UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de una aplicación de monitorización facial utilizando aprendizaje profundo

Trabajo fin de grado

Grado en Ingeniería Informática

Autor: Blom-Dahl Oliver, Álvaro
Tutor: Agustí Melchor, Manuel

Curso 2020-2021

Resum

Hui en dia, els sistemes de detecció i reconeixement facial formen part del quotidià de moltes persones. Ordinadors, tablets i mòbils tenen integrat, majoritàriament, un sistema d'aquestes característiques. No obstant, fins a quin punt és possible estendre la quantitat d'informació obtinguda del rostre humà?

Aquest projecte proposa la creació d'un sistema de monitorització facial, el qual ha de ser capaç d'estimar diverses mesures, tals com emocions, identificació, posició /postura, expressions facials o direcció de la mirada, a temps real mitjançant la utilització de tècniques de *Deep Learning*, *Machine Learning*, i *Computer Vision*. Amb tal propòsit, es realitza un estudi de les distintes tècniques, eines i algoritmes aptes per desenvolupar tal funció. Després d'identificar les eines que poden resultar útils per el desenvolupament del sistema, es defineixen detalladament els requisits que aquest ha de verificar, així com el disseny i arquitectura d'aquest.

Posteriorment, s'exposarà el desenvolupament del sistema explicant amb precisió el seu funcionament intern i les particularitats de la solució final, realitzant després proves d'avaluació de robustesa i rendiment amb la finalitat d'ampliar el coneixement sobre el camp sotmès a estudi.

Finalment, una vegada realitzats el anàlisis dels resultats obtinguts, s'expliquen els problemes identificats i es proposen potencials solucions per al desenvolupament de futurs treballs relacionats amb la matèria que ens ocupa.

Paraules clau: *Machine Learning*, *Deep Learning*, *Computer Vision*, OpenCV , Dlib, *Kalman filter*, *Image processing*, Detecció facial, Reconeixement facial, Monitorització facial

Resumen

Actualmente, los sistemas de detección y reconocimiento facial son algo del día a día de muchas personas. Ordenadores, *tablets* y móviles tienen, en su mayoría, integrado un sistema de éstas características. Sin embargo, ¿Hasta dónde podemos extender la cantidad de información extraída del rostro humano?

En este proyecto se propone la creación de un sistema de monitorización facial, el cual debe ser capaz estimar diversas mediciones, tales como emociones, identificación, pose, gestos faciales o la dirección de la mirada, a tiempo real mediante el uso de técnicas de *Deep Learning*, *Mahine Learning* y *Computer Vision*. Para ello se hará un estudio de las diversas técnicas, herramientas y algoritmos disponibles útiles para dicha tarea. Tras identificar las herramientas que pueden resultar de ayuda para el desarrollo del sistema se definirán de forma detallada los requisitos que éste debe cumplir y el diseño y arquitectura de éste.

Posteriormente se expondrá el desarrollo del sistema explicando de forma precisa el funcionamiento interno de éste y las peculiaridades de la solución final, efectuando después pruebas de evaluación de robustez y rendimiento con el fin de extender el conocimiento sobre el campo sometido a estudio.

Finalmente, tras realizar el análisis de los resultados obtenidos, se explicarán los problemas encontrados y se aportarán posibles soluciones a éstos para el desarrollo de futuros trabajos relacionados con esta tarea.

Palabras clave: Aprendizaje automático, Aprendizaje profundo, Visión por Computador, OpenCV , Dlib, Filtro de Kalman, Procesamiento de imágenes, Detección facial, Reconocimiento facial, Monitorización facial

Abstract

Facial detection and recognition systems are something familiar for people day by day. Computers, tablets and mobiles have (for the most part) this kind of system integrated. However, how far can we go in terms of the amount of information extracted from the human face?

In this Project, the creation of a facial monitoring system is proposed, which must be able to estimate various measurements, such as emotions, identification, pose, facial gestures or the direction of the gaze in real time using techniques of Deep Learning, Machine Learning and Computer Vision. To do that, a study will be made about various techniques, tools and algorithms available useful for this task. After identifying the tools that can be helpful for the development of the system, the requirement, the design and the architecture will be defined in detail.

Subsequently, the development of the system will be exposed, precisely explaining its internal functioning and the peculiarities of the final solution, then carrying out robustness and performance evaluation tests to extend the knowledge about the topic under study.

Finally, after analysing the results obtained, the problems encountered will be explained and possible solutions to these will be provided for the development of future work related to this task.

Key words: Machine Learning, Deep Learning ,Computer Vision, OpenCV , Dlib,Kalman filter, Image processing, Facial detection, Facial recognition, Facil monitoring

Índice general

Índice general VII
Índice de figuras XI
Índice de tablas XIII

1 Introducción 1
1.1 Motivación . . . 1
1.2 Objetivos . . . 2
1.3 Metodología . . . 2
1.4 Estructura de la memoria . . . 3
2 Estado del arte y fundamentos teóricos 5
2.1 Extracción de características . . . 5
2.1.1 Filtros *Haar* . . . 5
2.1.2 HOG (*Histogram of Oriented Gradients*) . . . 6
2.1.3 LBP (Local Binary Patterns) . . . 6
2.2 Detección facial . . . 7
2.2.1 Clasificador en cascada . . . 7
2.2.2 HOG + SVM . . . 8
2.2.3 CNN . . . 9
2.2.4 MTCNN . . . 11
2.3 Identificación facial . . . 13
2.4 Extración de puntos de referencia faciales . . . 13
2.4.1 Extracción de *landmarks* mediante árboles de regresión . . . 14
3 Análisis del problema 17
3.1 Requisitos . . . 17
3.1.1 Requisitos funcionales . . . 18
3.1.2 Requisitos no funcionales . . . 23
3.2 Identificación y análisis de soluciones posibles . . . 24
3.2.1 Detección Facial . . . 24
3.2.2 Extracción de puntos de referencia faciales . . . 27
3.2.3 Identificación facial . . . 27
3.2.4 Estimación de edad, género y emociones mediante CNN's . . . 27
3.3 Solución propuesta . . . 28
3.3.1 Extracción y estimación de las mediciones . . . 28
3.3.2 Interdependencias entre las mediciones . . . 30
3.3.3 Filtrado y predicción de las mediciones . . . 30
3.3.4 Carga, guardado y filtrado de las mediciones . . . 32
4 Diseño de la solución 33
4.1 Arquitectura del sistema . . . 33
4.2 Diseño detallado . . . 36
4.2.1 `Camera` . . . 36
4.2.2 `Camera Calibration` . . . 36
4.2.3 `Face Detector` . . . 37

4.2.4 stabilizer . . . 37
4.2.5 Landmarks predictor . . . 37
4.2.6 Landmarks stabilizer . . . 38
4.2.7 Pose Estimator . . . 38
4.2.8 Pose Stabilizer . . . 38
4.2.9 Gaze Estimator . . . 39
4.2.10 Gaze Stabilizer . . . 39
4.2.11 Action Unit Detector . . . 40
4.2.12 Aciton Unit Stabilizers . . . 40
4.2.13 Emotion Detector AU . . . 41
4.2.14 Emotion Detector CNN . . . 41
4.2.15 Age Gender Detector . . . 41
4.2.16 Face Recognition . . . 42
4.2.17 Recorder . . . 42
4.2.18 Face Monitoring . . . 43

5 Desarrollo de la solución propuesta **45**
5.1 Estabilización y predicción de las mediciones . . . 46
5.2 Extracción de *frames* . . . 47
5.3 Detección facial . . . 47
5.3.1 Optimización en la detección . . . 48
5.3.2 Extracción de la región de interés . . . 49
5.4 Extracción de *landmarks* . . . 49
5.4.1 Estabilización y predicción de los *landmarks* . . . 50
5.5 Estimación de la pose de la cabeza . . . 51
5.5.1 Estabilización y predicción de la pose . . . 53
5.5.2 Visualización de la pose . . . 54
5.6 Estimación de la dirección de la mirada . . . 54
5.6.1 Estabilización y predicción de la dirección de la mirada . . . 57
5.6.2 Visualización de la dirección de la mirada . . . 57
5.7 Estimación de gestos faciales . . . 58
5.7.1 Apertura de la boca . . . 59
5.7.2 Separación de las comisuras de la boca (sonrisa) . . . 60
5.7.3 Acercamiento de las comisuras de la boca (beso) . . . 61
5.7.4 Pestañeo . . . 62
5.7.5 Movimiento vertical de las cejas . . . 63
5.8 Detección de emociones mediante los gestos faciales estimados . . . 64
5.9 Detección de emociones mediante CNN's . . . 65
5.10 Estimación de edad y género mediante CNN's . . . 66
5.11 Identificación facial mediante CNN . . . 67
5.12 Guardado, cargado y filtrado de las mediciones . . . 68
5.13 Módulo gestor . . . 69
5.14 Aplicación para la visualización de la monitorización facial . . . 72

6 Pruebas **77**
6.1 Extracción de los *landmarks* . . . 77
6.2 Estimación de la pose de la cabeza . . . 79
6.3 Estimación de la dirección de la mirada . . . 79
6.4 Estimación de los gestos faciales . . . 80
6.5 Estimación de emociones mediante los gestos faciales . . . 81
6.6 Estimación de emociones mediante CNN's . . . 82
6.7 Estimación del género y la edad . . . 83
6.8 Identificación facial . . . 85
6.9 Pruebas de rendimiento . . . 87

7 Casos de uso del sistema **89**
7.1 Pasa páginas automático . . . 89
7.2 Monitorización del sueño en la conducción . . . 91
7.3 Separación de conjuntos de imágenes por identificación facial . . . 92
7.4 Otros casos de uso . . . 93
7.4.1 Porcentaje de satisfacción . . . 93
7.4.2 Aplicaciones para personas con discapacidad funcional . . . 94
8 Conclusiones y trabajos futuros **95**
8.1 Relación del trabajo desarrollado con los estudios cursados . . . 97

Apéndices

A Especificaciones de la máquina utilizada **103**
B Procesamiento de imágenes **105**
B.0.1 *Thresholding* . . . 105
B.0.2 Erosión . . . 105
B.0.3 Dilatación . . . 105
B.0.4 Desenfoque por filtro mediano . . . 105
B.0.5 Detección de contornos . . . 106
B.0.6 *Blob From Image* . . . 106

Índice de figuras

2.1 Representación gráfica de los filtros *Haar* [8]. 6
2.2 Representación gráfica de la extracción de características mediante HOG. 6
2.3 Represencación grádfica de la técnica de extracción de características LBP [43]. 7
2.4 Representación gráfica de algoritmo Viola-Jones [18]. 8
2.5 Representación gráfica del *Kernel* o núcleo. [49] 9
2.6 Representación gráfica del algoritmo de detección facial HOG+SVM. . . . 9
2.7 Representación gráfica del proceso de convolución. [11] 10
2.8 Representación gráfica del proceso de *Pooling*. [11] 11
2.9 Representación gráfica de dos capas ocultas totalmente conectadas. [11] . 11
2.10 Representación gráfica de la red MTCNN. [22] 12
2.11 Representación gráfica de la codificación facial. [30] 13
2.12 Representación gráfica de los puntos de referencia faciales [42] 14

3.1 Proyecto de experimentación en `Google Colab`. Elaboración propia. 25
3.2 %Detectado y FPS para el vídeo de entrada `control`. 26
3.3 %Detectado y FPS para el vídeo de entrada `iluminación`. 26
3.4 %Detectado y FPS para el vídeo de entrada `rotación`. 26
3.5 %Detectado y FPS para el vídeo de entrada `oclusión`. 27
3.6 Grafo de interdependencia entre las posibles mediciones. 30
3.7 Representación gráfica del funcionamiento del filtro de kalman [47]. . . . 31

4.1 Diagrama de módulos del sistema de monitorización facial. 35

5.1 Diagrama de procesos del sistema. 46
5.2 Visualización de la detección facial. 48
5.3 FPS medio de procesamiento mediante optimizaciones. 49
5.4 Visualización de la extracción de la región de interés. 49
5.5 Extracción de los 68 *landmarks*. 50
5.6 Predicción del subconjunto de *landmarks* utilizados para la estimación de la pose. 50
5.7 *landmarks* propuestos en [23] (puntos rojos) para la estimación de la pose. 51
5.8 Estimación de pose mediante los *landmarks* propuestos en [23] (puntos rojos). 52
5.9 Análisis de la variabilidad de los 68 *landmarks* ante gestos faciales (ordenados de mayor a menor). 52
5.10 Representación visual de la variabilidad de los 68 *landmarks* ante gestos faciales). 53
5.11 Estimación de pose mediante los *landmarks* propuestos en este proyecto. . 53
5.12 Predicción de la pose de la cabeza. 54
5.13 Visualización de la pose de la cabeza. 54
5.14 Alineamiento del rostro y sus *landmarks*. 55
5.15 Extracción de las regiones de interés de los ojos. 55
5.16 Procesamiento de las regiones de interés. 56
5.17 Detección de contornos y extracción del centro de masa. 56

5.18 Visualización de la dirección de la mirada. 57
5.19 Estimación del gesto `mor`. [27] 59
5.20 Estimación y filtrado del gesto `mor`. [42] 59
5.21 Estimación del gesto `msr`. [42] 60
5.22 Estimación y filtrado del gesto `msr`. 60
5.23 Estimación del gesto `mkr`. [42] 61
5.24 Estimación y filtrado del gesto `mkr`. 61
5.25 Estimación del gesto `ear`. [42] 62
5.26 Estimación y filtrado del gesto `ear`. [42] 62
5.27 Estimación del gesto `ebr`. [42] 63
5.28 Estimación y filtrado del gesto `ebr`. 63
5.29 Representación visual de la estimación de emociones mediante los gestos faciales. 64
5.30 Proceso de detección de emociones mediante CNN's. 65
5.31 Proceso de detección de edad y género mediante CNN's. 66
5.32 Representación gráfica de la distancia entre codificaciones faciales. 67
5.33 Representación gráfica del proceso de identificación facial mediante la base de datos de codificaciones faciales. 68
5.34 Proceso de búsqueda según las mediciones monitorizadas. 69
5.35 Representación gráfica de la función `analyze`. 70
5.36 Representación gráfica de la función `analyze` ante fallos de detección. . . . 71
5.37 Programa en `Python` para la detección de la pose y emociones. 71
5.38 Captura de la ventana inicial de `realtime`. 72
5.39 Captura del cuadro *Camera* de `realtime`. 72
5.40 Captura del cuadro *Action Units* de `realtime`. 73
5.41 Captura del cuadro *Action Unit Based Emotion* de `realtime`. 73
5.42 Captura del cuadro *Pose* de `realtime`. 73
5.43 Captura del cuadro *Gaze* de `realtime`. 74
5.44 Captura del cuadro *CNN Based* de `realtime`. 74
5.45 Captura del cuadro *CNN Based* de `realtime`. 74
5.46 Captura del cuadro *CNN Based* de `realtime`. 75
5.47 Visualización completa de la interfaz `realtime`. 76

6.1 Extracción de *landmarks* mediante el algoritmo de detección HOG+SVM. . 78
6.2 Extracción de *landmarks* mediante el algoritmo de detección CNN (`cvlib`). 78
6.3 Estimación de la pose de la cabeza para distintos ángulos y gestos faciales. 79
6.4 Estimación de la dirección de la mirada para distintos ángulos. 79
6.5 Pruebas de estimación de gestos faciales. 80
6.6 Pruebas de estimación de emociones mediante gestos faciales. 81
6.7 Pruebas de estimación de emociones mediante CNN. 82
6.8 Distribución de la edad en el *dataset* AAF. 83
6.9 Distribución del error asociado a la estimación de la edad. 83
6.10 Matriz de confusión asociada a la clasificación de edad evaluada en el *dataset* AAF. 84
6.11 Matriz de distancias entre las codificaciones faciales. 85
6.12 Matriz de distancias entre las codificaciones faciales filtrada por la condición `valor <= 6`. 86
6.13 Matriz de distancias entre las codificaciones faciales (2003-2021). 86
6.14 Uso de CPU y memoria realizando el análisis de todas las mediciones. . . 88
6.15 Uso de procesadores lógicos realizando el análisis de todas las mediciones. 88

7.1 Representación visual de la variable `state` en función de la inclinación de la cabeza. 90

7.2 Código de la aplicación `pasa páginas`. 90
7.3 Código de la aplicación de monitorización del sueño en la conducción. . . 91
7.4 Visualización del programa definido en la figura 7.3 92
7.5 Código de la aplicación de separación de conjuntos de imágenes por identificación facial. 93

Índice de tablas

4.1 Módulos del sistema y sus respectivas funcionalidades. 33
4.2 Atributos y funciones del módulo `Camera`. 36
4.3 Atributos y funciones del módulo `Camera Calibration`. 36
4.4 Atributos y funciones del módulo `Face Detector`. 37
4.5 Atributos y funciones del módulo `stabilizer`. 37
4.6 Atributos y funciones del módulo `Landmarks predictor`. 37
4.7 Atributos y funciones del módulo `Landmarks stabilizer`. 38
4.8 Atributos y funciones del módulo `Pose Estimator`. 38
4.9 Atributos y funciones del módulo `Pose Stabilizer`. 38
4.10 Atributos y funciones del módulo `Gaze Estimator`. 39
4.11 Atributos y funciones del módulo `Gaze Stabilizer`. 39
4.12 Atributos y funciones del módulo `Action Unit Detector`. 40
4.13 Atributos y funciones del módulo `Action Unit Stabilizers`. 40
4.14 Funciones del módulo `Emotion Detector AU`. 41
4.15 Atributos y funciones del módulo `Emotion Detector CNN`. 41
4.16 Atributos y funciones del módulo `Age Gender Detector`. 41
4.17 Atributos y funciones del módulo `Face Recognition`. 42
4.18 Atributos y funciones del módulo `Recorder`. 42
4.19 Atributos y funciones del módulo `Face Monitoring`. 43

5.1 Número de estabilizadores para cada medición cuantitativa. 46
5.2 Modos de detección facial. 47
5.3 Gestos faciales detectados por el sistema. 58
5.4 Rangos establecidos para la normalización de los gestos estimados. 58
5.5 Tabla de condiciones para la detección de expresiones mediante los gestos faciales estimados. 64
5.6 Atributos guardados en la lista `record` del módulo `Recorder`. 68
5.7 Argumentos de la función `analyze`. 69
5.8 Posibles elementos de la lista argumento `target`. 70

6.1 *Frames* por segundo para distintos valores del parámentro `target`. 87

7.1 Tabla de posibles valores de la variable `state`. 89

A.1 Especificaciones de la máquina utilizada. 103

CAPÍTULO 1
Introducción

El campo de la visión por computador tiene como propósito la creación de sistemas autónomos capaces de imitar tareas propias de la percepción visual humana y, en algunos casos, superarla [25]. Este campo nació con la aportación del considerado padre de la visión por computador, Larry Roberts, quién propuso en 1960 la posibilidad de extraer información tridimensional desde un perspectiva bidimensional, es decir, a partir de una cámara convencional.

Este campo tuvo una rápida evolución, la cual se vió acelerada con la llegada de los avances en Inteligencia Artificial y *Machine Learning*. Dichos avances permitieron la creación de herramientas cada vez más complejas, las cuales muestran un gran potencial en diversos sectores del mercado. Algunas de las aplicaciones de la visión por computador más conocidas son el control de calidad en procesos industriales, los sistemas de conducción autónoma y el identificación de objetos, entre muchas otras.

Una de las aplicaciones de este campo, la cual ha experimentado un significativo a avance en la últimas décadas, es la relacionada con la detección y el identificación del rostro humano. Sin embargo, ¿Hasta qué punto podemos extender la cantidad de información que podemos extraer del rostro humano?

En esta memoria nos centraremos en la tarea de la monitorización facial. Entendiendo monitorización cómo la acción de " observar mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías " [10]. Por lo que podemos definir la monitorización facial como el proceso de detección de parámetros fisiológicos relativos al rostro humano, tales como gestos faciales, emociones, posición, etc.

Por lo tanto, el propósito de este proyecto es el estudio de esta tarea y los fundamentos teóricos detrás de ésta para su posterior análisis con el fin de crear un sistema autónomo capaz de realizar la detección y seguimiento de los parámetros faciales para su uso en múltiples aplicaciones posibles.

1.1 Motivación

Es bien sabido que el campo de la visión por computador y la inteligencia artificial han supuesto un cambio de paradigma para una gran cantidad de sectores del mercado que actualmente utilizan este tipo de técnicas con el fin de mejorar sus productos, innovar e investigar nuevas tecnologías. Resulta evidente que este tipo de herramientas suponen un gran avance y auguran un futuro prometedor. Por ello, la elección de este proyecto viene dada principalmente por la importancia que este campo tiene en el desarrollo de futuro software relacionado con la monitorización facial.

