Universidade Vigo

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

Diego Nóvoa Paradela

para a obtención do Título de Graduado en Enxeñaría Informática

Sistema de ayuda a la prevención de estados de somnolencia en vehículos mediante visión artificial

Xullo, 2022

Traballo de Fin de Grao Nº: EI 21/22-29

Titor/a: Rubén Romero González
Co-titor/a: Eva Lorenzo Iglesias
Área de coñecemento: Lenguajes y sistemas informáticos
Departamento: Informática

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1. Introducción

En los últimos años, los accidentes de tráfico han causado una gran preocupación en la sociedad. Cada día mueren una gran cantidad de personas debido a estas catástrofes. De hecho, un gran porcentaje de estas muertes son causadas por distracciones del conductor del vehículo cuando está circulando. Unas de las razones por las que se producen estas distracciones, se debe, en gran parte, a la somnolencia y el cansancio del conductor [1][2].

A modo de resumen, la somnolencia, es aquel estado de cansancio o sueño anormal, que se produce en una persona en un momento determinado. Una persona con somnolencia, suele tener la sensación de que en cualquier momento puede quedarse dormido. Por esta razón, un conductor con somnolencia puede producir escenarios peligrosos en el entorno.

Actualmente, se han implementado sistemas en los vehículos que suelen hacer mediciones a través de la presión que ejerce el conductor sobre el volante, analizar patrones de movimiento, etc. Pero estas soluciones no permiten obtener un nivel de precisión y exactitud suficiente. Incluso, observando sistemas más modernos e innovadores, hay casos en los que se utilizan cámaras, pero no están optimizadas para que funcionen correctamente en tiempo real.

El auge de la tecnología está en continuo crecimiento, y por lo tanto, en constante mejora sobre el desarrollo e investigación de nuevos algoritmos y nuevas versiones más eficientes que las actuales. Este avance ha permitido mejorar gran parte de las tareas que se llevaban a cabo en la mayoría de las áreas actuales, haciendo especial hincapié en la inteligencia artificial. Gracias al crecimiento del deep learning [3], se ha comenzado a utilizar en mayor medida las redes neuronales en todos los ámbitos, y en este caso, en el área de visión artificial, permitiendo el desarrollo de algoritmos de detección de caras y movimientos de forma mucho más eficiente.

En este proyecto se proponen diferentes técnicas de machine learning para la detección de síntomas de somnolencia, con el objetivo de reducir drásticamente los accidentes en la carretera. Para ello, se hace uso de algoritmos y tecnologías propias de visión artificial, como son las redes neuronales convolucionales [4], el algoritmo de histogramas de gradientes orientados [5], y técnicas de procesamiento de imágenes para poder abarcar los diferentes escenarios de distracción asociados al conductor.

Observando los resultados de las pruebas, se puede comprobar la gran utilidad y uso práctico del sistema en situaciones críticas de somnolencia y distracciones, enfocado en la prevención de accidentes y catástrofes tanto en los conductores de vehículos como en su entorno.

2. Antecedentes y contexto

La detección de objetos [6] es una tarea importante de visión artificial que trata de detectar instancias de objetos de cierta clase (humanos, animales, vehículos, etc) en imágenes. Su objetivo es desarrollar modelos computacionales y técnicas que proporcionen la información clave requerida por cualquier aplicación de visión artificial: ¿Qué objetos hay? y ¿Dónde?.

Siendo uno de los problemas fundamentales de la visión artificial, la detección de objetos forma la base de muchas otras tareas de computer vision, como son: *instance segmentation*, *image captioning*, *object tracking*, [7] etc. Desde el punto de vista de la aplicación, la detección de objetos se puede agrupar en dos temas de investigación "detección general de objetos" y "aplicaciones de detección", donde el primero tiene como objetivo explorar los métodos de detección de diferentes tipos de objetos bajo un marco unificado para simular la visión humana y cognición, y el último se refiere a la detección bajo escenarios de aplicación específicos, como la detección de peatones, detección de caras, detección de texto, etc.

La detección de objetos era considerada extremadamente difícil hace tan solo unos años, pero está siendo totalmente viable en la actualidad. Esto se debe a que antiguamente se han implementado diferentes algoritmos sin enfoques basados en redes neuronales, pero, con el impulso del deep learning, las redes neuronales convolucionales han servido como un gran punto de inflexión para poder conseguir realizar tareas de detección de objetos de forma eficiente y en tiempo real.

20 años atrás, la mayoría de los algoritmos de detección de objetos se han implementado en base a características “hechas a mano”. Debido a la ineficiente representación de las imágenes en ese momento, tenían que diseñar representaciones de características sofisticadas y mejorar su eficiencia acorde a las limitaciones de los recursos computacionales.

Hace 18 años P.Viola y M. Jones lograron en tiempo real, la detección de rostros humanos [8] por primera vez, sin ninguna restricción (por ejemplo, segmentación del color de la piel). El detector era decenas o incluso cientos de veces más rápido que cualquier otro algoritmo. El algoritmo de detección, que más tarde se denominó “Viola-Jones (VJ) detector”, fue nombrado así por los nombres de los autores en recuerdo de sus importantes aportes. El detector VJ sigue la forma más directa de detección, es decir, ventanas corredizas: recorrer todas las posibles ubicaciones y escalas en una imagen para ver si alguna ventana contiene una cara. Aunque parece ser un proceso muy simple, el cálculo detrás de él fue mucho más allá del poder computacional de la época. El detector VJ ha mejorado drásticamente su velocidad de detección al incorporar tres técnicas importantes: *integral image*, *feature selection*, y *detection cascades*.

El histograma de gradientes orientados (HOG) [9], es un descriptor de características propuesto en 2005 por N. Dalal y B. Triggs. HOG puede considerarse como una mejora importante en los descriptores de transformación de características invariantes de escala, diseñado para ser calculado en una cuadrícula densa de celdas uniformemente espaciadas y utiliza la normalización de contraste local superpuesta para mejorar la precisión. Aunque se pueda utilizar para detectar diferentes tipos de clases de objetos, inicialmente fue pensado para la detección de peatones, siendo usado en una gran variedad de aplicaciones de *computer vision* durante todos estos años

Hasta entonces, se ha continuado A el desarrollo de pequeñas variantes de todo este tipo de algoritmos tradicionales. El progreso conseguido comenzó a ralentizarse durante el 2010-2012, dando el salto a las redes neuronales convolucionales [10]. Es a partir de este momento cuando la detección de objetos comenzó a evolucionar enormemente.

En los últimos años el aprendizaje profundo [11] (es decir, las redes neuronales profundas) se ha convertido en el enfoque de análisis de datos dominante debido a su capacidad para lograr precisiones impresionantemente altas en una variedad de tareas informáticas importantes, entre ellas la detección de objetos. El deep learning es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial. Funciona imitando el sistema humano, permitiendo computar los datos en diferentes capas de neuronas interconectadas. Además, transforman y comunican la información de una a otra, generando el aprendizaje profundo.

Impulsadas por esta tecnología, empresas como Google, Facebook, Microsoft y Amazon están adoptando este avance y utilizan el aprendizaje profundo como técnica central para impulsar muchos de sus servicios.

En la era del deep learning, la detección de objetos se puede agrupar en dos géneros: *two-stage detection* y *one-stage detection* [12], donde el primero describe la detección como un proceso “grueso a fino” mientras que el último lo describe como “completar en un paso”.

Dentro del primer género, destacan sobre todos las redes neuronales convolucionales, mencionadas anteriormente.

Si hablamos del último género de la detección de objetos *one-stage detection*, cabe destacar el desarrollo de YOLO (You Only Look Once) [13], uno de los algoritmos que más se han utilizado, y que sigue siendo muy común en múltiples aplicaciones de la actualidad. YOLO fue el primer detector *one-stage* de la era del deep learning y es extremadamente rápido. Su abreviatura muestra cómo los autores abandonaron el anterior paradigma, dando paso a nuevos desarrollos a partir del último género del deep learning.

La detección de objetos se ha utilizado en múltiples aplicaciones, como por ejemplo en sistemas de detección de rostros, detección de peatones, detección de texto, detección de señales de tráfico, detección de objetivos con sensores remotos… etc. siendo la detección de caras una de las técnicas que se utilizan en este proyecto.

La detección de caras es una de las aplicaciones más antiguas de computer vision, consiguiendo a día de hoy que se utilice con una gran variedad de propósitos, incluso en tareas de detección de sonrisas en cámaras digitales, *face swiping* en el comercio electrónico, aplicaciones móviles de maquillaje facial, etc. Pero para que se consiga su correcto funcionamiento hay que tener en cuenta los siguientes aspectos: variaciones de las clases, ya que los rostros humanos pueden mostrar diferentes expresiones faciales, poses, movimientos, colores de piel, etc. La oclusión [14] es un factor que evita la visibilidad de un objeto, debido a otros objetos o la luminosidad del ambiente. Es un factor relevante, debido a que pueden haber caras parcialmente tapadas por otros objetos. Otro aspecto importante es la detección multiescala [15], el cual es un factor que ocurre cuando existe la posibilidad de detectar caras en una gran variedad de escalas diferentes. Por último, debe funcionar en tiempo real, teniendo en cuenta que esta tarea puede funcionar también en dispositivos móviles, los cuales requieren recursos de CPU. Para mejorar el rendimiento y asegurar el correcto funcionamiento del detector, hay diversas soluciones entre las cuales destacan métodos como la detección en cascada [16] para conseguir acelerar el proceso de detección de caras y estrategias para la mejora de la detección multiescala y oclusión.

En este caso, la aplicación principal de estas técnicas durante este proyecto recae en la detección de síntomas de somnolencia, concretamente en conductores de vehículos. De esta manera, el objetivo principal de este proyecto es construir un sistema capaz de detectar y analizar síntomas de somnolencia y posibles distracciones que pueda sufrir un conductor de un vehículo, analizando los puntos de referencia de la cara, gestos, etc, utilizando varias de las técnicas anteriormente mencionadas.

La detección de síntomas de somnolencia se puede dividir en 3 categorías principales: basado en el vehículo, basado en las reacciones físicas del conductor y basado en las señales fisiológicas del conductor del vehículo. Estas tratan de analizar la posición lateral, la velocidad y el ángulo del vehículo, la monitorización de los ojos y las posturas de la cabeza, y por último la presión sanguínea, los latidos del corazón etc. respectivamente. En este caso, el proyecto se centra en un análisis enfocado a las reacciones físicas del conductor, como son el parpadeo, las posturas y gestos…

Como se ha comentado anteriormente, una de las técnicas más comunes en la detección de somnolencia, y que se utiliza en el proyecto, es la monitorización de la tasa de parpadeo y su duración, de forma que es posible detectar posibles síntomas que permitan conocer el estado del conductor en cualquier momento. Esto se debe a que cuando una persona sufre tales síntomas en un momento dado, su forma de parpadeo y su mirada entre los párpados difieren de los mismos en situaciones normales, por lo que es fácilmente detectable. Este tipo de

sistema usa una o varias cámaras remotas, obteniendo el vídeo y aplicando más tarde los métodos de *computer vision* correspondientes, localizando secuencialmente la posición de la cara, los ojos y/o los labios, para medir el radio de cierre.

Otra de las técnicas utilizadas en el proyecto es la detección de caras. La técnica anterior (detección de puntos clave de los ojos) se podría considerar en cierta medida un caso particular de la detección de caras, ya que a partir de esta última se podrían extraer esos puntos clave. De todas formas, en este proyecto se trata esta técnica al mismo nivel que la anterior, siendo independientes entre si. Este método trata de localizar la cara del objetivo, y así poder extraer información de gran utilidad. Por ejemplo, en caso de no detectar la cara, se puede concluir que el conductor no está mirando en la dirección de la carretera, o según los valores de las medidas obtenidas a la hora de localizar el objetivo, se puede asumir que su posición facial no es correcta.

Por último, en este proyecto también se trata la posición corporal del conductor. Para ello se utilizan técnicas que permiten obtener los puntos clave del cuerpo, incluido los puntos faciales (si fuese necesario), para así poder obtener valores cuya interpretación puede dar lugar a conclusiones sobre, por ejemplo, la posición de los brazos. De esta forma, es posible obtener la posición de estos y saber si se encuentra en un estado de distracción, ya que sería posible conocer si está sujetando el volante.

Utilizando estas medidas y técnicas, se puede conseguir detectar los posibles síntomas de somnolencia del conductor. Tal sistema, se monta en un rincón discreto del vehículo, sin incidir en la visión del conductor, controlando cualquier signo de inclinación de la cabeza, ojos caídos o incluso bostezos simultáneamente.

Como se describe a lo largo del documento, existen muchas tecnologías para detectar la fatiga del conductor. Con el paso del tiempo y el avance de la tecnología, se analizan las tecnologías emergentes y determinan los mejores enfoques tratando de prevenir la causa número uno de accidentes mortales en la actualidad.

3. Objetivos

Este proyecto tiene como objetivo principal, el desarrollo de una herramienta para la prevención de estados de somnolencia, y su aplicación en carretera. Para ello, se hará uso de diversos algoritmos de *computer vision,* los cuales permiten realizar detecciones en tiempo real de la cara, los puntos clave de la misma, o el análisis de diferentes poses del cuerpo. En conjunto, permitirá detectar los posibles escenarios de somnolencia. Dichas situaciones se podrán estimar a partir de la extracción de métricas una vez obtenidos los resultados de las detecciones, conociendo valores como el EAR (métrica sobre la distancia de los párpados, tomando así decisiones sobre el parpadeo del conductor).

Como método de optimización, se realizarán diversas pruebas entre los diferentes modelos y algoritmos para poder así realizar una comparativa sobre la precisión de cada uno.

Concretamente, los objetivos son los siguientes:

- Construir un sistema que permita detectar y prevenir escenarios de somnolencia, teniendo su principal aplicación en vehículos tripulados.

- Utilizar como fuente de análisis la cara del usuario, permitiendo detectar posibles casos de fatiga o sueño.

- Integrar y utilizar diferentes técnicas de computer vision de forma simultánea en un mismo sistema

- Comparar los resultados de la implementación de varios modelos a fin de seleccionar aquellos más adecuados.

4. Marco teórico y práctico

La detección de objetos es un campo que ha tenido mucho protagonismo en los últimos años. Las técnicas de *computer vision* y el procesamiento de imágenes están implicados en esta tecnología y son ampliamente utilizados.

A medida que la investigación fue avanzando en esta área, se han encontrado numerosos algoritmos, desde técnicas más antiguas cuya funcionalidad es detectar bordes, hasta redes neuronales que han sido desarrolladas durante el auge del *deep learning* y que se han integrado en múltiples campos diferentes. Uno de ellos es la detección de objetos, y por consecuencia, se ampliaron a casos más específicos como la detección de caras, rasgos faciales o puntos clave del cuerpo entre otros.

El proyecto se basa, de forma general, en 3 fases: detección facial, detección corporal y estimación de alertas. Para ello se obtendrán las detecciones de la cara, los puntos clave de la misma y los puntos clave corporales. Además, se extraerán resultados para poder estimar las alertas correspondientes para la detección de somnolencia.

A continuación se explicarán los diversos algoritmos que han servido como herramientas para el desarrollo del proyecto.

4.1 YOLO - You Only Look Once

YOLO es la abreviatura de la expresión "You Only Look Once" (Sólo miras una vez). Se trata de un algoritmo que detecta y reconoce varios objetos en una imagen (en tiempo real). La detección de objetos en YOLO se realiza como un problema de regresión y proporciona las probabilidades de clase de las imágenes detectadas.

El algoritmo YOLO emplea redes neuronales convolucionales (CNN) para detectar objetos en tiempo real. Como su nombre indica, el algoritmo sólo requiere una propagación hacia delante a través de una red neuronal para detectar objetos.

Esto significa que la predicción en toda la imagen se realiza en una sola ejecución del algoritmo. La CNN se utiliza para predecir varias probabilidades de clase y cuadros delimitadores simultáneamente.

El algoritmo YOLO es importante por las siguientes razones:

Velocidad: Este algoritmo mejora la velocidad de detección porque puede predecir objetos en tiempo real.

Alta precisión: YOLO es una técnica de predicción que proporciona resultados precisos con mínimos errores de fondo.

Capacidad de aprendizaje: El algoritmo tiene excelentes capacidades de aprendizaje que le permiten aprender las representaciones de los objetos y aplicarlas en la detección de objetos.

