UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

TESIS DOCTORAL

Desarrollo de un método de análisis de pose facial mediante inteligencia artificial y estéreo visión con aplicación en ADAS

Autor:

Jonathan Jesus Sanchez Castro

Director de Tesis:

Julio César Rodríguez Quiñonez

Co-Director de Tesis:

Guillermo Galaviz Yañez

Tesis presentada para cumplir con los requisitos
para el grado de Doctor en Ciencias
en el campo del conocimiento de
Instrumentación y mediciones automáticas

Facultad de Ingeniería

Campus Mexicali

23 de septiembre de 2025

Declaración de Autoría

Yo, Jonathan Jesus Sanchez Castro, declaro que esta tesis titulada, «Desarrollo de un método de análisis de pose facial mediante inteligencia artificial y estéreo visión con aplicación en ADAS», y el trabajo presentado en ella son de mi autoría. Confirmo que:

- Este trabajo fue realizado en su totalidad o principalmente mientras estaba matriculado en un programa de investigación en esta Universidad.
- Donde alguna parte de esta tesis fue presentada previamente para un grado o cualquier otra calificación en esta Universidad u otra institución, esto se ha indicado claramente.
- Cuando he consultado el trabajo publicado de otros, siempre se ha atribuido claramente.
- Cuando he citado el trabajo de otros, siempre se ha indicado la fuente. Con la excepción de tales citas, esta tesis es enteramente de mi propia autoría.
- He reconocido todas las fuentes principales de ayuda.
- Cuando la tesis se basa en trabajo realizado por mí en colaboración con otros, he dejado claro exactamente lo que hicieron otros y lo que contribuí yo mismo.

Firmado:

Fecha:

«You don't become a champion by doing what is easy. You become a champion by constantly pushing your boundaries and challenging yourself.»

Nigel Mansell

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Resumen

Facultad de Ingeniería
Campus Mexicali

Doctor en Ciencias

Desarrollo de un método de análisis de pose facial mediante inteligencia artificial y estéreo visión con aplicación en ADAS

por Jonathan Jesus Sanchez Castro

El análisis facial es una disciplina de gran relevancia en la actualidad, impulsada por los avances tecnológicos y su creciente aplicación en áreas como la psicología, la salud y el sector automotriz. Esta tesis se enfoca en una tarea clave del análisis facial: la estimación de la pose facial, fundamental para múltiples sistemas que requieren mediciones precisas con bajo margen de error. Por ello se propone un enfoque basado en un sistema de visión estéreo para el cálculo de profundidad, matrices de rotación para recuperar los ángulos y detección de puntos faciales en 2D en ambas cámaras, sustituyendo a algoritmos de emparejamiento de patrones convencionales.

Al obtener los puntos coincidentes en el par de imágenes, se aplica la técnica de triangulación para obtener coordenadas tridimensionales de las características faciales. Esto permite modelar el problema como un caso de "n puntos de perspectiva" y recuperar la pose mediante rotaciones. El sistema demostró un baja incertidumbre obteniendo mediciones menores a un angulo de error, estos resultados son comparables o superiores a los de metodologías basadas en IA, y alcanzó una velocidad de procesamiento de 17 cuadros por segundo, lo cual lo hace apto para aplicaciones en tiempo real, como la captura de expresiones faciales ola estimación de pose facial.

Por último, debido a su buen desempeño, el sistema es aplicado en un contexto ADAS (Sistema Avanzado de Asistencia al Conductor), permitiendo advertir escenarios peligrosos mediante el análisis de la atención del conductor, combinando la estimación y clasificación de la pose facial con la detección de objetos y peatones en la vía.

Agradecimientos

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), por brindar el apoyo económico para financiar la investigación con becas para realizar los estudios de posgrado.

A la Universidad Autónoma de Baja California (UABC), por la oportunidad para estudiar un posgrado y obtener el grado de Doctor en Ciencias. A la Facultad de Ingeniería de la UABC, por las aulas, laboratorios y profesores para la formación académica de estudios de posgrado.

Al Dr. Julio Cesar Rodríguez Quiñonez, mi director de tesis, por darme la oportunidad de realizar este proyecto de investigación, por su apoyo constante a lo largo de mis estudios de posgrado y por compartir su conocimiento y experiencia.

Al Dr. Guillermo Galaviz, mi co-director de tesis, por trabajar con el Dr. Julio Cesar Rodríguez Quiñonez para realizar este proyecto de investigación, ádemas por su ayuda y tiempo.

A mi comité de tesis, conformado por el Dr. Oleg Sergiyenko, Dr. Jesús Elías Miranda Vega y Dra. Wendy Flores Fuentes, por su apoyo y retroalimentación de cada semestre en las presentaciones de avance de tesis.

A la Universidad Autónoma de Sinaloa (UAS), por el apoyo económico brindado para estudiar un posgrado y obtener el grado de Doctor en Ciencias.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Antecedentes

En el área de visión por computadora convergen múltiples disciplinas que abordan tareas esenciales para la vida cotidiana, con aplicaciones en sectores como el comercial, vehicular, psicológico y de salud. Estas disciplinas emplean diversas metodologías para desarrollar sistemas de clasificación, detección y análisis, utilizados en tareas como la identificación de vehículos, peatones, detección de rostros y análisis facial. Este último constituye el enfoque principal de investigación en la presente tesis doctoral. En este trabajo se propone una nueva metodología para una tarea específica del análisis facial: la estimación de la pose, lograda mediante la combinación de algoritmos de inteligencia artificial (IA) y herramientas matemáticas como la triangulación y las matrices de rotación. Asimismo, se explorará su utilidad en aplicaciones de alto impacto dentro del sector automotriz, particularmente en el diseño de sistemas avanzados de asistencia al conductor (ADAS, por sus siglas en inglés).

Dentro de sectores como los videojuegos, multimedia, salud, seguridad y automovilismo, el análisis facial juega un papel crucial para resolver problemáticas inherentes a las aplicaciones que se desean desarrollar. Por ejemplo, es útil en sistemas de reconocimiento facial para seguridad, en imágenes generadas por computadora (CGI) y en la extracción de la pose para analizar patrones y movimientos faciales en videojuegos o el entretenimiento, como la creación de películas. En este último caso, los estudios cinematográficos han comenzado a utilizar sistemas sofisticados para la captura de movimientos (Khan et al., 2021; Ju et al., 2022; Abate et al., 2022). Estas aplicaciones, décadas atrás, dependían de algoritmos matemáticos que limitaban los grados de libertad de los sistemas. Sin embargo, con el auge de la IA se comenzaron a implementar algoritmos de aprendizaje de máquina (ML) y, más recientemente, métodos avanzados como el aprendizaje profundo (DL), destacando

las redes neuronales. La IA ha transformado el área de visión por computadora al superar muchas de las limitaciones de los enfoques tradicionales y ha permitido una revolución en múltiples áreas del análisis de imágenes.

Analizar la geometría y movimientos faciales mediante imágenes para tareas de estimación y reconocimiento no es trivial, ya que implica extraer características visuales de imágenes en blanco y negro o en RGB. Posteriormente, estas características son procesadas mediante métodos de IA que generan espacios de interpretación (mapeo) de las variables y facilitan la comprensión de los datos. Ejemplos de estas técnicas incluyen el Análisis de Componentes Principales (PCA), Máquinas de Soporte Vectorial (SVM), agrupamiento de K-medias (K-means clustering), Árboles de Decisión, y métodos más recientes como Redes Neuronales Convolucionales (CNN) (Alqahtani et al., 2020).

La inteligencia artificial es una herramienta poderosa que, gracias a su auge reciente, ha cobrado gran relevancia en todo el mundo. Su integración en áreas esenciales como la salud, seguridad y automovilismo ha permitido resolver problemas complejos, incluso combinando estas áreas en aplicaciones dentro de los sistemas avanzados de asistencia al conductor (ADAS). Por ejemplo, la IA permite tareas como la detección facial, el reconocimiento de expresiones y la estimación de la pose facial (HPE) de un conductor, fundamentales para evaluar su nivel de atención al volante. Estas aplicaciones presentan retos significativos, como el diseño de algoritmos de detección facial y de estimación de características y pose. Aunque los algoritmos basados en imágenes 2D han mostrado buenos niveles de precisión, su dependencia exclusiva de esta dimensión presenta limitaciones al intentar analizar movimientos faciales o realizar clasificaciones complejas, debido a la falta de información tridimensional. Esto se debe a que las imágenes 2D solo proporcionan información parcial de los tres ejes de rotación conocidos como ángulos de Euler (Yaw, Pitch y Roll).

Debido a estas complicaciones, los investigadores han comenzado a utilizar información más allá del plano 2D, desarrollando sistemas 2.5D y 3D (Aouada et al., 2021; Zhao et al., 2015). Los modelos 2.5D extraen datos visuales mediante métodos matemáticos o a partir de múltiples imágenes, obteniendo puntos de referencia espaciales para crear sistemas más robustos. Por otro lado, los modelos 3D reconstruyen el rostro en tres dimensiones, no para medir con precisión todos sus detalles, sino para identificar características clave como ojos, boca y orejas, según la aplicación deseada. Por ejemplo, en la rehabilitación infantil, estas técnicas se utilizan para evaluar el compromiso y atención de los niños durante actividades terapéuticas (Malek y Rossi,

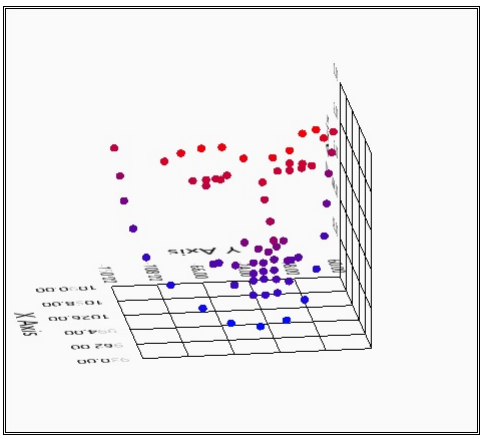

FIGURA 1.1: Puntos faciales en 3D generados por el sistema propuesto.

2021). Una representación gráfica de características faciales en 3D se puede observar en la Figura 1.1.

Adicionalmente, algunos algoritmos utilizan métodos de reconstrucción facial mediante sistemas LIDAR, módulos Kinect o visión estéreo 1.2. Los sistemas LIDAR, aunque efectivos, requieren elementos mecánicos que pueden ser problemáticos en aplicaciones como el análisis facial en automoviles, donde el espacio es limitado y la precisión es crítica. Por otro lado, los sistemas de visión estereoscópica (SVS), que trabajan con dos canales de imágenes RGB y emergen como una alternativa ideal debido a sus ventajas frente al LIDAR o Kinect (Salmane et al., 2023), entre las cuales destacan:

- Mayor resolución en mapas de disparidad.
- Uso de imágenes RGB en lugar de sensores IR.
- Mayor área de inspección.
- Menor susceptibilidad a problemas de reflexión.
- Ausencia de partes mecánicas.
- Versatilidad en aplicaciones.

Como se mencionó anteriormente, la inteligencia artificial (IA) es una herramienta en auge, utilizada no solo con información 2D, sino que ahora también incorpora la estimación y el manejo de información tridimensional. Esto es crucial para incrementar la fiabilidad de sistemas que dependen del análisis de movimientos tridimensionales, como los sistemas de seguridad en detección facial, clasificación de expresiones y el cambio de pose de un conductor.

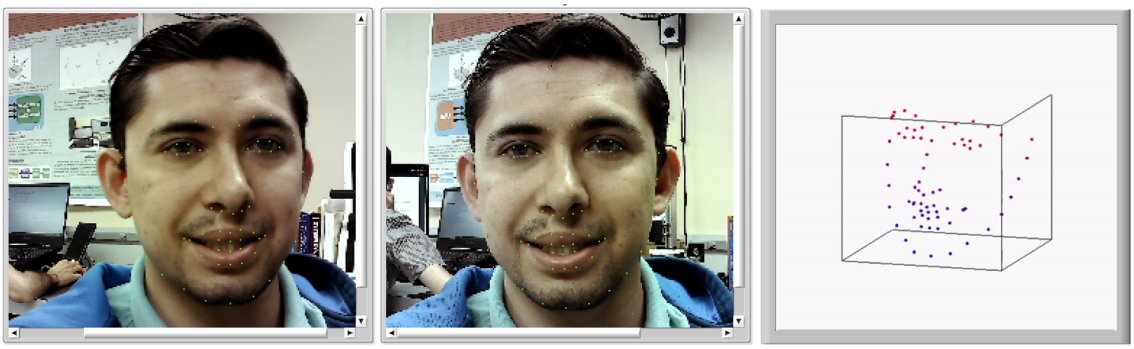

FIGURA 1.2: Aplicación de estéreo visión para cálculos de puntos 3D.

Un ejemplo destacado es el trabajo de Alagha et al., 2022, donde los autores presentan un sistema de visión estéreo enfocado en el análisis de parálisis facial para uso clínico, evitando la implementación de métodos invasivos como las electroneurografías. En dicha investigación, se resalta el uso de sistemas estéreo debido a su capacidad para obtener mediciones tridimensionales precisas. Estas mediciones son especialmente útiles en aplicaciones como la generación de mapas de profundidad facial, permitiendo analizar grandes volúmenes de información mediante algoritmos de inteligencia artificial. Para esta aplicación, la precisión en las mediciones es crucial, por lo que se emplearon cámaras de alta resolución. Además, los rostros de los sujetos fueron analizados en condiciones estáticas y bajo iluminación controlada, con el objetivo de habilitar mediciones de simetría facial. Finalmente, los resultados obtenidos fueron correlacionados con métodos convencionales, concluyendo que el sistema demuestra potencial para ser utilizado como herramienta de asistencia clínica.

La visión estéreo habilita una amplia gama de aplicaciones mediante el análisis de imágenes. Esto se debe a la creciente tendencia en el área de visión por computadora e inteligencia artificial de utilizar imágenes para diversas tareas. En este contexto, la visión estéreo adquiere gran relevancia, ya que los métodos no invasivos son ideales para aplicaciones humanas, como el análisis facial. Una de las tareas clave en este ámbito es la estimación de la pose, que esta investigación busca abordar mediante el desarrollo de un método capaz de aplicarse en escenarios de seguridad vial dentro de los sistemas ADAS.

Para implementar una metodología basada en imágenes en un caso real,

como el entorno de un vehículo, es fundamental que esta funcione a una velocidad adecuada (mayores a 15 cuadros por segundo), para la captura de movimientos faciales. Sin embargo, muchas metodologías existentes, aunque logran buenos resultados en términos de precisión, presentan limitaciones significativas en cuanto a la exactitud de la estimación y la velocidad de procesamiento, entre otros aspectos.

En este contexto, métodos como el propuesto por Xu, Jung y Chang, 2022, que utiliza un enfoque RGBD (imagen RGB más información de profundidad), resultan fundamentales. Este tipo de cámara no solo captura imágenes, sino también datos de profundidad, lo que permite registrar información espacial del rostro. Aunque podría no detectar características faciales específicas como los ojos, la nariz o la boca, facilita la obtención de los ángulos de Euler necesarios para estimar la pose. El trabajo también incluye un análisis detallado sobre la relación entre la cantidad de puntos capturados, el procesamiento y el error de medición. Los resultados muestran que, a mayor cantidad de puntos de profundidad, la velocidad de procesamiento y el error de medición disminuye. En particular, el sistema alcanza una tasa de 16 cuadros por segundo (FPS) con un margen de error cercano a 1 grado en todos los ejes, utilizando 4096 puntos en la nube de puntos.

Además de la inteligencia artificial, existen métodos convencionales basados en cálculos matemáticos de alto consumo de recursos computacionales, los cuales permiten obtener buenos resultados en la estimación de la pose, pero presentan deficiencias en términos de velocidad de procesamiento (Yu, Mora y Odobez, 2017). Un ejemplo de esto es el trabajo de Yuan et al., 2020, donde se utiliza un método de parametrización 3D conocido como 3DMM (Modelos Morfables en 3D). Este enfoque consiste en construir un modelo tridimensional a partir de información bidimensional (imágenes 2D) y, posteriormente, resolver la estimación mediante el uso de matrices de rotación. Estos métodos suelen emplear técnicas avanzadas de modelado 3D, como los 3DMM, que permiten generar modelos tridimensionales basados en múltiples imágenes. Sin embargo, este enfoque puede resultar altamente demandante en términos de recursos computacionales y tiempo de procesamiento debido a la cantidad de imágenes requeridas. Esto limita su aplicación en escenarios donde se necesita una respuesta rápida. Además, su uso está restringido a fotografías frontales del rostro.

Aunque los sistemas 2D han mostrado buenos resultados en el análisis facial, estos pueden limitar a las aplicaciones más avanzadas como el modelado

3D, la estimación de pose y la detección de expresiones. Estas tareas son críticas para sistemas enfocados en la salud, la seguridad y la conducción. Además, el uso de métodos de inferencia de profundidad convencionales (por ejemplo los 3DMM) pueden resultar en una alta demanda de procesamiento, lo que dificulta su implementación práctica. Por ellos se ha planteado utilizar la visión estereoscópica, gracias a su capacidad de generar datos tridimensionales y versatilidad de aplicación de algoritmos de IA, resulta adecuado para tareas críticas en los sistemas ADAS, como la evaluación de la atención del conductor en tiempo real. Asimismo, permite la implementación de distintas tareas de análisis facial, como el estimación de la pose facial.

1.2. Planteamiento del Problema

El análisis facial es un área amplia de investigación con aplicaciones que abarcan la detección de rostros, extracción de características, reconocimiento facial, codificación de movimientos faciales y clasificación de expresiones, entre otras. Estas tareas se logran mediante la extracción de características faciales, que permite clasificar emociones básicas, movimientos faciales y pose facial. Dentro de este campo de investigación, los sistemas de clasificación de expresiones y los analizadores de pose facial son temas de estudio continuo que enfrentan ciertos desafíos, dado que tradicionalmente se trabaja con información bidimensional (imágenes RGB o en escala de grises). Este enfoque presenta limitaciones debido a la falta de información tridimensional; el uso de mapas de profundidad o nubes de puntos permite un tratamiento más detallado de los datos y brinda información estructural adicional sobre el objeto. En este contexto, el uso de un sistema de visión estereoscópica puede mejorar los sistemas de análisis facial, como los estimadores de pose, al ofrecer un tercer grado de libertad en el análisis; de esta forma, se cuenta no solo con información visual, sino también con datos de distancia que representan las dimensiones reales del rostro.

1.2.1. Preguntas de Investigación

¿Cuál es el efecto del tipo de enfoque de estimación de la pose facial sobre la latencia e incertidumbre en tiempo real para ADAS, y cómo la evidencia obtenida justifica el desarrollo y validación de un método original basado en IA con FLD-68 y visión estereoscópica?

¿Cómo se relacionan las características faciales: nariz, boca, ojos, labios, orejas, cejas, geometría y simetría, con el porcentaje de error en la estimacion de la pose mediante características en 3D, cuándo estas son proporcionadas por un sistema de estéreo visión?

1.3. Justificación y uso de los Resultados

El análisis facial, específicamente en esta investigación sobre la estimación de pose facial (HPE), es una aplicación multidisciplinaria con relevancia en áreas como la psicología, la salud y la seguridad automotriz (Malek y Rossi, 2021; Rodriguez-Quiñonez et al., 2024). Estos sistemas de análisis facial se adaptan a diversas aplicaciones que buscan ser no invasivas para el cuerpo humano y automatizar el proceso de análisis. Como mencionan Li et al., 2021, un aspecto clave en la implementación de sistemas ADAS para la prevención de accidentes es la reducción de muertes y lesiones, pero también es importante considerar el impacto económico. Según la Administración Nacional de Seguridad del Tráfico en Carreteras (NHTSA) de Estados Unidos Administration et al., 2016, en 2016 los accidentes representaron un costo total de 242 mil millones de dólares. Específicamente, los accidentes causados por conductores distraídos representaron aproximadamente el 16 % de este costo total.

En muchos casos, la falta de atención o distracción del conductor puede deducirse a partir de su postura, lo que compromete su capacidad para tomar decisiones críticas que podrían prevenir colisiones, como lo menciona Addanki et al., 2020. Por ello, las herramientas de análisis de pose y clasificación resultan fundamentales, ya que su aplicación puede contribuir a la reducción de accidentes viales, disminuyendo tanto los daños materiales como la pérdida de vidas humanas.

Es importante destacar la relevancia de los algoritmos de inteligencia artificial (IA) en este tipo de aplicaciones, ya que permiten la automatización de tareas complejas. Una característica común de los algoritmos de IA, como las SVM, CNN y ANN, es la necesidad de entrenarse con bases de datos específicas para la aplicación que se desea resolver. Sin embargo, en el caso del análisis de imágenes 2D, se han identificado limitaciones en la exactitud alcanzada por estos sistemas, como se menciona en Abate et al., 2022. Para abordar este desafío, se han empleado bases de datos que incluyen información de profundidad con el objetivo de entrenar algoritmos capaces de estimar la variación de la pose facial a partir de una sola imagen. Ejemplos de

estos trabajos incluyen Borghi et al., 2018; Baltrušaitis, Robinson y Morency, 2016, donde se utilizan bases de datos como BIWI, ICT-3DHP y AFLW2000. Estos algoritmos emplean un canal de información común basado en mapas de profundidad existentes o en un sistema híbrido. Para superar esta problemática, de buscar bases de datos de estéreo visión, las cuales son prácticamente inexistentes, se plantea el uso de visión estéreo como un mecanismo principal de medición en tres dimensiones.

Además de los enfoques basados en inteligencia artificial, existen métodos convencionales fundamentados en cálculos matemáticos que, si bien pueden ofrecer resultados precisos en la estimación de la pose, presentan una alta demanda computacional, lo que afecta su velocidad de procesamiento (Yu, Mora y Odobez, 2017). Un ejemplo es el trabajo de Yuan et al., 2020, donde se emplea un método de parametrización 3D conocido como Modelos Morfables en 3D (3DMM). Este enfoque permite construir un modelo tridimensional a partir de imágenes 2D y estimar la pose mediante matrices de rotación. Sin embargo, estos métodos requieren una gran cantidad de imágenes y un elevado consumo de recursos computacionales, lo que limita su aplicación en escenarios que requieren respuestas en tiempo real. Además, su uso suele restringirse a fotografías frontales del rostro.

Considerando lo anterior, esta investigación busca desarrollar un método de estimación de pose que logre un equilibrio entre exactitud y cuadros por segundo, con el objetivo de aplicarlo en sistemas ADAS. Para ello, se propone el uso de visión estéreo, aprovechando sus ventajas y su capacidad de integración con modelos de inteligencia artificial. Esto permitirá diseñar un sistema de análisis con velocidades de procesamiento adecuadas y adaptable a múltiples aplicaciones.

Además, se plantea el uso de modelos matemáticos para recuperar la pose facial a partir de la información tridimensional obtenida mediante triangulación. Dicho proceso se combinará con un algoritmo de extracción de características faciales que permita obtener datos precisos del rostro. Finalmente, la versatilidad del uso de un sistema de visión estéreo (SVS) permitirá la evolución del sistema en el futuro, ya que su desarrollo se basa en el conjunto de imágenes capturadas.

1.4. Objetivos de la Investigación

1.4.1. Objetivo General

Desarrollar un método de estimación de pose facial (HPE) utilizando visión estereoscópica y algoritmos de inteligencia artificial para la medición de características faciales tridimensionales, y un método de clasificación basado en redes neuronales para la clasificación de las zonas de atención de conductores para los sistemas de ADAS.

1.4.2. Objetivos Específicos

- Crear un programa para la generación de una base de datos de estéreo visión.
- Desarrollar un sistema de detección facial y extracción de características faciales en 3D.
- Implementar un algoritmo de estimación de pose, para una estimación continua de los ángulos de Euler.
- Implementar un sistema de detección facial el cual permita el cambio de pose con rotaciones de la cabeza de +/- 20° en el eje vertical y horizontal.
- Desarrollar un método de clasificación de pose facial con las facciones tridimensionales. recuperadas, utilizando IA y la propia base de datos fabricada, para las distintas zonas de atención del sistema ADAS.

1.5. Hipótesis

El uso de un sistema de visión estereoscópica y métodos de IA utilizados en el emparejamiento de patrones para la estimacion de la pose facial incrementará la precisión de la estimación continua de los ángulos de Euler, ademas de la velocidad de procesamiento superior a las 15 cuadros por segundo en comparación con métodos similares en el estado del arte. Lo que permitirá su implementación como algoritmo de detección de zonas de atención del conductor para su implementación como sistema ADAS.

Capítulo 2

Marco Teórico

Este capítulo tiene como objetivo explorar las teorías fundamentales detrás de los métodos utilizados en visión estereoscópica, emparejamiento de patrones, extracción de características faciales, detección de rostros y las bases teóricas de las matrices de rotación. Se llevará a cabo un análisis detallado de cada una de estas técnicas, destacando su relevancia e importancia en aplicaciones relacionadas con el análisis facial y la estimación de la pose, además de su uso en los sistemas ADAS.

2.1. Visión Estereoscópica

Los sistemas de visión estereoscópica (SVS) son una tecnología dentro del campo de la visión por computadora (CV) las cuales tienen como objetivo realizar mediciones tridimensionales de un espacio, objeto o punto específico, mediante la percepción de profundidad. Esto se logra mediante la aplicación de los principios de la visión humana. Al igual que una persona percibe la profundidad con ambos ojos, estos sistemas replican ese proceso utilizando dos o más cámaras y basándose principalmente en el principio del paralaje de imagen.

Los SVS emplean el principio del paralaje para habilitar metodologías basadas en trigonometría y óptica, permitiendo el cálculo preciso de la distancia de un objeto. Estos sistemas son fundamentales en el área de CV debido a sus características únicas:

- Versatilidad de aplicaciones
- Escalabilidad
- Resolución adaptativa
- Funcionamiento sin iluminación activa

FIGURA 2.1: Sistema de visión estereoscópica utilizado para la aplicación planteada.

- Ausencia de componentes dinámicos
- Tecnología de cero contacto

Los sistemas de visión estereoscópica (SVS) tienen aplicaciones en diversos sectores, como seguridad, automovilismo y salud Flores-Fuentes et al., 2023. En navegación autónoma de robots, permiten suavizar y brindar mayor estabilidad en las trayectorias de robots autónomos Statello et al., 2016. En ciberseguridad Rehman, Po y Liu, 2020, contribuyen a aumentar la fiabilidad de los sistemas biométricos de reconocimiento facial. En el ámbito de la salud Alagha et al., 2022, un SVS permite la medición de propiedades faciales para clasificar la parálisis facial.

En seguridad vial, la investigación de Khairdoost et al., 2020 muestra cómo la implementación de un SVS permite obtener información sobre los patrones de conducción de un usuario. Esta información facilita el uso de algoritmos de aprendizaje automático (ML) para predecir cambios de carril. Para ello, se utilizan dos SVS: uno con cámaras infrarrojas (IR) para la estimación de la posición y orientación de la mirada, y otro con cámaras RGB para la fabricación de mapas de profundidad del entorno frontal del vehículo. Si bien el objetivo principal de este tipo de investigaciones es la predicción de maniobras vehiculares, también contribuyen a la recopilación de datos esenciales para el desarrollo de sistemas de navegación autónoma para vehículos.

Como se ha mencionado, los SVS ofrecen grandes beneficios en diversas aplicaciones, particularmente en aquellas relacionadas con mediciones del cuerpo humano, como el análisis tridimensional del rostro para aplicaciones de reconocimiento facial, detección facial, clasificación de expresiones y análisis de pose. Al ser un sistema alternativo de medición, no requiere contacto físico con el cuerpo humano y posee una gran versatilidad en aplicaciones. Por ello, las investigaciones relacionadas con estos sistemas y las tecnologías emergentes de inteligencia artificial (IA) cobran gran importancia en el mundo actual, ya que ofrecen soluciones innovadoras a problemas contemporáneos y permiten la automatización de ciertos procesos.

Los SVS utilizan el principio de paralaje para su funcionamiento, pero no es la única metodología integrada en el sistema para realizar mediciones en 3D. En las siguientes secciones se abordarán temas como triangulación, calibración y correspondencia de características, entre otros, los cuales son fundamentales para alcanzar el objetivo final del sistema.

2.1.1. Principio de Paralaje

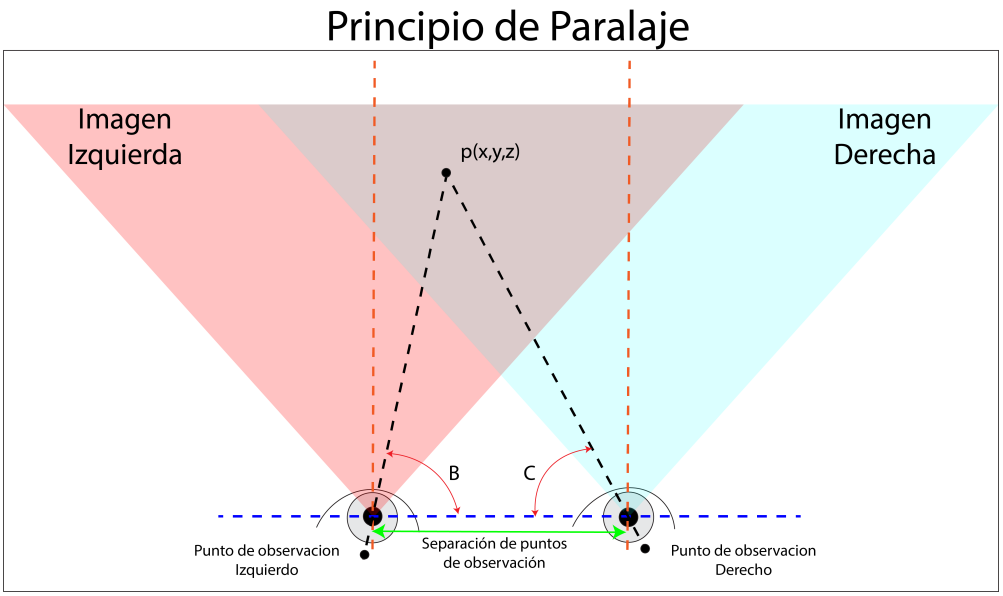

FIGURA 2.2: Descripción gráfica del principio de paralaje.

Así como el ser humano utiliza sus ojos para percibir profundidad, captando dos imágenes estereoscópicas desde puntos ligeramente distintos, su cerebro interpreta la diferencia angular entre ellas para reconstruir la profundidad en el mundo real. Esto se ilustra en la Figura 2.2, donde se representa el principio del paralaje. Dicho principio se basa en la diferencia de posición

entre dos imágenes (Izquierda y Derecha), capturadas desde distintos puntos de vista. En estas imágenes, un mismo punto en el espacio se observa con distintos ángulos de visión respecto al eje óptico. En la Figura 2.2, se puede notar que el ángulo B es mayor que el ángulo C. Esta variación angular permite al cerebro percibir la profundidad y ajustar el enfoque en objetos cercanos o lejanos.

Así como el cerebro humano logra percibir profundidad gracias a las disparidades en la imagen percibida, podemos modelar este proceso mediante el uso de métodos matemáticos basados en geometría. Como se observa en la Figura 2.2, los centros de los ojos y el punto P forman un triángulo, con el cual es posible analizar los grados de visión, los ángulos formados B, C y la separación entre los ojos para calcular la distancia a la que se encuentra el objeto. Para ello, se emplea la técnica conocida como triangulación.

2.1.2. Triangulación

Método de Parámetros Intrínsecos

Tomando como base el principio de paralaje de la visión humana, este puede ser modelado mediante un SVS, en el cual dos cámaras actúan como los sensores principales sensibles al espectro visible. Cuando ambas cámaras se encuentran paralelas entre sí y alineadas en un mismo eje a cierta distancia, es posible implementar el principio que rige nuestra visión.

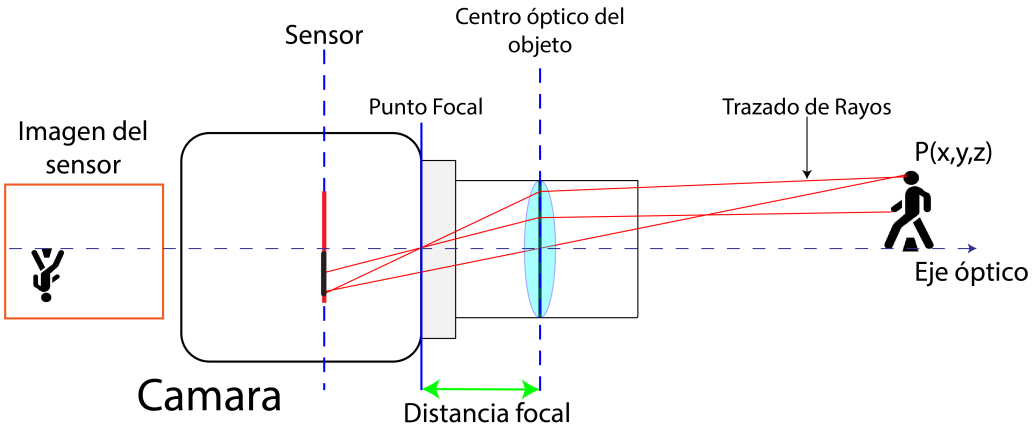

FIGURA 2.3: Representación del plano focal de una imagen que pasa a través del lente de la cámara a sensor CCD.