1.2 Objetivos

El principal objetivo de este proyecto es el de crear un sistema de monitorización facial multiplataforma a tiempo real y robusto mediante librerías de código abierto, tales como `OpenCV` y `Dlib`. El sistema deberá ser capaz de monitorizar distintos parámetros faciales, tales como la pose de la cabeza, la dirección de la mirada, emociones y gestos faciales. Además, otro de los objetivos es que el sistema sea capaz de realizar todas estas tareas tanto sobre un vídeo ya realizado como en tiempo real, a través de una cámara o *webcam* convencional.

Una de las principales prioridades de este proyecto es la de crear un sistema a modo de prototipo con el fin de explorar diferentes técnicas y herramientas útiles para extender la cantidad de información que se puede extraer de un rostro como expresiones, emociones, etc. Además el sistema de monitorización debe poder realizar las mediciones a tiempo real con una tasa de FPS aceptable (10-30 FPS).

Por lo tanto, el proyecto se fundamenta sobre la creación de un sistema de monitorización facial construido sobre el lenguaje de programación `Python` para computadoras de escrito, el cual podrá ser utilizado a modo de librería por el usuario experto.

Como objetivo secundario, éste se adaptará a un par de aplicaciones entendidas como casos de uso del sistema, con una interfaz sencilla que podrán ser utilizadas por cualquier usuario sin conocimiento en programación con el fin de mostrar el potencial de esta tecnología para diversos casos de uso prácticos. Para ello, el sistema a implementar recogerá diferentes herramientas, técnicas y algoritmos tanto del estado del arte como de elaboración propia.

1.3 Metodología

Con el fin de cumplir los objetivos propuestos anteriormente la metodología relativa a la elaboración de este proyecto viene dada en cuatro fases diferenciadas:

- **Fase de estudio:** el propósito de esta fase es el estudio del trasfondo teórico relativo a la monitorización facial.
- **Fase de experimentación:** el propósito de esta fase es el estudio de las diferentes técnicas, algoritmos y herramientas disponibles para el correcto funcionamiento del sistema.
- **Fase de implementación del sistema:** en esta fase se produce todo el desarrollo de código necesario para la creación del sistema en cuestión en función de los conocimientos adoptados en la anterior fase.
- **Fase de pruebas:** en esta fase se realizarán diferentes pruebas con el fin de comprobar que el sistema implementado en la anterior funciona correctamente y analizar sus posibles defectos o características a mejorar.
- **Fase de implementación de aplicaciones:** en esta fase se implementarán distintas aplicaciones a partir del sistema efectuado en la anterior fase con el fin de mostrar el potencial de éste ante distintos sectores del mercado.

1.4 Estructura de la memoria

La presente memoria consta de los siguientes cinco capítulos:

- **Capítulo 1. Introducción:** en este primer capítulo se hace una breve introducción a la visión por computador y se muestra el propósito del proyecto, sus objetivos a alcanzar y la motivación por la cual se ha elegido este campo.
- **Capítulo 2. Estado del arte y fundamentos teóricos:** este capítulo sirve de introducción a los fundamentos teóricos necesarios para la comprensión del proyecto además de poner en contexto la situación actual del campo aportando información acerca de los algoritmos, técnicas y herramientas utilizadas actualmente.
- **Capítulo 3. Análisis del problema:** en este capítulo se especificarán los requisitos que el sistema implementado debe cumplir y, tras ello, se realizará un análisis de las posibles soluciones disponibles para el desarrollo de dicho sistema para, finalmente, desarrollar la propuesta final.
- **Capítulo 4. Diseño de la solución:** en este capítulo se definirá la estructura del sistema, sus partes, y qué tipo de arquitectura se propone para su implementación. Tras ello, se documentará dicha arquitectura detallando las peculiaridades de cada una de las partes que la componen.
- **Capítulo 5. Desarrollo de la solución:** en este capítulo se se abordará todo el proceso de desarrollo del sistema, especificando las fases seguidas para su construcción y el funcionamiento interno de éste.
- **Capítulo 6. Pruebas:** este capítulo servirá de guía para definir la robustez y eficiencia del sistema implementado. Para ello, se realizarán numerosas pruebas sobre cada una de las partes que componen al sistema con el fin de destacar los posibles elementos a mejorar y aportar mayor entendimiento sobre el funcionamiento del sistema y sus peculiaridades.
- **Capítulo 7. Aplicaciones demostrativas y casos de uso:** en este capítulo, con el fin de mostrar la versatilidad y facilidad de uso del sistema, se mostrarán una seria de aplicaciones demostrativas a modo de casos de uso en diferentes sectores del mercado.
- **Capítulo 8. Conclusiones:** tras las pruebas realizadas y la información obtenida a lo largo de la memoria, en este capítulo se abordarán las diversas conclusiones obtenidas, los problemas encontrados durante el desarrollo y la evaluación del sistema y se aportarán posibles soluciones a éstos con el fin de proponer dichas soluciones como futuros trabajos.

CAPÍTULO 2
Estado del arte y fundamentos teóricos

En este capítulo se hará una breve introducción sobre la situación actual del campo de la visión por computador en relación con la detección facial y su análisis. Para ello, nos situaremos en el contexto actual del campo adoptando una visión general de los distintos algoritmos, técnicas, herramientas y fundamentos teóricos utilizados actualmente para dicho propósito. En concreto, se abordarán en primer lugar los distintos algoritmos ampliamente utilizados para el identificación facial, entendiendo tal proceso como el análisis de imágenes con el fin de identificar rostros en ella. En segundo lugar, se abordará la tarea de extracción de puntos de referencia faciales, necesarios para su posterior análisis con el fin de ser capaces de monitorizar los rostros detectados.

En primer lugar cabe plantearse cómo los humanos somos capaces de detectar y reconocer rostros. A pesar de que se trata de una compleja cuestión, en primer lugar extraemos características de lo que vemos para, después, analizarlas y así discernir lo que es un rostro de lo que no. Este proceso es en el que se ha inspirado el campo de la visión por computador con el fin de resolver esta tarea.

2.1 Extracción de características

Como es bien sabido, en una imagen podemos encontrar una vasta cantidad de posibles características. Por lo que el análisis de una imagen para, por ejemplo, el identificación de objetos o formas, puede ser altamente costoso computacionalmente. Es por ello por lo que resulta de gran interés disminuir la complejidad de la información contenida en una imagen, aportando exclusivamente las características estrictamente necesarias para realizar el análisis en cuestión. Este proceso es conocido como extracción de características, y supone una importante contribución al campo de la visión por computador [48].

2.1.1. Filtros *Haar*

Los filtros *Haar* son ventantas con dos regiones diferenciadas que se aplican sobre una área de detección de la imagen, con las cuales se computan valores definidos como características *Haar*.

A grandes rasgos, estos valores son calculados mediante la operación de restar la suma de los pixeles relativos a cada una de las regiones de la ventana o filtro *Haar*. Al aplicar dichas ventanas sobre la imagen, estos filtros son capaces de extraer diferentes caracterís-

ticas de ésta. En la figura 2.1 se representa graficamente la aplicación de diversos filtros *Haar* sobre una imagen de entrada.

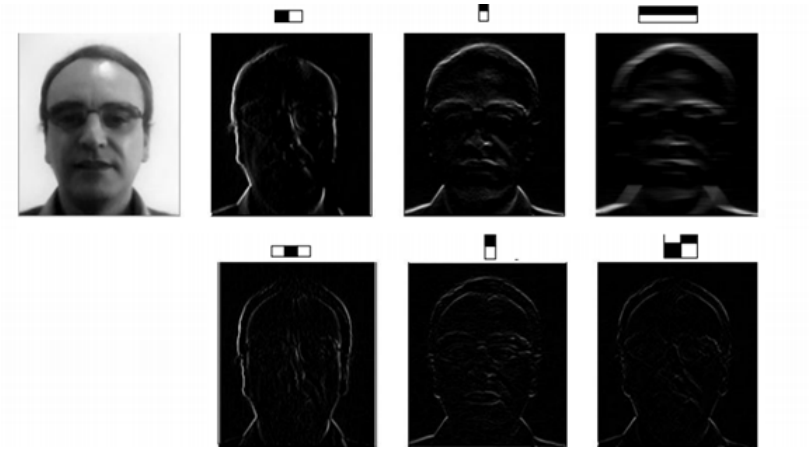

Figura 2.1: Representación gráfica de los filtros *Haar* [8].

2.1.2. HOG (*Histogram of Oriented Gradients*)

El Histograma de Gradientes Orientados es otro de los métodos ampliamente utilizados para la extracción de características de una imagen cuyo objetivo es el de generalizar la información del objeto a detectar de forma que las características extraídas de éste sean robustas bajo diferentes condiciones, por ejemplo, bajo cambios de iluminación.

Para ello, este método se basa en comparar la intensidad de cada píxel con sus respectivos vecinos, calculando así el vector gradiente respectivo a cada zona, es decir, obteniendo el vector que muestra la dirección de la intensidad. Después, se divide la imagen en pequeños cuadros, por ejemplo de 16 x 16, y se calcula el gradiente medio para cada cuadro. De esta forma, el método HOG es capaz de analizar la estructura de un objeto. En la figura 2.2 podemos observar la aplicación del Histograma de Grandientes Orientados sobre una imagen de un rostro.

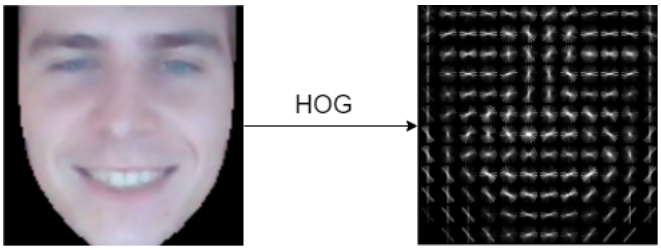

Figura 2.2: Representación gráfica de la extracción de características mediante HOG.

2.1.3. LBP (Local Binary Patterns)

Otra de las técnicas de extracción de características es la conocida como LBP o *Local Binary Patterns* decrita por primera vez en 1990 [32]. Concretamente, esta técnica tiene como objetivo extraer las características relativas a la textura de una imagen de entrada. Para ello, esta técnica sigue los siguiente pasos:

1. Dividir la imagen en celdas (por ejemplo de 16x16 píxeles cada una).

2. Para cada celda, comparar el píxel con sus ocho vecinos siguiendo un orden circular.

3. Cuando el valor del píxel central es mayor al de sus vecinos escribir un "0" y, en caso contrario, un "1". De esta forma se obtiene un número binario de ocho dígitos, el cual se convierte a base decimal.

4. Para cada celda, computar el histograma de frecuencias para cada número extraído en el paso anterior, obteniendo así un vector de 256 dimensiones.

5. Concatenar los histogramas obtenidos en el paso anterior, obteniendo así el vector de características LBP de la imagen de entrada.

En la figura 2.3 podemos observar la aplicación de ésta técnica para una celda de 3x3 píxeles.

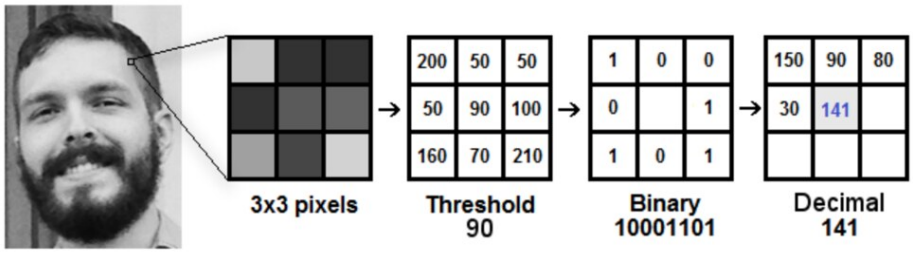

Figura 2.3: Represencación grádfica de la técnica de extracción de características LBP [43].

2.2 Detección facial

Los algoritmos de detección facial se enfocan en la tarea de reconocer rostros a partir de una imagen dada. Desde el punto de vista de la ingeniería informática, esta tarea se reduce a encontrar los cuadros delimitadores asociados a cada uno de los rostros en una imagen y, si procede, extraer la confianza que tiene el sistema de que dicha detección sea realmente un rostro en forma de probabilidades. Actualmente existen diversos algoritmos capaces de realizar esta tarea de forma satisfactoria. Seguidamente se hará un análisis de los algoritmos más utilizados actualmente.

2.2.1. Clasificador en cascada

El clasificador en cascada es un sistema de detección de objetos basado en *Machine Learning* basado en la extracción de características *Haar*, ya mencionadas previamente. Este método fue propuesto por Paul Viola y Michael Jones en 2001 [48]. De hecho, fue el primer algoritmo capaz de detectar caras de forma autónoma en tiempo real.

El sistema es capaz de aprender a detectar rostros mediante aprendizaje supervisado con una gran cantidad de imágenes positivas (que contienen rostros) y negativas (que no contienen rostros). Para ello, inicialmente, se extraen una gran cantidad de características características *Haar* y, para cada una, se encuentran los mejores parámetros que clasifican la imagen.

Sin embargo, la extracción de características supone un coste computacional importante. Aquí es donde entra el clasificador en cascada. En lugar de aplicar todos los filtros de una vez, el sistema agrupa los filtros en grupos y los aplica por separado. De forma que, si un grupo falla, la ventana de detección se desechará.

La librería `OpenCV` contiene un modelo de detección facial basado en esta tecnología disponible para cualquier usuario.[34]

En la figura 2.4 se pueden observar diversos filtros *Haar* para la detección de bordes (*Edge Features*), líneas (*Linc Features*) y centros rodeados (*Center-surround Features*)

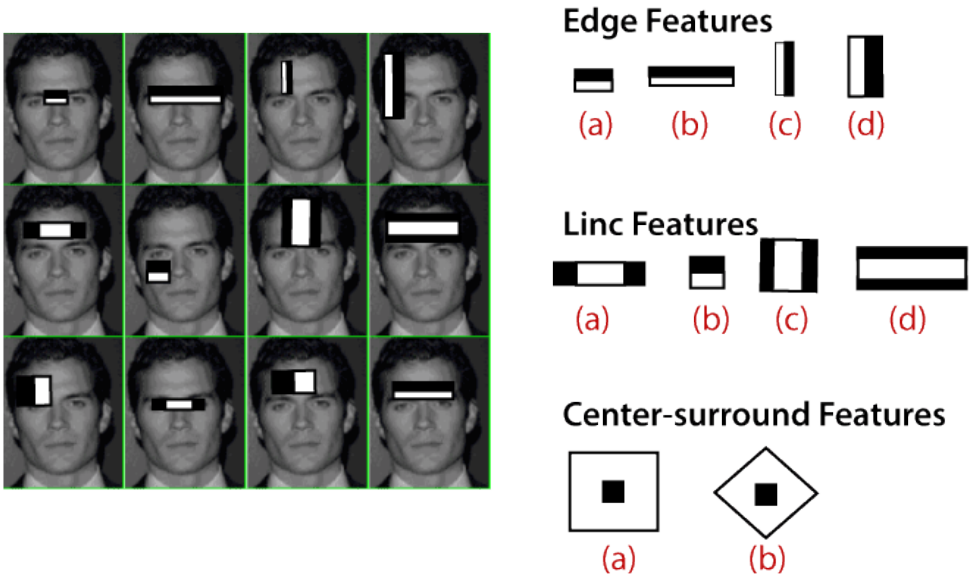

Figura 2.4: Representación gráfica de algoritmo Viola-Jones [18].

2.2.2. HOG + SVM

El *Machine Learning* es una rama de la Inteligencia Artificial la cual se encarga del estudio de sistemas capaces de aprender automáticamente a realizar determinadas tareas por medio de técnicas estadísticas. En definitiva, un sistema de *Machine Learning* es capaz de identificar patrones sobre un conjunto de datos y mejorar sus resultados o disminuir su error por medio de la experiencia.

Las SVM o *Support Vector Machines* son un modelo propio del aprendizaje supervisado perteneciente a la familia de los clasificadores lineales desarrollado por Vladimir Vapnik. Este tipo de modelo es capaz de resolver problemas de clasificación y regresión mediante un conjunto de datos de entrenamiento. Como idea básica, las SVM encuentran el hiperplano que separa de forma óptima cada par de clases sobre el espacio de las muestras, las cuales pueden ser previamente proyectadas a un espacio de mayor dimensionalidad mediante funciones núcleo o *Kernels*.

Se puede dar el caso en el que las clases no sean linealmente separables sobre el espacio de muestras, por ello, las *SVM* aportan el concepto de núcleo o *Kernel*. Los *Kernel* son un tipo de funciones capaces de transformar las muestras de su espacio a un nuevo espacio de mayor dimensionalidad, es decir, mediante éstos las *SVM* son capaces de transformar un conjunto de muestras no linealmente separables a un nuevo conjunto de muestras sobre un espacio de mayor dimensionalidad dónde sí son linealmente separables. En la figura 2.5 podemos observar como la aplicación del *Kernel* sobre un conjunto de datos no linealmente separables transformar el espacio de forma que las clases terminan siendo linealmente separables.

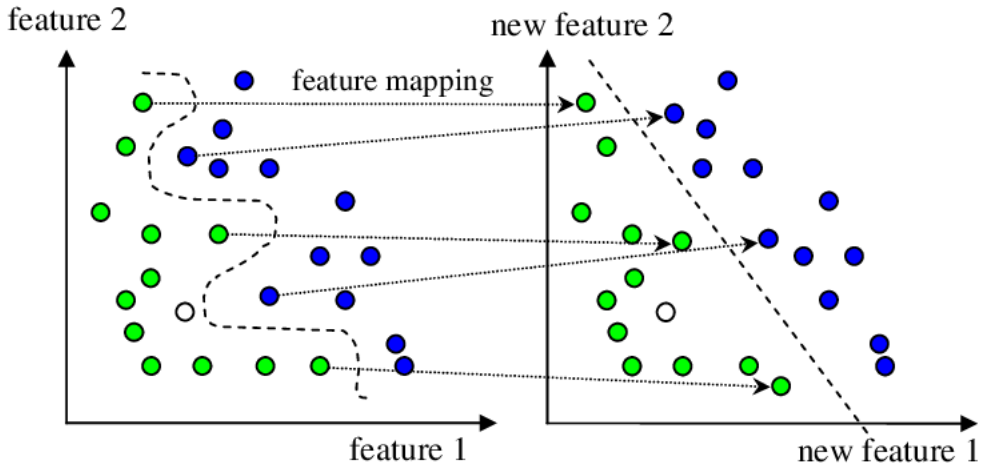

Figura 2.5: Representación gráfica del *Kernel* o núcleo. [49]

La unión de la extracción de características por Histograma de Gradientes Orientados y el modelo de *Machine Learning* de Máquina de Vectores Soporte, ambas expuestas anteriormente, permite también la identificación de rostros en una imagen de forma precisa y computacionalmente eficiente.

El funcionamiento de este sistema es sencillo de comprender una vez adquirido el conocimiento sobre los HOG y las SVM. El proceso consiste en deslizar una ventana a través de la imagen en la cual, en primer lugar, se extraen las características HOG, las cuales son propagadas a la SVM para su clasificación, es decir, para saber si en dicha ventana hay un rostro. Tras entrenar la SVM para una gran cantidad de imágenes positivas y negativas con sus respectivas características HOG, el sistema es capaz de detectar de forma exitosa rostros en una imagen [24], tal y como se representa en la figura 2.6. La librería `Dlib` incluye un modelo de detección basado en esta técnica y preentrenado para la detección de rostros en una imagen. [14]

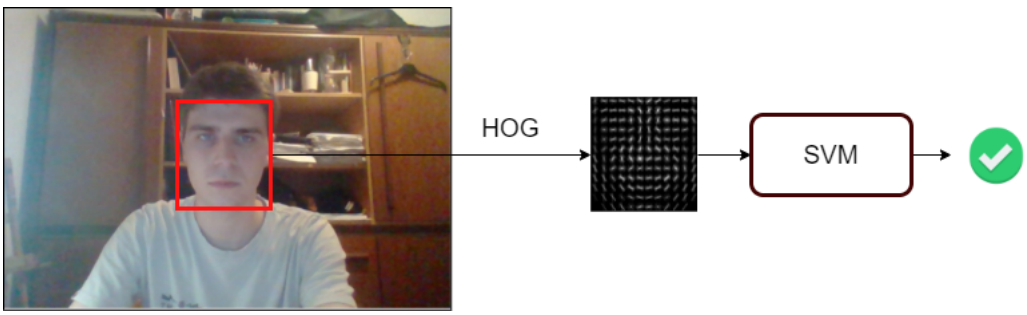

Figura 2.6: Representación gráfica del algoritmo de detección facial HOG+SVM.