El algoritmo YOLO funciona con las tres técnicas siguientes: *residual blocks*, *bounding box regression* y *intersection over union*.

Residual blocks. En primer lugar, la imagen se divide en varias cuadrículas. Cada cuadrícula tiene una dimensión de S x S. La siguiente imagen muestra cómo se divide una imagen de entrada en cuadrículas.

Ilustración 1: Grid con cada detección, Fuente: Grace Karimi. [13]

Cada celda de la cuadrícula detectará los objetos que aparezcan en su interior. Por ejemplo, si el centro de un objeto aparece dentro de una determinada celda de la cuadrícula, esta celda se encargará de detectarlo.

Bounding box regression. Un cuadro delimitador es un contorno que resalta un objeto en una imagen. Cada cuadro delimitador de la imagen consta de los siguientes atributos: anchura (bw), altura (bh), clase (por ejemplo, persona, coche, semáforo, etc. Se representa con la letra c) y centro del cuadro delimitador (bx,by).

La siguiente imagen muestra un ejemplo de cuadro delimitador. El cuadro delimitador se ha representado con un contorno amarillo.

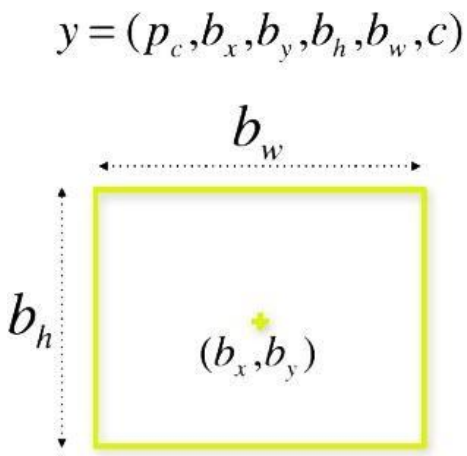

Ilustración 2: Cuadro delimitador, Fuente: Grace Karimi. [13]

YOLO utiliza una única regresión del cuadro delimitador para predecir la altura, la anchura, el centro y la clase de los objetos. En la imagen anterior, representa la probabilidad de que un objeto aparezca en el cuadro delimitador.

Intersection over union. La intersección sobre la unión (IOU) [17] es un fenómeno en la detección de objetos que describe cómo se solapan las cajas. YOLO utiliza IOU para proporcionar una caja de salida que rodea perfectamente los objetos.

Cada celda de la cuadrícula se encarga de predecir las cajas delimitadoras y sus puntuaciones de confianza. El IOU es igual a 1 si el cuadro delimitador predicho es el mismo que el cuadro real. Este mecanismo elimina los cuadros delimitadores que no son iguales al cuadro real.

A continuación, se muestra una imagen de la arquitectura interna de la red neuronal.

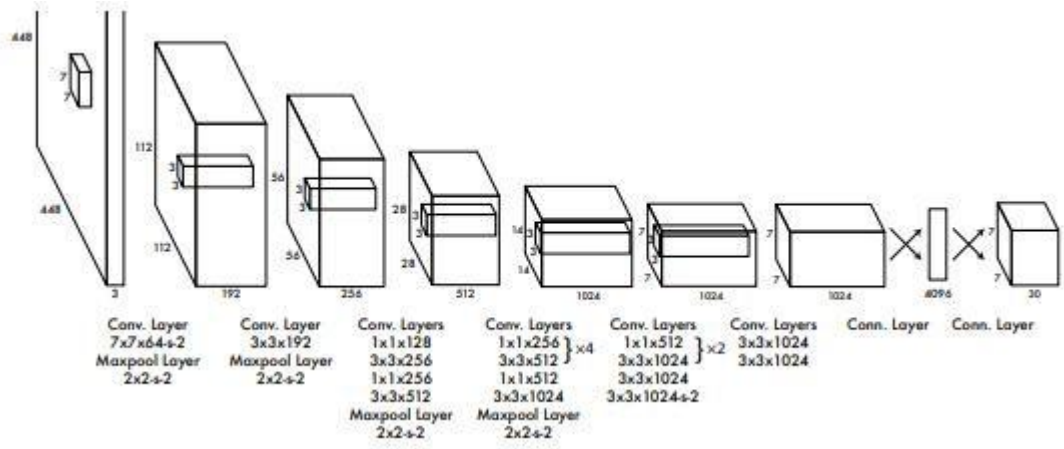

Ilustración 3: Arquitectura YOLO, Fuente: Grace Karimi. [13]

La red contiene 24 capas convolucionales seguidas de 2 capas totalmente conectadas. Además, en esta arquitectura se utilizan capas de reducción de 1 × 1 seguidas de capas convolucionales de 3 × 3. Este algoritmo utiliza imágenes de entrada de 448 x 448 para la detección, obteniendo como resultado final una salida de un tensor de 7 x 7 x 30 predicciones.

YOLO impone fuertes restricciones espaciales a las predicciones de los cuadros delimitadores, ya que cada celda de la cuadrícula sólo predice dos recuadros y sólo puede tener una clase. Esta restricción espacial limita el número de objetos cercanos que nuestro modelo puede predecir.

Además, a veces tiene dificultades con los objetos pequeños que aparecen en grupos, como las bandadas de pájaros. Dado que el modelo aprende a predecir los cuadros delimitadores a partir de los datos, tiene dificultades para generalizar los objetos con relaciones de aspecto o configuraciones nuevas o inusuales.

El modelo también utiliza características relativamente gruesas para predecir los cuadros delimitadores, ya que la arquitectura tiene múltiples capas de reducción de la muestra de la imagen de entrada.

Por último, aunque se entrene con una función de pérdida que se aproxima al rendimiento de la detección, nuestra función de pérdida trata igual los errores en los cuadros delimitadores pequeños que en los grandes. Un pequeño error en un recuadro grande suele ser benigno, pero un pequeño error en un recuadro pequeño tiene un efecto mucho mayor en el IOU. La principal fuente de error son las localizaciones incorrectas.

4.2 SSD - Single Shot Detector

Los frameworks de detección recientes pueden dividirse en dos categorías: (i) detectores de dos etapas y (ii) detectores de una etapa [18].

En los detectores de dos etapas, la primera etapa propone un conjunto de regiones candidatas a ser objeto de estudio. Tras una operación de agrupación de características en la segunda etapa, los candidatos propuestos se clasifican.

Los detectores de dos etapas han logrado el máximo rendimiento en varios datasets, como PASCAL VOC o MS COCO [19]. Por otro lado, los detectores de una etapa clasifican directamente a partir de las casillas iniciales predefinidas.

Los detectores recientes de una etapa han logrado resultados prometedores con una mayor velocidad y un menor consumo de memoria. Sin embargo, la precisión de los detectores de una etapa suele ser inferior a la de los detectores de dos etapas.

Esta diferencia de rendimiento puede atribuirse principalmente a una limitación arquitectónica de los detectores de una etapa, es decir, la falta del mecanismo *propose-and-attend* que se incluye en los detectores de dos etapas.

Durante el entrenamiento, sólo se seleccionan las cajas positivas por defecto cuando su intersección sobre la unión (IoU) con su caja real está por encima de un determinado umbral (por ejemplo, 0,5). De este modo, la configuración de los tamaños y localizaciones iniciales de las cajas por defecto es crucial para el rendimiento de la detección. De lo contrario, una configuración inicial inferior de las *bounding boxes* por defecto conduce a unas muestras de entrenamiento escasas o desequilibradas.

La mayoría de los enfoques utilizan un gran número de cajas iniciales por defecto de cajas iniciales por defecto con diferentes escalas y relaciones de aspecto, lo que no sólo requiere más parámetros y gastos generales de cálculo, sino que también es heurístico.

A continuación, se muestra una imagen de la arquitectura [20] interna de la red neuronal.

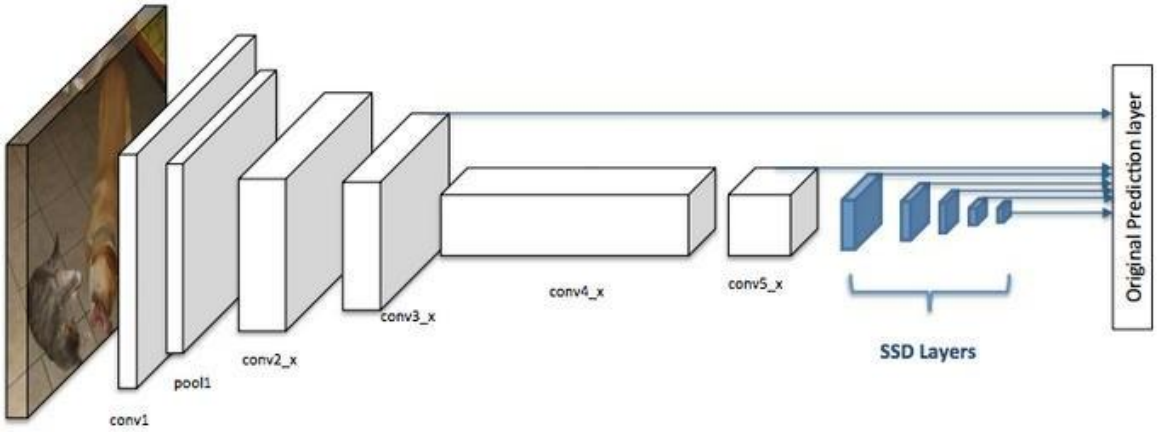

Ilustración 4: Arquitectura SSD, Fuente: ArcGIS Developer [20]

El SSD tiene dos componentes: un modelo *backbone* y la cabeza del SSD.

El modelo *backbone* suele ser una red de clasificación de imágenes preentrenada como extractor de características. Normalmente se trata de una red como ResNet entrenada en ImageNet de la que se ha eliminado la capa final de clasificación totalmente conectada. De este modo, se obtiene una red neuronal profunda que es capaz de extraer el significado semántico de la imagen de entrada al tiempo que preserva la estructura espacial de la imagen, aunque a una resolución inferior.

En el caso de ResNet34, la columna vertebral da lugar a 256 mapas de características de 7x7 para una imagen de entrada. El cabezal SSD no es más que una o varias capas convolucionales añadidas al *backbone* y los resultados se interpretan como los cuadros delimitadores y las clases de objetos en la ubicación espacial de las activaciones de las capas finales.

4.3 MTCNN - Multi Task Cascaded Convolutional Neural Network

La detección de caras es una tecnología muy necesaria en la vida real. Como las cámaras son baratas y fáciles de desplegar, la detección de caras tiene amplios escenarios de aplicación.

Los usuarios pueden desplegar sistemas de detección de caras en sistemas embebidos, que pueden separarse del entorno de red y ser más cómodos de usar. Los métodos de detección facial incluyen métodos tradicionales y métodos de aprendizaje profundo.

El método tradicional suele utilizar características Haar en el marco de AdaBoost. En cuanto a los métodos de aprendizaje profundo, la CNN multitarea en cascada (MTCNN) se utiliza ampliamente en muchos escenarios

Como variante especial de red neuronal artificial, la red neuronal convolucional (CNN) se ha utilizado ampliamente en el campo de *computer vision*.

MTCNN [21] es una red neuronal de convolución en cascada que combina la alineación de rostros y la regresión de puntos faciales. Su framework en cascada incluye redes convolucionales profundas multitarea de tres etapas.

Es un excelente método de detección facial propuesto por Zhang et al. en 2016. Este método puede alcanzar una tasa de verdaderos positivos del 95,3%. MTCNN utiliza una estructura en cascada con tres etapas (P-Net, R-Net y O-Net) para marcar las ubicaciones de las caras y las posiciones de los puntos de referencia faciales.

Sin embargo, existe un problema de tiempo real. Por defecto, la velocidad de la red no es muy elevada. Se ha investigado y obtenido hasta 10 frames por segundo en una Rasberry Pi. De todas formas, existen otros métodos que aceleran y resuelven en cierta parte este problema.

A continuación, se muestra una imagen de la estructura interna de la MTCNN [22].

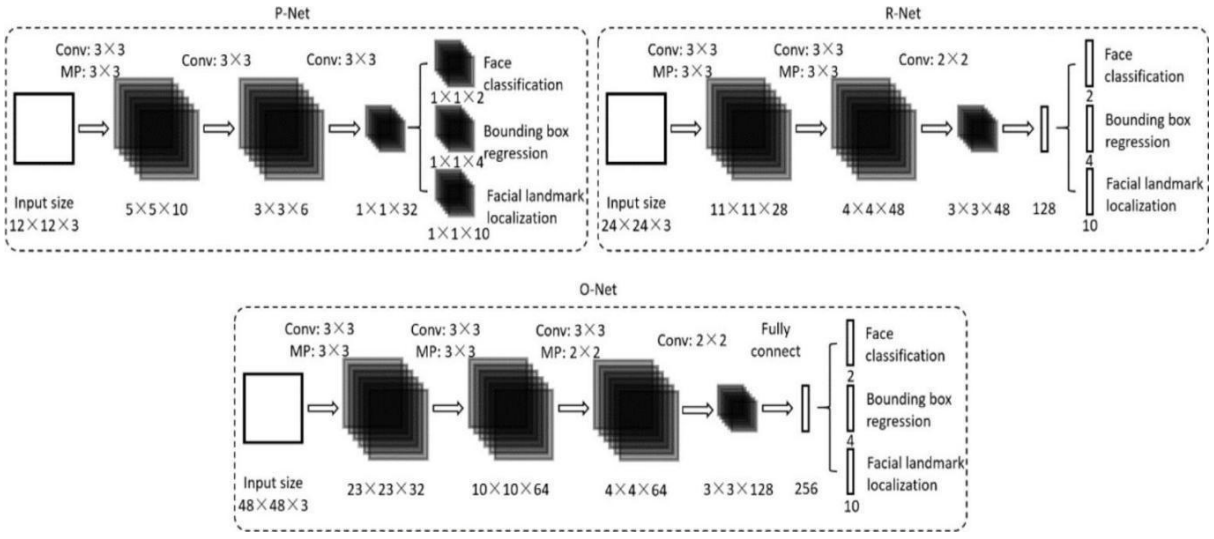

Ilustración 5: Arquitectura MTCNN, Fuente: Long-Hua Ma, Hang-Yu Fan [22]

La estructura de MTCNN consta de tres redes, donde la primera red P-Net es una red totalmente convolucional, y las dos últimas redes R-Net y O-Net son CNNs ordinarias.

El tamaño de las imágenes de entrada de MTCNN puede ser cualquiera. Dada una imagen, a menudo se redimensiona a diferentes escalas para construir una pirámide de imágenes como las entradas del siguiente marco en cascada de tres etapas.

Para entrenar las redes hay que realizar tres tareas, que son la clasificación de caras, la regresión del cuadro delimitador y la localización de puntos de referencia faciales. La pérdida para la clasificación de caras es la pérdida de entropía cruzada. El cuadro delimitador de la imagen puede contener el rostro humano, y el desplazamiento entre éste y la bounding box real más cercana debe reducirse en el proceso de entrenamiento.

La pérdida para el cuadro delimitador es la pérdida euclidiana, donde se consigue el resultado del cuadro delimitador obtenido por la red y la bounding box real más cercana.

La dimensión del cuadro delimitador es 4, y contiene las coordenadas superiores izquierdas, la altura y la anchura. Del mismo modo, la regresión del punto de referencia facial utiliza la pérdida euclidiana, donde se consigue la coordenada del punto de referencia facial obtenida de la red y la coordenada de la bounding box real.

En el proceso de entrenamiento, se utiliza un optimizador, como por ejemplo el optimizador RMSProp, la función de activación ReLU, el inicializador de pesos Xavier, y se establecen regularizadores de pesos como es L2 para cada uno de los filtros convolucionales para evitar el *overfitting*.

Por último, es recomendable considerar el uso de técnicas como NMS y mejorar su eficiencia para reducir el tiempo de cálculo de la MTCNN.

4.4 MoveNet (Pose Estimation)

MoveNet [24] es un modelo ultrarrápido y preciso que detecta 17 puntos clave de un cuerpo. El modelo se encuentra en Tensorflow con dos variantes, conocidas como Lightning y Thunder.

Lightning está pensado para aplicaciones de latencia crítica, mientras que Thunder está pensado para aplicaciones que requieren una gran precisión. Ambos modelos funcionan aproximadamente a 30+ FPS en la mayoría de los ordenadores de sobremesa, portátiles y teléfonos modernos, lo que resulta vital para las aplicaciones de fitness, deportes y salud en tiempo real.