Al utilizar cámaras, es necesario considerar diversos factores para realizar el cálculo de la profundidad. En este proceso entran en juego las características de la cámara (ángulos de visión, punto focal), la teoría de trazado de rayos y puntos focales de los lentes en el ámbito de óptica. Esto se puede ver

ejemplificado por la Figura 2.3, en donde observamos el trazado de rayos de una imagen pasando por el lente hasta el sensor CCD de la cámara, además del ángulo de visión. Se puede observar que la Figura se ha invertido al pasar por el punto focal.

Partiendo de la Figura 2.3, podemos definir una relación entre el objeto real y su proyección en la imagen mediante el análisis de triángulos semejantes. Esto es posible gracias al trazado de rayos, partiendo del punto en el espacio en cuestión, pasando por el punto focal donde la imagen comienza a invertirse y llegando al sensor.

Este análisis permite establecer una relación entre el punto $P(X, Y, Z)$ en el espacio, con coordenadas en el mundo real, y su contraparte $p(x, y)$, determinada por las dimensiones en píxeles de la proyección de la imagen en el sensor, como se observa en la Figura 2.4.

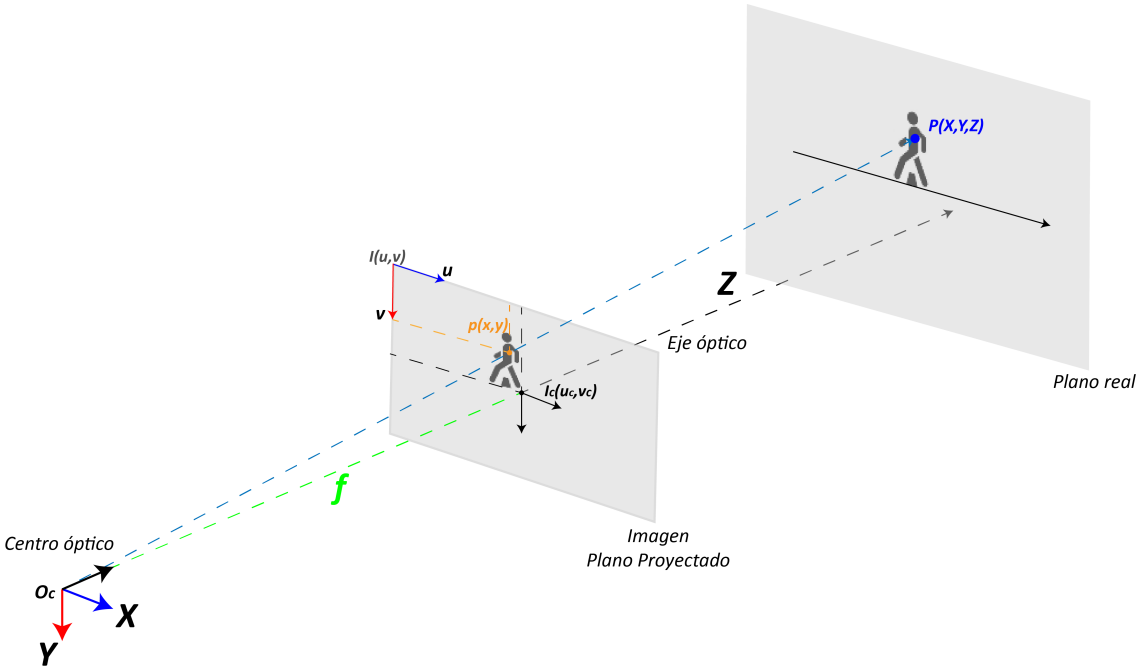

FIGURA 2.4: Representación geométrica equivalente del sistema formado entre el plano real y el plano de la imagen.

En la Figura 2.4, se ha realizado una modificación para visualizar adecuadamente la comparación de triángulos semejantes. Bajo las leyes de la óptica, cuando un rayo de luz cruza el punto focal, su trayectoria invierte el eje Y. Para mitigar este cambio de manera gráfica, se utiliza una representación geométrica equivalente del sistema sensor-imagen, donde se traslada la proyección de la imagen en el sensor y se indica el punto de partida del nuevo sistema geométrico para el cálculo de coordenadas. A partir de esta

representación, se puede observar la semejanza de los triángulos presentes en el sistema.

Para formular las ecuaciones de triangulación, es necesario establecer el eje central, también conocido como eje óptico O_c, como se muestra en la Figura 2.4. A partir de esta definición, podemos utilizar el análisis de semejanza de triángulos y observar que el cálculo de las coordenadas x, y en la imagen se puede realizar mediante las ecuaciones 2.1 y 2.2.

$$x = f\frac{X}{Z} \tag{2.1}$$

$$y = f\frac{Y}{Z} \tag{2.2}$$

Donde f es el valor del punto focal de la cámara. Estas ecuaciones parten de la premisa de que conocemos las coordenadas verdaderas, pero en este y muchos casos, en realidad, las coordenadas reales del punto $P(X, Y, Z)$ son las que se desean calcular.

En las ecuaciones anteriores, solo se utilizaba una cámara. Para recuperar las coordenadas 3D, se aplica el principio de paralaje en un SVS. A partir de los parámetros conocidos, es posible calcular dichos valores reales, como se representa en la Figura 2.5.

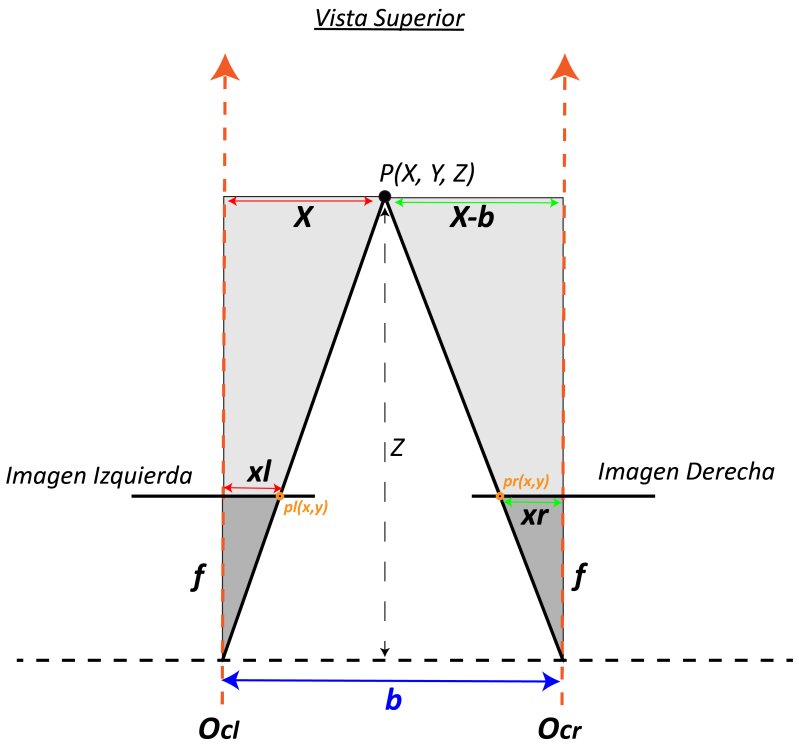

FIGURA 2.5: Representación geométrica equivalente de un SVS donde se observa la relación entre los triángulos.

Los sistemas estéreo están formados por dos cámaras paralelas entre sí con una separación conocida, lo que permite reformular las ecuaciones 2.1 y 2.2 para calcular coordenadas 3D. Mediante la semejanza de triángulos, obtenemos las siguientes ecuaciones:

$$\frac{X}{xl} = \frac{Z}{f} \tag{2.3}$$

$$\frac{Y}{yl} = \frac{Z}{f} \tag{2.4}$$

$$\frac{X-b}{xr} = \frac{Z}{f} \tag{2.5}$$

Donde b es la distancia entre los centros ópticos de las cámaras o imágenes, xr y xl representan el ancho en píxeles del área que ocupa la imagen dentro del sensor, y yr es el alto en píxeles de la imagen.

Despejando X de las ecuaciones 2.3 y 2.5, obtenemos:

$$X = \frac{Z}{f}xl = \frac{Z}{f}xr + b \tag{2.6}$$

A partir de esta ecuación, se puede observar el principio de disparidad $d = xr - xl$, lo que indica que la disparidad es la diferencia entre lo que percibe cada imagen en términos de píxeles.

Ahora, resolviendo para Z, obtenemos la siguiente ecuación:

$$Z = \frac{b * f}{xl - xr} \tag{2.7}$$

Sustituyendo el término de disparidad, obtenemos:

$$Z = \frac{b * f}{d} \tag{2.8}$$

La ecuación 2.8 permite calcular la profundidad de un objeto en la imagen a partir de la disparidad entre ambas imágenes. Con este valor de Z, también podemos calcular las coordenadas X, Y despejando los valores de píxel en sus respectivas ecuaciones 2.3 y 2.5, obteniendo:

$$X = \frac{Z}{f}x_l \tag{2.9}$$

$$Y = \frac{Z}{f}y_l \tag{2.10}$$

El análisis previo es posible si se considera que el SVS está calibrado y/o que sus cámaras se encuentran en paralelo, lo que indica que no hay ningún movimiento en el centro óptico de la imagen. En caso de que las cámaras presenten algún giro o traslación relativa entre sí, es necesario emplear los parámetros extrinsecos e intrínsecos de las cámaras, los cuales deben obtenerse mediante un proceso de calibración.

En un caso general, los puntos x, y pueden representarse en forma de sistema de ecuaciones para acomodar la matriz de parámetros intrínsecos y la matriz de rotación, como se muestra en la ecuación 2.11:

$$\begin{bmatrix} x \\ y \\ s \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & x_0 \\ 0 & \beta & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.11}$$

Donde α, β y γ representan la distancia focal en píxeles a lo largo de los ejes horizontal y vertical, y el factor de oblicuidad en la imagen, respectivamente, también esta matriz es conocida como matriz de parámetros intrínsecos. Además, x_0 y y_0 corresponden a los centros ópticos de la imagen. Estos cuatro elementos conforman lo que se conoce como la matriz de parámetros extrínsecos A. Por otro lado, los términos R_{jk} y t_i representan los valores de rotación y traslación de la imagen, respectivamente, formando en conjunto lo que se conoce como matriz de rotación.

La ecuación 2.11 se emplearía en un método de calibración para obtener los parámetros intrínsecos y extrínsecos de las cámaras, con el objetivo de habilitar el calculo de profundidad mediante las ecuaciones 2.8, 2.9 y 2.10 de triangulación (Rodríguez-Quiñonez et al., 2024, Lau, Sacil e Ibarra, 2024, Taryudi y Wang, 2018, He et al., 2025).

Triangulación por Ángulos de Visión

El procedimiento de triangulación basado en parámetros intrínsecos de las cámaras es uno de los dos métodos con un reconocido arraigo en las aplicaciones de medición en 3D, las cuales cuentan con un vasto número de aplicaciones. Sin embargo, este no es el único método utilizado para el cálculo de la tercera dimensión, ya que también existe un enfoque basado en los ángulos de visión de las cámaras del sistema estéreo (Rodriguez-Quiñonez et al., 2024, Ramírez-Hernández et al., 2020), como el que se observa en 2.6.

En este enfoque, conociendo los ángulos de visión y la distancia de separación entre las cámaras, es posible utilizar nuevamente la geometría de los

triángulos. Sin embargo, en este caso, los ángulos de visión de las cámaras juegan un papel fundamental en la formación de los sistemas de triángulos entre los píxeles de la imagen y el objeto o punto P en cuestión, y mediante la aplicación de la ley de senos se pueden derivar ecuaciones capaces de obtener valores de profundidad.

La Figura 2.7 ilustra el principio de triangulación, mediante el cual se calcula el valor de profundidad dado el plano epipolar de las dos imágenes y sus ángulos (B, C, β), los cuales se determinan a partir de los ángulos de visión correspondientes de cada cámara y la resolución de ambas imágenes. Además, el proceso de estimación de profundidad utiliza la imagen de la cámara izquierda como referencia para los cálculos, como en el método de parámetros intrínsecos.

La técnica de estimación de profundidad se lleva a cabo mediante las ecuaciones 2.12, 2.13 y 2.14, las cuales calculan las coordenadas espaciales X, Y y Z, respectivamente. Con estas coordenadas, es posible obtener una descripción tridimensional de un objeto, incluso de un rostro humano, la cual puede utilizarse para tareas de análisis de pose facial.

$$X = a\left(\frac{sinC * sinB}{sin(B+C)}\right) \tag{2.12}$$

$$Y = a\left(\frac{cosB * sinC}{sin(B+C)} - \frac{1}{2}\right) \tag{2.13}$$

$$Z = a\left(\frac{sinB * sinC * tan\beta}{sin(B+C)} - \frac{1}{2}\right) \tag{2.14}$$

De las ecuaciones 2.12, 2.13 y 2.14, a es la distancia entre cámaras del SVS. Los parámetros B, C y β son los ángulos formados bajo el sistema objeto-imagen y los ángulos de visión de la cámara. Estos parámetros pueden ser observados en las Figuras 2.7 y 2.8.

Las variables B y C son representadas por las ecuaciones 2.15 y 2.16.

$$B = B_i + B_0 \tag{2.15}$$

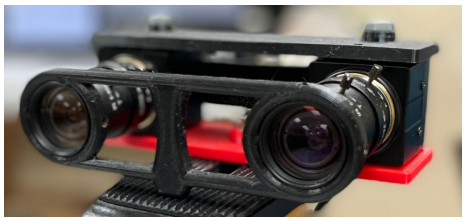

FIGURA 2.6: Sistema de Visión Estereoscopica.

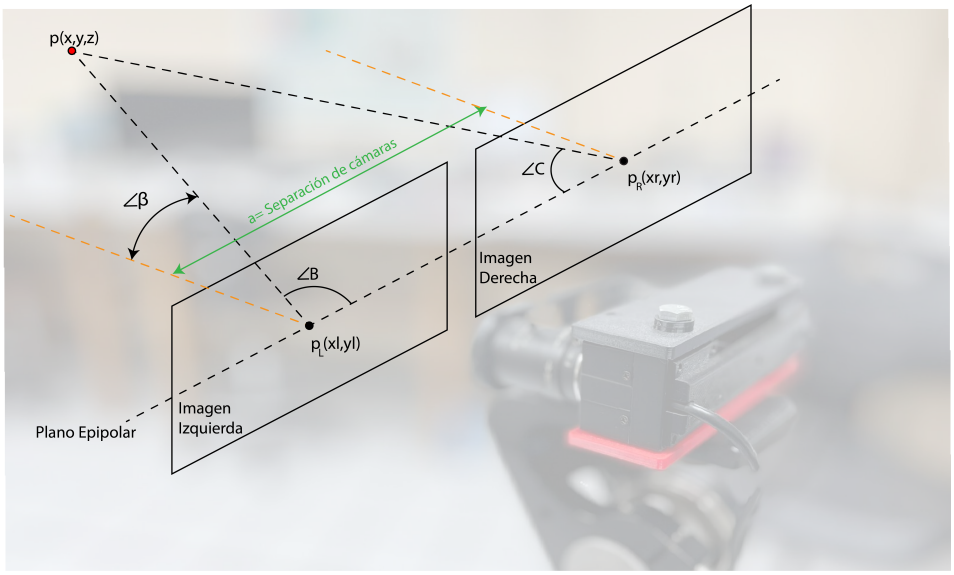

FIGURA 2.7: Estructura visual de un sistema estereoscópico, donde se pueden apreciar los ángulos propios de los triángulos formados (B, C, β).

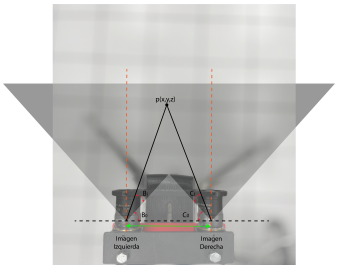

FIGURA 2.8: Vista superior del SVS, donde los ángulos B y C se muestran en correspondencia con su respectivo ángulo de visión y ejes centrales de la imagen.

$$C = C_i + C_0 \tag{2.16}$$

En donde B_i y C_i son los ángulos correspondientes que se encuentran dentro del ángulo de visión de las cámaras derecha e izquierda, respectivamente. B_0 y C_0 son los ángulos complementarios que van desde la base a hasta el comienzo del ángulo de visión B_i y C_i, respectivamente, y estos pueden observarse en detalle en la Figura 2.8.

Ahora, los ángulos B_i y C_i se calculan mediante la relación de ubicación en píxeles del objeto dentro de la imagen y el ángulo de visión de la cámara, por ello se implementan las siguientes ecuaciones:

$$B_i = H_{fov}\left(\frac{W - u_{xl}}{W}\right) \tag{2.17}$$

$$C_i = H_{fov}\frac{u_{xr}}{W} \tag{2.18}$$

En donde H_{fov} es el campo de visión en el eje horizontal (FOV, por sus siglas en inglés) expresado en grados y W el ancho de la imagen en píxeles de las cámaras. u_{xr} y u_{xl} son las coordenadas en píxeles del objeto o punto P observado en las imágenes derecha e izquierda sobre el eje horizontal, respectivamente.

Los términos B_i y C_i son dos variables que se calculan en base a la relación entre el ángulo de visión de la cámara y el tamaño del sensor. Se asume que todo lo que está presente dentro de la imagen corresponde exactamente a lo que está dentro del ángulo de visión, por lo que se divide la coordenada entre la resolución horizontal de la imagen, generando una medida de proporción para posteriormente multiplicar por el ángulo de visión y obtener el valor del ángulo correspondiente a la coordenada en píxeles.

Para los términos constantes B_0 y C_0 se utilizan las ecuaciones 2.19 y 2.20.

$$B_0 = 90 - \frac{H}{2} \tag{2.19}$$

$$B_0 = C_0 \tag{2.20}$$

En la ecuación 2.20, se puede identificar que los términos constantes que residen fuera de los ángulos de visión de las cámaras con respecto a la línea base a se consideran iguales. Esto se debe a que, al utilizar dos sensores idénticos, sus FOV también se consideran iguales, lo cual se asume por simplicidad de cálculo.

Ahora, para el ángulo correspondiente al eje vertical de las cámaras β, utilizamos la ecuación 2.21

$$\beta = (V_{fov})\left(\frac{\frac{V}{2} - P_{yl}}{V}\right) \tag{2.21}$$

En la cual, V_{fov} es el FOV en el eje vertical de la imagen y V la resolución de la imagen en el eje vertical. Por último, P_{yl} es la coordenada en píxeles de la imagen en el eje vertical de la imagen izquierda. Se realiza la división $\frac{V}{2}$ debido a que, en este caso, el sistema de coordenadas tendrá su origen en el borde de la mitad de la imagen izquierda.

La aplicación de este método de triangulación es posible gracias a las restricciones físicas del SVS, el cual garantiza el paralelismo entre las imágenes

y una separación exacta entre las cámaras. Además, aunque la diferencia de paralaje no se determine explícitamente, este valor estará implícito debido a los ángulos formados en el sistema de triángulos. Es importante recordar que la suma de los ángulos internos de un triángulo siempre es de 180 grados, por lo que cualquier variación en uno de ellos afectará proporcionalmente a los otros dos.

Para ambos métodos de triangulación existen parámetros que se deben cálcular para habilitar su respectivas ecuaciones de calculo de coordenadas 3D, pero en común estos dos métodos dependen de la correspondencia de patrones, ya que si bien un objeto puede encontrarse dentro de ambas imágenes, se debe de implementar metodologías para lograr automatizar el reconocimiento de patrones y crear correspondencias entre el set de imágenes para habilitar la triangulación. Esta metodología se presentara en la siguiente subsección. Además este método de triangulación por ángulos de visión es el que será implementado en esta investigación para obtener coordenadas 3D.

2.2. Inteligencia Artificial

La visión estéreo habilita la medición de coordenadas 3D mediante la técnica de triangulación, que hace uso del principio del paralaje. Como se ha mencionado, los SVS son sistemas de dos cámaras paralelas entre sí, separadas por una distancia base. Para que la técnica de triangulación funcione, un punto u objeto específico debe ser observado por ambas cámaras del sistema. Sin embargo, aunque lo que se observa en ambas cámaras pueda identificarse manualmente, el sistema necesita realizar este proceso de forma automática. Por esta razón, se utilizan los algoritmos de regresión, clasificación y detección de objetos.

En el área de la inteligencia artificial, existen diferentes categorías para la inteligencia artificial, las cuales se pueden ordenan según la "profundidad"de la estructura de red creada. Esta forma de clasificar las tecnologías de IA se puede observar en la Figura 2.9, donde se muestra que, a medida que nos adentramos en algoritmos de mayor complejidad, también avanza la categoría de los métodos de IA.

Los métodos de regresión, clasificación y detección mediante inteligencia artificial (IA) se implementan a través de distintos procesos, ya que cada uno opera de forma diferente. Existen enfoques que abordan estas tareas de manera individual, así como métodos híbridos que las combinan para cumplir un objetivo específico. Incluso, algunas técnicas modernas son capaces

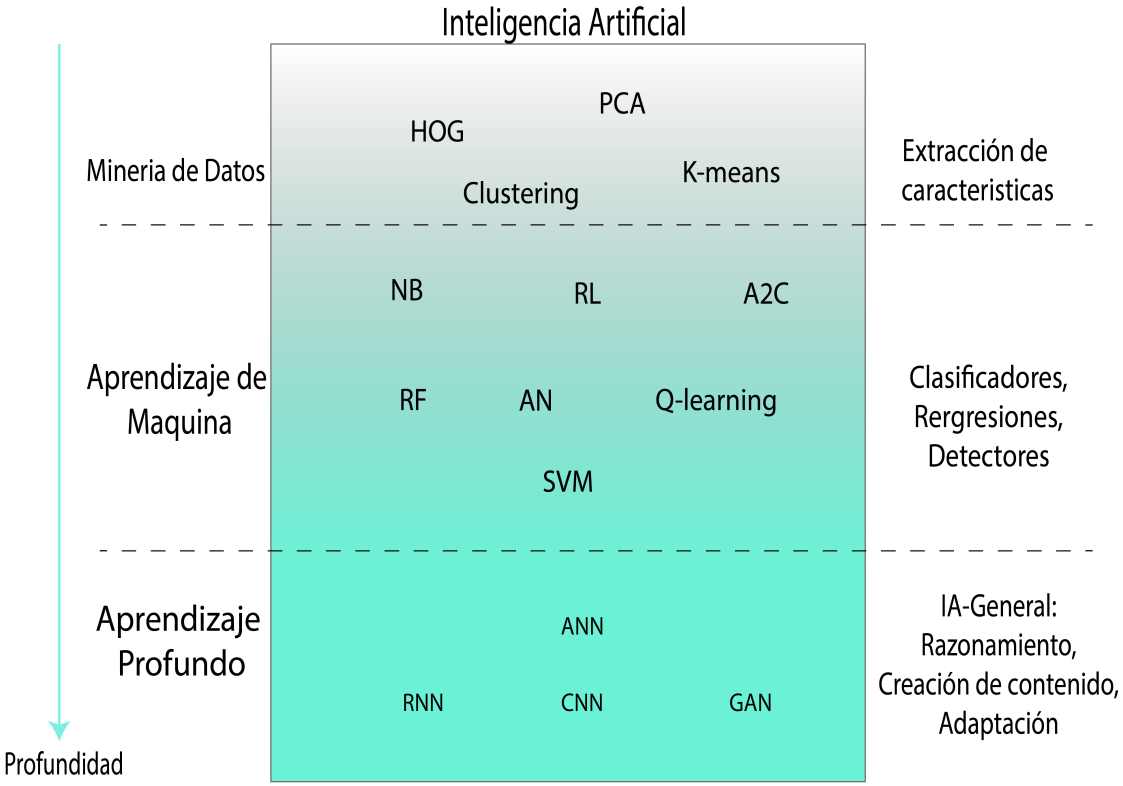

FIGURA 2.9: Ilustración de clasificación de algoritmos de IA acorde su profundidad.

de realizar simultáneamente las tres funciones: identificar, clasificar y localizar un objeto dentro de una imagen. A continuación, se presentan algunos de los métodos más reconocidos para la detección y clasificación de objetos.

2.2.1. Detección de Objetos

Los algoritmos de detección de objetos representan el siguiente paso después de haber adquirido las imágenes estéreo, dado que las ecuaciones de triangulación requieren la información correspondiente al objeto cuya profundidad se desea medir. Para este propósito, se emplean algoritmos de ML o DL, como máquinas de soporte vectorial (SVM, por sus siglas en inglés), K vecinos cercanos (KNN), redes neuronales (NN), árboles de decisión (RF, por sus siglas en inglés) y redes neuronales convolucionales (CNN, por sus siglas en inglés), entre otros (Singh et al., 2021; Sanchez-Castro et al., 2020; Terven, Córdova-Esparza y Romero-González, 2023; Kazemi y Sullivan, 2014; Mady e Hilles, 2018; Amir, Tadas y Louis-Philippe, 2017). La detección de objetos es utilizada en una amplia gama de aplicaciones, para el estudio realizado en

esta tesis se implemento la detección de rostros frontales en las imágenes del SVS.

Cada tipo de algoritmo tiene características únicas, y su selección depende de la aplicación específica. Para este trabajo, se utilizo el detector de rostros de la librería *DLib* de Python (King, 2009a; Kazemi y Sullivan, 2014), donde existen dos algoritmos para la detección de rostros: uno basado en el histograma de gradientes orientados (HOG) y SVM, y otro basado en CNN. En la Tabla 2.1 se puede observar las diferencias entre estos dos métodos, donde el algoritmo para aplicaciones de respuesta rápida es el HOG+SVM y en el caso de necesitar mayor precisión se puede aplicar el CNN.

TABLA 2.1: Características generales entre los dos métodos de detección de rostros existentes en la librería *Dlib*.

Método	Precisión	Velocidad	Ángulos extremos
HOG+SVM	Media	Alta	No
CNN	Alta	Baja (Require GPU)	Si

Detección de Rostros: HOG y SVM

En esta investigación, se utiliza un detector facial basado en histogramas de gradientes orientados (HOG) y un algoritmo de detección, el SVM. Convencionalmente, los algoritmos como HOG se emplean para la extracción de características, de modo que posteriormente un algoritmo de aprendizaje automático (ML) las utilice para detectar la ubicación del rostro en la imagen.

Los algoritmos HOG se utilizan para extraer la información de los bordes de un objeto en una imagen en escala de grises (BW, por sus siglas en inglés). No se emplean imágenes RGB, ya que las tonalidades proporcionan poca información relacionada con los bordes, ya que se considera que un borde es aquel donde existe una diferencia considerable de intensidad entre un pixel o conjunto de pixeles, es decir, se busca la primera derivada para conocer la razón de cambio entre valores de intensidad, por lo que se pueden calcular las derivadas parciales entre pixeles subsecuentes en ambos ejes para calcular las magnitudes de los gradientes Vincent, Folorunso et al., 2009.

Este método de extracción de características hace relación a las diferencias entre pixeles con teoría de vectores (Gradientes), es decir, calcula magnitudes de cambios (Magnitud del Gradiente) y dirección entre pixeles (Orientación del Gradiente). El proceso de cálculo de los HOG comienza obteniendo las

magnitudes de los cambios de intensidad en los bordes dentro de una imagen $I(u, v)$, mediante el proceso de convolución con operadores de diferencias direccionales (también conocidos como filtros o kernels) establecido para ambos ejes. Posteriormente, la imagen se divide en bloques de celdas de tamaño $j \times k$, donde cada celda forma otra submatriz de píxeles de tamaño $n \times n$. Para finalmente fabricar un histograma de las intensidades calculadas por cada bloque de celdas, a base del ángulo calculado de los gradientes en cada pixel (Dalal y Triggs, 2005; Pang et al., 2011).

El procedimiento para obtener las magnitudes de los gradientes se lleva a cabo mediante el cálculo de diferencias entre dos pixeles adyacentes de $i(u, v)$. Para el caso general, esto se expresa mediante las ecuaciones 2.22 y 2.23.

$$G_u = I(u+1, v) - I(u-1, v) \tag{2.22}$$

$$G_v = I(u, v+1) - I(u, v-1) \tag{2.23}$$

Donde los G_u y G_v representa los cambios de intensidad de los pixeles en la imagen en los ejes horizontal y vertical respectivamente. $I(u, v)$ representa la imagen en cuestión.

El proceso de el cálculo de magnitud puede tratarse como un problema de procesamiento de imágenes, lo que permite reescribir las ecuaciones en términos de la convolución de un filtro de (3×1) para el eje horizontal y (1×3) para el vertical, como se observa en las ecuaciones 2.24 y 2.25.

$$G_u = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * [I_{u,v}] \tag{2.24}$$

$$G_v = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * [I_{u,v}] \tag{2.25}$$

Partiendo desde el ámbito de procesamiento de imágenes estos algoritmos al trabajarse de manera matricial permiten implementar distintos filtros de detección de bordes como lo son el filtro Sobel y Prewitt. Para el caso del filtro de Sobel, sus kernels se muestran en las ecuaciones 2.26 y 2.27.

$$S_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.26}$$

$$S_v = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.27}$$

Se puede observar que existe un incremento en dimensiones de los filtros, esto porque ahora se considera a un vecindario de $(j \times k)$ de pixeles que influyen en la magnitud del gradiente y dirección, con respecto al pixel $i(u, v)$ analizado, lo que brinda una reducción de los efectos del ruido. El filtro de Sobel es utilizado con frecuencia en análisis de bordes de imágenes ya que debido a su simplicidad, uso de valores enteros es un filtro de bajo consumo de recursos computacionales (Khlamov, Tabakova y Trunova, 2022; Vincent, Folorunso et al., 2009; Chethan et al., 2019).

Sustituyendo el filtro de Sobel a las ecuaciones 2.24 y 2.25, se obtiene las ecuaciones 2.28 y 2.29.

$$G_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I(u, v) \tag{2.28}$$

$$G_v = \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I(u, v) \tag{2.29}$$

Estas ecuaciones también se pueden representar para un caso general con un kernel distinto de filtrado, este cambio se muestra en las ecuaciones 2.30 y 2.31.

$$G_u = F(j, k)_u * I(u, v) \tag{2.30}$$

$$G_v = F(j, k)_v * I(u, v) \tag{2.31}$$

Donde $F(j, k)$ es el filtro, kernel o mascara de la convolución. La Figura 2.10 representa el resultado de realizar el procedimiento de filtrado para el cálculo de diferencias de intensidad, lo que genera una imagen detallada con los bordes marcados de cada objeto presente.

Terminado el procedimiento de cálculo de diferencias en ambos ejes (u, v), se procede a calcular la magnitud del vector formado $|G(u, v)|$, por lo que se utiliza la expresión del cálculo de distancia de la ecuacion 2.32

$$|G_{u,v}| = \sqrt{(G_u)^2 + (G_v)^2} \tag{2.32}$$

FIGURA 2.10: Imagen filtrada con un detector de bordes, donde mientras más abrupta sea la diferencia entre intensidades mayor sera la magnitud del gradiente de la ecuación 2.32.