2.2.3. CNN

El *Deep Learning* o aprendizaje profundo es una rama del *Machine Learning* que nace como un intento de simular la estructura neuronal humana con el fin de realizar algoritmos capaces de desempeñar todo tipo de tareas, especialmente relacionadas con tareas que podemos realizar los seres humanos, tales como el identificación del habla, realizar predicciones o la identificación de objetos en una imagen. Las redes neuronales se pueden definir como un grupo de nodos conectados de forma similar a las neuronas en un cerebro biológico. Cada uno de estos nodos representa una neurona artificial con sus propios parámetros internos, la cual es capaz tanto de recibir como de enviar información a otras neuronas. Normalmente estos nodos suelen ir agrupados en grupos, a los cuales defini-

mos como capas, por lo que la información recibida en la capa de entrada es propagada hacia las capas intermedias u ocultas hasta la capa de salida. El término profundo en el aprendizaje profundo simplemente hace referencia al hecho de que las redes neuronales contienen capas ocultas de procesamiento.

Las redes neuronales convolucionales o CNN's son un tipo de red neuronal artificial aplicadas mayoritariamente en el campo de la visión por computador y detección facial. La arquitectura de este tipo de redes aprovecha el hecho de que las imágenes a diferencia de otro tipo de estructuras de datos, tienen una estructura de la información clara. Es decir, la relación entre un píxel y otro viene dada por la propia estructura de la imagen, por lo que la información de un píxel estará altamente relacionada con sus "píxeles vecinos". Gracias a este tipo de redes, podemos realizar diferentes tareas como la extracción de características o la clasificación de clases dada una imagen entrada.

La arquitectura de este tipo de redes viene dada por la alternancia de dos tipos de capas, las capas de convolución y las capas de *Pooling*.

Capa de convolución

En esta capa se produce un filtrado de la imagen mediante núcleos, los cuales son capaces de extraer característica como bordes o esquinas. Durante el aprendizaje, los filtros de las capas de convolución son actualizados de forma que pueden adquirir características cada vez más abstractas. En la figura 2.7 se puede apreciar la aplicación del núcleo sobre una celda de la imagen de entrada.

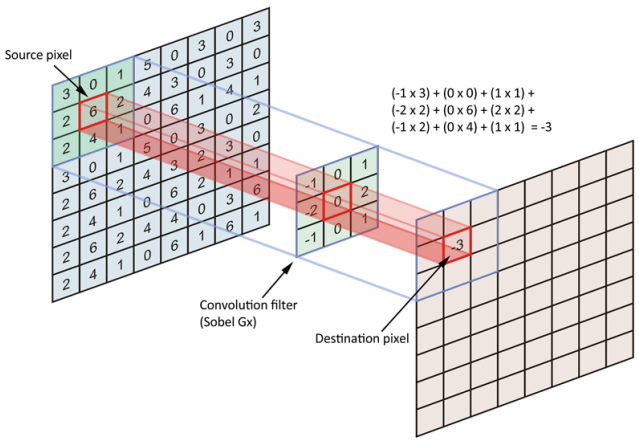

Figura 2.7: Representación gráfica del proceso de convolución. [11]

Capa de *Pooling*

Las capas de convolución, definidas en el apartado anterior, son capaces de extraer características de la imagen con éxito. Sin embargo, pequeños cambios en la imagen de entrada pueden generar variaciones no deseadas en la extracción de características. Por ello, entre cada capa de convolución se añaden previamente las capas de *Pooling* o de reducción del muestreo. Este tipo de capas generan una versión de las características de menor resolución, manteniendo la información estructural de importancia. Ta y como se puede apreciar en la figura 2.8.

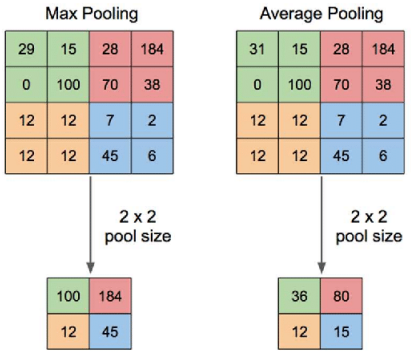

Figura 2.8: Representación gráfica del proceso de *Pooling*. [11]

Capa totalmente conectada

Tras la alternancia de las capas de convolución y *Pooling* anteriormente expuestas, la red es capaz de extraer de la imagen de entrada las características más significativas. De forma que obtendremos un mapa de características de ésta. A través de este mapa de características podemos realizar diferentes tareas, entre ellas, la clasificación.

En el caso de que la tarea de la red sea la de clasificar la imagen entre distintas posibles clases, se aplica finalmente una capa totalmente conectada o *Fully Connected Layer*, la cual recibirá el mapa de características y a través de éste computará la clase a la cual pertenece la imagen, como por ejemplo, detectar si existe un rostro en ésta. En la figura 2.9 se puede apreciar una red neuronal compuesta de un total de dos capas ocultas totalmente conectadas.

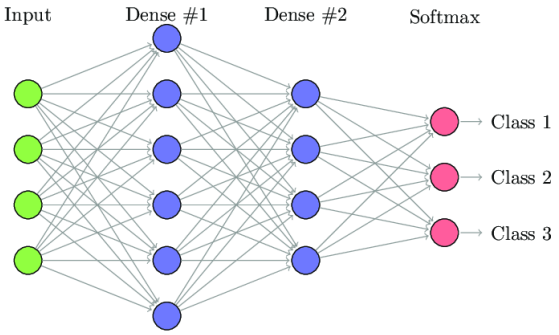

Figura 2.9: Representación gráfica de dos capas ocultas totalmente conectadas. [11]

La aparición de las redes neuronales convolucionales o CNN's y sus derivados supone un importante avance en el campo de la visión por computador, especialmente en el campo de detección de formas, objetos y, como era de esperar, de rostros. La librería `Dlib` incluye un modelo de detección basado en esta técnica y preentrenado para la detección de rostros en una imagen. [14]

2.2.4. MTCNN

Las redes MTCNN (*Multi-task Cascaded Convolutional Networks*) son una variación de las redes convolucionales capaces de realizar, como su nombre indica, varias tareas. En este caso, este tipo de redes pueden efectuar la detección de rostros al mismo tiempo que extraen puntos de referencia para cada uno de estos, tal y como presenta Kaipeng Zhang

y otros investigadores en el reconocido *artículo Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks* [26].

La arquitectura de esta red viene dado por la concatenación de tres subredes neuronales convolucionales, expuestas anteriormente:

1. **P-Net**: el primer nivel donde se producen ventanas candidatas rápidamente mediante una CNN sencilla.
2. **R-Net**: en este segundo nivel una CNN algo más compleja es la encargada de descartar una gran cantidad de ventanas candidatas propagadas por la anterior red.
3. **O-Net**: finalmente una red compleja es la encargada de refinar las predicciones propagadas por la anterior red y localizar las posiciones de los puntos de referencia faciales.

Además este tipo de redes son capaces de extraer un mapa de características sobre un rostro dado, que en su concreción matemática se define como un punto dentro de un espacio latente de alta dimensionalidad, por ejemplo, 128 dimensiones. Por lo que mediante estas redes se pueden comparar rostros. `FaceNet` es el modelo MTCNN más popular actualmente. Ya que existe un modelo preentrenado disponible para diversos lenguajes de programación, entre ellos, `Python` [20]. En la figura 2.10 se puede apreciar el proceso de detección mediante el uso de las subredes anteriormente expuestas.

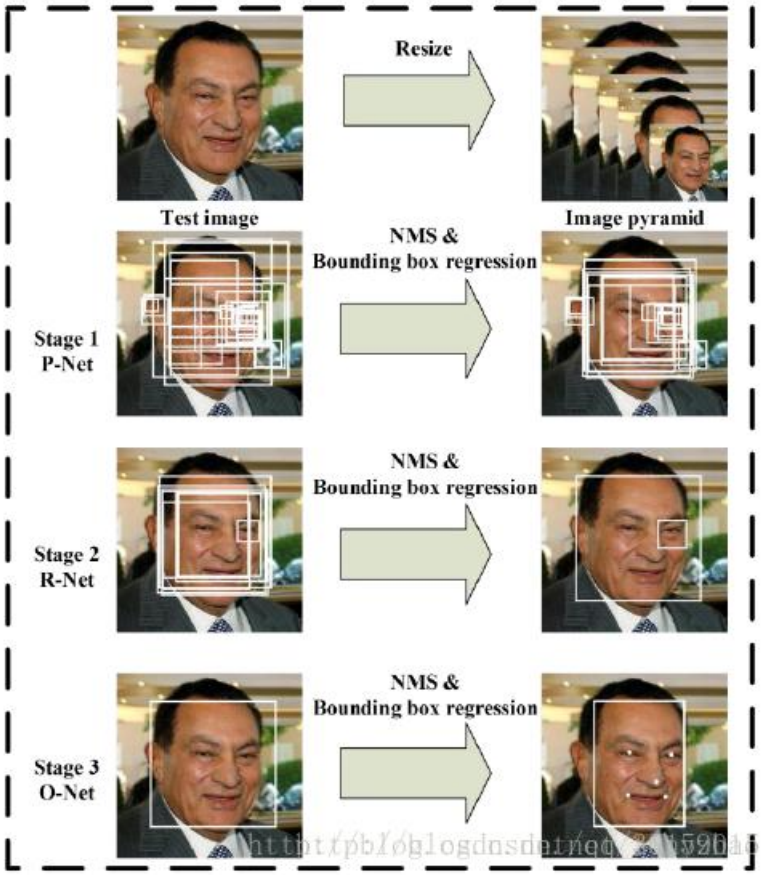

Figura 2.10: Representación gráfica de la red MTCNN. [22]

2.3 Identificación facial

Tras la detección de rostros mediante alguno de los algoritmos expuestos anteriormente, una tarea interesante es la de reconocer a qué individuo pertenece dicho rostro. Actualmente, los sistemas de identificación facial con mejores resultados son los basados en *Deep Learning*, concretamente mediante el uso de CNN's.

Como se ha mencionado anteriormente, las redes neuronales convolucionales pueden resultar extremadamente útiles a la hora de extraer características no triviales de una imagen de entrada. `Face Recognition (Python)` [19] es una de las librerías más populares capaces de realizar esta tarea. Su función `face-encodings ( detected-face )` es capaz de extraer un mapa de características dado un rostro detectado mediante el uso de redes neuronales convolucionales, también definido como codificación facial. Dicha codificación se define matemáticamente como un punto dentro de una 128-esfera, es decir, una esfera de 128 dimensiones.

La ventaja de este método es que, como ha sido expuesto anteriormente en el apartado de extracción de características mediante CNN's, es posible realizar operaciones entre las codificaciones faciales. De forma que podemos calcular la "distancia" entre rostros y de esta forma estimar su parecido. En el caso de `Face Recognition` estimaremos que dos caras pertenecen al mismo individuo siempre que la distancia euclídea entre sus respectivos mapas de características es menor o igual que 0.6 unidades. En la figura 2.11 se puede apreciar el proceso de codificación facial obteniendo un vector de 128 dimensiones (*Bootleneck*).

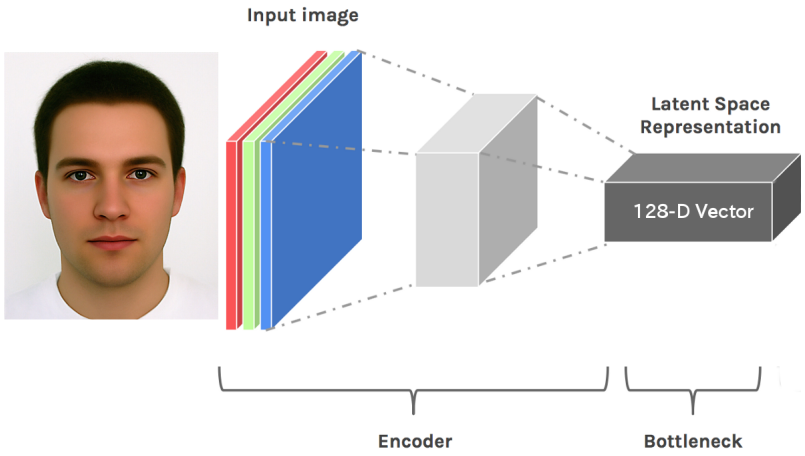

Figura 2.11: Representación gráfica de la codificación facial. [30]

2.4 Extración de puntos de referencia faciales

Una vez expuestos los conceptos de detección y identificación facial, nos plantemos la siguiente cuestión: ¿Hasta qué punto podemos extender la cantidad de información relativa al rostro humano? En el proyecto de identificación facial desarrollado por Morcillo [13] se introduce una funcionalidad añadida al sistema para la detección de *'liveness'*, es decir, discernir si el rostro detectado se trata de un sujeto real o, en cambio, de una fotografía. Para ello, Morcillo hace uso de los puntos de referencia faciales para detectar si el rostro detectado realiza cambios de expresión, muecas o gestos faciales.

Para el caso de estudio que nos concierne, la monitorización facial, la extracción de puntos de referencia faciales, a los cuales nos referiremos a partir de este momento como *landmarks*, resulta crucial. La correcta extracción de estos puntos resulta de gran utilidad

para computar distintos parámetros faciales como la posición de la cabeza, los gestos y las emociones. En la figura 2.12 podemos observar la disposición de un modelo de 68 *landmarks*.

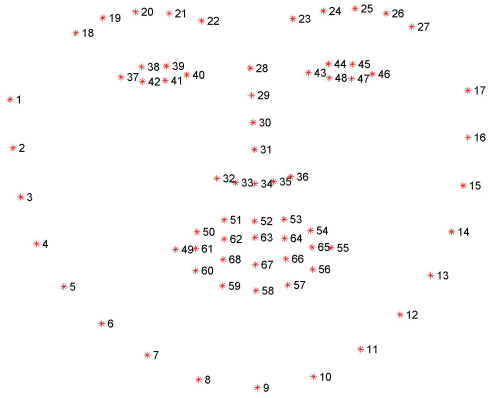

Figura 2.12: Representación gráfica de los puntos de referencia faciales [42]

Como hemos visto en el apartado anterior relativo a las redes MTCNN, éstas son capaces de extraer puntos de referencia. Sin embargo, en este apartado nos centraremos en una de las técnicas de extracción de puntos de referencia faciales más popular. La cual ha sido implementada en la librería de código abierto `Dlib`.

Cabe destacar que este método, a diferencia de las redes MTCNN, requiere de la previa detección del área del rostro mediante algún algoritmo de detección facial, entre los cuales se incluyen los expuestos anteriormente, inclusive las propias redes MTCNN.

2.4.1. Extracción de *landmarks* mediante árboles de regresión

Los árboles de decisión en el campo del *Machine Learning* fueron propuestos en 1984 por Leo Breiman [4]. Un árbol de decisión se construye como un conjunto de reglas sucesivas que ayuda a tomar una decisión para un problema dado. En el caso del Aprendizaje Automático, este tipo de modelos son utilizados para predecir una variable respuesta en función de la entrada.

Según la naturaleza de la variable respuesta los árboles de decisión se pueden dividir en dos casos:

1. **Árbol de clasificación**: cuando la variable respuesta es cualitativa.
2. **Árbol de regresión**: cuando la variable respuesta es cuantitativa.

Uno de los métodos más populares y ampliamente utilizados para la extracción de *landmarks* es el presentado en el artículo "*One Millisecond Face Alignment with an Ensemble of Regression Trees*" [33]. El cual presenta un método de resolución basado en un conjunto de árboles de regresión.

Para ello, el sistema consta de distintas etapas. En primer lugar, aprovechando el hecho de que con mayor probabilidad la orientación rostro será frontal, la primera aproximación del método serán los *landmarks* situados como si el rostro detectado es directamente frontal. A partir de esta primera aproximación, se aplica iterativamente cada etapa realizando una transformación sobre el conjunto de *landmarks* hasta converger a la solución final.

Así mismo, cada etapa consta de un conjunto de árboles de regresión. Los cuales, en función de la relación entre los píxeles de la región del rostro, predicen una transformación sobre el conjunto de *landmarks*. De esta forma, después de entrenar el modelo, éste es

capaz de aplicar sucesivas transformaciones sobre la predicción inicial hasta converger a la estimación correcta de forma altamente eficiente computacionalmente.

Este método se encuentra implementado en la librería de código abierto `Dlib`, la cual incluye también un modelo preentrenado de 68 *landmarks* [15]. Dicho conjunto de *landmarks* es el relativo a la figura 2.12 anteriormente expuesta.

CAPÍTULO 3
Análisis del problema

Actualmente existen multitud de herramientas y algoritmos capaces de detectar, reconocer y extraer *landmarks* sobre rostros, tal y como expone Morcillo [13]. Sin embargo, existen muchos parámetros medibles relativos al rostro humano como los gestos, la mirada, expresiones, pose, etc.

En este capítulo se analiza con detalle el problema al que nos enfrentamos, con el fin de implementar un sistema de monitorización facial a tiempo real y robusto, en la medida de lo posible. El sistema debe ser capaz de resolver distintas tareas relativas a la monitorización facial tales como:

- Extracción de imágenes mediante sensor o fichero de vídeo.
- Extracción de rostros.
- Extracción de puntos de referencia faciales para los rostros detectados.
- Extracción de la posición y rotación de dichos rostros.
- Estimar la dirección de la mirada.
- Estimar diferentes gestos faciales.
- Estimar emociones en función de los gestos faciales.
- Estimación de la edad y género.
- identificación facial.
- filtrado los datos recogidos y corrección de sus inestabilidades.
- Guardado de todos los datos analizados en un fichero.

Además, a partir de este sistema se pretende estudiar la robustez que éste presenta realizando esta mediciones en tiempo real, entendiendo mediciones como los parámetros faciales que podemos estimar (posición, pose, gestos, emociones, etc). Así mismo, se analizará su rendimiento ante perturbaciones. Por ejemplo, ante cambios de iluminación, rotación de la pose del rostros, oclusiones, etc.

3.1 Requisitos

En esta sección se detallan tos los posibles requisitos funcionales y no funcionales que debe cumplir el sistema de monitorización.Así mismo, dichos requisitos establecen un código identificador, de forma que en la etapa de pruebas se pueda analizar con detalle el cumplimiento de cada uno de éstos.

3.1.1. Requisitos funcionales

Los requisitos funcionales hacen referencia a las distintas tareas o funcionalidades que el sistema debe ser capaz de cumplir con éxito.

Requisitos relativos a la extracción de imágenes

Identificador	RF-CAM-001
Título	Definir lista de sensores disponibles
Descripción	El sistema debe ser capaz de reconocer los sensores (cámaras) disponibles en el equipo.
Módulo asociado	Extracción de imágenes mediante sensor o fichero de vídeo.

Identificador	RF-CAM-002
Título	Seleccionar el sensor
Descripción	El sistema debe ser capaz de seleccionar uno los sensores disponibles en el equipo.
Módulo asociado	Extracción de imágenes mediante sensor o fichero de vídeo.

Identificador	RF-CAM-003
Título	Seleccionar fichero de vídeo
Descripción	El sistema debe ser capaz de seleccionar un fichero de vídeo para su posterior análisis.
Módulo asociado	Extracción de imágenes mediante sensor o fichero de vídeo.

Identificador	RF-CAM-004
Título	Escalar *frames* obtenidos mediante el sensor
Descripción	El sistema debe ser capaz de escalar la imagen recibida mediante el sensor o vídeo.
Módulo asociado	Extracción de imágenes mediante sensor o fichero de vídeo.

Requisitos relativos a la detección de rostros

Identificador	RF-FD-001
Título	Detectar rostros en una imagen
Descripción	El sistema debe ser capaz de detectar los rostros frontales a la cámara en una imagen.
Módulo asociado	Extracción de rostros.

Identificador	RF-FD-002
Título	Extraer puntuación relativa a la detección de rostros
Descripción	El sistema debe ser capaz de, tras detectar los rostros, asignar a cada uno de estos una puntuación sobre el grado de confianza que hay de que tal detección sea un rostro y por tanto, en caso de duda, poder discernir entre ellos.
Módulo asociado	Extracción de rostros.

Identificador	RF-FD-003
Título	Detectar el rostro más cercano al sensor
Descripción	El sistema debe ser capaz de, tras detectar los rostros, extraer el rostro más cercano al sensor, es decir, el rostro con mayor área sobre la imagen.
Módulo asociado	Extracción de rostros.

Requisitos relativos a la extracción de *landmarks*

Identificador	RF-LP-001
Título	Extraer puntos de referencia faciales
Descripción	El sistema debe ser capaz de, tras detectar los rostros, extraer los puntos de referencia faciales relativos a dichos rostros.
Módulo asociado	Extracción de puntos de referencia faciales para los rostros detectados.

Identificador	RF-LP-002
Título	Dibujar los puntos de referencia faciales
Descripción	El sistema debe ser capaz de, tras detectar los rostros y la extracción de los puntos de referencia faciales, dibujar éstos sobre la imagen.
Módulo asociado	Extracción de puntos de referencia faciales para los rostros detectados.

Requisitos relativos a la estimación de la pose

Identificador	RF-PE-001
Título	Estimar la traslación de la cabeza
Descripción	El sistema debe ser capaz de estimar de forma aproximada la posición de la cabeza en el espacio tridimensional con su origen en el sensor.
Módulo asociado	Extracción de la posición y rotación de dichos rostros.