La estimación de la postura humana ha avanzado mucho en los últimos cinco años, pero sorprendentemente aún no ha aparecido en muchas aplicaciones. Esto se debe a que se ha prestado más atención a hacer modelos de pose más grandes y precisos, en lugar de hacer el trabajo de ingeniería para hacerlos rápidos y desplegables en todas partes.

Con MoveNet, es posible diseñar y optimizar un modelo que aproveche los mejores aspectos de las arquitecturas más avanzadas, manteniendo los tiempos de inferencia lo más bajos posible. El resultado es un modelo que puede proporcionar puntos clave precisos en una amplia variedad de poses, entornos y configuraciones de hardware.

MoveNet es un modelo de estimación ascendente que utiliza mapas térmicos para localizar con precisión los puntos clave humanos. La arquitectura consta de dos componentes: un extractor de características y un conjunto de cabezas de predicción. Todos los modelos se entrenan utilizando la API de detección de objetos de TensorFlow.

El extractor de características en MoveNet es MobileNetV2 [23] con una red piramidal de características (FPN) adjunta, que permite una salida de mapa de características de alta resolución y semánticamente lograda. Hay cuatro salidas de predicción adjuntas al extractor de características, responsables de predecir:

Mapa de calor del centro de la persona: predice el centro geométrico de las instancias de la persona.

Campo de regresión de puntos clave: predice el conjunto completo de puntos clave de una persona, utilizado para agrupar los puntos clave en instancias.

Mapa de calor de los puntos clave de la persona: predice la ubicación de todos los puntos clave, independientemente de las instancias de la persona.

Campo de desplazamiento 2D por punto-clave: predice los desplazamientos locales de cada píxel del mapa de características de salida a la ubicación precisa de subpíxeles de cada punto clave.

A continuación, se muestra una imagen de la estructura interna de la MoveNet. En ella aparecen referenciadas múltiples parámetros que se salen fuera del área de este proyecto. Mas información en el paper asociado. [24]

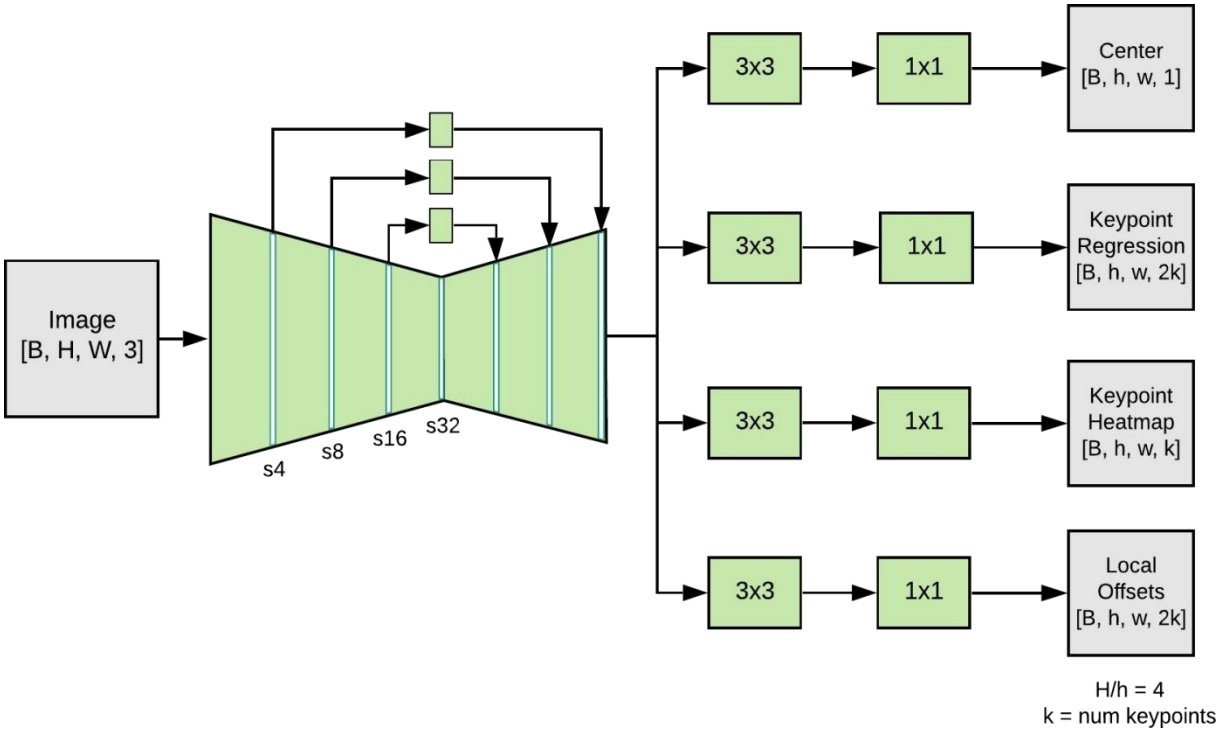

Ilustración 6: Arquitectura MoveNet, Fuente: Ronny Votel, Na Li [24]

Aunque estas predicciones se calculan en paralelo, se puede comprender el funcionamiento del modelo considerando la siguiente secuencia de operaciones:

Paso 1: El mapa de calor del centro de la persona se utiliza para identificar los centros de todos los individuos del marco, definidos como la media aritmética de todos los puntos clave pertenecientes a una persona. Se selecciona la ubicación con la mayor puntuación (ponderada por la distancia inversa al centro del marco).

Paso 2: Se produce un conjunto inicial de puntos clave para la persona cortando el resultado de la regresión de puntos clave desde el píxel correspondiente al centro del objeto. Dado que se trata de una predicción de centro-fuera -que debe operar en diferentes escalas- la calidad de los puntos clave regresados no será muy precisa.

Paso 3: Cada píxel del mapa térmico de puntos clave se multiplica por un peso que es inversamente proporcional a la distancia del punto clave corregido correspondiente. Esto garantiza que no se acepten puntos clave de personas del fondo, ya que normalmente no estarán en la proximidad de los puntos clave corregidos y, por tanto, tendrán una puntuación baja.

Paso 4: El conjunto final de predicciones de puntos clave se selecciona recuperando las coordenadas de los valores máximos del mapa de calor en cada canal de puntos clave. A continuación, se añaden a estas coordenadas las predicciones de desplazamiento local en 2D para obtener estimaciones más precisas.

MoveNet se entrenó en dos conjuntos de datos: COCO y un conjunto de datos interno de Google llamado Active. Si bien COCO es el conjunto de datos de referencia estándar para la detección, debido a su diversidad de escenas y escalas, no es adecuado para aplicaciones de fitness y baile, que exhiben poses desafiantes y un desenfoque de movimiento significativo. Active se produjo etiquetando puntos clave (adoptando los 17 puntos clave corporales estándar de COCO) en videos de yoga, fitness y baile de YouTube.

Las evaluaciones en el conjunto de datos de validación activa muestran un aumento significativo del rendimiento en relación con arquitecturas idénticas entrenadas solo con COCO. Esto no es sorprendente, ya que COCO rara vez exhibe individuos con poses extremas (por ejemplo, yoga, flexiones, y más).

4.5 HOG - Histogram of Oriented Gradients

Este método se basa en la evaluación de histogramas locales bien normalizados de las orientaciones del gradiente de la imagen en una cuadrícula densa.

A continuación se muestra la comparativa entre una imagen original y la misma aplicando el filtro de gradientes:

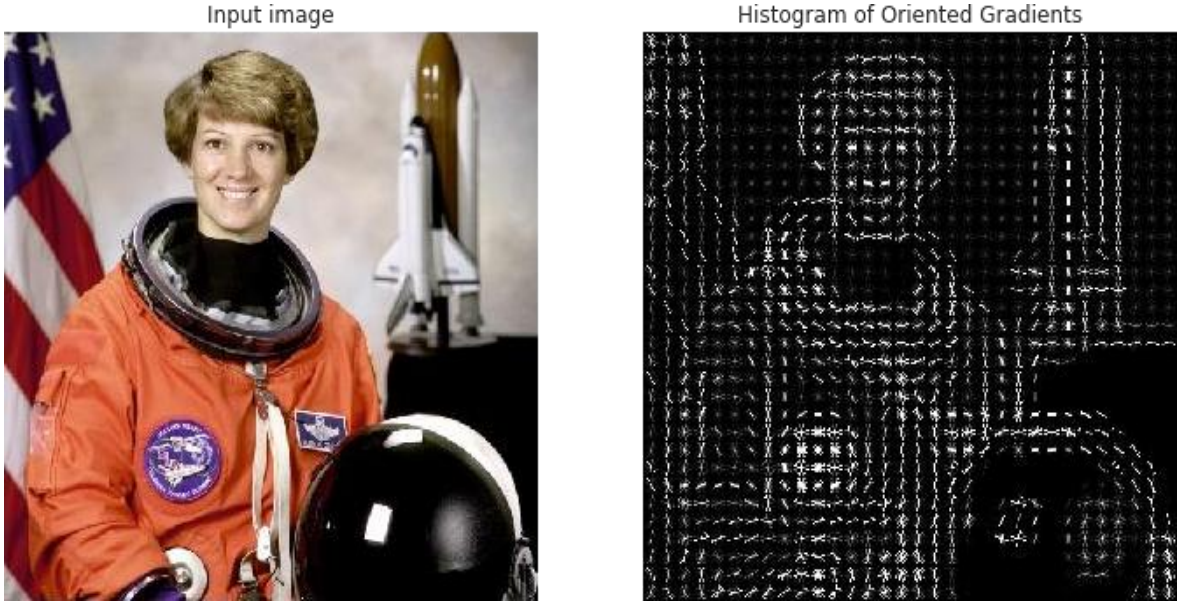

Ilustración 7: Gradient image, Fuente: Using HOG for Object Detection [25]

Características similares a estas se han utilizado cada vez más en la última década. La idea básica es que la apariencia y la forma del objeto local pueden caracterizarse bastante bien por la distribución de los gradientes de intensidad locales o las direcciones de los bordes, incluso sin un conocimiento preciso de las posiciones correspondientes de los gradientes o los bordes.

En la práctica, esto se lleva a cabo dividiendo la ventana de la imagen en pequeñas regiones espaciales ("celdas"), acumulando para cada celda un histograma local de las direcciones de los gradientes o de las orientaciones de los bordes sobre los de los píxeles de la celda. Las entradas combinadas del histograma forman la representación.

Para mejorar la invariabilidad a la iluminación, las sombras, etc., también es útil normalizar el contraste de las respuestas locales antes de utilizarlas. Esto puede hacerse acumulando una medida de la "energía" del histograma local en regiones espaciales algo más grandes ("bloques") y utilizando los resultados para normalizar todas las celdas del bloque. Se refiere a los bloques de descriptores normalizados como descriptores del Histograma de Gradiente Orientado (HOG).

4.6 Haar Cascade

Para la detección del rostro, las características haar [26] son la parte principal del clasificador en cascada.

Las características haar se utilizan para detectar la presencia de una característica en una imagen. Cada característica da lugar a un único valor que se calcula restando la suma de píxeles bajo el rectángulo blanco de la suma de píxeles bajo el rectángulo negro como se muestra en (1).

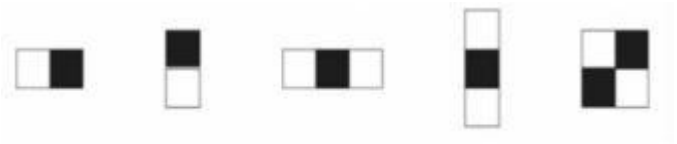

Fig 1. Haar features

Ilustración 8: Haar features, Fuente: Singh, Preeti and Mukesh Kumar Tripathi [26]

La función Haar comienza a escanear la imagen para la detección de la cara desde la esquina superior izquierda y termina el proceso en la esquina inferior derecha de la imagen. La imagen se escanea varias veces a través de las características Haar para detectar la cara de una imagen.

Para calcular las características del rectángulo se utiliza el concepto de *integral image*. Sólo se necesitan cuatro valores en las esquinas del rectángulo para calcular la suma de todos los píxeles dentro de un rectángulo determinado. En este concepto, el valor en el píxel (x,y) es la suma de los píxeles situados por encima y a la izquierda de (x,y).

La detección de rostros puede mejorarse mediante una cascada que utiliza características similares a las de Haar, como se muestra en la siguiente imagen . En esta cascada, una imagen será un rostro humano si pasa todas las etapas. Si no pasa ninguna de las etapas significa que la imagen no es un rostro humano.

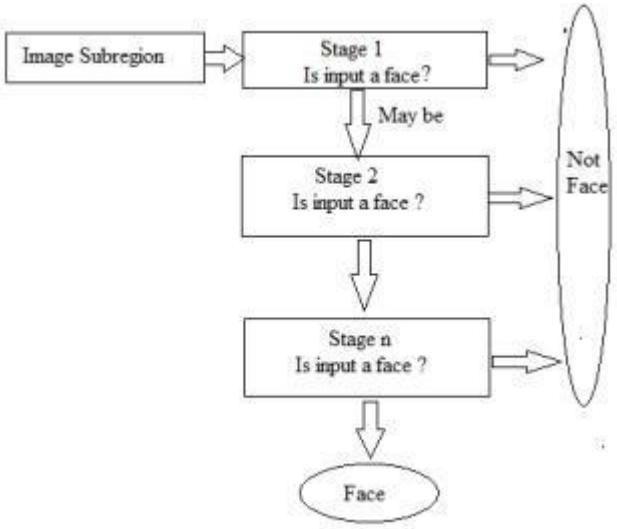

Ilustración 9: Clasificador, Fuente: Singh, Preeti and Mukesh Kumar Tripathi [26]

Desde el punto de vista experimental el clasificador en cascada haar muestra un excelente rendimiento para las imágenes que contienen un fondo simple y monótono. El enfoque de esta técnica tiene varias ventajas:

1. Gran rendimiento en términos de velocidad y fiabilidad a la hora de utilizar datasets de gran tamaño.

2. Aunque la imagen se vea afectada por la iluminación, los resultados son buenos utilizando el clasificador en cascada haar.

3. No existe ninguna restricción a la hora de utilizar complementos como gafas, etc.

4.7 Herramientas

4.7.1 Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que es un lenguaje orientado a objetos, de programación imperativa y, en menor medida, de programación funcional.

Es el lenguaje seleccionado para la implementación del sistema ya que actualmente es una de las mejores opciones para el desarrollo dentro del área de Inteligencia Artificial. Esto se debe sobre todo a la cantidad de librerías desarrolladas por la comunidad para este lenguaje.

4.7.2 Google Colab

Como plataforma de desarrollo se ha seleccionado Google Colab. Esto se debe a que es una plataforma de Google que permite trabajar en el lenguaje de programación Python en un notebook de Jupyter, a través de una máquina virtual, el cual te proporciona directamente el hardware para uso libre. Además, permite optar por una gran flexibilidad, ya que es posible ejecutar el programa desde cualquier dispositivo, con la única condición de disponer de conexión a Internet. De esta forma, el usuario podría utilizarlo simplemente accediendo al notebook.

4.7.3 Datasets

A lo largo del proyecto, se han estudiado los resultados obtenidos en diferentes datasets. En concreto, se ha utilizado el siguiente para la detección de caras: WIDER FACE: A Face Detection Benchmark. [27]

El conjunto de datos WIDER FACE es un conjunto de datos de referencia para la detección de caras, cuyas imágenes se seleccionan del conjunto de datos WIDER, que se encuentra disponible de forma pública. Se seleccionan 32.203 imágenes y etiquetan 393.703 caras con un alto grado de variabilidad en la escala, la pose y la oclusión.

El conjunto de datos WIDER FACE está organizado en base a 61 clases de eventos. Para cada clase de evento, se seleccionan aleatoriamente un 40%/10%/50% de datos como

conjuntos de entrenamiento, validación y prueba. Se adopta la misma métrica de evaluación empleada en el conjunto de datos PASCAL VOC [33].

Al igual que en el conjunto de datos Caltech [34], no se publican las cajas delimitadoras *ground truth* de las imágenes de prueba. Se deben enviar los archivos de predicción finales.