Posteriormente el cálculo del ángulo θ del gradiente esta definido por la ecuación 2.33.

$$\theta_{u,v} = \arctan\left(\frac{G_v}{G_u}\right) \tag{2.33}$$

Por último se procede a establecer los vecindarios de celdas $\mathbf{C_{p,q}}$ de tamaño $(n_u \times n_v)$ pixeles y los bloques $\mathbf{B_{r,c}}$ de tamaño de $(j \times k)$ celdas a dividir la imagen, realizado con la información de la matriz de magnitudes $|G(u,v)|$ y ángulos θ_{uv}, para crear un histograma en base de las direcciones de los gradientes de cada pixel en la imagen $I(u,v)$. Este procedimiento puede realizare individualmente por celda o aplicando la agrupación por bloques, donde mediante el desplazamiento parcial del bloque de una o mas celdas a la vez se realizan histogramas con información de $j \times k$ celdas. Cada histograma calculado por el desplazamiento del bloque se divide en una cantidad N_{bins} de bins, dependiendo de los intervalos de ángulos que se deseen, definido por la ecuación 2.34.

$$bin_\alpha = \frac{180^\circ}{N_{bins}} \tag{2.34}$$

Donde bin_α es el tamaño del intervalo angular de cada bin del histograma, dividido en N_{bins} bins, donde la selección de esta cantidad es dependiendo del rango de separación angular deseado. Utilizando la separación obtenida entre bins del histograma, se procede a analizar píxel por píxel en cada celda definida para generar los histogramas correspondientes, para posteriormente utilizar la información como entrada para un clasificador, como una SVM,

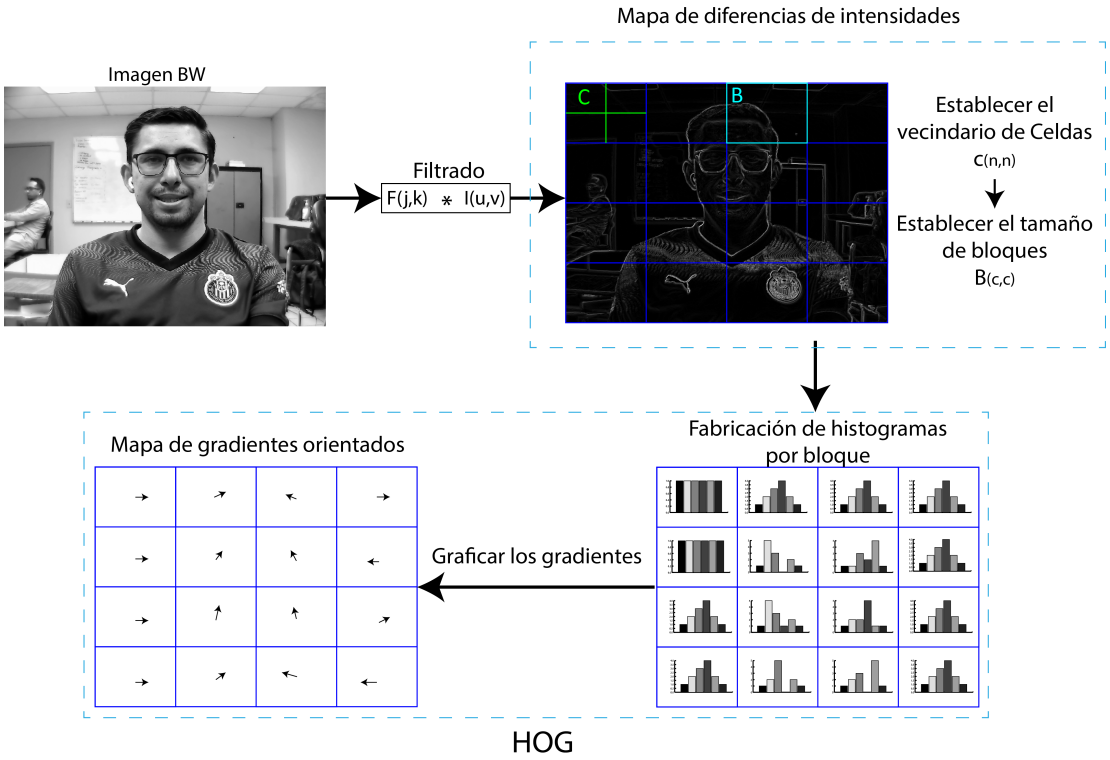

FIGURA 2.11: Proceso de extracción de características basado en HOG, donde **C** es la matriz de pixeles correspondientes a una celda y **B** es una matriz formada por una matriz de celdas.

para la detección de objetos en la imagen. La representación final de un HOG se observa en la Figura 2.11.

Realizado el proceso de extracción de características mediante el HOG, la información se introduce en una SVM para el proceso de detección de objetos. La información proporcionada al SVM serán los vectores producidos por los datos recopilados en cada histograma, es decir, se proporcionarán una cantidad determinada de vectores de características por celda $[x_0, x_1, ..., x_{N_{bins}}]$ o conjunto de histogramas por bloque $[h_0, h_1, ..., h_{N_{celdas}}]$. La totalidad de las características por bloque estaría dada por la siguiente ecuación 2.35.

$$N_{caractersticas} = N_{bloques} \times N_{cb} \tag{2.35}$$

Donde N_{cb} es la cantidad de características por bloque y $N_{bloques}$ es la cantidad de bloques de celdas formados, considerando solapamiento de celdas. N_{cb} se encuentra definida por la ecuación 2.36.

$$N_{cb} = j_u \times j_v \times N_{bins} \tag{2.36}$$

Donde, como se mencionó previamente, j_u y j_v son las dimensiones de los bloques en los ejes horizontal y vertical en la imagen respectivamente, y N_{bins}

es la cantidad de bins del histograma. Por último, la 2.37 define la cantidad de bloques superpuestos considerando el solapamiento.

$$N_{bloques} = \left(\frac{u}{n_u} - j_u + P\right) \times \left(\frac{v}{n_v} - j_v + P\right) \tag{2.37}$$

Donde $\frac{u}{n_u}$ y $\frac{v}{n_v}$ son la cantidad de celdas definidas en los ejes horizontal y vertical de la imagen, P es el valor de movimiento o paso de la ventana de bloques alrededor de la imagen, lo cual permite el solapamiento de bloques.

La ecuación 2.35 da a conocer la totalidad de características extraídas por un HOG, las cuales hay que conocer su organización para poder transmitirlas al clasificador SVM para la detección de rostros. La ecuación 2.38 detalla el formato del arreglo de información que se entrega a la SVM, donde se observa que se entregará un arreglo de vectores $F_{N_{bloques}}$.

$$F_{N_{bloques}} = [B_0, B_1, ..., B_{N_{bloques}}] \tag{2.38}$$

Donde B es un bloque de la imagen, el cual también está conformado por una cantidad de $(n_u \times n_v)$ celdas, las cuales forman a su vez un histograma de N_{bins} por cada una. Las ecuaciones 2.39 y 2.40 detallan el tamaño de los arreglos de cada bloque y posteriormente de cada celda conformada por los histogramas de los gradientes.

$$B_{N_{celdas}} = [h_0, h_1, ..., h_{N_{celdas}}] \tag{2.39}$$

$$h_{N_{bins}} = [x_0, x_1, ..., x_{N_{bins}}] \tag{2.40}$$

Una vez terminado el proceso de extracción de características de la imagen mediante HOG, se procede a proporcionar como entrada los datos recopilados en la ecuación 2.38 a la máquina de soporte vectorial.

El SVM utilizado en esta investigación tiene como objetivo estimar la posición del objeto dentro de la imagen, proporcionando como salida las dimensiones de lo que en inglés se conoce como *"Bounding Box"*, el cual es un recuadro que delimita el área del objeto dentro de la imagen. Esto se realiza mediante el uso de la técnica conocida como "desplazamiento de ventanas" (lo cual ya se realiza al momento de desplazar los bloques de celdas B), donde por cada ventana (los bloques de celdas) el SVM dará una clasificación de la presencia de un rostro, que posteriormente, utilizando la técnica de supresión no máxima (non-max suppression, en inglés) y la intersección sobre

unión (IoU, por sus siglas en inglés), se puede proporcionar una estimación de la ubicación del rostro en la imagen.

Mencionado lo anterior, para esta aplicación de detección de rostros, la SVM es utilizada como un clasificador lineal, el cual entregará por cada bloque analizado una clasificación de existencia o no existencia de rostro.

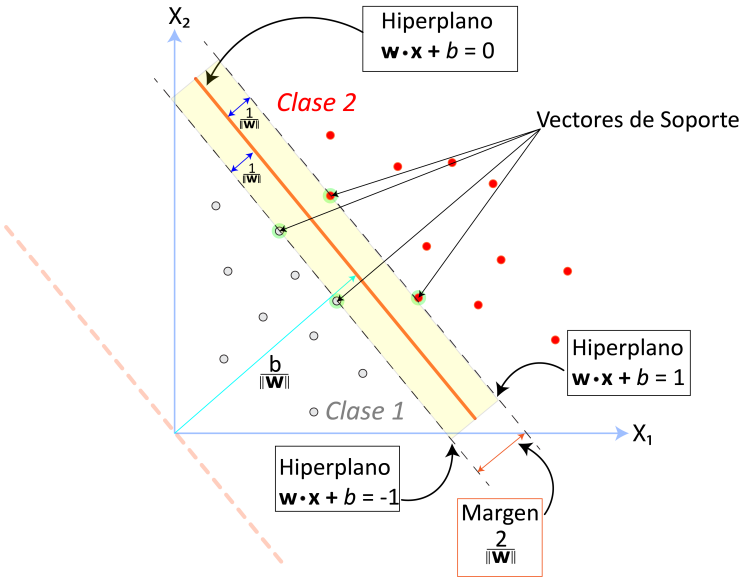

FIGURA 2.12: Descripción gráfica del mapeo de datos en su espacio original, basado en sus características X_1 y X_2 de entrada. El color representa la etiqueta de la clase, además se observa cómo el hiperplano divide el set de datos a la mitad.

Las máquinas de soporte vectorial son algoritmos de ML, específicamente del tipo supervisado, cuyo objetivo es encontrar un hiperplano con dimensiones $\mathbb{R}^{n-1}$ dentro del espacio euclidiano $\mathbb{R}^n$ que delimite de manera *"optima"* los datos en cuestión, logrando una adecuada separación de clases. Esta técnica puede interpretarse como un método de mapeo, dado que el resultado final después del entrenamiento es un conjunto de pesos $\mathbf{w}$ y bias b que forman el hiperplano, el cual logra una separación adecuada del *"espacio"* de entrada original de los datos y su etiqueta correspondiente, como se observa en la Figura 2.12. El SVM es un algoritmo que busca el mayor margen de distancia posible entre los vectores de soporte, ya que métodos como las ANN o LDC encuentran un plano de separación *"aceptable"*, mientras que el SVM busca encontrar el de mayor separación de los datos. Este mayor margen es proporcionado por los vectores de soporte $\mathbf{x_{vs}}$ que se observan en la Figura 2.12 y que están dentro de un círculo sombreado verde (Suárez, 2014).

El proceso completo de una SVM para clasificación lineal se puede observar en la Figura 2.13, donde en el último paso del flujo de datos se muestra

otro ejemplo de mapeo de datos, ádemas se denota que este tipo de representación es para casos de datos 2D.

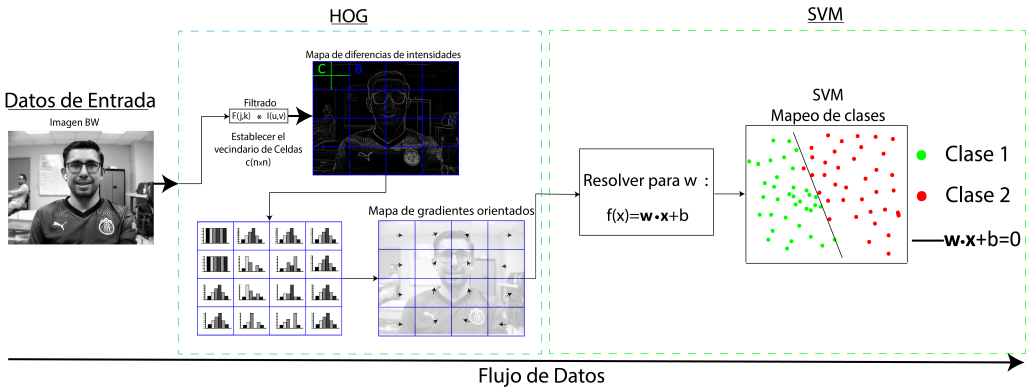

FIGURA 2.13: Proceso de obtención de un detector de objetos mediante SVM.

El proceso de desarrollo de un SVM de clasificación lineal, se comienza definiendo la ecuación 2.41, la cual es la ecuación del hiperplano a aproximar.

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b \tag{2.41}$$

Donde $\mathbf{w}$ son los pesos del SVM a calcular y es conocido como el vector perpendicular al hiperplano, y b es el valor conocido como *"bias"* (en inglés) o sesgo, el cual se obtiene a partir de los vectores de soporte $\mathbf{x_{vs}}$ y es utilizado para definir el desplazamiento en el espacio del hiperplano.

Tomando en cuanta la ecuación 2.41, se toma la siguiente suposición:

$$\mathbf{w} \cdot \mathbf{x_i} + b = 0 \tag{2.42}$$

Donde $\mathbf{x_i}$ es el arreglo de datos de entrada al SVM y se considera que los puntos x que se encuentren sobre el hiperplano hacen que la ecuación 2.41 sea igual a cero. Para el caso más simple de clasificación lineal, la salida del clasificador y las etiquetas de las clases solamente pueden pertenecer a dos valores $y_i \in \{-1, 1\}$. Dadas estas suposiciones de la ecuación 2.41, se pueden derivar las siguientes ecuaciones:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq +1 \quad \text{para} \quad y_i = 1 \tag{2.43}$$

$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \quad \text{para} \quad y_i = -1 \tag{2.44}$$

Estas ecuaciones indican que para las dos distintas clases de los datos, existe un hiperplano asociado que tiene la máxima separabilidad de las clases, como es observado en la Figura 2.12. Además, estos hiperplanos se encuentran sobre los vectores de soporte $\mathbf{x_{vs}}$. Las ecuaciones 2.43 y 2.44 pueden combinarse en una sola definición, la cual se expresa de la siguiente manera:

$$y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \geq 0 \quad \text{para} \quad i = 1, ..., n \tag{2.45}$$

La ecuación 2.45 permite conocer si la clasificación realizada es correcta, esto en base a los parámetros del hiperplano de separación definido. Por consiguiente, se puede definir que, dependiendo del signo que entregue la ecuación del hiperplano 2.41, con sus pesos y bias calculados, la clasificación estará basada únicamente en el signo del resultado. Por lo tanto, la regla de clasificación se expresarse de la siguiente manera:

$$y_l = \text{sgn}\left(\mathbf{w} \cdot \mathbf{x}' + b\right) \tag{2.46}$$

donde $\mathbf{x}'$ son los nuevos valores a clasificar, posteriormente al entrenamiento del SVM, y y_l es la correspondiente etiqueta de clasificación de la entrada $\mathbf{x}'$.

Relacionando las ecuaciones de los tres hiperplano mencionados, estas ecuaciones se deben resolver para $\mathbf{w}$ y posteriormente utilizando los vectores de soporte calcular el sesgo adecuado b del SVM con el objetivo de calcular la máxima separabilidad entre clases. La máxima separabilidad se obtiene mediante la maximización del margen entre los hiperplanos definidos por las ecuaciones 2.43 y 2.44, como se observa en la Figura 2.12.

El margen de separación entre los planos definidos se determina mediante el cálculo de la distancia euclidiana del hiperplano de la ecuación 2.42 al origen $p(0, 0)$ del plano $\mathbb{R}^n$, la cual está definida por la ecuación 2.47.

$$d_1 = \frac{|\mathbf{w} \cdot 0 + b|}{||\mathbf{w}||} = \frac{b}{||\mathbf{w}||} \tag{2.47}$$

Donde $||\mathbf{w}||$ es la norma euclidiana del origen con el vector normal $\mathbf{w}$. Con base en esta distancia, se llega a una estimación del ancho del margen entre los hiperplanos de las clases, el cual corresponde a la suma de las distancias de los hiperplanos con respecto al hiperplano de máxima separabilidad, como se muestra en las siguientes ecuaciones:

$$d_2 = \frac{b - (b - 1)}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||} \tag{2.48}$$

$$d_3 = \frac{(b+1)-b}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||} \tag{2.49}$$

donde d_2 y d_3 son las distancias entre el hiperplano 2.41 y los hiperplanos posicionados sobre los vectores de soporte, definiendo así la separación entre las clases. Por último, el margen de separabilidad estará definido por la suma de las distancias d_2 y d_3, como se muestra en la ecuación 2.50.

$$Margen = \frac{d_2 + d_3}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||} \tag{2.50}$$

Por lo tanto, las SVM están regidas por las siguientes reglas de restricción:

$$\begin{cases} Maximizar & \frac{2}{||\mathbf{w}||} \\ s.a. & y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right) \geq 0 \end{cases} \tag{2.51}$$

El conjunto de ecuaciones en 2.51 se reescribe por conveniencia matemática como:

$$\begin{cases} Minimizar & \frac{1}{2}||\mathbf{w}||^2 \\ s.a. & y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right) \geq 0 \end{cases} \tag{2.52}$$

Para lograr la minimización de $||\mathbf{w}||^2$, se emplean métodos matemáticos como los multiplicadores de Lagrange, los cuales transforman el entrenamiento de una SVM en un problema de programación cuadrática convexa, lo cual incrementa la complejidad del algoritmo. Considerando la implicación de optimizadores se tienen ya programas pre-establecidos para el entrenamientos de máquinas de soporte vectorial, donde por ejemplo, un caso de clasificación binaria es representado por el algoritmo 1.

El algoritmo 1 corresponde a un modelo de entrenamiento para una Máquina de Vectores de Soporte (SVM), el cual emplea multiplicadores de Lagrange para optimizar la obtención del vector de pesos $\mathbf{w}$. En este modelo, los parámetros s y m representan la cantidad de vectores de soporte x_{vs}, mientras que α_i denota los multiplicadores de Lagrange asociados a cada muestra de entrenamiento. Por su parte, $\mathbf{K}$ corresponde al kernel, una función que define el espacio de transformación en el que se maximiza la función objetivo L.

Dado que este método involucra conceptos avanzados de optimización convexa y teoría dual, se recomienda consultar referencias especializadas para una comprensión más profunda. En particular, los trabajos de Suthaharan y Suthaharan, 2016; Suárez, 2014; Burges, 1998 ofrecen una descripción detallada del uso de multiplicadores de Lagrange, así como de las condiciones

Algoritmo 1 Algoritmo de Máquina de Soporte Vectorial (SVM)

Entrada: Conjunto de datos $(\mathbf{x}_i, y_i)$
Salida: Hiperplano óptimo $\mathbf{w}, b$
Computar el kernel: $K = y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$
Resolver el problema de optimización:
Maximizar $L = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \alpha^T K \alpha$
sujeto a:
    $\alpha_i \geq 0$
    $\sum_{i=1}^{l} \alpha_i y_i = 0$
Calcular el vector de pesos:
$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$
Determinar los vectores de soporte:
Seleccionar $\mathbf{x}_i$ tales que $\alpha_i > 0$
Calcular el bias b
$b = \frac{1}{N_{vs}} \sum_{vs \in S} \left(y_{vs} - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_{vs} \right)$
Clasificar nuevos puntos:
Para un nuevo $\mathbf{x}'$, calcular $y' = \text{sgn}(\mathbf{w} \cdot \mathbf{x}' + b)$

de Karush-Kuhn-Tucker (KKT), fundamentales para garantizar la solución óptima del problema.

Para conocer un ejemplo de menor complejidad matemática pero que inicio con la teoría de hiperplanos, puede consultarse el libro de Cristianini y Shawe-Taylor, 2004, donde se describe el *perceptrón de Rosenblatt*. Este constituye el primer clasificador lineal basado en la teoría de separación de datos mediante hiperplanos. Su entrenamiento sigue un procedimiento iterativo análogo al de una neurona artificial: el vector de pesos $\mathbf{w}$ se ajusta progresivamente en función de los errores de clasificación durante el aprendizaje.

Arboles de Regresión

Posteriormente al proceso de detección de rostros mediante un clasificador basado en Máquinas de Vectores de Soporte (SVM), se emplea un algoritmo conocido Ensamble de Árboles de Regresión (*Regression Trees*, RT por sus siglas en inglés). Estos modelos son seleccionados debido a su alta eficiencia computacional en tareas de estimación, como lo señalan los autores en Kazemi y Sullivan, 2014.

Los árboles de regresión, tradicionalmente utilizados en minería de datos para el descubrimiento de patrones, también se consideran algoritmos de aprendizaje automático supervisado, ya que permiten resolver problemas tanto de regresión como de clasificación. En función del contexto y el nivel

de abstracción requerido, pueden ser integrados como parte de un sistema de inteligencia artificial, como se ilustra en la Figura 2.9.

En el presente trabajo, los árboles de regresión son utilizados para estimar la posición de las características faciales (*facial landmarks*, FL por sus siglas en ingles), es decir, puntos clave en la estructura del rostro. El algoritmo predice la ubicación de estos puntos en coordenadas de píxeles dentro de la región de la imagen correspondiente al rostro detectado.

La estructura de los árboles de regresión es jerárquica y se asemeja a la de un árbol binario, en el cual, a partir de una raíz, se generan nodos internos que dividen los datos con base en condiciones de umbral sobre las características de entrada, hasta alcanzar nodos hoja. Estas divisiones continúan de manera recursiva hasta cumplir un criterio de parada, como una profundidad máxima o un número mínimo de muestras por nodo.

Algoritmo 2 Entrenamiento de un Árbol de Regresión con Criterio MSE - Parte 1

Entrada: Conjunto de datos (X, y), profundidad máxima $d_{\text{máx}}$, mínimo de muestras por nodo $n_{\text{mín}}$
Salida: Árbol de regresión entrenado
if $d_{\text{máx}} = 0$ **o** número de muestras $< n_{\text{mín}}$ **then**
    Retornar: Hoja con valor = media de y
end if
Inicializar: Mejor división *mejor_divisin* $\leftarrow$ Ninguna
Inicializar: Mejor error *error_ms_bajo* $\leftarrow \infty$
for cada característica f en el conjunto de datos X **do**
    for cada umbral t en los valores posibles de f **do**
        Dividir los datos:
            Dividir los datos en (X_L, y_L) y (X_R, y_R) usando el umbral t, es decir, para $f_i < t$, (X_L, y_L) y para $f_i \geq t$, (X_R, y_R)
        Calcular el error para cada división:
            Calcular la media de los valores en cada división:
            $\hat{y}_L = \frac{1}{|y_L|} \sum_{i \in L} y_i$
            $\hat{y}_R = \frac{1}{|y_R|} \sum_{i \in R} y_i$
        Cálculo del error cuadrático medio (MSE):
            El error en cada división se calcula como:

$$\text{MSE}_L = \frac{1}{|y_L|} \sum_{i \in L} (y_i - \hat{y}_L)^2$$

            Similarmente, para el lado derecho:

$$\text{MSE}_R = \frac{1}{|y_R|} \sum_{i \in R} (y_i - \hat{y}_R)^2$$

    end for
end for

El algoritmo 2 que se presenta a continuación, ilustra un ejemplo básico de un Árbol de Regresión (RT), en el cual se utiliza el cálculo del error cuadrático medio (MSE) como criterio para determinar si una partición de los datos de entrada es adecuada. A partir de ello, se continúa de forma iterativa generando nuevas ramas hasta alcanzar la profundidad o condiciones de parada previamente definidas.

Algoritmo 3 Entrenamiento de un Árbol de Regresión con Criterio MSE - Parte 2

1: **Calcular el error total:**

$$\text{MSE} = \frac{|y_L|}{|y_L| + |y_R|}\text{MSE}_L + \frac{|y_R|}{|y_L| + |y_R|}\text{MSE}_R$$

2: **if** MSE $<$ *error_ms_bajo* **then**
3: *error_ms_bajo* $\leftarrow$ MSE
4: *mejor_divisin* $\leftarrow (f, t)$
5: **end if**
6: **if** no se encontró una división válida **then**
7: **Retornar:** Hoja con valor = media de y
8: **end if**
9: **Aplicar la mejor división:** Dividir los datos en (X_L, y_L) y (X_R, y_R) usando *mejor_divisin*
10: **Entrenamiento recursivo:**
11: *subrbol_izquierdo* $\leftarrow$ entrenar recursivamente sobre (X_L, y_L) con $d_{\text{máx}} - 1$
12: *subrbol_derecho* $\leftarrow$ entrenar recursivamente sobre (X_R, y_R) con $d_{\text{máx}} - 1$
13: **Retornar:** Nodo de decisión con característica f, umbral t, subárbol izquierdo = *subrbol_izquierdo*, subárbol derecho = *subrbol_derecho*

Este proceso iterativo basado en RT es implementado por Kazemi y Sullivan, 2014 para estimar las 68 características faciales mediante un ensamble de árboles de regresión en cascada. En este enfoque, en cada etapa del modelo (o división), se entrena un nuevo regresor R_t, el cual ajusta las posiciones de los 68 puntos faciales en función de las dimensiones de la imagen $I(u, v)$.

El algoritmo 4 expresa el proceso general para obtener las 68 FL y establece que cada regresor R_t genera un vector de desplazamiento que ajusta la posición relativa de los puntos faciales actuales. Estas correcciones se basan en las diferencias de intensidad en los píxeles cercanos a las ubicaciones actuales de los landmarks. Gracias a su naturaleza iterativa, el proceso continúa hasta completar T etapas, reduciendo progresivamente el error y refinando la estimación de la forma.

Algoritmo 4 Estimación de 68 puntos faciales mediante árboles de regresión en cascada

1: **Entrada:** Imagen I con un rostro detectado
2: **Salida:** Forma estimada final $\mathbf{S}_T$ con 68 landmarks
3: **Inicialización:** Asignar $\mathbf{S}_0 \leftarrow$ forma promedio (mean shape) alineada al rostro detectado
4: **for** $t = 1$ hasta T **do**
5: **Extracción de características locales:**
6: Para cada landmark en $\mathbf{S}_{t-1}$, extraer diferencias de intensidad:

$$\text{features}_t = \left\{ I(p_i^{(t-1)}) - I(p_j^{(t-1)}) \right\}_{i,j}$$

donde $p_i^{(t-1)}$ son píxeles relativos a los landmarks actuales.
7: **Predicción de desplazamientos:**

$$\Delta \mathbf{S}_t = R_t(\text{features}_t)$$

donde R_t es el regresor basado en árboles en la etapa t.
8: **Actualización de la forma:**

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \Delta \mathbf{S}_t$$

9: **end for**
10: **Retornar:** $\mathbf{S}_T$ como la estimación final de los 68 puntos faciales.

Finalmente, es importante mencionar que el enfoque descrito por Kazemi y Sullivan, 2014 también incorpora criterios adicionales, como el establecimiento inicial de los puntos, estrategias específicas para la partición de los datos, y técnicas para la evaluación de los regresores. Por lo que se recomienda revisar el trabajo original del autor para un análisis profundo.

2.2.2. IA: Clasificadores

Las herramientas de Inteligencia Artificial (IA) son técnicas matemáticas avanzadas utilizadas en una amplia variedad de aplicaciones, donde desempeñan distintas tareas. Toman especial relevancia en el área de visión por computadora (CV), particularmente en la clasificación y detección de objetos en imágenes. Como se muestra en la Figura 2.9, existe una variedad de métodos dentro de la IA, cuya selección depende de la complejidad, la cantidad de variables y las restricciones establecidas por las bases de datos en la aplicación a investigar.

Los algoritmos de IA para clasificación tienen como objetivo reconocer

patrones a partir de la información suministrada, ya sea mediciones de señales o píxeles de imágenes, donde se les entrena mediante etiquetas que indican los patrones que deben identificar para automatizar procesos de clasificación. Para casos en los que la información puede separarse linealmente, o mediante técnicas de preprocesamiento como PCA (Análisis de Componentes Principales), LDA (Análisis Discriminante Lineal) o técnicas de promediado, algoritmos de bajo costo computacional pueden ser adecuados para la clasificación, tales como clustering (agrupamiento), Naive Bayes (NB) o K-means. Una adecuada separación de los datos facilita el uso de algoritmos de minería de datos como clasificadores, ya que estos pueden predecir la clase más cercana para un dato que no pertenezca explícitamente a la base de datos.

Para aplicaciones de mayor complejidad en cuanto a la separabilidad de los datos, se utilizan algoritmos más avanzados tanto en profundidad como en complejidad matemática (observese la Figura 2.9), como las redes neuronales (NN), las máquinas de soporte vectorial (SVM) o las redes neuronales convolucionales (CNN), que incorporan mecanismos de aprendizaje como funciones de activación y funciones de error. En el análisis de imágenes, se emplean algoritmos de aprendizaje automático (ML) y aprendizaje profundo (DL), debido a que, por su naturaleza, están diseñados para procesar grandes cantidades de datos.

A continuación, se presentarán clasificadores que pueden implementarse en diversas tareas.

Agrupamiento (Clustering)

Los algoritmos de agrupamiento (clustering) son algoritmos del tipo no supervisado, dado que estos son utilizados para tareas de descubrimiento de patrones o creación de conjuntos y subconjuntos dentro de una base de datos, la cual carece de un etiquetado de clasificación. Estos algoritmos son simples de ejecutar ya que su objetivo solo es la agrupación de los datos y permite con ciertas limitaciones su ejecución como clasificadores simples pero efectivos, pero resulta contraproducente ejecutarlos ante una base de datos de grandes dimensiones (mayor cantidad de variables o atributos) (Hart, Stork y Wiley, 2001). A continuación se presentan algunos métodos de agrupación para el etiquetado de datos.

K-medias(K-means)

El algoritmo de K-medias es un método de minería de datos utilizado para la agrupación (clustering), que permite identificar patrones o clases en función de la dimensionalidad y la separabilidad de la información disponible. Cualquier conjunto de datos puede representarse gráficamente dependiendo de la cantidad de variables que lo componen, lo cual permite tratar cada elemento de la base de datos como un punto en un espacio n-dimensional. El algoritmo calcula una cantidad K de medias o centroides dentro del conjunto de datos en base a la distancia que hay entre cada uno con respecto a la media y agrupa los elementos en función de la cercanía a cada uno de estos centroides establecidos Ahmed, Seraj e Islam, 2020.

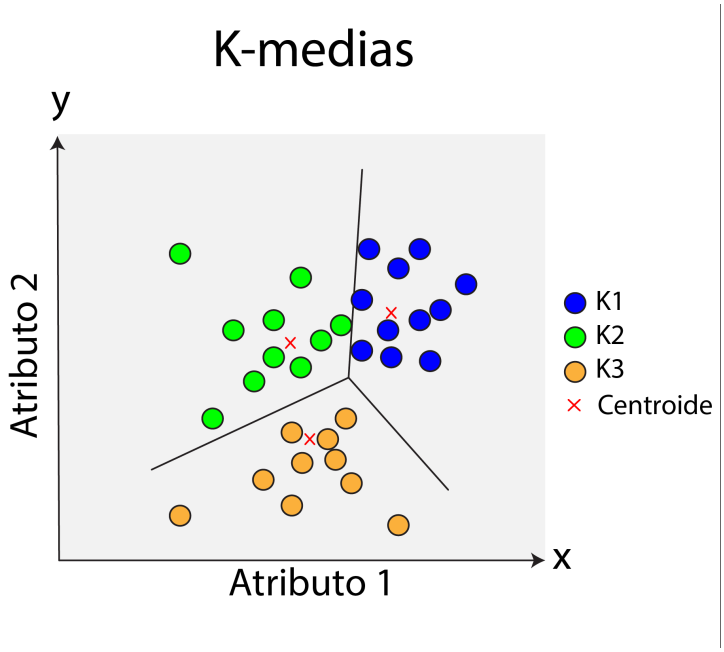

FIGURA 2.14: Algoritmo de K-meadias para el proceso de clustering jerarquico, se observa que por cada centroide establecido se identifican una cantidad K de vecinos que pertenecen a una misma clase.

El algoritmo de K-medias, se emplea en función del calculo de distancias, donde frecuentemente se utiliza la distancia euclidiana, la cual se observa en la ec.2.53.

Distancia Euclidiana

$$d_{\text{Eucl}}(\mathbf{x}, \mathbf{c}) = \sqrt{\sum_{i=1}^{n} (x_i - c_i)^2} \tag{2.53}$$

Donde:

- $\mathbf{x} = [x_1, x_2, \ldots, x_n]$: vector que representa los puntos de datos.
- $\mathbf{c} = [c_1, c_2, \ldots, c_n]$: vector de centroides generados por el algoritmo.
- n: número de dimensiones o características del espacio.

No obstante, el algoritmo de agrupamiento no se limita solamente al cálculo de distancia euclidiana, ya que existen varias ecuaciones de distancia que también pueden ser implementadas, las cuales se muestran a continuación:

Distancia Manhattan

$$d_{\text{Man}}(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{n} |x_i - c_i| \tag{2.54}$$

Donde:

- $\mathbf{x} = [x_1, x_2, \ldots, x_n]$: vector que representa los puntos de datos.
- $\mathbf{c} = [c_1, c_2, \ldots, c_n]$: vector de centroides generados por el algoritmo.
- n: número de dimensiones o características.

Distancia de Minkowski

$$d_{\text{Mink}}(\mathbf{x}, \mathbf{c}) = \left(\sum_{i=1}^{n} |x_i - c_i|^p \right)^{1/p} \tag{2.55}$$

Donde:

- $\mathbf{x} = [x_1, x_2, \ldots, x_n]$: vector que representa los puntos de datos.
- $\mathbf{c} = [c_1, c_2, \ldots, c_n]$: vector de centroides generados por el algoritmo.
- p: parámetro de orden de la distancia. Si $p = 1$, equivale a la distancia Manhattan; si $p = 2$, a la distancia Euclideana.
- n: número de dimensiones del espacio de características.

Distancia de Mahalanobis

$$d_{\text{Mah}}(\mathbf{x}, \mathbf{c}) = \sqrt{(\mathbf{x} - \mathbf{c})^\top \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{c})} \tag{2.56}$$

Donde:

- $\mathbf{x} = [x_1, x_2, \ldots, x_n]$: vector que representa los puntos de datos.
- $\mathbf{c} = [c_1, c_2, \ldots, c_n]$: vector de centroides generados por el algoritmo.
- $\mathbf{\Sigma}$: matriz de co varianza de las variables del conjunto de datos.