Identificador	RF-PE-002
Título	Estimar la rotación de la cabeza
Descripción	El sistema debe ser capaz de estimar de forma aproximada la rotación de la cabeza sobre los tres ejes del sistema de coordenadas.
Módulo asociado	Extracción de la posición y rotación de dichos rostros.

Identificador	RF-PE-003
Título	Dibujar el cubo que contiene a la cabeza asociada al rostro
Descripción	El sistema debe ser capaz de dibujar sobre el frame un cubo asociado al rostro detectado con su respectiva traslación y rotación.
Módulo asociado	Extracción de la posición y rotación de dichos rostros.

Identificador	RF-PE-004
Título	Dibujar los puntos de referencia faciales utilizados para la estimación de la pose
Descripción	El sistema debe ser capaz de dibujar sobre el frame los puntos faciales asociados a la estimación de la pose.
Módulo asociado	Extracción de la posición y rotación de dichos rostros.

Requisitos relativos a la estimación de la dirección de la mirada

Identificador	RF-GE-001
Título	Estimar la dirección de la mirada
Descripción	El sistema debe ser capaz de estimar la dirección de la mirada sobre el eje horizontal relativo al rostro detectado.
Módulo asociado	Estimar la dirección de la mirada.

Identificador	RF-GE-002
Título	Dibujar la dirección de la mirada
Descripción	El sistema debe ser capaz de dibujar la dirección de la mirada sobre el *frame*.
Módulo asociado	Estimar la dirección de la mirada.

Identificador	RF-GE-003
Título	Ajustar el parámetro *threshold* sobre la detección de la mirada
Descripción	El usuario debe ser capaz de ajustar el parámetro *threshold* utilizado para la estimación de la dirección de la mirada.
Módulo asociado	Estimar la dirección de la mirada.

Requisitos relativos a la detección de gestos faciales

Identificador	RF-AUD-001
Título	Detectar sonrisa
Descripción	El sistema debe ser capaz de detectar la sonrisa en un rostro en el rango [0,1].
Módulo asociado	Estimar los gestos faciales.

Identificador	RF-AUD-002
Título	Detectar apertura de boca
Descripción	El sistema debe ser capaz de detectar la apertura de boca en un rostro en el rango [0,1].
Módulo asociado	Estimar los gestos faciales.

Identificador	RF-AUD-003
Título	Detectar gesto de beso
Descripción	El sistema debe ser capaz de detectar el gesto de beso en un rostro en el rango [0,1].
Módulo asociado	Estimar los gestos faciales.

Identificador	RF-AUD-004
Título	Detectar pestañeo
Descripción	El sistema debe ser capaz de detectar pestañeos sobre un rostro en el rango [0,1].
Módulo asociado	Estimar los gestos faciales.

Identificador	RF-AUD-005
Título	Detectar gesto de cejas
Descripción	El sistema debe ser capaz de detectar el movimiento de las cejas sobre el eje vertical relativo a un rostro en el rango [0,1].
Módulo asociado	Estimar los gestos faciales.

Identificador	RF-AUD-006
Título	Robustez de la detección de gestos
Descripción	El sistema de detección de gestos faciales debe de ser robusta ante cambios de iluminación, rotación, traslación, etc.
Módulo asociado	Estimar los gestos faciales.

Requisitos relativos a la detección de emociones

Identificador	RF-EE-001
Título	Detección de felicidad
Descripción	El sistema debe ser capaz de estimar si un rostro presenta rasgos relativos a la felicidad en función de los gestos faciales detectados.
Módulo asociado	Estimar emociones en función de los gestos faciales.

Identificador	RF-EE-002
Título	Detección de sorpresa
Descripción	El sistema debe ser capaz de estimar si un rostro presenta rasgos relativos a la sorpresa en función de los gestos faciales detectados.
Módulo asociado	Estimar emociones en función de los gestos faciales.

Identificador	RF-EE-003
Título	Detección de cansancio
Descripción	El sistema debe ser capaz de estimar si un rostro presenta rasgos relativos al cansancio en función de los gestos faciales detectados.
Módulo asociado	Estimar emociones en función de los gestos faciales.

Identificador	RF-EE-004
Título	Detección de enfado
Descripción	El sistema debe ser capaz de estimar si un rostro presenta rasgos relativos al enfado en función de los gestos faciales detectados.
Módulo asociado	Estimar emociones en función de los gestos faciales.

Requisitos relativos a la estimación de la edad y género

Identificador	RF-AGE-001
Título	Estimación de la edad
Descripción	El sistema debe ser capaz de estimar la edad dado un rostro.
Módulo asociado	Estimación de la edad y género.

Identificador	RF-AGE-002
Título	Estimación del género
Descripción	El sistema debe ser capaz de estimar el género dado un rostro.
Módulo asociado	Estimación de la edad y género.

Requisistos relativos al identificación facial

Identificador	RF-FR-001
Título	Codificación del rostro
Descripción	El sistema debe ser capaz de codificar el rostro mediante ténicas de *Deep Learning*.
Módulo asociado	Identificación facial.

Identificador	RF-FR-002
Título	Comparación de rostros
Descripción	El sistema debe ser capaz de comparar rostros mediante sus respectivas codificaciones
Módulo asociado	Identificación facial.

Identificador	RF-FR-003
Título	Carga y guardado de las codificaciones
Descripción	El sistema debe ser capaz de cargar y guardar las codificaciones de los rostros con sus respectivos nombre asociados
Módulo asociado	Identificación facial.

Requisitos relativos al filtrado y predicción de las mediciones

Identificador	RF-STB-001
Título	Filtrado de las mediciones
Descripción	El sistema debe ser capaz de filtrar las mediciones recogidas mediante el análisis y filtrarlas mediante un filtro de Kalman para suavizar la inestabilidad de éstas.
Módulo asociado	Filtrado de los datos recogidos y corrección de sus inestabilidades.

Identificador	RF-STB-002
Título	Predicción de las mediciones
Descripción	El sistema de filtrado debe ser capaz de predecir cual será la medición de los parámetros en el frame actual basándose en los datos recogidos en los frames anteriores.
Módulo asociado	Filtrado de los datos recogidos y corrección de sus inestabilidades.

Requisitos relativos a la carga, guardado y filtrado de las mediciones analizadas

Identificador	RF-REC-001
Título	Guardado de las mediciones
Descripción	El sistema debe ser capaz de guardar todos los datos analizados en un fichero.
Módulo asociado	Carga, guardado y filtrado de las mediciones analizadas.

Identificador	RF-REC-002
Título	Cargado de las mediciones
Descripción	El sistema debe ser capaz de cargar todos los datos analizados en un fichero.
Módulo asociado	Carga, guardado y filtrado de las mediciones analizadas.

Identificador	RF-REC-003
Título	Filtrado de las mediciones
Descripción	El sistema debe ser capaz de filtrar el fichero de datos guardado y mostrar los *frames* asociados a una serie de condiciones establecidas por el usuario. Por ejemplo, se puede filtrar un vídeo y extraer de éste los *frames* donde un rostro esté sonriendo.
Módulo asociado	Carga, guardado y filtrado de las mediciones analizadas.

3.1.2. Requisitos no funcionales

Identificador	RNF-001
Descripción	El sistema debe funcionar correctamente en `Python 3.9`

Identificador	RNF-002
Descripción	La tasa de *frames* por segundo debe superar los 10 FPS realizando todas las mediciones al mismo tiempo

3.2 Identificación y análisis de soluciones posibles

Existe una gran variedad de técnicas y herramientas de gran utilidad para el desarrollo de un sistema de monitorización facial. En esta sección se identificarán las posibles soluciones relativas a esta tarea y se justificará la elección de las herramientas escogidas.

En primer lugar, se ha optado por el uso de la librería de código abierto `OpenCV`. `OpenCV` es una herramienta ampliamente utilizada en el campo de la visión por computador y el *Machine Learning*, la cual cuenta con una amplia variedad de algoritmos relacionados con el procesamiento de imágenes, identificación de formas y objetos, extracción de características, etc. `OpenCV` se encuentra disponible para múltiples lenguajes de programación como `Python, C, C++ o Java`.

En este proyecto se ha optado por el uso de `OpenCV-Python` para el desarrollo del sistema. La razón de esta elección se basa en que el lenguaje de programación `Python` es sencillo, ampliamente utilizado y, a pesar de que una de las principales críticas de este lenguaje es el de su velocidad, cabe destacar que las funciones de la librería `OpenCV-Python` son ejecutadas como código `C++` por lo que se trata de una herramienta altamente eficiente independientemente del lenguaje de programación escogido para su uso, en este caso `Python`.

Para la manipulación de vectores, matrices y funciones matemáticas de alto nivel se ha optado por el uso de la librería `NumPy`. Se trata de una de las librerías más populares dentro del lenguaje de programación `Python` dado que se trata de una herramienta altamente eficiente y de sencillo uso.

A continuación se analizarán las distintas técnicas para la detección, identificación y demás subproblemas a los que el sistema desarrollado debe dar solución

3.2.1. Detección Facial

Para la detección de rostros se ha optado por la implementación de tres algoritmos diferentes. El primero de éstos hace uso de la librería `Dlib` mediante su método basado en la fusión de la extracción de características HOG y una SVM para la detección dado que se trata de uno de los métodos más rápidos actualmente.

El segundo, también perteneciente a la librería `Dlib`, hace uso de una CNN para la detección de rostros. Cabe destacar que, tal y como se expresa en la documentación de `Dlib` acerca de este método [5], se recomienda el uso de este algoritmo siempre que se pueda ejecutar en GPU, ya que en caso contrario resulta extremadamente lento. Por desgracia, no contamos con un equipo capaz de realizar la detección en GPU. A pesar de ello, el algoritmo está implementado en el sistema para aquellos usuarios que dispongan de equipos capaces de ejecutar el algoritmo sobre GPU.

Finalmente, el último algoritmo implementado es el de la librería `cvlib` [7], cuya arquitectura es semejante a la de la librería `FaceNet`, el cual funciona también mediante CNN's y, al contrario que el anterior, funciona correctamente a tiempo real.

Para la elección de estos algoritmos se ha realizado un proceso de experimentación en la plataforma *Google Colab* (3.1),la cual permite ejecutar y programar en `Python` en tu navegador de forma gratuita, sin arduos procesos de configuración y que permite permite compartir contenido fácilmente [21].

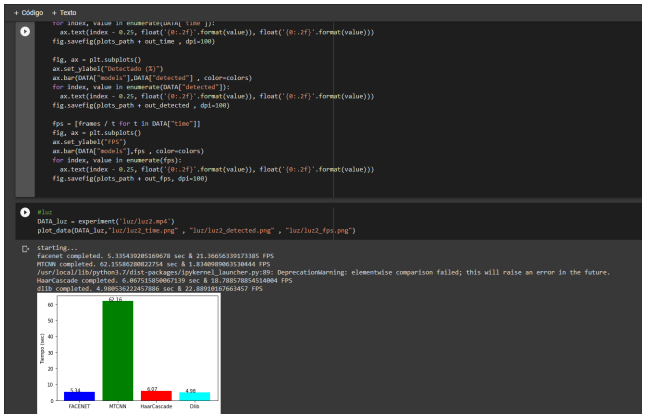

Figura 3.1: Proyecto de experimentación en `Google Colab`. Elaboración propia.

Para comparar el funcionamiento de distintos tipos de algoritmos de detección facial (FaceNet, MTCNN, HaarCascade, Dlib HOG+SVM), las mediciones que se han considerado relevantes para el análisis de estos algoritmos son las siguientes:

1. **FPS o *Frames* por segundo**: describe cuántos *frames* por segundo es capaz de analizar el algoritmo.

2. **Porcentaje de detección**: describe que porcentaje de tiempo el algoritmo ha sido capaz de detectar un rostro acertadamente.

Para ello, se han realizado cuatro vídeos diferentes con el fin de estudiar el comportamiento de dichos algoritmos ante distintas situaciones:

1. **Control**: se trata de un vídeo en el cual no se realiza ningún movimiento brusco, sin cambios de iluminación u otras posibles perturbaciones.

2. **Iluminación**: se trata de un vídeo con el que se pretende analizar el comportamiento de los algoritmos ante distintas iluminaciones. En el vídeo se pasa de una habitación totalmente a oscuras a otra con buena iluminación.

3. **Rotación**: en este vídeo se realizan rotaciones y traslaciones de la cabeza con el fin de estudiar el comportamiento de los algoritmos ante poses de la cabeza distintas.

4. **Oclusión**: se trata de un vídeo dónde, en primer lugar, el sujeto lleva gafas de sol y mascarilla para posteriormente quitárselas. La finalidad de este vídeo es estudiar el comportamiento de los algoritmos ante oclusiones.

Cabe destacar que los FPS que se han obtenido como resultado son una referencia y no se verán reflejados en la solución final dado que en ésta se aplicarán distintas técnicas de optimización con el fin de aumentar los FPS del detector facial.

Control

Ante mínimas perturbaciones podemos observar, como se muestra en la gráfica izquierda de la figura 3.2, que los cuatro algoritmos de estudio han sido capaces de detectar el rostro en todo momento. Sin embargo, podemos observar en la parte derecha de la figura 3.2 una importante diferencia de FPS según el algoritmo de detección, siendo FaceNet y Dlib (HOG+SVM) los más rápidos, y MTCNN el más lento con diferencia.

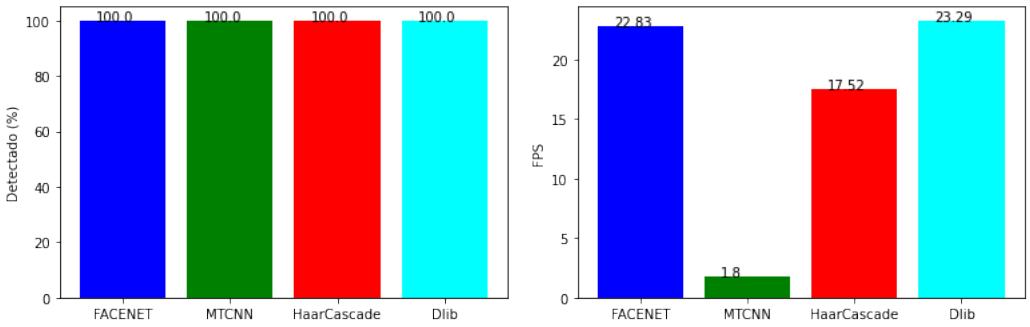

Figura 3.2: %Detectado y FPS para el vídeo de entrada `control`.

Iluminación

Ante cambios de iluminación podemos observar en la parte izquierda de la figura 3.3 que Dlib (HOG+SVM) y HaarCascade han sido los que mejor resultado han obtenido en porcentaje de detección, estando FaceNet y MTCNN por debajo. Para la medición de FPS, en la parte derecha de la figura 3.3 se un comportamiento similar al anterior caso. De hecho, se ha observado que los FPS no presentan mayores cambios ante distintas condiciones, por lo que a partir de este momento se aportarán los datos relativos a los FPS pero no se realizarán comentarios acerca de éstos.

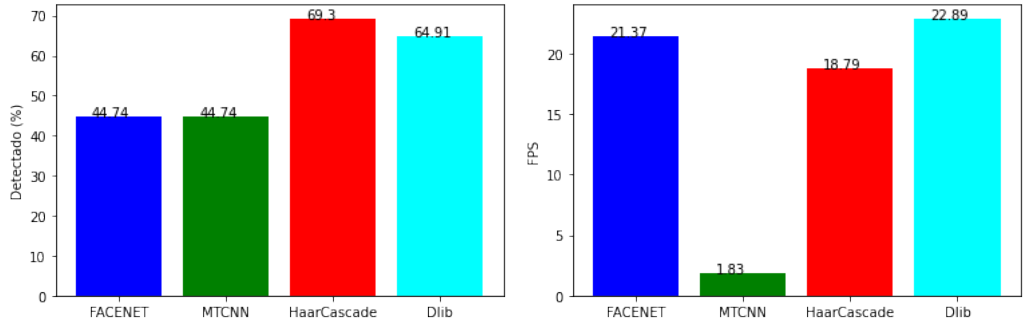

Figura 3.3: %Detectado y FPS para el vídeo de entrada `iluminación`.

Rotación

Ante rotaciones y traslaciones podemos observar, tal y como muestra la figura 3.4, que FaceNet y MTCNN son los algoritmos que mejores resultados obtienen respecto al porcentaje de detección, siguiéndole Dlib y finalmente HaarCascade.

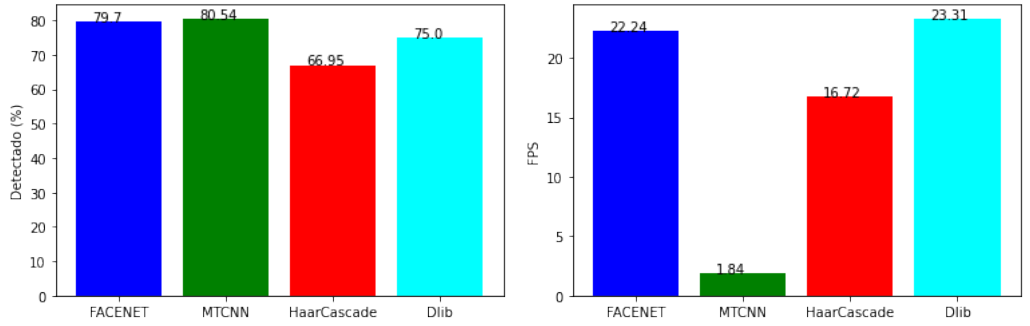

Figura 3.4: %Detectado y FPS para el vídeo de entrada `rotación`.

Oclusión

Por último, ante oclusiones podemos observar que los algoritmos presentan un comportamiento similar, siendo HaarCascade el que mejor resultados obtiene. Tal y como se muestra en la figura 3.5

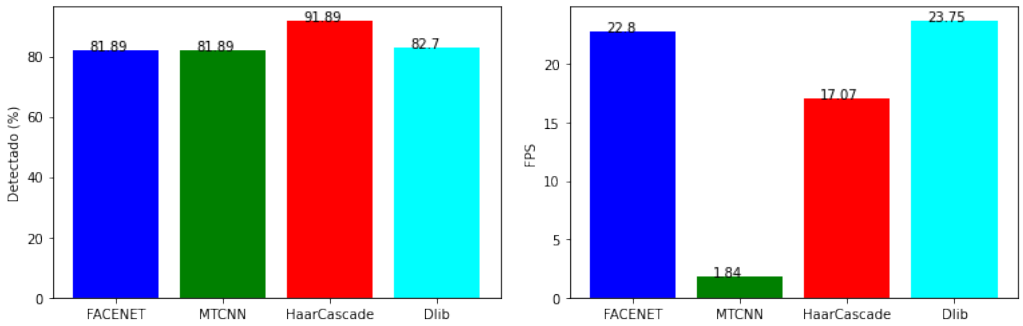

Figura 3.5: %Detectado y FPS para el vídeo de entrada `oclusión`.

3.2.2. Extracción de puntos de referencia faciales

Para la extracción de puntos de referencia faciales, se ha optado por el uso de la librería `Dlib`, la cual hace uso del método basado en árboles de regresión, expuesto en el segundo capítulo, para esta tarea. Se ha escogido este método dado que se trata de uno de los más rápidos y precisos del estado del arte, dado que uno de los principales objetivos de este proyecto es el desarrollo de un sistema que funcione a tiempo real. La librería `Dlib` cuenta con múltiples modelos de extracción de puntos de referencia faciales. `Dlib` cuenta con dos modelos de extracción de *landmarks*, uno de 5 *landmarks* y el otro de 68. En este caso se ha escogido el modelo de 68 *landmarks* dado que un mayor número de puntos aporta más información, la cual puede ser utilizada posteriormente para realizar otro tipo de mediciones en base a dichos *landmarks*.

Además, estos puntos de referencia faciales serán posteriormente utilizados para la detección de diferentes gestos faciales y, a su vez, dichos gestos para la detección de emociones. Este último caso se trata de un proceso de experimentación dado que actualmente muchos de los sistemas de identificación de emociones están basados en redes neuronales convolucionales.

3.2.3. Identificación facial

La librería `Face Recognition` [19], basada en *Deep Learning*, será utilizada para la extracción de la codificación facial de los rostros para su posterior análisis. `Face Recognition` se trata de una de las librerías de identificación facial más populares del campo y se caracteriza por poseer una sintaxis sencilla y fácil de aprender.

`DeepFace` es otra de las populares librerías capaces de realizar dicha codificación facial. Sin embargo, ésta cuenta con otro tipo de funcionalidades que en éste proyecto no son de utilidad. Es por ello por lo que, pensando en términos de uso de recursos, se ha optado por el uso de `Face Recognition`, dado que la librería ocupa menos espacio en memoria y ofrece la misma utilidad al sistema.