A continuación se muestra una tabla sobre la distribución de las imagenes del dataset:

	Train	**Validation**	**Test**
Wider Face (Images)	157.481	39.370	196.852
Pascal Voc (Images)	1.464	1.449	private
Caltech (Images)	12.000	3.000	15.000

Tabla 1: Distribución de imágenes del dataset

4.7.4 Librerías

Para poder desarrollar el sistema en su plenitud, se ha hecho uso de diferentes librerías que aportan diversas funcionalidades permitiendo implementarlo de una forma más sencilla y rápida. Entre estas, destacan: OpenCV [35], Tensorflow [36] y Dlib [32].

- OpenCV es una librería de código libre que proporciona múltiples funcionalidades, que, en conjunto, construyen una infraestructura que da soporte a las aplicaciones de computer vision. La librería cuenta con más de 2500 algoritmos optimizados, que incluyen un amplio conjunto de algoritmos de de aprendizaje automático, tanto clásicos como de última generación. Estos algoritmos pueden utilizarse para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en vídeos, seguir los movimientos de la cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D a partir de cámaras estereoscópicas, unir imágenes para producir una imagen de alta resolución de toda una escena, encontrar imágenes similares a partir de una base de datos de imágenes, eliminar los ojos rojos de las imágenes tomadas con flash, seguir los movimientos de los ojos, reconocer paisajes y establecer marcadores para superponerlos a la realidad aumentada, etc. Durante este proyecto, ha servido de gran ayuda en la implementación, tanto de la mayoría de los algoritmos utilizados, como del procesado y obtención de las imágenes.

- TensorFlow es una librería de código abierto, creada y mantenida por Google y publicada bajo la licencia de Apache 2.0. La API es nominalmente para el lenguaje de programación Python, aunque hay acceso a la API C++. A diferencia de otras librerías numéricas destinadas al deep learning, TensorFlow fue diseñado para su uso tanto en investigación y desarrollo como en sistemas de producción. Puede ejecutarse en sistemas de una sola CPU, en GPUs, así como en dispositivos móviles y en sistemas distribuidos a gran escala de cientos de máquinas. Durante este proyecto, ha servido de gran ayuda para la implementación de algunos algoritmos, como por ejemplo para el framework que soporta el modelo YOLO.

- Dlib es una librería de software multiplataforma de propósito general escrita en C++. Esta librería también es de código libre. Contiene componentes de software para tratar con redes, hilos, interfaces gráficas de usuario, estructuras de datos, álgebra lineal, aprendizaje automático, procesamiento de imágenes, minería de datos, análisis sintáctico de XML y texto, optimización numérica, redes bayesianas y muchas otras tareas. En este caso, se ha utilizado para conseguir tanto el detector como el clasificador de los puntos clave de la cara.

5. Resumen de la solución propuesta

5.1 Esquema general

El objetivo de este trabajo es llevar a cabo la implementación en python de un sistema de detección de síntomas de somnolencia utilizando y fusionando diferentes técnicas de visión artificial, además de observar el funcionamiento y rendimiento de diferentes algoritmos en conjunto.

Para llevar a cabo este proyecto, el sistema se implementa en Python. Para ello se obtienen los frames en tiempo real de una cámara de video que se encuentra enfocando al objetivo, preprocesando cada una de las imágenes obtenidas y preparándolas para aplicar cada una de las técnicas utilizadas.

Los métodos de *computer vision* aplicados abarcan desde la detección de landmarks faciales, como la detección de la cara del objetivo y la estimación de las poses del mismo. Una vez aplicadas, se procede a extraer las métricas correspondientes (EAR, estimación de las poses, etc) para poder tomar decisiones y comprobar si realmente la persona se encuentra en un posible estado de somnolencia.

A continuación se muestra una imagen de la arquitectura y funcionamiento general del sistema:

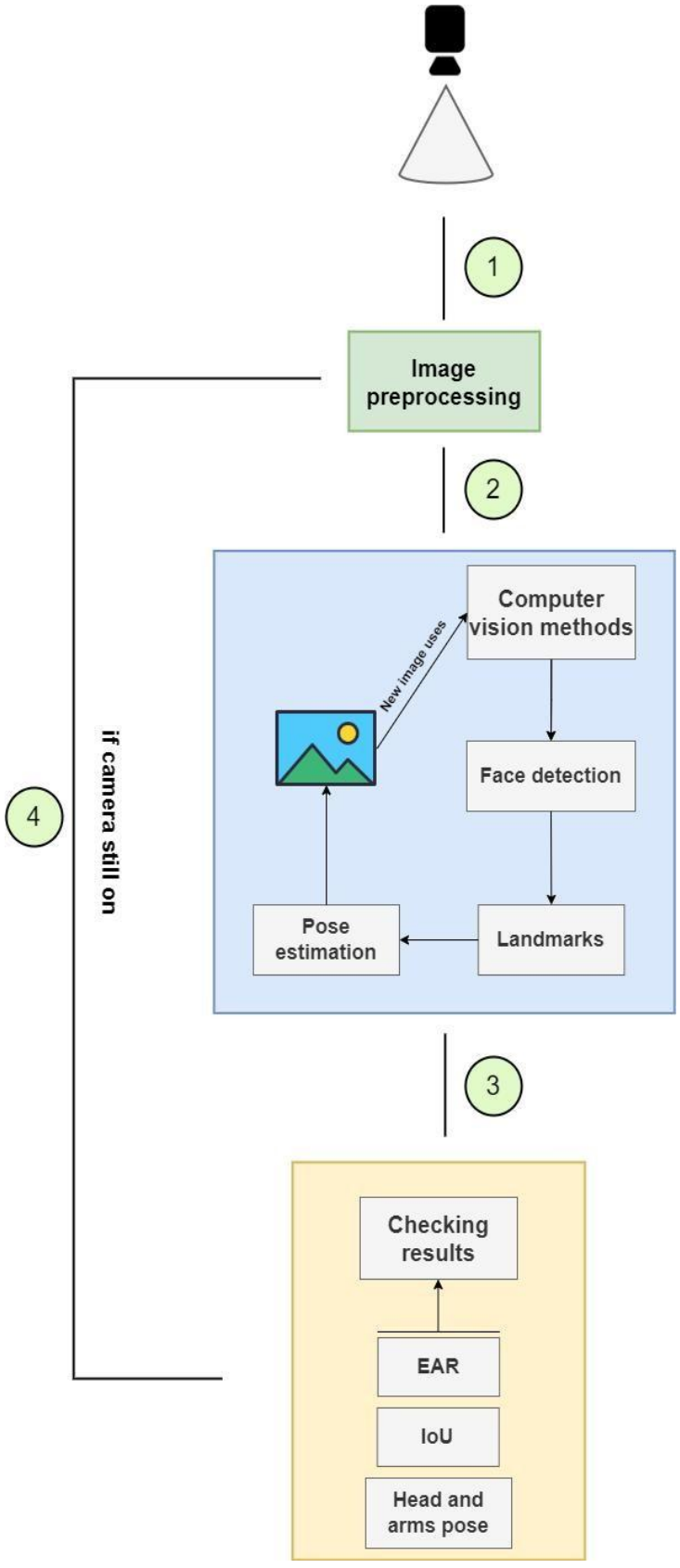

Ilustración 10: Esquema general del sistema

El funcionamiento del sistema consta de los siguientes pasos:

(1) **Procesado de la imagen**. El sistema comienza con la entrada de una imagen, la cual es captada a partir de una cámara de vídeo o webcam. Esta imagen será una entre todos los frames que la cámara capta hasta que el sistema deje de utilizarse o la webcam deje de grabar. Una vez se obtenga la imagen, se realizan diferentes modificaciones sobre esta (reescalados, conversiones de color, filtros, etc) para que el frame se ajuste a los requisitos de entrada de los algoritmos de *computer vision*.

(2) **Algoritmos de detección**. Una vez se haya procesado la imagen, esta llega como entrada a los métodos de *computer vision* para que realicen sus detecciones respectivamente. En concreto, primero la imagen llega a la tarea de *face detection*. Luego, su salida consiste en otra imagen con los resultados obtenidos y mostrados en esa imagen, la cual es redirigida al siguiente algoritmo, en este caso, detección de *landmarks*. Por último una vez conseguido los nuevos resultados en esta tarea, estos se añaden a la imagen que ha recibido para poder entregársela al último proceso de *pose estimation*. Finalmente, esta imagen es recogida y entregada al siguiente proceso: extracción de métricas y resultados.

(3) **Extracción de métricas y resultados**. En este instante, se recogen todas las métricas a partir de los resultados obtenidos en la imágen de salida del anterior paso. Ejemplos claros de estas métricas son los valores EAR y IoU, los cuales se explicarán más adelante, en el apartado 4.4.

(4) **Obtención de la imagen.** Una vez se hayan terminado los pasos anteriores, si la cámara sigue activa, se reinicia todo el proceso y se obtiene una nueva imagen, retornando al paso (1).

5.2 Preprocesado de imágenes

El preprocesado de imágenes es uno de los primeros pasos fundamentales a la hora de implementar cualquier sistema de visión artificial o tratamiento de imágenes. En este caso, la entrada del programa, son imágenes recogidas tanto en tiempo real como capturas de imágenes de una cámara de vídeo.

El sistema se implementa principalmente en Google Colab, por lo que el acceso a la webcam es diferente a si se implementara en local, ya que se trabaja sobre una máquina virtual que proporciona Google, con su correspondiente hardware.

Para ello, se utiliza un snippet de código que proporciona la plataforma, el cual invoca a un método codificado en javascript, y así poder acceder a la cámara de vídeo. De esta forma,

capturando cada uno de los frames con javascript, es posible convertir la imagen a formato OpenCV para continuar trabajando con Python.

A partir de aquí, se realizan los cambios necesarios para trabajar correctamente con cada frame. Dependiendo del algoritmo, se debe o no reescalar la imagen, crear overlays para poder mostrar las bounding boxes en caso de utilizar técnicas de detección de objetos, o incluso utilizar conversiones del espacio de color de la imagen inicial a formatos como escala de grises para asegurar el correcto funcionamiento de algoritmos como el HOG.

Otros enfoques con los que se utiliza el preprocesado de imágenes [28] son, principalmente, la supresión de cualquier tipo de "ruido" de la imagen a utilizar (normalmente el ruido se genera por la digitalización o transmisión de la imagen), eliminar la distorsión, o incluso, suprimir o resaltar otras características y atributos necesarios para la implementación posterior en técnicas como la detección de bordes y segmentación.

A continuación se muestra una imagen de la arquitectura y se explican los pasos y el funcionamiento general del procesado de imagen:

La tarea de procesamiento de imagen es el primer paso para que se cumpla el correcto funcionamiento del sistema. Como se ha explicado anteriormente, el paso 1 corresponde a la entrada de una imagen obtenida a partir de una cámara de vídeo o webcam, y el paso 2 corresponde a la salida de dicha imagen modificada y lista para que los algoritmos de detección la puedan utilizar.

Esas modificaciones pueden variar dependiendo del algoritmo a utilizar, ya que a pesar de que los algoritmos principales sean YOLO para la detección facial, HOG+SVM para detectar los landmarks de los ojos y MoveNet para la estimación de la pose, también se ha comprobado el funcionamiento utilizando otros algoritmos, como MTCNN, SSD, Haar etc, los cuales necesitan imágenes de entrada con formatos diferentes.

Los principales métodos son las siguientes:

Resize Image. Dependiendo de la resolución de la imagen de entrada, es probable que sea necesario en gran parte de los casos reescalar la imagen para que coincida con el formato de entrada del modelo que vaya a usarlo.

RGB to Grayscale. Otra de las modificaciones suele ser realizar operaciones de cambio de espacio de color a otros formatos, como escala de grises. Esto es necesario para algunos modelos o algoritmos como el HOG.

Transparent Overlay. En este caso, más que una modificación o preprocesado de la imagen inicial, es un añadido para poder recopilar la imagen de salida de cada algoritmo y poder mostrarla luego.

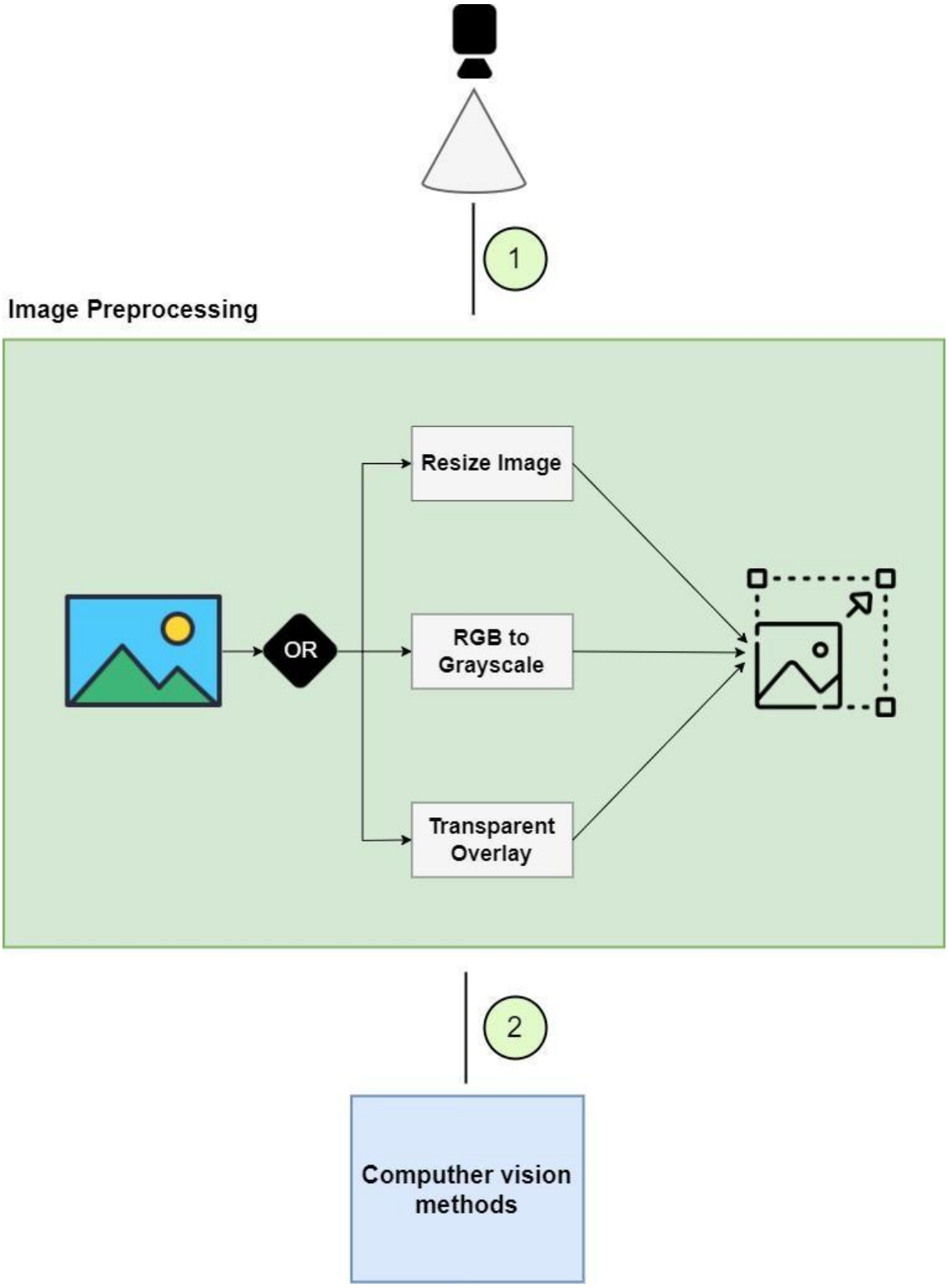

Ilustración 11: Preprocesado de imagen

5.3 Técnicas y métodos de visión

Una vez se han completado todos los procesos y cambios necesarios en el preprocesamiento de la imagen, se aplican los métodos y técnicas correspondientes. En este proyecto, se pueden resaltar 3 procesos principales consecutivos: detección de caras, detección de landmarks y estimación de la pose.

5.3.1 Face Detection

La detección de caras es una de las tareas visuales que los humanos pueden realizar sin esfuerzo. Sin embargo, en términos de computer vision, esta tarea no es fácil. Este proceso puede implicar la segmentación, extracción y verificación de rostros y posiblemente de rasgos faciales a partir de un fondo no controlado. Un sistema de detección de caras también debe ser capaz de lograr realizar la tarea independientemente de la iluminación, la orientación y la distancia de la cámara.