Estas ecuaciones de distancia son de utilidad para distintas aplicaciones, dado que para datos adecuadamente separados, bajo la misma escala se utiliza la distancia euclidiana, por ejemplo, en reconocimiento de patrones en imágenes normalizadas en escalas de grises. Para casos donde existen valores aleatorios de mayor magnitud que la media o el limite superior de los datos, se puede utilizar la distancia Manhattan, la cual ofrece mayor robustez ante valores extremos. Para casos donde existe correlación entre datos o se utilizan variables de diferentes escalas se puede utilizar el cálculo de la distancia Mahalanobis, la cual al incluir la matriz de co-varianza de los datos permite adaptarse a las variaciones.

El algoritmo K-medias, es considerado uno de los de mayor presencia en el ámbito de minería de datos y descubrimiento de patrones Ahmed, Seraj e Islam, 2020, esto dado a las características del algoritmo, el cual se caracteriza por tener una complejidad matemática sencilla y una rápida implementación. Pero, hay ciertas debilidades que pueden presentarse, como la inicialización aleatoria de centroides que puede provocar el no converger en la clase correcta, además que este algoritmo es dependiente de la magnitud de los datos, esto debido a su complejidad de $O(n^2)$, por lo que al elevar el volumen de muestras en la base de datos provoca que el algoritmo se convierta computacionalmente costoso debido a los cálculos.

Dendogramas

Los dendogramas son un algoritmo de clustering jerárquico utilizado para agrupar distintos atributos, variables o elementos de un conjunto de datos. Este algoritmo sigue un proceso iterativo en el que se calcula la distancia entre cada punto del conjunto, utilizando algunas de las ecuaciones ya conocidas para las distancias: Euclidiana, Manhattan, Mahalanobis, entre otras. A partir de estas distancias, se agrupan los elementos más cercanos para formar

nuevos clusters en cada iteración; repitiendo el proceso hasta integrar todos los elementos en una única estructura. El resultado se representa mediante un gráfico en forma de árbol, donde en cada iteración se combinan grupos y se refleja la cercanía entre ellos, como se observa en la Figura 2.15.

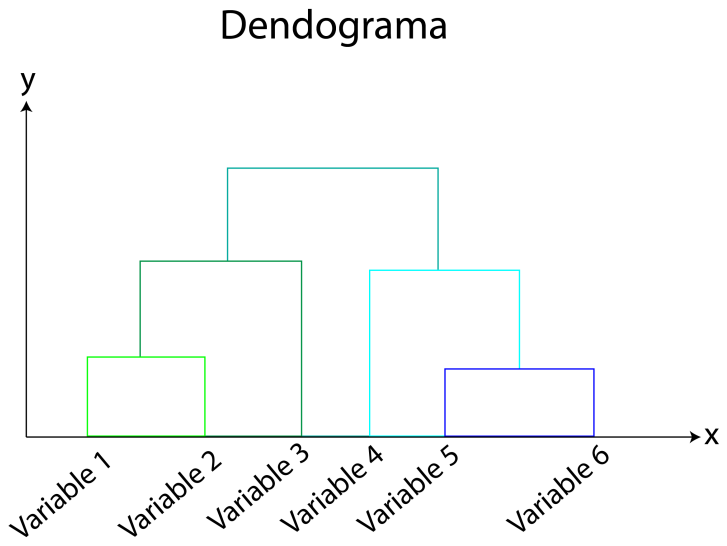

FIGURA 2.15: Dendograma, utilizado para el proceso de agrupamiento, en donde gráficamente se observa como se agrupan los datos de forma específica hasta una clasificación general.

Para el algoritmo 5, se utiliza una condición que establece lo que se conoce como *criterio de enlace* (*linkage criterion*, en inglés), el cual define cómo se calcula la distancia entre clusters en cada paso de la fusión de los datos. Si se selecciona la opción *mínima*, la distan entre clusters se determina por la menor distancia entre dos puntos, uno en cada cluster. En el caso de la opción *máxima*, se utiliza la mayor distancia entre dos puntos, uno en cada cluster. Finalmente, el criterio *promedio* calcula la distancia media entre todos los pares posibles de puntos entre ambos clusters.

Algoritmo 5 Agrupamiento Jerárquico con Distancia Euclidiana

Entrada: Matriz de datos $X \in \mathbb{R}^{n \times m}$ (n muestras, m características), Condición.

Salida: Dendrograma

Inicialización:

Calcular matriz de distancias $D \in \mathbb{R}^{n \times n}$, donde

$$D_{ij} = \sqrt{\sum_{k=1}^{m} (X_{ik} - X_{jk})^2}$$

Crear n clusters: $C = \{\{1\}, \{2\}, \ldots, \{n\}\}$

Inicializar dendrograma: dendro $= [\,]$

while $|C| > 1$ **do**

  Encontrar $(i, j) = \arg\min_{a,b \in C, a \neq b} D(a, b)$

  Fusionar $C_{\text{new}} = C_i \cup C_j$

  Añadir $(C_i, C_j, D(C_i, C_j))$ a dendro

  Actualizar C: eliminar C_i y C_j, añadir C_{new}

  Re-calcular D **para** C_{new}**:**

  for cada cluster $C_k \in C, C_k \neq C_{\text{new}}$ **do**

    if `condición` = "single" **then**

      $D(C_{\text{new}}, C_k) = \min_{x \in C_{\text{new}}, y \in C_k} D(x, y)$

    else if `condición` = ″complete" **then**

      $D(C_{\text{new}}, C_k) = \max_{x \in C_{\text{new}}, y \in C_k} D(x, y)$

    else if `condición` = ″average" **then**

      $D(C_{\text{new}}, C_k) = \frac{1}{|C_{\text{new}}||C_k|} \sum_{x \in C_{\text{new}}} \sum_{y \in C_k} D(x, y)$

    end if

  end for

  Eliminar filas/columnas de C_i y C_j en D, añadir fila/columna para C_{new}

end while

return dendro

Desplazamiento de Media

El desplazamiento de media, o *Mean Shift* en inglés, es un algoritmo de agrupamiento *no paramétrico*, es decir, no requiere conocer de antemano cuántas agrupaciones (clusters) o clases existen. Este algoritmo se basa en la búsqueda de aglomeraciones de datos en función de su densidad (picos de densidad). El cálculo de dicha densidad se realiza mediante la aplicación de kernels de densidad, siendo el kernel gaussiano uno de los más comúnmente utilizados para este propósito.

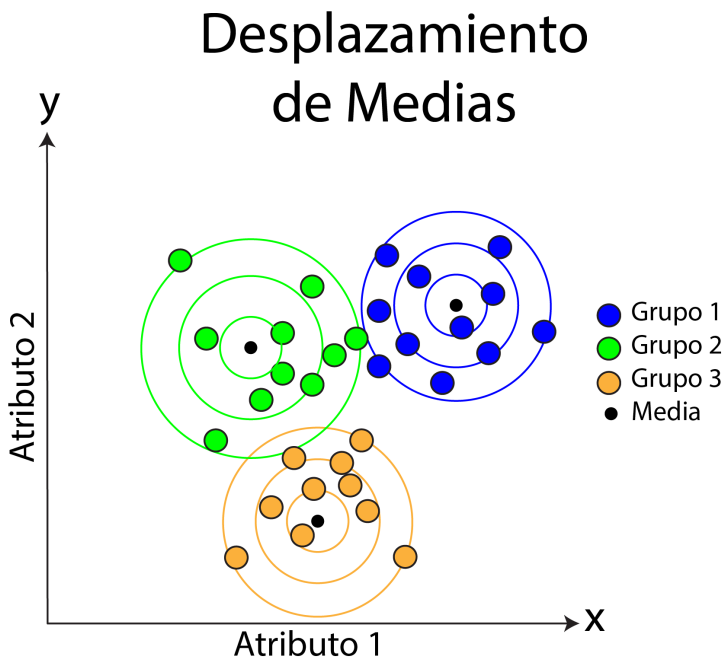

FIGURA 2.16: Ejemplo del algoritmo Mean Shift.

Para analizar el funcionamiento del algoritmo, se utilizará el kernel gaussiano como ejemplo, cuya ecuación se muestra en la ecuación 2.57.

$$K(x, x') = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{2.57}$$

Donde:

- $x, x' \in \mathbb{R}^n$: Coordenadas o vectores en el espacio de entrada.
- $\|x - x'\|^2$: Distancia euclidiana al cuadrado entre las coordenadas x y x'.
- $\sigma > 0$: Parámetro de ancho del kernel o desviación estándar (también llamado *bandwidth* en inglés). Controla cuánto decae la similitud con la

distancia. Comúnmente, el término $2\sigma^2$ se sustituye por la letra h, para generalización y simplificación.

- d: Dimensión de los a datos.

Para construir el kernel gaussiano utilizado para el caso simple de un agrupamiento por K-medias, se toma únicamente la porción exponencial de la distribución normal, la cual se observa en la ecuación 2.58), ya que esta define la cercanía entre los datos mediante un parámetro de distancia conocido como h, que puede interpretarse como el radio de cercanía entre un punto específico $\mathbf{x}$ y el resto de los datos.

Además para un algoritmo simple y rápido en ejecutar se generaliza el denominador del kernel solamente se puede considerar el parámetro h. Lo que genera un algoritmo ideal para agrupar datos sin conocimiento previo de las clases ni del número de agrupaciones.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{h}\right) \tag{2.58}$$

Un ejemplo completo del algoritmo de desplazamiento de medias puede consultarse en el algoritmo 6, donde se emplea el kernel para evaluar la cercanía entre puntos inicialmente, y conforme avanza el algoritmo, se define un nuevo valor de media que se desplaza dentro del espacio de características según cuántos elementos conformen su vecindad. Este comportamiento también se ilustra en la Figura 2.16.

Donde:

- $X \in \mathbb{R}^d$: conjunto de datos en un espacio de d dimensiones.
- $m_i^{(t)}$: estimación actual del modo (pico de densidad) para el punto x_i.
- $\mu_i^{(t)}$: nuevo centro de masa dentro del vecindario definido por h.
- x_j: puntos de datos dentro del vecindario de $m_i^{(t)}$.
- N_h: vecindario de radio h centrado en $m_i^{(t)}$, que contiene los puntos cercanos.

Algoritmo 6 Mean Shift para Clustering

Entrada: Datos $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$, ancho de banda h, tolerancia ϵ
Salida: Clusters y sus modos

1: Inicializar: cada punto x_i es un candidato a modo $m_i^{(0)} = x_i$
2: **for** cada punto x_i **do**
3: **repeat**
4: Calcular el vector de desplazamiento (mean shift):

$$\mu_i^{(t+1)} = \frac{\sum_{x_j \in N_h(m_i^{(t)})} K\left(\frac{||m_i^{(t)} - x_j||}{h}\right) x_j}{\sum_{x_j \in N_h(m_i^{(t)})} K\left(\frac{||m_i^{(t)} - x_j||}{h}\right)}$$

5: Actualizar: $m_i^{(t+1)} \leftarrow \mu_i^{(t+1)}$
6: **until** $||m_i^{(t+1)} - m_i^{(t)}|| < \epsilon$
7: Asignar x_i al modo convergido m_i^*
8: **end for**
9: Fusionar modos cercanos (si $||m_i^* - m_j^*|| < h$)
10: Asignar cada punto al cluster de su modo más cercano

2.2.3. Algoritmo de Vecinos Cercanos (KNN)

El algoritmo de K Vecinos Cercanos (KNN, por sus siglas en inglés) es un algoritmo de inteligencia artificial supervisado, ya que requiere un conjunto de datos con etiquetas conocidas para realizar una comparación de distancias y así tomar una decisión de clasificación o realizar una estimación (regresión). En la construcción de un modelo KNN no se entrena un modelo en el sentido tradicional; en cambio, el conjunto de entrenamiento sirve como base de comparación para evaluar nuevos datos, es decir, se busca la similitud entre un dato desconocido y los elementos almacenados en la base de datos.

Este algoritmo de clasificación guarda cierta similitud con el algoritmo de K-medias presentado en la Sección 2.2.2, ya que ambos se basan en el cálculo de distancias para definir la similitud entre los datos. No obstante, la diferencia principal radica en que K-medias no utiliza etiquetas, sino que las genera de forma automática, mientras que KNN sí cuenta con este conocimiento previo (*a priori*). Por lo tanto, en KNN la similitud calculada permite asignar una clase conocida a un nuevo dato.

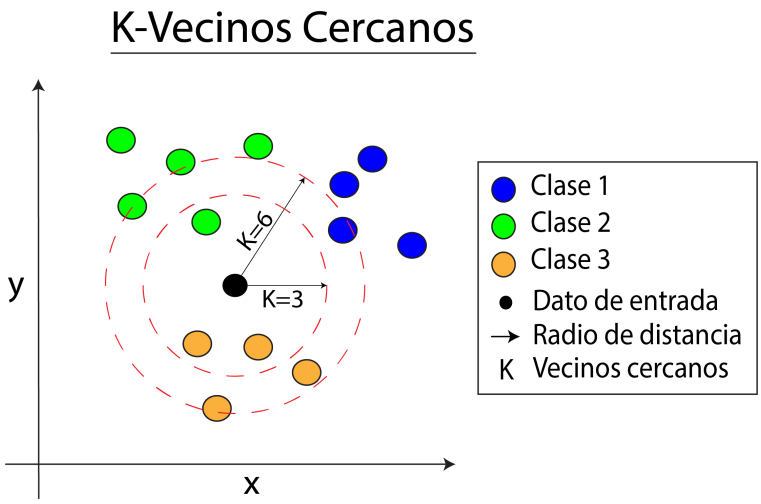

FIGURA 2.17: Descripción gráfica de un algoritmo de KNN, donde se aprecia mediante radios de distancia la cantidad de vecinos cercanos, con los cuales se establecerá la clasificación del valor de entrada.

El Algoritmo 7 muestra el procedimiento mediante el cual se implementa el método KNN. En él se observa que es necesario seleccionar previamente la cantidad de vecinos cercanos al dato de prueba. Un punto importante a considerar es que el número de vecinos, k, debe ser impar para evitar empates en el proceso de votación, lo cual podría afectar la toma de decisiones de clasificación o estimación.

Algoritmo 7 Algoritmo KNN con distancia euclidiana

Entrada: Conjunto de entrenamiento $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, punto de prueba x_q, número de vecinos k
Salida: Clase predicha para x_q

1: **for** cada punto (x_i, y_i) en D **do**
2: Calcular la distancia euclidiana $d_i = \sqrt{\sum_{i=1}^{n}(x_i - x_q)^2}$
3: **end for**
4: Ordenar las distancias d_i en orden ascendente
5: Seleccionar los k vecinos más cercanos
6: Contar las clases de los k vecinos
7: Asignar a x_q la clase más frecuente entre los k vecinos **return** Clase predicha

La ecuación utilizada en el Algoritmo 7 corresponde a la distancia euclidiana, presentada en la ecuación 2.53. Sin embargo, esta métrica puede ser reemplazada por otras, como la distancia de Mahalanobis, Manhattan, del coseno, entre otras, dependiendo de las características del problema y los datos disponibles. Los algoritmos de KNN son simples de ejecutar y traen consigo

ciertas ventajas para casos de clasificado de pocas clases, cuando existe buena separación de los datos, entre otras cosas. Algunas ventajas para el uso de un KNN se enumeran a continuación:

- Simplicidad de ejecución.
- La votación de clasificación o estimacion es simple sin necesidad de comparaciones o uso de cálculos complejos como la distribución de los datos.
- Buenos resultados de exactitud y rapidez manejando un volumen de datos bajo o mediano.

En el mismo ámbito los KNN tienen ciertas desventajas cuando se trata de analizar grandes cantidades de datos , cuando la información no tiene buena separabilidad y es de alta dimensionalidad. Sus desventajas se enumeran a continuación:

- Sensible a cambio de escalas en los datos.
- Es un algoritmo rígido, es decir, ya que solo analiza la base de datos almacenada, se ve limitado a clasificar datos que se encuentren cerca de múltiples clases.
- Ante un volumen elevado de información se vuelve computacionalmente costoso de implementar (debido al cálculo de distancias).

Red Neuronal Artificial (ANN)

Las Redes Neuronales Artificiales (ANN, por sus siglas en inglés) son algoritmos de inteligencia artificial de tipo supervisado, es decir, requieren datos etiquetados durante el proceso de entrenamiento para que el modelo aprenda a predecir salidas correctas a partir de entradas conocidas (Goodfellow et al., 2016).

Una ANN está compuesta por un conjunto de neuronas artificiales (también llamadas perceptrones, basados en el modelo de McCulloch y Pitts) interconectadas entre sí, formando capas que permiten resolver tareas complejas. Se denominan “redes” porque su estructura se inspira en el proceso biológico de comunicación entre neuronas en el cerebro humano. La Figura 2.18 muestra el diseño general de una ANN.

De la Figura 2.18 se puede mostrar que una red neuronal típica consta de tres tipos principales de capas: entrada, ocultas y salida.

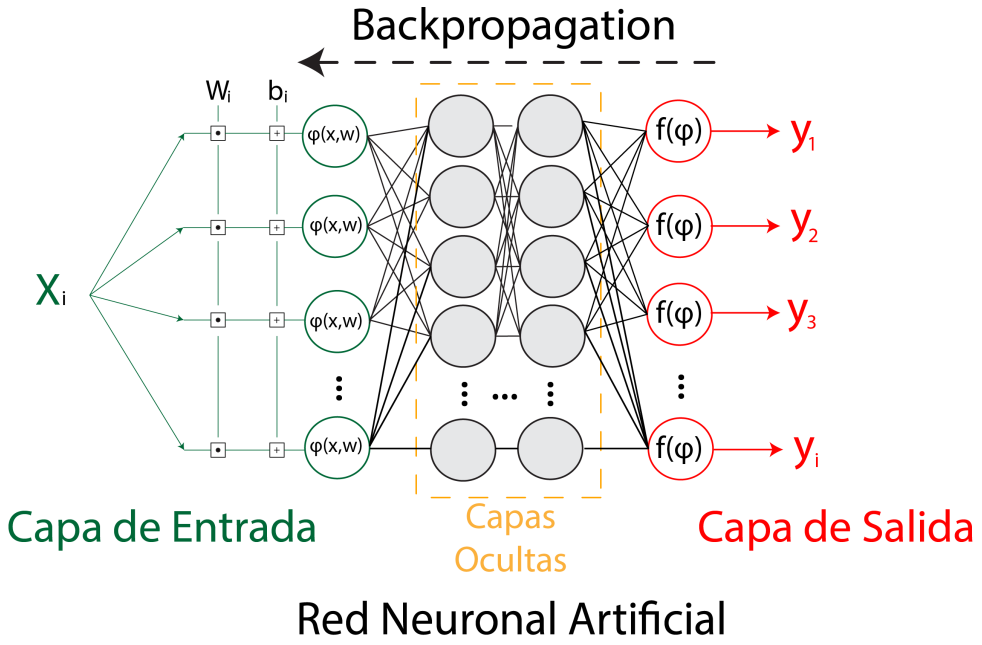

FIGURA 2.18: ANN.

- **Capa de entrada:** está formada por neuronas que reciben los datos del entorno, y su tamaño usualmente corresponde al número de variables o características en el conjunto de datos.
- **Capas ocultas:** (una o varias) determinan la profundidad de la red y están compuestas por un número arbitrario de neuronas que realizan transformaciones intermedias. Estas neuronas están conectadas con pesos ajustables tanto a la capa de entrada como a la de salida.
- **Capa de salida:** contiene las neuronas responsables de producir la predicción final del modelo, ya sea una categoría (en clasificación) o un valor numérico (en regresión).

Ademas, cada neurona esta compuesta por varios elementos: los pesos W, el bias o sesgo b, función de activación $f(\mathbf{x})$, .

- **Pesos (W):** son los valores asociados a cada entrada de una neurona. Constituyen parámetros de entrenamiento que se ajustan durante el proceso de aprendizaje de la red neuronal. Los pesos pueden interpretarse como una matriz de transformación $\phi(\mathbf{x}, \mathbf{W})$, la cual mapea la información de entrada hacia el espacio de salida. También pueden entenderse como la representación del conocimiento aprendido por la red.

- **Sesgo o *bias* (*b*):** es un parámetro adicional de entrada para cada neurona, generalmente inicializado de forma aleatoria. Su función es permitir el desplazamiento del espacio de mapeo $\phi(\mathbf{x})$, otorgando mayor flexibilidad a la red en la representación de funciones.

- **Función de activación *f(ϕ)*:** es la función que opera sobre el resultado del mapeo $\phi(\mathbf{x}, \mathbf{w})$ y el sesgo, generando como salida la activación de la neurona. Su propósito es introducir no linealidad en la red, lo cual permite modelar relaciones complejas entre los datos.

- **Función de error *L(f(ϕ), x')*:** esta función es la encargada de evaluar la salida del sistema, entregando una métrica de error que existe entre el valor real y la salida predicha de la red. La función de error de mayor uso puede ser el error cuadrático medio (RMSE, por sus siglas en inglés).

Como se mencionó en secciones previas, algoritmos supervisados como las redes neuronales artificiales (ANN) y las máquinas de vectores de soporte (SVM) buscan encontrar el hiperplano que mejor separa los datos. Este hiperplano está representado por los pesos **W**, los cuales son aprendidos durante el proceso de entrenamiento.

En el caso de las ANN, dicho entrenamiento se realiza mediante el algoritmo de retropropagación del error, conocido como *backpropagation*. Este método se aplica en redes del tipo *feedforward*, en las cuales los datos de entrada se propagan hacia adelante a través de las distintas capas hasta generar una salida. A esta fase se le denomina *feedforward*.

Posteriormente, se ejecuta el proceso de *backpropagation*, el cual consiste en comparar la salida generada con la etiqueta conocida utilizando una función de error. Con base en este error, se ajustan los pesos de la red de manera inversa calculando el gradiente de error, capa por capa, con el objetivo de minimizar dicha función de error y mejorar el desempeño del modelo. Este es un proceso iterativo el cual requiere comúnmente un número considerable de repeticiones para lograr converger a un conjunto de valores **W** que disminuyan el error de la red neuronal. Este proceso se puede observar de manera detallada en el algoritmo 8.

El entrenamiento de una red neuronal se realiza de manera iterativa, como se muestra en el algoritmo 8. En este proceso se introduce el término de gradiente de error δ, el cual permite propagar el error hacia las capas internas con el objetivo de determinar en qué medida deben ajustarse los pesos en cada capa.

Algoritmo 8 Entrenamiento de una Red Neuronal con Backpropagation

Entrada: Datos de entrenamiento $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, arquitectura de la red, tasa de aprendizaje η, número de épocas E
Salida: Pesos entrenados de la red neuronal

Inicializar: Pesos $W^{[l]}$ y sesgos $b^{[l]}$ aleatoriamente para cada capa l
for época $= 1$ **hasta** E **do**
  for Elementos (x, y) de entrada **do**
    Feedforward:
    for cada capa l de la red **do**
      $\phi^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ ▷ Pre-activación
      $a^{[l]} = f(\phi^{[l]})$ ▷ Función de activación
    end for
    Cálculo del Gradiente de error:
    $\delta^{[L]} = (a^{[L]} - y) \odot f'(\phi^{[L]})$ ▷ Gradiente de error
    for cada capa $l = L-1, \ldots, 1$ **do**
      $\delta^{[l]} = (W^{[l+1]})^{\top} \delta^{[l+1]} \odot f'(\phi^{[l]})$ ▷ Gradiente de cada capa, considerando la propagación del error
    end for
    Actualización de pesos y sesgos:
    for cada capa l **do**
      $W^{[l]} \leftarrow W^{[l]} - \eta \cdot \delta^{[l]} (a^{[l-1]})^{\top}$
      $b^{[l]} \leftarrow b^{[l]} - \eta \cdot \delta^{[l]}$
    end for
  end for
end for
return pesos $W^{[l]}$ y sesgos $b^{[l]}$ de cada capa

Este término es fundamental, ya que representa el principal componente responsable de la actualización de los pesos. A su vez, dicho ajuste está regulado por un parámetro conocido como tasa de aprendizaje η (*learning rate*, en inglés), el cual controla la magnitud de los cambios en los pesos inducidos por el gradiente.

Dado que el gradiente puede depender de múltiples variables, es posible interpretarlo como una superficie en el espacio de parámetros, la cual puede contener varios mínimos locales. En este contexto, la tasa de aprendizaje actúa como un factor de amortiguamiento, permitiendo que el algoritmo realice ajustes más pequeños o más grandes en función de la etapa del entrenamiento.

Para conocer mas sobre el proceso de entrenamiento de una red neuronal, se pueden revisar las siguientes referencias bibliográficas (Ruder, 2016; Rasamoelina, Adjailia y Sinčák, 2020; Cilimkovic, 2015), las cuales dan profundidad a los temas de gradiente de error, funciones de activación, entre otros.

2.2.4. Coincidencia de plantillas

Los algoritmos de coincidencia de plantillas (template matching, en ingles) son técnicas de análisis utilizadas para encontrar patrones en conjuntos de datos como señales, texto e imágenes. Estos métodos encuentran aplicación en dominios como el procesamiento de lenguaje natural, la detección de anomalías en ciberseguridad, el análisis de transacciones y el reconocimiento de estructuras visuales. En el presente proyecto, se emplea la coincidencia de plantillas aplicado a imágenes como mecanismo base para la habilitación de la triangulación, permitiendo la estimación precisa de mediciones tridimensionales a partir de información visual multivista.

Como se menciona en la Sección 2.1, la estimación de coordenadas tridimensionales se fundamenta en el principio de visión estéreo, el cual aprovecha el paralaje generado por la observación de un mismo objeto desde dos vistas ligeramente desplazadas. Una vez que el objeto de interés ha sido identificado mediante los métodos descritos en las Secciones 2.2.1 y 2.2.1, se procede a la etapa de template matching, cuyo objetivo es localizar la posición del mismo patrón en ambas imágenes, este proceso es representado en la Figura 2.19. Este procedimiento permite establecer correspondencias espaciales entre regiones homólogas, lo cual es esencial para aplicar la triangulación y obtener las posiciones 3D con precisión.

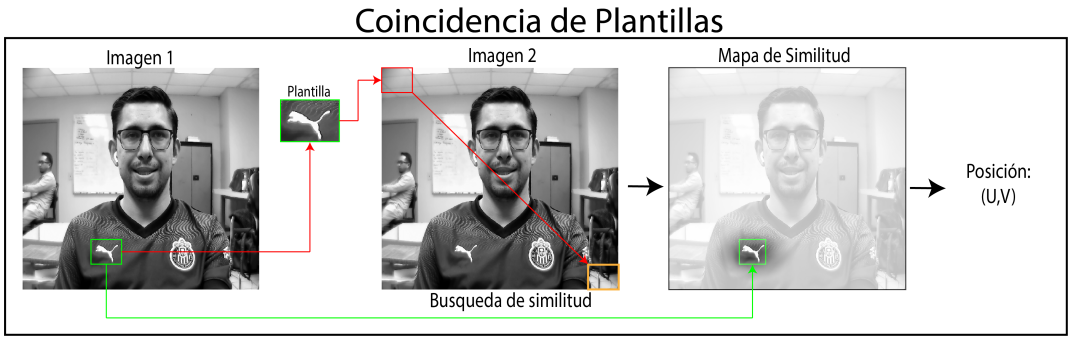

FIGURA 2.19: Proceso general realizado por los algoritmos de coincidencia de plantilla.

En el área de visión por computadora (computer vision, en inglés) se emplean algoritmos de coincidencia de plantillas, entre los cuales algunos de los más reconocidos son: Suma de Diferencias Absolutas (SAD, por sus siglas en inglés), Suma de Cuadrados de Diferencias (SSD, por sus siglas en inglés) y Correlación Cruzada Normalizada (NCC, por sus siglas en inglés), entre otros Kuhn, 1999; Po y Guo, 2007; Yoo y Han, 2009. Este tipo de técnicas resulta ideal para aplicaciones como la visión estereoscópica, donde es fundamental localizar el mismo objeto dentro de un par de imágenes. Esto es

respaldado por Real-Moreno et al., 2023, quien además propone una técnica innovadora de TM conocida como Sustracción de Arreglos de Relaciones (SoRA, por sus siglas en inglés) que reduce el tiempo de ejecución respecto a las técnicas mencionadas previamente.

Los algoritmos de TM en visión por computadora tienen como objetivo identificar pares de puntos correspondientes entre imágenes, mediante el análisis de las intensidades de píxeles. La premisa fundamental es que, si un objeto está presente en ambas imágenes, entonces el algoritmo debe producir un índice de similitud alto (o un valor de disimilitud bajo) en la ubicación correspondiente dentro de la imagen.

Algoritmo de Suma de Diferencias Absolutas (SAD)

El algoritmo de Suma de Diferencias Absolutas (SAD) se implementa a través de la ecuación 2.59:

$$\mathrm{SAD}(u,v) = \sum_{i=0}^{w-1}\sum_{j=0}^{h-1} |I(u+i, v+j) - \tau(i,j)| \tag{2.59}$$

Donde *I(u,v)* representa la imagen a analizar, y $\tau(u,v)$ es la plantilla (template) que se desea localizar. Los parámetros w y h corresponden al ancho y alto de la plantilla, respectivamente.

La ecuación indica que se cálcula la suma de las diferencias absolutas entre los valores de intensidad de los píxeles de la imagen *I(u,v)* y los de la plantilla $\tau(w,h)$, posicionando esta última en la coordenada *(u,v)* de la imagen. Este proceso se repite de forma iterativa para cada posible posición dentro de la imagen, generando un mapa de similitud (Figura 2.19). El valor mínimo de SAD corresponde a la región donde la plantilla se encuentra con mayor coincidencia.

El algoritmo de Suma de Diferencias Absolutas (SAD) se considera computacionalmente eficiente, ya que su cálculo se basa únicamente en operaciones de resta y valor absoluto, lo cual lo convierte en una opción atractiva para aplicaciones en sistemas de visión estéreo o multivista.

A pesar de su baja complejidad, el algoritmo SAD es sensible a variaciones de iluminación y a la presencia de ruido. Esto se debe a que la métrica de similitud está basada en diferencias de intensidad; por tanto, cualquier alteración entre el par de imágenes puede afectar significativamente el resultado final de la correspondencia. Para mitigar estos efectos, es común aplicar técnicas de normalización o filtrado previo a la ejecución del algoritmo,

mejorando así su robustez ante dichas perturbaciones Goel, Nehra y Vishwakarma, 2011.

Otro aspecto a destacar es el nivel de complejidad algorítmica, la cual es cuadrática ($O(n \cdot m)$), debido a que se realizan m operaciones de suma de diferencias absolutas en cada una de las n posiciones posibles dentro de la imagen. Por consiguiente, los tiempos de ejecución aumentan considerablemente al trabajar con imágenes y plantillas de mayor resolución

Sustracción de Arreglo de Relaciones (SoRA)

Dado el papel crucial de los sistemas basados en seguimiento de objetos o análisis de movimiento corporal donde los algoritmos de coincidencia de plantillas (template matching) como SAD son fundamentales, es necesario que estos puedan ejecutarse a un ritmo adecuado que permita el procesamiento en tiempo real.

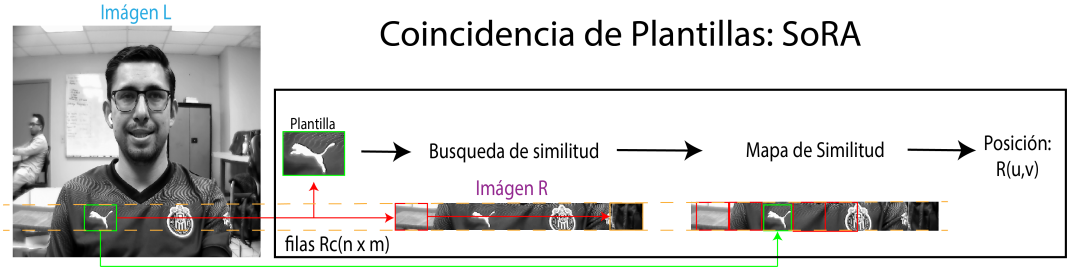

FIGURA 2.20: Representación gráfica del proceso realizado por el algoritmo de coincidencia de plantillas SoRA, donde observamos que el área de interés en la imagen R para el proceso de coincidencia son las filas paralelas a la imagen L de donde se obtiene la plantilla τ.

En este contexto, el autor Real-Moreno et al., 2023 propone un algoritmo de coincidencia de plantillas que reduce significativamente el tiempo de ejecución en comparación con métodos similares. El algoritmo, conocido como *Sustracción de Arreglo de Relaciones* (Subtraction of Relationship Array, SoRA), se basa en SAD, pero en lugar de operar con restas entre matrices completas, introduce el uso de arreglos de relación y aprovecha las características geométricas de un sistema de visión estéreo (SVS). Esto permite disminuir el costo computacional y el tiempo de procesamiento. Una descripción general del algoritmo SoRA se presenta en el Algoritmo 9; para una explicación más detallada, se recomienda consultar el artículo de Real-Moreno et al., 2023.

