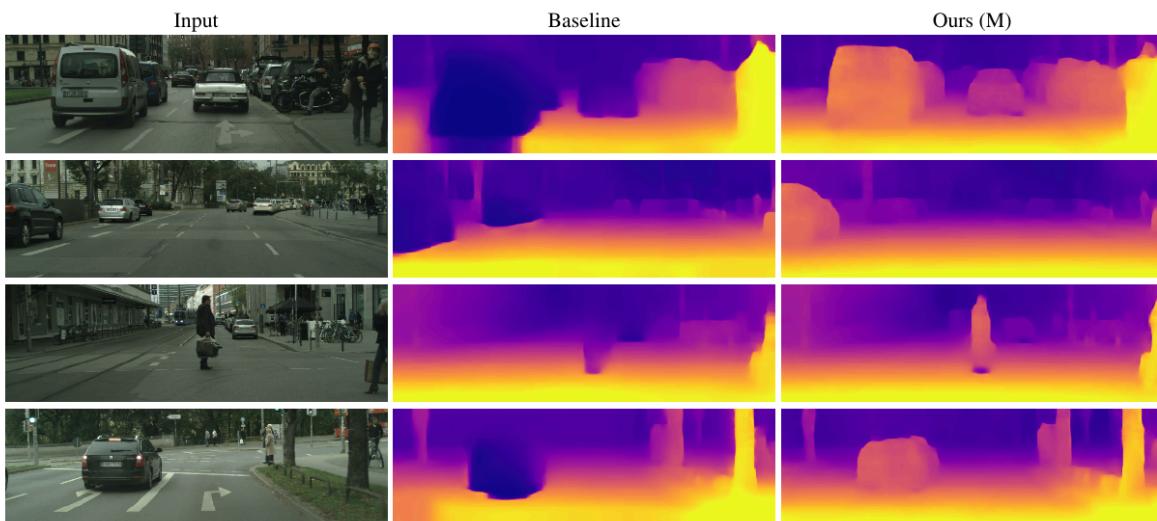


Figura 2.2: Algunos ejemplos de mapas de profundidad monocular [3]

2.2.2.1 Reconocimiento de patrones

Szeliski explica este concepto como el proceso mediante el cual los sistemas informáticos identifican patrones repetitivos o regularidades en los datos visuales, como imágenes o secuencias de vídeo. Esto incluye la identificación de formas, texturas, colores o cualquier otro atributo visual que puede usarse para diferenciar

un objeto o característica de otro [11]. En el caso de este proyecto, dicho proceso es importante para identificar las distintas condiciones adversas y, de esta manera, tomar las decisiones que se plasman en la Figura 1.2.

2.2.2.2 Extracción de características

Como explican Kreowsky y Stabernack [12], la extracción de características en visión por computadora se refiere al proceso de identificar atributos únicos o informativos de las imágenes, que son esenciales para realizar tareas como clasificación, reconocimiento y seguimiento de objetos. Estas particularidades, que pueden ser bordes, esquinas, texturas o contornos, permiten que un sistema informático interprete y comprenda la información visual, lo que reduce la complejidad de los datos y se enfoca en elementos relevantes para el análisis posterior.

2.2.2.3 Visión estéreo

Según Georgoulas *et al.* [13] la visión estéreo en visión por computadora es una técnica para estimar la profundidad de una escena a partir de dos o más imágenes tomadas desde diferentes puntos de vista, simulando la percepción de profundidad del ojo humano. Al comparar las diferencias entre las imágenes (disparidad), se puede calcular la distancia de los objetos en la escena.

En la actualidad, se busca implementar la visión monocular en visión por computadora. Esta se refiere a la estimación de profundidad utilizando una sola imagen, en contraste con la técnica estéreo que utiliza múltiples vistas. Aunque es más desafiante debido a la falta de información de paralaje, existen técnicas avanzadas que permiten su desarrollo.

2.2.2.3 LiDAR (Light Detection and Ranging)

LiDAR es una metodología avanzada de sensores que emplea la luz láser para obtener mediciones detalladas de distancias y movimientos en un espacio, lo que facilita la creación de representaciones topográficas y modelos tridimensionales de alta precisión. Esta tecnología es esencial para el desarrollo de sistemas de conducción autónoma [14].

Un sensor LiDAR genera un mapa de puntos tridimensionales, conocido como *point cloud*, que es una colección de puntos de datos en el espacio que representa una medida tridimensional del entorno. Cada punto del conjunto contiene información de su ubicación en tres dimensiones y es el resultado de los pulsos láser emitidos por un sistema LiDAR que rebotan en objetos y superficies, lo que permite modelar con precisión la forma y el tamaño de características físicas en espacios, tanto naturales como construidos. En la Figura 2.3 se muestra un ejemplo de un *point cloud*.

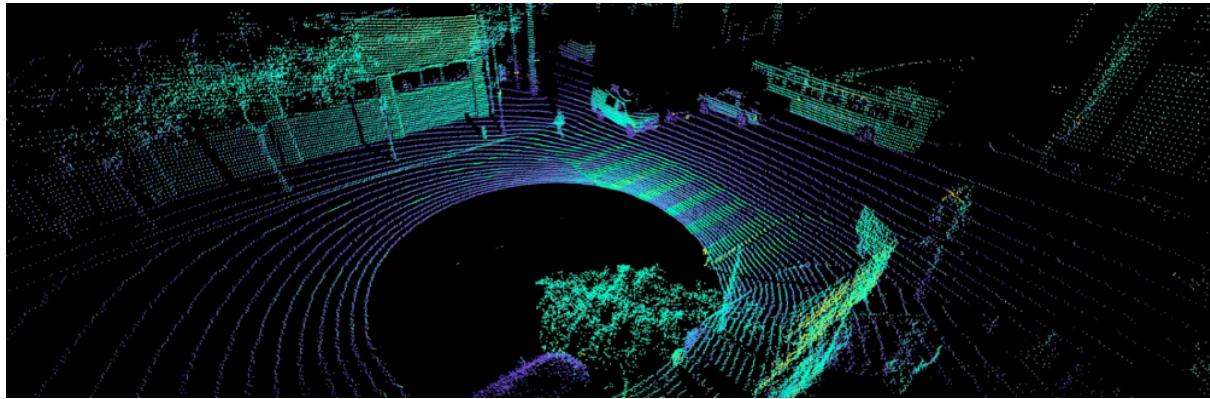


Figura 2.3: Point *cloud* generado por un sensor LiDAR [15]

2.2.3 Procesamiento de imágenes digitales

Como indican González y Woods [16] el procesamiento de imágenes digitales es el acto de manipular datos visuales mediante operaciones computacionales. Cada imagen se trata como una matriz de píxeles, donde cada píxel contiene información de intensidad o color. Esta manipulación puede implicar diversas tareas como la mejora del contraste, la reducción de ruido, la segmentación y el reconocimiento de patrones. El propósito fundamental del procesamiento de imágenes es mejorar la interpretación de los datos por parte de los usuarios o para hacerla más apta para un análisis automatizado.

En el ámbito de la visión por computadora y la inteligencia artificial, el procesamiento de imágenes sirve como la base para la extracción y el análisis de características, lo que permite a los sistemas aprender de los datos visuales y realizar inferencias inteligentes. Este proceso comprende, desde el preprocessamiento básico hasta tareas de alto nivel como la clasificación y el examen de atributos, lo que a menudo ocasiona la *comprensión* computacional de un conjunto de objetos en una imagen.

2.2.3.1 Condiciones ideales de una imagen

Szeliski [10] explica cómo las condiciones ideales para el procesamiento de imágenes son aquellas donde la imagen capturada contiene una iluminación uniforme, alto contraste y mínima interferencia de ruido o distorsiones atmosféricas como neblina o lluvia. Esto permite que los algoritmos de procesamiento de imágenes operen con mayor eficiencia y precisión, lo que facilita tareas como la segmentación, detección de bordes y extracción de características. La ausencia de condiciones adversas reduce la necesidad de preprocessamiento extenso y mejora la fiabilidad de las técnicas de visión por computadora aplicadas posteriormente, desde la estimación de fondo hasta el reconocimiento de patrones avanzados.

2.2.3.2 Filtrado de imágenes

Como menciona González y Woods [16], el filtrado de imágenes es un paso crítico en el procesamiento de estas que implica la aplicación de algoritmos específicos para alterar o mejorar una imagen. Este proceso puede tener varios

objetivos, como suavizar la imagen, eliminar ruido, enfatizar bordes o corregir la iluminación. Los filtros pueden ser lineales, como el filtro Gaussiano para el desenfoque o no lineales, como el filtro de mediana que se utiliza para reducir el ruido de sal y pimienta. En la [Figura 2.4](#) hay un ejemplo de la aplicación de un filtro Gaussiano y un filtro de mediana a una imagen de entrada.

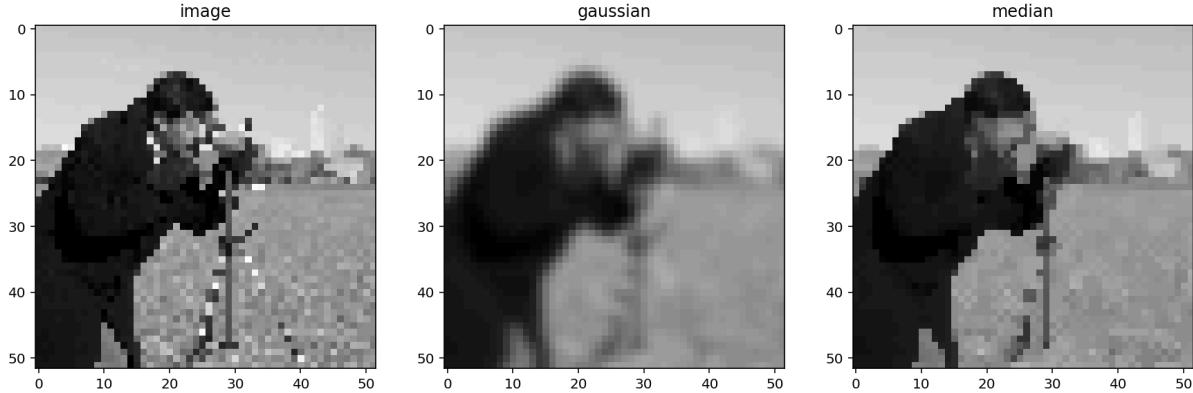


Figura 2.4: Aplicación de filtro Gaussiano y filtro de mediana a una imagen [17]

2.2.3.3 Condiciones adversas en imágenes

Szeliski [\[10\]](#) expone cómo existen condiciones adversas para el procesamiento de imágenes que incluyen factores ambientales como iluminación insuficiente, lluvia, neblina y otros fenómenos atmosféricos que pueden degradar la calidad de la imagen. Estos factores introducen ruido, disminuyen el contraste, alteran los colores y tienen la posibilidad de oscurecer detalles importantes, lo que dificulta la aplicación efectiva de algoritmos de visión por computadora.

2.2.4 Inteligencia artificial para procesamiento de imágenes

La inteligencia artificial facilita el procesamiento de imágenes mediante algoritmos que permiten a las computadoras identificar patrones y características visuales complejas. Especialmente, con el uso de redes neuronales convolucionales, la IA puede clasificar imágenes, reconocer rostros y objetos e incluso diagnosticar enfermedades a partir de imágenes médicas. Su eficacia radica en la habilidad de aprender y mejorar con base en grandes volúmenes de datos visuales [\[18\]](#).

2.2.4.1 Entrenamiento

En la inteligencia artificial, el entrenamiento supervisado para el procesamiento de imágenes implica alimentar a la máquina con grandes cantidades de datos etiquetados, donde cada imagen tiene una anotación que describe su contenido. Esto permite que los modelos aprendan a reconocer patrones y tomen decisiones con base en las entradas y salidas conocidas. Aunque también existe el entrenamiento no supervisado, que detecta patrones sin etiquetas previas, el enfoque supervisado es más común para tareas específicas como la identificación de objetos, la clasificación o procesamiento de imágenes.

2.2.4.3 Aprendizaje profundo

El *deep learning* o aprendizaje profundo es una rama de la inteligencia artificial que utiliza redes neuronales profundas. Estas redes están compuestas de capas de unidades básicas conocidas como neuronas. Cada neurona recibe una serie de entradas, las combina usando pesos y un término de sesgo y luego aplica una función de activación para generar una salida. Estas salidas se convierten en las entradas de la siguiente capa de la red. La profundidad de la red proviene de la presencia de múltiples capas ocultas entre la entrada y la salida, lo que le permite a la red aprender características a diferentes niveles de abstracción. Las neuronas en una red trabajan juntas para realizar tareas complejas como reconocimiento de imágenes, traducción de lenguaje y muchas otras aplicaciones de IA.

En la [Figura 2.5](#), Larrañaga y Moujahid [20] muestran cómo se compone una red neuronal, desde su estructura más básica, la neurona, hasta el sistema neuronal completo.

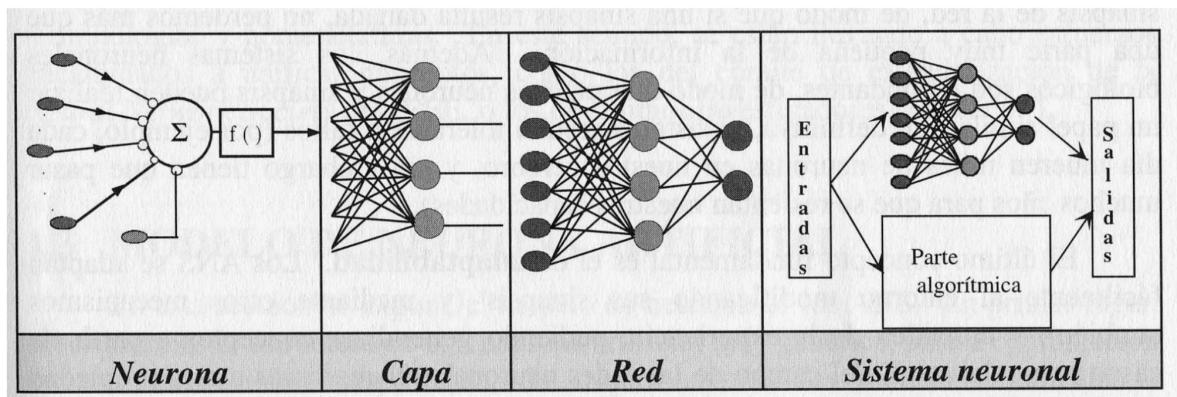


Figura 2.5: Composición de una red neuronal [20]

2.2.4.2 Funciones de activación

Szeliski explica [19] cómo las funciones de activación en redes neuronales son cruciales, ya que determinan si una neurona se activará y de qué forma influye en la red. La función de activación más común es la unidad lineal rectificada (ReLU), que da una salida a cero para cualquier entrada negativa y una salida lineal para las positivas.

2.2.4.4 Redes neuronales convolucionales (CNN)

Según LeCun *et al.* [21] las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo especializado de red neuronal profunda diseñada para procesar datos que tienen una estructura en forma de rejilla, como las imágenes. Una CNN típicamente consta de una secuencia de capas que incluyen capas convolucionales, las cuales aplican una serie de filtros para extraer características importantes de la imagen; capas de *pooling*, que reducen la dimensionalidad y capturan la invarianza espacial y capas completamente conectadas, que interpretan las particularidades extraídas para realizar tareas como clasificación o detección.

Estas redes son capaces de capturar de manera automática jerarquías de particularidades visuales, desde bordes y texturas en las primeras capas hasta aspectos complejos en capas más profundas, lo que las hace muy eficaces para el reconocimiento visual en una gran variedad de aplicaciones.

En la Figura 2.6 Biswal [22] presenta un ejemplo de una CNN que recibe como entrada la imagen de un animal. Esta imagen pasa a través de las distintas capas convolucionales, sus respectivas funciones de activación y las capas de *pooling*, para, por último, atravesar la capa completamente conectada y mediante dos neuronas de salida, define si el animal en la imagen de entrada es un pájaro o un perro.

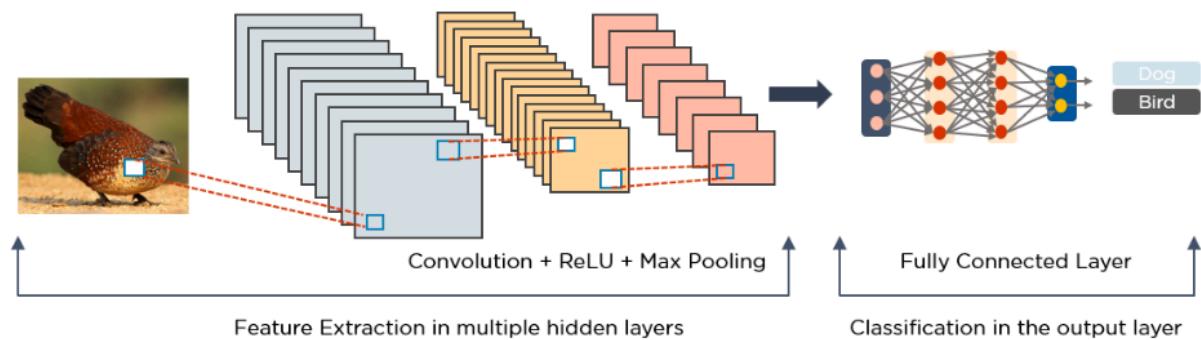


Figura 2.6: Ejemplo de CNN que se utiliza para identificar animales [22]

2.2.4.5 Redes U-Net

Como explica Ronneberger [23] las redes U-Net son una arquitectura de red neuronal que destaca por su capacidad de segmentación precisa en imágenes. Se caracterizan por tener una parte de codificación, que comprime la información visual a través de capas convolucionales y de *pooling*, lo que reduce su dimensionalidad y una parte de decodificación, que reconstruye la segmentación deseada a partir de esta representación comprimida. Los *canales* o mapas de características aumentan a medida que la imagen se reduce durante la codificación, capturando detalles complejos y luego disminuyen en el camino de decodificación, donde la red aprende a localizar y delinear con precisión las regiones de interés dentro de la imagen. Esta dualidad permite que la U-Net no solo reconozca patrones, sino que también mantenga la información espacial necesaria para la segmentación detallada.

2.2.4.6 Redes generativas antagónicas

Goodfellow *et al.* [24] explican cómo las redes generativas antagónicas (GAN, por sus siglas en inglés) son un enfoque en el campo del aprendizaje profundo donde dos redes neuronales, el generador y el discriminador, se entrena simísticamente en un juego adversario. El generador intenta crear datos que sean indistinguibles de los reales, mientras que el discriminador busca diferenciar entre los datos reales y los que se generan. A través de este entrenamiento competitivo, las GAN aprenden a producir resultados altamente realistas y detallados, siendo capaces de generar imágenes nuevas que pueden ser difíciles de distinguir de las

imágenes auténticas. Este enfoque ha encontrado aplicaciones en mejoramiento de imágenes, arte, síntesis de fotografías y muchos otros campos que requieren generación de contenido visual convincente.

Capítulo 3: Marco metodológico

En este capítulo se desglosa la metodología híbrida que dirigió el desarrollo del proyecto. Para esto se integra la flexibilidad de Scrum con la secuencia estructurada de la gestión tradicional de los proyectos y se explica cómo esta combinación promovió un equilibrio entre agilidad y previsibilidad en todas las etapas del trabajo.

Además, se presenta el cronograma que se utiliza durante el proyecto, detallando la secuencia y duración de las tareas, mientras se ilustra el ciclo iterativo de las iteraciones semanales con un diagrama de flujo. Este enfoque permitió ajustes oportunos y aseguró que el proyecto avanzara de acuerdo con los objetivos establecidos.