Dada una imagen, el objetivo de la detección de caras es determinar si hay o no una o más caras en la imagen, y, en caso de haberlas, devolver donde estas se encuentran localizadas. Esta técnica tiene asociado diversos factores que hay que tener en cuenta:

La pose. Las imágenes faciales pueden cambiar dependiendo del ángulo y la pose de la cara a detectar (pose frontal, de perfil, lateral, girado 45 grados, etc), además de otras características faciales como son la boca y los ojos que hay que tener en cuenta.

La presencia o ausencia de posibles componentes faciales. Otros de los factores que influyen en la técnica de detección de caras es la presencia de barba, bigote, gafas, etc. También, la forma, el color y el tamaño otorgan una gran variabilidad entre estos componentes.

Las expresiones faciales. Las diversas expresiones faciales afectan directamente en la detección de la cara de una persona.

La oclusión. Las caras pueden ser ocluidas o tapadas parcialmente por otros objetos. Un ejemplo de oclusión sería en una imagen con un grupo de personas, donde unas caras pueden tapar a otras si las personas se encuentran muy juntas.

La orientación de la imagen. Dependiendo del eje óptico de las cámaras, una imagen puede variar según su rotación

Por último, las condiciones de la imagen. La complejidad de poder detectar una cara varía según varios factores como la luminosidad de la imagen y la calidad de la cámara.

En este proyecto se han utilizado y probado diferentes algoritmos para, en este caso, implementar la tarea de detección de caras. Desde algoritmos más antiguos como HOG (Histogram of oriented gradients) o Haar cascade, hasta redes neuronales como MTCNN (multi-task cascaded convolutional neural network), SSD (single shot detector) y YOLO, comprobando el correcto funcionamiento de todos en el momento de integrarlos en el proyecto. De todas formas, se ha escogido por utilizar finalmente YOLO, aunque cualquiera de las anteriores opciones son igualmente correctas.

Para ello, se ha utilizado Darknet [29], facilitando el uso y mejorando el rendimiento de la red neuronal YOLO.

Darknet es un framework de código libre implementado en C y CUDA diseñado para el desarrollo de redes neuronales, sobre todo en tareas de detección de objetos. Es muy rápido y sencillo de instalar, dando la posibilidad de utilizar tanto la GPU como la CPU. Es utilizada sobre todo en redes como YOLO.

Una vez configurada, a continuación, y de forma paralela, se ha entrenado una red neuronal YOLOv3 en local, con un dataset de imágenes con bounding boxes localizando como objeto la cara de las personas. Una vez entrenada, se ha obtenido y exportado el archivo de los pesos conseguidos del entrenamiento anterior, necesario para poder introducirlo en una YOLOv4 en google colab.

Finalmente, haciendo uso de la Darknet y del snippet en javascript que se ha comentado anteriormente, conseguimos obtener las detecciones de cada frame en tiempo real.

A continuación se muestra una imagen de la arquitectura y funcionamiento general de la tarea de detección de caras:

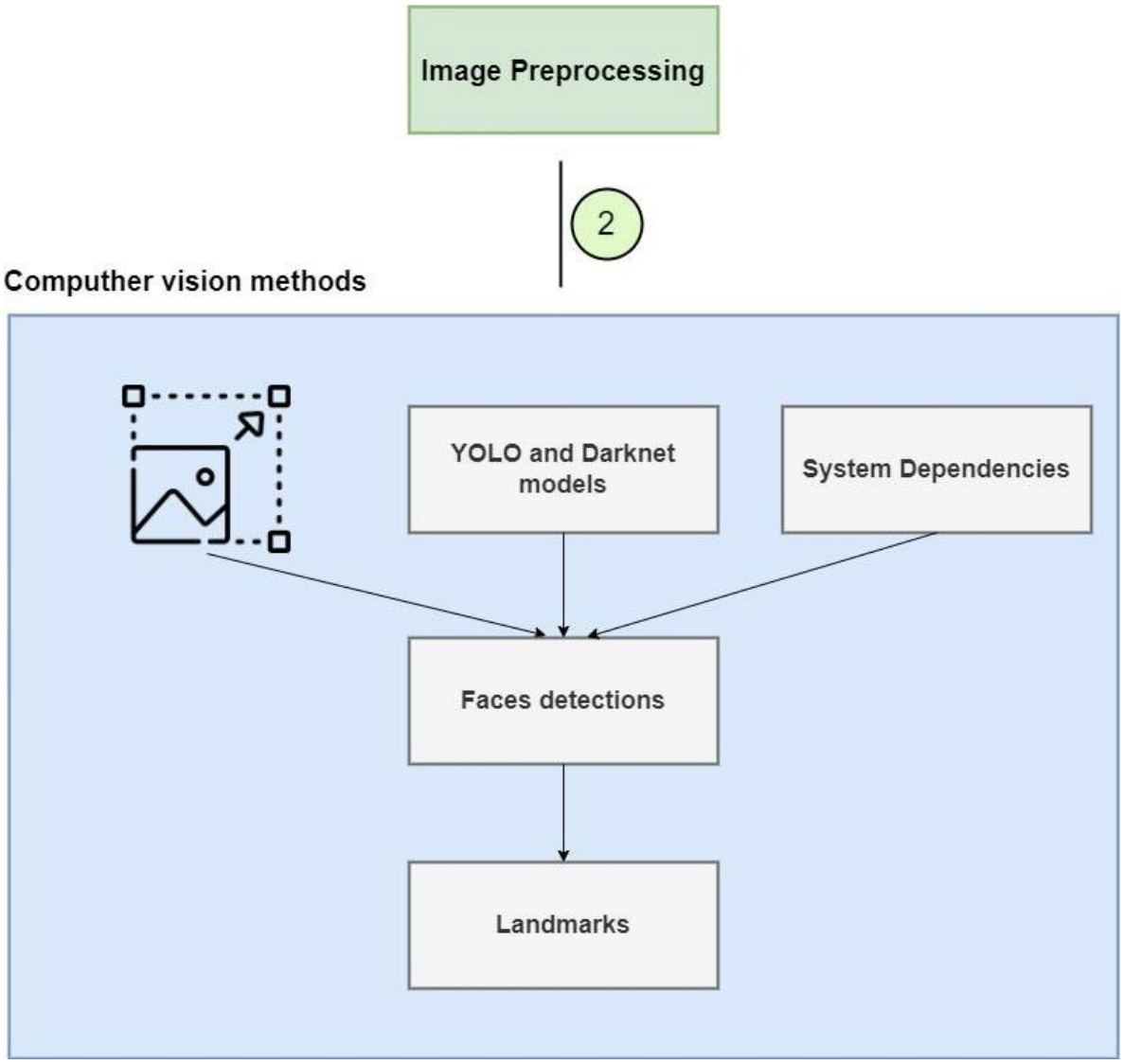

Ilustración 12: Face detections

5.3.2 Landmarks

Numerosas investigaciones realizadas en el campo de la psicología, han demostrado que un humano no puede controlar sus microexpresiones [30] y tampoco puede suprimirlas. Por lo tanto, alguien que sea capaz de reconocerlas siempre puede obtener información auténtica sobre el estado del interlocutor en el momento presente.

Como la cara es un canal muy importante de la comunicación no verbal, el análisis del comportamiento facial se ha utilizado en diferentes aplicaciones para facilitar la interacción hombre-ordenador.

Recientemente han aparecido una serie de desarrollos que están demostrando la posibilidad del análisis automatizado del comportamiento facial para la detección de los estados médicos como la depresión y el trastorno de estrés postraumático.
Además, el progreso de la tecnología de análisis de microexpresiones, junto con las tecnologías sensoriales y el procesamiento de señales, dio la oportunidad de crear el sistema destinado a mejorar la seguridad en las carreteras. El desarrollo de técnicas de *computer vision* ha permitido crear sistemas interactivos destinados a aplicaciones en los campos de la criminología, la jurisprudencia, la medicina y la seguridad informática.

Una de las etapas principales y fundamentales en las tareas de análisis de las microexpresiones faciales es la etapa de detección de puntos de referencia faciales. Puede realizarse utilizando tecnologías desarrolladas en computer vision, con el fin de minimizar el equipo utilizado y también los recursos humanos.

La detección de los ojos [31] es un caso particular de todos estos ejemplos mencionados anteriormente. Utilizando las cámaras que se encuentran disponibles en el mercado, incluidas las de los dispositivos móviles, pueden servir para múltiples aplicaciones, mejorar la experiencia del usuario en tareas cuya interacción se basa en la mirada, e incluso ayudar a los usuarios con discapacidades motoras evitando el costo de hardware especializado.

Sin embargo aunque los métodos convencionales de detección de los ojos basados en características y modelos han demostrado tener un buen rendimiento en entornos con iluminación controlada y cámaras especializadas, en entornos del mundo real sin restricciones estos métodos son superados por los métodos recientes basados en la apariencia debido a las dificultades en el modelado de factores como la iluminación cambios y otros artefactos visuales.

Para la implementación de esta tarea, se han utilizado diversas librerías, destacando sobre todo dlib. Esta librería contiene, entre otras funcionalidades, un detector de landmarks. Esta, se ha implementado utilizando el algoritmo HOG + SVM (Super Vector Machine), siendo estos algoritmos el detector y el clasificador respectivamente.

A continuación se muestra, tanto un ejemplo del cálculo de las distancias entre los puntos claves de los ojos, como el funcionamiento del detector de los puntos clave faciales utilizando la librería dlib:

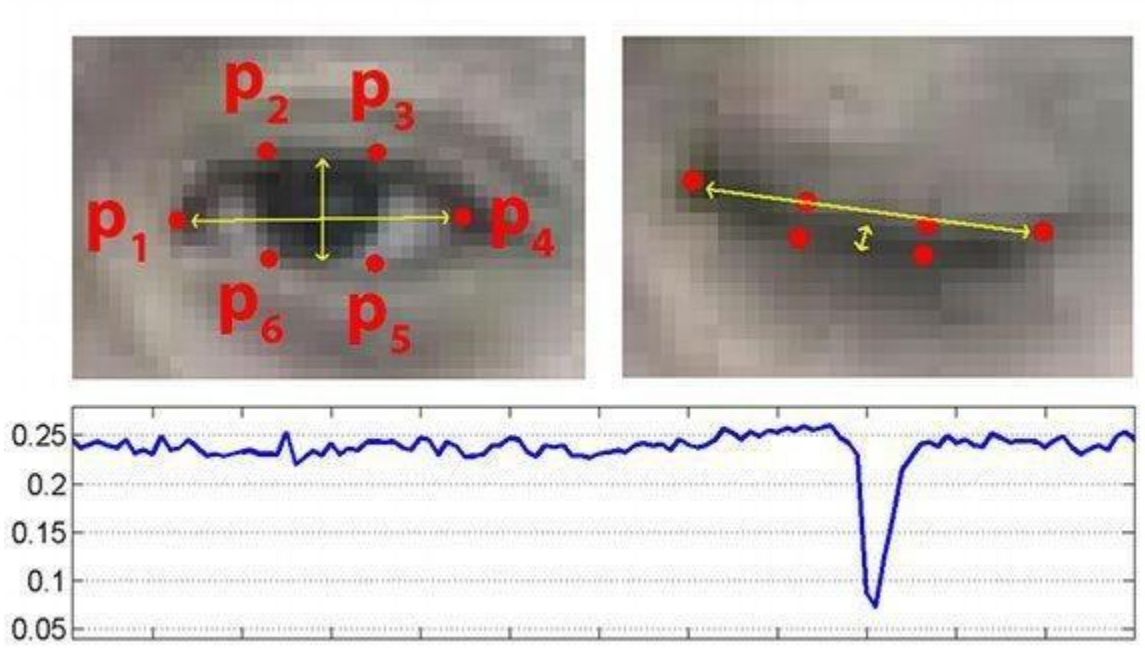

Ilustración 13: Eye distances, Fuente: Adrian Rosebrock [32]

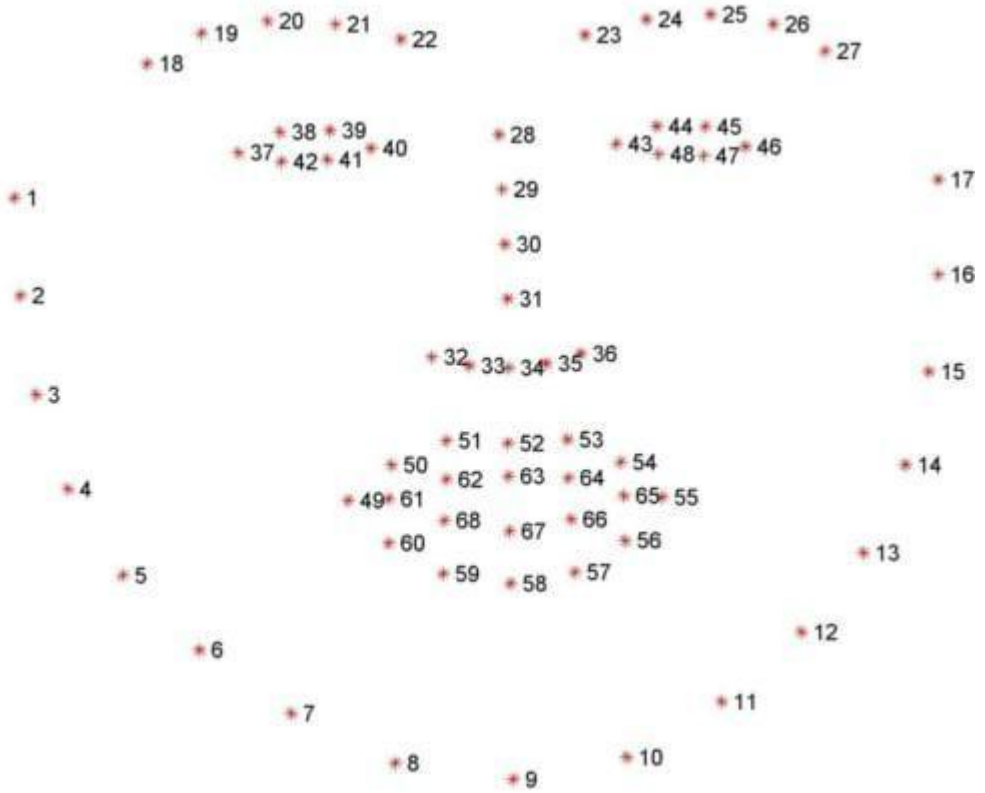

Ilustración 14: Facial Landmarks, Fuente: Adrian Rosebrock [32]

Una vez importadas las dependencias necesarias y obtener el detector, ya se podría realizar las respectivas detecciones de los landmarks. Utilizando el snippet que proporciona Google Colab, se puede trabajar con cada frame en tiempo real, y una vez realizado el preprocesamiento de la imagen (sobre todo la conversión de espacio de color de la imagen a formato de escala de grises, el cual es necesario para el correcto funcionamiento del algoritmo HOG), ya sería posible recorrer las detecciones en cada uno de los frames.