A diferencia de SAD, SoRA no realiza un análisis completo de la imagen, tampoco procesa plantillas de dimensiones rectangulares $h \times w$, sino

que trabaja exclusivamente con plantillas de dimensiones cuadradas $n \times n$. El procedimiento general del algoritmo es el siguiente:

Algoritmo 9 SoRA

1: Capturar el par de imágenes: **L(**$u \times v$**)** y **R(**$u \times v$**)**
2: Establecer el tamaño de plantilla: **template(**$n \times n$**)**
3: **Seleccionar la plantilla en la imagen izquierda:**

- Inicializar la plantilla τ
- Calcular los promedios por columna de τ:

$$\mu_{T,v_\tau} = \frac{\sum_{i=1}^{n} I_\tau(u_\tau + i, v_\tau)}{n}, \quad v_\tau = 1, 2, ..., n$$

- Calcular el arreglo de relación de la plantilla **Tr** a partir de los promedios:

$$T_{r,1} = \mu_{T,1}$$
$$T_{r,v_{\tau>1}} = |\mu_{T,1} - \mu_{T,v_{\tau+1}}|, \quad v_\tau = 2, 3, ..., n$$

4: **Buscar similitud en la imagen derecha:**

- Seleccionar filas paralelas **Rc**$(n \times m)$ de **R(**$n \times m$**)**, correspondientes a las filas de τ en **L**
- Calcular los promedios por columna de **Rc**, análogamente a μ_{T,v_τ}
- Calcular el arreglo de relación **Rr** a partir de **Rc**, de forma similar a **Tr**
- **Para cada posición** i **en Rr:**
  - Calcular disimilitud:

$$dis_s = \sum_{s=1}^{n} |T_r[s] - R_r[i][s]|$$

5: **Determinar la mejor coincidencia:**

- Fila = posición de las filas utilizadas en la imagen izquierda **L**
- Columna = mín$(dis_s) + n/2$

Donde:

- **L** y **R** son las imágenes izquierda y derecha del sistema de visión estéreo.
- τ es la plantilla extraída de **L**.
- μ_{τ,v_τ} representa los promedios por columna en la plantilla.
- T_r es el vector de relación entre columnas de τ.
- $\mathbf{R}_c$ es la submatriz de **R** que contiene las filas candidatas correspondientes.
- $\mathbf{R}_r$ es el vector de relación calculado para $\mathbf{R}_c$.
- dis_s es el vector de valores de disimilitud entre las relaciones de plantilla y candidatos.

Un algoritmo SoRA mejora los tiempos de ejecución gracias a dos factores clave. Primero, considera las restricciones geométricas impuestas por la visión estéreo, donde se asume que un objeto detectado en una imagen tendrá su correspondencia en las mismas filas (disparidad horizontal); este proceso se explica de manera gráfica en la Figura 2.20. Segundo, SoRA no requiere comparar todos los píxeles de la plantilla, sino que utiliza promedios por columna para generar vectores compactos. Por último, su complejidad algorítmica es de tipo lineal (O(n)), ya que solo se realizan diferencias de arreglos de una dimensión. Esto reduce considerablemente el volumen de datos a comparar, eliminando gran parte del análisis que realiza SAD, sin sacrificar la precisión en contextos controlados, como en la visión estéreo.

Comparación de Técnicas de Coincidencia de Plantilla

Bajo este contexto, donde se requieren algoritmos de coincidencia de plantillas con bajo costo computacional y una velocidad adecuada para aplicaciones de seguimiento de objetos, investigaciones como las de Real-Moreno et al., 2023 y Korman et al., 2013 cobran relevancia en el área de visión por computadora. Asimismo, el trabajo de Hashemi et al., 2016 resulta significativo, ya que compara la complejidad de algoritmos de coincidencia de plantillas conocidos e introduce el uso de redes neuronales convolucionales (CNN) como una alternativa en este campo.

TABLA 2.2: Principales características de algoritmos representativos de coincidencia de plantillas de búsqueda por renglón, basada en información cualitativa documentada.

Método	**Error**	**Velocidad**			**Complejidad**
		Tamaño de plantilla			
		Chica 30<	Mediana <50	Grande <100	
SAD	Bajo	Alta	Media	Baja	Cuadrática
SoRA	Bajo	Alta	Alta	Alta	Lineal
LDS	Bajo	Media	Media	Media	Cuadrática + Logarítmica
Fast-Match-SAD	Medio	Baja	Baja	Baja	Cúbica + Logarítmica

La elección del algoritmo de TM depende de diversos factores, como su rendimiento o su precisión, según las necesidades específicas de la aplicación. A continuación, se presenta la Tabla 2.2, la cual muestra una comparativa entre algoritmos de TM de búsqueda por renglón. Esto se debe a que, como se establece que en visión estéreo las imágenes se encuentran paralelas entre sí, no es necesario realizar una búsqueda completa, sino una búsqueda por renglón.

Esta Tabla 2.2 resume algunos algoritmos de coincidencia de plantillas, proporcionando información cualitativa que permite describir sus características generales y orientar su selección según las necesidades del sistema. No obstante, se recomienda revisar la documentación sugerida para conocer las características específicas de cada algoritmo de TM y sus aplicaciones para las cuales fueron diseñados.

2.3. Matrices de Rotación

Como se ha mencionado en secciones previas, una de las tareas principales es la obtención de la pose facial, mediante las características en 3D calculadas mediante el sistema de visión estéreo. Al ser puntos de tres dimensiones se puede tratar a la tarea de estimacion de la pose como un problema de puntos de perspectiva (PnP, por sus siglas en inglés), los cuales se manejan con una teoría basada en robótica, conocida como matrices de rotación (Kneip, Scaramuzza y Siegwart, 2011).

Cualquier punto dentro de un plano $\mathbb{R}^3$ puede ser modificado mediante distintas transformaciones geométricas, conocidas como matrices de rotación, traslación, escalabilidad, entre otras. Las matrices de rotación son un proceso matemático que permite rotar un punto alrededor de cualquier eje

y trasladarlo dentro de todo el espacio, manteniendo co-linealidad y proporciones entre los puntos. Este tipo de matrices es comúnmente utilizado en aplicaciones de robótica, automatización, visión por computadora, entre otras, en las cuales se requiere modificar posiciones de objetos o gráficos dentro de un plano de referencia real o virtual (Zingoni, Diani y Corsini, 2019; Lee, 1982; Kneip, Scaramuzza y Siegwart, 2011).

Después de calcular los puntos de referencia en 3D, se puede estimar la Pose Facial (Head Pose, en ingles). Debido a que la estimación de los puntos 3D proporciona información espacial, es posible recuperar los ángulos conocidos como Yaw, Pitch y Roll, de cualquier objeto dado. Con la disponibilidad de al menos tres puntos 3D, se puede aplicar la teoría de matrices de rotación para recuperar los ángulos mencionados anteriormente (Slabaugh, 1999). Convencionalmente las matrices de rotación se emplean para rotar puntos dentro de un espacio definido, pero no obstante, también se permiten emplearse de manera inversa y calcular los ángulos de Euler de al menos tres puntos dentro del espacio $\mathbb{R}^3$. Para la el proceso inverso de las matrices de rotación se parte de las siguientes Ecs. 2.60, 2.61 y 2.62.

$$\theta = -sin^{-1}(R_{31}) \tag{2.60}$$

$$\psi = atan2\left(\frac{R_{23}}{cos\theta}, \frac{R_{33}}{cos\theta}\right) \tag{2.61}$$

$$\phi = atan2\left(\frac{R_{21}}{cos\theta}, \frac{R_{11}}{cos\theta}\right) \tag{2.62}$$

$$\mathbf{R_x}(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix} \tag{2.63}$$

$$\mathbf{R_y}(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{2.64}$$

$$\mathbf{R_z}(\phi) = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.65}$$

Donde las constantes R_{ij} son elementos de la Matriz de Rotación, y ψ, θ y ϕ son conocidos como ángulos de Euler, también referidos como Yaw, Pitch y Roll respectivamente. La definición de una rotación en cada eje x, y, y z está

$$\mathbf{R} = \mathbf{R}_x(\psi)\mathbf{R}_y(\theta)\mathbf{R}_z(\phi) \tag{2.66}$$

$$\mathbf{R} = \begin{bmatrix} cos\theta cos\phi & sin\psi sen\theta cos\phi - cos\psi sin\phi & cos\psi sen\theta cos\phi + sin\psi sin\phi \\ cos\theta sin\phi & sin\psi sen\theta cos\phi + cos\psi sin\phi & cos\psi sen\theta cos\phi - sin\psi sin\phi \\ -sen\theta & sin\psi cos\theta & cos\psi cos\theta \end{bmatrix} \tag{2.67}$$

dada por las Ecs. 2.63, 2.64 y 2.65. Donde los ángulos de rotación se expresan en radianes.

Existen soluciones especiales para recuperar los ángulos, dadas las propiedades de las funciones seno y coseno, debido a las limitaciones establecidas para el presente trabajo se ha definido no considerar los casos de $\theta = 0$ o $R_{31} = \pm 1$.

Dado que la multiplicación de matrices no es conmutativa, el orden en el que se realiza la rotación es importante. El orden establecido, es primero alrededor del eje X, posteriormente alrededor del eje Y y finalmente alrededor del eje Z, como se muestra en la Ec. 2.66. Partiendo ahora desde la teoría de matrices, se agrupan estas tres ecuaciones 2.63, 2.64 y 2.65, para formar lo que se conoce como matriz de rotación. La matriz de rotación completa se muestra en la Ec. 2.67.

Es común que se defina una versión simplificada de la matriz de rotación, para simplicidad de notación, esta se muestra en la ecuación 2.68.

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{2.68}$$

Donde R_{ij} son elementos de la matriz de rotación.

Es mediante la ecuación 2.68 que se pueden generar rotaciones de sistemas de puntos en tres dimensiones, lo cual representa un procedimiento común. A partir de estas mismas ecuaciones, también es posible recuperar el ángulo de rotación de un conjunto de puntos si se dispone de su posición inicial en el espacio euclidiano.

Capítulo 3

Procedimiento de Investigación

En este trabajo, se propone un método de estimación de la orientación facial utilizando un sistema de visión estéreo para la captura de imágenes para demostrar la calidad y el rendimiento de la visión estéreo aplicada junto con el aprendizaje automático (ML) para la detección de puntos de referencia en 3D, así como sus capacidades como sistema de seguimiento de la orientación de la cabeza.

El capturar con precisión los movimientos del rostro o la cabeza de una persona permite realizar tareas de gran relevancia como el análisis de emociones, la vigilancia, el modelado facial y la asistencia en la conducción, entre otras. Es en estos ámbitos donde la visión estereoscopica resalta como un medio importante para su uso en temas de visión por computadora, gracias a que no solo perciben la información de los objetos (contorno, color, irregularidades, etc.), es capaz de medir profundidad, demostrando su versatilidad en temas de mediciones tridimensionales, ademas de su habilidad de combinarse con algoritmos de IA, como lo son detección y clasificación de objetos, estimación de coordenadas, entre otras tareas para realizar aplicaciones de mayor complejidad como la estimación de la pose facial (HPE).

El enfoque se basa en aplicar un detector de puntos faciales en ambas imágenes del sistema estéreo, con el objetivo de reemplazar los esquemas tradicionales de coincidencia de plantillas, los cuales presentan dificultades cuando se trata del rostro humano debido a la textura y geometría irregular de la piel.

Posteriormente, se habilita el proceso de triangulación para obtener mediciones tridimensionales. A partir de estos puntos 3D, se calcula la orientación de la cabeza mediante matrices de rotación. Finalmente, se realiza un análisis para demostrar que la precisión del sistema está directamente relacionada con la selección de características faciales clave para el cálculo de los ángulos de Euler.

El enfoque propuesto demuestra que, mediante el principio de triangulación para la obtención de puntos faciales 3D, es posible estimar la orientación del rostro sin necesidad de recurrir al entrenamiento de algoritmos de aprendizaje automático (Machine Learning, ML) como redes neuronales (NN), máquinas de vectores de soporte (SVM), redes convolucionales (CNN), entre otros.

Por último, se realizo un análisis para la implementación del sistema de estimación de pose aplicado a la tarea de asistencia de conducción, para demostración de su utilidad dentro del área de seguridad automotriz.

3.1. Extracción de Características 3D Mediante Visión Estereoscópica

Como se ha mencionado previamente, la visión estereoscópica se logra mediante la implementación de un sistema compuesto por dos módulos de cámaras idénticas dispuestas de forma paralela y separadas por una distancia base conocida. El diseño de una base con las características estructurales adecuadas es fundamental para el correcto funcionamiento del sistema propuesto en esta tesis doctoral.

Dado que el objetivo principal de este trabajo es el análisis de la pose facial en el contexto de sistemas inteligentes de asistencia al conductor, se planteó el desarrollo de un sistema estéreo versátil, capaz de operar tanto en entornos controlados (como un laboratorio o sala cerrada) como en condiciones reales dentro de la cabina de un vehículo.

FIGURA 3.1: Sistema de visión estereoscópica (SVS) utilizado para la experimentación dentro de un vehículo.

El sistema desarrollado, mostrado en la Figura 3.1, fue fabricado mediante técnicas de impresión 3D, lo cual permitió una separación precisa de 10 milímetros entre las cámaras con una tolerancia de $\pm$0,2 milímetros en la impresión, asegurar la rigidez estructural del montaje, y proporcionar flexibilidad de uso. Gracias a esta construcción, el sistema puede instalarse sobre un trípode para su utilización en experimentos de laboratorio, así como montarse fácilmente en el interior de un vehículo para pruebas en condiciones reales.

Cumpliendo con el requisito de fabricación del SVS se procede con el siguiente paso en el desarrollo del método el cual es la programación de los múltiples algoritmos necesarios para lograr la extracción de características en tres dimensiones.

3.1.1. Extracción de Características Faciales

La estimación de la pose facial es un componente fundamental en las aplicaciones de análisis facial, ya que en muchas de ellas, el movimiento del rostro proporciona información crucial sobre el comportamiento de una persona. Por esta razón, se han desarrollado diversas aproximaciones para llevar a cabo esta estimación, las cuales emplean distintos tipos de sensores para medir coordenadas tridimensionales, como cámaras RGB-D, sensores LiDAR o sistemas de luz estructurada. Asimismo, existen alternativas en 2D, basadas en modelos matemáticos complejos, que pueden llegar a ser computacionalmente costosos en función del nivel de precisión requerido.

Un punto en común entre los distintos métodos y sistemas de estimación de pose facial es el uso de características relevantes del rostro para analizar su comportamiento. Esto implica localizar elementos como las cejas, los ojos, la nariz, la boca y el contorno facial en la imagen, y a partir de ellos, estimar la posición del rostro en el espacio tridimensional. Esta tarea se conoce de manera general como detección de objetos, donde para este caso de estudio también se le conoce como detección de características faciales (Facial Landmark Detection, FLD, por sus siglas en inglés), y constituye la base no solo para la estimación de pose facial, sino también para otras tareas relacionadas con el análisis facial.

La extracción de características faciales es una tarea propia de la inteligencia artificial (IA), utilizada como etapa base en procesos como la estimación de pose (Head Pose Estimation, HPE). Este tipo de aplicaciones requiere conocer con precisión la ubicación del rostro humano en cada instante y en cada

imagen procesada. Por tanto, la detección precisa de estas características es de vital importancia para esta investigación.

El proceso de extracción de características se implementa mediante el algoritmo propuesto por Kazemi y Sullivan Kazemi y Sullivan, 2014, descrito en la Sección 2.2.1, el cual se utilizó debido a su eficiencia computacional y versatilidad de aplicación en diferentes lenguajes de programación. Para su ejecución, se integraron dos entornos de programación: Python y LabVIEW.

En primer lugar, se desarrolló una función en Python (mostrada en la Figura 3.2), diseñada para recibir como entrada una imagen $I(x, v)$, analizarla y ejecutar el modelo de IA para la detección de rostros. Una vez identificada la ubicación del rostro en la imagen, se aplica el algoritmo de árboles de regresión para extraer las 68 características faciales, las cuales se retornan al programa principal implementado en LabVIEW.

El código mostrado en la Figura 3.2 corresponde a una función desarrollada en Python para extraer las 68 características faciales del modelo entrenado de *Dlib*. Esta función recibe como entrada una matriz de datos **t**, que representa una imagen codificada. Para procesarla, primero se reconstruye la matriz *RGB* a partir de los valores codificados por canales, lo cual permite su posterior conversión a escala de grises (*GS*), utilizando la librería *OpenCV*.

```
import os
from imutils import face_utils
import dlib
import cv2
from PIL import Image
import numpy
import json
import tensorflow as tf
import numpy as np
from keras.preprocessing import image

# Inicializar el detector facial de Dlib (basado en HOG-SVM)
# despues generar el predictor de caracteristicas faciales
p = 'G:\My Drive\Instrumentacion maestria\
    shape_predictor_68_face_landmarks.dat'
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(p)


def function(t):

#Recuperacion de imagen RGB a partir de t
a = numpy.array(t)
blue = a & 255
green = (a >> 8) & 255
red = (a >> 16) & 255
rgbArray = numpy.zeros((y[0], y[1], 3), dtype=numpy.uint8)
rgbArray[..., 0] = red
rgbArray[..., 1] = green
rgbArray[..., 2] = blue
# Convertir imagen a Escala de Grises

gray = cv2.cvtColor(rgbArray, cv2.COLOR_BGR2GRAY)

# Deteccion de rostros
rects = detector(gray, 0)
y2 = numpy.zeros((68,2))
# Ciclo for para multiples rostros en la imagen
for (i, rect) in enumerate(rects):
 # determinacion de las 68 caracteristicas faciales
 shape = predictor(gray, rect)
 shape = face_utils.shape_to_np(shape)
 #convertir las coordenadas de las 68 marcas a un tipo arreglo
 y2 = y2.astype(int)
 y2 = numpy.ndarray.tolist(numpy.round(y2))
return y2
```

FIGURA 3.2: Función para la extracción de las 68 características faciales del modelo de Dlib disponible en Python.

La conversión a *GS* es necesaria porque el detector de rostros de *Dlib* está diseñado para operar únicamente con imágenes en dicho formato. Una vez detectado el rostro mediante la función `detector()`, se emplea el algoritmo de árboles de regresión `predictor()`, que recibe tanto la imagen en *GS* como la ubicación del rostro, y devuelve un conjunto de 68 puntos de referencia faciales (*landmarks*). Estas coordenadas son devueltas como una lista para su posterior uso dentro del entorno *LabVIEW*.

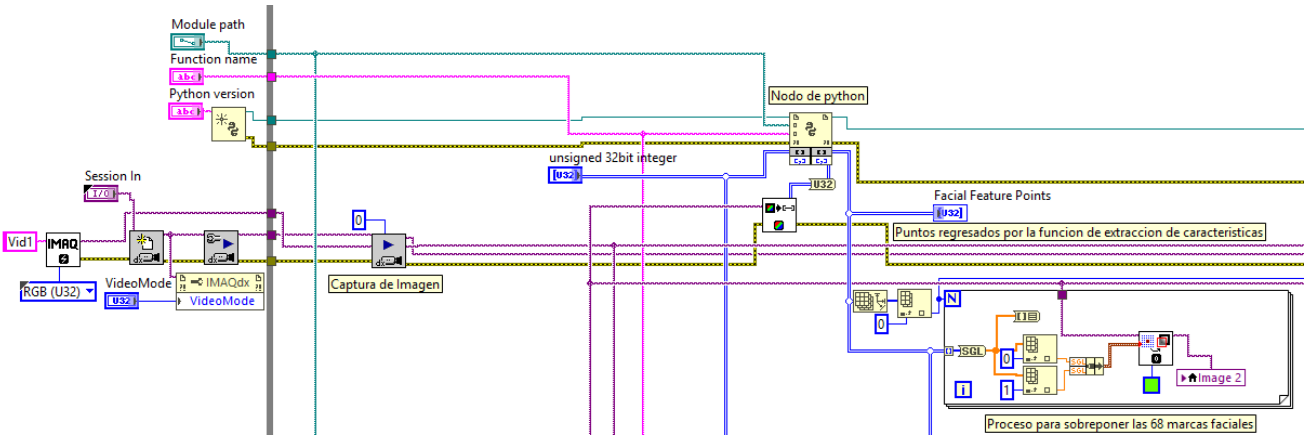

FIGURA 3.3: Sección de código para la comunicación entre Python y Labview para la colocación de las 68 marcas faciales en la imagen capturada, mostrada en la interfaz de usuario.

La Figura 3.3 muestra la sección del código desarrollado en *LabVIEW* para la inicialización, captura y envío de la imagen al nodo de *Python* para su análisis (donde se ejecuta el código mostrado en la Figura 3.2). Posteriormente, se reciben las 68 marcas faciales, las cuales se colocan sobre la imagen para finalmente ser visualizadas en un indicador del panel frontal.

El resultado final de la ejecución del programa de extracción de características representado de manera visual, se ve ilustrado por la Figura 3.4, en la cual se observa la colocación de las 68 marcas faciales.

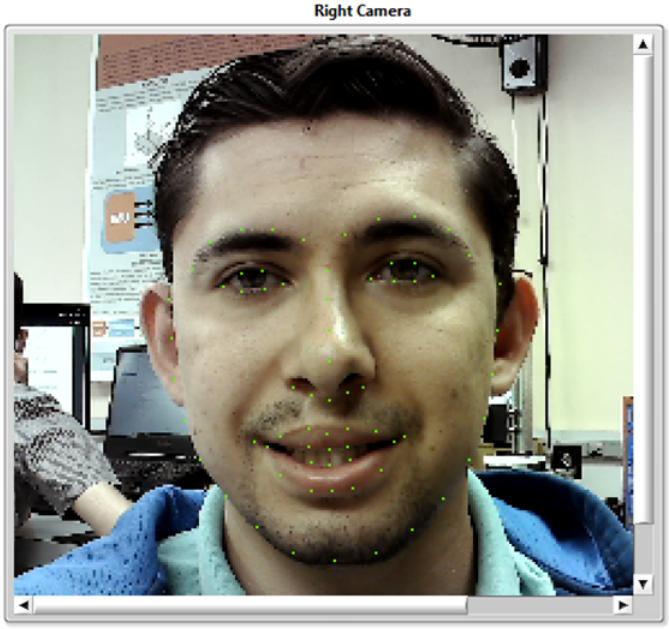

FIGURA 3.4: Ilustración de como se observan las 68 marcas obtenidas por el algoritmo implementado.

Uno de los requisitos fundamentales del sistema de estimación de pose facial HPE es mantener, en todo momento, la detección precisa de las características faciales del rostro dentro de la imagen. Una vez estimadas dichas características mediante el algoritmo de árboles de regresión, se procede a la siguiente fase del sistema, que consiste en realizar el emparejamiento de características en la segunda imagen capturada por el sistema de visión estéreo. A partir de este emparejamiento, se aplica la técnica de triangulación para obtener las mediciones tridimensionales del rostro.

3.1.2. Coincidencia de Plantillas

Como se mencionó en la Sección 2, la técnica de coincidencia de plantillas (Template Matching, TM, por sus siglas en inglés) se utiliza para localizar un mismo patrón o característica en diferentes imágenes. Esta técnica resulta fundamental en la siguiente fase del presente proyecto, ya que uno de los requisitos clave para aplicar la triangulación es haber identificado el mismo punto u objeto en ambas imágenes del sistema estéreo.

En esta etapa, se analizaron distintas técnicas de TM descritas en la Sección 2.2.4, con el objetivo de evaluar su aplicabilidad dentro del flujo general del sistema propuesto. Bajo los criterios establecidos, se evaluaron las características de los métodos SAD, LDS, Fast-Match y, finalmente, SoRA. El análisis incluyó aspectos como la velocidad de ejecución, el error de estimación y

el nivel de complejidad de cada técnica, a fin de seleccionar la más adecuada para el proceso de estimación de pose facial.

El algoritmo inicialmente seleccionado fue SoRA, debido a que está diseñado específicamente para sistemas de visión estéreo con cámaras paralelas, y presenta una baja sensibilidad al tamaño de la plantilla en términos de desempeño. Esto representa una ventaja significativa para los objetivos de este trabajo, cuyo enfoque es lograr estimación de pose facial en tiempo real. La descripción detallada del algoritmo SoRA se encuentra en la Sección 2.2.4.

La técnica de Sustracción de Arreglo de Relaciones (SoRA) se implementa posteriormente a la ejecución del nodo de *Python*, una vez que se han obtenido las características faciales mediante el modelo de regresión, dentro del programa principal de LabVIEW. La Figura 3.5, muestra la sección de código donde se implementa el proceso de SoRA junto con sus Vi's principales.

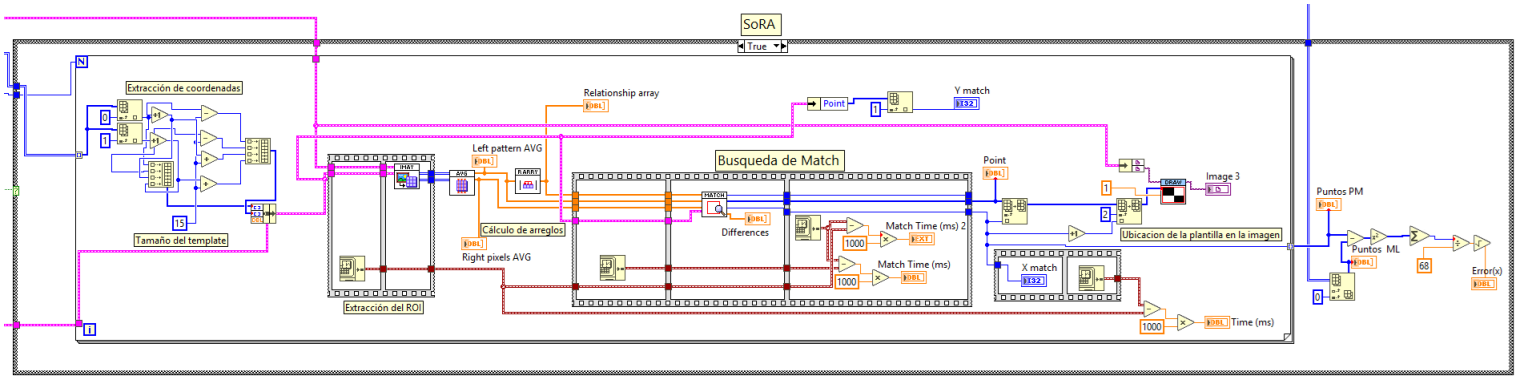

FIGURA 3.5: Sección del código que muestra la comunicación entre *Python* y *LabVIEW* para la colocación de las 68 marcas faciales en la imagen capturada y su visualización en la interfaz de usuario.

Aunque el algoritmo SoRA ofreció resultados con alta precisión en la localización de puntos, también presentó ciertas limitaciones. Factores como las condiciones de iluminación, en presencia de fondos claros en la imagen, los movimientos faciales y la geometría irregular de la piel ocasionaron emparejamientos incorrectos en regiones con alta similitud visual. La Figura 3.6 ilustra el comportamiento del algoritmo SoRA, donde es posible observar marcas ausentes o incorrectamente emparejadas.

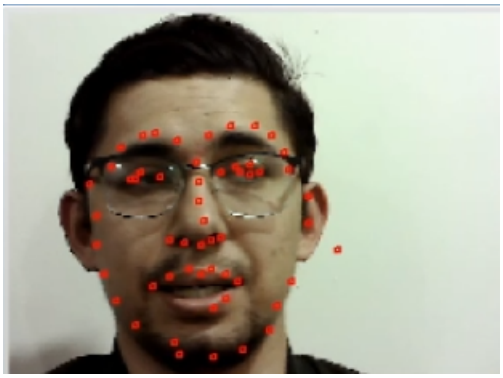

FIGURA 3.6: Resultado del algoritmo SoRA: ilustración de las marcas faciales proyectadas sobre el rostro.

Estas deficiencias motivaron la búsqueda de métodos complementarios para mejorar el proceso de coincidencia de plantillas. A partir de esta búsqueda surgió la idea de utilizar el mismo extractor de marcas faciales (FLD) en ambas imágenes capturadas por el sistema estéreo. Esta propuesta se basa en que el algoritmo FLD ha sido entrenado para colocar consistentemente las 68 marcas faciales, respetando las proporciones relativas entre ellas. Esta característica resulta especialmente útil en un sistema de visión estéreo, ya que, salvo por el valor del paralaje, la posición espacial y la orientación del rostro permanecen prácticamente constantes entre ambas imágenes. Por tanto, esta estrategia se presenta como una solución viable y eficiente frente a las limitaciones del método tradicional de coincidencia de plantillas. El producto final se ve representado por la Figura 3.7, donde vemos la comparación de esta propuesta de solución, demostró ser una opción viable para la fase de emparejamiento de patrones.

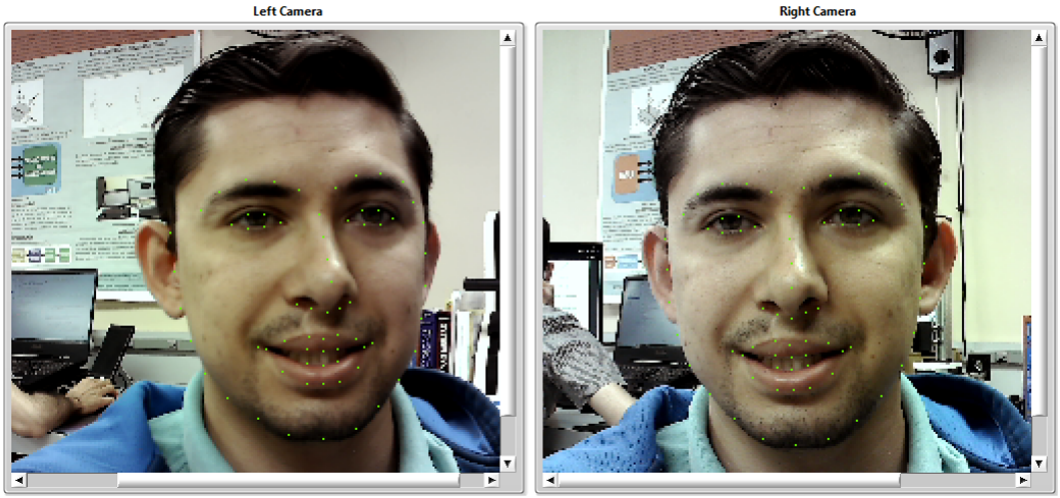

FIGURA 3.7: Ejemplo de la aplicación del algoritmo de FLD de Dlib en las imágenes estéreo.

3.1.3. Aplicación de Triangulación Para Puntos 3D

Al tener disponibles los puntos de características en ambas imágenes, se habilita el siguiente paso del proceso: la implementación de la triangulación.

Conociendo la posición en píxeles de las características faciales, se utilizan las ecuaciones planteadas en la Sección 2.1.2, las cuales requieren los ángulos correspondientes de cada coordenada para su resolución. La triangulación se realiza, en primer lugar, calculando los ángulos correspondientes a la ubicación en píxeles de cada marca estimada en ambas imágenes. La Figura 3.8 muestra la sección del código implementado, en la cual se utiliza un sub-VI que contiene las ecuaciones necesarias para calcular los ángulos representados en la Figura 2.7.

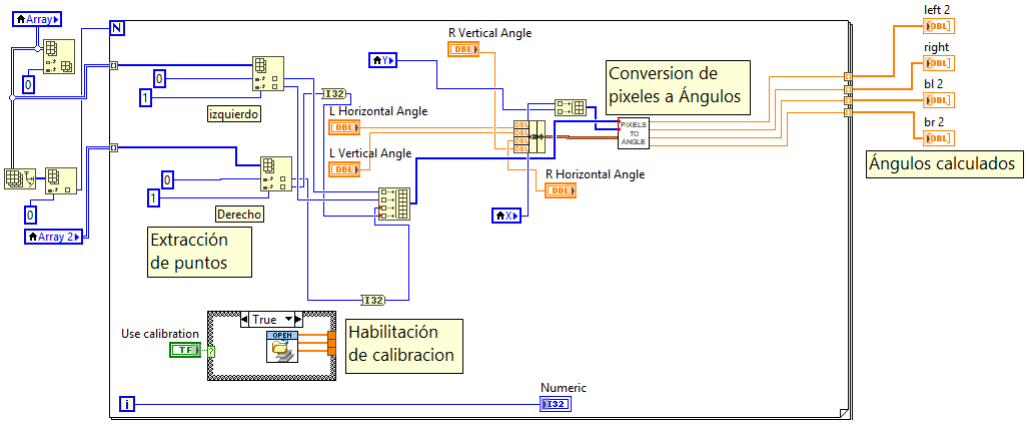

FIGURA 3.8: Cálculo de los ángulos B, C y β.

Una vez obtenidos los valores correspondientes de los ángulos (B, C, β) de las marcas dentro del par de imágenes estéreo, se procede con la ejecución de las ecuaciones 2.12, 2.13 y 2.14, las cuales permiten determinar el valor de profundidad de cada marca, gracias al paralaje existente. La siguiente etapa, por consiguiente, es la ejecución de la sección de código responsable del cálculo de las coordenadas en el espacio $\mathbb{R}^3$. Esta se muestra en la Figura 3.9, donde los tres ángulos necesarios para la ejecución de las ecuaciones, además de un pequeño ángulo de compensación para β, son dados como entrada al sub-VI llamado Triangulation.vi, que contiene la programación de las ecuaciones previamente mencionadas.