Por último, se proporciona una tabla detallada que mapea las actividades clave, los entregables y las herramientas empleadas. Esto evidencia la estructura metódica y la ejecución estratégica detrás del éxito del proyecto.

3.1 Explicación y justificación de la estrategia elegida

En este proyecto se implementó una metodología híbrida que fusiona la agilidad del marco de trabajo Scrum con la rigurosidad de los enfoques de gestión de proyectos tradicionales y algunas características del modelo de desarrollo en espiral. Scrum es un marco de trabajo ágil que promueve el desarrollo iterativo e incremental, al organizar el trabajo en ciclos repetitivos conocidos como *sprints*. Estos *sprints* se centran en la colaboración y retroalimentación constante para adaptarse rápidamente a los cambios, lo que resulta ideal para proyectos de investigación y desarrollo donde los requisitos pueden evolucionar con el tiempo [25].

Por otro lado, como explica Laoyan [26], la gestión de proyectos tradicional sigue un enfoque más lineal y secuencial, conocido comúnmente como modelo de cascada. Este enfoque se caracteriza por etapas bien definidas y una planificación detallada, lo que proporciona una estructura predecible y ordenada. Las tareas se completan una tras otra, con dependencia de que cada etapa sea finalizada antes de pasar a la siguiente.

El modelo de desarrollo en espiral, desarrollado por Barry Boehm [41], combina elementos de ambos enfoques mencionados anteriormente y enfatiza la repetición iterativa de cuatro fases principales: planificación, análisis de riesgos, ingeniería y evaluación. Este modelo es particularmente útil para proyectos que requieren una evaluación continua y ajustes frecuentes, ya que permite incorporar mejoras y reducir riesgos a través de múltiples iteraciones.

La combinación de estos tres enfoques se adaptó en el proyecto para aprovechar las fortalezas de cada uno.. Las reuniones semanales proporcionaron un foro para la revisión continua y la adaptación del proyecto, principios fundamentales de Scrum, asegurando que el desarrollo pueda mantenerse ágil, receptivo a los cambios y en sincronía con la contraparte. Simultáneamente, el seguimiento de un

plan de proyecto detallado y el avance secuencial en las tareas permitieron que el equipo mantuviera un sentido claro de dirección y progreso medible. Este enfoque híbrido aseguró que, mientras se mantiene la flexibilidad para la innovación y el cambio, también se cumplan las expectativas de entrega y calidad en los plazos previstos. Además, las características del modelo en espiral, como la identificación y mitigación de riesgos en cada ciclo, se incorporaron para asegurar que cualquier problema potencial fuera abordado de manera temprana y efectiva.

Dicho plan de proyecto se puede observar en la Figura 3.1, donde se detalla el cronograma que se utiliza durante la realización del proyecto. Además, es posible ver el cronograma al ingresar al siguiente [enlace](#), el cual dirige a una hoja de cálculo en Google Sheets con la información para visualizarla de una manera más cómoda.

Instituto Tecnológico de Costa Rica

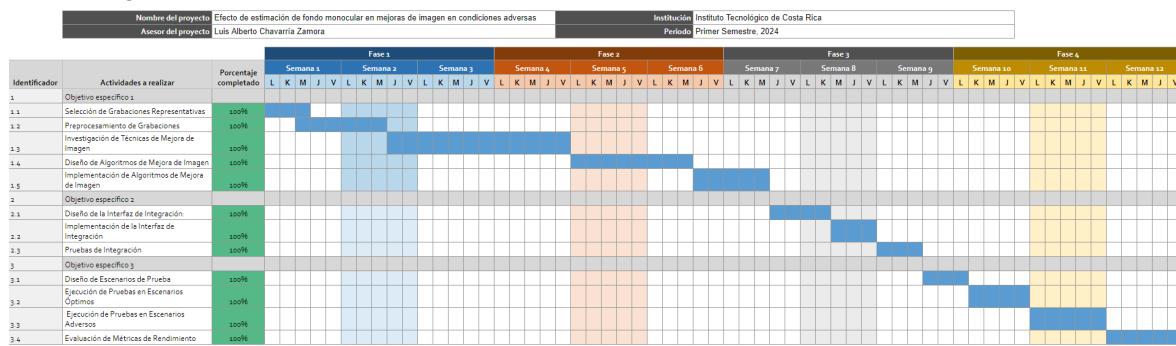


Figura 3.1: Cronograma del proyecto

3.2 Diagrama de la secuencia del proceso seguido

La metodología híbrida adoptada para este proyecto se ilustra en el diagrama de la Figura 3.2, el cual sintetiza cómo las prácticas ágiles se incorporaron en una secuencia estructurada de desarrollo. Comienza con la definición de tareas y cronograma, lo que establece una base sólida y un plan claro, las cuales se observan en el cronograma de la Figura 3.1. Siguiendo el flujo, se encuentra el desarrollo de *tareas*, donde se ejecutan las actividades planificadas. Es clave destacar que se realizaron iteraciones semanales para revisar el avance, para asegurar que el proyecto se mantenga en curso y alineado con los objetivos.

Esta revisión incluye una retroalimentación activa que puede llevar a ajustes en el plan de trabajo y una revisión del alcance para asegurar que todos los entregables cumplan con las expectativas. Finalmente, la etapa de proporcionar entregables cierra el ciclo, presentando los resultados del trabajo hecho. Este enfoque iterativo asegura adaptabilidad y mejora continua, mientras que la estructura general garantiza que el proyecto progrese de manera ordenada y predecible.

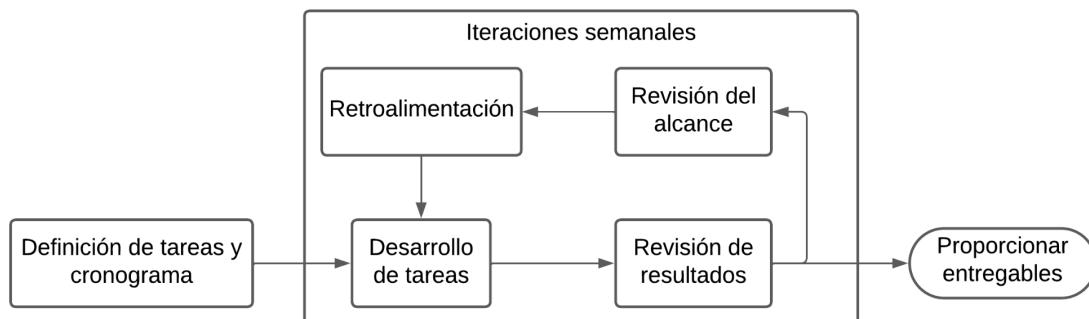


Figura 3.2: Diagrama de secuencia de la metodología

3.3 Tabla de desglose del marco metodológico

A continuación, en la Tabla 3.1, se puede observar el cuadro de desglose para cada objetivo específico. Seguidamente, se encuentran las herramientas que se utilizan para desarrollar el proyecto.

Objetivo específico	Principales actividades	Entregables asociados	Técnicas o herramientas que se utilizan.	Estrategias de verificación
1. Diseñar métodos efectivos de mejora de imagen, mediante la generación de mapas de fondo precisos, lo que optimiza la calidad de las imágenes en condiciones adversas.	1.1 Selección de grabaciones representativas. 1.2 Preprocesamiento de grabaciones. 1.3 Investigación de técnicas de mejora de Imagen. 1.4 Diseño de algoritmos de mejora de imagen. 1.5 Implementación de algoritmos de mejora de imagen.	<u>1.</u> Código fuente del algoritmo módulo etiquetador (<i>tagger</i>). <u>2.</u> Código fuente del algoritmo reductor de lluvia (<i>deraining</i>). <u>3.</u> Código fuente del algoritmo reductor de neblina (<i>dehazing</i>) <u>4.</u> Código fuente del algoritmo mejorador de condiciones luminosas (<i>light enhancer</i>) <u>7.</u> Informe final	- Python. - OpenCV - Pytorch. - Visual Studio Code. - GitHub. - Lucidchart. - Windows 11.	- Verificación del conjunto de datos preprocesado mediante análisis exploratorio y visualización. - Validación de los algoritmos de mejora de imagen a través de pruebas controladas con datos de prueba específicos.
2. Integrar los algoritmos de mejora de imagen con la estimación de fondo monocular, utilizando una integración efectiva de enfoques probados, para que se garantice la precisión de los mapas de fondo en condiciones	2.1 Diseño de la interfaz de integración. 2.2 Implementación de la interfaz de integración. 2.3 Pruebas de integración.	<u>5.</u> Código fuente de la herramienta centralizada <u>7.</u> Informe final	- Python. - Tkinter - Visual Studio Code. - GitHub. - Windows 11. - Lucidchart. -Figma	- Pruebas de integración para comprobar la comunicación fluida entre los módulos y la ejecución correcta de las funcionalidades conjuntas.

adversas.				
3. Validar el desempeño y la eficacia del sistema desarrollado mediante un análisis de resultados en escenarios, tanto en condiciones óptimas como en condiciones adversas, garantizando la función y utilidad en entornos diversos y desafiantes.	3.1 Diseño de escenarios de prueba. 3.2 Ejecución de pruebas en escenarios óptimos. 3.3 Ejecución de pruebas en escenarios adversos. 3.4 Evaluación de métricas de rendimiento.	6. Resultados de pruebas extensivas realizadas. 7. Informe final	- Python. - Visual Studio Code. - GitHub. - Lucidchart. - Windows 11. - Excel	- Comparación de los resultados con los valores de referencia para validar la precisión y eficacia del sistema en diferentes condiciones.

Tabla 3.1: Desglose para cada objetivo específico

Capítulo 4: Descripción del trabajo realizado

En este capítulo se detalla el desarrollo práctico del proyecto, siguiendo el diagrama de flujo de la [Figura 1.2](#) como hoja de ruta. Se explica cómo se abordó cada uno de los pasos, desde el etiquetado de imágenes hasta la producción de mapas de profundidad. Esta sección desglosa el trabajo paso a paso y muestra las decisiones técnicas y la metodología aplicada para superar los obstáculos que se encontraron y alcanzar los resultados que se esperan.

4.1 Descripción del proceso de solución

4.1.1 Procesado inicial de las grabaciones

Este proyecto comenzó con una extensa colección de datos: 700 GB de grabaciones de video de una cámara de tipo *dashcam* en las calles de Costa Rica. La cámara utilizada es de la marca Steren [27] y generó grabaciones en una resolución de 1080p.

Antes de poder entrenar un modelo para identificar distintas condiciones ambientales, fue necesario no solo etiquetar manualmente cada imagen para clasificar las condiciones adversas, sino también aplicar un preprocesamiento. Este paso inicial fue esencial para limpiar y estandarizar las imágenes, mejorando la calidad de los datos que alimentaron el proceso de aprendizaje automático.

4.1.1.1 Renombramiento de los videos

Antes de proceder con el etiquetado y preprocesamiento, fue crucial organizar el amplio volumen de videos crudos, los cuales inicialmente solo se distinguían por nombres genéricos que indicaban la fecha y hora de la grabación. Para imponer orden y claridad, el primer paso consistió en renombrar cada uno de

los archivos de video siguiendo una convención descriptiva y sistemática que reflejara su contenido. Se adoptó el formato *punto de partida-destino-etiquetas*, que proporciona información inmediata sobre la trayectoria y las características visuales de cada grabación, por ejemplo, *Cartago-Hatillo-Noche*. Este método de nomenclatura permitió una identificación rápida y una selección eficiente para la posterior etapa de análisis y clasificación.

4.1.1.2 Extracción de fotogramas

Tras renombrar los videos para reflejar su contenido de manera precisa, el siguiente paso en la preparación de los datos fue la extracción de *frames* individuales de cada video. Este proceso se llevó a cabo para transformar el contenido de video dinámico en una serie de imágenes estáticas que podían analizarse con mayor facilidad.

Para asegurar diversidad en las imágenes y evitar redundancia, se establece un intervalo específico entre los *frames* seleccionados, optando por capturar un *frame* cada dos segundos del vídeo. Esto permite conservar la variabilidad en las condiciones mostradas sin sobrecargar el conjunto de datos con imágenes excesivamente similares.

Cada video tuvo su propio directorio de salida, nombrado de manera idéntica al archivo de video correspondiente, donde se almacenaron los *frames* extraídos. Este método proporcionó una organización lógica y una recuperación sencilla de imágenes por video, lo que facilita las etapas de etiquetado y análisis subsecuentes. Con estos pasos, se pasa de un formato de video continuo a un conjunto manejable de imágenes individuales, listas para ser procesadas y utilizadas en el entrenamiento del modelo de inteligencia artificial.

4.1.1.3 Etiquetado de las imágenes

Para asociar de manera efectiva las condiciones ambientales con cada *frame* extraído, se utiliza un archivo CSV como mecanismo de etiquetado. Este archivo sirve como un puente entre los videos y sus correspondientes imágenes, asegurando que cada *frame* conserve las etiquetas apropiadas sobre las condiciones en que fue capturado.

El archivo CSV está estructurado, de manera que cada fila representa un *frame*, con la primera columna y contiene la ruta al archivo de imagen y las columnas subsiguientes marcando la presencia o ausencia de condiciones específicas como *noche*, *soleado*, *nublado*, *lluvia* y *neblina*. Cada condición está representada con un 1 para indicar su presencia o un 0 para su ausencia en ese *frame* particular.

Este enfoque permite una automatización eficiente en el proceso de etiquetado, lo que después facilita el entrenamiento de modelos de aprendizaje automático que requieren un etiquetado claro y consistente para cada instancia de datos. Con este sistema, se garantiza que las características relevantes de cada

imagen están claramente definidas y accesibles para análisis y procesamientos adicionales.

Para etiquetar de manera eficiente los *frames* extraídos de los videos, se utilizó un *script* diseñado específicamente para facilitar la asignación de etiquetas en bloques. Este enfoque permitió al desarrollador procesar grandes cantidades de imágenes rápidamente, aplicando etiquetas consistentes a series de *frames* que presentaban las mismas condiciones atmosféricas o de iluminación.

El proceso de etiquetado manual implica visualizar y categorizar los *frames* según las características visibles, como la presencia de lluvia, neblina, noche o condiciones soleadas, con la posibilidad de marcar múltiples condiciones simultáneamente si el *frame* lo requiere. Esto es crucial para entrenar modelos de clasificación precisos, capaces de reconocer y diferenciar entre diversas condiciones ambientales.

Al final del proceso de etiquetado, se acumula un conjunto de datos diversos, reflejado en la [Tabla 4.1](#) que representa la cantidad de *frames* etiquetados para cada combinación de condiciones.

Etiquetas	Cantidad de frames	Porcentaje de los datos
Soleado	5223	29.55 %
Lluvia y nublado	1876	10.61 %
Lluvia y noche	1214	6.87 %
Noche	4518	25.56 %
Nublado	643	3.64 %
Neblina	3020	17.09 %
Lluvia	1180	6.68 %
Total	17674	100 %

Tabla 4.1: Cantidad total de frames por condición en la base de datos

4.1.2 Modelos de procesado

En este segmento se describe detalladamente cada uno de los modelos de procesamiento de imagen que se desarrollaron y optimizaron para mejorar la estimación de fondo monocular bajo diversas condiciones ambientales. El flujo de trabajo, como se muestra en el diagrama de la [Figura 1.2](#), abarca desde la etapa inicial de etiquetado de imágenes hasta la generación final de mapas de profundidad. Cada modelo aborda un desafío específico, como optimizar imágenes nocturnas, reducir neblina y eliminar lluvia.

4.1.2.1 Estructura común de los modelos

Cada uno de los modelos con base en redes neuronales que se desarrollaron durante este proyecto, es decir, el etiquetador de imágenes (*tagger*), el modelo para reducción de neblina (*dehazing*) y el modelo para eliminación de lluvia (*deraining*), sigue una estructura modular compuesta por tres componentes principales:

- **Data Loader:** este módulo es responsable de cargar y preparar los datos de entrada para el modelo. Organiza y formatea las imágenes y etiquetas, de manera que puedan ser eficientemente procesadas durante el entrenamiento y la evaluación del modelo.
- **Código del modelo:** Aquí se define la arquitectura del modelo de aprendizaje profundo, lo que incluye capas, funciones de activación y mecanismos de propagación hacia adelante y hacia atrás. Este código es el núcleo del proceso de aprendizaje, donde se realizan las computaciones necesarias para aprender las características relevantes de las imágenes.
- **Archivo de entrenamiento:** contiene los *scripts* necesarios para ejecutar el entrenamiento del modelo, ajustar los parámetros y validar el rendimiento utilizando conjuntos de datos de prueba. Este archivo también gestiona la optimización de hiperparámetros y supervisa el proceso de aprendizaje para garantizar que el modelo se ajuste de manera efectiva a las especificaciones deseadas.

Este desglose detallado en tres secciones fundamentales para cada modelo es importante para proporcionar claridad y comprensión sobre cómo se organiza y opera cada aspecto del procesamiento de imagen en este proyecto. Al explicar estos componentes, se facilita la navegación y el entendimiento de las secciones técnicas subsecuentes del documento.

4.1.2.2 Tagger: modelo etiquetador de imágenes

Siguiendo la estructura explicada previamente, el primer módulo por detallar es el *tagger*, que desempeña un papel crucial en la preparación inicial de los datos para los procesos subsiguientes de mejoramiento y análisis de imagen. Este módulo está diseñado para etiquetar de manera automática los *frames* extraídos de los videos, al asignar etiquetas pertinentes que describen las condiciones ambientales visibles en cada imagen.

Con respecto a su funcionalidad, el *tagger* es esencial para garantizar que los datos ingresados en los modelos de mejora de imagen y de generación de mapas de profundidad estén correctamente clasificados. Al utilizar un conjunto de etiquetas predefinidas, como *noche*, *neblina*, *lluvia*, entre otras, el módulo categoriza cada *frame* según las características visuales observadas. Esta etiquetación precisa es fundamental para el entrenamiento efectivo de los modelos subsiguientes, pues permite que los algoritmos aprendan a identificar y reaccionar de manera adecuada a diferentes condiciones ambientales.

La automatización del proceso de etiquetado mediante el *tagger* no solo aumenta la eficiencia del proyecto al reducir la necesidad de intervención manual

extensa, sino que también mejora la consistencia y precisión de las etiquetas aplicadas, lo que facilita una base sólida para las etapas de procesamiento y análisis que siguen. A continuación, se detalla el código del modelo del *tagger*, explicando cómo se implementa esta funcionalidad y de qué forma contribuye al flujo de trabajo general del proyecto.

4.1.2.2.1 Arquitectura del modelo

El modelo *tagger* es clave para el proyecto, ya que etiqueta de forma automática las imágenes según las condiciones ambientales capturadas en cada *frame*. Este modelo emplea una arquitectura de red neuronal convolucional de doble entrada ("DualInputCNN"), diseñada para evaluar y etiquetar de manera efectiva las distintas condiciones ambientales que pueden variar significativamente entre diferentes secciones de una imagen.

El modelo DualInputCNN fue seleccionado basado en el artículo "A Deep Learning Approach for Speed Prediction and Energy Management of Hybrid Electric Vehicles" [42]. Este enfoque se eligió debido a su capacidad para manejar múltiples entradas y canales, permitiendo una mejor extracción de características relevantes.