A continuación se muestra una imagen de la arquitectura y funcionamiento general de la tarea de detección de ojos:

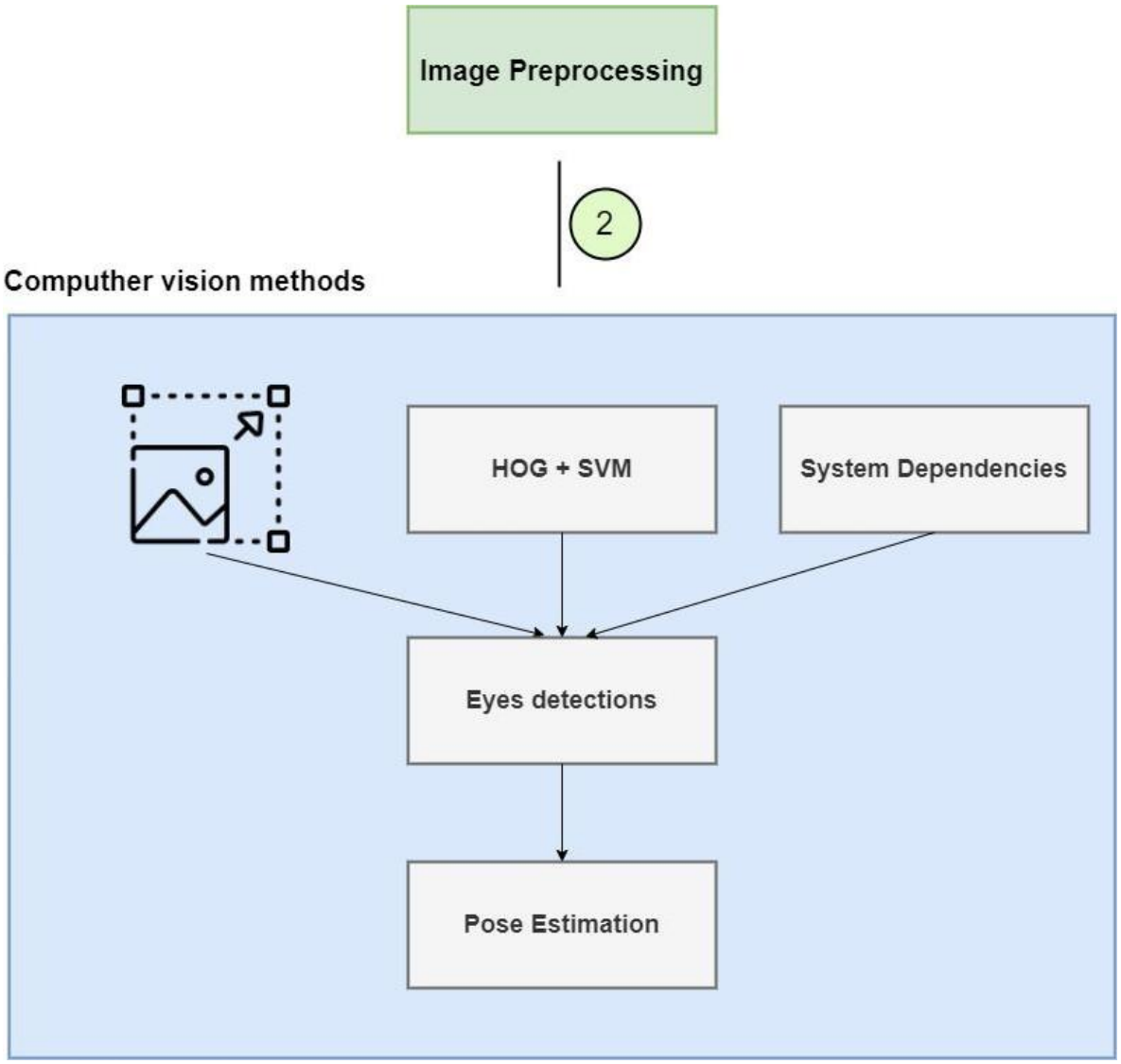

Ilustración 15: Eye detections

5.3.3 Pose estimation

Por último, otra de las técnicas utilizadas en este proyecto es la estimación de la pose humana.

La estimación de la pose humana tiene como objetivo predecir las poses de las partes del cuerpo humano en imágenes o videos. Dado que los movimientos de pose a menudo son impulsados por algunas acciones humanas específicas, conocer la pose del cuerpo de un ser humano es fundamental para el reconocimiento de la acción.

La estimación de la pose humana basada en la visión se refiere a la clasificación y localización de los puntos clave humanos en un fotograma determinado y a la conexión de los puntos clave correspondientes para aproximar el esqueleto. Los puntos clave son subjetivos y varían de una aplicación a otra. Sin embargo, una articulación que caracteriza la forma del cuerpo humano puede definirse como un punto clave. Por ejemplo, las articulaciones que conectan el hombro y la cadera para permitir movimientos hacia atrás, hacia delante, hacia los lados y de rotación se consideran puntos clave necesarios incluidos en la mayoría de los algoritmos de estimación de la pose.

La siguiente imagen muestra un ejemplo de localización de los puntos clave de una persona:

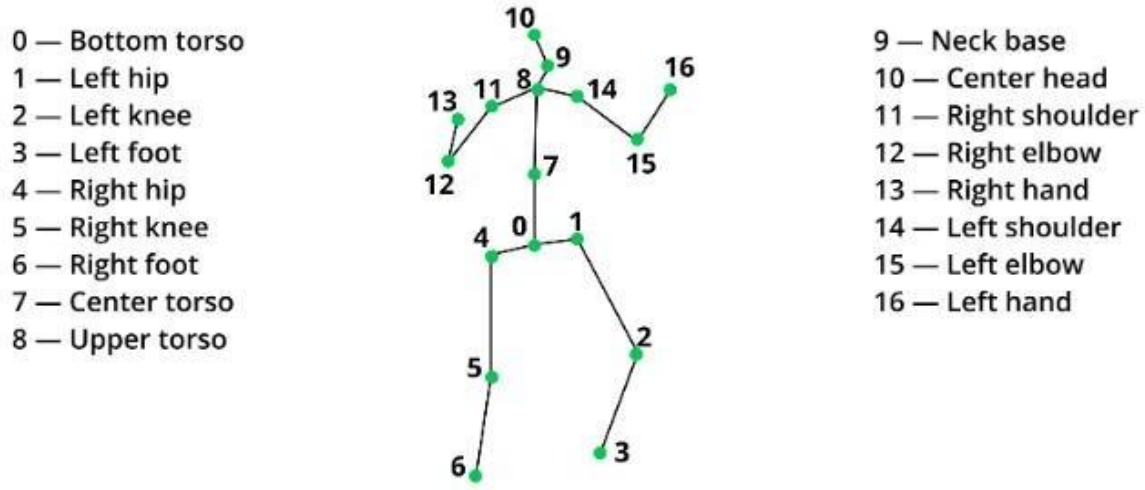

Ilustración 16: Ejemplo de estimación de pose, Fuente: Ronny Votel, Na Li [24]

La estimación de la pose es uno de los problemas clásicos de *computer vision* y ha sido un campo de investigación activo durante décadas. A medida que el campo evolucionó, el espectro de aplicaciones también se amplió, y nuevos productos innovadores lo incorporaron a sus productos y servicios. Por ejemplo, la estimación de la pose humana desempeña un papel fundamental en la realidad virtual y aumentada.

Es un ingrediente esencial en muchas aplicaciones multimedia relacionadas con el entretenimiento, como los juegos y las películas de animación. Sin embargo, las aplicaciones

no se limitan a esto y juegan un papel importante en otros dominios como las imágenes médicas, el análisis de multitudes, la cría de animales, el reconocimiento de acciones, la interacción persona-ordenador, la desprotección, el seguimiento y la videovigilancia. Otros ejemplos más generales son la segmentación, la conducción autónoma, el análisis del comportamiento y el reconocimiento de las emociones faciales.

En resumen, la estimación de la pose humana es un tema importante y tiene un enorme potencial en la industria. Con el desarrollo del aprendizaje profundo en la comunidad científica hoy en día, muchos métodos de última generación han dado excelentes resultados y han resuelto varios retos que los métodos clásicos no podían manejar.

La estimación de la pose humana se clasifica en dos tipos de metodologías: enfoques de estimación de la pose de una sola persona y de varias personas. Por un lado, los métodos unipersonales pretenden resolver el problema de regresión localizando las partes humanas de la persona que se supone que dominan el contenido de la imagen, como las orejas izquierdas/derechas, el centro del cuello y los hombros izquierdos/derechos. Por otro lado, los enfoques multipersona abordan el problema sin restricciones, ya que presenta un número desconocido de imágenes.

En la estimación de la pose de una persona, el proceso se divide en dos tipos basados en la predicción de los puntos vitales: los enfoques basados en la regresión directa y los enfoques basados en los mapas térmicos. Además, las metodologías multipersonales se clasifican en enfoques ascendentes y descendentes. En los métodos ascendentes, el primer paso es predecir todos los puntos críticos antes de asociarlos con la persona a la que pertenecen. El workflow de los enfoques descendentes tiene pasos similares pero con el orden inverso, que comienza detectando y localizando a las personas en bounding boxes individuales, y luego estima la composición de la misma.

La integración de la pose estimation en el proyecto es muy similar a la implementación de las anteriores técnicas, sobre todo tanto en los primeros como en los últimos pasos.

Una vez adquiridas las dependencias para el correcto funcionamiento del programa, es necesario obtener el modelo. Para ello existen diferentes tipos de modelos dependiendo de los requisitos y de la situación. Es posible utilizar movenets “ligeras” cuyo rendimiento es mayor pero menos preciso, y al contrario, movenets cuyo rendimiento es inferior pero que proporcionan mayor precisión a la hora de detectar los keypoints. También, es posible utilizar una posenet. En este caso, se ha utilizado una movenet singlepose ligera, ya que es suficiente.

Por último, como en los casos anteriores, haciendo uso del código que proporciona Google Colab, es posible trabajar con los frames en tiempo real, dibujando en cada una de las imágenes los puntos claves detectados y las líneas que los unen.

A continuación se muestra una imagen de la arquitectura y funcionamiento general de la tarea de estimación de poses:

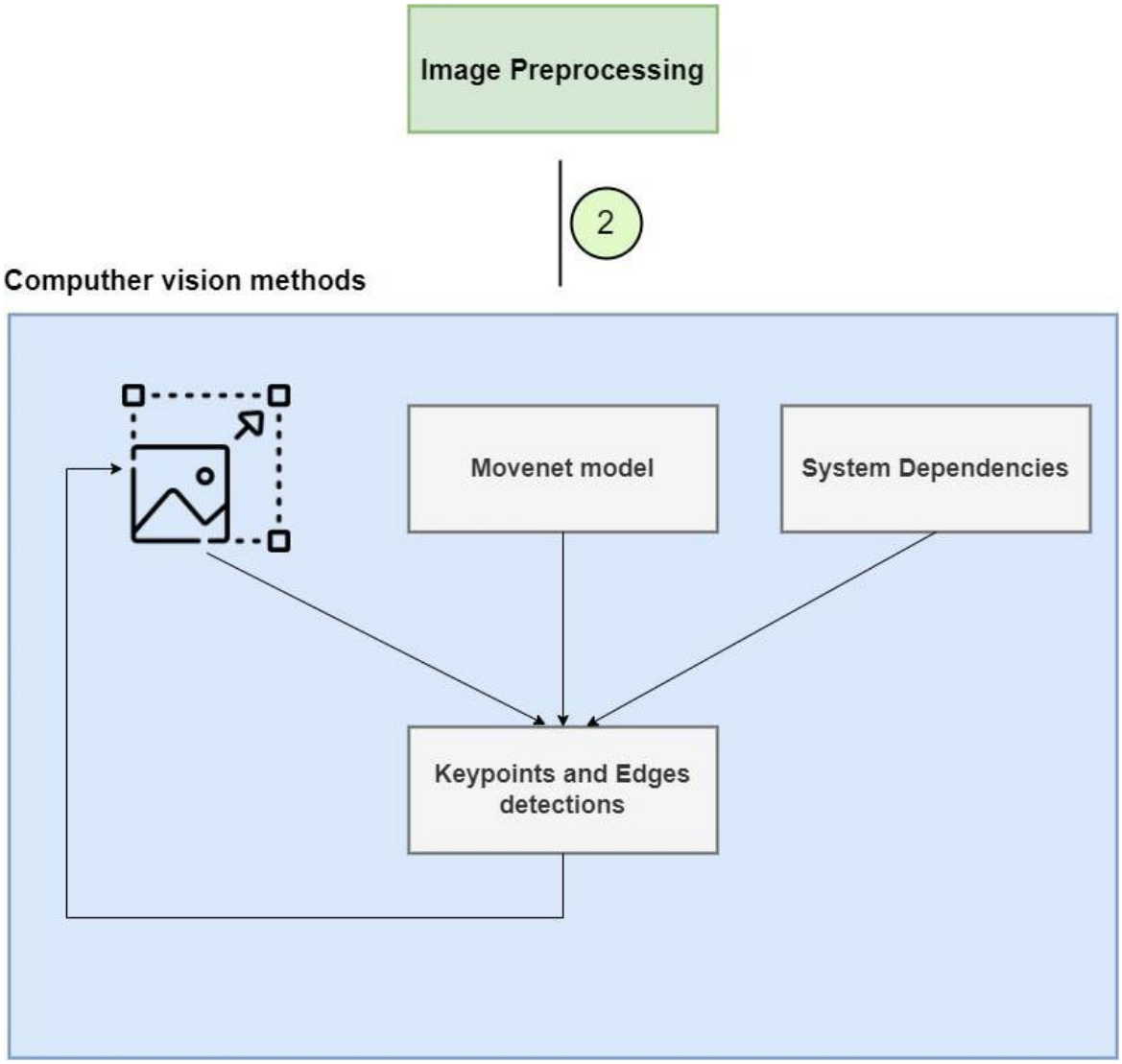

Ilustración 17: keypoints detections

5.4 Obtención de resultados y extracción de métricas

Una vez terminado el preprocesamiento de la imagen (frames, en caso de un vídeo) y haber aplicado las técnicas de computer vision, se obtienen los diferentes resultados a partir de los cuales se pueden extraer las métricas y tomar decisiones. Entre estas destacan la métrica IoU y el valor EAR.

A continuación se muestra una imagen de la arquitectura y funcionamiento general de la tarea de extracción de métricas y obtención de resultados:

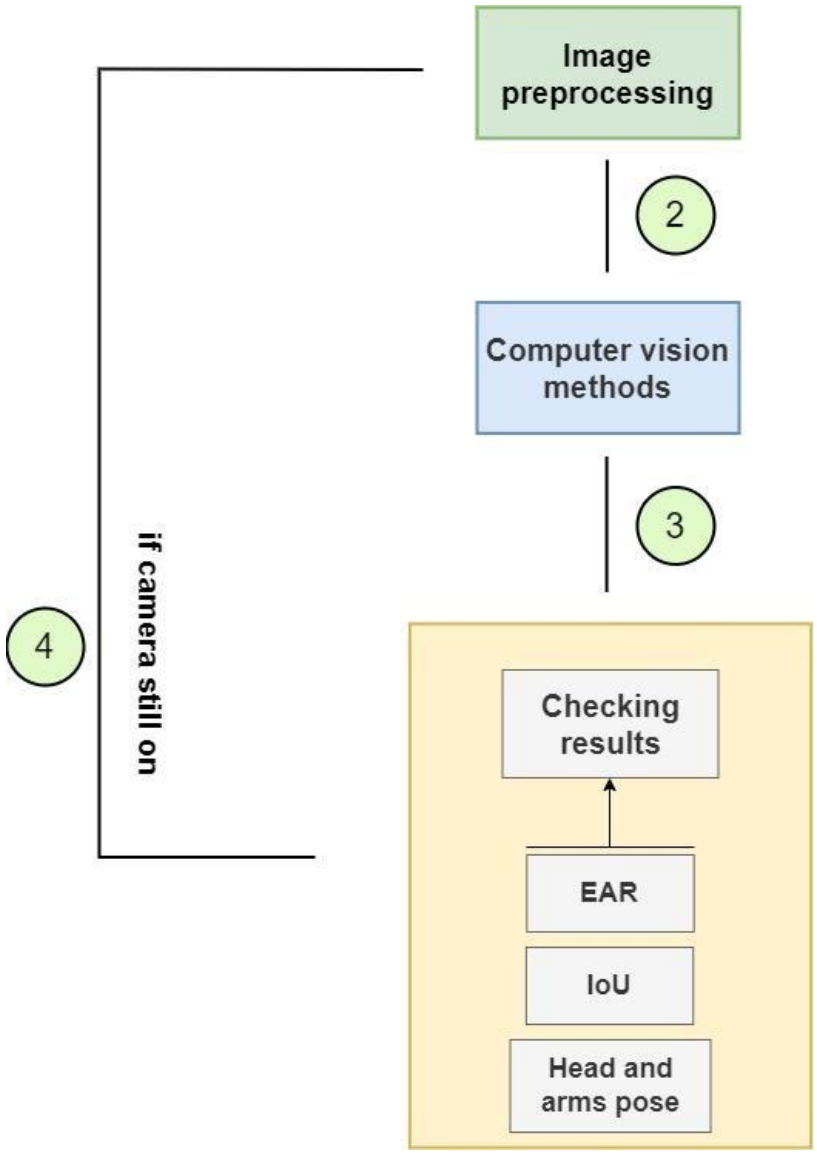

Ilustración 18: Extracción de métricas

La métrica IoU (Intersection over Union) es una métrica de evaluación, la cual es usada para medir la precisión de un detector de objetos en un dataset particular. Cualquier algoritmo que produce bounding boxes como resultado, puede ser evaluado utilizando esta métrica, como por ejemplo en redes neuronales convolucionales, HOG, etc. Para ello es necesario utilizar un dataset de imágenes con bounding boxes etiquetadas manualmente que especifica dónde se encuentra el objeto en la imagen, y un dataset de imágenes sobre el cual el modelo ha realizado predicciones localizando el objeto con bounding boxes.