3.2.4. Estimación de edad, género y emociones mediante CNN's

Para la estimación de la edad, género y emociones las técnicas más efectivas son las basadas en CNN's. Existen multitud de modelos de *Deep Learning* preentrenados para

realizar esta tarea. Sin embargo, como se ha descrito anteriormente, uno de los principales objetivos de este proyecto es que la monitorización se pueda realizar a tiempo real con una tasa de FPS aceptable (10-30 FPS). Es por ello por lo que se ha optado por el uso de CNN's de bajo peso, es decir con menos parámetros internos y, por tanto, con un menor tiempo de cómputo.

Por ejemplo, la librería `Deep Face` cuenta también con esta funcionalidad. Sin embargo, tal y como se expone en su documentación [9], ésta es capaz de realizar dicho análisis en un tiempo medio de 5 segundos, lo cual equivale a 0.2 FPS.

Para la clasificación de edad y género se ha optado por el uso de los modelos preentrenados desarrollados por Tal Hassner en su publicación "*Age and Gender Classification Using Convolutional Neural Networks*" [1] ya que como se expresa en la publicación "*Real-time Age, Gender and Emotion Prediction from Webcam with Keras and OpenCV*" [38], presentan un buen rendimiento a tiempo real. Una de las razones de su eficiencia es que la imagen de entrada se preprocesa para formar una imagen de 64x64 píxeles en escala de grises, es decir, la entrada tiene un total de 4096 parámetros, un tamaño asequible teniendo en cuenta que no precisa de los tres canales de color RGB.

Por último, para la clasificación de emociones se ha optado por el uso del modelo desarrollado por Abhijeet Kumar en su publicación "*Demonstration of Facial Emotion Recognition on Real Time Video Using CNN : Python and Keras*" [12], el cual, al igual que que las redes anteriormente mencionadas, presenta un rendimiento asequible para el propósito que nos concierne con un tiempo de cómputo medio de 30 milisegundos.

3.3 Solución propuesta

En este proyecto se propone extender la cantidad de mediciones que se pueden extraer de un rostro mediante un sistema de monitorización facial completo fusionando diferentes técnicas y algoritmos existentes propios del *Deep Learning*, *Machine Learning* y la visión por computador. Para ello, el sistema contará con diferentes módulos interdependientes encargados de realizar las posibles mediciones que se deseen extraer de un rostro. Finalmente, se agruparán los distintos módulos en un sólo sistema capaz de realizar todas las mediciones al mismo tiempo resolviendo todas las interdependencias entre sus respectivos módulos. A continuación, se hará una breve exposición de los procesos implicados en la solución final, los cuales serán explicados con más detalle en el capítulo 5.

3.3.1. Extracción y estimación de las mediciones

En lo respectivo a la detección facial, se ha optado por implementar tres algoritmos diferentes en base al análisis expuesto previamente, entre los cuales se encuentran el HOG+SVM de `Dlib`, el CNN de `Dlib` y el CNN de `cvlib`.

A partir de la detección facial se procederá a la extracción de los puntos de referencia faciales haciendo uso del algoritmo implementado en `Dlib` basado en árboles de regresión, cuyo funcionamiento ha sido expuesto en el capítulo 2.

Una vez detectada la región de interés relativa al rostro y sus respectivos puntos de referencia faciales, podemos dividir el resto de mediciones en dos grandes bloques.

Mediciones estimadas a partir de los puntos de referencia faciales

Para la estimación de los gestos faciales, se ha seguido una aproximación inspirada en el artículo "*Eye Aspect Ratio(EAR) and Drowsiness detector using dlib*" [17], en el cual se

expone cómo detectar el pestañeo de un sujeto mediante el cálculo de distancias entre los *landmarks* extraídos del rostro. A partir de esta aproximación se ha realizado un módulo capaz de detectar otro tipo de gestos faciales como el movimiento de las cejas, la apertura de la boca, etc.

A pesar de que se ha implementado un módulo de detección de emociones mediante el uso de CNN's, se ha optado por realizar, a modo de experimento, un módulo de detección de emociones en base a los gestos faciales expuestos previamente. La razón de esta decisión se fundamenta en que, como es sabido, a pesar de que las CNN's puedan dar buenos resultados, su cómputo puede resultar costoso, sobre todo en equipos de baja potencia. Por lo tanto, se ha optado por la implementación de éste módulo con el fin de experimentar cuan efectiva puede resultar dicha aproximación.

Para la estimación de la pose de la cabeza se ha optado por una aproximación inspirada en el artículo "*Real-Time Head Pose Estimation With OpenCV and Dlib*" [41], en el cual se hace uso de la función `solve-pnp` de `OpenCV` mediante los *landmarks* detectados, sus respectivos puntos tridimensionales y la matriz de parámetros intrínsecos de la cámara.

La matriz intrínseca de la cámara, denotada como A, proyecta puntos en tres dimensiones sobre el plano imagen bidimensional. Entre los elementos de dicha matriz podemos encontrar las distancias focales para cada uno de los ejes del plano (f_x y f_y) y el punto central (c_x, c_y) [37], tal y como se puede apreciar en la ecuación 3.1.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \rightarrow s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.1}$$

Desafortunadamente, en la mayoría de dispositivos como móviles o computadores con cámara integrada no se aporta la información necesaria para extraer la matriz intrínseca. Sin embargo, existe una popular aproximación que se basa en la resolución del dispositivo (*width and height*). Donde $f_x = f_y = width$ y $(c_x, c_y) = (width/2, height/2)$, tal y como se expresa en el artículo [41].

Por último, para estimar la dirección de la mirada se ha optado por un proceso de análisis de la región de los ojos mediante los *landmarks* previamente extraídos. En primer lugar, se localiza la posición de la pupila mediante un procesamiento de imagen basado en el artículo "*Real-time eye tracking using OpenCV and Dlib*" [39]. Una vez extraída la posición de la pupila, se estima la dirección de la mirada a partir de la estimación de la pose de la cabeza.

Mediciones estimadas a partir de la región de interés del rostro

El segundo bloque se trata de las mediciones estimadas a partir de la región de interés del rostro, las cuales emplean técnicas de *Deep Learning* para su procesamiento. En concreto, se ha hecho uso de redes neuronales convolucionales o CNN's preentrenadas [1][38][12] para la estimación del género, la edad y los sentimientos representados del rostro detectado.

Por último, para el identificación facial, se hará uso de la librería `Face Recognition` mediante su función `face-encodings`, la cual, dada la región de interés del rostro como entrada, devuelve un vector de características de 128 dimensiones que representa la cara detectada. Mediante la librería `pickle` se podrá guardar la lista de *face encondings* de cada cara con su nombre asociado. Finalmente se compararán los diferentes *face encodings* mediante la función `linalg.norm` de la librería `NumPy`.

3.3.2. Interdependencias entre las mediciones

Como se puede observar, el conjunto de mediciones que deseamos extraer de un rostro presenta interdependencias entre sí. Es por ello por lo que se propone la creación de un nuevo módulo capaz de resolver este problema. Por lo tanto la propuesta da pie, tal y como se expondrá en el diseño de la solución, a la creación de un sistema cuya arquitectura sea la descomposición modular de control centralizado o arquitectura gestor.

En la figura 3.6 se puede observar el árbol de interdependencias relativo a la estimación de las diferentes mediciones.

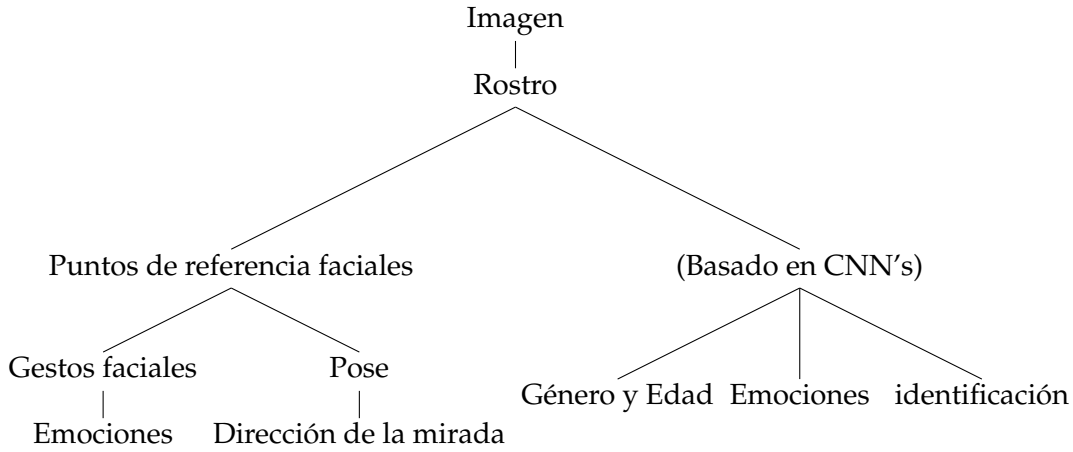

Figura 3.6: Grafo de interdependencia entre las posibles mediciones.

3.3.3. Filtrado y predicción de las mediciones

Las mediciones de naturaleza cuantitativa, en concreto los puntos de referencia faciales, los gestos faciales, la pose de la cabeza y la dirección de la mirada, pueden resultar, en mayor o menor medida, algo inestables en el tiempo. Por ello, una vez estimadas todas las mediciones, se propone su posterior filtrado mediante filtros de Kalman, expuestos en el capítulo 2. Por ello se ha propuesto el uso de la clase `KalmanFilter` de la librería `OpenCV` con el fin de "suavizar" las mediciones obtenidas.

El filtro de Kalman es un algoritmo diseñado por Rudolf E. Kalman en 1960 [44]. Todo tipo de sensor, como por ejemplo un sensor visual, es decir, una cámara, presenta errores o ruido en las mediciones que efectúa. Por lo que este algoritmo nace como la extensión de un modelo matemático recursivo de predicción del estado oculto de un sistema dinámico lineal a tiempo real haciendo uso de las mediciones efectuadas por el sensor.

Para calcular el futuro estado del sistema el filtro de Kalman hace uso de las ecuaciones de movimiento de Newton, las cuales se encuentran definidas en la ecuación 3.2 donde x es el futuro estado de la variable, x_0 es el estado inicial de dicha variable, v_0 es la velocidad inicial y Δt es el intervalo de tiempo.

$$x = x_0 + v_0 \Delta t + \frac{1}{2} \Delta t^2 \tag{3.2}$$

De esta forma, el estado interno del filtro viene dado por el vector $[x, v_x, a_x]$ en el caso de una sola variable respuesta. El cual se actualizará de forma recursiva mediante dos etapas principales.

En la etapa de predicción el algoritmo realiza una estimación a priori del futuro estado del sistema y la covarianza de error asociada a éste. Las ecuaciones 3.3 y 3.4 definen esta etapa, donde $\hat{x}_k$ es la estimación *a priori*, Φ_k la matriz de transición y P_k la covarianza del error asociado a $\hat{x}_k$

$$\hat{x}_k = \Phi_k x_{k-1} \tag{3.3}$$

$$P_k = \Phi_k P_{k-1} \Phi_k^T \tag{3.4}$$

En la etapa de corrección el algoritmo actualiza sus parámetros internos mediante la aportación de una medición por parte del sensor. Esta etapa es definida por las ecuaciones 3.5, 3.6, 3.7 y 3.8 donde $\tilde{y}_k$ es la actualización del residuo de la medición, K_k es la ganancia de Kalman, $\hat{x}_{k-post}$ es la estimación *a posteriori*, P_{k-post} es la covarianza del error asociado a $\hat{x}_k - post$, z_k es la medición realizada en el momento k, H_k es la matriz de relación entre el estado y las mediciones y, por último, R_k es la matriz de covarianza de error asociado a z_k.

$$\tilde{y}_k = z_k - H_k \hat{x}_k \tag{3.5}$$

$$K_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} \tag{3.6}$$

$$\hat{x}_k - post = \hat{x}_k + K_k \tilde{y}_k \tag{3.7}$$

$$P_k = (I - K_k H_k) P_k \tag{3.8}$$

En la figura 3.7 se puede apreciar el funcionamiento del proceso, donde *Measurement* es la medición efectuada, *Predicted state estimate* es la predicción efectuada por el filtro y *Optimal state estimate* es el estado final tras la etapa de correción.

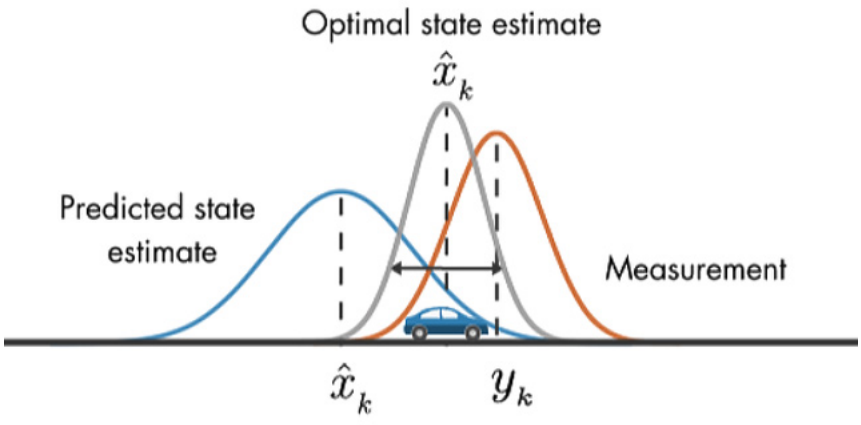

Figura 3.7: Representación gráfica del funcionamiento del filtro de kalman [47].

De esta forma, se propone la creación de un módulo (`stabilizer`), el cual aplica el filtro de Kalman para mediciones de una o dos dimensiones, dependiendo de la dimensionalidad de la medición. Posteriormente, se propone la creación de un módulo específico para cada una de las mediciones cuantitativas a estabilizar (*landmarks*, pose, gestosy dirección de la mirada).

Por último, cabe destacar que, al igual que un GPS es capaz de predecir la posición ante pérdidas de conexión, como por ejemplo cuando se pierde la señal bajo un túnel, los filtros de Kalman pueden predecir el siguiente estado en ausencia de mediciones en un instante concreto. Aprovechando esta característica, se propone utilizar los estabilizadores para predecir el siguiente estado durante una cantidad máxima de *frames* previamente establecida. De esta forma se mejora la robustez del sistema ante fallos de detección debidos a diferentes perturbaciones, como por ejemplo la oclusión parcial o completa del rostro.

3.3.4. Carga, guardado y filtrado de las mediciones

Como último punto de la solución, se propone la creación de un módulo de carga, guardado y filtrado de las mediciones. Dicho módulo, definido como `Recorder`, será el encargado de recibir las mediciones monitorizadas en cada *frame* y añadirlas a una estructura de datos de tipo diccionario. Para la carga y guardado de las mediciones se propone el uso de la librería `pickle` [36].

`pickle` es una librería estándar de `Python` para la carga y guardado de estructuras de datos propias del lenguaje. Por ello, se hará uso de ésta para la carga y guardado en disco de objetos `Python` como la lista de mediciones monitorizadas o la base de datos de codificaciones faciales.

Finalmente, el módulo en cuestión tendrá implementada una función `search` para el filtrado de los datos. Por ejemplo, supongamos que deseamos hallar los *frames* donde aparece una persona específica con los ojos cerrados y mirando hacia la derecha. En ese caso bastaría con, tras analizar el vídeo con las mediciones en cuestión, filtrar dicho vídeo mediante la función `search` y los parámetros específicos.

CAPÍTULO 4
Diseño de la solución

Tras el análisis realizado en el anterior capítulo y, antes del desarrollo del sistema, se ha de definir en qué consistirá su diseño. En este capítulo se abordará esta cuestión, definiendo cual será la estructura del sistema, sus partes, y qué arquitectura se ha propuesta para el posterior desarrollo de éste.

4.1 Arquitectura del sistema

En primer lugar, tal y como se ha descrito anteriormente, se propone dividir el sistema en distintos módulos según la funcionalidad de éstos, es decir, la tarea a la que están destinados. Los distintos módulos pertenecientes al sistema se describen en la tabla 4.1.

Módulo	**Función**
Camera	Extraer *frames* a partir de un sensor o vídeo de entrada
Camera Calibration	Estimar la matriz de parámetros intrínsecos de la cámara
Face Detector	Detectar rostros a partir de una imagen de entrada
Landmarks Predictor	Extraer puntos de referencia faciales dado un rostro de entrada
Pose Estimator	Estimar la pose de la cabeza mediante los *landmarks*
Gaze Estimator	Estimar la dirección de la mirada dado un rostro, sus *landarmks* y la pose de la cabeza como entrada
Action Unit Detector	Estimar diferentes gestos faciales mediante los *landmarks*
Emotion Detector AU	Estimar emociones mediante los gestos faciales (*Action Units*)
Emotion Detector CNN	Estimar emociones dado un rostro mediante CNN
Age Gender Detector	Estimar edad y género mediante un rostro y CNN's
Face Recognition	Reconocer, guardar y comparar rostros mediante CNN
Stabilizer	Estabilizar puntos de 1 ó 2 dimensiones (clase padre de los estabilizadores)
Landmarks Stabilizer	Estabilizar los *landmarks* y predecir su futuro estado
Gaze Stabilizer	Estabilizar la estimación de la dirección de la mirada y predecir su futuro estado
Pose Stabilizer	Estabilizar la estimación de la pose de la cabeza y predecir su futuro estado
Action Unit Stabilizer	Estabilizar la estimación de los gestos faciales y predecir su futuro estado.
Recorder	Guardar, cargar y filtrar todas la mediciones estimadas

Tabla 4.1: Módulos del sistema y sus respectivas funcionalidades.

Como se ha expuesto en el anterior capítulo, los módulos anteriormente definidos presentan complejas interdependencias entre sí. Por ello, se ha optado por un diseño basado en la descomposición modular de control centralizado o arquitectura gestor. El módulo gestor, definido como `Face Monitoring`, será el encargado de resolver todas las interdependencias entre los anteriores módulos y propagar la información necesaria de un módulo a otro. Este tipo de arquitectura presenta dos ventajas principales:

1. **Eficiencia**: Dado que el módulo gestor resuelve todas las interdependencias, se efectuarán las mediciones sólo estrictamente necesarias para una determinada tarea. Por ejemplo, si nuestro único objetivo es el de monitorizar la edad y el género detectados, el módulo gestor, en función del árbol de interdependencias mostrado en el anterior capítulo, realizará las estimaciones estrictamente necesarias para dicho objetivo, en este caso únicamente la detección de la región de interés del rostro.

2. **Sencillez**: Al tratarse de una arquitectura de control centralizado, el posible usuario que haga uso del sistema será capaz de realizar múltiples mediciones sin necesidad de conocer todo el proceso que hay detrás. De esta forma, tras extraer una imagen, el usuario solamente deberá ejecutar una función (`analyze`) con los parámetros relativos a las mediciones que desea estimar. Por ejemplo, si se desea estimar la posición de la cabeza y las emociones, el usuario de ejecutar la función `analyze(['pose','emotion'])`.

En la figura 4.1 se puede apreciar el diagrama de módulos propuesto bajo la arquitectura de control centralizado. En ella se puede apreciar como el módulo gestor `Face Monitoring` está vinculado a un total de 17 submódulos donde cada uno de éstos será el encargado de un subproblema del sistema de monitorización, tal y como se detallará en el punto 4.2.

Además, cabe destacar que cada uno de los cuatro módulos que podemos ver en la parte superior izquierda de la figura 4.1 y encargados de la estabilización, está compuesta de un conjunto de estabilizadores implementados en el módulo `Stabilizer`.

La parte superior derecha muestra los tres módulos encargados de realizar estimaciones por medio de redes neuronales convolucionales (CNN's). Dichos modelos se cargan directamente del disco y se encuentran en una carpeta llamada *Models*, tal y como se aprecia en la figura. En el caso del módulo *Face Recognition*, éste almacena y carga las codificaciones faciales estimadas a través del archivo `Faces Dataset`.

Por último, en el caso del módulo `Camera`, encargado de la extracción de *frames*, podemos observar en la figura que es capaz de extraer dichos *frames* tanto a partir del sensor como de un fichero de vídeo.