La elección de un diseño de doble entrada se fundamenta en la necesidad de diferenciar y analizar por separado las características atmosféricas y de iluminación que suelen concentrarse en diferentes secciones de una imagen. Por ejemplo:

- **Sección superior:** por lo general, incluye el cielo, que es crítico para identificar condiciones como nublado, soleado o de noche, ya que estos estados se reflejan predominantemente en el color y la iluminación del cielo.
- **Sección inferior:** esta área es crucial para detectar fenómenos como lluvia o neblina, donde la presencia de gotas en el aire o la reducción de la visibilidad cerca del suelo son indicativos claros de estas condiciones.

Para observar dicha división de una manera clara en un ejemplo práctico, se puede observar la [Figura 4.1](#), con un diagrama sobre cómo se dividen las imágenes para ingresarlas al modelo.



Figura 4.1: Secciones de la entrada dual del tagger

Esta separación permite que el modelo aplique enfoques de procesamiento específicos para cada sección, lo que optimiza la detección y clasificación basada en las características visuales más relevantes de cada parte. La arquitectura del *tagger* utiliza ResNet-152 [28] como base, aprovechando sus capas convolucionales profundas para extraer características ricas de las imágenes. Las principales partes del modelo incluyen:

1. **Capas base:** este modelo aprovecha un sistema existente llamado ResNet-152 [28], conocido por su eficacia en reconocer diversos patrones en imágenes. No se utilizan todas las partes de ResNet-152; en lugar de eso, solamente se toman las primeras secciones, que son expertas en detectar características generales en las imágenes, como formas y texturas. Esto permite usar conocimientos ya aprendidos por el modelo para hacer el trabajo de manera eficiente y efectiva.
2. **Pooling adaptativo:** una vez que se obtienen las características básicas de la parte superior e inferior de la imagen es necesario asegurarse de que todas las secciones tengan el mismo tamaño para que puedan compararse y analizarse correctamente. Aquí es donde entra el *pooling adaptativo*, que es un paso que ajusta el tamaño de estas particularidades para que sean uniformes, lo que facilita mucho el proceso de análisis posterior.
3. **Clasificador:** por último, estas características normalizadas se toman y se combinan para obtener una imagen completa de lo que sucede en cada foto. Esta información combinada se pasa a través de unas capas especiales que decidirán cuáles etiquetas aplicar a la imagen, como *noche*, *lluvia* o *neblina*. Este paso final es donde realmente se determina el estado del ambiente capturado en la imagen.

El modelo ResNet-152 fue elegido para esta aplicación debido a su alta eficacia en la identificación y clasificación de imágenes. Este modelo, basado en la arquitectura de aprendizaje residual, permite entrenar redes neuronales muy profundas sin perder precisión. Según el artículo "Deep Residual Learning for Image Recognition" de Kaiming He et al. [43], ResNet-152 tiene un excelente rendimiento en la clasificación de imágenes, con una tasa de error top-5 del 4.49% en el conjunto de validación de ImageNet [44].

Para comprender mejor el proceso de clasificación realizado por el *tagger*, se puede observar la [Figura 4.2](#), donde, tras la unificación de las características extraídas de las secciones superior e inferior de la imagen, se realiza una clasificación minuciosa. Este procedimiento utiliza una capa clasificadora que se compone de varios nodos de salida, específicamente cinco, cada uno correspondiente a una condición ambiental específica: noche, nublado, soleado, lluvia o neblina.

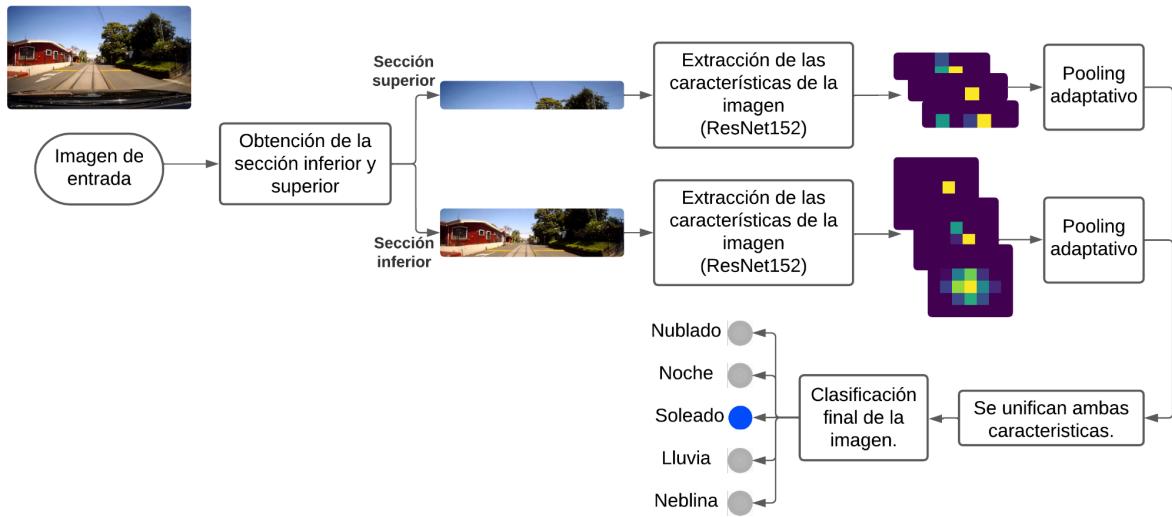


Figura 4.2: Diagrama de flujo del tagger

Cada *bit* de salida de la clasificación representa la presencia o ausencia de una de estas condiciones en la imagen analizada. Por ejemplo, un *bit* activo (1) en la posición correspondiente a *noche* indica que la imagen se tomó en condiciones nocturnas, mientras que un *bit* activo en la posición de *lluvia* señala que está lloviendo. Este esquema es consecuente con la clasificación explicada en la [sección 4.1.1.3](#) del presente documento, obteniendo una estructura de *bits* igual a la que se utiliza en el archivo de etiquetas para cada uno de los *frames*.

4.1.2.2.2 Data loader del modelo

Se desarrolló un componente de carga de datos, conocido como *data loader*, diseñado para manejar eficientemente las imágenes y sus etiquetas asociadas para el entrenamiento del modelo. Este cargador de datos es esencial para la preparación y el manejo adecuado de los conjuntos de datos que se utilizan en el entrenamiento de algoritmos de aprendizaje automático.

El *data loader* funciona leyendo desde el archivo CSV que contiene las rutas a las imágenes y las etiquetas correspondientes a las condiciones ambientales, como noche, neblina, lluvia, entre otras. Este archivo actúa como una guía que relaciona cada imagen con sus características ambientales específicas, que son esenciales para el entrenamiento del modelo.

El proceso comienza con la lectura de este archivo CSV. Luego, se filtran las entradas que no tienen etiquetas asignadas, lo que asegura que solo se incluyan en el procedimiento de entrenamiento aquellas imágenes que tienen información relevante. Cada imagen se carga del disco según la ruta especificada en el CSV y se aplican transformaciones predefinidas si es necesario, como cambio de tamaño. Estas transformaciones son fundamentales para preparar las imágenes para que se procesen de manera uniforme por el modelo de red neuronal.

Finalmente, cada imagen junto con sus etiquetas se devuelve en un formato adecuado para su uso en el entrenamiento del modelo. Esta estructura no solo facilita la manipulación eficiente de grandes volúmenes de datos, sino que también

asegura que cada imagen se presente al modelo en la forma óptima para aprender de sus características.

4.1.2.2.3 Entrenamiento del modelo

El proceso de entrenamiento del modelo se aborda con un enfoque estructurado para optimizar el desempeño en la identificación de condiciones atmosféricas adversas en imágenes. Este procedimiento integra varios componentes clave que trabajan en conjunto para afinar la capacidad del modelo de predecir etiquetas como *noche*, *lluvia*, *neblina*, entre otras.

Inicialmente, el entrenamiento empieza con la configuración de los datos de entrada y la preparación de los conjuntos de datos de entrenamiento y validación. Se emplea el cargador de datos personalizado que se explicó en la [sección 4.1.2.2.2](#), que organiza y proporciona acceso a las imágenes y sus etiquetas asociadas a partir del archivo CSV. Este enfoque garantiza que el modelo recibe información coherente y etiquetada de manera correcta durante las sesiones de entrenamiento.

Una vez configurados los datos, el modelo se entrena utilizando un bucle que procesa los datos en lotes. Durante cada época del entrenamiento, se ajustan los pesos de la red neuronal para minimizar el error en las predicciones del modelo, comparando la salida del modelo con las etiquetas reales a través de una función de pérdida.

El modelo utiliza una función de pérdida de entropía cruzada binaria con logits (BCEWithLogitsLoss [29]), ideal para problemas de clasificación multietiqueta como el que se utiliza en este caso, donde cada etiqueta (noche, lluvia, neblina, etc.) se trata de manera independiente.

Para explicar la función de pérdida de manera sencilla, es posible describirla como un método que mide cuánto se equivoca el modelo en sus predicciones. Es decir, cada predicción es una apuesta del modelo sobre si una característica específica, como *noche* o *lluvia*, está presente en la imagen o no. La función de pérdida calcula el error de estas apuestas al comparar lo que el modelo pensaba que era cierto con lo que realmente lo es, de acuerdo con las etiquetas reales del archivo CSV.

La función primero transforma las predicciones del modelo, que pueden ser cualquier número, en valores entre 0 y 1, que representan probabilidades. Esto se hace mediante la función sigmoid, que suaviza los resultados para que se asemejen a verdaderas probabilidades.

Luego, la función de pérdida evalúa estas probabilidades: si el modelo estaba muy seguro de algo que resultó ser falso o no estaba seguro de algo que resultó ser cierto, la pérdida es mayor. Esto se calcula utilizando la fórmula de entropía cruzada, que penaliza las predicciones incorrectas y alejadas de la realidad.

4.1.2.3 Algoritmo para el mejoramiento de imágenes nocturnas

En el contexto del proyecto actual, se enfrenta un desafío particular con el procesamiento de imágenes de conducción nocturna, donde las condiciones de baja iluminación se complican aún más por la presencia de fuentes de luz intensamente brillantes, como los faros de los vehículos o las luces de calle. Estas fuentes pueden crear elementos significativos en las imágenes, como el deslumbramiento y los reflejos en ciertas áreas, mientras que otras zonas se encuentran completamente oscuras. Esto deteriora la calidad de las imágenes y dificulta su análisis posterior con técnicas convencionales de visión por computadora.

Este problema se observa en la [Figura 4.3](#), donde las fuentes de luz de la imagen son sumamente fuertes para el lente, lo que causa deslumbramiento, mientras que en las zonas oscuras se observa muy poca información por la falta de iluminación.



Figura 4.3: Ejemplo de imagen nocturna

Por otro lado, no se logra encontrar un conjunto de datos en escenarios para conducción que tenga las características mencionadas. Esto se debe a que hay varios *datasets* enfocados en el esclarecimiento de imágenes, pero dichos datos se crean tomando la misma foto con diferente tiempo de exposición. En la Figura 4.4 se puede observar un ejemplo entre dos imágenes del *dataset* LOL (LOw-Light dataset) [30].



Figura 4.4: Ejemplo de par de imágenes del *dataset* LOL [30]

Por estos motivos se desarrolló un algoritmo que utiliza la librería OpenCV, aprovechando su robustez y eficiencia en el procesamiento de imágenes. La

elección de este método se justifica por la capacidad del algoritmo para manejar de manera efectiva las variaciones intensas de luz presentes en escenas nocturnas.

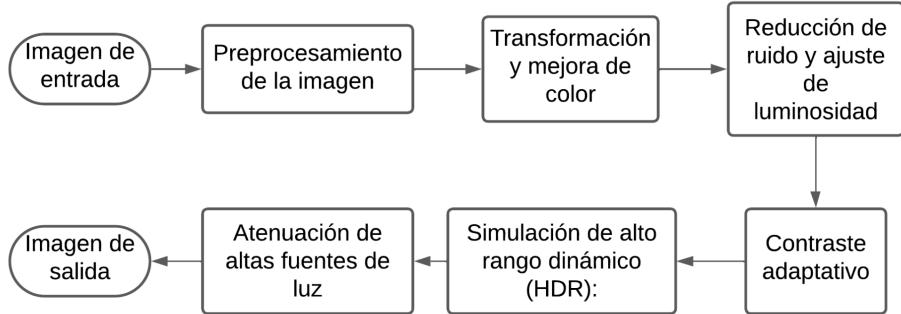


Figura 4.5: Diagrama de flujo del algoritmo para el mejoramiento de imágenes nocturnas

En la [Figura 4.5](#) se muestra el diagrama de flujo que se sigue para obtener el mejoramiento de las imágenes nocturnas, donde se presenta cada una de sus etapas. Ahora bien, al detallar el diagrama anterior, se puede hacer un desglose del paso a paso y se tiene lo siguiente:

1. **Preprocesamiento:** la imagen inicial, ya sea en forma de matriz o tensor, se estandariza a un formato uniforme que facilita los pasos de procesamiento posteriores. Esta etapa incluye ajustes dimensionales para trabajar con una imagen individual en lugar de un conjunto.
2. **Transformación y mejora de color:** se convierte la imagen al espacio de color LAB, que separa la luminancia de los componentes de color. Esto permite manipular la luminancia sin alterar los colores, lo cual es crítico para mejorar la visibilidad sin distorsionar la paleta original. Esto se realizó con la función cvtColor [31] de OpenCV con el código de color COLOR_RGB2BGR.
3. **Contraste adaptativo:** se utiliza CLAHE (Contrast Limited Adaptive Histogram Equalization) en el canal de luminancia. CLAHE mejora el contraste de la imagen adaptativamente y **mejora** la visibilidad en áreas oscuras mientras evita la sobreexposición en áreas más brillantes. Se utilizó la función createCLAHE de OpenCV.
4. **Reducción de ruido y ajuste de luminosidad:** se aplican técnicas de reducción de ruido para suavizar la imagen, seguido de un ajuste de gamma para modificar la luminosidad general, haciendo la imagen más clara u oscura según sea necesario para aproximarse a la apariencia de condiciones de luz diurna. Se utilizó la función fastNIMeansDenoisingColored de OpenCV para la reducción de ruido.
5. **Simulación de alto rango dinámico (HDR):** para maximizar los detalles, tanto en zonas oscuras como iluminadas, se simula un efecto HDR combinando imágenes con diferentes niveles de exposición. Esto ayuda a recrear un rango más amplio de intensidades luminosas que lo que la cámara capturó originalmente. Se utilizó la función createMergeMertens de OpenCV.
6. **Atenuación de altas fuentes de luz:** se identifican y atenúan áreas excesivamente brillantes para evitar distracciones y mejorar el equilibrio

general de la imagen. Esto es útil en imágenes nocturnas donde las fuentes de luz pueden ser desproporcionadamente brillantes.

Por lo tanto, el algoritmo presentado, aunque se basa exclusivamente en funciones nativas de OpenCV para su implementación, demuestra una notable eficacia al tratar con imágenes nocturnas. A pesar de los desafíos presentes en este tipo de imágenes, como las grandes fuentes de luz provenientes de faros de vehículos o iluminación pública, el algoritmo logra resultados satisfactorios. La combinación de técnicas como CLAHE, simulación HDR y la atenuación de altas luces permite mitigar los efectos adversos de estas intensas fuentes de luz y mejorar de manera significativa la visibilidad y la calidad de las imágenes. Esto recalca la capacidad del algoritmo para adaptarse y funcionar de forma eficiente dentro de las limitaciones de las herramientas disponibles y ofrecer una solución práctica y efectiva para optimizar imágenes en condiciones de baja luminosidad.

4.1.2.4 Dehazing: modelo eliminador de neblina

La neblina es un fenómeno atmosférico que tiene un impacto significativo sobre las imágenes capturadas al aire libre. Este fenómeno se produce cuando partículas diminutas de agua en el aire dispersan la luz solar, lo que provoca que la luz se esparza en múltiples direcciones y crea un efecto blanquecino que reduce la visibilidad y el contraste en las fotografías. Este efecto de dispersión de la luz no solo afecta la claridad y los detalles visuales de las imágenes, sino que también distorsiona la percepción de profundidad y color, lo cual es crítico en diversas aplicaciones de visión computarizada.

La presencia de neblina se considera una condición adversa en la visión por computadora debido a que interfiere con la capacidad de los algoritmos para interpretar y analizar correctamente las escenas. En aplicaciones críticas como la navegación autónoma y la generación de mapas de fondo monocular, una visión clara y precisa es fundamental para la seguridad y eficacia del sistema. La neblina puede ocultar obstáculos, señales y otros elementos importantes, lo que aumenta el riesgo de errores en la interpretación de datos visuales.

Ante estos desafíos, se desarrolló un modelo específico basado en la arquitectura de red neuronal Unet para el proceso de eliminación de neblina, conocido comúnmente como *dehazing*. El modelo Unet es adecuado para este tipo de tareas debido a su eficacia en el manejo de imágenes en el ámbito de píxel y su capacidad para trabajar con detalles finos, lo que es crucial para restaurar la calidad original de las imágenes afectadas por la neblina. Este enfoque permite no solo recuperar detalles y mejorar la visibilidad, sino también facilitar procesos subsecuentes como la generación de mapas de fondo monocular con mayor precisión y fiabilidad.

4.1.2.4.1 Datos que se utilizan para el entrenamiento

Para el entrenamiento del modelo se utiliza un enfoque que se basa en el aprendizaje supervisado, que requiere pares de imágenes donde una muestra la

escena sin neblina y la otra con neblina. Este método permite al modelo aprender a identificar y eliminar la neblina de las imágenes de manera efectiva.

Se buscaba utilizar un conjunto robusto y diversificado de datos que simularan de manera realista las condiciones de visibilidad adversas causadas por la neblina. El uso de un solo conjunto de datos a menudo limita la variabilidad y la cantidad de datos necesarios para un entrenamiento efectivo. Por lo tanto, se optó por combinar dos conjuntos de datos complementarios, aprovechando las fortalezas distintivas de cada uno para desarrollar un modelo más generalizado y eficiente.

El primer *dataset* que se utiliza es el O-Haze [33], que incluye un total de 45 pares de imágenes capturadas en entornos al aire libre con temáticas variadas. Lo notable de O-Haze es que la neblina presente en las imágenes no es simulada por *software*, sino creada físicamente usando máquinas de neblina, lo que proporciona un realismo superior en términos sobre cómo la neblina interfiere con la visibilidad natural. Un ejemplo de pares de imágenes de este *dataset* se puede observar en la Figura 4.6.



Figura 4.6: Ejemplo de par de imágenes del *dataset* O-Haze [33]

Sin embargo, la cantidad de imágenes en O-Haze no es suficiente para entrenar un modelo robusto debido a su limitado volumen. Por esta razón, se integró también el *dataset* Foggy Cityscapes [32], que contiene 2,975 pares de imágenes con base en el conocido *dataset* Cityscapes [2], pero con neblina agregada artificialmente. Esta inclusión masiva de datos no solo amplía el total de ejemplos de entrenamiento disponibles, sino que también introduce una diversidad de escenarios urbanos y configuraciones que son esenciales para la generalización del modelo. En la Figura 4.7 se muestra un ejemplo de un par de imágenes del *dataset* Foggy Cityscapes.



Figura 4.7: Ejemplo de par de imágenes del dataset Foggy Cityscapes [32]

La combinación de estos dos *datasets* ha permitido al modelo aprender de imágenes con neblina generada de manera controlada y real, así como de un gran volumen de datos con variaciones artificiales. Esto asegura una robustez y versatilidad significativas en el desempeño del modelo final de *dehazing*.