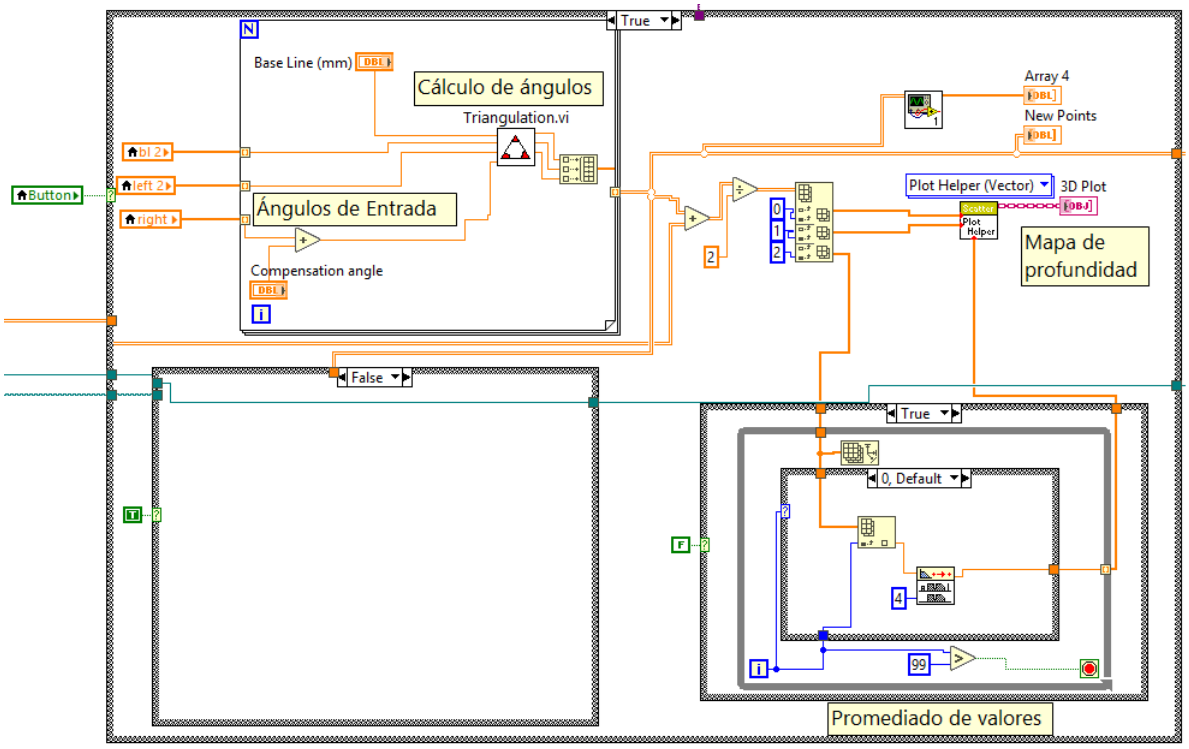

FIGURA 3.9: Sección de código referente al cálculo de coordenadas 3D y su gráfica correspondiente en un plot 3D.

Como se mencionó, el sub-VI (Triangulation.vi) fue programado utilizando el nodo de MATLAB en LabVIEW para ejecutar las ecuaciones planteadas en el proceso de triangulación. La Figura 3.10 muestra cómo fue programado dicho nodo.

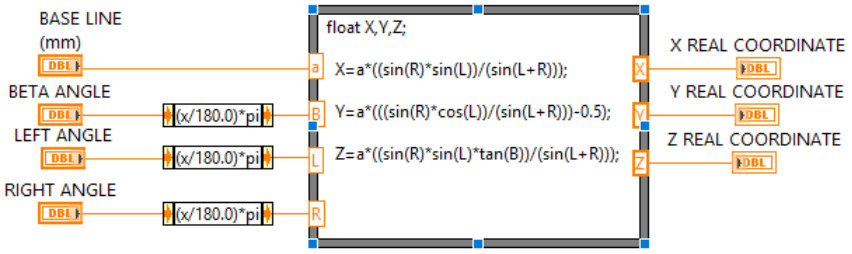

FIGURA 3.10: Ecuaciones de triangulación programadas dentro del nodo de MATLAB en LabVIEW.

Una vez obtenidas las mediciones de profundidad para cada uno de los 68 puntos disponibles, estas son proporcionadas a los siguientes procesos. El principal es el cálculo de la pose facial; los procesos auxiliares incluyen la visualización del mapa de profundidad en la interfaz de usuario, como se muestra en la Figura 3.11. En caso de que se desee mostrar un resultado tridimensional con poca variación en el eje de profundidad, se incorporó una

etapa de promediado de valores, con el fin de reducir la variación y presentar una imagen más fluida.

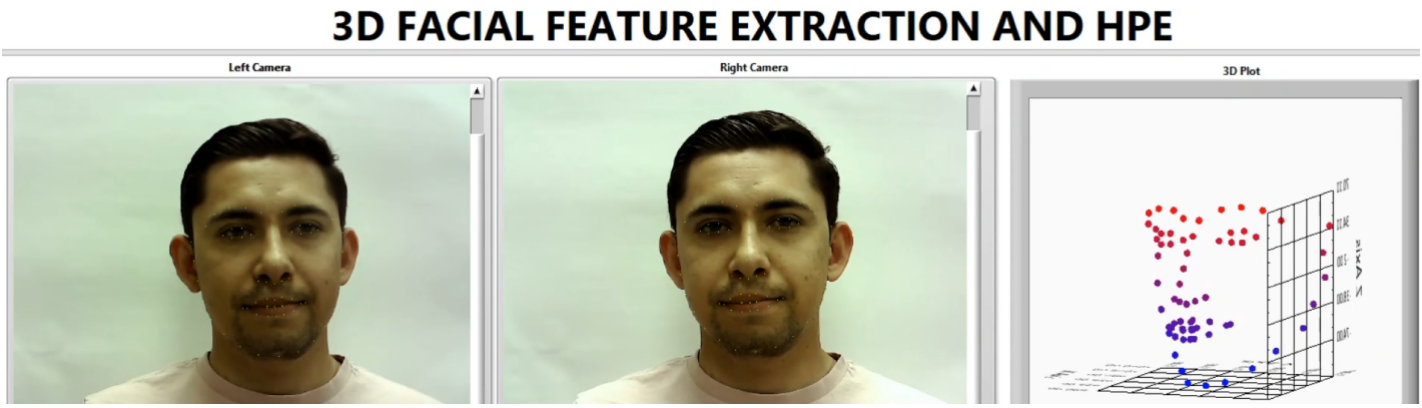

FIGURA 3.11: Ejemplificación de los resultados de la extracción de características faciales y su correspondiente mapa de profundidad, basado en los puntos calculados.

En la Figura 3.11 se pueden observar tres componentes: las dos imágenes estéreo y, en el lado derecho, la gráfica de los puntos 3D obtenidos a través del cálculo por triangulación. El siguiente paso para las coordenadas 3D consiste en utilizar las matrices de rotación para el cálculo de la pose facial utilizando al menos tres puntos de los 68 disponibles. Dado que se dispone de un conjunto de puntos en tres dimensiones, es posible abordar el problema de estimación de pose como un *Problema de Perspectiva de n Puntos* (*PnP*, por sus siglas en inglés).

En el área de Visión por Computadora, esta técnica (*PnP*) se utiliza para estimar la posición y orientación de un objeto, siendo una de sus aplicaciones más comunes la estimación de la ubicación de cámaras en procesos de calibración. Sin embargo, la teoría que sustenta esta técnica permite su aplicación a cualquier nube de puntos disponible, como en este caso, los correspondientes a las características faciales de una persona.

3.2. Estimación de la Pose Facial

3.2.1. Rotación de Puntos

Como se mencionó al final de la sección anterior, al disponer de un conjunto de puntos en tres dimensiones, el problema de estimación de la pose facial puede abordarse como una tarea de *Perspective-n-Point* (PnP) (Baltrusaitis et al., 2018; Amir, Tadas y Louis-Philippe, 2017). Comúnmente, las matrices

de rotación se utilizan como mecanismos dentro de las transformaciones afines, las cuales preservan las relaciones de paralelismo entre los puntos. Por esta razón, dichas ecuaciones son aplicadas en este trabajo para la estimación de la pose.

Convencionalmente, las matrices de rotación permiten transformar puntos en tres dimensiones (por ejemplo, mediante rotaciones, traslaciones o escalados), partiendo del conocimiento de un conjunto de valores base. En este trabajo, se parte de la existencia de dos conjuntos de datos: uno corresponde a un conjunto de puntos de referencia, obtenido al iniciar el sistema, y el otro a las mediciones actuales de las características faciales.

Dado que se conoce la base de referencia de los puntos y su correspondiente transformación, es posible invertir el proceso y estimar las matrices de rotación, resolviendo así los ángulos de Euler asociados, conocidos como *Yaw*, *Pitch* y *Roll*. Para ello se deben encontrar cada coeficiente $R_{i,j}$ de la matriz de rotación.

El proceso para el cálculo de los coeficientes R_{ij} parte de la premisa de que se tienen un conjunto de puntos en el espacio que han sido rotados. Esto en comparación con una plantilla inicial de puntos de referencia 3D. La plantilla inicial $\mathbf{P}$ (de al menos 3 puntos) será el primer conjunto de puntos 3D calculados por el sistema. La Ec. 3.1 muestra que la rotación se realiza multiplicando la matriz de rotación $\mathbf{R}$ por los puntos de la plantilla $\mathbf{P}$.

$$\mathbf{P}' = \mathbf{R} * \mathbf{P} \tag{3.1}$$

Donde $\mathbf{P}'$ son los elementos transformados de la plantilla original.

Luego sustituimos la Ec. 2.68 en la Ec. 3.1, lo que da como resultado la Ec. 3.2.

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \end{bmatrix} = \mathbf{P}' \tag{3.2}$$

Realizando la multiplicación de matrices en la Ec. 3.2, se obtienen las ecuaciones 3.3, 3.4 y 3.5.

$$\mathbf{P}'_{:,1} = \begin{bmatrix} x_1 R_{11} + y_1 R_{12} + z_1 R_{13} \\ x_1 R_{21} + y_1 R_{22} + z_1 R_{23} \\ x_1 R_{31} + y_1 R_{32} + z_1 R_{33} \end{bmatrix} \tag{3.3}$$

$$\mathbf{P'}_{:,2} = \begin{bmatrix} x_2R_{11} + y_2R_{12} + z_2R_{13} \\ x_2R_{21} + y_2R_{22} + z_2R_{23} \\ x_2R_{31} + y_2R_{32} + z_2R_{33} \end{bmatrix} \tag{3.4}$$

$$\mathbf{P'}_{:,3} = \begin{bmatrix} x_3R_{11} + y_3R_{12} + z_3R_{13} \\ x_3R_{21} + y_3R_{22} + z_3R_{23} \\ x_3R_{31} + y_3R_{32} + z_3R_{33} \end{bmatrix} \tag{3.5}$$

Donde x_n, y_n y z_n son los puntos espaciales utilizados como plantilla para calcular los ángulos de rotación. El sistema de ecuaciones obtenido en la Ec. 3.2 ahora se resuelve para los coeficientes de la matriz de rotación necesarios para obtener los ángulos de Euler de la rotación, dado que los únicos datos disponibles son el conjunto original de puntos de referencia 3D $\mathbf{P}$ y los puntos rotados $\mathbf{P'}$. Teniendo $\mathbf{P}$ y $\mathbf{P'}$, las únicas variables desconocidas serán los elementos R_{ij} de la matriz de rotación, y a partir de la Ec. 3.2 se resuelve para $R_{11}, R_{21}, R_{31}, R_{13}$ y R_{33}, los cuales son los elementos necesarios para computar las ecuaciones 2.60, 2.61 y 2.62.

$$R_{11} = \frac{(x'_2y_1 - y_2x'_1)(-z_1y_3 + z_3y_1)}{(x_2y_1 - x_1y_2)(-z_1y_3 + z_3y_1)} \frac{-(x'_3y_1 - y_3x'_1)(-z_1y_2 + z_2y_1)}{-(x_3y_1 - x_1y_3)(-z_1y2 + z_2y_1)} \tag{3.6}$$

$$R_{21} = \frac{(y'_2y_1 - y_2y'_1)(-z_1y_3 + z_3y_1)}{(x_2y_1 - x_1y_2)(-z_1y_3 + z_3y_1)} \frac{-(y'_3y_1 - y_3y'_1)(-z_1y_2 + z_2y_1)}{-(x_3y_1 - x_1y_3)(-z_1y2 + z_2y_1)} \tag{3.7}$$

$$R_{31} = \frac{(z'_2y_1 - y_2z'_1)(-z_1y_3 + z_3y_1)}{(x_3y_1 - x_1y_3)(-z_1y2 + z_2y_1)} \frac{-(z'_3y_1 - y_3z'_1)(-z_1y_2 + z_2y_1)}{-(x_3y_1 - x_1y_3)(-z_1y2 + z_2y_1)} \tag{3.8}$$

$$R_{13} = \frac{(x'_2x_1 - x_2x'_1)(-y_1x_3 + y_3x_1)}{-((-z_1x_3 + z_3x_1)(-y_1x_2 + x_1y_2))} \frac{-(x'_3x_1 - x_3x'_1)(-y_1x_2 + y_2x_1)}{+(-z_1x_2 + x_1z_2)(-y_1x_3 + y_3x_1)} \tag{3.9}$$

$$R_{33} = \frac{(z_2'x_1 - x_2z_1')(-y_1x_3 + y_3x_1)}{-((-z_1x_3 + z_3x_1)(-y_1x_2 + x_1y_2))} \frac{-(z_3'x_1 - x_3z_1')(-y_1x_2 + y_2x_1)}{+(-z_1x_2 + x_1z_2)(-y_1x_3 + y_3x_1)} \tag{3.10}$$

Donde x_n', y_n' y z_n' son los puntos que traen consigo el cambio de pose.

Es en base a este análisis de despeje por el cual se recuperan los ángulos de rotación de un objeto conocido, el cual para esta investigación se considera la rotación de un rostro humano.

Posteriormente al realizar los despejes necesarios de las matrices de rotación para resolver para los elementos R_{ij}, se procede a codificar dichas ecuaciones en el programa desarrollado para la HPE. Las ecuaciones 3.6, 3.7, 3.8, 3.9 y 3.10, se programan por medio de un nodo de MATLAB dentro de LabVIEW, esto para aprovechar la lógica de codificación y operacional de MATLAB y las capacidades graficas de LabVIEW.

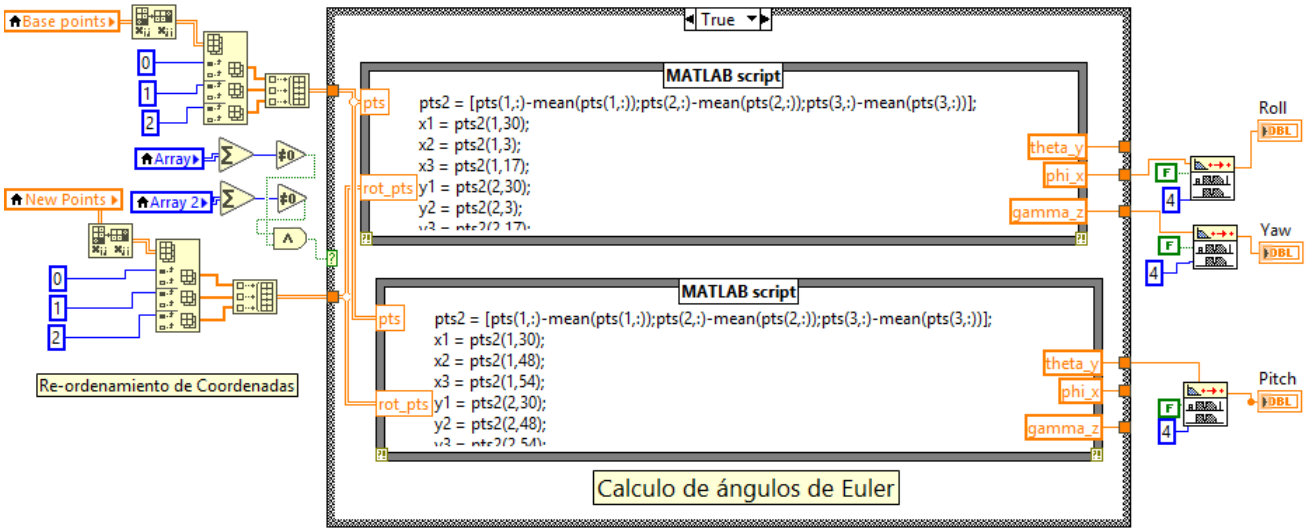

FIGURA 3.12: Ejemplificación de los resultados de la extracción de características faciales y su correspondiente mapa de profundidad, basado en los puntos calculados.

Aquí podemos observar que se reciben dos grupos de coordenadas: *Base points* y *New points*. El conjunto denominado *Base points* corresponde al arreglo de coordenadas faciales tomado al momento de inicializar el sistema de estimación de pose, y es utilizado como plantilla para el cálculo de los ángulos de Euler. Por otro lado, *New points* representa el conjunto de nuevas coordenadas extraídas de las imágenes actualmente analizadas.

Ambos arreglos son reorganizados antes de ingresar a la estructura de casos, con el objetivo de hacer coincidir los valores de cada columna con su eje correspondiente, ya que la operación de rotación no es conmutativa. Una vez dentro de la estructura de casos, los dos conjuntos de coordenadas son enviados a los scripts de MATLAB para realizar el cálculo correspondiente de los ángulos de Euler. El código presente en el primer script se muestra en la Figura 3.13.

```
Inputs(pts, rot_pts)
pts2 = [pts(1,:)-mean(pts(1,:));pts(2,:)-mean(pts(2,:));pts
   (3,:)-mean(pts(3,:))];
x1 = pts2(1,30); x2 = pts2(1,3); x3 = pts2(1,17);
y1 = pts2(2,30); y2 = pts2(2,3); y3 = pts2(2,17);
z1 = pts2(3,30); z2 = pts2(3,3); z3 = pts2(3,17);
rot_pts2 = [rot_pts(1,:)-mean(rot_pts(1,:));rot_pts(2,:)-mean(
   rot_pts(2,:));rot_pts(3,:)-mean(rot_pts(3,:))];
x1r = rot_pts2(1,30); x2r = rot_pts2(1,3); x3r = rot_pts2
   (1,17);
y1r = rot_pts2(2,30); y2r = rot_pts2(2,3); y3r = rot_pts2
   (2,17);
z1r = rot_pts2(3,30); z2r = rot_pts2(3,3); z3r = rot_pts2
   (3,17);

R11_n = (x2ry1-y2x1r)((-z1)y3+z3y1)-(x3ry1-y3x1r)((-z1)y2+z2y1
   );
R11_d = (x2y1-x1y2)((-z1)y3+z3y1)-(x3y1-x1y3)((-z1)y2+z2y1);
R11 = R11_n/R11_d;

R21_n = (y2ry1-y2y1r)((-z1)y3+z3y1)-(y3ry1-y3y1r)((-z1)y2+z2y1
   );
R21_d = (x2y1-x1y2)((-z1)y3+z3y1)-(x3y1-x1y3)((-z1)y2+z2y1);
R21 = R21_n/R21_d;

R31_n = (z2ry1-y2z1r)((-z1)y3+z3y1)-(z3ry1-y3z1r)((-z1)y2+z2y1
   );
R31 = R31_n/R21_d;

R32_n = (z2rx1-x2z1r)((-z1)x3+z3x1)-(z3rx1-x3z1r)((-z1)x2+z2x1
   );
R32_d = ((-y1)x2+y2x1)((-z1)x3+z3x1)-((-y1)x3+y3x1)((-z1)x2+
   z2x1);
R32 = R32_n/R32_d;

R33_n = (z2rx1-x2z1r)((-y1)x3+y3x1)-(z3rx1-x3z1r)((-y1)x2+y2x1
   );
R33_d = -((-z1)x3+z3x1)((-y1)x2+y2x1)+((-z1)x2+z2x1)((-y1)x3+
   y3x1);
R33 = R33_n/R33_d;

R = [R11,R21, R31, R32, R33];
theta_y = (-(asin(R31)))*180/pi;
phi_x = (atan(R32/R33))*180/pi;
gamma_z = atan(R21/R11)*180/pi;
```

FIGURA 3.13: Código de MATLAB para el cálculo de los ángulos de Euler utilizados en la estimación de la pose facial, empleando los puntos 3, 17 y 30 del arreglo de características.

Es importante señalar que en la Figura 3.12 se visualizan dos scripts de MATLAB. El primero emplea un conjunto específico de puntos 3D para el cálculo de los ángulos de *Roll* y *Yaw*, mientras que el segundo utiliza un conjunto distinto de puntos para la estimación del ángulo de *Pitch*. La selección de puntos correspondientes a cada ángulo será detallada en secciones posteriores.

Al ejecutarse la porción de código mostrada en la Figura 3.12, se obtienen como salida los ángulos de rotación. Estos se presentan en la interfaz de usuario mediante un indicador superpuesto a la imagen derecha, como se observa en la Figura 3.14.

FIGURA 3.14: Ejemplo de resultados de extracción de características faciales y su correspondiente mapa de profundidad, junto con la visualización del ángulo recuperado.

La Figura 3.14 ilustra un caso de rotación en el eje Y. Como puede observarse en el conjunto de imágenes, el rostro de la persona se encuentra orientado hacia un objeto ubicado a mayor altura, lo cual se refleja tanto en la disposición espacial de los puntos 3D como en el ángulo de *Pitch* estimado.

Finalmente, con la implementación de las técnicas descritas, se ha desarrollado un sistema basado en visión estéreo mediante la integración de tres entornos de programación distintos: LabVIEW, Python y MATLAB. Este sistema es capaz de estimar la pose facial de cualquier persona presente en la escena, operando con tiempos de ejecución adecuados para su uso en aplicaciones en tiempo real. Los detalles específicos sobre su rendimiento y comportamiento serán presentados en secciones posteriores.

Una vez concluido el desarrollo del sistema de estimación de pose, se aprovechan sus capacidades para su integración en aplicaciones orientadas a sistemas avanzados de asistencia al conductor (ADAS, por sus siglas en

inglés). En este contexto, el análisis del comportamiento del conductor constituye una tarea crucial para evaluar su nivel de atención durante la conducción. Con este propósito, se propone la incorporación de un clasificador de posiciones dentro de la cabina del vehículo, con el objetivo de determinar la dirección de atención del conductor.

Entrenamiento del Clasificador

Para este trabajo se decidió la implementación de las ANN debido a que se desea un algoritmo capaz de aprender información de las distribuciones de los datos y mantenga cierta holgura entre la distancia de los centroides de cada clase.

El uso de técnicas de inteligencia artificial (IA), para trabajos de clasificación, ha estado en tendencia en los últimos años ya que son efectivas para resolver una amplia variedad de tareas específicas, entre ellas la clasificación de vehículos, frutas o incluso rostros humanos. Del conjunto de herramientas presentadas en la Sección 2 para procesos de clasificación, se optó por seleccionar las Redes Neuronales Artificiales (ANN, por sus siglas en inglés), debido a su versatilidad. Estas permiten utilizar distintas funciones de activación y error, lo cual ofrece la posibilidad de que la red aprenda de forma óptima las regiones asociadas a cada clase dentro de la base de datos a construir.

A diferencia de métodos como K-medias, dendrogramas, KNN o SVM, que tienden a generar delimitaciones rígidas entre las regiones de clasificación, las ANN pueden proporcionar una mayor flexibilidad en la asignación de clases. Esto se debe al uso de funciones de activación con interpretaciones probabilísticas, que permiten modelar mejor la incertidumbre inherente a las distribuciones de probabilidad de cada clase.

Como se describe en la Sección 2.2.3, las ANN son algoritmos de aprendizaje supervisado, lo que implica que requieren una base de datos previamente etiquetada. Dado este requerimiento, se desarrolló una base de datos específica mediante la adquisición de mediciones con el sistema de estimación de pose facial (HPE) dentro de la cabina de un vehículo. Estas mediciones están relacionadas con las distancias y posiciones correspondientes a las zonas comunes de atención de un conductor durante la acción de manejo.

La Figura 3.15 muestra la ubicación del sistema de visión estéreo (SVS) y la disposición del conductor durante la captura de la base de datos.

FIGURA 3.15: Vista del sistema de visión estéreo (SVS) y del conductor durante las pruebas del sistema de estimación de pose facial (HPE) para la captura de la base de datos.

Para la construcción de la base de datos, se definieron seis zonas de atención comúnmente observadas en el comportamiento del conductor. Estas son: vista al frente, retrovisores (izquierdo, derecho y central), sistema de radio y vista al teléfono móvil. Al tener identificadas las distintas zonas se prosiguió a realizar la captura de los datos, mediante una serie de pasos establecidos.

Primeramente, se selecciono una hora del día donde la luz no impactara directo con el interior del vehículo con el fin de evitar problemas de iluminación, sobre exposición y otras variaciones que pudieran afectar la consistencia de las mediciones. En segundo lugar, el proceso de medición de los ángulos de la pose se lleva a cabo de manera sistemática el cual es explicado por el algoritmo 10.

Las mediciones realizadas post experimentación, se ven observadas en la Figura 3.16, donde se aprecian las 6 diferentes zonas correctamente delimitadas para el entrenamiento de la red.

Algoritmo 10 Captura de datos para estimación de pose

Entrada: Plantilla base P_{base}, nueva plantilla P_{new}
Salida: Ángulos de pose (Yaw, Pitch, Roll)

1. Inicializar el sistema y capturar los puntos P_{base} con la vista al frente.
2. Capturar una nueva imagen y calcular los puntos P_{new}.
3. Verificar que el sistema funciona correctamente (valores de salida coherentes).
4. **Si** los valores son correctos:
   a) Inicializar contador $i \leftarrow 1$.
   b) **Mientras** no se hayan visitado todas las regiones de interés:
      1) Dirigir la mirada a la región i.
      2) Durante 10 segundos:
         - Capturar y almacenar mediciones usando P_{new}.
         - Calcular la pose usando P_{base} y P_{new}.
      3) Incrementar $i \leftarrow i + 1$.
   c) Guardar todas las mediciones en un archivo Excel.

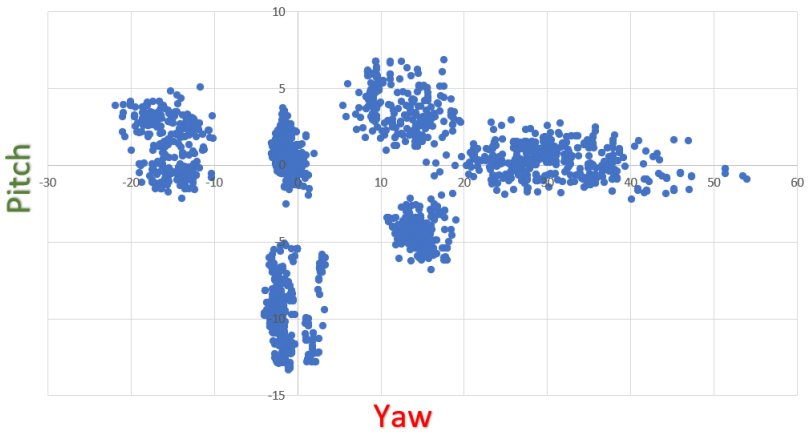

FIGURA 3.16: Datos de entrenamiento utilizados en el NN para la clasificación de zonas, estos datos se obtienen al ejecutar el sistema HPE mencionado, con el cual se realiza una prueba de las seis posiciones analizadas.

Cada una de estas zonas esta representada gráficamente mediante indicadores virtuales en la interfaz del programa principal para la visualización de la zona de atención del conductor. Estos indicadores se pueden ver en la Figura 3.17, donde L es para el retrovisor izquierdo, R para el derecho, M para el central, S para el sistema de radio y T para vista hacia el teléfono.

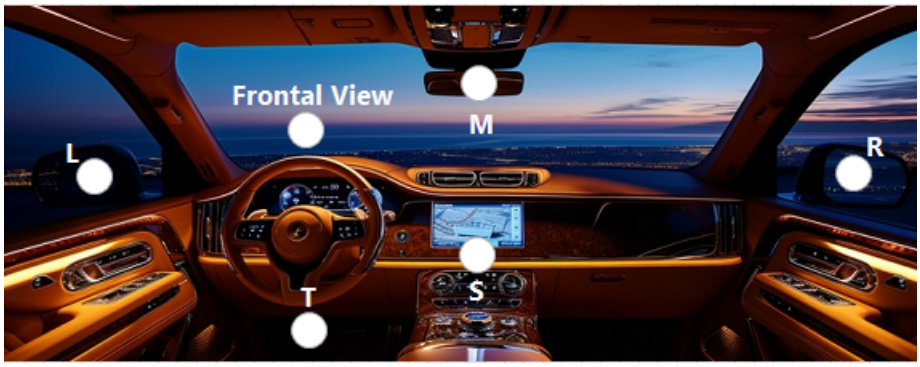

FIGURA 3.17: Visualización virtual de la clasificación zonal de la pose de los conductores.

Por último, para generar la activación de los indicadores en la interfaz de usuario, se debe de indicar la clasificación de la pose de la persona, lo que representa a nuestra siguiente etapa, la cual es el entrenamiento de la ANN.

Clasificación de la Pose Facial

El entrenamiento de un algoritmo de inteligencia artificial (IA) requiere tiempo, ya que es un proceso iterativo. Durante este proceso, se ajustan los parámetros de la red neuronal en cada iteración con el objetivo de mejorar métricas como la exactitud, la precisión y el error del modelo. Entre los parámetros que pueden modificarse se encuentran: la cantidad de neuronas por capa, las funciones de activación, la función de error y el optimizador. Cada uno de estos elementos impacta de forma significativa en el rendimiento de la red, por lo que su correcta selección es fundamental. Esta selección puede realizarse de manera automática (mediante técnicas de búsqueda) o manual, dependiendo del enfoque del investigador.

Dada la creciente disponibilidad de herramientas de IA, actualmente se cuenta con abundante información sobre las mejores prácticas para el diseño de redes neuronales. Esto permite optimizar el proceso de entrenamiento y reducir considerablemente el tiempo requerido para alcanzar un modelo eficiente. Como resultado de múltiples experimentaciones con diferentes arquitecturas, se logró identificar una estructura que alcanza una exactitud del 99 % en la clasificación de las zonas de atención. La arquitectura final se presenta en la Tabla 3.1.

La red neuronal consta de cuatro capas: una de entrada, dos ocultas y una de salida. La capa de entrada está compuesta por dos neuronas, correspondientes a los ángulos de Yaw y Pitch, los cuales fueron seleccionados por ser los que aportan mayor información sobre la orientación de la cabeza. Las

TABLA 3.1: Estructura de la red neuronal implementada para la clasificación de zonas de atención automática.

Estructura	Neuronas	Función de activación
Entrada	2	ReLu
Capa 1	15	ReLu
Capa 2	15	ReLu
Salida	6	Softmax

capas ocultas cuentan con 15 neuronas cada una, empleando la función de activación ReLU, que introduce no linealidad al modelo y facilita el aprendizaje durante el proceso de retropropagación (backpropagation). Finalmente, la capa de salida contiene seis neuronas, una por cada clase correspondiente a las zonas de atención. La función de activación Softmax permite generar una distribución probabilística sobre las posibles clases, asignando una mayor probabilidad a la clase más probable y valores bajos a las restantes.

Definida la arquitectura de la red neuronal, se procedió a evaluar su desempeño mediante una partición de la base de datos en dos subconjuntos: uno para entrenamiento y otro para prueba. Este procedimiento es una práctica común en el desarrollo de modelos de inteligencia artificial, ya que permite validar la capacidad de generalización del modelo sobre datos no vistos durante el entrenamiento. Los resultados son presentados en la siguiente sección.

3.3. Aplicación ADAS: Alarma de Frenado Inteligente

Dentro de las problemáticas que los sistemas ADAS buscan resolver, se pueden encontrar sistemas de frenado automático para evitar colisiones con objetos, vehículos y peatones, en caso de emergencias. En nuestro planteamiento buscamos utilizar el sistema de HPE como agente principal en la toma de decisión si se es necesario activar los frenos o con mayor flexibilidad solamente indicar el conductor de un evento peligroso. El proceso general para el sistema ADAS propuesto es aquel que se muestra en la Figura 3.18, donde podemos observar que el sistema de HPE es apoyado por un segundo SVS el cual se encarga de detectar y medir la distancia de algún objeto en la vialidad. El sistema de detección de objetos implementado nombrado para este trabajo como Detección de objetos de estéreo visión (SOD, por sus siglas en inglés), consiste en la aplicación del algoritmo de detección de objetos

YOLO-R, el cual ha demostrado ofrecer excelentes resultados en la detección y clasificación de objetos, así como en el uso del algoritmo de emparejamiento de patrones SoRA (detallada en el trabajo de Real-Moreno et al., 2023), seleccionado por sus tiempos de procesamiento más cortos en comparación con otras alternativas.