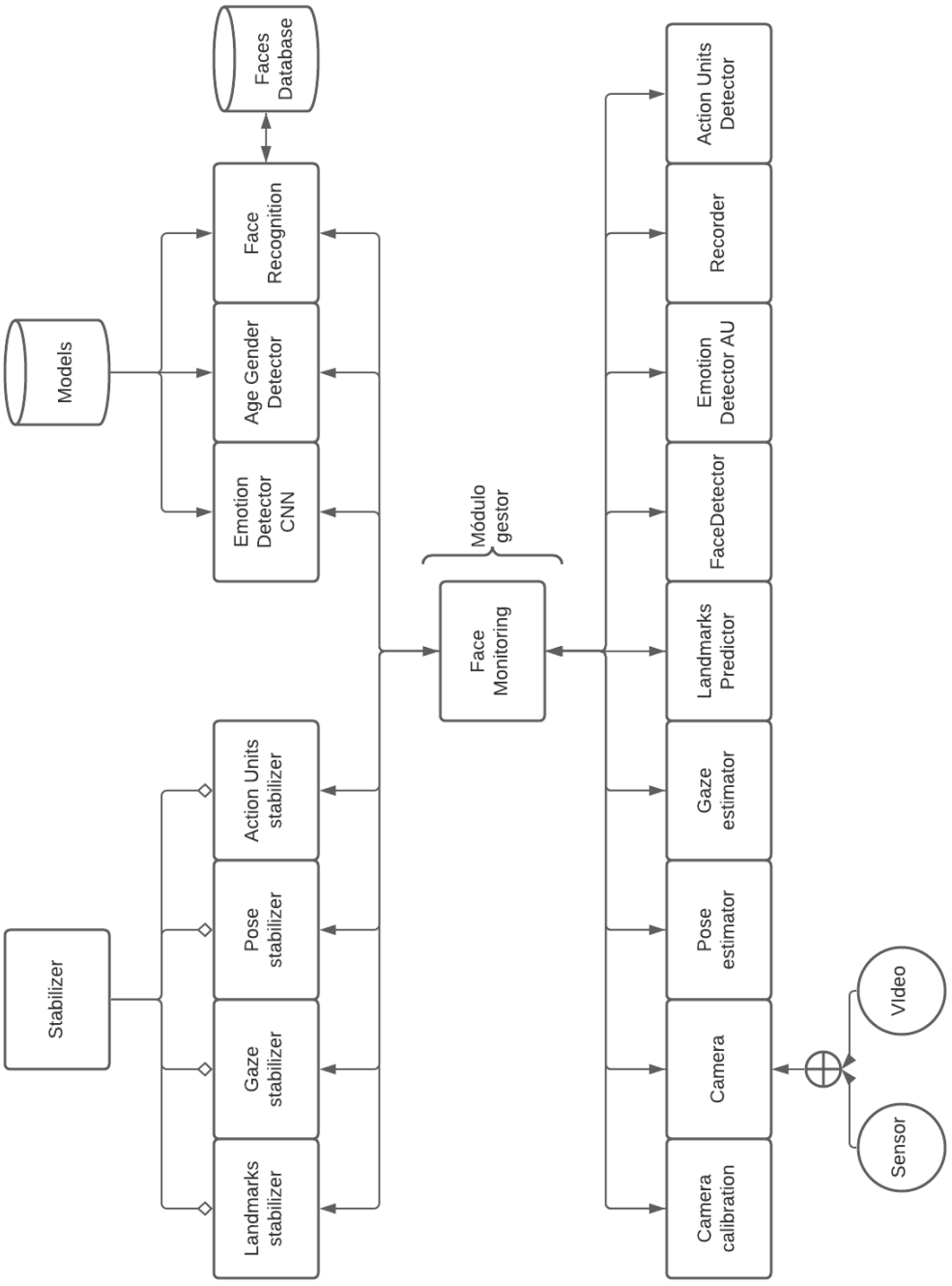

Figura 4.1: Diagrama de módulos del sistema de monitorización facial.

4.2 Diseño detallado

Una vez definida la arquitectura del sistema, es preciso definir qué procesos debe realizar cada uno de los módulos. En esta sección se realizará un análisis algo más detallado sobre los atributos, parámetros y funciones necesarios para el entendimiento del funcionamiento interno del sistema, a grandes rasgos, para cada uno de los módulos descritos con anterioridad, cuyo funcionamiento será ampliamente detallado en el capítulo 5.

4.2.1. `Camera`

`Camera` es el módulo encargado de la extracción de *frames* o imágenes a partir de un sensor o fichero de vídeo. Sus atributos y funciones están definidos en la tabla 4.2.

Atributo	**Descripción**
`size`	tamaño de escalado de los *frames* extraídos.
`videoIn`	ruta del fichero de vídeo (`None` si se usa un sensor)
`cameras-list`	lista de sensores disponibles en el dispositivo
Función	**Descripción**
`select-camera`	selecciona un sensor entre los disponibles en el equipo
`get-frame`	extrae un *frame* del sensor o el vídeo de entrada

Tabla 4.2: Atributos y funciones del módulo `Camera`.

4.2.2. `Camera Calibration`

Se trata del módulo encargado de estimar la matriz de parámetros intrínsecos de la cámara o vídeo. Sus atributos y funciones están definidos en la tabla 4.3.

Atributo	**Descripción**
`K`	matriz de parámetros intrínsecos
Función	**Descripción**
`EstimateCamMatrix`	estima la matriz de parámetros intrínsecos a partir del módulo `Camera`

Tabla 4.3: Atributos y funciones del módulo `Camera Calibration`.

4.2.3. `Face Detector`

`Face Detector` es el módulo encargado de la detección de rostros. Sus atributos y funciones están definidos en la tabla 4.4.

Atributo	**Descripción**
`opt`	si `opt` es `True` aplica heurísticas para su uso en aplicaciones de tiempo real
`detector-mode`	modelo de detección (HOG+SVM, CNN `Dlib`, CNN `cvlib`)
Función	**Descripción**
`getFaces`	detecta los rostros dada una imagen de entrada
`getLargestFace`	detecta el rostro más próximo al sensor, es decir, el más grande

Tabla 4.4: Atributos y funciones del módulo `Face Detector`.

4.2.4. `stabilizer`

`stabilizer` es el módulo encargado de filtrar y predecir mediciones de 1 o 2 dimensiones mediante el uso del filtro de Kalman. Es el "padre" de los estabilizadores de las mediciones. Sus atributos y funciones están definidos en la tabla 4.5.

Atributo	**Descripción**
`state-num`	número de dimensiones del estado interno
`measure-num`	número de dimensiones de la medición
`cov-process`	covarianza asociada a la predición
`cov-measure`	covarianza asociada a la medición
Función	**Descripción**
`set-q-r`	actualiza las matrices de covarianza asociadas a la medición y a la predicción
`update`	actualiza el estado interno en el caso de recibir una medición, en caso contrario predice el siguiente estado

Tabla 4.5: Atributos y funciones del módulo `stabilizer`.

4.2.5. `Landmarks predictor`

Su función es la estimar el conjunto de 68 *landmarks* dado un rostro a partir del modelo implementado en `Dlib`. Sus atributos y funciones están definidos en la tabla 4.6.

Atributo	**Descripción**
`opt`	si `opt` es `True` aplica heurísticas para su uso en aplicaciones de tiempo real
Función	**Descripción**
`getFaceLandmarks`	estima el conjunto de *landmarks* dada la región de interés del rostro detectado
`drawLandmarks`	dibuja los *landmarks* sobre una imagen dada

Tabla 4.6: Atributos y funciones del módulo `Landmarks predictor`.

4.2.6. `Landmarks stabilizer`

Se trata del módulo encargado de estabilizar los *landmarks* estimados haciendo uso de su módulo "padre" stabilizer, descrito anteriormente. Sus atributos y funciones están definidos en la tabla 4.7.

Atributo	**Descripción**
`idxs`	lista de los *landmarks* a estabilizar. Por defecto será el subconjunto necesario para la estimación de la pose
`stabilizers`	conjunto de estabilizadores necesarios para filtrar los *landmarks*
Función	**Descripción**
`update`	actualiza el estado interno en el caso de recibir una medición, en caso contrario predice el siguiente estado
`drawSteadyLandmarks`	dibuja los *landmarks* estabilizados sobre una imagen dada

Tabla 4.7: Atributos y funciones del módulo `Landmarks stabilizer`.

4.2.7. `Pose Estimator`

`Pose Estimator` es el módulo encargado de estimar la posición y rotación de la cabeza en base a los *landmarks* extraídos. Sus atributos y funciones están definidos en la tabla 4.8.

Atributo	**Descripción**
`cam-calibration`	módulo de calibración de cámara necesario para la estimación de la pose
`model-points`	puntos tridimensionales del modelo de cabeza
`mp-idx`	lista de índices de los *landmarks* asociados a los puntos del modelo tridimensional
Función	**Descripción**
`getPose`	estima la rotación y traslación de la cabeza mediante los *landmarks* extraídos
`rotvec2euler`	convierte el vector rotación a vector de euler en grados
`draw2Dbox`	dibuja la caja tridimensional que contiene a la cabeza

Tabla 4.8: Atributos y funciones del módulo `Pose Estimator`.

4.2.8. `Pose Stabilizer`

Módulo encargado de estabilizar la pose estimada. Sus atributos y funciones están definidos en la tabla 4.9.

Atributo	**Descripción**
`stabilizers`	conjunto de 6 estabilizadores necesarios para filtrar la pose
Funciones	**Descripción**
`update`	actualiza el estado interno en el caso de recibir una medición, en caso contrario predice el siguiente estado

Tabla 4.9: Atributos y funciones del módulo `Pose Stabilizer`.

4.2.9. `Gaze Estimator`

`Gaze Estimator` es el módulo encargado de estimar la dirección de la mirada en base a los *landmarks* extraídos y la región de interés del rostro. Sus atributos y funciones están definidos en la tabla 4.10.

Atributos	**Descripción**
`left-eye-idxs`	índices asociados a los *landmarks* relativos al ojo izquierdo
`right-eye-idxs`	índices asociados a los *landmarks* relativos al ojo derecho
`threshold-left`	párametro *threshold* para el procesamiento de la región de interés del ojo izquierdo
`threshold-right`	párametro *threshold* para el procesamiento de la región de interés del ojo derecho
Función	**Descripción**
`contouring`	detecta la posición de la pupila mediante la región de interés del ojo previamente procesada
`get-eye-gaze`	procesa las regiones de interés del ojo y detecta la posición de las pupilas
`draw-eye-rays`	dibuja la dirección de la mirada sobre el rostro

Tabla 4.10: Atributos y funciones del módulo `Gaze Estimator`.

4.2.10. `Gaze Stabilizer`

`Gaze Stabilizer` es el módulo encargado de estabilizar la dirección de la mirada previamente estimada. Sus atributos y funciones están definidos en la tabla 4.11.

Atributos	**Descripción**
`stabilizers`	conjunto de estabilizadores necesarios para filtrar la dirección de la mirada
Función	**Descripción**
`update`	actualiza el estado interno en el caso de recibir una medición, en caso contrario predice el siguiente estado

Tabla 4.11: Atributos y funciones del módulo `Gaze Stabilizer`.

4.2.11. `Action Unit Detector`

`Action Unit Detector` es el módulo encargado de estimar los gestos faciales mediante los *landmarks* extraídos previamente. Sus atributos y funciones están definidos en la tabla 4.12.

Atributo	**Descripción**
`mor-range`	rango utilizado para normalizar el ratio de estimación de apertura de boca
`msr-range`	rango utilizado para normalizar el ratio de separación de las comisuras de la boca
`mkr-range`	rango utilizado para normalizar el ratio de acercamiento entre las comisuras de la boca
`ear-range`	rango utilizado para normalizar el ratio de estimación del pestañeo
`ebr-range`	rango utilizado para normalizar el ratio de estimación del movimiento vertical de las cejas
Función	**Descripción**
`ratio2range`	normaliza los ratios estimados relativos a los gestos faciales
`get-mor`	estimación de apertura de boca
`get-msr`	estimación de la separación entre las comisuras de la boca
`get-mkr`	estimación del acercamiento entre las comisuras de la boca
`get-ear`	estimación del pestañeo
`ebr-range`	estimación del movimiento vertical de las cejas
`get-all-au`	estimación todos los gestos previamente mencionados

Tabla 4.12: Atributos y funciones del módulo `Action Unit Detector`.

4.2.12. `Aciton Unit Stabilizers`

El módulo `Aciton Unit Stabilizers` es el encargado de estabilizar las mediciones relativas a los gestos faciales. Sus atributos y funciones están definidos en la tabla 4.13.

Atributo	**Descripción**
`stabilizers`	estabilizadores necesarios para filtrar los gestos faciales estimados
Función	**Descripción**
`update`	actualiza el estado interno en el caso de recibir una medición, en caso contrario predice el siguiente estado

Tabla 4.13: Atributos y funciones del módulo `Action Unit Stabilizers`.

4.2.13. `Emotion Detector AU`

`Emotion Detector AU` es el módulo encargado de la estimación de emociones mediante los gestos faciales previamente estiamdos por el módulo `Action Unit Detector`. Sus funciones están definidos en la tabla 4.14.

Función	Descripción
`happy`	detección de felicidad
`angry`	detección de enfado
`evilness`	detección de maldad
`yawning`	detección de bostezo/cansancio
`suprised`	detección de sorpresa
`get-emotion`	detección de emociones

Tabla 4.14: Funciones del módulo `Emotion Detector AU`.

4.2.14. `Emotion Detector CNN`

Se trata del módulo encargado de estimar emociones dada la región de interés de un rostro mediante una CNN. Sus atributos y funciones están definidos en la tabla 4.15.

Atributo	Descripción
`model`	CNN encargada de la estimación de emociones
Función	**Descripción**
`get-emotion`	estima las emociones mediante su modelo de CNN

Tabla 4.15: Atributos y funciones del módulo `Emotion Detector CNN`.

4.2.15. `Age Gender Detector`

`Age Gender Detector` es el módulo encargado de estimar la edad y el género dada la región de interés de un rostro mediante el uso de CNN's. Sus atributos y funciones están definidos en la tabla 4.17.

Atributo	Descripción
`gender-model`	CNN encargada de las estimación del género
`age-model`	CNN encargada de las estimación de la edad
`onlyOnce`	si `onlyOnce` es `True` se realizará la estimación una única vez
`ageList`	lista de intervalos de edad que la red es capaz de estimar
Función	**Descripción**
`detect`	estima la edad y el género dad la región de interés como entrada

Tabla 4.16: Atributos y funciones del módulo `Age Gender Detector`.

4.2.16. `Face Recognition`

`Face Recognition` es el módulo encargado del identificación, codificación y comparación de rostros mediante el uso de la librería `face-recognition`. Sus atributos y funciones están definidos en la tabla 4.12.

Atributo	**Descripción**
`onlyOnce`	si `onlyOnce` es `True` la codificación facial se realizará una sola vez
`face-database`	base de datos de codificaciones faciales con sus respectivos nombres
Función	**Descripción**
`detect`	realiza la codificación facial dada la región de interés de un rostro
`compare`	compara dos codificaciones como la distancia euclídea entre éstas.
`add-face`	añade la codificación con su respectivo nombre a la lista de codificaciones
`save`	guarda las nuevas codificaciones en la base de datos
`get-face-name`	devuelve el nombre de la persona reconocida dada la codificación del rostro, en caso de no encontrarla, devuelve `None`

Tabla 4.17: Atributos y funciones del módulo `Face Recognition`.

4.2.17. `Recorder`

`Recorder` es el módulo encargado recopilar, guardar, cargar y filtrar todas las mediciones efectuadas mediante los módulos anteriores. Sus atributos y funciones están definidos en la tabla 4.18.

Atributo	**Descripción**
`record`	lista de *frames* recopilados con sus respectivas mediciones
Función	**Descripción**
`add-frame`	añade un *frame* con sus respectivas mediciones a la lista `record`
`save-record`	guarda la lista de *frames* analizados en un ruta determinada
`load-record`	carga la lista de *frames* analizados, previamente guardada en un ruta determinada
`search`	busca en la lista de *frames* analizados según una serie de condiciones establecidas por el usuario

Tabla 4.18: Atributos y funciones del módulo `Recorder`.

4.2.18. `Face Monitoring`

Por último, tal y como se ha expuesto anteriormente, `Face Monitoring` es el módulo gestor encargado de resolver todas las interdependencias entre los módulos de medición con el fin de obtener un sistema de monitorización eficiente. Sus atributos y funciones están definidos en la tabla 4.19.

Atributo	**Descripción**
`size`	tamaño de escalado de *frames*
`videoIn`	ruta de fichero de vídeo a analizar. En caso de que sea `None` se hará uso del sensor del dispositivo
`modules`	16 módulos de medición de parámetros faciales (los expuestos anteriormente)
`detector`	detector facial utilizado (HOG+SVM, CNN `Dlib`, CNN `cvlib`)
`max-frames-without-face`	máximo número de *frames* en los que se utilizan los estabilizadores para la predicción del siguiente estado en ausencia de mediciones
Función	**Descripción**
`get-frame`	extrae un nuevo *frame* mediante el sensor o fichero de vídeo
`analyze`	realiza las mediciones determinadas por el usuario resolviendo todas las interdependencias entre los módulos de medición y las guarda en el `Recorder`, si así lo estipula el usuario.

Tabla 4.19: Atributos y funciones del módulo `Face Monitoring`.

CAPÍTULO 5
Desarrollo de la solución propuesta

Una vez definida la arquitectura del sistema, en este capítulo se abordará todo el proceso de desarrollo del éste, detallando las fases de su construcción y las peculiaridades de la solución final.

Con el fin de facilitar de la comprensión del proceso, el orden seguido será el asociado al árbol de interdependencias expuesto anteriormente (3.6), comenzado por la raíz y descendiendo hacia las hojas (DFS). De esta forma el orden de exposición del proceso será el siguiente.

1. Estabilización y predicción de las mediciones
2. Extracción de *frames*
3. Detección facial
4. Extracción de *landmarks*
5. Estimación de la pose
6. Estimación de la dirección de la mirada
7. Estimación de gestos faciales
8. Detección de emociones mediante gestos faciales
9. Detección de emociones mediante CNN
10. Detección de género y edad mediante CNN's
11. Identificación facial mediante CNN

Cabe destacar que únicamente se detallarán los procesos, métodos o herramientas utilizadas más relevantes para la comprensión del funcionamiento del sistema y sus peculiaridades. En la figura 5.1 se puede observar el diagrama de procesos del sistema donde a partir de la detección del rostro se estiman los demás parámetros faciales.

Posteriormente a la exposición de los procesos de desarrollo anteriormente enumerados se realizará una explicación del desarrollo del módulo gestor para, finalmente, abarcar el desarrollo de una interfaz gráfica demostrativa para la visualización de los parámetros faciales monitorizados a tiempo real mediante el uso de la *webcam* del dispositivo.

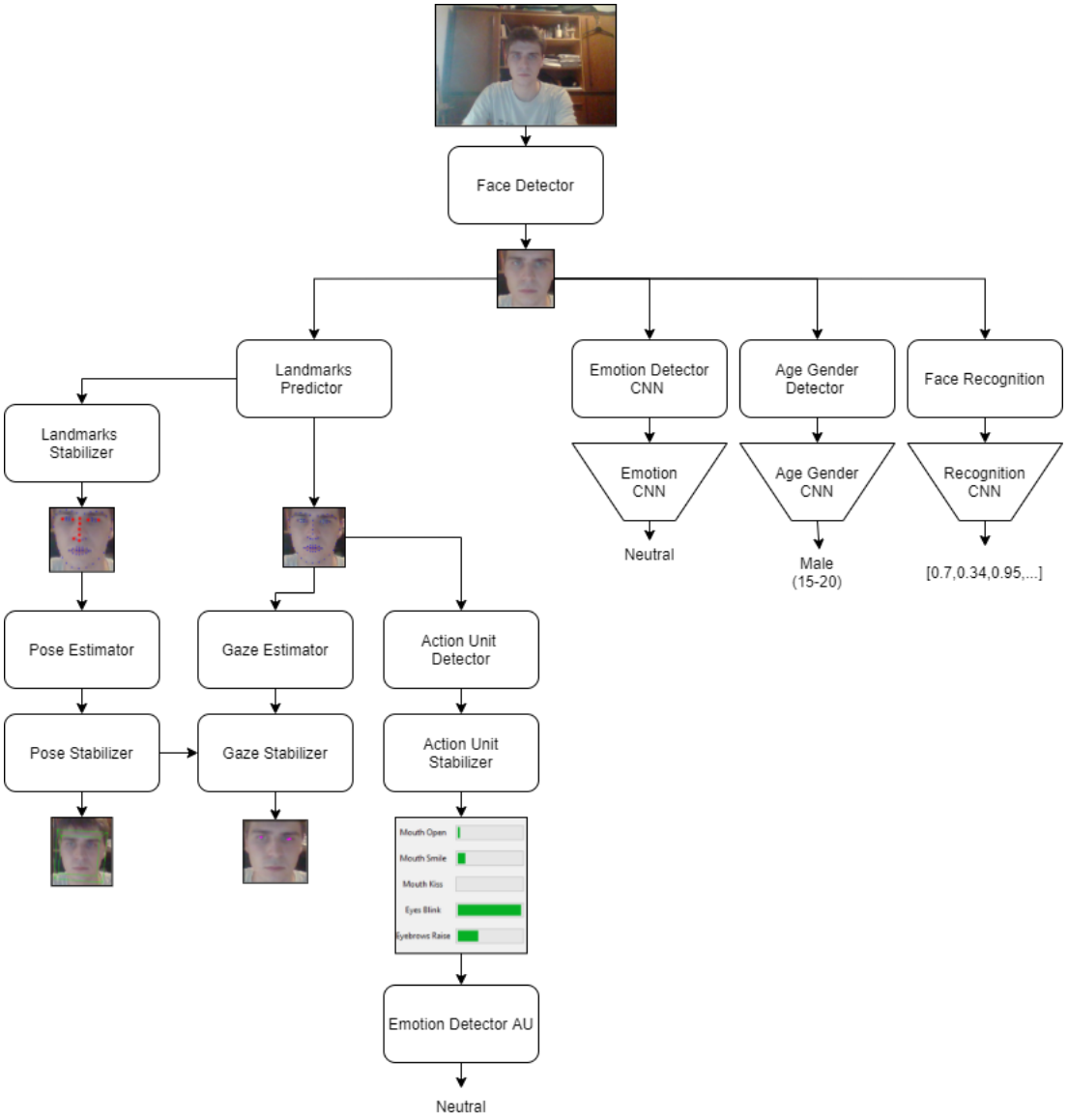

Figura 5.1: Diagrama de procesos del sistema.