4.1.2.4.2 Arquitectura del modelo

La arquitectura del modelo de eliminación de neblina se basa en la red UNet, la cual se explicó en la [sección 2.2.4.5](#), una estructura bien establecida y eficaz para tareas de procesamiento de imágenes donde la precisión espacial es importante, como la segmentación semántica y, en este contexto, la eliminación de neblina. La U-Net es particularmente adecuada para este propósito debido a su capacidad para trabajar con detalles a nivel de píxel, lo que es esencial cuando se trata de restaurar la claridad en imágenes afectadas por la neblina.

El modelo implementado consiste en una serie de capas codificadoras seguidas de capas decodificadoras, con conexiones de concatenación entre las capas correspondientes del codificador y del decodificador. Esta estructura permite que el modelo no solo aprenda a identificar y eliminar la neblina, sino también a reconstruir la imagen con detalles precisos preservados, algo crucial para aplicaciones como la generación de mapas de fondo monocular.

En la sección del codificador, el modelo comienza con capas convolucionales que progresivamente reducen la dimensión espacial de la imagen mientras aumentan la profundidad de los canales. Esto permite que el modelo capture características a diferentes escalas y abstracciones, lo que es vital para entender la distribución y densidad de la neblina en la imagen.

Por otro lado, en el decodificador se utilizan capas de convolución transpuestas para incrementar gradualmente las dimensiones espaciales de las representaciones de las características mientras se reduce la profundidad de los canales. Esto ayuda a reconstruir la imagen desde la representación codificada, afinando detalles y claridad a medida que avanza hacia la salida del modelo.

Además, se implementaron conexiones de salto entre las capas del codificador y las del decodificador, que fueron fundamentales en la arquitectura UNet, ya que permiten que la información de alta resolución ignore las capas más profundas del modelo. Esto facilita la restauración efectiva de detalles finos en la imagen de salida, que pueden perderse solo con las operaciones de codificación y decodificación.

Al final de la red, se utilizan capas convolucionales para refinar las características y asegurar que la salida del modelo sea una imagen clara y libre de neblina. Estas capas trabajan para ajustar los detalles finales y mejorar la nitidez de la imagen resultante.

Es importante destacar que cada capa en la estructura U-Net maneja una cantidad variable de canales, por lo que comienza con menos canales en las primeras capas y aumenta a medida que la imagen se desplaza hacia el centro de la

red. Esta expansión y posterior reducción de canales facilita el aprendizaje de representaciones complejas y detalladas de la neblina y su eliminación.

Lo explicado se puede observar en la [Figura 4.8](#), en la que se muestra el paso a paso del diagrama de flujo para dicha arquitectura, donde es claro la forma de U que forma su arquitectura debido al proceso de codificación y decodificación, el cual es crucial para asegurar que la red pueda enfocarse en las características más relevantes para la tarea de reducir la neblina, mientras mantiene o mejora la calidad de la imagen de salida. La forma de representar la arquitectura UNet en la [Figura 4.8](#) fue inspirada por los diagramas presentados por A. Hilbert *et al.* [34].

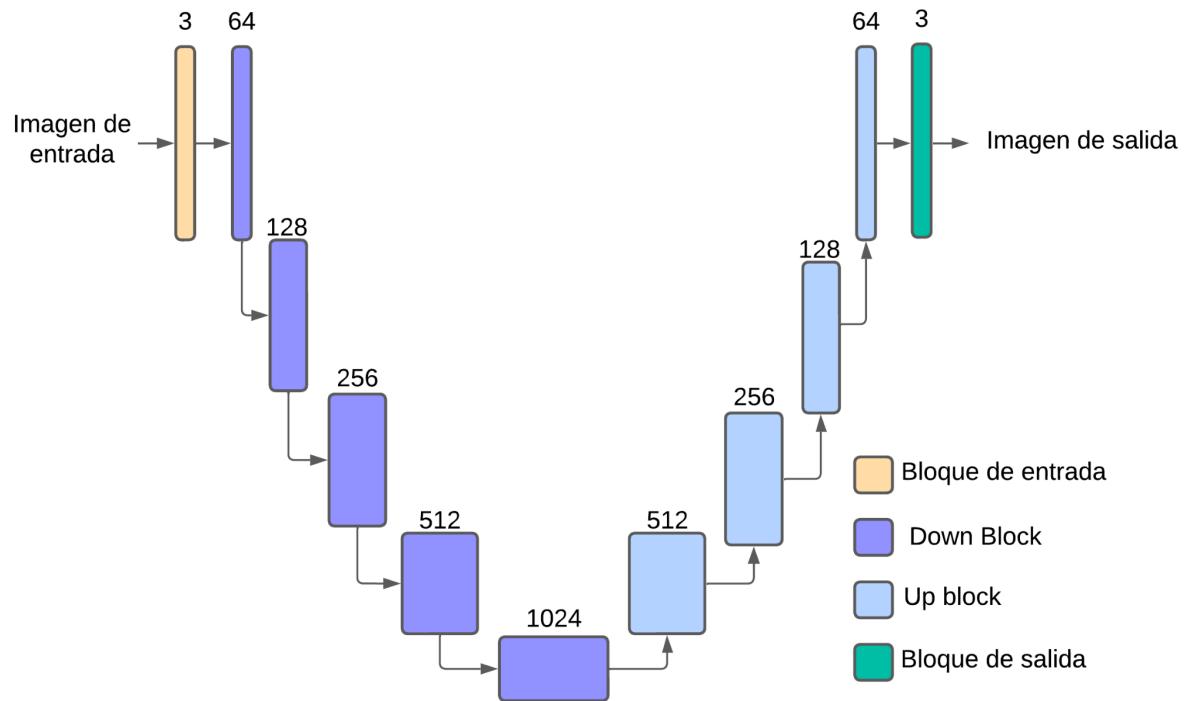


Figura 4.8: Diagrama de arquitectura de la Dehazing Net

Entrando más en detalle en el diseño, el *down block*, representado en la [Figura 4.8](#) de color morado, es un bloque de procesamiento dentro de la red U-Net que está compuesto por varios componentes clave que trabajan en conjunto para procesar y condensar la información de la imagen de entrada. La arquitectura específica de este tipo de bloques se puede observar en la [Figura 4.9](#).

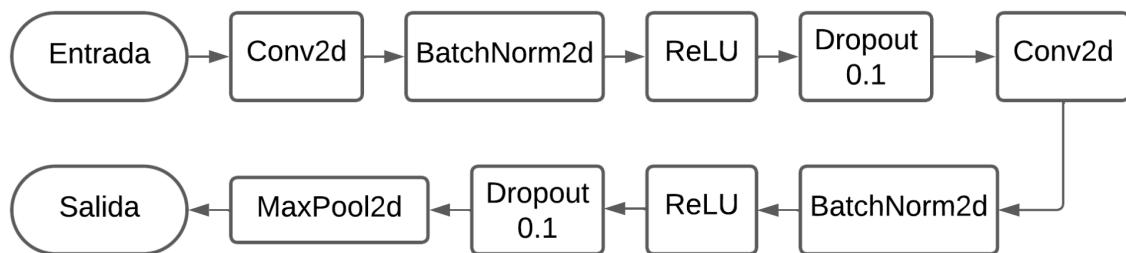


Figura 4.9: Diagrama de arquitectura del down block

El propósito de cada uno de estos componentes se puede explicar de la siguiente manera:

- **Conv2d:** las capas de convolución aplican filtros a la imagen para detectar características específicas como texturas, bordes y otros patrones visuales. Estas capas son fundamentales para extraer información útil de las imágenes.
- **BatchNorm2d:** las capas de normalización por lotes ayudan a estabilizar el proceso de aprendizaje al ajustar las activaciones para mantener una distribución más consistente.
- **ReLU:** la función de activación ReLU se utiliza para introducir no linealidades en el modelo, lo que permite que la red capture relaciones complejas en los datos. Además, activa solo las neuronas que identifican características relevantes, apagando las demás para mejorar la eficiencia computacional y la capacidad del modelo.
- **Dropout:** se emplea para mitigar el sobreajuste, descartando aleatoriamente una fracción de las características durante el entrenamiento. Esto asegura que la red no depende demasiado de cualquier entrada o patrón específico, lo que promueve un modelo más robusto y generalizable.
- **MaxPool2d:** esta operación reduce la dimensión espacial de las representaciones para disminuir la cantidad de parámetros y cálculos en la red. El agrupamiento máximo selecciona las características más prominentes. Siguiendo con la explicación de los bloques de procesamiento de la [Figura 4.8](#) se encuentran los *up blocks*, representados de color celeste. Estos bloques se encargan de reconstruir y refinar la información de la imagen a medida que el flujo de datos avanza desde la codificación hacia la decodificación. La estructura de estos bloques se puede observar en la [Figura 4.10](#).

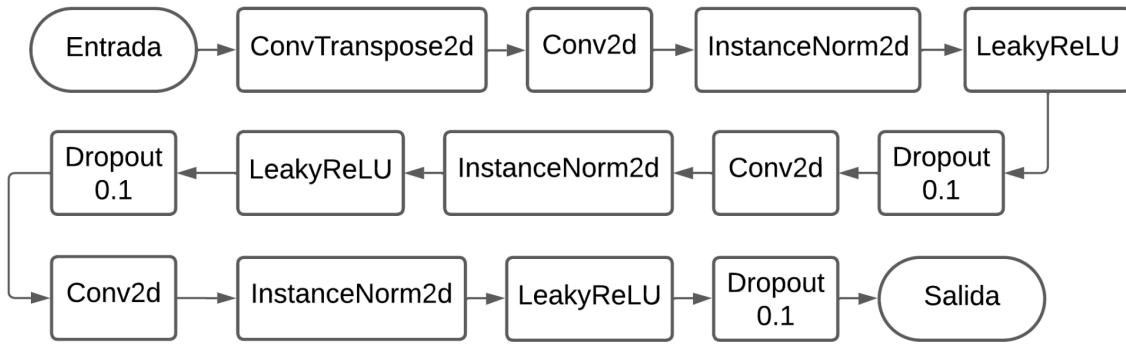


Figura 4.10: Diagrama de arquitectura del up block

En el *up block* de la red U-Net, cada componente tiene un propósito en el proceso de reconstrucción y refinamiento de las características de la imagen. Enseguida, se detalla la función específica de cada elemento dentro de este bloque.

- **ConvTranspose2d:** esta capa, a veces llamada deconvolución, se utiliza para aumentar el tamaño espacial de las características. Su papel es invertir el efecto de compresión.
- **Conv2d:** las capas de convolución estándar se usan en la fase de decodificación para refinar las características a medida que se expanden. Ayudan a ajustar y mejorar los detalles que se reintroducen en la imagen a través del proceso de *upsampling*.
- **InstanceNorm2d:** en lugar de normalizar las características sobre un lote entero de datos, la normalización de instancias se aplica individualmente a cada imagen en el lote. Esto es útil en aplicaciones donde los datos de entrada pueden tener variaciones significativas en iluminación o color, como en tareas de procesamiento de imágenes.
- **LeakyReLU:** esta función de activación es una variante de la ReLU que permite una pequeña cantidad de gradiente cuando la unidad no está activa, es decir, permite un pequeño flujo de gradientes negativos. Esto ayuda a mantener viva la información durante el proceso de entrenamiento, previniendo el problema de las neuronas muertas común en ReLU.
- **Dropout:** al igual que en los bloques descendentes, la aplicación de *dropout* ayuda a evitar el sobreajuste, pero con un enfoque más conservador en la reconstrucción para asegurar que la diversidad de características se mantenga a través del modelo.

Funcionalmente, los *up blocks* en la red U-Net ayudan a recuperar los detalles de la imagen que se pueden perder durante el proceso inicial de compresión. La función de concatenación es muy útil aquí porque mezcla características detalladas de pasos anteriores con la información actual que se está expandiendo. Esto es clave para asegurarse de que la imagen final se vea clara y precisa. Este procedimiento ayuda a que la red reconstruya la imagen original cuidadosamente, manteniendo todos sus detalles importantes, lo cual es vital para

aplicaciones donde la calidad de la imagen es crítica, como en la mejora de imágenes con neblina.

Con respecto al bloque de salida, representado en color verde en la [Figura 4.8](#), este se trata de una capa de convolución final de tipo Conv2d que realiza una conversión final de 64 canales a 3 para obtener una imagen en formato RGB tradicional.

Lo explicado se puede resumir con la [Figura 4.11](#), donde se observa el paso a paso de una imagen de entrada por los distintos niveles de la red UNet. Para visualizarlo de manera amigable, se promedió cada uno de los canales a un solo canal en escala de grises. A pesar de que esto no es completamente fiel a cómo se realiza el procesamiento en la realidad, permite observar de qué forma la imagen se modificó hasta tener una imagen de salida sin neblina.

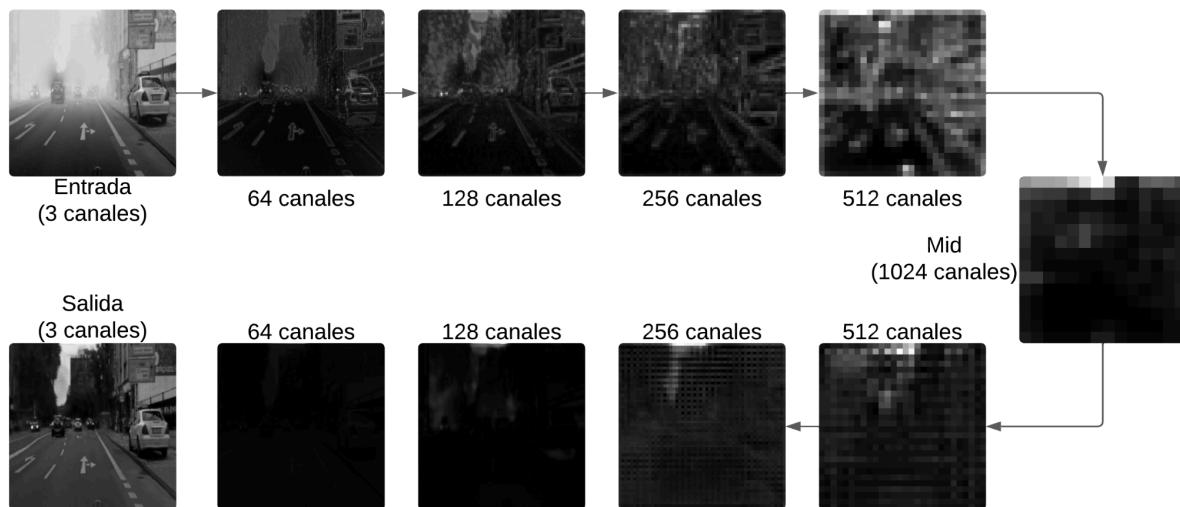


Figura 4.11: Visualización de una imagen a través de la red de desneblinado

4.1.2.4.3 Data loader del modelo

El *data loader* que maneja imágenes con y sin neblina funciona de una manera bastante sencilla, pero efectiva. Primero, necesita saber desde dónde obtener las imágenes, lo cual se le indica mediante una dirección de un directorio. En esta carpeta, busca dos tipos de imágenes: las que tienen neblina y las correspondientes imágenes claras, sin neblina.

Utiliza estas rutas para formar pares de imágenes que luego pasan por un proceso de transformación opcional, que ajusta el tamaño de las imágenes y las convierte en tensores, que son el formato adecuado para trabajar con redes neuronales en PyTorch.

Por último, cada vez que se solicita un par de imágenes del *dataset*, el *data loader* devuelve estos pares transformados. Esto permite que el modelo de red neuronal trabaje directamente en comparar y analizar las diferencias entre la imagen con neblina y su versión clara, aprendiendo cómo eliminar de manera eficaz la neblina para restaurar la visión clara de la escena.

4.1.2.4.4 Entrenamiento del modelo

Para entrenar el modelo, se partió dividiendo el conjunto de datos en una proporción del 80 % para entrenamiento y un 20 % para validación. Esto permite que el modelo aprenda con la mayoría de los datos mientras se reserva una parte para verificar la generalización del aprendizaje sin exponer el modelo a estos datos durante el entrenamiento.

El entrenamiento del modelo se estructuró en torno a una función de pérdida diseñada para optimizar varios aspectos críticos de la calidad de imagen. Esta función de pérdida no solo enfatiza la precisión al reconstruir imágenes, sino que también asegura la fidelidad en la reproducción del color y la textura, elementos esenciales para aplicaciones de visión por computadora para generar mapas de fondo monocular.

El proceso de entrenamiento utiliza una combinación de métricas de pérdida que juntas forman una estrategia integral para evaluar y mejorar el rendimiento del modelo.

- **Reconstrucción directa (L1 Loss):** este es el componente principal de la función de pérdida, con un peso del 60 %. Utiliza la pérdida L1 para minimizar las diferencias absolutas píxel a píxel entre la imagen generada y la imagen objetivo (sin neblina), lo que es crucial para asegurar una reconstrucción precisa de detalles visuales en la imagen resultante.
- **Pérdida perceptual:** se emplea la pérdida de error cuadrático medio (MSE) para capturar las diferencias perceptuales entre la imagen generada y la imagen clara enfocándose en la fidelidad general de la imagen. Esta pérdida tuvo un peso del 20 %.
- **Dispersión de píxeles:** la pérdida MSE también se aplica a la desviación estándar de los píxeles de las imágenes, para asegurar que la variabilidad de color en la imagen generada se asemeje a la de la imagen clara, preservando la textura y variación natural. Este ámbito tiene un peso del 5 %.
- **Pérdida de diferencia de color:** calcula la diferencia de color promedio entre las imágenes que se generan y las claras, utilizando también MSE, para asegurar que los colores se mantengan fieles al original.
- **Pérdida de equilibrio de color:** esta métrica se concentra en alinear los canales de color promedio de las imágenes que se generan con los de las imágenes claras. Especialmente, se da más peso al canal verde si se detecta un sesgo hacia este color en las imágenes generadas.

Para obtener el 15 % restante de la función de pérdida se suma la pérdida de diferencia de color y la pérdida de equilibrio de color. Por lo tanto, la función de pérdida total se define de la siguiente manera:

$$L = 0.60 * L + 0.20 * \text{Perceptual} + 0.05 * \text{Dispersión} + 0.15 * (\text{Color}_{\text{diff}} + \text{Color}_{\text{balance}})$$

Esta función de pérdida asegura que cada aspecto de la imagen, desde la estructura hasta el color y la textura, se optimice durante el entrenamiento. Esto resulta en imágenes de alta calidad que son vitales para tareas de visión por computadora de alta precisión.

4.1.2.5 Deraining: modelo eliminador de lluvia

La lluvia, al igual que la neblina, representa un desafío considerable para la captura de imágenes claras y precisas en entornos exteriores. La presencia de gotas de lluvia en el aire y sobre la lente puede causar distorsiones significativas en las fotografías, debido a que las gotas actúan como lentes pequeñas que refractan y dispersan la luz, creando artefactos luminosos y velos que reducen la nitidez y el contraste de la imagen. Estos efectos no solo afectan la calidad estética de las imágenes, sino que también complican la interpretación y su análisis por parte de sistemas automatizados de visión computarizada.

En aplicaciones críticas, como la conducción autónoma y la monitorización del tráfico, es esencial contar con imágenes libres de distorsiones causadas por la lluvia para garantizar una operación segura y eficiente. Las gotas de lluvia pueden ocultar o distorsionar información crucial del entorno, como señales de tráfico, peatones y otros vehículos, lo que incrementa el potencial de errores en la percepción y decisiones automatizadas con base en imágenes.