Calcular este valor es tan sencillo como dividir el área de superposición entre los cuadros delimitadores, tanto del modelo como el real, por el área de unión.

También hay que definir el valor EAR (eye aspect ratio), el cual se consigue a través de una función que se utiliza para calcular la relación de distancias entre los puntos de referencia verticales del ojo y las distancias entre los puntos de referencia horizontales del ojo.

El valor de retorno del EAR será constante cuando el ojo esté abierto. Sin embargo, el valor disminuirá rápidamente hacia cero durante el parpadeo. De la misma manera, si el ojo está cerrado, el eye aspect ratio del ojo permanecerá de nuevo constante, pero será mucho más pequeño que cuando el ojo esté abierto.

Para la obtención de los resultados IoU, se han utilizado diversos modelos y algoritmos para obtener así sus comparativas, como son YOLO, HOG y HAAR.

A continuación se muestra 1 tabla y 1 gráfico de los resultados de las pruebas utilizando estos algoritmos. Para ello se ha utilizado un dataset de prueba con las bounding boxes manualmente calculadas, y cada uno de los modelos anteriormente mencionados sobre estas imágenes, obteniendo la precisión correspondiente.

	YOLO	**Haar Cascade**	**HOG**
Precisión	0.80	0.40	0.78

Tabla 2: Precisión en detecciones

En la primera parte del gráfico, se puede observar la precisión obtenida en las imágenes de prueba utilizando YOLO.

En la segunda parte del gráfico, se pueden observar los valores de la métrica obtenidos utilizando Haar Cascade.

En este caso, hay algunas imágenes que tienen una precisión de 0. Esto se debe a que, a pesar de predecir la bounding box sobre el objeto a localizar con una buena precisión, también dibuja una segunda caja en otra lugar de la imagen donde no se encuentra el objeto. Por lo tanto, cuando se calcula la métrica IoU, no existe ningún área de intersección entre la caja del modelo y la real, obteniendo como resultado el valor 0.

En la última parte del gráfico, se pueden observar los valores de la métrica obtenidos utilizando HOG.

La escala de la gráfica de barras es 0:1.

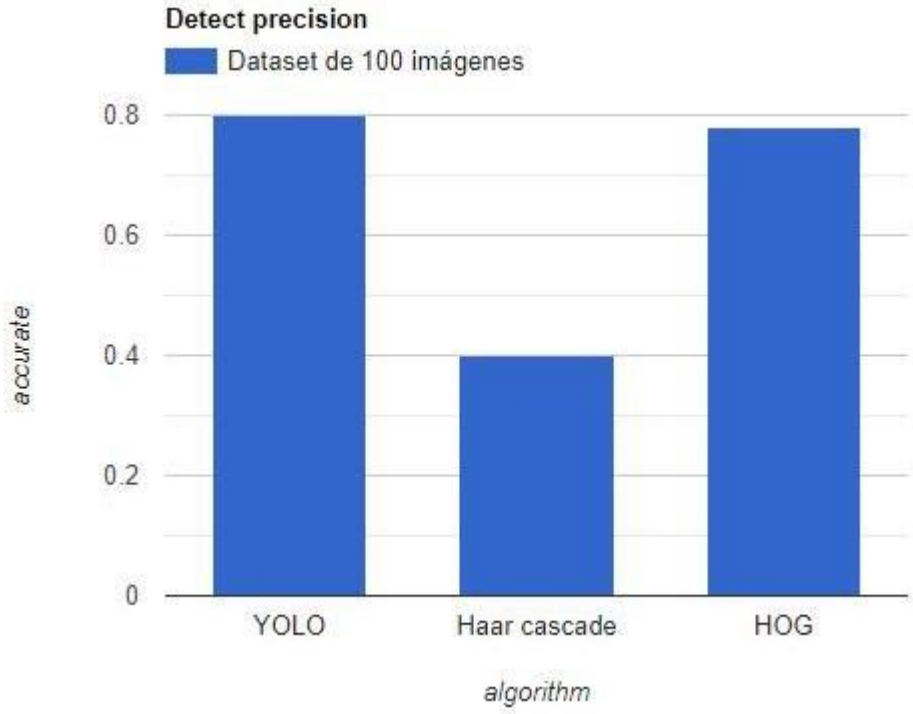

Ilustración 19: Resultados precisión

A continuación, se muestra un diagrama de ejemplo del funcionamiento del sistema en un escenario general de somnolencia:

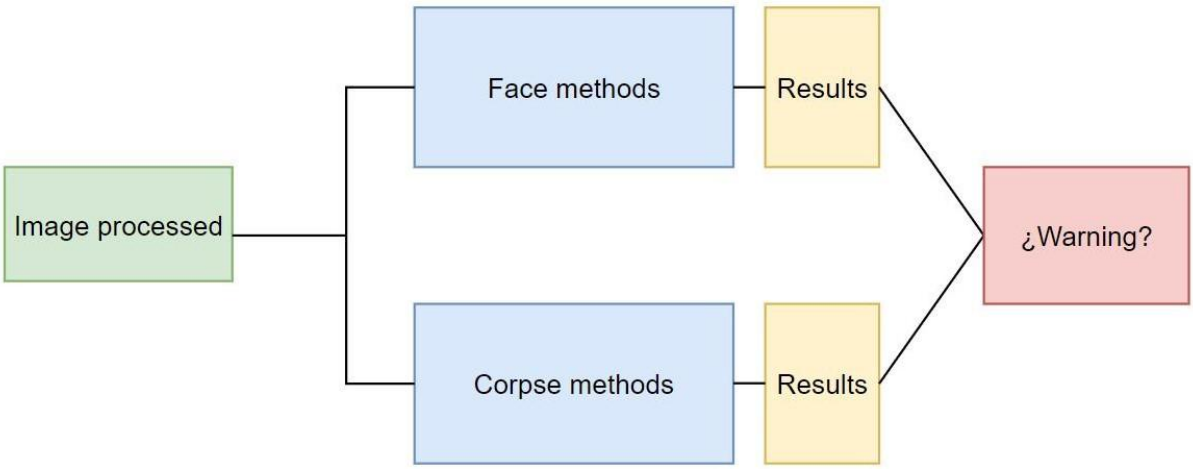

Ilustración 20: Diagrama base

Como se ha explicado durante el proyecto, el sistema se basa en la obtención de imágenes a partir de una cámara de vídeo y el tratamiento de estas para la detección de los posibles escenarios de somnolencia. Las detecciones se pueden dividir en los dos grandes bloques conocidos: la detección de puntos clave faciales y la detección de puntos clave corporales.

Una vez se han obtenido los resultados, estos son procesados y se obtiene la salida correspondiente, analizando si realmente se ha producido un escenario de somnolencia, alertando al usuario en caso de ser así.

A partir de aquí, es posible diferenciar los 2 principales escenarios del sistema:

El primero trata en alertar al usuario en base a las posibles distracciones sufridas por el conductor en relación al parpadeo (somnolencia en rasgos faciales). En el segundo caso, se alerta al usuario según su posición corporal, siendo capaz de reconocer posibles casos de somnolencia en base a la posición de sus brazos a la hora de sujetar el volante del vehículo.

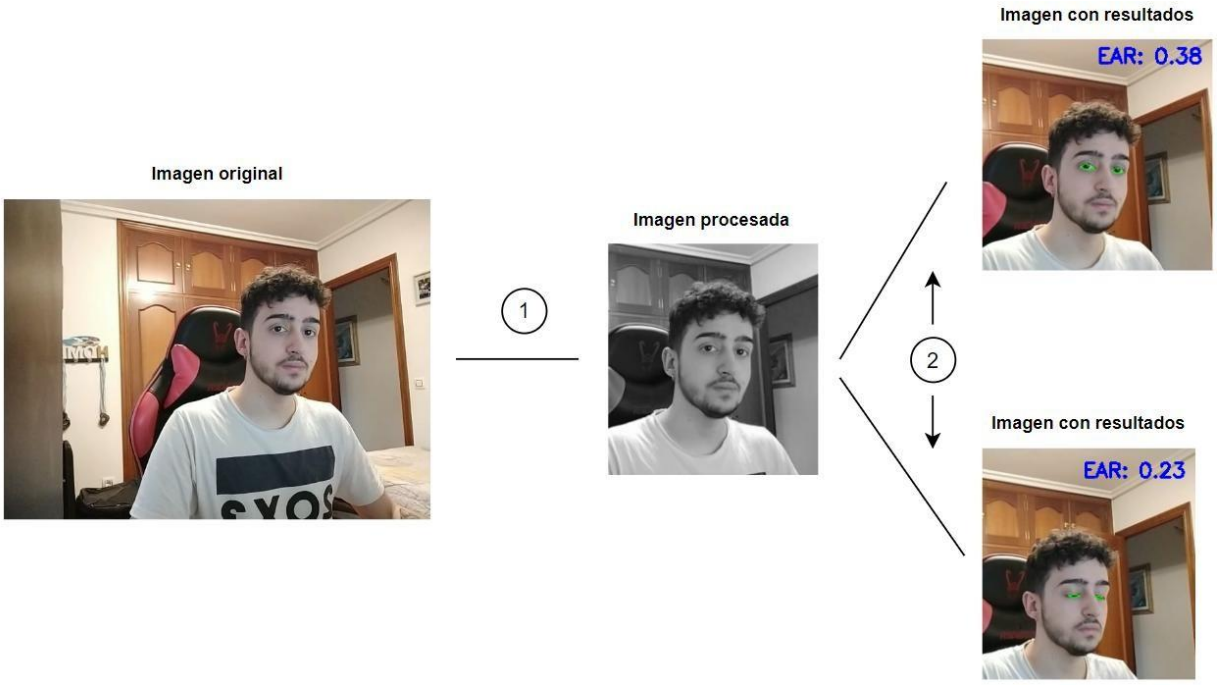

Ilustración 21: Escenario EAR

En la **ilustración 20**, se muestra el primer escenario, donde comienza con un frame original obtenido por la cámara de vídeo. Siguiendo la arquitectura del diagrama, este frame es procesado (1), como se puede observar en la siguiente imagen, para que los algoritmos de detección de puntos clave faciales correspondientes la puedan utilizar como entrada.

Una vez procesada, se redirige a estos algoritmos (2). En este caso, para extraer el valor EAR, se utiliza únicamente la detección de los landmarks de los ojos. Esto es posible observarlo en las dos últimas imágenes del diagrama, en las cuales se muestran los resultados de una persona con los ojos abiertos y la misma con los ojos cerrados respectivamente, representando un escenario de posible síntoma de somnolencia.

El threshold es posible configurarlo según las necesidades. En este caso, el intervalo de confianza resulta ser [0.3, 1). Si la métrica EAR desciende de 0.3, se puede considerar que la distancia de los párpados es lo suficientemente baja como para asumir un escenario de presencia de somnolencia.

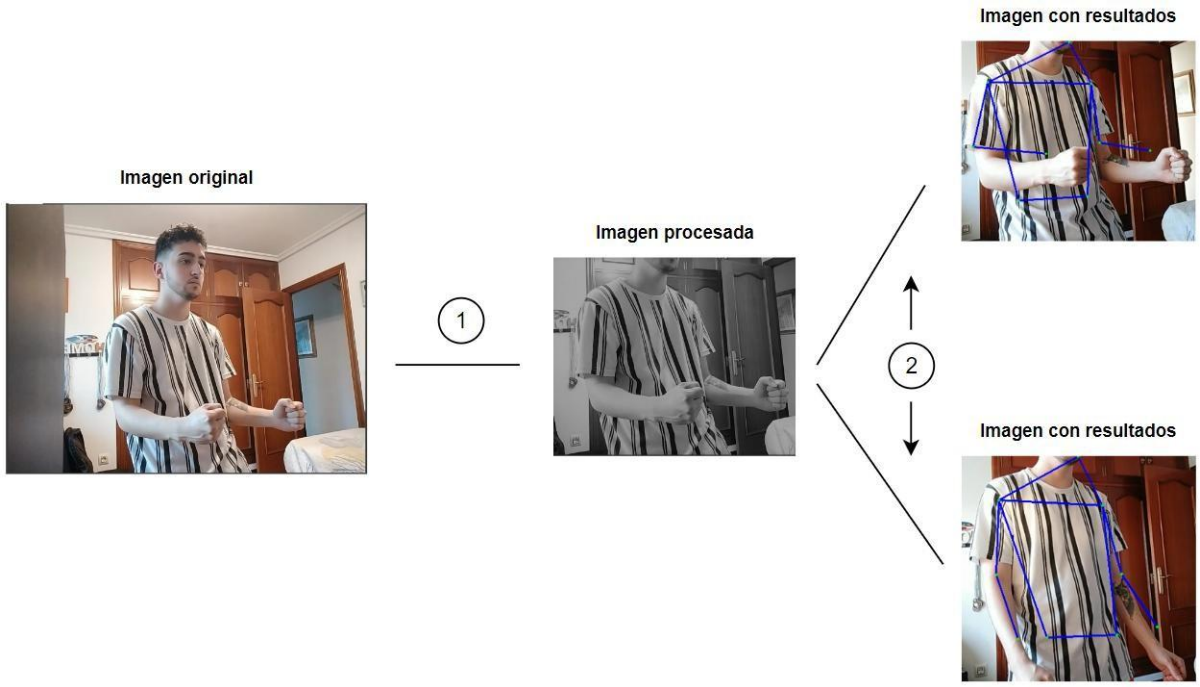

Ilustración 22: Escenario Pose

En la **ilustración 21**, se muestra el segundo escenario. Este sigue el mismo procedimiento que en el anterior caso. Comienza con un frame original obtenido por la cámara de vídeo. Siguiendo la arquitectura del diagrama, este frame es procesado (1), como se puede observar en la siguiente imagen, para que los algoritmos de detección del cuerpo correspondientes la puedan utilizar como entrada.

Una vez procesada, se redirige a estos algoritmos (2). En este caso, para extraer los puntos clave, se utiliza únicamente el modelo MoveNet para obtener las detecciones. Esto es posible observarlo en las dos últimas imágenes del diagrama, en las cuales, se muestran los resultados. Se puede ver como en la imagen de arriba, hay una persona con los brazos elevados simulando

que está sujetando el volante del vehículo, mientras que en la otra posiciona los brazos cara abajo.

En este tipo de escenarios, se analiza la posición de los antebrazos, utilizando el ángulo que conforma esa unión con el brazo. Si el ángulo formado en el punto clave entre ambos lados es >95º y <180º, entonces se puede clasificar como posible caso de somnolencia. Sin embargo, si el usuario conforma un ángulo <= 90º, entonces el sistema no alerta al usuario.

En resumen, la siguiente imagen muestra un escenario final donde se unifican ambas casuísticas:

Ilustración 23: Prueba de somnolencia

También, existen otros posibles escenarios que pueden alterar el funcionamiento del sistema.

A continuación, se muestra una tabla con los datos recogidos durante las pruebas realizadas en estos escenarios:

	Detección facial	**Detección corporal**
Distancia (metros)		
1-3 m	Accuracy = 1 Precision = 1 Recall = 1 F Score = 1	Accuracy = 0.8 Precision = 0.83 Recall = 0.83 F Score = 0.83
3-5 m	Accuracy = 0.7 Precision = 0.67 Recall = 0.8 F Score = 0.73	Accuracy = 0.6 Precision = 0.6 Recall = 0.6 F Score = 0.6
Perspectiva		
Paralela	Accuracy = 1 Precision = 1 Recall = 1 F Score = 1	Accuracy = 0.8 Precision = 0.83 Recall = 0.83 F Score = 0.83
Oblícua	Accuracy = 0.8 Precision = 0.83 Recall = 0.83 F Score = 0.83	Accuracy = 0.5 Precision = 0.5 Recall = 0.6 F Score = 0.54
Luminosidad (lúmenes)		
450-520 lm	Accuracy = 1 Precision = 1 Recall = 1 F Score = 1	Accuracy = 0.8 Precision = 0.83 Recall = 0.83 F Score = 0.83
300-360 lm	Accuracy = 1 Precision = 1 Recall = 1 F Score = 1	Accuracy = 0.8 Precision = 0.83 Recall = 0.83 F Score = 0.83

180-250 lm	Accuracy = 0.7 Precision = 0.67 Recall = 0.8 F Score = 0.73	Accuracy = 0.6 Precision = 0.6 Recall = 0.6 F Score = 0.6
50-0 lm	Accuracy = 0 Precision = 0 Recall = 0 F Score = 0	Accuracy = 0 Precision = 0 Recall = 0 F Score = 0

Tabla 3: Otros escenarios

La tabla anterior, muestra los valores obtenidos a la hora de evaluar el sistema en los escenarios anteriores.