5.1 Estabilización y predicción de las mediciones

Con el fin de facilitar la comprensión del desarrollo del sistema, comenzaremos definiendo cómo se ha desarrollado la estabilización de los *landmarks*, la pose, los gestos faciales y la dirección de la mirada. De forma que cada vez que se haga referencia a la estabilización de alguna de estas mediciones se presupone que existe un módulo de estabilización específico para cada una de éstas, el cual contiene sus respectivos `stabilizers` tal y como se muestra en la tabla 5.1.

Landmarks	Un `stabilizer` de dos dimensiones por *landmark* a estabilizar
Pose	6 `stabilizers` de una dimensión
Gestos faciales	Un `stabilizer` de una dimensión por gesto estimado
Dirección de la mirada	2 `stabilizers` de una dimensión

Tabla 5.1: Número de estabilizadores para cada medición cuantitativa.

Cada `stabilizer` puede ser de una o dos dimensiones en función del parámetro `meauser-num`. Así mismo, contiene un filtro de kalman de la librería `OpenCV` (`cv2.KalmanFilter`) capaz de realizar predicciones y correcciones del estado de interno.

La función implementada `update` actualiza el estado interno en base a la medición pasada como parámetro. Para ello, en primer lugar se realiza la predicción mediante la función `cv2.KalmanFilter.predict()` y, en caso de que el parámetro de la medición (`measurement`) sea distinto a `None`, se procede a la corrección de la predicción mediante la función `cv2.kalmanFilter.correct(measurement)`. De esta forma, en caso de no haber podido realizar la medición por algún problema en la detección, el estabilizador será capaz de predecir el siguiente estado.

5.2 Extracción de *frames*

El primer paso que debe tomar el sistema para posteriormente estimar todas las mediciones es la extracción de *frames* mediante sensor o fichero de vídeo. Para ello, el módulo `Camera` hace uso de la clase `VideoCapture` implementada en `OpenCV`. La cual permite, tal y como se especifica en los requisitos de la aplicación, extraer *frames* tanto de un sensor integrado en el dispositivo como de un fichero de vídeo. Para especificar la resolución del *frame* extraído se hace uso de las funciones `set(cv2.CAP-PROP-FRAME-WIDTH, width)` y `set(cv2.CAP-PROP-FRAME-HEIGHT, height)`.

Estimación de la matriz de parámetros intrínsecos de la cámara

Una vez escogido el sensor o fichero de vídeo podemos estimar la matriz de parámetros intrínsecos de la cámara, la cual nos será especialmente útil para la posterior estimación de la pose de la cabeza y la proyección de puntos tridimensionales sobre el plano imagen.

Para ello, se ha implementado una función el módulo `Cam Calibration` definida como `EstimateCamMatrix` que estima dicha matriz a partir de la resolución de la cámara.

5.3 Detección facial

Una vez extraído el *frame*, el siguiente paso a seguir es la detección de rostros en éste. Para ello, tal y como ha sido expuesto en la solución propuesta (3.3). Se han implementado tres algoritmos diferentes dicha tarea. De forma que se pueda seleccionar el algoritmo para la detección mediante el parámetro `detector-mode`, como se puede observar en la tabla 5.2.

`detector-mode`	**Método**	**Librería**
0	HOG+SVM	`Dlib`
1	CNN	`Dlib`
2	CNN	`cvlib`

Tabla 5.2: Modos de detección facial.

En el caso del segundo método (CNN `Dlib`), no se recomienda su uso en CPU ya que el tiempo de cómputo supone una frecuencia de detección de menos de 1 FPS. Además, cabe destacar que cada uno de estos modelos de detección presenta su propio formato de salida. Por ello, con el fin de unificar los modelos en un sólo módulo, tras la detección, la

salida del detector se convierte a formato `NumPy`, es decir, a un vector. Dicho vector estará compuesto compuesto de cuatro componentes en la forma $[x_1, x_2, y_1, y_2]$. De forma que el recuadro que enmarca el rostro tiene su esquina superior izquierda e inferior derecha en (x_1, y_1) y (x_2, y_2) respectivamente, tal y como se puede apreciar en la figura 5.2.

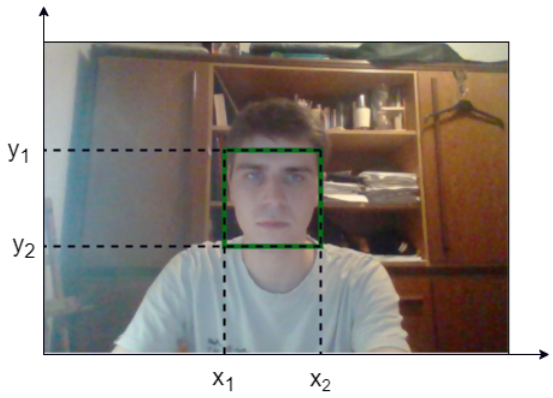

Figura 5.2: Visualización de la detección facial.

Finalmente, para extraer el rostro más cercano al sensor, simplemente se ha de computar el área de cada recuadro detectado y devolver el máximo mediante la función `max(faces, key = lambda x0 , x1 , y0 , y1: (x1-x0)*(y1-y0))`.

5.3.1. Optimización en la detección

Dado que uno de los principales objetivos es que el sistema realice las mediciones en tiempo real, se han implementado una serie de optimizaciones inspiradas en el artículo "*Speeding up Dlib's Facial Landmark Detector*" [29]. Las optimizaciones realizadas son las siguientes:

1. `Gray`: se convierte el *frame* de entrada a escala de grises.

2. `Resize`: se escala el *frame* de forma que la altura de este sea de 500 píxeles.

3. `Skip frames`: dado que la típica *webcam* es capaz de grabar a 30 FPS, normalmente rostro no se moverá significativamente entre un *frame* y otro (equivale a 30 ms). Por lo tanto, no hay necesidad de detectar el rostro en cada uno de los *frames*. Por ello, la detección de rostros se realizará cada 3 *frames* mediante esta optimización.

Cabe destacar que los parámetros relativos a estas optimizaciones pueden ser cambiados y/o eliminados.

Tras realizar un análisis práctico mediante la herramienta `Google Colab` sobre la eficacia de estas optimizaciones podemos concluir que resultan altamente efectivas, obteniendo una mejora de FPS del 752 % y, como se verá en el producto final, no suponen una pérdida de robustez del sistema, dado que el porcentaje de detección ha sido del 100 % en todos los casos. Además cabe destacar que en la figura 5.3, MIX es todas las optimizaciones al mismo tiempo) y que éste análisis se ha realizado extrayendo *frames* de un fichero de vídeo, por lo que los FPS estimados son teóricos pero nos sirven de referencia para comprobar la eficacia de las optimizaciones realizadas. .

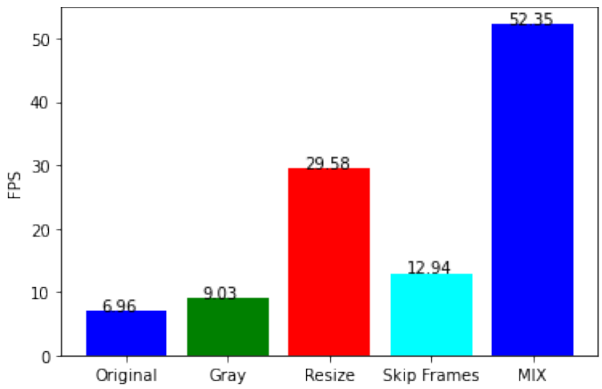

Figura 5.3: FPS medio de procesamiento mediante optimizaciones.

5.3.2. Extracción de la región de interés

Una vez conocidas las coordenadas del recuadro que enmarca al rostro $[x_1, x_2, y_1, y_2]$ y el *frame* extraído, resulta muy sencillo extraer de éste la región de interés (ROI). Simplemente se ha recortar el *frame* de entrada mediante el método $frame[y_1 : y_2, x_1 : x_2]$, tal y como se puede apreciar en la figura 5.4.

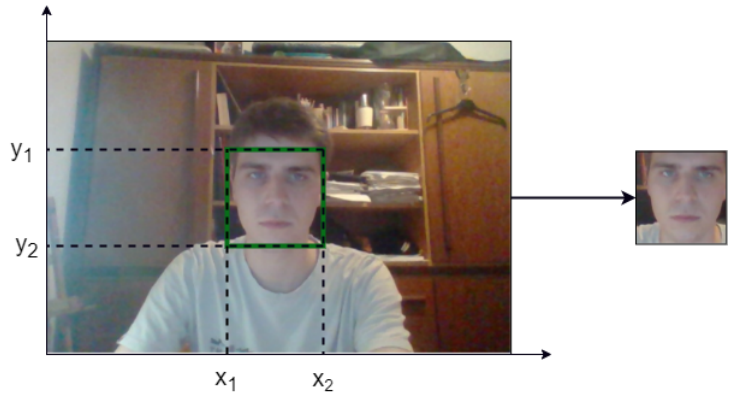

Figura 5.4: Visualización de la extracción de la región de interés.

5.4 Extracción de *landmarks*

Una vez extraída la región de interés del rostro se procede a la extracción de los 68 *landmarks* mediante el modelo preentrenado de `Dlib` ("`shape-predictor-68-face-landmarks.dat`"). Para ello, en primer lugar se convierte la región de interés del rostros mediante la función `cv2.cvtColor(frame,cv2.COLOR-BGR2GRAY)` y, finalmente se extraen los *landmarks* mediante la función `predictor(frame , face)`.

Una vez extraídos los *landmarks*, al igual que en la detección, éstos se convierten a un vector `NumPy` en la forma $[[x_0, y_0], [x_1, y_1], ..., [x_{67}, y_{67}]]$ para facilitar las futuras operaciones entre éstos, las cuales serán realizadas para estimar las mediciones basadas en *landmarks*, tal y como se puede apreciar en la figura 5.5.

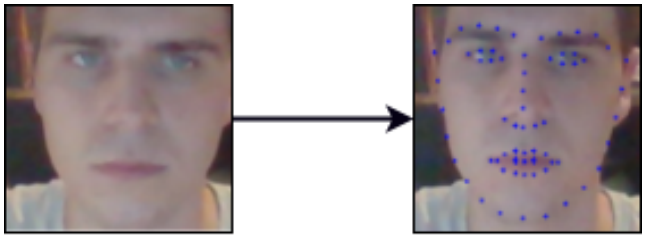

Figura 5.5: Extracción de los 68 *landmarks*.

5.4.1. Estabilización y predicción de los *landmarks*

Estabilización

Una vez extraídos los 68 *landmarks* se procede a la estabilización de éstos. Para ello se hace uso del módulo `Landmarks Stabilizer` previamente descrito, el cual es una extensión de su "módulo padre" `stabilizer`, conteniendo un `stabilizer` de dos dimensiones por *landmark* a estabilizar.

Concretamente, se estabiliza un subconjunto de los *landmarks* ([29, 28, 39, 42, 27, 33, 31, 30, 36, 45]) ya que éstos serán necesarios para la estimación de la pose posteriormente. Para ello se ha implementado la función `LandmarksStabilizer.update(`*`landmarks`*`)`. Sin embargo, es posible configurar el módulo para realizar la estabilización de otros *landmarks*, o incluso todos ellos, mediante su parámetro *idxs* (lista de *landmarks* a estabilizar).

Predicción

En el caso de no haber podido extraer los *landmarks* debido a un fallo en una de las etapas posteriores, en este caso la de detección, se procederá a la predicción del siguiente estado mediante la función `LandmarksStabilizer.update(None)`. De esta forma, ante posibles oclusiones momentáneas y/o fallos en la detección el sistema será capaz de predecir el siguiente estado de los *landmarks* en función de los estados previamente extraídos. En la figura 5.6 se puede apreciar como al pasar la mano delante de rostro y, por tanto, perder la detección, el sistema es capaz de predecir el estado intermedio de los *landmarks* estabilizados.

Figura 5.6: Predicción del subconjunto de *landmarks* utilizados para la estimación de la pose.

5.5 Estimación de la pose de la cabeza

Una vez extraídos y estabilizados los *landmarks* podemos estimar la posición y rotación de la cabeza. La estimación de la pose se realizará mediante la función `solvePnP` de la librería `OpenCV`. La función `solvePnP` cuenta con los siguientes argumentos:

1. ***Model Points***: puntos de referencia faciales tridimensionales. Los cuales han sido extraidos de un modelo de cabeza tridimensional gratuito.
2. ***Image Points***: proyección de los *Model points* sobre el plano imagen. Los cuales son extraidos a partir del extractor de puntos de referencia faciales de expuesto anteriormente.
3. ***Rotation vector***: se trata de la estimación inicial del vector rotación que utilizará el método iterativo para converger a la solución final.
4. ***Translation vector***: se trata de la estimación inicial del vector traslación que utilizará el método iterativo para converger a la solución final.
5. ***Camera matrix***: matriz de parámetros intrínsecos de la cámara.

Puntos de referencia escogidos

En numerosas fuentes [23] los puntos de referencia faciales escogidos para la estimación de la pose son los bordes de los ojos, la punta de la nariz, las comisuras de la boca y la punta de la barbilla, tal y como se puede apreciar en la figura 5.7.

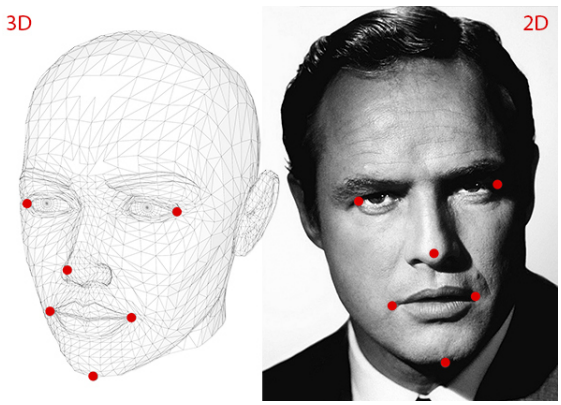

Figura 5.7: *landmarks* propuestos en [23] (puntos rojos) para la estimación de la pose.

Sin embargo, la elección de estos puntos puede suponer un problema a la hora de estimar la pose por dos motivos. El primero de ellos es que el conjunto de los puntos es simétrico respecto al eje vertical. Esto implica que la proyección de los puntos sobre el plano imagen será la misma bajo una rotación de 180 grados sobre el eje vertical, lo cual puede llevar a una estimación errónea de la pose. En segundo lugar, a pesar de que el sujeto no realice ninguna rotación ni traslación, los gestos faciales como abrir la boca, fruncir el ceño o pestañear suponen un movimiento de los puntos de referencia y por lo tanto, influyen en la estimación de la pose.

Como se puede observar en la figura 5.8, al mantener la pose invariante pero realizar un gesto como el de abrir la boca, la estimación de la pose es errónea mediante el subconjuto de *landmarks* propuestos en [23].

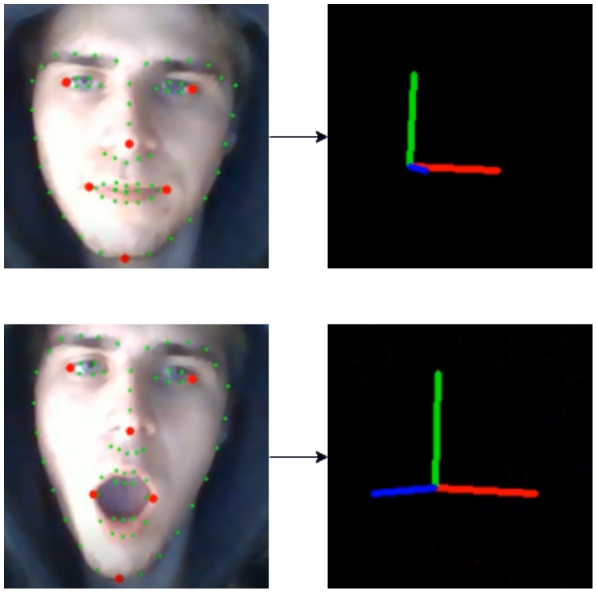

Figura 5.8: Estimación de pose mediante los *landmarks* propuestos en [23] (puntos rojos).

Por ello, para solucionar este problema, se ha realizado un análisis sobre la variabilidad de los puntos de referencia ante diferentes gestos faciales y muecas. El proceso consiste en extraer los puntos de referencia faciales para cada *frame* de un vídeo dónde el sujeto mantiene invariante la posición y rotación de la cabeza al mismo tiempo que realiza diferentes muecas. De esta forma, podemos computar la distancia que recorre cada uno de los 68 puntos de referencia faciales entre un *frame* y el siguiente.

Tras realizar este análisis, podemos observar en la figura 5.9 la variabilidad de los 68 *landmarks* ordenados de menor a mayor (verde a rojo) y su representación visual (5.10) donde dichos colores están representados sobre los *landmarks* detectados. De esta forma podemos concluir que los puntos relativos a la zona de los ojos y la boca son los menos invariantes ante diferentes gestos faciales. Por lo que se han escogido estos puntos para estimación de la pose.

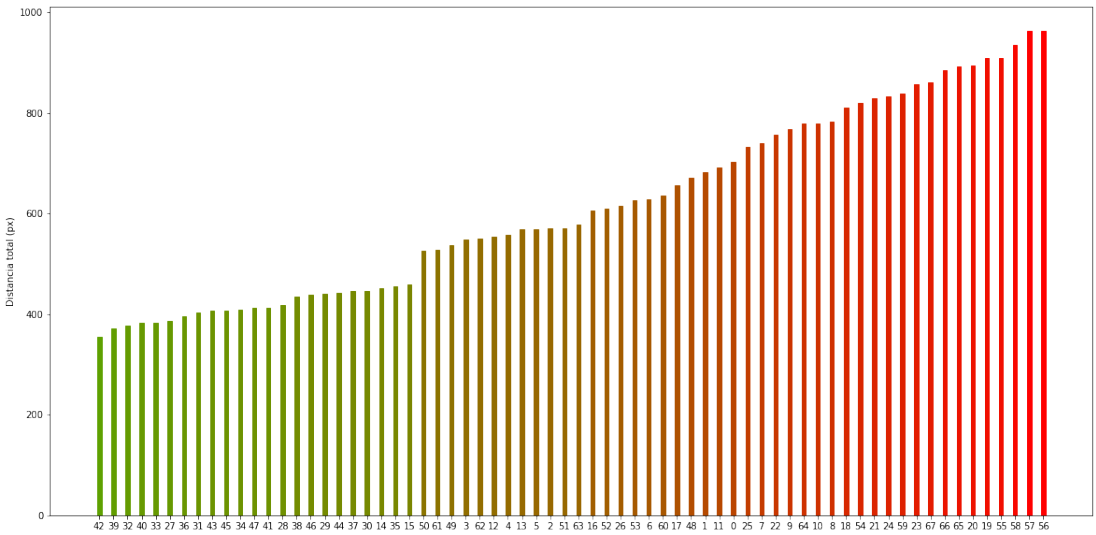

Figura 5.9: Análisis de la variabilidad de los 68 *landmarks* ante gestos faciales (ordenados de mayor a menor).

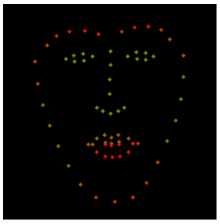

Figura 5.10: Representación visual de la variabilidad de los 68 *landmarks* ante gestos faciales).

Para solucionar el problema de la simetría, se han añadido dos de los puntos de la región inferior de la nariz, véase 5.11, de forma que el conjunto de puntos finales es asimétrico bajo todos los ejes de rotación. En la figura 5.11 se puede observar que el uso de este subconjunto de *landmarks* para la estimación de la pose soluciona el problema de la estimación ante muecas y/o gestos faciales.

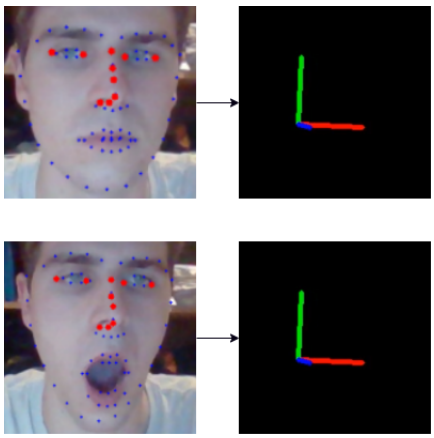

Figura 5.11: Estimación de pose mediante los *landmarks* propuestos en este proyecto.