Debido a los resultados con el modelo de *dehazing* utilizando la arquitectura Unet, se decidió implementar una estrategia similar para el desarrollo de un modelo de eliminación de lluvia o *deraining*. La red Unet es ideal para este propósito debido a su capacidad para manejar detalles a nivel de píxel y su habilidad para reconstruir imágenes con alta fidelidad. Esta arquitectura permite una separación efectiva de las gotas de lluvia de la imagen original, restaurando la claridad y la visibilidad sin comprometer los detalles cruciales necesarios para aplicaciones de visión por computadora.

4.1.2.5.1 Datos que se utilizan para el entrenamiento

Para abordar el problema de la visibilidad reducida en imágenes afectadas por gotas de lluvia adheridas a ventanas o lentes de cámaras, se empleó el *dataset* Raindrop [5], que consta de 1110 pares de imágenes: una con gotas de lluvia y otra sin ellas. Este enfoque permite entrenar modelos de visión computarizada para identificar y eliminar las distorsiones causadas por la lluvia, transformando imágenes degradadas por la lluvia en versiones limpias y claras. En la [Figura 4.12](#) se observa un par de imágenes de ejemplo extraídas de dicho *dataset*.

En este *dataset*, las imágenes con lluvia se generaron de manera controlada al rociar gotas directamente sobre el lente de la cámara. Este método permite capturar la misma escena, tanto en condiciones normales como bajo la influencia de la lluvia. Al tomar imágenes de esta escena con y sin gotas, se facilita el entrenamiento del modelo para identificar y eliminar las distorsiones causadas por las gotas, manteniendo intactos los detalles y la integridad de la escena de fondo. Esta técnica asegura que el modelo pueda aprender con precisión cómo la lluvia afecta visualmente a las imágenes y desarrollar estrategias efectivas para restaurar la claridad visual en condiciones adversas.



Figura 4.12: Ejemplo de par de imágenes del dataset Raindrop [5]

4.1.2.5.2 Arquitectura del modelo

En el desarrollo del modelo de eliminación de lluvia se opta por mantener la arquitectura Unet debido a su comprobada eficacia y versatilidad en tareas de procesamiento de imágenes donde la exactitud en la reconstrucción detallada es crucial, gracias a la experiencia ganada por el modelo eliminador de neblina.

Esta arquitectura demuestra ser útil en la manipulación de imágenes con distorsiones específicas y localizadas, como las causadas por gotas de lluvia. Al utilizar la Unet, el modelo puede aprender eficientemente, tanto las características globales del entorno como las perturbaciones locales inducidas por la lluvia, lo que facilita su identificación y corrección en el proceso de *deraining*. Además, la capacidad de Unet para trabajar con detalles finos es esencial para restaurar la calidad original de las imágenes afectadas por la lluvia, lo que asegura que la información visual clave no se pierda en el procedimiento de eliminación de la lluvia.

Después de establecer la efectividad de la arquitectura Unet, es importante considerar estrategias que puedan mejorar aún más su capacidad para manejar situaciones específicas como la eliminación de gotas de lluvia en imágenes. Aunque la Unet por sí sola es poderosa, la introducción de bloques de atención convolucionales representa un avance significativo que aprovecha y extiende las capacidades inherentes de la Unet. Esta estrategia de bloques de atención fue inspirada en el trabajo de Qian *et al.* en su publicación *Attentive generative adversarial network for raindrop removal from a single image*.

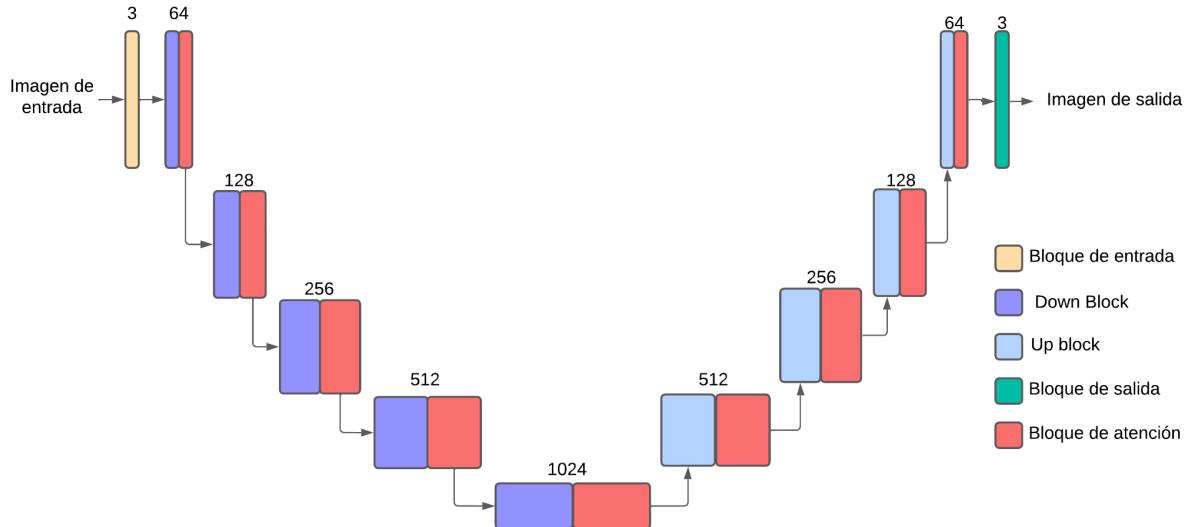


Figura 4.13: Diagrama de arquitectura de la Deraining Net

A diferencia de la arquitectura de la Dehazing Net de la [Figura 4.8](#), se incorporaron bloques de atención convolucional (en rojo en el diagrama de la [Figura 4.13](#)) después de cada bloque convolucional tradicional. Estos bloques de atención están diseñados para que el modelo se enfoque específicamente en las áreas de la imagen donde aparecen las gotas de lluvia, que a menudo son pequeñas y localizadas. A diferencia de la neblina, que afecta de manera uniforme y generalizada a toda la imagen, las gotas de lluvia crean distorsiones localizadas que requieren una atención especial. Esta atención enfocada permite al modelo aprender a identificar y manejar de forma efectiva estas características específicas de las imágenes con lluvia, lo que mejora de manera significativa la calidad de la imagen restaurada.

Los bloques de atención de la red neuronal están diseñados para concentrar el aprendizaje del modelo en áreas específicas de las imágenes. Estos bloques reciben la salida de las capas convolucionales y, a través de una secuencia de operaciones, generan un *mapa de atención*. Este mapa es una matriz en la que cada píxel tiene un valor entre 0 y 1. Los valores cercanos a 1 indican que el modelo debe prestar más atención a esa área específica, mientras que los valores cercanos a 0 sugieren que esa región es menos relevante para la tarea en cuestión.

La arquitectura de los bloques de atención se observa en la [Figura 4.14](#). La lógica detrás es muy similar a la que se utiliza en los otros bloques de la red, pero la gran diferencia está en la función sigmoide final. Esta función es una operación matemática que convierte valores de entrada en un rango entre 0 y 1.

Este mecanismo de atención ayuda a mejorar la precisión del modelo al permitirle concentrarse en los aspectos más importantes y desafiantes de la imagen, como las gotas de lluvia en el caso del modelo *deraining*. Esta capacidad de enfocar de manera selectiva la atención no solo mejora la eficacia del modelo, sino que también optimiza el proceso de aprendizaje al dirigir los recursos computacionales hacia las áreas que realmente lo necesitan.



Figura 4.14: Diagrama de arquitectura de los bloques de atención

A modo de recapitulación, la arquitectura UNet en el modelo de eliminación de lluvia se ha enriquecido con bloques de atención convolucional para abordar específicamente las áreas afectadas por las gotas de lluvia. La incorporación de estos bloques permite al modelo concentrar su capacidad de procesamiento en regiones clave de la imagen, asegurando que se preste más atención donde es más necesario. Esta estructura es crucial para manejar de forma eficaz las distorsiones causadas por la lluvia, lo que facilita una reconstrucción más detallada de la escena visual.

En la [Figura 4.15](#) se puede ver el proceso por el que pasa una imagen que atraviesa la red. De la misma manera que se realizó la [Figura 4.11](#) se promedió cada uno de los canales a un solo canal en escala de grises para visualizarlos.

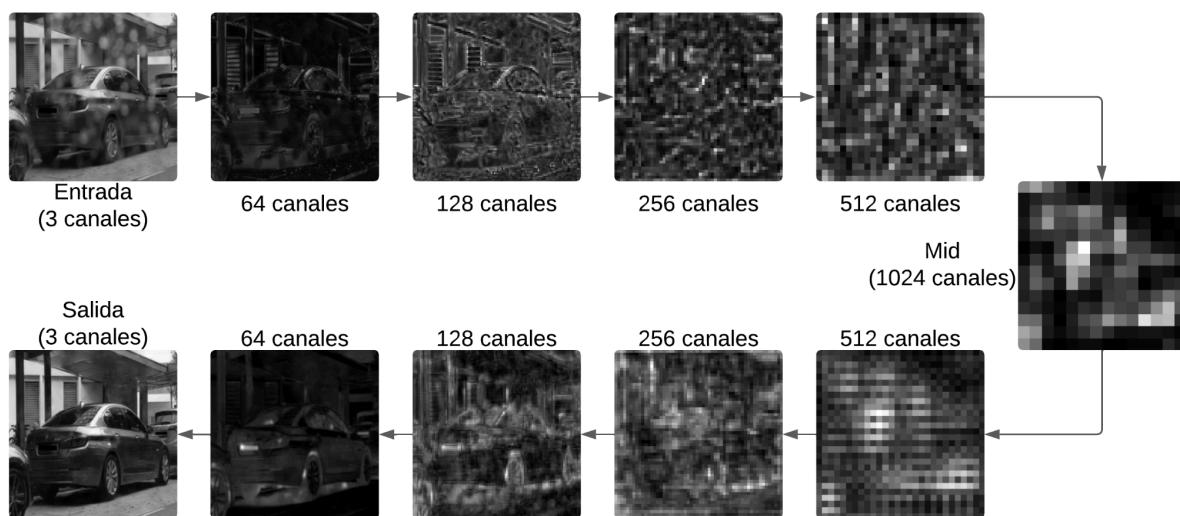


Figura 4.15: Visualización de una imagen a través de la red deraining

4.1.2.5.3 Data loader del modelo

Se creó un *data loader* enfocado en cargar las imágenes de acuerdo con la estructura base del conjunto de datos Raindrop. Este conjunto de datos está diseñado específicamente para proporcionar pares de imágenes, donde cada par consiste en una imagen afectada por la lluvia y su correspondiente imagen limpia, sin lluvia, similar al ejemplo de la [Figura 4.12](#). Esta estructura de pares es crucial porque permite que el modelo de aprendizaje automático aprenda a diferenciar y reparar las distorsiones específicas causadas por las gotas de lluvia en la lente.

El *data loader* manipula estas imágenes desde su almacenamiento, aplicando transformaciones predefinidas como redimensionar a un tamaño estándar y convertir las imágenes a tensores, lo que facilita el procesamiento por parte de redes neuronales. Este proceso asegura que el modelo tenga acceso directo y

sistemático a ejemplos claros de antes y después de la intervención de la lluvia, lo que optimiza el entrenamiento del modelo para restaurar con precisión la visibilidad y la claridad en condiciones meteorológicas adversas.

4.1.2.5.4 Entrenamiento del modelo

Para el entrenamiento, utilizando la misma metodología del modelo enfocado en la neblina, se dividió el conjunto de datos en una proporción del 80 % para entrenamiento y el 20 % para validación, como se explica en la [sección 4.1.2.4.4](#).

En el entrenamiento del modelo de eliminación de lluvia, a diferencia del enfoque que se utiliza para el *dehazing*, se optó por una función de pérdida más simplificada para centrarse en la fidelidad visual directa entre la imagen generada y la imagen clara. La función de pérdida principal que se usa es la pérdida de reconstrucción directa (*L1 loss*), que mide la diferencia absoluta promedio entre los píxeles correspondientes de las dos imágenes. Esta métrica es efectiva para asegurar que los detalles visuales y la textura de la imagen generada se alineen estrechamente con la imagen original sin lluvia.

Además de la pérdida L1, se incorpora un componente de pérdida perceptual. Esta parte de la función de pérdida utiliza un módulo perceptual, que típicamente evalúa cómo las características de alto nivel percibidas por un modelo de percepción, como una red preentrenada, difieren entre la imagen generada y la imagen objetivo. Este aspecto de la pérdida ayuda a asegurar que las cualidades estéticas y el contexto general de las imágenes se mantengan, por ejemplo, la identificación de objetos como vehículos. Esto es crucial para aplicaciones en las que no solo la exactitud píxel a píxel es importante, sino también la apariencia global y la coherencia visual. La función de pérdida como tal se define de la siguiente manera:

$$L = L1_{loss} + 0.1 * Perceptual_{loss}$$

La combinación de estas dos pérdidas proporciona un balance entre mantener la fidelidad estructural directa y preservar la integridad perceptual de las imágenes restauradas, sin complicar en exceso la función de pérdida con múltiples componentes y ponderaciones. Este enfoque simplificado facilita la interpretación de los resultados y la optimización durante el entrenamiento, centrando el aprendizaje en recuperar la claridad visual esencial en condiciones de lluvia.

4.1.2.6 Interfaz gráfica

Tras abordar en detalle la arquitectura y la función de los distintos algoritmos de procesamiento, es crucial considerar cómo estos desarrollos técnicos se presentan al usuario final para su aplicación práctica. Para garantizar que las soluciones que se desarrollaron sean accesibles y evaluables de manera efectiva, se diseñó una interfaz gráfica de usuario. Esta interfaz no solo facilita la demostración de la capacidad de los modelos para mejorar imágenes en condiciones adversas, sino que también sirve como puente entre la complejidad técnica de los algoritmos y la practicidad requerida por los usuarios finales.

Enseguida, se describe de qué forma esta interfaz permite una integración fluida y una interacción intuitiva con las tecnologías que se desarrollaron.

La interfaz está diseñada con el objetivo principal de permitir al usuario seleccionar y cargar imágenes para procesar, lo que facilita la visualización directa del impacto de las técnicas de mejora de imagen aplicadas. Esta capacidad de comparación visual es esencial para evaluar la efectividad de las soluciones propuestas en tiempo real y bajo diferentes condiciones atmosféricas como neblina, lluvia, entre otras.

Para asegurar una experiencia de usuario eficiente y centrada, el diseño de la interfaz se realizó inicialmente en Figma. Esta herramienta de diseño permitió crear un prototipo, por lo que fue posible iterar sobre el diseño y la disposición de los componentes de la interfaz antes de proceder con la implementación del código. Esto no solo optimizó el flujo de trabajo de desarrollo, sino que también aseguró que las necesidades y la facilidad de uso fueran prioritarias desde el inicio del diseño de la interfaz. El diseño final se puede observar en la [Figura 4.15](#).

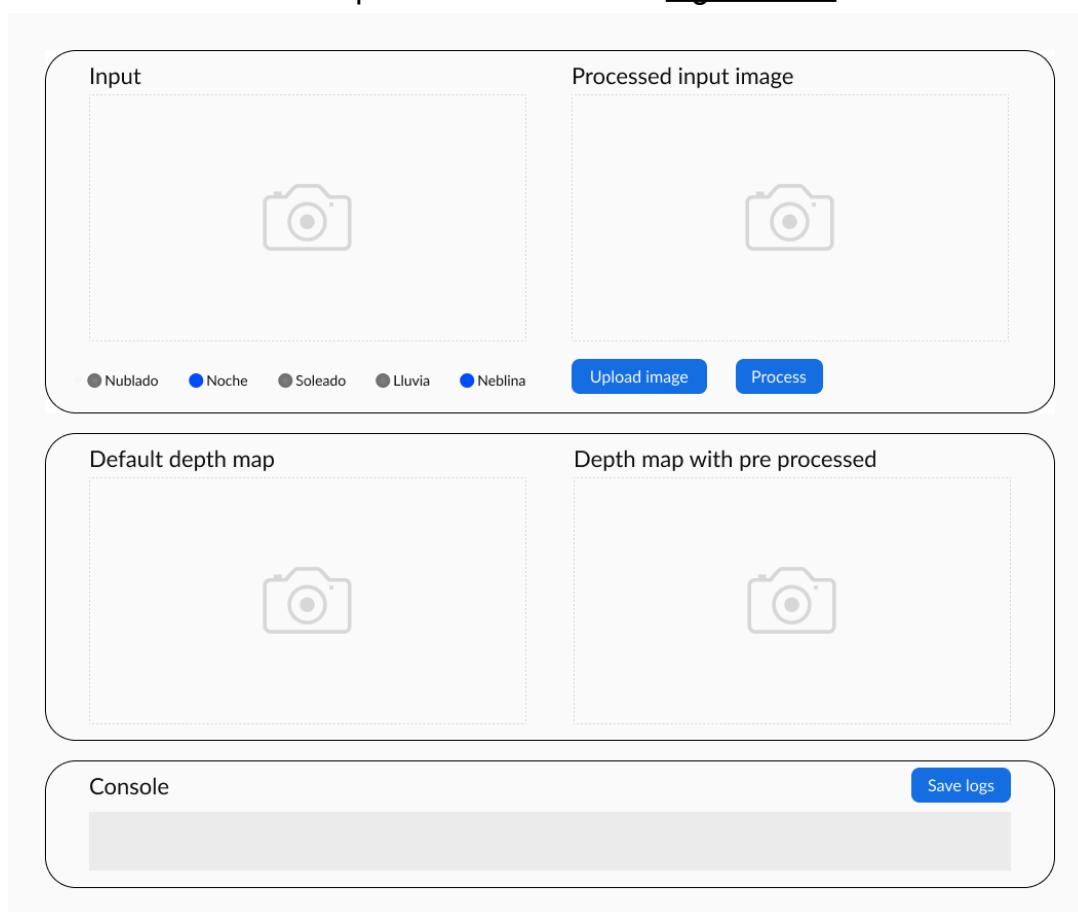


Figura 4.16: Diseño de la interfaz gráfica de usuario en Figma

4.2 Análisis de los resultados

En este apartado se realiza una evaluación del rendimiento y la eficacia de los algoritmos de mejora de imágenes desarrollados para condiciones adversas. Para asegurar la objetividad y la imparcialidad de esta evaluación, se utiliza un

conjunto de datos externo, no usado previamente en el entrenamiento de los modelos. Este conjunto de datos permite verificar la capacidad del modelo para generalizar a nuevas situaciones, una faceta crucial para su aplicación práctica en el campo de la visión computarizada.

Etiquetas	Cantidad de frames	Porcentaje respecto al total de datos (%)
Soleado	370	29.96
Noche	361	29.23
Noche y lluvia	1	0.08
Nublado	42	3.40
Nublado y lluvia	53	4.29
Lluvia	40	3.24
Lluvia y neblina	9	0.73
Neblina	356	28.83
Neblina y noche	3	0.24
Total	1235	100 %

Tabla 4.2: Cantidad total de frames por condición del conjunto de pruebas

La composición de los datos de prueba se puede observar en la [Tabla 4.2](#). Las imágenes que se utilizan son una combinación entre el conjunto de datos DAWN (Detection in Adverse Weather Nature) *dataset* [35], Real Haze Video Database [36] y videos pertenecientes a las grabaciones explicadas en la [sección 4.1.1](#) que no se utilizaron durante el entrenamiento de los distintos modelos.