Para ello, se ha realizado un estudio del número de detecciones correctas por cada 10 detecciones. Estos escenarios abarcan desde distancias variables (de 0 a 5 metros divididos en 2 tramos equidistantes), pasando por diferentes perspectivas del objetivo (de forma frontal y formando 45º en relación a la cámara de vídeo) utilizando cantidades de luz variables en el espacio de pruebas.

En este caso, se ha utilizado como dispositivo de grabación de vídeo un Smartphone Xiaomi Redmi Note 7, con cámara de 48 Megapíxeles.

6. Planificación e seguimento

6.1. Planificación temporal

A continuación se mostrará la planificación temporal del proyecto, calculando los tiempos que hayan sido necesarios para la realización de cada tarea en semanas. Esta se comparará con la planificación real que se llevó a cabo, estudiando posibles variaciones en el tiempo para cada tarea.

Se estima que este proyecto tendrá una duración de 300 horas, distribuidas en 12 semanas. De esta forma se le dedicarán 25 horas a la semana, divididas en ~ 3.5 horas diarias de lunes a domingo.

A continuación se detallan las tareas a realizar y la duración estimada en semanas.

Fases y tareas	**Estimación (horas)**	**Fecha inicio**	**Fecha fin**
Análisis	30	01/02/2022	10/02/22
Análisis inicial	10	01/02/2022	03/02/2022
Búsqueda de algoritmos y modelos	5	03/02/2022	05/02/2022
Análisis de algoritmos y modelos	15	05/02/2022	10/02/2022
Elaboración	30	10/02/2022	19/02/2022
Estudio de implementación de los modelos	20	10/02/2022	16/02/2022
Diseño	10	16/02/2022	19/02/2022
Implementación	120	19/02/2022	25/03/2022
Implementación de los algoritmos	70	19/02/2022	10/03/2022
Automatización del código en Colab	30	10/03/2022	19/3/2022
Pruebas de desarrollo	20	19/3/2022	25/03/2022
Estudio de resultados	30	25/03/2022	02/04/2022
Obtención de resultados	20	25/03/2022	31/03/2022
Estudio de resultados	10	31/03/2022	02/04/2022

Documentación	90	02/04/2022	01/05/2022
Documentar estado del arte	30	02/04/2022	12/04/2022
Documentar estudio e implementación	40	12/04/2022	24/04/2022
Documentar resultados obtenidos	15	24/04/2022	29/04/2022
Retoques finales de documentación	5	29/04/2022	01/05/2022

Tabla 4: Planificación temporal

A continuación se muestra el diagrama de Gantt correspondiente a la planificación detallada anteriormente.

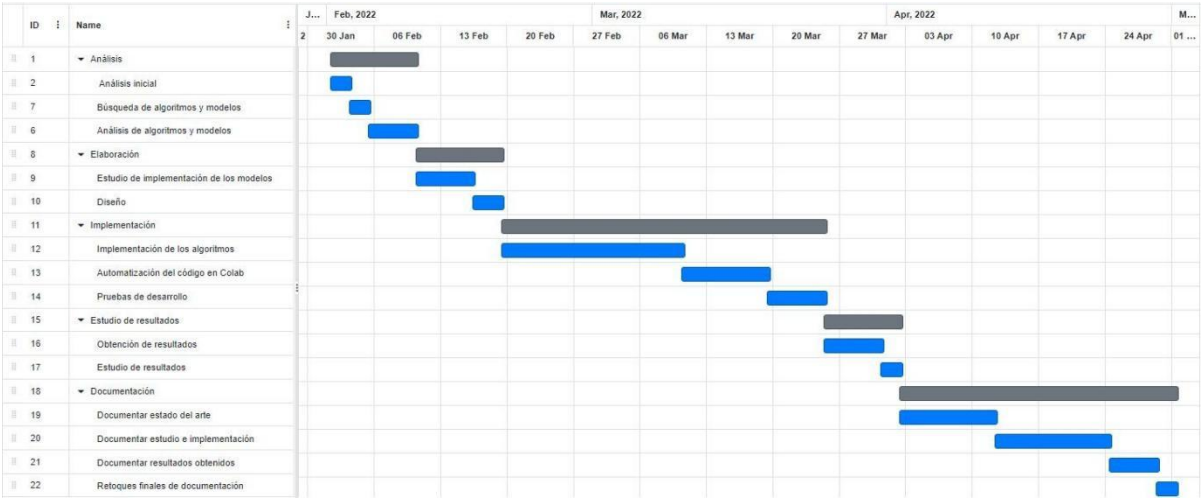

Ilustración 24: Planificación inicial

6.1.1. Planificación real

Como se puede observar en el siguiente diagrama, no ha habido cambios en la duración del proyecto respecto a la planificación original, pero sí que se ven cambios en la duración de algunas tareas.

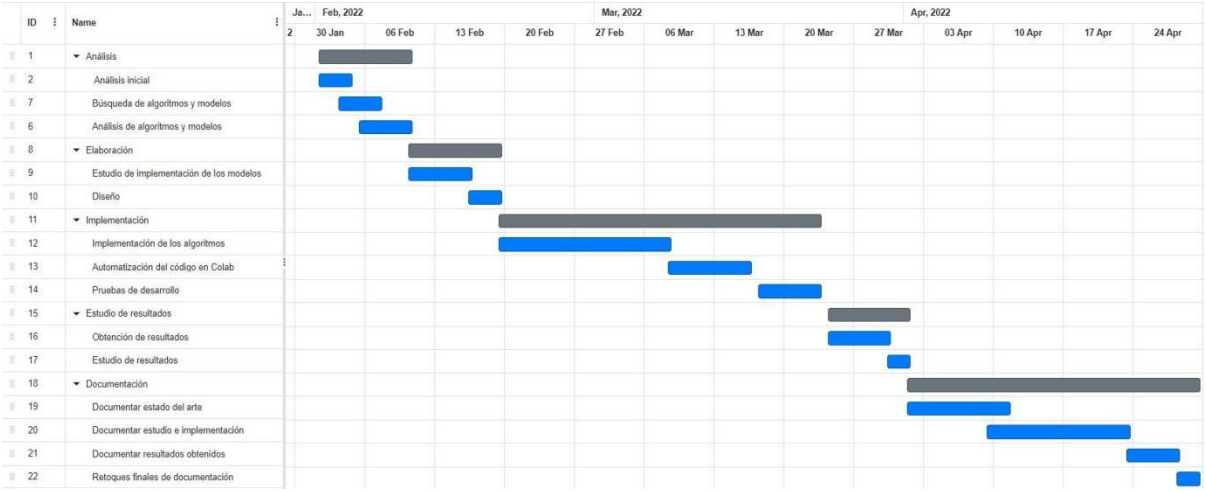

Ilustración 25: Planificación real

7. Principales aportaciones

Este proyecto se centró en la realización de un objetivo principal marcado antes del inicio del mismo y el posterior análisis de los resultados obtenidos.

Este objetivo es desarrollar un sistema que permita detectar síntomas de somnolencia con el fin de solventar de forma notable las catástrofes provocadas en la carretera, aplicando diversas técnicas de visión artificial, donde se obtienen los resultados que posteriormente se analizan.

Las aportaciones realizadas teniendo en cuenta este objetivo principal:

- Se logra la detección de síntomas de somnolencia obteniendo de esta forma un sistema funcional que cumple los objetivos fijados.

Teniendo en cuenta los objetivos más concretos que derivan del principal se han hecho las siguientes aportaciones:

- Utilizando este sistema, se obtiene una comparativa de resultados de varios modelos y algoritmos de visión artificial a través de múltiples pruebas incluyendo la representación gráfica de los datos.

- Se ha estudiado el comportamiento de los diferentes algoritmos integrados dentro del mismo sistema durante su funcionamiento simultáneo.

- Se ha conseguido implementar y desarrollar el sistema objetivo de forma totalmente flexible y práctica tanto para el usuario como para el desarrollador

8. Conclusiones

Gracias a los resultados obtenidos tras la ejecución de los distintos modelos y algoritmos integrados en el mismo sistema, y observando los resultados obtenidos de la comparativa en base a la extracción de métricas, es posible concluir que se trata de una entre tantas vías posibles de desarrollo para lograr el objetivo principal de creación de un sistema de ayuda contra las posibles catástrofes causadas por, no solamente somnolencia, sino también otras posibles distracciones.

Al ser un proyecto de este calibre, y pertenecer a un área tan extensa que se encuentra actualmente en su auge, además de que a medida que pasan los años su progreso es exponencial, esto permite desarrollar aplicaciones muy útiles en la vida diaria de cualquier persona en todo tipo de campos. Estos resultados concuerdan con lo que se esperaba obtener, apoyando por lo tanto la teoría estudiada a lo largo de todo el proyecto.

Desde un punto de vista personal ha resultado especialmente interesante participar en un proyecto del área de Inteligencia Artificial, ya que es el área que más me atrae, pero no había tenido apenas la oportunidad de implicarme en un proyecto de estas características y seguir aumentando mis conocimientos sobre este campo.

Profundizando un poco más, no tan solo la IA seguirá en pleno crecimiento, sino también todas las subáreas de la misma, en concreto, en este caso, el área de visión artificial y tratamiento de imágenes, el cual tiene una cantidad gigante de aplicaciones en la vida cotidiana, ya sea a nivel de salud, seguridad, filtrados y reconocimiento facial, e incluso en la realidad virtual y aumentada. Además, actualmente su progreso se centra sobre todo en la mejora del rendimiento y eficiencia de los algoritmos y modelos actuales, aumentando su precisión a la par que la velocidad de procesamiento de las imágenes en tiempo real, permitiendo que cada vez sean más viables en el entorno del día a día.

La realización de esta documentación me ha ayudado a comprender cómo es realizar un proyecto de investigación desde cero y me permitió poner en práctica muchas de las cosas aprendidas durante los cuatro años de carrera.

En general el resultado del proyecto es muy satisfactorio. Existen muchas vías por las que seguir y mejorar los resultados obtenidos, además de poder seguir investigando y continuar con un futuro desarrollo del sistema de forma más eficiente y eficaz.

9. Vías de trabajo futuro

Teniendo en cuenta el éxito obtenido en los experimentos realizados, es posible continuar la implementación del sistema por diferentes caminos, mejorando las técnicas utilizadas en el proyecto actual. Algunas de las mejoras y avances que se pueden realizar son:

- Introducir nuevos objetivos en el proyecto, partiendo del código actual como base, ya que cualquiera de los modelos y algoritmos utilizados tienen múltiples usos dentro del área.

- Utilizar nuevos modelos de redes neuronales, algoritmos, frameworks e incluso modificar sus propias arquitecturas.

- Mejorar la eficiencia y velocidad de procesamiento de los frames utilizando modelos de redes neuronales más rápidos y con un entrenamiento más preciso.

- Probar diferentes plataformas para una posible exportación del proyecto a otro entorno.

10. Referencias

[1] Brian C. Tefft AAA Foundation for Traffic Safety. Prevalence of Motor Vehicle Crashes Involving Drowsy Drivers, United States, 2009 – 2013 (November 2014) .

[2] The National Highway Traffic Safety Administration. Drowsy Driving: Avoid falling Asleep Behind The Wheel | NHTSA . [Consulta: 17 de Febrero de 2022]

[3] Varun Bansal (Apr 5, 2020).The Evolution of Deep Learning. [Consulta: 10 de Marzo de 2022]

[4] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[6] Zhengxia Zou, Zhenwei Shi, Member, IEEE, Yuhong Guo, and Jieping Ye, Senior Member, IEEE. Object Detection in 20 Years: A Survey.

[7] Wiley, Victor and Thomas Lucas. “Computer Vision and Image Processing: A Paper Review.” International Journal of Artificial Intelligence Research (2018): n. pag.

[8] Yang, Ming-Hsuan et al. “Detecting Faces in Images: A Survey.” IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002): 34-58.

[9] Zhang, Shibo and Xiaojie Wang. “Human detection and object tracking based on Histograms of Oriented Gradients.” 2013 Ninth International Conference on Natural Computation (ICNC) (2013): 1349-1353.

[10] Chauhan, Rahul et al. “Convolutional Neural Network (CNN) for Image Detection and Recognition.” 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC) (2018): 278-282.

[11] Zhang, Mi et al. “Deep Learning in the Era of Edge Computing: Challenges and Opportunities.” ArXiv abs/2010.08861 (2020): n. pag.

[12] Soviany, Petru and Radu Tudor Ionescu. “Frustratingly Easy Trade-off Optimization Between Single-Stage and Two-Stage Deep Object Detectors.” ECCV Workshops (2018).

[13] Grace Karimi. “Introduction to YOLO Algorithm for Object Detection” on April 15, 2021. [Consulta: 17 de Mayo de 2022]

[14] Jacob Solawetz. “Occlusion Techniques in Computer Vision” [Consulta: 28 de Junio de 2022]

[15] Multiscale Object Detection. [Consulta: 28 de Junio de 2022]

[16] Get Started with Cascade Object Detector [Consulta: 28 de Junio de 2022]

[17] Adrian Rosebrock. “Intersection over Union (IoU) for object detection” on November 7, 2016. [Consulta: 20 de Abril de 2022]

[18] H. -D. Jang, S. Woo, P. Benz, J. Park and I. S. Kweon, "Propose-and-Attend Single Shot Detector," 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 804-813, doi: 10.1109/WACV45572.2020.9093364.

[19] COCO Common Objects in Context [Consulta: 26 de Junio de 2022]

[20] ArcGIS Developer “How single-shot detector (SSD) works” [Consulta: 9 de Abril de 2022]

[21] L. Zhang, H. Wang and Z. Chen, "A Multi-task Cascaded Algorithm with Optimized Convolution Neural Network for Face Detection," 2021 Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), 2021, pp. 242-245, doi: 10.1109/ACCTCS52002.2021.00054.

[22] Long-Hua Ma, Hang-Yu Fan, Zhe-Ming Lu, Dong Tian, “Acceleration of multi-task cascaded convolutional networks” on July 24, 2020. [Consulta: 12 de Marzo de 2022]

[23] MobileNetV2: Inverted Residuals and Linear Bottlenecks [Consulta: 26 de Junio de 2022]

[24] Ronny Votel, Na Li (Tensorflow) “Next-Generation Pose Detection With Movenet” on May 17, 2021. [Consulta: 5 de Mayo de 2022]

[25]Using Histogram of Oriented Gradients (HOG) for Object Detection [Consulta: 27 de Junio de 2022]

[26] Singh, Preeti and Mukesh Kumar Tripathi. “HAAR CASCADE CLASSIFIER PROVIDES HIGH ACCURACY EVEN THE IMAGES ARE HIGHLY AFFECTED BY THE ILLUMINATION.” (2013).

[27] WIDER FACE: A Face Detection Benchmark [Consulta: 26 de Junio de 2022]

[28] Stastny, Jiri & Minařík, Martin. (2007). A Brief Introduction to Image Pre- Processing for Object Recognition.

[29] Darknet: Open Source Neural Netrowks in C [Consulta: 22 de Mayo de 2022]

[30] A. D. Sergeeva and V. A. Sablina, "Eye Landmarks Detection Technology for Facial Micro-Expressions Analysis," 2020 9th Mediterranean Conference on Embedded Computing (MECO), 2020, pp. 1-4, doi: 10.1109/MECO49872.2020.9134338.

[31] Park, Seonwook et al. “Learning to find eye region landmarks for remote gaze estimation in unconstrained settings.” Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (2018): n. pag.

[32] Adrian Rosebrock. “Drowsiness detection with OpenCV” on May 8, 2017. [Consulta: 1 de Febrero de 2022]

[33] Dong. “PASCAL VOC (PASCAL Visual Object Classes Challenge) [Consulta: 1 de Marzo de 2022]

[34] Griffin. Caltech-256 [Consulta: 1 de Marzo de 2022]

[35] OpenCV (Open Source Computer Vision) [Consulta: 16 de Abril de 2022]

[36] TensorFlow Core Api Documentation [Consulta: 19 de Mayo de 2022]