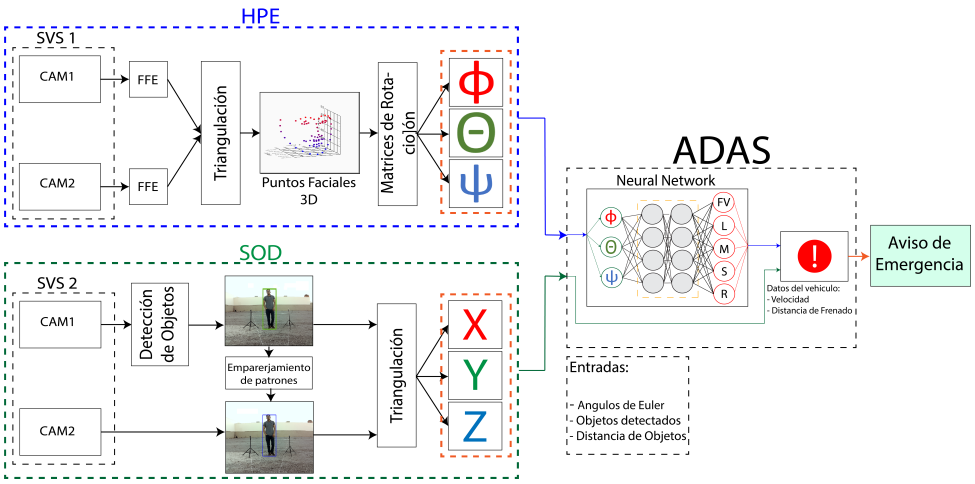

FIGURA 3.18: Diagrama del sistema ADAS propuesto. Se muestran las unidades principales: HPE, SOD y la red neuronal propuesta. El símbolo de exclamación representa una advertencia, y este nodo proporciona la clasificación del escenario de conducción en cuestión.

Esta unidad SOD se describe de la siguiente manera: Primero, se capturan los dos fotogramas disponibles de cada cámara, y utilizando el algoritmo de detección de objetos YOLO-R con la imagen proveniente de la cámara "CAM1", es posible detectar múltiples personas y vehículos dentro del encuadre. Por último, se aplica la técnica de emparejamiento de patrones SoRA para localizar el mismo objeto en la imagen proveniente de "CAM2". Finalmente, se realiza el proceso de triangulación utilizando ambas imágenes para obtener las coordenadas X, Y y Z de los objetos. Este proceso se detalla en la Figura 3.19.

Con estas coordenadas, se puede identificar la posición del objeto y utilizar dicha información para evaluar la situación en tiempo real, debido a que las condiciones de frenado cambian con respecto a la velocidad del vehículo.

Para el sistema de asistencia al conductor, se utiliza tanto la clasificación de la zona de atención como la distancia entre el vehículo y el objeto detectado para emitir una respuesta, advertencia o alarma al conductor. Esta advertencia indica si hay una zona del entorno que no está siendo observada por el conductor y en la cual existe un objeto a una distancia considerable para un evento riesgoso, tomando en cuanta a la velocidad que se viaja en

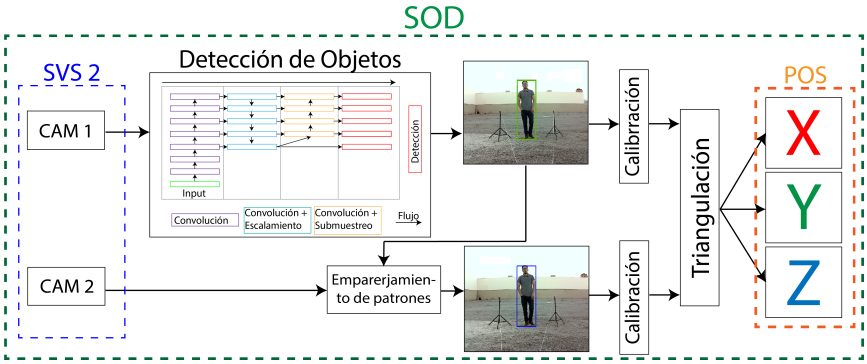

FIGURA 3.19: Descripción gráfica de la sección SOD. Donde la detección de objetos esta basado en YOLOv4 y para el emparejamiento de plantillas se utiliza SoRA.

el vehículo. Las condiciones necesarias para emitir dicha advertencia se establecen mediante la ecuación booleana 3.11. Esta ecuación depende tanto de la detección de la zona hacia la cual el conductor está mirando como de la detección de objetos cercanos al vehículo.

$$Alarm = L(A + B) + M + S + R(C + B) + T \tag{3.11}$$

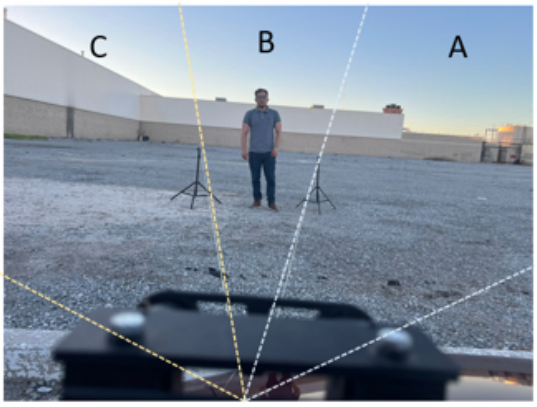

FIGURA 3.20: SVS 2 ubicado en el cofre de vehículo, donde a partir del centro entre las cámaras se delimitan los sectores A, B y C para la experimentación.

La zonas A, B y C son detalladas en la Figura 3.20. Donde A, B y C son los sectores correspondientes del área visible en las imágenes estereoscópicas del segundo SVS. Estos sectores pueden ajustarse según el valor deseado, mediante los ángulos de visión. Además, existe un parámetro de profundidad que define la distancia máxima para considerar que un objeto está "cerca" del vehículo y, por tanto, activar la señal de alarma. La zonas A, B y C son detalladas en la Figura 3.20.

Capítulo 4

Experimentos y Análisis de Resultados

4.1. Experimentación y Resultados de HPE

El sistema propuesto en la sección anterior se analiza en función de su rendimiento en tres etapas. La primera consiste en verificar la aplicación del mismo método de extracción de características sobre ambas imágenes estéreo. La segunda etapa se enfoca en el análisis de diferentes combinaciones de puntos de la nube de datos que representa el rostro, con el objetivo de identificar aquellas que permiten recuperar los movimientos angulares del rostro con el menor error posible así como su variabilidad en cada paso del experimento de verificación. Finalmente, se realiza una comparación cuantitativa del sistema de estimación de pose de la cabeza (HPE) con otros métodos del estado del arte. Como cierre, se analiza la aplicación del sistema HPE como parte de un sistema ADAS, orientado a la identificación del nivel de atención del conductor durante la conducción.

4.1.1. Verificación del FLD

Un objetivo fundamental de esta investigación es lograr una coincidencia rápida y precisa de puntos faciales para la estimación de la pose en visión estéreo. Para ello, se seleccionó el detector de marcas faciales (FLD) de la biblioteca Dlib King, 2009b, ampliamente reconocido en la literatura por ofrecer mapas de características con bajo tiempo de procesamiento Kazemi y Sullivan, 2014. Esta elección se alinea con el enfoque basado en triangulación para el cálculo de coordenadas 3D, el cual requiere que el objeto (el rostro) sea completamente visible en ambas imágenes.

Existen alternativas al predictor de 68 puntos de Dlib, como MediaPipe de Google, que proporciona puntos faciales incluso en poses extremas. Sin

embargo, MediaPipe estima puntos no visibles en ambas cámaras, lo cual interfiere con el enfoque estéreo. De manera similar, RetinaFace, otro método para la localización facial, produce errores mayores que nuestro enfoque (Lugaresi et al., 2019, Deng et al., 2020).

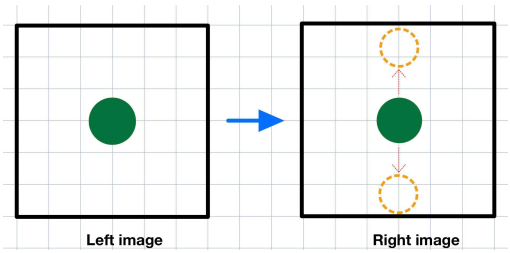

FIGURA 4.1: Ejemplo visual de la variación de un punto facial en el eje Y.

El primer enfoque probado para el emparejamiento de puntos entre imágenes estéreo fue el algoritmo SoRA, el cual mostró una velocidad de procesamiento adecuada para aplicaciones en tiempo real (más de 20 fps). No obstante, su desempeño se vio comprometido en condiciones de iluminación variables, generando emparejamientos incorrectos o incluso ausentes. Durante la fase de pruebas del algoritmo de emparejamiento de datos, se observó un hallazgo relevante: aplicar el detector de marcas faciales directamente en ambas imágenes (izquierda y derecha) producía pares consistentes de puntos faciales, adecuados para la triangulación.

Este resultado fue validado mediante comparaciones con los emparejamientos generados por SoRA. Al aplicar el mismo FLD en ambas imágenes, se comprobó que el desplazamiento en el eje X era prácticamente idéntico, con diferencias mínimas cuando las había. A partir de esta observación, se analizó la variación en el eje Y de los puntos obtenidos con FLD (como se muestra en la Figura 4.1) para confirmar su estabilidad y utilidad como método de emparejamiento.

Al aplicar el FLD a las imágenes estéreo para recuperar los puntos 3D, se observó que en una muestra de 220 fotogramas el desplazamiento promedio en píxeles en el eje vertical fue de 1.78 píxeles (Figura 4.2), lo cual representa una tolerancia aproximada de 2.4 milímetros en la estimación de profundidad, debido a la variación del punto en el eje Y.

Esta prueba se realizó con una distancia de un metro entre el objeto de prueba y las cámaras, donde la mayor variación de profundidad calculada se mantiene dentro del rango de los milímetros, lo cual es aceptable, ya que el

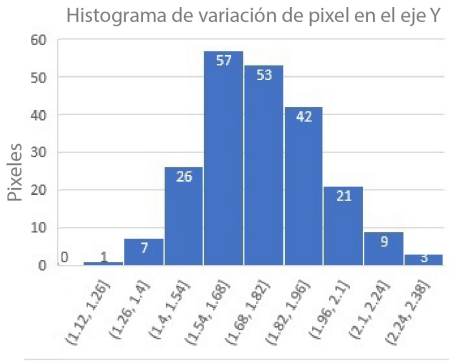

FIGURA 4.2: Desplazamiento en píxeles en el eje Y de cada punto sobre una muestra de 220 mediciones.

objetivo principal es recuperar los ángulos de Euler asociados a la transformación afín del rostro. En este contexto, la información relevante se encuentra en el giro del conjunto de puntos y no en los pequeños desplazamientos provocados por variaciones subpíxel.

Análisis Comparativo entre Características Faciales

Una vez comprobada la utilidad del extractor de características en ambas imágenes, se procede a realizar una comparación cuantitativa del funcionamiento del sistema de estimación de pose (HPE, por sus siglas en inglés) mediante distintos conjuntos de características faciales. Esta etapa se motivó por observaciones preliminares, en las que se detectó que los puntos faciales se comportaban de manera diferente dependiendo del ángulo de rotación.

Para la experimentación, se utilizó una cabeza de maniquí con características similares a las de una cabeza humana, ya que el experimento requería un objeto que pudiera ajustarse libremente y fijarse en una base angular graduada. Esto garantiza la estabilidad en cada paso de la rotación y evita sesgos causados por movimientos involuntarios al utilizar un sujeto real. La prueba consistió en realizar rotaciones sobre los tres ejes principales (Yaw, Pitch y Roll), posicionando la cabeza en diferentes ángulos sobre la base y manteniéndola fija durante 15 segundos en cada posición. Para el ángulo Yaw, el rango de prueba fue de $-20°$ a $20°$ con incrementos de $5°$; para los ángulos Pitch y Roll, el rango fue de $-18°$ a $18°$, con incrementos de $6°$. Esta definición de pasos se debió a las características físicas de la base y la marcación entre cada incremento.

Detallado el proceso experimental con el maniquí, se seleccionaron cinco conjuntos de puntos faciales para analizar su influencia en el cálculo de los

FIGURA 4.3: Modelo de cabeza con los conjuntos de puntos seleccionados para el análisis de recuperación de ángulos: a) Ojos externos (E-E), b) Ojos internos (I-E), c) Contorno facial (F-C), d) Cejas y e) Boca, respectivamente.

ángulos y determinar cuáles resultan más adecuados para la estimación de pose. Las agrupaciones de puntos se muestran en la Figura 4.3, donde en todos los casos se incluyó el punto correspondiente a la nariz, con el objetivo de asegurar una referencia tridimensional de profundidad en cada combinación.

Los resultados de la estimación de los ángulos para los distintos conjuntos de puntos se presentan en las Figuras 4.4, 4.5 y 4.6. Estas muestran que diferentes referencias 3D pueden generar variaciones significativas en el cálculo de los ángulos de pose de la cabeza. Por ello, se realizó un análisis comparativo para identificar los conjuntos de puntos que permiten una recuperación más precisa de los ángulos de Euler.

La Tabla 4.1 resume los errores cuadráticos medios (RMSE) obtenidos para cada conjunto de puntos. Se observa que los puntos del contorno facial (F-C) ofrecen el mejor rendimiento para los ángulos de Yaw y Roll, lo cual sugiere que una base más amplia entre los puntos seleccionados mejora la precisión en rotaciones horizontales y de inclinación lateral. En contraste, para el ángulo de Pitch, el conjunto de puntos con menor cálculo de error fue el de los ojos internos (I-E), lo que indica que configuraciones más concentradas pueden favorecer la estimación en rotaciones verticales.

Cabe destacar que todos los conjuntos de puntos probados se ubican principalmente en orientación horizontal. Se realizaron pruebas adicionales utilizando puntos dispuestos verticalmente, pero estas presentaron errores significativamente mayores.

Dado estos resultados, los conjuntos mencionados con menor RMSE se tomaron en cuenta el análisis comparativo con los distintos sistemas similares para HPE.

Para confirmar la consistencia de estos resultados, las Tablas 4.2 y 4.3 presentan el RMSE promedio por cada paso de rotación en los ángulos Yaw,

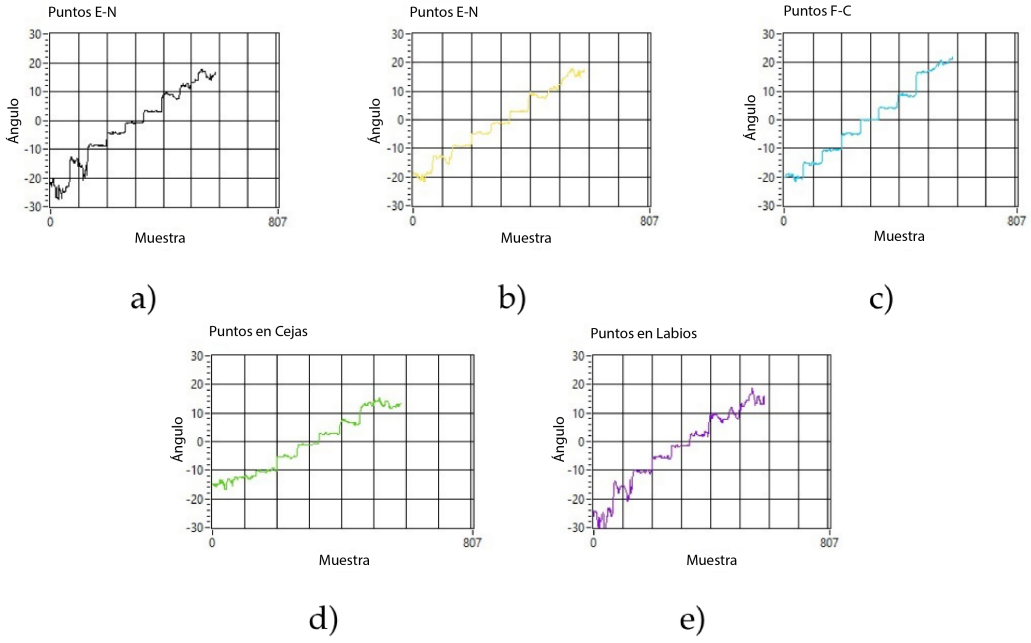

FIGURA 4.4: Ángulos de yaw correspondientes a los conjuntos de puntos seleccionados para la comparación del rendimiento: a) Ojos externos (E-E), b) Ojos internos (I-E), c) Contorno facial (F-C), d) Cejas y e) Puntos de la boca.

TABLA 4.1: Resultados RMSE angular para cada conjunto de puntos.

Ángulo	E-E	I-E	F-C	Cejas	Boca
Yaw	2.32	1.94	**0.89**	3.02	3.29
Pitch	2.07	**0.84**	4.8	2.68	3.23
Roll	3.98	5.17	**0.74**	2.99	3.66

Pitch y Roll, respectivamente, corroborando el buen desempeño de los conjuntos seleccionados.

Como resultado de esta evaluación cuantitativa, se seleccionaron los conjuntos de puntos con menor error RMSE para su uso en el análisis comparativo con otros sistemas similares de estimación de pose facial, con el objetivo de evaluar la competitividad del sistema propuesto frente a algoritmos del estado del arte.

Comparación de Sistemas de HPE

El sistema de HPE se valido experimentalmente al analizar el nivel de error que este obtiene utilizando los puntos establecidos, donde, al final se seleccionaron los puntos óptimos en los cuales existe aproximadamente un grado de error en cada ángulo de Euler. Ahora, el sistema se ha de comparar

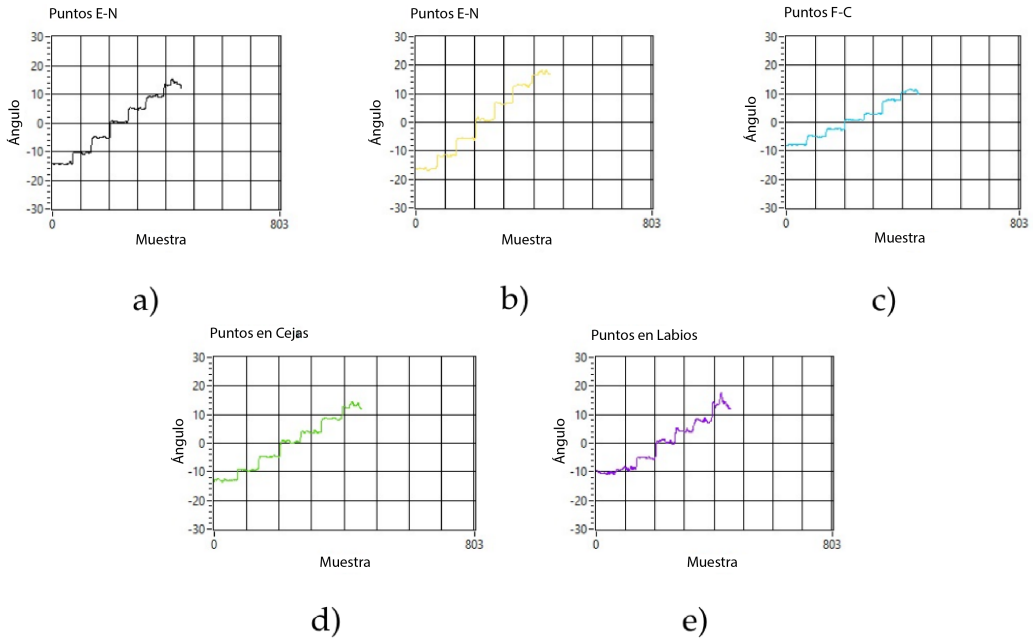

FIGURA 4.5: Ángulos de Pitch correspondientes a los conjuntos de puntos seleccionados para la comparación del rendimiento: a) Ojos externos (E-E), b) Ojos internos (I-E), c) Contorno facial (F-C), d) Cejas y e) Puntos de la boca.

con seis diferentes métodos mediante los resultados cuantitativos presentados por sus autores en sus respectivos documentos, y donde solo uno de ellos existe como software libre, con el cual nos comparamos directamente realizando el mismo procedimiento de experimentación que con el sistema propuesto.

Primeramente damos inicio a la comparación, obteniendo los resultados del software libre conocido como OpenFace desarrollado por Baltrušaitis, Robinson y Morency, 2016, con el cual se realiza una experimentación bajo el mismo procedimiento establecido. OpenFace es un sistema todo en uno para análisis facial, el cual realiza mediciones de distintos comportamientos faciales, como extracción de histogramas de gradientes, medición de Unidades de Acción (AU, por sus siglas en inglés) y además entrega la estimación de pose facial. Existen dos versiones de OpenFace (1.0 y 2.0) de las cuales la diferencia primordial es el algoritmo para HPE implementado, uno es basado en Campos Locales de Redes Neuronales Condicionales (CLNF, por sus siglas en inglés) los cuales se entrenaron para inferir profundidad desde una imagen 2D. El segundo algoritmo es basado en CNN's entrenadas con una base de datos con información de profundidad fabricada mediante un segundo algoritmo diseñado para estimar la tercera dimensión (Baltrušaitis, Robinson

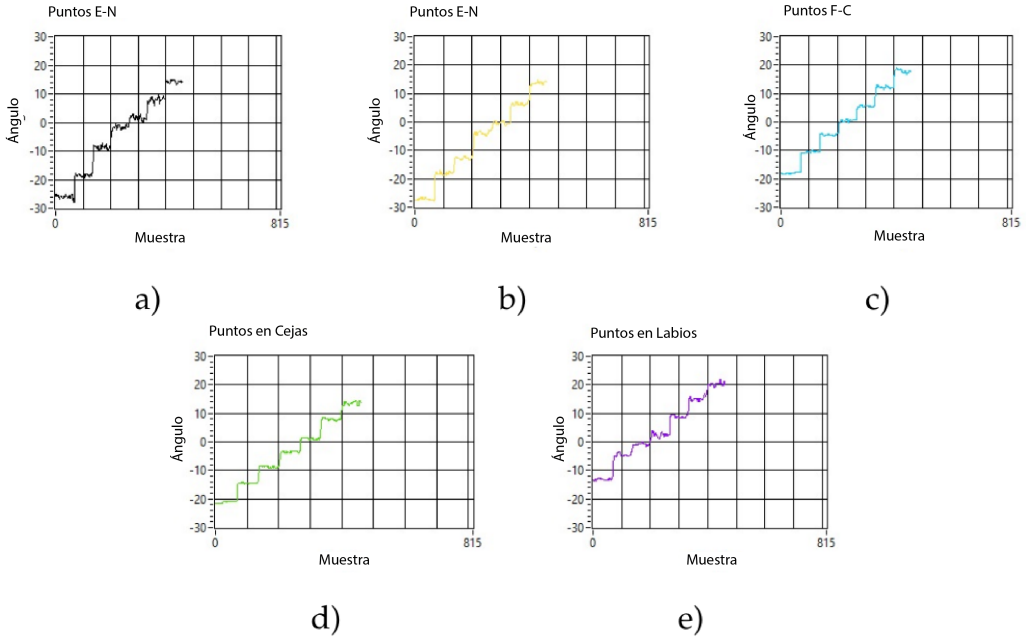

FIGURA 4.6: Ángulos de Roll correspondientes a los conjuntos de puntos seleccionados para la comparación del rendimiento: a) Ojos externos (E-E), b) Ojos internos (I-E), c) Contorno facial (F-C), d) Cejas y e) Puntos de la boca.

y Morency, 2016; Baltrusaitis et al., 2018). Para cuestiones de análisis se utilizará OpenFace 2.0, versión reportada por los autores con el menor valor de error en la captura del HPE.

Las Tablas 4.4 y 4.5 muestran el RMSE para OpenFace 2.0 Baltrusaitis et al., 2018 en cada paso realizado, utilizando el mismo modelo de maniquí y base angular. OpenFace 2.0 mostró que es capaz de mantener una estimación de ángulos con precisión, mas no exactos, esto debido al alto nivel de error medido, como se observar en la Figura 4.7. OpenFace 2.0 alcanzó un error máximo de 8.75°, que es significativamente mayor que el de nuestro enfoque, que mostró un error máximo de aproximadamente 1.67°.

La Tabla 4.6 muestra que la implementación propuesta presentó un menor RMSE en comparación con OpenFace 2.0, y también con otros métodos, lo cual demuestra las capacidades del enfoque propuesto como sistema de estimación de la pose de la cabeza. Donde las mediciones realizadas por OpenFace 2.0 entregaron resultados con error elevado comparado al sistema propuesto.

Posteriormente a la experimentación con OpenFace 2.0, se procede a una comparación cuantitativa entre el sistema propuesto y otros modelos desarrollados para el mismo fin, la estimación de la pose de la cabeza.

Los modelos comparados en esta investigación se presentan en la Tabla

TABLA 4.2: RMSE de los ángulos de Yaw estimados en cada paso.

Ángulo Yaw	
Posición angular	**RMSE**
-20	0.62
-15	0.43
-10	0.55
-5	0.18
0	0.12
5	0.87
10	1.52
15	1.94
20	0.90

TABLA 4.3: RMSE de los ángulos de Pitch y Roll estimados en cada paso.

Posición Ángulos	**Ángulo**	
	Pitch	**Roll**
-18	1.67	0.23
-12	0.86	1.49
-6	0.38	1.55
0	1.01	0.58
6	0.67	0.56
12	1.11	0.41
18	1.06	0.63

4.6. Todos ellos utilizan distintas técnicas, pero comparten el objetivo de estimar la pose facial empleando información de profundidad, ya sea medida o inferida.

La diferencia entre el método propuesto y aquellos con los que se compara radica en el nivel de error alcanzado y la velocidad de ejecución de cada algoritmo. Por ejemplo, el método de Madrigal y Lerasle, 2020 emplea un modelo de cara en 3D (3DMM, por sus siglas en inglés) como referencia para estimar con precisión la pose a partir de una nube de puntos. Este enfoque incluye el uso de descriptores de características robustas aceleradas (SURF, por sus siglas en inglés) e histogramas de características de puntos rápidos (FPFH, por sus siglas en inglés), con el objetivo de aumentar la robustez frente a variaciones en la iluminación, precisión y orientación. Además, incorpora un esquema de fotogramas clave que utiliza el algoritmo de Iterative Closest Point (ICP) para mantener una estimación continua de los ángulos.

Por otro lado, el método propuesto no requiere una fase de entrenamiento como ocurre en los enfoques de aprendizaje profundo de Baltrusaitis et al.,

TABLA 4.4: RMSE de OpenFace 2.0 para ángulo Yaw.

Ángulo Yaw	
Posición angular	**RMSE**
-20	9.80
-15	7.98
-10	3.92
-5	2.66
0	0.28
5	0.13
10	3.99
15	4.18
20	6.33

TABLA 4.5: RMSE del ángulo de Pitch y Roll con OpenFace 2.0.

Posición angular	**Ángulo**	
	Pitch	**Roll**
-18	5.50	4.73
-12	8.75	4.57
-6	3.80	3.89
0	0.66	3.61
6	3.13	3.2
12	5.44	2.56
18	7.60	3.05

2018, Huang et al., 2020 y Xu, Jung y Chang, 2022, ya que la idea principal fue desarrollar un método que aprovechara las propiedades de la visión estéreo. Cabe señalar que los enfoques basados en aprendizaje profundo requieren bases de datos específicas con información de profundidad, las cuales son escasas en el contexto de visión estereoscópica, lo que dificulta la implementación de redes neuronales convolucionales (CNN). Debido a esta limitación, se planteó estimar la pose facial mediante cálculos directos sobre las nubes de puntos generadas por las marcas faciales, resultando en un método mayoritariamente matemático, salvo por el algoritmo de detección de marcas.

Como resultado, el enfoque propuesto es capaz de estimar con exactitud y precisión la pose de la cabeza sin necesidad de técnicas auxiliares como fotogramas clave, ICP y descriptores SURF, las cuales podrían reducir el rendimiento del modelo. Además, es importante destacar que, a diferencia de otros trabajos, el sistema desarrollado ofrece una estimación continua de la pose en lugar de valores angulares discretos.

Al comparar la precisión entre modelos en la Tabla 4.6, se emplea la métrica de Error Medio Absoluto (MAE, por sus siglas en inglés), utilizada por

varios autores. El método propuesto alcanza valores de MAE de 0.73°, 0.86° y 0.72° para los ángulos de Yaw, Pitch y Roll, respectivamente. El método más cercano en términos de MAE es el de Xu, Jung y Chang, 2022, con errores de 1.39°, 1.82° y 1.09° en los mismos ángulos, y un error medio de 1.42°, más del doble del error medio del enfoque propuesto (0.77°).

Los resultados muestran que nuestro sistema ofrece un bajo RMSE y MAE al trabajar con imágenes de alta resolución. Asimismo, el sistema mantiene un bajo nivel de error incluso al operar con imágenes de 320x240 píxeles, aumentando su rendimiento hasta un promedio de 17 cuadros por segundo (fps), lo que lo hace adecuado como sistema de seguimiento de cabeza en aplicaciones de análisis facial. En esta configuración, el RMSE de los ángulos de Yaw, Pitch y Roll se incrementa a 1.12°, 2.42° y 1.69° grados respectivamente, cifras que aún son inferiores a las de la mayoría de los métodos comparados en la Tabla 4.6. Finalmente, los trabajos de Madrigal y Lerasle, 2020, Li et al., 2018 y Baltrusaitis et al., 2018 reportan velocidades de procesamiento de 11, 18 y 15 fps, respectivamente, lo que demuestra que nuestro enfoque resulta ser competitivo en términos de velocidad de procesamiento.

Es entonces en términos generales que el sistema de HPE desarrollado, es un sistema competitivo gracias a su nivel de exactitud en la medición y a su vez a la velocidad a la que obtiene dichas mediciones de pose, 7 y 17 FPS para altas y bajas resoluciones de imagen respectivamente, donde estas métricas de exactitud y latencia fueron entregados por una computadora portátil, la cual cuenta con un procesador Intel Core i5 de séptima generación y 16 GB de memoria RAM.

TABLA 4.6: Comparación del RMSE y MAE entre el enfoque propuesto y el mejor resultado de los métodos comparados.

Método	**RMSE**				**MAE**			
	Yaw	Pitch	Roll	Media	Yaw	Pitch	Roll	Media
Propuesto	**0.89**	**0.84**	**0.75**	**0.83**	**0.73**	**0.86**	**0.72**	**0.77**
Baltrusaitis et al., 2018 (OpenFace 2.0)	5.35	5.59	3.37	4.89	4.32	4.94	3.65	4.29
Li et al., 2018	2.10	1.90	2.20	2.07	-	-	-	-
Madrigal y Lerasle, 2020 (KFv3)	-	-	-	-	2.13	2.12	2.28	2.18
Huang et al., 2020	-	-	-	-	4.57	5.18	3.12	4.29
Xu, Jung y Chang, 2022	-	-	-	-	1.39	1.82	1.09	1.42
Abate et al., 2021 (Regresión lineal)	-	-	-	-	6.57	5.47	3.80	5.28

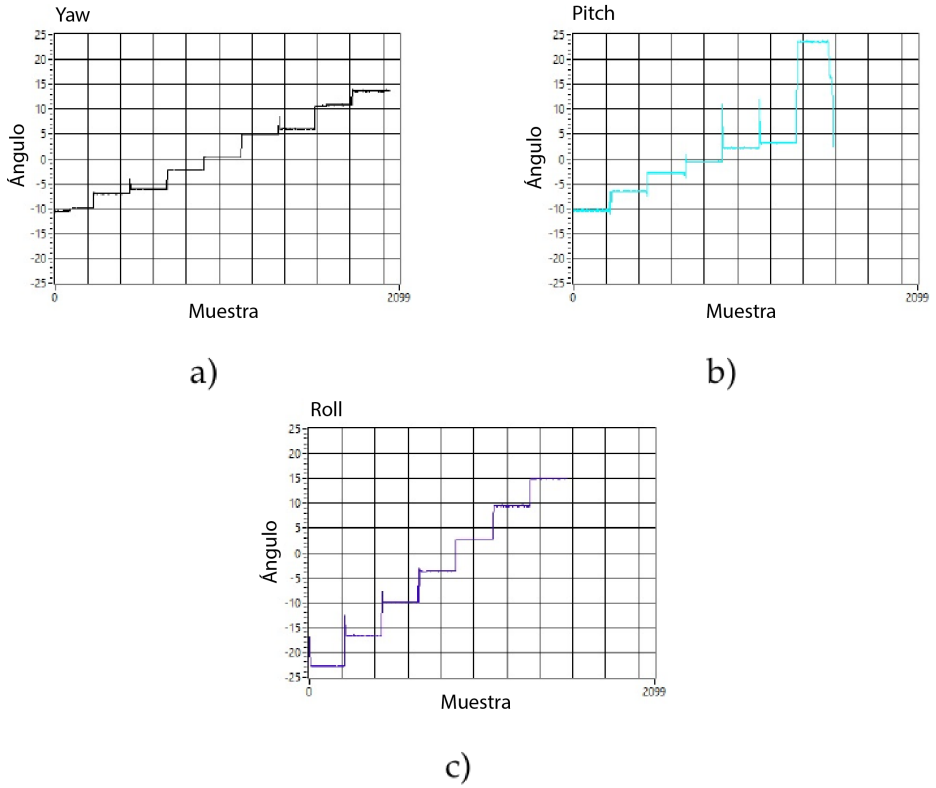

FIGURA 4.7: Resultados de los ángulos con OpenFace 2.0. a)Ángulo de Yaw , b)Ángulo de Pitch, c)Ángulo de Roll.