Para evaluar cuantitativamente la mejora en los mapas de profundidad se utiliza el algoritmo de detección de bordes de Canny [37]. Este método permite contar la cantidad de bordes detectados en las imágenes de entrada y las procesadas, lo que proporciona una métrica objetiva sobre la mejora en la definición de bordes. La detección de bordes es especialmente útil para modelos de estimación de fondo como AdaBins, ya que la calidad de los mapas de profundidad se refleja en la claridad y precisión de los contornos de los objetos. La función que se utiliza aplica un filtro Gaussiano para suavizar la imagen y reducir el ruido, seguido de la detección de bordes con umbrales ajustables. Esto asegura que los bordes detectados sean significativos y no se vean afectados por el ruido de la imagen. Al normalizar los resultados según el tamaño de la imagen, se pueden realizar comparaciones consistentes entre diferentes conjuntos de datos y condiciones experimentales, proporcionando una evaluación robusta de la calidad del modelo AdaBins en diversas situaciones. El resultado de este algoritmo se puede observar en la [Figura 4.17](#).

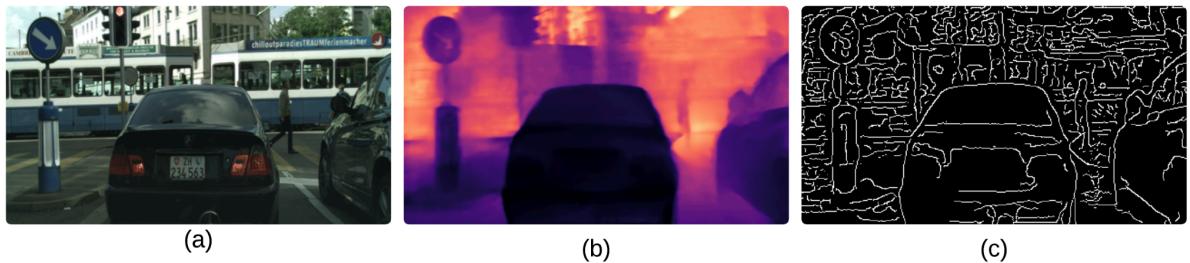


Figura 4.17: Ejemplo de algoritmo de detección de bordes: (a) imagen de entrada (b), mapa de profundidad y (c) mapa de bordes.

Este enfoque es útil para confirmar la función de los desarrollos y también para identificar las áreas específicas de mejora que pueden abordarse en iteraciones futuras. Al cuantificar objetivamente la definición de bordes en diferentes condiciones ambientales, es posible evaluar la eficacia de los algoritmos y ajustar los parámetros necesarios para optimizar su rendimiento.

4.2.1 Tagger: modelo etiquetador de imágenes

Al analizar los resultados del modelo, se adopta un enfoque metódico para evaluar su precisión. El proceso de validación consiste en etiquetar manualmente las 1235 imágenes. Después, estas etiquetas manuales se comparan con las predicciones que se generan por el modelo para identificar discrepancias. Los errores totales luego de realizar estas pruebas se pueden observar en la Tabla 4.3.

Es importante notar que el total en la cantidad de *frames* es diferente entre la Tabla 4.2 y la Tabla 4.3. Esto se debe a que en la Tabla 4.3 las categorías compuestas como *lluvia* y *nublado* se sumaron para todas las condiciones para obtener una fila por condición. Se hizo de esta manera para evaluar cada condición adversa de forma individual.

Etiquetas	Cantidad de frames	Errores totales	Porcentaje de error sobre los datos totales (%)
Neblina	368	209	16.08
Lluvia	103	45	3.46
Nublado	95	29	2.23
Soleado	370	24	1.85
Noche	364	40	3.08
Total	1300	347	26.69

Tabla 4.3: Pruebas realizadas al tagger

En la tabla 4.3 se muestra el porcentaje de error calculado para cada condición ambiental. Este porcentaje se obtiene al calcular la proporción de

predicciones incorrectas respecto al total de imágenes evaluadas. La fórmula utilizada para calcular el porcentaje de error es la siguiente:

$$\% \text{ de Error} = \left(\frac{\text{Número de predicciones incorrectas}}{\text{Total de imágenes evaluadas}} \right) \times 100$$

Este enfoque se utilizó porque el modelo debe evaluar cada predicción tanto si se esperaba que predijera la presencia de una condición adversa como la falta de esta. Este cálculo proporciona una medida clara y comprensible del rendimiento del modelo en diversas condiciones ambientales.

La evaluación de los resultados del modelo *tagger* muestra que las condiciones de neblina y lluvia presentaron los mayores desafíos en términos de precisión de clasificación. Según los datos suministrados, de los 368 *frames* categorizados bajo neblina, el modelo cometió 209 errores, lo que ocasiona un porcentaje de error del 16.08 % con respecto a la cantidad total de datos. En el caso de la lluvia, de 103 *frames*, se cometieron 45 errores, lo que representa un porcentaje de error del 3.46 %.

Con respecto al porcentaje de error total mostrado en la tabla, correspondiente a 26.69 %, cabe aclarar que este fue calculado con respecto al total de errores, en este caso, 347, con respecto al total de imágenes evaluadas.

Es importante notar que el entrenamiento del modelo se basó en casos extremos para estas dos condiciones. Esto implica que el modelo puede no ser tan sensible a condiciones menos severas de neblina o lluvia, lo que, de alguna manera, mitiga la necesidad de una detección precisa si las condiciones no son lo suficientemente graves como para justificar la aplicación de algoritmos de mejora. Un ejemplo de una imagen que el modelo no fue capaz de identificar cómo neblina es la [Imagen 4.18](#), donde se puede apreciar que, aunque sí haya una neblina notable para el ojo humano, los distintos vehículos están definidos.



Figura 4.18: Ejemplo de imagen que no fue identificada por el tagger como neblina

Por otro lado, en la [Figura 4.19](#) se puede observar una foto que fue etiquetada de manera satisfactoria por el modelo como neblina. En este caso, la neblina es mucho más notable que en la imagen anterior. No obstante, los resultados también indican una clara oportunidad de mejora.

Una estrategia efectiva es enriquecer el conjunto de entrenamiento con una gama más amplia de ejemplos que incluya variaciones moderadas de neblina y lluvia. Esto puede ayudar a ajustar mejor el modelo para detectar y actuar en un espectro más amplio de intensidades, lo que a la vez mejoraría la generalización y efectividad del modelo en situaciones prácticas. Este enfoque le permite al modelo manejar con mayor precisión diversas condiciones sin comprometer la aplicación de mejoras cuando sean realmente necesarias.



Figura 4.19: Ejemplo de imagen que fue identificada por el tagger como neblina

4.2.2 Algoritmo para el mejoramiento de imágenes nocturnas

Sin lugar a duda, el aclarado de imágenes fue el mayor reto del proyecto. La escasez de detalles en condiciones de baja luminosidad complicó significativamente el desarrollo y la optimización de los algoritmos. A pesar de estos desafíos, los resultados muestran una mejora notable en la visibilidad y calidad de las imágenes nocturnas. No obstante, se reconoce que aún queda mucho por mejorar en futuras iteraciones del proyecto para alcanzar una mayor precisión y robustez en el procesamiento de imágenes nocturnas bajo diversas condiciones adversas.

A pesar de que es el único algoritmo de la solución que no se implementó mediante una red neuronal, los resultados son bastante satisfactorios, tal y como se observa en la [Imagen 4.20](#).

Los resultados muestran mejoras significativas en la calidad y utilidad de las imágenes procesadas. El algoritmo intensifica la poca luz disponible en las imágenes originales, aumenta el contraste entre los elementos de la escena y hace visibles características que usualmente permanecen ocultas. Además, incluye un paso de reducción de ruido para suavizar las imperfecciones sin comprometer la claridad general. Aunque para el ojo humano la imagen mejorada puede parecer menos realista, este procesamiento tiene un impacto positivo significativo en la

generación de mapas de profundidad utilizando el modelo AdaBins, ya que los mapas que se generan presentan una mayor coherencia.

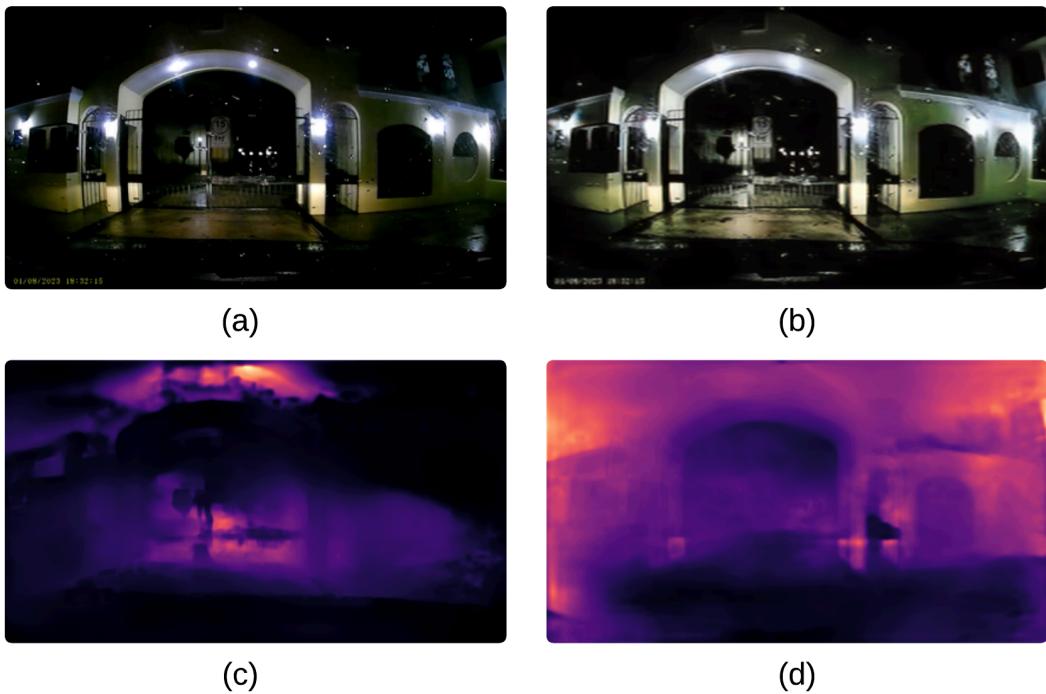


Figura 4.20: Resultados del algoritmo de mejoramiento de imágenes nocturnas donde (a) es la imagen de entrada, (b) es la imagen de entrada mejorada, (c) es el mapa de profundidad de la imagen de entrada y (d) es el mapa generado de la imagen de entrada mejorada

Para cuantificar la mejora lograda se realizó una comparación de los puntajes de detección de bordes en los mapas de profundidad que se generan antes y después del procesamiento, con el método explicado al inicio de esta sección. Los resultados se muestran en la tabla adjunta. Se calcularon los puntajes de detección de bordes para un conjunto de 50 imágenes.

Sin algoritmo de imágenes nocturnas	Con algoritmo de imágenes nocturnas
0.0551	0.0836

Tabla 4.4: Densidad media de bordes en imágenes nocturnas

Los resultados cuantitativos reflejan una mejora notable del 51.65 % en la detección de bordes en los mapas de profundidad procesados en comparación con los mapas sin procesar. Este aumento en la cantidad de bordes detectados indica que el algoritmo de procesamiento ha sido eficaz en la mejora de la visibilidad y la claridad de los elementos presentes en las imágenes. La intensificación de la poca luz disponible, combinada con la reducción de ruido y el realce de los detalles ha contribuido significativamente a esta mejora.

Estos resultados muestran cómo un algoritmo relativamente sencillo, que solo utiliza las capacidades de OpenCV, puede lograr mejoras significativas en la calidad de los mapas de profundidad que se generan. Esto demuestra que incluso

soluciones menos complejas pueden tener un impacto considerable en aplicaciones de visión por computadora, lo que facilita obtener resultados más precisos y realistas sin la necesidad de recurrir a métodos excesivamente complejos o costosos.

Para obtener estos resultados se procesaron más de 50 pruebas individuales. En la [Figura 5.2.1](#) se muestran algunos de estos casos sobre cómo el algoritmo de mejoramiento de imágenes nocturnas trabaja en distintos escenarios.

4.2.3 Dehazing: modelo eliminador de neblina

En esta sección se presentan los resultados al implementar el modelo de eliminación de neblina basado en la arquitectura de red neuronal UNet, diseñada para abordar estos desafíos. Uno de los principales retos fue encontrar datos con y sin neblina para entrenar el modelo de manera efectiva, por este motivo se optó por combinar dos conjuntos de datos, tal y como se explicó en la [sección 4.1.2.4.1](#).

El entrenamiento del modelo con datos mayoritariamente del *dataset* Foggy Cityscapes [32], el cual contiene imágenes con una neblina generada por un algoritmo, permitió que el modelo aprendiera los patrones específicos de esta neblina artificial. Si bien esto ocasionó un desempeño efectivo del modelo con las imágenes de entrenamiento y otras imágenes similares, tal y como se observa en la [Figura 4.21](#), reveló una deficiencia al aplicarse a imágenes con neblina real, como se ve en la [Figura 4.22](#). Por lo tanto, aunque el modelo trabaja bien con las imágenes del *dataset* Foggy Cityscapes, su rendimiento con imágenes completamente externas y de neblina real es muy mejorable.

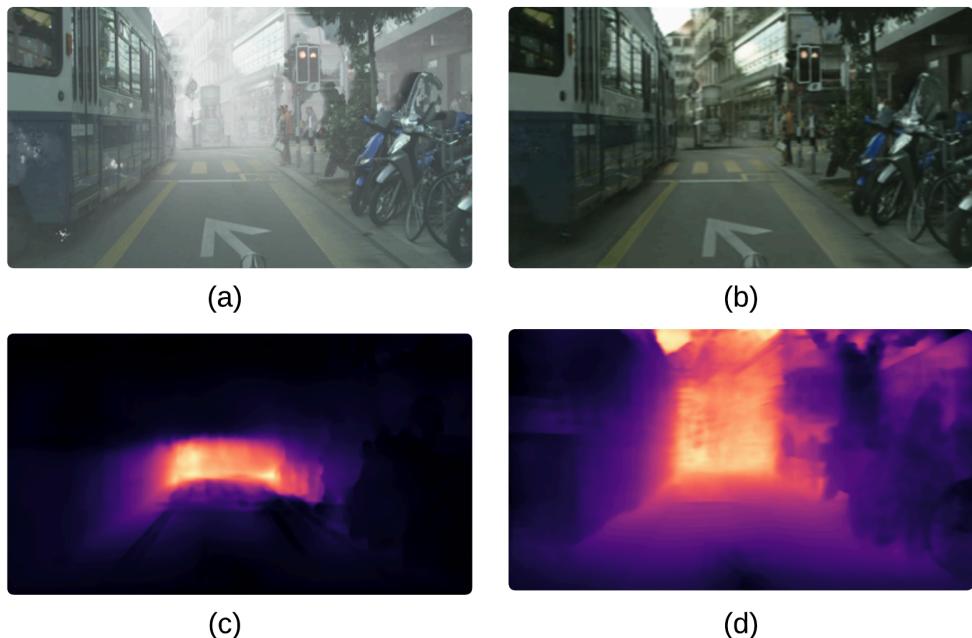


Figura 4.21: Resultados del dehazing con neblina artificial: (a) la imagen de entrada, (b) la imagen de entrada mejorada, (c) el mapa de profundidad de la imagen de entrada y (d) el mapa generado de la imagen de entrada mejorada

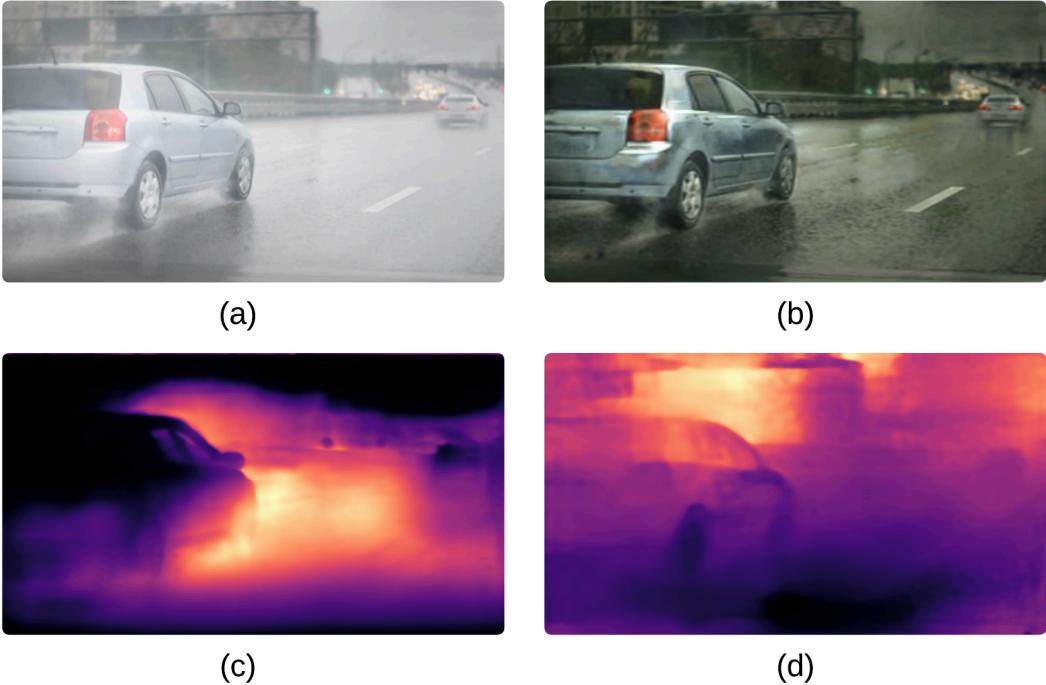


Figura 4.22: Resultados del dehazing con neblina real: (a) la imagen de entrada, (b) la imagen de entrada mejorada, (c) el mapa de profundidad de la imagen de entrada y (d) el mapa generado de la imagen de entrada mejorada

Con base en los datos presentados en la Tabla 4.5, se observa que, aunque las valoraciones cuantificadas son muy similares entre ambos *datasets* (Foggy Cityscapes [32] y DAWN [35]), los resultados de las imágenes con información real (DAWN) presentan deficiencias notables.

La puntuación presentada en la Tabla 4.5 fue obtenida promediando el resultado de la evaluación de cada una de las imágenes de prueba mediante una función de detección de bordes, tal y como se explicó en la sección 4.2. La función utilizada para este cálculo toma un mapa de profundidad como entrada y sigue los siguientes pasos: primero, se carga la imagen y se convierte a escala de grises. Luego, se aplica un filtro Gaussiano para suavizar la imagen y reducir el ruido. Posteriormente, se utiliza el algoritmo de Canny para detectar los bordes presentes en la imagen suavizada. La cantidad de píxeles de borde detectados se cuenta y se normaliza según el tamaño total de la imagen, resultando en una puntuación de detección de bordes que refleja la proporción de bordes detectados en relación con el total de píxeles de la imagen. Esta puntuación se promedia a través de todas las imágenes de prueba para obtener la puntuación final que se muestra en la tabla. Este método también será utilizado para calcular la puntuación de la Tabla 4.6.

Aunque el modelo genera buenos resultados con la información artificial del *dataset* Foggy Cityscapes, no se observa una gran mejoría al generar mapas con imágenes que contienen información real del *dataset* DAWN. Esto evidencia que el modelo necesita ser entrenado con datos que reflejen más fielmente las condiciones

reales de neblina para mejorar su robustez y generalización, dejando un claro punto de mejora para futuros trabajos.

Dataset	Procesamiento	Puntuación
Foggy CityScapes	Default	0.0664
	Dehazing Net	0.0769
DAWN	Default	0.0637
	Dehazing Net	0.0702

Tabla 4.5: Densidad media de bordes en imágenes con neblina

Para obtener estos resultados se procesaron 112 pruebas individuales. En la [Figura 5.3.1](#) se muestran algunos de estos casos para una mejor comprensión del desempeño del modelo.

4.2.4 Deraining: modelo eliminador de lluvia

En esta sección se presentan los resultados al implementar el modelo de eliminación de lluvia basado en la arquitectura UNet, diseñada para abordar los desafíos de visibilidad reducida en imágenes afectadas por gotas de lluvia. Una de las ventajas principales fue la abundancia de datos con gotas de agua reales, que se generan mediante técnicas que simulan condiciones de lluvia sobre superficies de vidrio, como fue el caso del *dataset* Raindrop [5]. Esto permitió disponer de un conjunto de datos extenso con imágenes, tanto con gotas de lluvia como sin estas, lo que facilitó el entrenamiento del modelo para identificar y eliminar eficazmente las distorsiones causadas por la lluvia. Debido a esta abundancia de datos realistas, la eliminación de lluvia resultó ser la condición adversa que obtuvo los mejores resultados en términos de precisión y efectividad.

La [Figura 4.23](#) muestra los resultados del modelo de eliminación de lluvia aplicado a una imagen del conjunto de validación del *dataset* Raindrop, que se utiliza durante el entrenamiento. Los resultados demuestran la eficacia del modelo para eliminar las gotas de lluvia y mejorar la calidad visual de las imágenes, manteniendo detalles importantes y mejorando la percepción general de la escena.

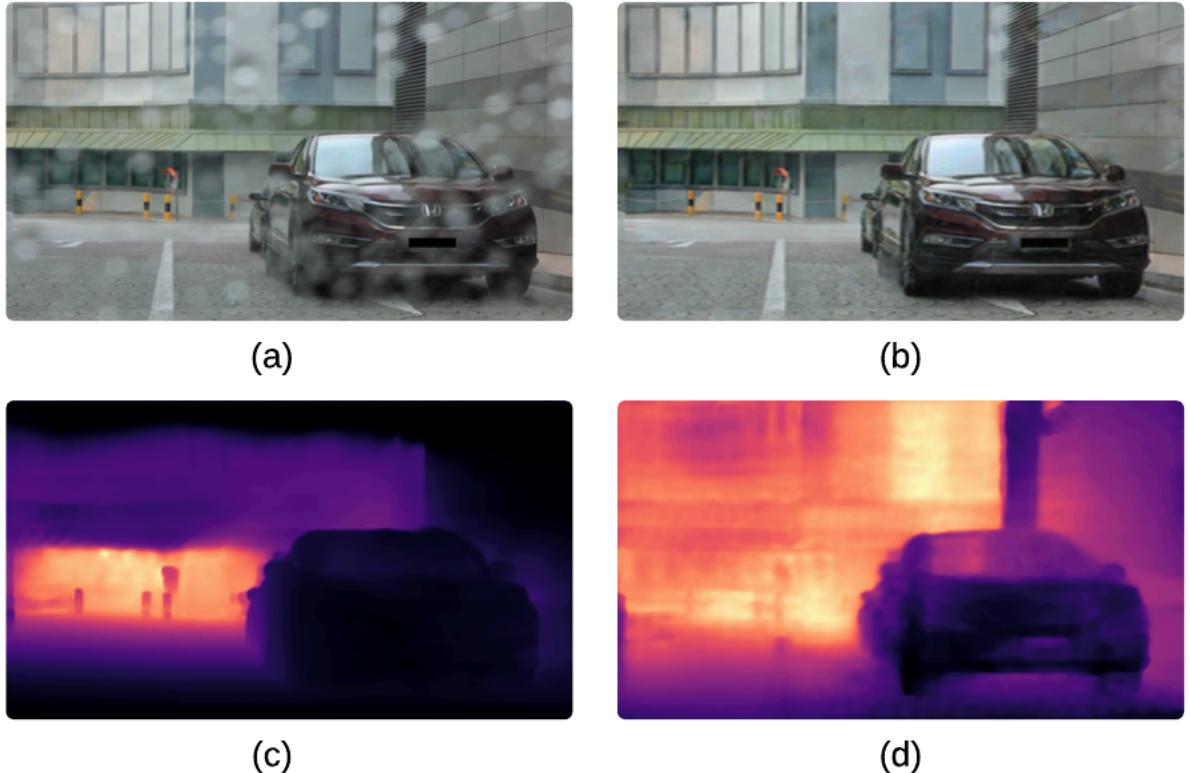


Figura 4.23: Resultados del deraining con imagen del dataset Raindrop: (a) la imagen de entrada, (b) la imagen de entrada mejorada, (c) el mapa de profundidad de la imagen de entrada y (d) el mapa generado de la imagen de entrada mejorada

Para comprobar el funcionamiento del modelo con datos completamente externos al entrenamiento se optó por el uso del *dataset* RainDS presentado por Quan *et al.* [38]. Un ejemplo de los resultados utilizando una imagen de este conjunto de datos se puede observar en la [Figura 4.24](#).

La [Tabla 4.6](#) presenta los resultados cuantitativos que se obtienen mediante la evaluación de la densidad media de bordes en imágenes con y sin el procesamiento de eliminación de lluvia. El análisis de los datos de la tabla revela que el procesamiento aplicado mejora significativamente la densidad de bordes en comparación con las imágenes originales, sobre todo en el caso del *dataset* externo RainDS, con una mejora del 17.48 %. Además, no solo existe una mejoría cuantitativa, sino que visiblemente los mapas que se generan luego del procesamiento están más definidos en el ámbito de detalles, tal y como se observa en la [Figura 4.24](#).

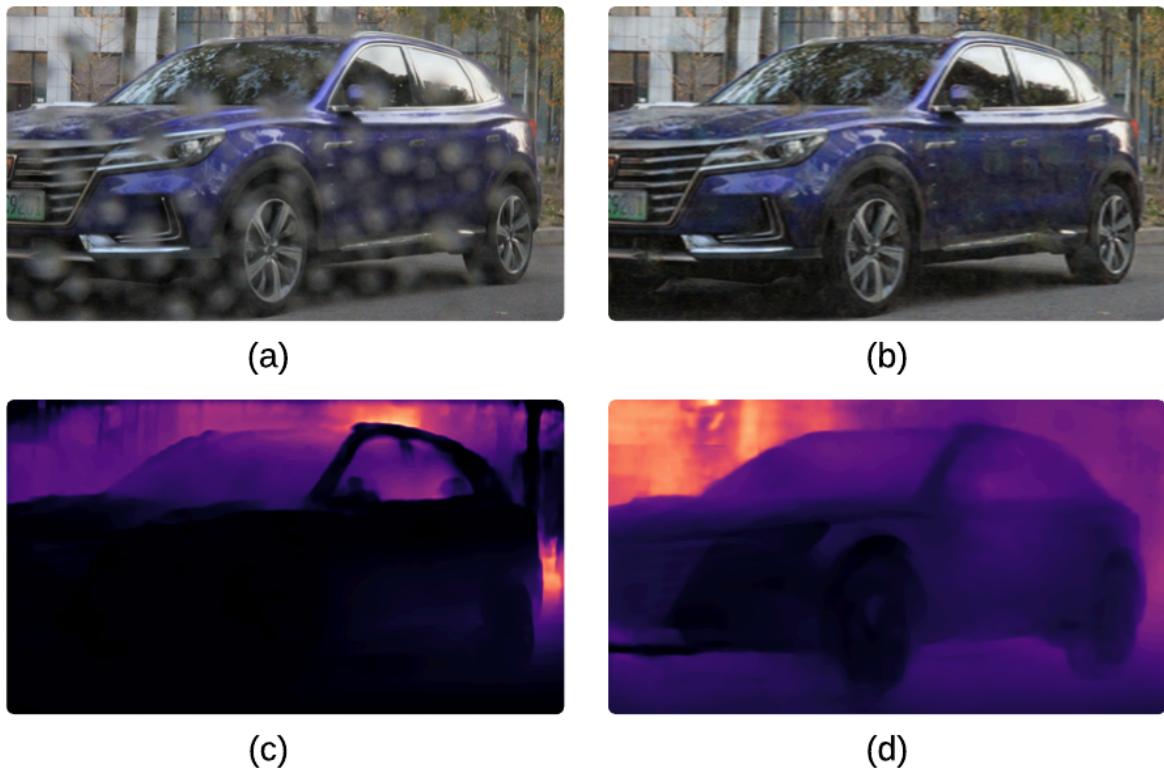


Figura 4.24: Resultados del deraining con imagen del dataset RainDs: (a) la imagen de entrada, (b) la imagen de entrada mejorada, (c) el mapa de profundidad de la imagen de entrada y (d) el mapa generado de la imagen de entrada mejorada

Estos resultados confirman que el modelo de eliminación de lluvia no solo elimina efectivamente las gotas de agua, sino que también mejora la definición de los bordes, lo que contribuye a una mayor claridad y precisión en las imágenes procesadas. Esto es esencial para aplicaciones que requieren una alta calidad visual y detalles precisos, como la generación de mapas de profundidad.

Dataset	Procesamiento	Puntuación
Raindrop	Default	0.0749
	Deraining Net	0.0788
RainDS	Default	0.0658
	Deraining Net	0.0773

Tabla 4.6: Densidad media de bordes en imágenes con lluvia

Por lo tanto, los resultados al implementar el modelo de eliminación de lluvia muestran una mejora significativa en la calidad de las imágenes procesadas. La abundancia de datos realistas permitió entrenar el modelo de manera efectiva, logrando eliminar las gotas de agua y mejorar la visibilidad en las imágenes. Estos resultados confirman que el modelo desarrollado no solo elimina las distorsiones

causadas por la lluvia, sino que también mejora la claridad de los detalles en las imágenes.

Para obtener estos resultados se procesaron 104 pruebas individuales. En la [Figura 5.4.1](#) se muestran algunos de estos casos para una mejor comprensión del desempeño del modelo.

4.2.5 Interfaz gráfica

La interfaz gráfica de usuario para la aplicación se desarrolló utilizando Python, manteniendo la coherencia en el lenguaje de programación a lo largo de todo el proyecto. Se optó por Tkinter como herramienta para crear esta interfaz debido a su integración nativa con Python, lo que facilita la interacción directa con los algoritmos de procesamiento de imagen que también se implementan en este lenguaje. El resultado de la interfaz a nivel de código se puede observar en la [Figura 4.25](#).

Además, la interfaz incluye una lógica de integración que permite a cada algoritmo de procesamiento de imagen operar dentro de un marco de *un solo clic*. Esto hace que la aplicación sea accesible y funcional, ya que combina todos los algoritmos en un sistema unificado. Esta integración convierte la interfaz en una herramienta amigable, que amarra todas las funcionalidades en un solo programa, lo que da la posibilidad de una manipulación intuitiva y resultados inmediatos para la persona usuaria.

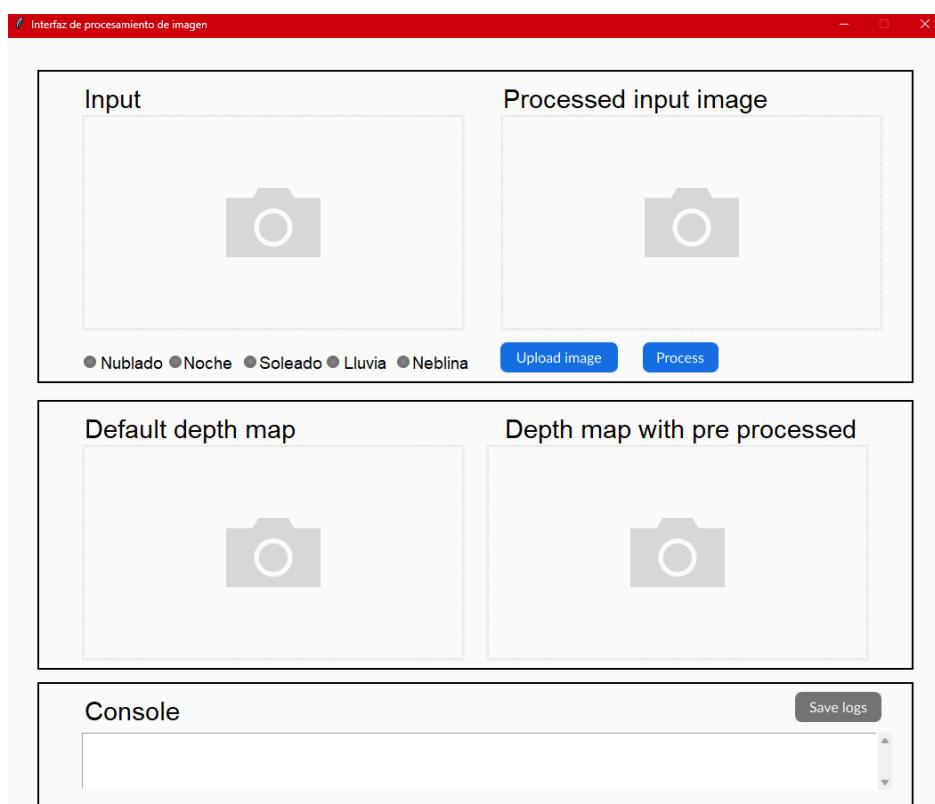


Figura 4.25: Interfaz gráfica de usuario desarrollada en Tkinter

El proceso por seguir se basa en que una vez que un usuario selecciona y carga una imagen en la interfaz, automáticamente se etiqueta la imagen con condiciones adversas detectadas, como lluvia o neblina. Sin embargo, la persona usuaria tiene la opción de ajustar estas etiquetas de forma manual si así lo desea, al hacer clic sobre el *LED* de la condición. Al proceder con el botón *Procesar*, el sistema aplica el algoritmo de mejora de imagen adecuado para la condición que se seleccionó siguiendo el flujo de la [Figura 1.2](#).

Una vez procesada la imagen, la interfaz muestra las cuatro imágenes más significativas para el usuario, las cuales son:

- **Imagen de entrada original:** se muestra la imagen cargada sin ninguna modificación, lo que le permite al usuario ver el estado inicial del entorno capturado.
- **Imagen de entrada procesada:** aquí se visualiza el resultado después de aplicar el algoritmo de mejora, ajustado para contrarrestar la condición adversa específica identificada o que se seleccionó. Esto muestra mejoras visibles en claridad y detalle.
- **Mapa de profundidad a la imagen de entrada original:** se presenta un mapa de profundidad generado a partir de la imagen original, lo que ofrece una representación visual de la distribución espacial de los objetos en la escena sin procesamiento.
- **Mapa de profundidad a la imagen de entrada procesada:** comparativamente, se muestra el mapa de profundidad correspondiente a la imagen procesada, lo cual es crucial para evaluar cómo las mejoras en la imagen afectan la percepción de profundidad y la precisión del modelo en contextos de visión computarizada.

En la [Figura 4.26](#) se observa la distribución que se mencionó. Esta disposición de la interfaz no solo proporciona una herramienta intuitiva y fácil de usar para el procesamiento de imágenes bajo diversas condiciones atmosféricas, sino que también permite evaluar de manera inmediata y efectiva los resultados del procesamiento.

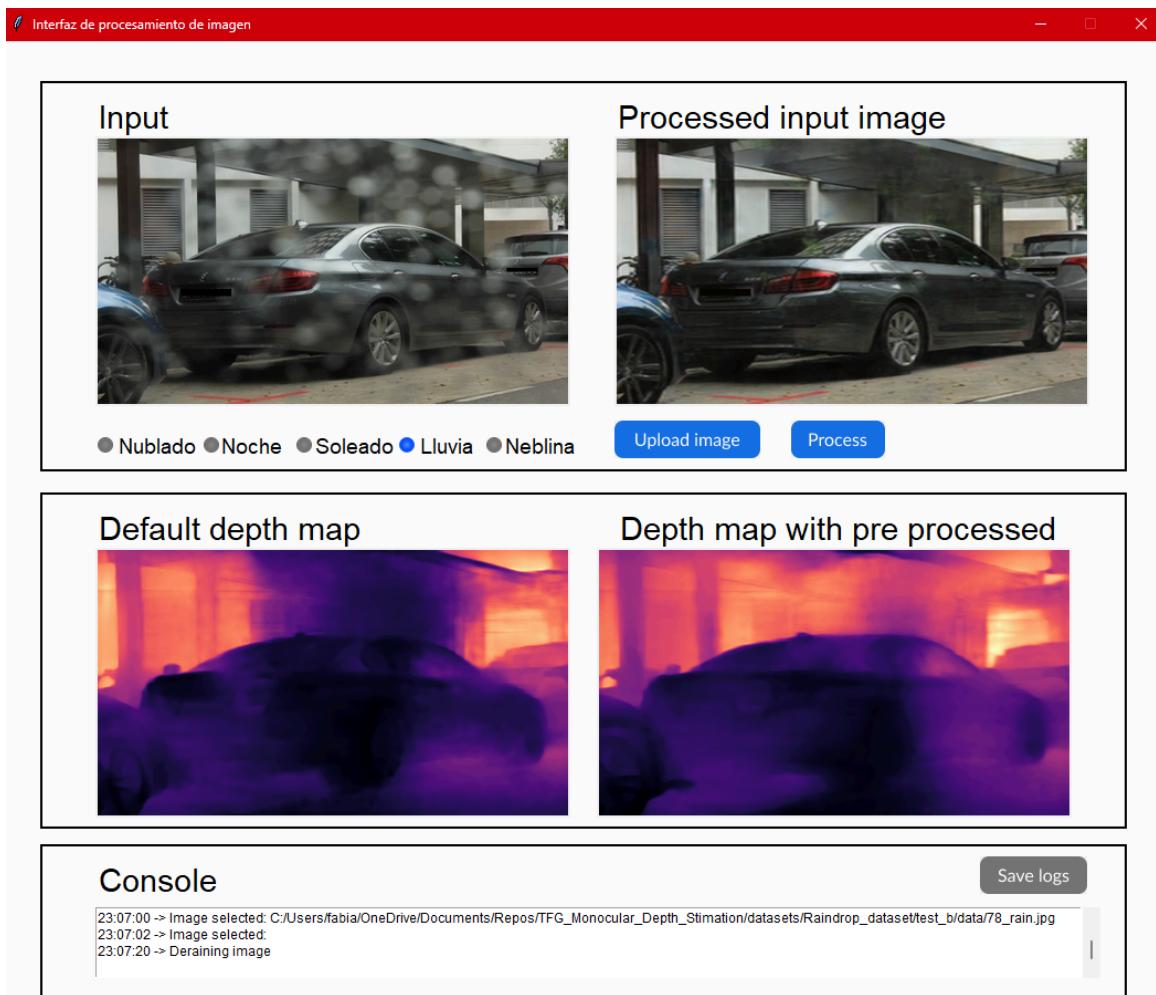


Figura 4.26: Interfaz gráfica de usuario después de procesar la imagen

Capítulo 5: Conclusiones

En este proyecto se ha demostrado que la calidad de los datos es crucial para obtener resultados precisos y útiles en trabajos de inteligencia artificial. Aunque la motivación inicial era la disponibilidad de una gran cantidad de datos, se evidenció que la búsqueda e integración de diversos conjuntos de datos externos era necesaria para complementar y enriquecer la información inicial. Esto permitió mejorar significativamente la calidad del entrenamiento y, en consecuencia, los resultados obtenidos. Este enfoque permitió cumplir el primer objetivo específico de diseñar algoritmos de mejora de imagen, optimizando la calidad de las imágenes en condiciones adversas.