Consecuentemente, al obtener resultados favorables en términos de exactitud y tiempos de ejecución del sistema, se planteó su uso en una aplicación dentro del campo de los sistemas avanzados de asistencia al conductor (ADAS). Dadas las cualidades del sistema y la creciente necesidad de mejorar la seguridad vial, se decidió incorporar la medición de la pose facial como uno de los dos criterios de alarma para la detección de la atención del conductor, tanto hacia el entorno interno como externo del vehículo. Por ello, se definieron y clasificaron seis zonas distintas de atención del conductor (precisamente delimitadas), utilizando únicamente los ángulos de la pose facial, mediante la implementación de una red neuronal.

Durante el proceso de entrenamiento, se alcanzó una exactitud del 99 % utilizando el subconjunto de entrenamiento. Posteriormente, se validó dicho desempeño con el conjunto de datos de prueba, generando una matriz de confusión que permitió evaluar la capacidad del modelo para clasificar correctamente las zonas de atención.

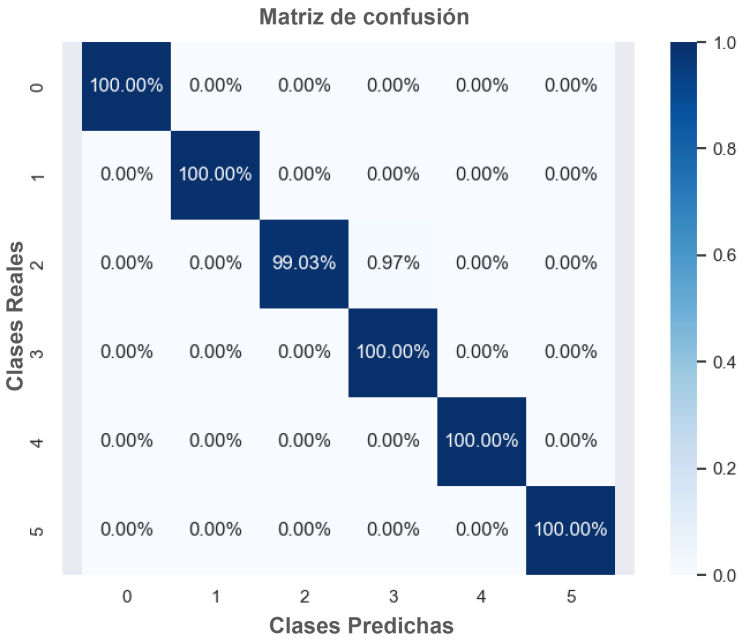

FIGURA 4.8: Matriz de confusión obtenida con los datos de prueba. Evalúa el desempeño del clasificador en la identificación de las zonas de atención del conductor.

Concluido el entrenamiento de la red neuronal y habiendo alcanzado los niveles de rendimiento deseados, se procedió a evaluar su funcionamiento en un caso de estudio aplicado. Con el sistema de estimación de pose facial en operación y el clasificador de zonas de atención entrenado, se avanzó hacia la última etapa de la investigación: su implementación en un contexto de alta relevancia actual, correspondiente a su aplicación dentro del área de los sistemas ADAS.

4.2. ADAS: Sistema de Detección de Eventos de Emergencia

En los últimos años, se ha incrementado la investigación relacionada con diversas tareas dentro de los Sistemas Avanzados de Asistencia al Conductor (ADAS), debido al alto riesgo que enfrentan los peatones de verse involucrados en accidentes en intersecciones, zonas de baja velocidad, carreteras, entre otros (Combs et al., 2019; Administration et al., 2016; Yanagisawa, Swanson, Najm et al., 2014; Malla et al., 2023). Para abordar esta problemática, los sistemas ADAS integran tareas como la detección de vehículos, la estimación de la mirada, la fatiga y la pose del conductor, así como el monitoreo de su atención en al carril del vehículo. El objetivo es incrementar la seguridad vial, dado que atender a la atención del conductor es crucial, considerando que una proporción significativa de los accidentes automovilísticos se debe a la falta de conciencia situacional (Peng et al., 2024).

Como lo mencionan Li et al., 2021, un aspecto fundamental en la implementación de sistemas ADAS es la prevención de muertes y lesiones. Sin embargo, desde otra perspectiva, los accidentes también conllevan elevados costos económicos. De acuerdo con la Administración Nacional de Seguridad del Tráfico en Carreteras (NHTSA) de los Estados Unidos, en 2018 los accidentes representaron un costo total de 242 mil millones de dólares, y específicamente en los casos confirmados de conductores distraídos, el costo representó aproximadamente el dieciséis por ciento del total.

En muchos casos, la falta de atención o distracción puede deducirse a partir de la postura del conductor, lo cual puede afectar su capacidad para tomar decisiones críticas que eviten colisiones. Un ejemplo de esto se encuentra en el trabajo de Addanki et al., 2020, donde se analiza la pose de la cabeza y se considera también el movimiento de los párpados como indicador de somnolencia, directamente relacionado con la fatiga del conductor. Asimismo, el trabajo de Jha y Busso, 2022 presenta un enfoque para estimar la dirección de la mirada del conductor a través de la orientación de la cabeza. Por tanto, las técnicas de estimación de pose, como la estimación de la Pose del rostro, desempeñan un papel importante en las tareas de aplicación de los sistemas ADAS. Es importante destacar que estos sistemas de análisis del conductor no actúan de manera aislada, sino que se integran con sistemas de monitoreo externo del entorno del vehículo (detección de obstáculos, peatones o vehículos cercanos), de modo que ambos niveles de análisis interno y externo trabajan en conjunto para anticipar y reducir escenarios de riesgo.

TABLA 4.7: Tabla de tiempo de reacción (RT) y la distancia de frenado (BD), dado los lineamientos de la AASHTO, para evaluar si el vehículo se detiene antes del impacto (SbI, sus siglas en inglés).

FPS (p/s)	**Vel.** (Km/h)	**Vel.** (m/s)	**RT** (s)	**RT** (m)	**FPS** (m)	**OD** (m)	**BD** ($3{,}4\,m/s^2$)	**BD** Ventana	**Min** BD	**SbI**
17	20	5.56	1.50	8.33	0.33	25	4.60	20.40	11.73	Si
17	30	8.33	1.50	12.50	0.49	25	10.30	14.70	1.71	Si
17	40	11.11	1.50	16.67	0.65	25	18.40	6.60	-10.72	No
17	50	13.89	1.50	20.83	0.82	25	28.70	-3.70	-25.35	No

Dada la necesidad de contar con alternativas para estimar el nivel de atención del conductor, el sistema HPE desarrollado representa una herramienta adecuada para su uso dentro de un sistema ADAS, ya que ha demostrado un alto nivel de exactitud en sus mediciones, además de contar con la ventaja de ser un sistema completamente visual. Esto garantiza un amplio margen para su evolución futura.

A continuación, se presenta un análisis del entorno vehicular con el objetivo de definir las circunstancias bajo las cuales este sistema puede ser implementado.

Como lo establece la American Association of State Highway and Transportation Officials (AASHTO) Officials, 2011, un conductor tiene un tiempo de reacción (RT) de aproximadamente 1.5 segundos ante una situación que requiera frenado. Como se muestra en la Tabla 4.7, el tiempo de reacción a diferentes velocidades impacta significativamente en las distancias de frenado (BD), ya que, a medida que la velocidad del vehículo aumenta, también lo hace la distancia recorrida durante dicho tiempo de reacción.

La Tabla 4.7 presenta un ejemplo de detección de un objeto a una distancia de 25 metros. Utilizando las guías de AASHTO para el frenado, considerando tanto la distancia recorrida durante el tiempo de reacción como la distancia adicional representada por el desplazamiento de un cuadro de imagen, se estima el rango de velocidades en el cual nuestro sistema puede ser aplicado (por debajo de los 30 km/h). Esta tabla muestra que, a 20 km/h, hay aproximadamente 12 metros disponibles para aplicar los frenos con suficiente anticipación para detenerse antes de un posible impacto (SbI, Stop before Impact); sin embargo, al aumentar la velocidad a 30 km/h, esta distancia disponible se reduce a casi 2 metros. Estas cifras se estiman considerando una desaceleración constante de $3{,}4 m/s^2$. En caso de un frenado de emergencia, el vehículo puede desacelerar a un ritmo mucho mayor, lo que permitiría ampliar las distancias mínimas de frenado a velocidades más altas.

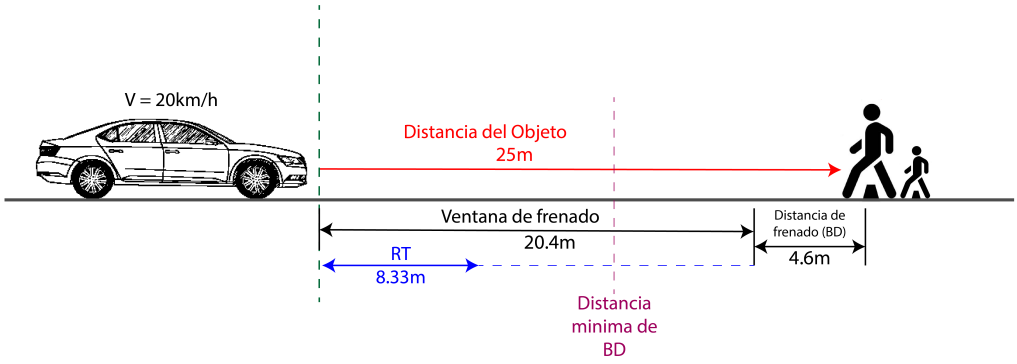

FIGURA 4.9: Distancias de frenado de un vehículo circulando a 20 km/h. Esta figura muestra una representación gráfica del proceso de frenado y todas las distancias primarias involucradas: distancia al objeto, de frenado (BD), recorrida durante el tiempo de reacción (RT) y la distancia mínima de frenado.

La Figura 4.9 representa el ejemplo de un objeto situado a 25 metros. En esta figura se puede apreciar gráficamente la cantidad de metros necesarios para una situación de frenado segura. Se observa que, a una velocidad de 20 km/h, existe distancia suficiente para que el conductor inicie el proceso de frenado manualmente o mediante un sistema automático de frenado de emergencia (AESB). En este ejemplo, una adecuada supervisión de la atención del conductor y la detección de objetos puede proporcionar una retroalimentación rápida que le permita aplicar los frenos antes de alcanzar la distancia mínima requerida para una detención segura. Sin embargo, al incrementarse la velocidad del vehículo, la distancia mínima de frenado también aumenta. De acuerdo con la Tabla 4.7, a una velocidad de 30 km/h, sólo quedarían 1.71 metros para iniciar el frenado, lo que subraya la importancia de contar con un sistema automático que alerte al conductor. Este análisis demuestra que el enfoque propuesto es aplicable para velocidades inferiores a 30 km/h, conforme a las guías estadounidenses que consideran una desaceleración constante del vehículo.

A partir de estas especificaciones sobre el frenado de vehículos y la velocidad de procesamiento de 17 FPS del sistema propuesto, se realizó la experimentación de forma segura y sin la presencia de objetos en movimiento, con el fin de evaluar el funcionamiento de clasificación del sistema completo, donde además se incorpora el segundo SVS con la unidad SOD para detectar a los objetos delante del vehículo y medir su distancia.

La unidad de detección de objetos (SOD) tiene documentado que en un entorno controlado muestra un error promedio de 0.4 metros en distancias

entre 5 y 15 metros. Para la experimentación de la aplicación donde el sistema HPE y el SOD trabajan en conjunto, en el exterior, se utilizaron dos tripies de cámara que definen las zonas de cada sector establecido, además de indicar la distancia para las pruebas realizadas. Estas bases se colocaron a 5 m del vehículo, y el sujeto de prueba se ubico a la misma distancia, como se muestra en la Figura 3.20. Adicionalmente, el SOD fue programado para detectar únicamente peatones, esto para tener delimitado los objetos que podrían aparecer en escena. Por último, el sistema de medición SOD se llevo al exterior y se analizo su comportamiento ante iluminación natural, lo cual arrojo un error RMSE de 0.74 metros, resultado que demuestra el sistema continúa ofreciendo mediciones confiables en escenarios reales de aplicaciones.

Teniendo entonces las métricas de funcionamiento de cada sistema se continuo con la etapa de experimentación en el exterior, la cual se detalla a continuación: se colocó al conductor dentro del vehículo y a un sujeto externo a la misma distancia que la separación entre las cámaras estéreo. La evaluación se realizó posicionando al sujeto externo sucesivamente en los sectores A, B y C. A medida que el sujeto se desplazaba entre cada sector, el conductor realizaba una prueba paralela dirigiendo su atención hacia las zonas de clasificación previamente definidas (FV, L, M, S, R y T). El objetivo de esta fase fue analizar tanto el tiempo de procesamiento como la capacidad del sistema, y de la red neuronal propuesta, para proporcionar retroalimentación sobre la presencia o ausencia de un escenario potencialmente peligroso en el entorno frontal del vehículo. Cabe destacar que, por razones de seguridad, en esta etapa se omitió el factor de velocidad vehicular, evaluando únicamente el funcionamiento estático de los clasificadores. Esto porque el parámetro de velocidad solo influye en las distancias de frenados las cuales se pueden calcular al mismo tiempo que los algoritmos se ejecutan. Es decir, teniendo en cuanta que la velocidad del vehículo influye proporcionalmente a la distancia de frenado (Tabla 4.7), solamente se ha de realizar una diferencia entre la distancia del objeto y el valor BD calculado para estimar la ventana de frenado y en combinación con la identificación de atención del conductor realizar la retroalimentación.

El sistema ADAS propuesto mostró un buen desempeño bajo condiciones de iluminación favorables (por ejemplo, cielo nublado o sombra proyectada sobre el vehículo), alcanzando una tasa de precisión del 98 %. Bajo diferentes condiciones de iluminación, los resultados fueron acordes a lo esperado: se obtuvo una precisión del 96 % en la mañana (10:00 a.m., con luz solar directa sobre el vehículo) y del 91 % en la tarde (6:00 p.m., con iluminación solar

directa al interior del vehículo).

Estos resultados en la clasificación de escenarios de posible peligro se correlacionan con los porcentajes de exactitud obtenidos por los algoritmos de inteligencia artificial utilizados: YOLO-R y la red neuronal entrenada con las zonas de atención definidas, ambos superando el 90 % de precisión. Este comportamiento puede explicarse mediante la formulación de la ecuación 3.11, donde la activación de la alarma depende de entradas binarias provenientes de las clasificaciones generadas por las redes neuronales. Por lo tanto, puede afirmarse que, mientras la clasificación se mantenga estable y constante, el sistema puede operar sin inconvenientes. En la mayoría de los casos, los errores en la activación o la falta de activación de la alarma estuvieron relacionados con una clasificación incorrecta de la pose facial o una estimación imprecisa de la distancia al objeto detectado.

Con base en las pruebas realizadas bajo condiciones reales de iluminación, se determinó que el sistema es aplicable a escenarios de baja velocidad (<30 km/h), como fraccionamientos, escuelas o cruces peatonales. En este rango, un rendimiento de 17 cuadros por segundo resulta adecuado, ya que proporciona una frecuencia de actualización suficiente para detectar oportunamente el estado del conductor y emitir alertas preventivas antes de que ocurra una colisión. Esta afirmación se encuentra en concordancia con las guías estadounidenses de seguridad vial, que consideran una desaceleración constante del vehículo en situaciones de emergencia.

Capítulo 5

Conclusiones

5.1. Conclusiones Generales

El sistema de estimación de pose demostró su capacidad para clasificar el movimiento de la cabeza mediante un cálculo continuo de la pose, con un error promedio de 0.87° en RMSE en los ángulos de Euler, lo cual representa una de las principales fortalezas del presente trabajo, lo cual a su vez habilitó el desarrollo de una red neuronal para clasificar los ángulos para detectar la zona de atención de un conductor. La medición continua a su vez abre la puerta a futuras investigaciones enfocadas en el análisis del flujo de movimiento de la cabeza del conductor para la clasificación de su comportamiento.

La técnica de triangulación demostró ser una solución efectiva para el cálculo de puntos en 3D, entregando mediciones precisas que permitieron resolver correctamente las ecuaciones necesarias para obtener los ángulos de Euler. Por otra parte, el algoritmo de extracción de características resultó ser una alternativa eficaz al método de coincidencia por plantillas, ya que logró emparejamientos confiables entre puntos faciales, gracias a la estructura del algoritmo FLD. Identificar esta cualidad permitió un ahorro considerable de tiempo en la etapa de búsqueda y optimización del emparejamiento.

El sistema ADAS desarrollado requiere únicamente un par de imágenes como entrada para generar, en tiempo real, una detección de posibles situaciones de peligro, con el objetivo de reducir el número de accidentes viales. Esta detección se logra mediante la estimación simultánea de la pose de la cabeza (HPE) y la clasificación de la zona visual del conductor, alcanzando una exactitud del 99.77 % bajo condiciones de iluminación controlada, con una velocidad de procesamiento de 17 FPS utilizando una resolución de 320×240, lo que genera que bajo las métricas establecidas de la AASHTO el sistema es viable para su implementación en zonas de baja velocidad, como se evidencia en la Tabla 4.7, dado a los cálculos de las ventanas de frenado.

Finalmente, es importante destacar que durante la etapa de investigación sobre métodos para el análisis facial en el estado del arte de detección de puntos de referencia 3D mediante visión estéreo, se encontró solo un conjunto de datos disponible. Sin embargo, este no cumplía con las características de nuestro sistema experimental Fransens, Strecha y Van Gool, 2005, lo que complicaba la posibilidad implementar algoritmos entrenables como redes neuronales convolucionales (CNN) para estimar pose facial, problemática que también es reportada por Li et al., 2021. Esta limitación nos restringió a realizar comparaciones generales entre modelos, debido a la falta de acceso a los programas originales o a los conjuntos de datos utilizados por otros autores, siendo la excepción el método OpenFace.

5.1.1. Cumplimiento de Objetivos

Las capacidades del sistemas visión estereoscopica, que habilita la percepción en tres dimensiones, permitieron desarrollar un sistema de extracción de características faciales 3D, generando nubes de puntos con las que se estimó la pose facial formulando el problema como una instancia de *Perspectiva de n Puntos* (PnP).

Se diseñó un sistema intuitivo de estimación de pose facial, apoyado en implementaciones matemáticas clave como la triangulación para obtener coordenadas tridimensionales, y un algoritmo de extracción de características que sustituyó eficazmente la coincidencia por plantillas para generar nubes de puntos consistentes. Estas nubes, capaz de calcularse a una frecuencia constante de 17 FPS, permitieron una estimación continua de la pose facial.

El sistema alcanzó resultados con bajo nivel de error, tanto en términos de RMSE como MAE, como se muestra en la Tabla 4.6. Además, fue capaz de realizar una estimación de los tres ángulos por conjunto de imágenes capturadas, lo que lo convierte en un sistema de análisis continuo en tiempo real con resultados competitivos de error ligeramente por encima de un grado de error (utilizando imágenes de 320×240), además si se desean mediciones con mejor exactitud se puede cambiar la resolución de entrada de las imágenes a 640×420 pixeles para obtener en promedio mediciones por debajo de un grado de error, pero esto disminuirá la velocidad de ejecución a 7 FPS. Por último, las pruebas experimentales demostraron su exactitud dentro del rango de 30° de visión en los ejes vertical y horizontal.

Asimismo, se construyó una base de datos con los registros de estimación de pose, utilizada para entrenar una red neuronal capaz de clasificar distintas

zonas de atención definidas en el contexto de la aplicación ADAS. La correcta delimitación durante la toma de datos fue un factor clave para la efectividad del modelo entrenado.

Por último, el sistema demostró aplicabilidad en escenarios reales con iluminación natural. Fue validado dentro de un vehículo, logrando clasificar correctamente las zonas de atención con una exactitud superior al 90 %, estando dentro de un escenario real de aplicación con iluminación natural y no controlada como la de un laboratorio.

Bibliografía

Abate, Andrea F et al. (2021). «Partitioned iterated function systems by regression models for head pose estimation». En: *Machine Vision and Applications* 32, págs. 1-8.

Abate, Andrea F et al. (2022). «Head pose estimation: An extensive survey on recent techniques and applications». En: *Pattern Recognition* 127, pág. 108591.

Addanki, Sai Charan et al. (2020). «Analysis of traffic related factors and vehicle environment in monitoring driver's driveability». En: *International Journal of Intelligent Transportation Systems Research* 18.2, págs. 277-287.

Administration, National Highway Traffic Safety et al. (2016). «2015 motor vehicle crashes: overview». En: *Traffic safety facts: research note* 2016, págs. 1-9.

Ahmed, Mohiuddin, Raihan Seraj y Syed Mohammed Shamsul Islam (2020). «The k-means algorithm: A comprehensive survey and performance evaluation». En: *Electronics* 9.8, pág. 1295.

Alagha, Mahmoud A et al. (2022). «Objective grading facial paralysis severity using a dynamic 3D stereo photogrammetry imaging system». En: *Optics and Lasers in Engineering* 150, pág. 106876.

Alqahtani, Faleh et al. (2020). «3D face tracking using stereo cameras: A review». En: *IEEE Access* 8, págs. 94373-94393.

Amir, Zadeh, Baltrusaitis Tadas y Morency Louis-Philippe (2017). «Convolutional experts constrained local model for facial landmark detection». En: *Proceedings of the IEEE CVPRW*, págs. 2051-2059.

Aouada, Djamila et al. (2021). «Dense and sparse 3D deformation signatures for 3D dynamic face recognition». En: *IEEE Access* 9, págs. 38687-38705.

Baltrušaitis, Tadas, Peter Robinson y Louis-Philippe Morency (2016). «Openface: an open source facial behavior analysis toolkit». En: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, págs. 1-10.

Baltrusaitis, Tadas et al. (2018). «Openface 2.0: Facial behavior analysis toolkit». En: *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE, págs. 59-66.

Borghi, Guido et al. (2018). «Face-from-depth for head pose estimation on depth images». En: *IEEE transactions on pattern analysis and machine intelligence* 42.3, págs. 596-609.

Burges, Christopher JC (1998). «A tutorial on support vector machines for pattern recognition». En: *Data mining and knowledge discovery* 2.2, págs. 121-167.

Chethan, KS et al. (2019). «Analysis of image quality using sobel filter». En: *2019 Third International Conference on Inventive Systems and Control (ICISC)*. IEEE, págs. 526-531.

Cilimkovic, Mirza (2015). «Neural networks and back propagation algorithm». En: *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin* 15.1, pág. 18.

Combs, Tabitha S et al. (2019). «Automated vehicles and pedestrian safety: exploring the promise and limits of pedestrian detection». En: *American journal of preventive medicine* 56.1, págs. 1-7.

Cristianini, Nello y John Shawe-Taylor (2004). *Support Vector Machines and other kernel-based learning methods*. Cambridge.

Dalal, Navneet y Bill Triggs (2005). «Histograms of oriented gradients for human detection». En: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee, págs. 886-893.

Deng, Jiankang et al. (2020). «Retinaface: Single-shot multi-level face localisation in the wild». En: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, págs. 5203-5212.

Flores-Fuentes, Wendy et al. (2023). «3D spatial measurement for model reconstruction: A review». En: *Measurement* 207, pág. 112321.

Fransens, Rik, Christoph Strecha y Luc Van Gool (2005). «Parametric stereo for multi-pose face recognition and 3D-face modeling». En: *AMFG*. Springer, págs. 109-124.

Goel, Tripti, Vijay Nehra y Virendra P Vishwakarma (2011). «Comparative analysis of various illumination normalization techniques for face recognition». En: *International Journal of Computer Applications* 28.9, pág. 5.

Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. 2. MIT press Cambridge.

Hart, Peter E, David G Stork y John Wiley (2001). «Pattern classification». En.

Hashemi, Nazanin Sadat et al. (2016). «Template matching advances and applications in image analysis». En: *arXiv preprint arXiv:1610.07231*.

He, Zhenni et al. (2025). «Stereo vision based broccoli recognition and attitude estimation method for field harvesting». En: *Artificial Intelligence in Agriculture*.

Huang, Bin et al. (2020). «Improving head pose estimation using two-stage ensembles with top-k regression». En: *Image and Vision Computing* 93, pág. 103827.

Jha, Sumit y Carlos Busso (2022). «Estimation of driver's gaze region from head position and orientation using probabilistic confidence regions». En: *IEEE Transactions on Intelligent Vehicles* 8.1, págs. 59-72.

Ju, Jianping et al. (2022). «AGCNNs: Attention-guided convolutional neural networks for infrared head pose estimation in assisted driving system». En: *Infrared Physics & Technology* 123, pág. 104146.

Kazemi, Vahid y Josephine Sullivan (2014). «One millisecond face alignment with an ensemble of regression trees». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 1867-1874.

Khairdoost, Nima et al. (2020). «Real-time driver maneuver prediction using LSTM». En: *IEEE Transactions on Intelligent Vehicles* 5.4, págs. 714-724.

Khan, Khalil et al. (2021). «Head pose estimation: A survey of the last ten years». En: *Signal Processing: Image Communication* 99, pág. 116479.

Khlamov, Sergii, Iryna Tabakova y Tetiana Trunova (2022). «Recognition of the astronomical images using the Sobel filter». En: *2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, págs. 1-4.

King, Davis E. (2009a). «Dlib-ml: A Machine Learning Toolkit». En: *Journal of Machine Learning Research* 10, págs. 1755-1758.

King, Davis E (2009b). «Dlib-ml: A machine learning toolkit». En: *The Journal of Machine Learning Research* 10, págs. 1755-1758.

Kneip, Laurent, Davide Scaramuzza y Roland Siegwart (2011). «A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation». En: *CVPR 2011*. IEEE, págs. 2969-2976.

Korman, Simon et al. (2013). «Fast-match: Fast affine template matching». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 2331-2338.

Kuhn, Peter M (1999). «Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation». En.

Lau, Ivan Victor J, Adrian I Sacil y Joseph Bryan G Ibarra (2024). «Triangulation Method for Camera-Based Distance Measurement with Application for the Blind». En: *2024 4th International Conference on Information Communication and Software Engineering (ICICSE)*. IEEE, págs. 80-85.

Lee, CS George (1982). «Robot arm kinematics, dynamics, and control». En: *Computer* 15.12, págs. 62-80.

Li, Chenglong et al. (2018). «Accurate and fast 3D head pose estimation with noisy RGBD images». En: *Multimedia Tools and Applications* 77.12, págs. 14605-14624.

Li, Wanli et al. (2021). «A survey on vision-based driver distraction analysis». En: *Journal of Systems Architecture* 121, pág. 102319.

Lugaresi, Camillo et al. (2019). «Mediapipe: A framework for building perception pipelines». En: *arXiv preprint arXiv:1906.08172*.

Madrigal, Francisco y Frederic Lerasle (2020). «Robust head pose estimation based on key frames for human-machine interaction». En: *EURASIP Journal on Image and Video Processing* 2020.1, págs. 1-19.

Mady, Huda y Shadi MS Hilles (2018). «Face recognition and detection using Random forest and combination of LBP and HOG features». En: *2018 international conference on smart computing and electronic enterprise (ICSCEE)*. IEEE, págs. 1-7.

Malek, Salim y Silvia Rossi (2021). «Head pose estimation using facial-landmarks classification for children rehabilitation games». En: *Pattern Recognition Letters* 152, págs. 406-412.

Malla, Srikanth et al. (2023). «Drama: Joint risk localization and captioning in driving». En: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, págs. 1043-1052.

Officials, Transportation (2011). *A Policy on Geometric Design of Highways and Streets, 2011*. AASHTO.

Pang, Yanwei et al. (2011). «Efficient HOG human detection». En: *Signal processing* 91.4, págs. 773-781.

Peng, Bo et al. (2024). «3D-STCNN: Spatiotemporal Convolutional Neural Network based on EEG 3D features for detecting driving fatigue». En: *Journal of Data Science and Intelligent Systems* 2.1.

Po, Lai-Man y Kai Guo (2007). «Transform-domain fast sum of the squared difference computation for H. 264/AVC rate-distortion optimization». En: *IEEE transactions on circuits and systems for video technology* 17.6, págs. 765-773.

Ramírez-Hernández, Luis R et al. (2020). «Improve three-dimensional point localization accuracy in stereo vision systems using a novel camera calibration method». En: *International Journal of Advanced Robotic Systems* 17.1, pág. 1729881419896717.

Rasamoelina, Andrinandrasana David, Fouzia Adjailia y Peter Sinčák (2020). «A review of activation function for artificial neural network». En: *2020 IEEE 18th world symposium on applied machine intelligence and informatics (SAMI)*. IEEE, págs. 281-286.

Real-Moreno, Oscar et al. (2023). «Fast template match algorithm for spatial object detection using a stereo vision system for autonomous navigation». En: *Measurement* 220, pág. 113299.

Rehman, Yasar Abbas Ur, Lai-Man Po y Mengyang Liu (2020). «SLNet: Stereo face liveness detection via dynamic disparity-maps and convolutional neural network». En: *Expert Systems with Applications* 142, pág. 113002.

Rodriguez-Quiñonez, Julio C et al. (2024). «A real-time vehicle safety system by concurrent object detection and head pose estimation via stereo vision». En: *Heliyon* 10.16.

Rodríguez-Quiñonez, Julio C. et al. (2024). «Anthropometric Stereo Vision System for Measuring Foot Arches Angles in Three Dimensions». En: *IEEE Transactions on Instrumentation and Measurement* 73, págs. 1-11. DOI: 10.1109/TIM.2023.3334341.

Ruder, Sebastian (2016). «An overview of gradient descent optimization algorithms». En: *arXiv preprint arXiv:1609.04747*.

Salmane, Pascal Housam et al. (2023). «3d object detection for self-driving cars using video and lidar: an ablation study». En: *Sensors* 23.6, pág. 3223.

Sanchez-Castro, Jonathan J et al. (2020). «A lean convolutional neural network for vehicle classification». En: *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*. IEEE, págs. 1365-1369.

Singh, Sunil et al. (2021). «Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment». En: *Multimedia tools and applications* 80, págs. 19753-19768.

Slabaugh, Gregory G (1999). «Computing Euler angles from a rotation matrix». En: *Retrieved on August* 6.2000, págs. 39-63.

Statello, Emiliano et al. (2016). «Navegación por Visión Estereoscópica Asistida por GPS». En: *IEEE Argencon 2016 Congreso Bienal de IEEE Argentina*.

Suárez, Enrique J Carmona (2014). «Tutorial sobre máquinas de vectores soporte (sVM)». En: *Tutorial sobre Máquinas de Vectores Soporte (SVM)* 1, págs. 1-12.

Suthaharan, Shan y Shan Suthaharan (2016). «Support vector machine». En: *Machine learning models and algorithms for big data classification: thinking with examples for effective learning*, págs. 207-235.

Taryudi y Ming-Shyan Wang (2018). «Eye to hand calibration using ANFIS for stereo vision-based object manipulation system». En: *Microsystem Technologies* 24, págs. 305-317.

Terven, Juan, Diana-Margarita Córdova-Esparza y Julio-Alejandro Romero-González (2023). «A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas». En: *Machine learning and knowledge extraction* 5.4, págs. 1680-1716.

Vincent, O Rebecca, Olusegun Folorunso et al. (2009). «A descriptive algorithm for sobel image edge detection». En: *Proceedings of informing science & IT education conference (InSITE)*. Vol. 40, págs. 97-107.

Xu, Yuanquan, Cheolkon Jung y Yakun Chang (2022). «Head pose estimation using deep neural networks and 3D point clouds». En: *Pattern Recognition* 121, pág. 108210.

Yanagisawa, Mikio, Elizabeth D Swanson, Wassim G Najm et al. (2014). *Target crashes and safety benefits estimation methodology for pedestrian crash avoidance/mitigation systems*. Inf. téc. United States. Department of Transportation. National Highway Traffic Safety ...

Yoo, Jae-Chern y Tae Hee Han (2009). «Fast normalized cross-correlation». En: *Circuits, systems and signal processing* 28, págs. 819-843.

Yu, Yu, Kenneth Alberto Funes Mora y Jean-Marc Odobez (2017). «Robust and accurate 3d head pose estimation through 3dmm and online head model reconstruction». En: *2017 12th ieee international conference on automatic face & gesture recognition (fg 2017)*. Ieee, págs. 711-718.

Yuan, Hui et al. (2020). «Single image-based head pose estimation with spherical parametrization and 3D morphing». En: *Pattern Recognition* 103, pág. 107316.

Zhao, Xi et al. (2015). «Automatic 2.5-D facial landmarking and emotion annotation for social interaction assistance». En: *IEEE transactions on cybernetics* 46.9, págs. 2042-2055.

Zingoni, Andrea, Marco Diani y Giovanni Corsini (2019). «Tutorial: Dealing with rotation matrices and translation vectors in image-based applications: A tutorial». En: *IEEE Aerospace and Electronic Systems Magazine* 34.2, págs. 38-53.