Se diseñaron y utilizaron los algoritmos de etiquetado de imágenes, el algoritmo para el mejoramiento de imágenes nocturnas, el modelo eliminador de neblina y el modelo eliminador de lluvia, que demostraron ser efectivos para mejorar la calidad de las imágenes en diferentes condiciones ambientales. En el capítulo 4, se observó una mejora significativa en la definición de los bordes y en la precisión de los mapas de profundidad generados por estos algoritmos, tal como se refleja en los resultados de las pruebas de detección de bordes utilizando el algoritmo de Canny. Además, los modelos demostraron una capacidad robusta para generalizar a nuevas situaciones, evaluadas mediante un conjunto de datos externos no utilizados en el entrenamiento, confirmando su eficacia en diversas condiciones ambientales adversas.

Además de la importancia de la calidad de los datos, el diseño modular adoptado resultó fundamental para abordar eficazmente el problema presentado. La creación de componentes independientes permitió enfrentar cada una de las condiciones adversas de manera específica y eficiente. Después, la integración de estos módulos conformó una solución integral, mejorando la flexibilidad y la escalabilidad del sistema. Esta estrategia facilita no solo el desarrollo y la implementación de cada componente, sino también la posibilidad de realizar ajustes y mejoras continuas sin afectar el funcionamiento general del sistema. Este proceso cumple con el segundo objetivo específico de integrar los algoritmos de mejora de imagen con la estimación de fondo monocular. Los resultados expuestos en la sección 4.2 demuestran que cada módulo fue efectivo en su tarea específica.

Para evaluar el impacto de estas estrategias, la cuantificación de los resultados se destacó como una práctica crucial. En el ámbito de la visión por computadora, muchas evaluaciones se realizan de manera subjetiva con base en la opinión del programador, lo que puede llevar a interpretaciones erróneas. Por ende, establecer métodos de cuantificación estandarizados permitió valorar objetivamente el rendimiento del sistema y hacer ajustes informados para mejorar su eficacia. La validación del desempeño y la eficacia del sistema desarrollado en escenarios, tanto óptimos como adversos, permitió cumplir el tercer objetivo específico. A continuación, se enumeran las mejoras obtenidas para las diferentes condiciones adversas:

- La densidad media de bordes en imágenes nocturnas mejoró un 51.65% después del preprocesamiento, lo que refleja una mejora tangible y medible en la calidad de las imágenes procesadas.
- El modelo de eliminación de neblina mostró una mejora del 34% en la densidad de bordes en condiciones de neblina artificial. Sin embargo, se observó un rendimiento más modesto con datos reales, lo que sugiere la necesidad de un entrenamiento más específico y datos adicionales para mejorar la precisión en escenarios reales.
- El modelo para la eliminación de lluvia demostró una mejora del 39.45% en la densidad de bordes, validando su eficacia en condiciones adversas de lluvia.

Los resultados mostraron una notable mejora en la calidad de los mapas de profundidad generados, especialmente en condiciones adversas. Esto validó la hipótesis de que un preprocesamiento inteligente y específico antes de ingresar los datos a la red generadora es beneficioso frente a los resultados. Implementar técnicas de preprocesamiento que aborden problemas como la baja iluminación, la presencia de neblina y lluvia optimizó el rendimiento de la red, obteniendo mapas de profundidad más precisos y detallados. Esto no solo logró cumplir con el objetivo general del proyecto, sino que también resaltó la importancia de las técnicas avanzadas de procesamiento de imágenes y la inteligencia artificial en la mejora de la precisión del proceso en entornos desafiantes.

5.1 Recomendaciones

Para investigaciones y desarrollos futuros, se recomienda explorar la posibilidad de entrenar los modelos de manera no supervisada, especialmente en escenarios donde obtener datos de referencia confiables se vuelve complicado, como en condiciones de neblina o nocturnas. Los enfoques no supervisados pueden ofrecer soluciones más flexibles y eficientes en estos contextos, lo que permite un avance significativo en la precisión y aplicabilidad de los sistemas de visión por computadora.

De manera ambiciosa, se sugiere la creación de un modelo generador de mapas de profundidad que no solo genera mapas precisos, sino que también incluye la identificación y procesamiento de condiciones adversas. Este enfoque puede mejorar la calidad y la utilidad de los mapas de profundidad en aplicaciones prácticas, adaptándose dinámicamente a diversas condiciones ambientales y garantizando un rendimiento óptimo.

Asimismo, se recomienda realizar una investigación sobre cómo evaluar de manera óptima un generador de mapas de profundidad en situaciones donde no se cuenta con mapas de referencia para comparar. Desarrollar metodologías de evaluación robustas y eficientes es crucial para garantizar la fiabilidad y la precisión de los generadores de mapas de profundidad, especialmente en entornos donde la recopilación de datos de referencia es difícil o inviable. Estas investigaciones pueden proporcionar nuevas perspectivas y técnicas para la evaluación objetiva y

detallada de los modelos, asegurando su rendimiento y confiabilidad en diversos escenarios.

Las recomendaciones propuestas están diseñadas para mejorar las técnicas actuales y ampliar el conocimiento en el campo de la visión por computadora. Al enfrentar los desafíos de la obtención de datos y la evaluación de modelos, es posible desarrollar sistemas más robustos y adaptables que ofrezcan soluciones innovadoras y efectivas en condiciones adversas.

Apéndices

Apéndices 1: implicaciones ambientales y de desarrollo sostenible

Este apéndice tiene como objetivo proporcionar una reflexión objetiva y con amplia proyección respecto al entorno sobre las implicaciones e impactos actuales y potenciales que se relacionan con aspectos sociales, económicos, ambientales, legales, de seguridad y salud que implica el proyecto de estimación de fondo monocular en condiciones no ideales. A través de un análisis, se busca destacar cómo este proyecto de ingeniería contribuye a la sociedad en general y en especial para la práctica de la ingeniería y el ejercicio profesional del ingeniero en computadores. En este análisis, se hace una especial mención a de qué forma el proyecto se alinea con varios Objetivos de Desarrollo Sostenible de las Naciones Unidas. [39]

1. Aspectos sociales

- Identificación y análisis

El proyecto de estimación de fondo monocular en condiciones no ideales tiene un impacto significativo en la sociedad al mejorar la seguridad y eficiencia de sistemas autónomos, como vehículos autónomos y sistemas de vigilancia. La mejora en la percepción visual en condiciones adversas contribuye a reducir accidentes y optimizar la seguridad pública.

- Relación con los Objetivos de Desarrollo Sostenible de las Naciones Unidas:

Este proyecto se alinea con el objetivo de desarrollo sostenible 11: "Ciudades y comunidades sostenibles", al contribuir a la creación de sistemas de transporte más seguros y eficientes. Además, se relaciona con el objetivo n.º 9: "Industria, innovación e infraestructura", al promover el desarrollo de tecnologías innovadoras.

- Proyecciones futuras:

En el futuro, se espera que este proyecto pueda integrarse en más aplicaciones de seguridad pública y transporte, mejorando la calidad de vida de las personas al proporcionar sistemas más seguros y confiables. La tecnología también se extiende a otras áreas como la asistencia médica y la agricultura, lo que promueve un desarrollo sostenible en diversas industrias.

2. Aspectos económicos

- Identificación y análisis:

El desarrollo de algoritmos avanzados para la estimación de fondo monocular puede reducir costos asociados con accidentes y fallos en sistemas autónomos. Además, al mejorar la eficiencia de estos sistemas, se optimizan recursos y se reducen gastos operativos.

- Relación con los Objetivos de Desarrollo Sostenible de las Naciones Unidas:

El proyecto apoya el objetivo n.º 8: "Trabajo decente y crecimiento económico", al fomentar la innovación tecnológica que impulsa la productividad y el desarrollo económico. Además, contribuye al objetivo n.º 12: "Producción y consumo responsables", al optimizar el uso de recursos en diversas aplicaciones.

- Proyecciones futuras:

La implementación de esta tecnología atrae inversiones y genera empleo en el sector tecnológico, lo que promueve el crecimiento económico. A largo plazo, se espera que la optimización de recursos y la reducción de costos operativos beneficien a múltiples industrias y promueva un desarrollo económico sostenible.

3. Aspectos legales

- Identificación y análisis:

El uso de tecnologías avanzadas en sistemas autónomos requiere actualizar y adaptar la legislación existente para garantizar la seguridad y privacidad de los datos. Este proyecto debe cumplir con regulaciones específicas de seguridad y protección de datos.

- Relación con los Objetivos de Desarrollo Sostenible de las Naciones Unidas:

Este aspecto se relaciona con el objetivo n.º 16: “Paz, justicia e instituciones sólidas”, al promover la creación de marcos legales robustos que apoyan la innovación tecnológica mientras protegen los derechos de los individuos.

- Proyecciones futuras:

Es probable que se desarrollen nuevas regulaciones y normativas específicas para el uso de inteligencia artificial en sistemas autónomos. Estas regulaciones deben equilibrar la promoción de la innovación con la protección de la seguridad y los derechos de los usuarios.

4. Aspectos ambientales

- Identificación y análisis:

El proyecto contribuye a reducir la contaminación y el consumo de energía mediante al optimizar sistemas autónomos que son más eficientes y menos dependientes de combustibles fósiles. La mejora en la percepción visual también ayuda en la monitorización y protección del ambiente.

- Relación con los Objetivos de Desarrollo Sostenible de las Naciones Unidas:

Este proyecto apoya el objetivo n.º 13: “Acción por el clima”, al promover tecnologías que reducen emisiones y mejoran la eficiencia energética. Además, se relaciona con el objetivo n.º 15: “Vida de ecosistemas terrestres”, al facilitar la monitorización y protección del ambiente.

- Proyecciones futuras:

Se espera que las tecnologías que se desarrollaron puedan integrarse en sistemas de monitorización ambiental y gestión de recursos naturales, contribuyendo a proteger el ambiente y a la lucha contra el cambio climático.

5. Aspectos de seguridad y salud.

- Identificación y análisis:

La mejora en la percepción visual en condiciones adversas tiene un impacto directo en la seguridad de los sistemas autónomos, lo que reduce el riesgo de accidentes. Además, estas mejoras pueden aplicarse en el campo de la salud, por ejemplo, en sistemas de diagnóstico por imagen.

- Relación con los Objetivos de Desarrollo Sostenible de las Naciones Unidas:

El proyecto se alinea con el objetivo n.º 3: “Salud y bienestar”, al mejorar tecnologías que pueden utilizarse en el diagnóstico y tratamiento médico. Además,

contribuye al objetivo n.º 9: “Industria, innovación e infraestructura”, al desarrollar tecnologías que mejoran la seguridad y eficiencia de infraestructuras críticas.

- Proyecciones futuras:

En el futuro, se anticipa que esta tecnología mejorada se utilice en diversas aplicaciones de seguridad y salud, lo que proporciona beneficios significativos a la sociedad. La integración de estos avances en sistemas médicos y de transporte puede mejorar la seguridad y el bienestar general de las personas.

Apéndices 2: algunos ejemplos de procesamiento de imágenes nocturnas



Figura 5.2.1: Algunos ejemplos del procesamiento nocturno: (a) la imagen de entrada, (b) la imagen de entrada mejorada, (c) el mapa de profundidad de la imagen de entrada y (d) el mapa generado de la imagen de entrada mejorada

Apéndices 3: algunos ejemplos de procesamiento de imágenes con neblina



Figura 5.3.1: Algunos ejemplos de procesamiento con la dehazing Net: (a) la imagen de entrada, (b) la imagen de entrada mejorada, (c) el mapa de profundidad de la imagen de entrada y (d) el mapa generado de la imagen de entrada mejorada

Apéndices 4: algunos ejemplos de procesamiento de imágenes con lluvia



Figura 5.4.1: Algunos ejemplos de procesamiento con la deraining Net: (a) es la imagen de entrada, (b) es la imagen de entrada mejorada, (c) es el mapa de profundidad de la imagen de entrada y (d) es el mapa generado de la imagen de entrada mejorada

Bibliografía

- [1] R. Szeliski, “12.8 Monocular depth estimation”, in Computer Vision: Algorithms and Applications, 2nd ed., Manhattan, NY: Springer, 2022, pp. 796–799.
- [2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding”, in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016.
- [3] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, “Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos”, Proc. AAAI Conf. Artificial Intelligence, vol. 33, no. 1, pp. 8001–8008, Jul. 2019, doi: 10.1609/aaai.v33i01.33018001.
- [4] T. Yu *et al.*, “Joint self-supervised enhancement and denoising of low-light images”, IEEE Trans. Emerging Topics in Computational Intelligence, pp. 1–14, 2024, doi: 10.1109/tetci.2024.3358200.
- [5] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, “Attentive generative adversarial network for raindrop removal from a single image”, in 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition, Jun. 2018, doi: 10.1109/cvpr.2018.00263.
- [6] M. Bijelic *et al.*, “Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather”, in 2020 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), Jun. 2020, doi: 10.1109/cvpr42600.2020.01170.
- [7] S. Farooq Bhat, I. Alhashim, and P. Wonka, “Adabins: Depth estimation using adaptive bins”, in 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), Nov. 2020, doi: 10.1109/cvpr46437.2021.00400.
- [8] Nvidia Corporation, “Cuda Toolkit Documentation 12.4”, CUDA Toolkit Documentation 12.4, <https://docs.nvidia.com/cuda/index.html> (accessed Apr. 1, 2024).
- [9] Pytorch Org, “PYTORCH documentation”, PyTorch documentation - PyTorch 2.2 documentation, <https://pytorch.org/docs/stable/index.html> (accessed Apr. 1, 2024).
- [10] R. Szeliski, “1.1 What is computer vision?”, in Computer Vision: Algorithms and Applications, 2nd ed., Manhattan, NY: Springer, 2022, pp. 3–10.
- [11] R. Szeliski, “Chapter 6. Recognition”, in Computer Vision: Algorithms and Applications, 2nd ed., Manhattan, NY: Springer, 2022, pp. 343–413.
- [12] P. Kreowsky and B. Stabernack, “A full-featured FPGA-based pipelined architecture for SIFT extraction”, IEEE Access, vol. 9, pp. 128564–128573, 2021, doi: 10.1109/access.2021.3104387.
- [13] C. Georgoulas, G. Ch., and I. Andreadis, “Real-time stereo vision applications”, Robot Vision, Mar. 2010, doi: 10.5772/9288.
- [14] D. Edwards, “¿Qué es el lidar?”, IBM, <https://www.ibm.com/mx-es/topics/lidar> (accessed Apr. 1, 2024).
- [15] A. Pacala, “How multi-beam flash lidar works”, Ouster, <https://ouster.com/insights/blog/how-multi-beam-flash-lidar-works> (accessed Apr. 1, 2024).

- [16] R. C. González and R. E. Woods, Digital Image Processing, 4th ed., Upper Saddle River: Pearson, 2019.
- [17] The scikit-image contributors, “Image analysis in Python”, Image filtering - Image analysis in Python,
https://scikit-image.org skimage-tutorials/lectures/1_image_filters.html (accessed Apr. 2, 2024).
- [18] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Frechen: MITP, 2018.
- [19] R. Szeliski, Computer Vision: Algorithms and Applications. Cham: Springer Nature, 2022.
- [20] P. Larrañaga and A. Moujahid, “Tema 8. Redes Neuronales”, Departamento de Ciencias de la Computación e Inteligencia Artificial,
<http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf> (accessed Apr. 3, 2024).
- [21] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, Nature, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [22] A. Biswal, “Top 10 deep learning algorithms you should know in 2024”, Simplilearn.com,
<https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm> (accessed Apr. 2, 2024).
- [23] O. Ronneberger, “U-net convolutional networks for biomedical image segmentation”, Informatik aktuell, pp. 3–3, Nov. 2017, doi: 10.1007/978-3-662-54345-0_3.
- [24] I. Goodfellow *et al.*, “Generative Adversarial Networks”, Commun. ACM, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.
- [25] Scrum Org, “What is Scrum?”, Scrum.org,
<https://www.scrum.org/learning-series/what-is-scrum/> (accessed Apr. 14, 2024).
- [26] S. Laoyan, “Qué es la Metodología waterfall y cuándo utilizarla [2024]”, Asana,
<https://asana.com/es/resources/waterfall-project-management-methodology> (accessed Apr. 14, 2024).
- [27] Steren, “Cámara Para Auto Con Grabador full HD Y WDR”, Electrónica Steren Costa Rica, <https://www.steren.cr/camara-para-auto-con-grabador-full-hd-y-wdr.html> (accessed Apr. 29, 2024).
- [28] Pytorch docs, “Resnet152”, resnet152 - Torchvision main documentation,
<https://pytorch.org/vision/main/models/generated/torchvision.models.resnet152.html> (accessed May 1, 2024).
- [29] Pytorch docs, “BCEWITHLOGITSLOSS”, BCEWithLogitsLoss - PyTorch 2.3 documentation,
<https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html> (accessed May 2, 2024).
- [30] W. Chen, W. Wang, W. Yang, and J. Liu, “Deep Retinex Decomposition for Low-Light Enhancement”, in British Machine Vision Conf., 2018.
- [31] OpenCV, “Color conversions”, OpenCV,
https://docs.opencv.org/4.x/de/d25/imgproc_color_conversions.html (accessed May 5, 2024).

- [32] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data", Int. J. Computer Vision, vol. 126, no. 9, pp. 973–992, Mar. 2018, doi: 10.1007/s11263-018-1072-8.
- [33] C. O. Ancuti, C. Ancuti, R. Timofte, and C. De Vleeschouwer, "O-haze: A Dehazing benchmark with real hazy and haze-free outdoor images", in 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops (CVPRW), Jun. 2018, doi: 10.1109/cvprw.2018.00119.
- [34] A. Hilbert *et al.*, "Brave-net: Fully automated arterial brain vessel segmentation in patients with cerebrovascular disease", Frontiers Artificial Intelligence, vol. 3, Sep. 2020, doi: 10.3389/frai.2020.552258.
- [35] M. KENK, "DAWN", Mendeley Data, V3, 2020, doi: 10.17632/766ygrbt8y.3.
- [36] Y. Chu, G. Luo, and F. Chen, "A Real Haze Video Database for Haze Level Evaluation", in Proc. Thirteenth Int. Conf. Quality of Multimedia Experience, Jun. 2021.
- [37] GeeksforGeeks, "Python opencv - canny() function", GeeksforGeeks, <https://www.geeksforgeeks.org/python-opencv-canny-function/> (accessed May 16, 2024).
- [38] R. Quan, X. Yu, Y. Liang, and Y. Yang, "Removing raindrops and rain streaks in one go", in 2021 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR), Jun. 2021, doi: 10.1109/cvpr46437.2021.00903.
- [39] Naciones Unidas, "Objetivos y Metas de Desarrollo Sostenible - Desarrollo Sostenible", United Nations, <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/> (accessed May 20, 2024).
- [40] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. doi:10.1109/cvpr.2015.7298925.
- [41] E. editorial de IONOS, "Modelo en espiral: El Modelo para la Gestión de Riesgos en el Desarrollo de software," IONOS Startup Guide, 2023. [Online]. Available: <https://www.ionos.es/startupguide/productividad/modelo-en-espiral/>. Accessed: June 9, 2024.
- [42] J. Xing, L. Chu, C. Guo, S. Pu, and Z. Hou, "Dual-Input and Multi-Channel Convolutional Neural Network Model for Vehicle Speed Prediction," *Sensors*, vol. 21, no. 21, pp. 7767, Nov. 2021, doi: 10.3390/s21227767.
- [43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- [44] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, 2015, doi: 10.1007/s11263-015-0816-y.