Por último, otra de las modificaciones que se ha realizado es que, dado que `solvePnP` se resuelve mediante un método iterativo, podemos establecer una posición y rotación iniciales como punto de partida del método, al contrario que en la fuente mencionada donde no se hacía uso de éstos parámetros. De esta forma, cada vez que deseemos estimar la pose, la posición y rotación iniciales desde donde partirá el método iterativo serán las de el anterior *frame*. De esta forma se facilita la convergencia del método y se mejora la robustez del sistema.

5.5.1. Estabilización y predicción de la pose

El estabilizador de la pose (`Pose Stabilizer`) cuenta con un total de seis `stabilizers` de una dimensión para su funcionamiento (tres para la posición y otros tres para la rotación). El funcionamiento es el mismo que el descrito anteriormente para la estabilización de los *landmarks*. Por tanto, al igual que en el anterior caso, se puede tanto estabilizar la pose como estimar su estado en ausencia de *landmarks* detectados.

Cabe destacar que, en este caso, se realiza un doble estabilización ya que primeramente los *landmarks* utilizados para la estimación de la pose son estabilizados y, seguidamente, la propia pose estimada es estabilizada. De esta forma nos aseguramos de que la estimación de la pose no sufra muchas variaciones ante gestos faciales y/o movimientos abruptos de la cabeza. En la figura 5.12 se puede observar como el uso de los `stabilizers` permiten la predicción de la pose ante fallos de detección

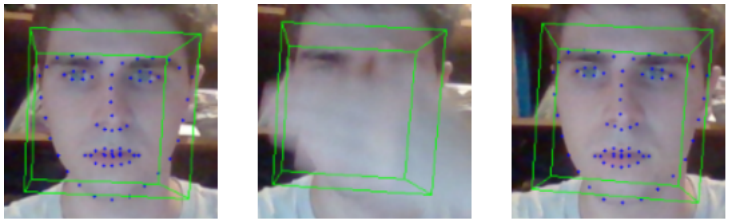

Figura 5.12: Predicción de la pose de la cabeza.

5.5.2. Visualización de la pose

Para la visualización de la pose de la cabeza se ha implementado la función `draw2Dbox` dentro del módulo `Pose Estimation`. Ésta, en función de la pose estimada y la matriz de parámetros intrínsecos de la cámara, dibuja un cubo rotado que contiene la cabeza del sujeto. Para ello, hace uso de la función `projectPoints` de `OpenCV`, con la cual podemos proyectar los puntos tridimensionales de un cubo que contiene al modelo de cabeza tridimensional sobre el plano imagen mediante la pose de la cabeza previamente estimada, tal y como se puede obervar en la figura 5.13.

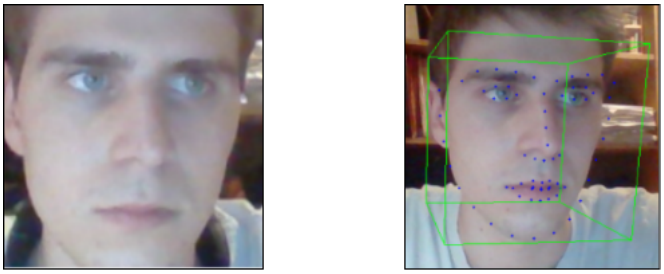

Figura 5.13: Visualización de la pose de la cabeza.

5.6 Estimación de la dirección de la mirada

Una vez extraídos los *landmarks* y la pose de la cabeza se procede a la estimación de la mirada. Para la estimación de la dirección de la mirada se ha seguido un proceso similar al descrito por Vardan Agarwal en su publicación "*Real-time eye tracking using OpenCV and Dlib*" [40].

Para ello, el proceso constará de las siguientes fases:

1. Alineamiento del rostro y sus respectivos *landmarks*.
2. Extracción de la región de interés de cada ojo.
3. Procesamiento de las regiones de interés.
4. Detección de contornos y extracción del centro de masa.
5. Estimación de la dirección de la mirada.

Alineamiento del rostro y sus respectivos *landmarks*

Una vez extraídos los puntos de referencia faciales podemos calcular el ángulo de rotación de la cabeza en el plano imagen. Para ello, extraemos los puntos medios de cada

ojo (mediante los *landmarks* de los ojos) y calculamos el ángulo de la línea que une ambos puntos. Una vez computado el ángulo podemos rotar la imagen mediante la función `warpAffine` de `OpenCV` y, después, rotar los *landmarks* para que coincidan con los de la imagen rotada. En la figura 5.14 se puede observar como el ángulo entre el eje horizontal (línea verde) y el eje que unes los ojos (puntos azules) sirve para alinear el rostro de la imagen de entrada.

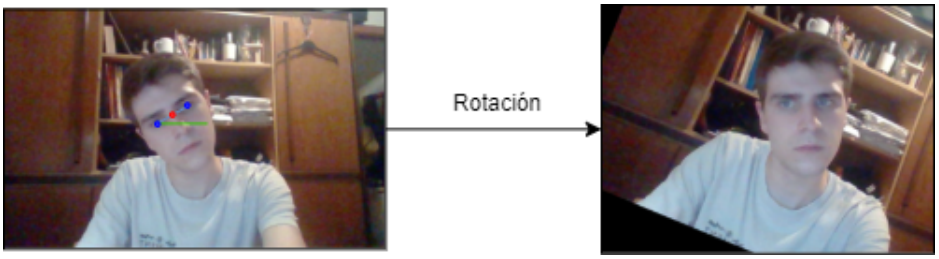

Figura 5.14: Alineamiento del rostro y sus *landmarks*.

Extracción de las regiones de interés de los ojos

Con el rostro y sus respectivos *landmarks* rotados, podemos extraer la región de cada ojo para sus posterior análisis. Para ello, en primer lugar debes crear lo que se conoce como máscara. Una máscara es una imagen que nos servirá para enmascarar, como su nombre indica, parte de una imagen. La máscara se construye dibujando dos polígonos blancos, mediante los *landmarks* en el interior de lo ojos sobre un fondo negro. En segundo lugar, mediante la función `bitwise-and` de `OpenCV`, la máscara y la imagen rotada, obtenemos la imagen rotada excluyendo toda la información que no es relativa al interior de los ojos. Por último, extraemos de ésta la región de cada ojo mediante los *landmarks*, tal y como se puede apreciar en la figura 5.15.

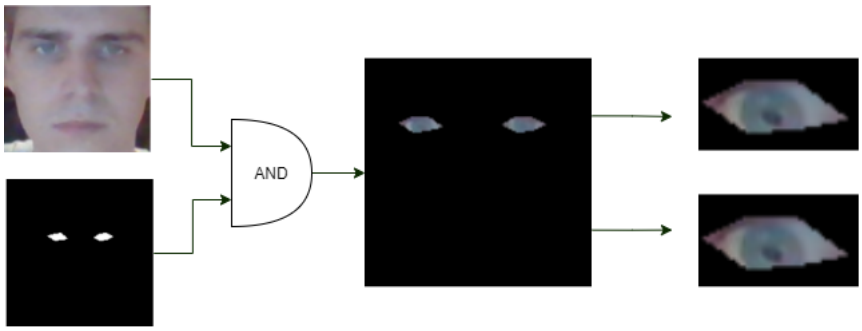

Figura 5.15: Extracción de las regiones de interés de los ojos.

Procesamiento de las regiones de interés

Una vez extraídas las regiones de interés relativas a los ojos, procesamos dichas regiones mediantes las técnicas de procesamiento expuestas en el apéndice B. Tras pasar las regiones a escala de grises se aplican los siguientes procesamientos:

1. ***Thresholding***: para diferenciar la región de la pupila del resto del ojo.
2. **Erosión**: para eliminar las posibles áreas que no pertenezcan a la región de la pupila.
3. **Dilatación**: una vez eliminadas las áreas que no pertenecen a la pupila, se dilata de nuevo para su posterior análisis.
4. **Desenfoque por filtro mediano**: para suavizar los límites de la región de la pupila para su posterior análisis.
5. **Inversión**: se invierte la imagen para que la zona de la pupila sea blanca y el resto negro.

En la figura 5.16 se puede apreciar como la concatenación de los procesamientos sobre la región de interés de los ojos aísla la región de la pupila.

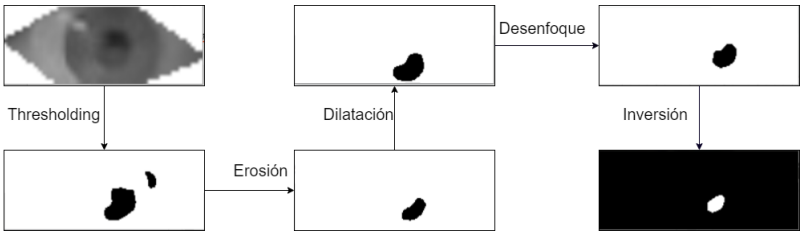

Figura 5.16: Procesamiento de las regiones de interés.

Detección de contornos y extracción del centro de masa

Una vez procesadas las regiones de interés de cada ojo se procede a la detección de contornos para hallar la posición de la pupila. Para ello se hace uso de la función `findContours` de `OpenCV` para encontrar los contornos. Una vez detectados los contornos, se extrae el contorno de mayor área mediante la función `contourArea` y posteriormente se extrae su centro de masa mediante la función `moments`, tal y como se puede apreciar en la figura 5.17.

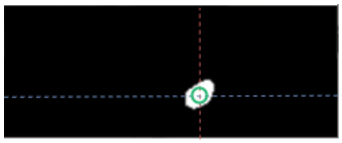

Figura 5.17: Detección de contornos y extracción del centro de masa.

Estimación de la dirección de la mirada

Una vez conocidos los centros de masa de los contornos relativos a las pupilas, podemos calcular su posición. Posteriormente se divide dicha posición entre el ancho y alto de la región de interés para estimar la dirección de la mirada. De forma que 0.5 será centrado, 1 será mirando a la derecha y 0 será mirando a la izquierda.

5.6.1. Estabilización y predicción de la dirección de la mirada

Para la estabilización y predicción de la dirección de la mirada se hace uso del módulo `Gaze Stabilizer` implementado. El cual cuenta con un total de dos `stabilizers` de una dimensión (uno para la componente horizontal y otro para la vertical).

5.6.2. Visualización de la dirección de la mirada

Para la visualización de la dirección de la mirada se ha implementado la función `draw-eye-rays` dentro del módulo `Gaze Estimator`. Para ello, la función hace uso de la rotación y traslación calculada mediante `Pose Estimator`. El proceso consiste en dibujar dos líneas cuyo origen es el centro de cada ojo y dirección es la estimada por el módulo `Pose Estimator`. Después, en función de la posición de la pupila, las líneas se rotan hasta $\pm$ 45 grados. En la figura 5.18 se puede observar la visualización de la dirección de la mirada (líneas de color magenta).

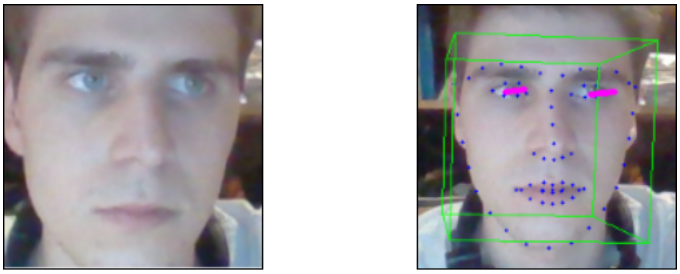

Figura 5.18: Visualización de la dirección de la mirada.

5.7 Estimación de gestos faciales

Otra de las mediciones que podemos estimar mediante los *landmarks* extraídos son los gestos faciales. Concretamente el módulo `Action Unit Detector` es capaz de estimar un total de cinco gestos diferentes en el rango $[0, 1]$, cuya descripción se encuentra definida en la tabla 5.3.

Nombre	Descripción
`mor (Mouth Open Ratio)`	Abrir de la boca
`msr (Mouth Smile Ratio)`	Separar de las comisuras de la boca (sonrisa)
`mkr(Mouth Kiss Ratio)`	Juntar las comisuras de la boca (beso)
`ear (Eye Aspect Ratio)`	Pestañear
`ebr (Eye Brow Raise Ratio)`	Levantar o bajar las cejas

Tabla 5.3: Gestos faciales detectados por el sistema.

Cabe destacar que para la estimación de estos gestos se hace uso de los *landmarks* no estabilizados. La razón de ello es que, dado que los gestos faciales pueden variar muy rápidamente, el uso de los *landmarks* estabilizados puede suponer una detección lenta de éstos. De forma que la estabilización de éstos se realizará posteriormente a su estimación mediante `stabilizers` de una dimensión.

Además, con el fin de evitar que la rotación y/o traslación de la cabeza influya de manera significativa a la estimación. Éstos serán siempre calculados en función de proporciones del rostro que no varíen ante diferentes poses. Por lo tanto, cada gesto se calcula como un ratio y posteriormente se normaliza mediante un rango establecido en la fase de experimentación, tal y como se puede observar en la tabla 5.4.

Gesto	Rango establecido para la normalización
`mor`	[0,0.5]
`msr`	[0.43,0.7]
`mkr`	[2.3,2.5]
`ear`	[0.15,0.3]
`ebr`	[1.55,1.7]

Tabla 5.4: Rangos establecidos para la normalización de los gestos estimados.

La función `get-all-au` implementada en el módulo `Actio Unit Detector` realiza la estimación de estos gestos mediante el cojunto de *landmarks* extraídos anteriormente. Tras su estimación se procede al filtrado mediante el módulo `Action Unit Stabilizer`, el cual contiene un total de cinco `stabilizers` de una dimensión (uno por cada gesto estimado).

5.7.1. Apertura de la boca

Para la estimación de la apertura de la boca el ratio escogido es la distancia entre los labios internos entre la distancia de la nariz, como muestran la figura 5.19 y la ecuación 5.1 donde D_1 es la distancia de la nariz y D_2 es la distancia entre los labios internos.

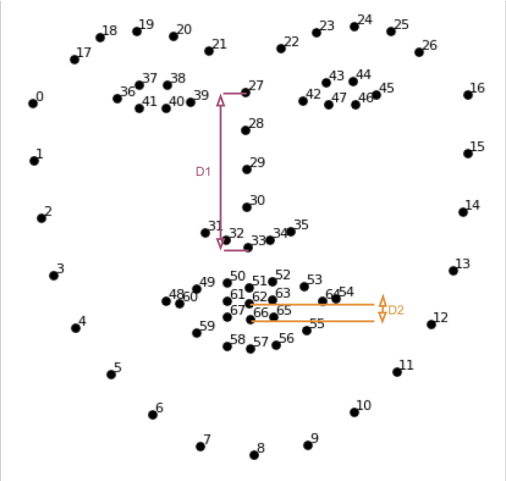

Figura 5.19: Estimación del gesto `mor`. [27]

$$MOR = \frac{D_1}{D_2} \tag{5.1}$$

En la figura 5.20 se puede apreciar la estimación de este gesto y su posterior filtrado para un vídeo donde el sujeto abre y cierra la boca lentamente.

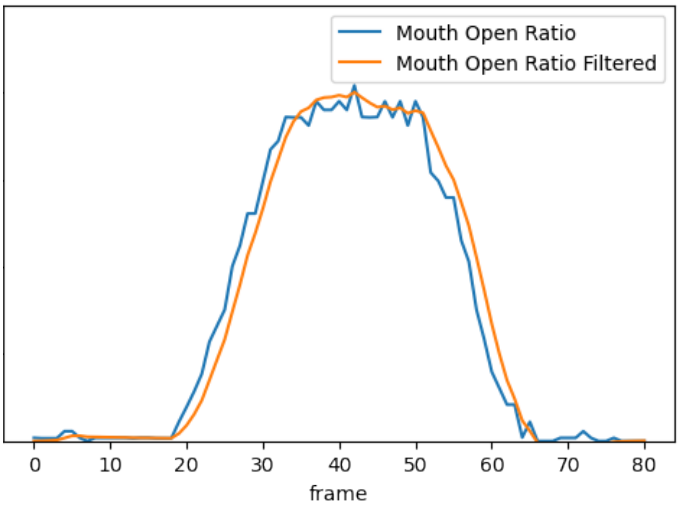

Figura 5.20: Estimación y filtrado del gesto `mor`. [42]

5.7.2. Separación de las comisuras de la boca (sonrisa)

Para este gesto realizamos la estimación calculando la distancia entre las comisuras de la boca (externas e internas) entre la distancia intraocular. Tal y como se puede apreciar en la figura 5.21 y la ecuación 5.2 donde las D_1 y D_2 son las distancias entre las comisuras de la boca y D_1 la distancia intraocular.

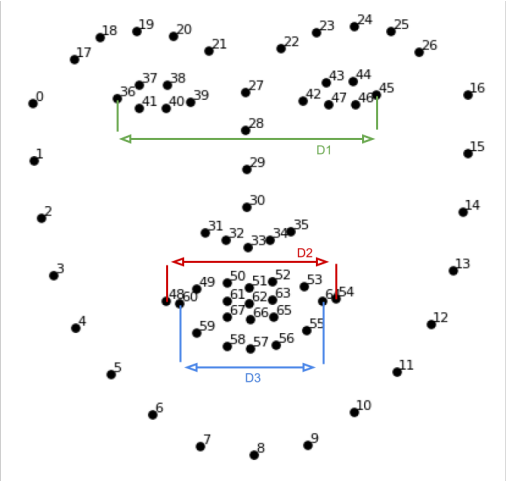

Figura 5.21: Estimación del gesto `msr`. [42]

$$MSR = \frac{D_2 + D_3}{2D_1} \tag{5.2}$$

Posteriormente, se procede al filtrado de la medición tal y como se puede apreciar en la figura 5.22. En ella podemos ver la estimación y filtrado del gesto `msr` para un vídeo donde el sujeto pasa de un estado neutral a sonreír para, finalmente, volver al estado neutral.

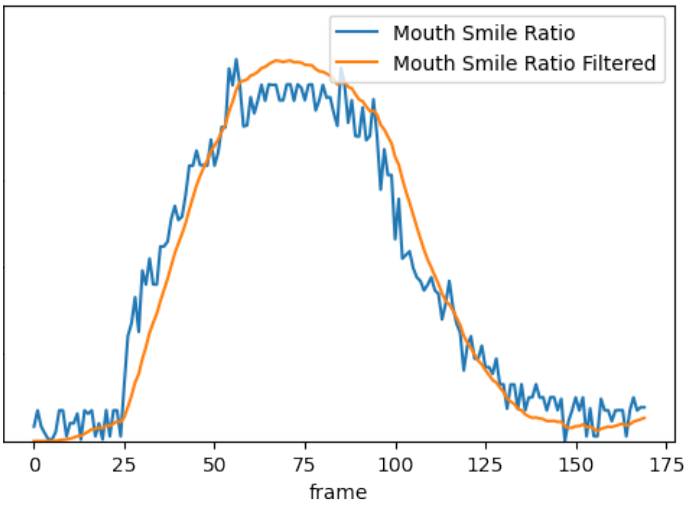

Figura 5.22: Estimación y filtrado del gesto `msr`.

5.7.3. Acercamiento de las comisuras de la boca (beso)

La peculiaridad de este gesto es que se puede calcular en función de los dos anteriormente expuesto. Tal y como se muestra en la figura 5.23 y la ecuación 5.3 donde D_2 y D_3 son las distancias entre las comisuras de la boca (externa e interna), D_5 es la distancia entre los labios internos, D_4 es la distancia de la nariz y D_1 es la distancia intraocular.

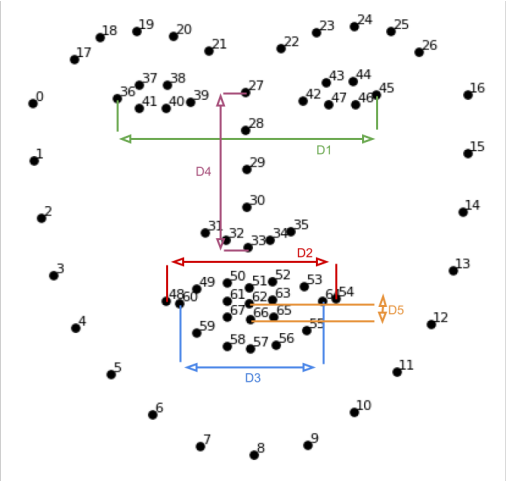

Figura 5.23: Estimación del gesto `mkr`. [42]

$$MKR = \frac{2D_1}{D_2 + D_3} - \frac{D5}{D4} = \frac{1}{MSR} - MOR \tag{5.3}$$

En la figura 5.24 se puede observar la estimación y filtrado de este gesto para un vídeo donde el sujeto pasa del estado neutral a realizar un gesto de beso para, finalmente, volver al estado neutral.

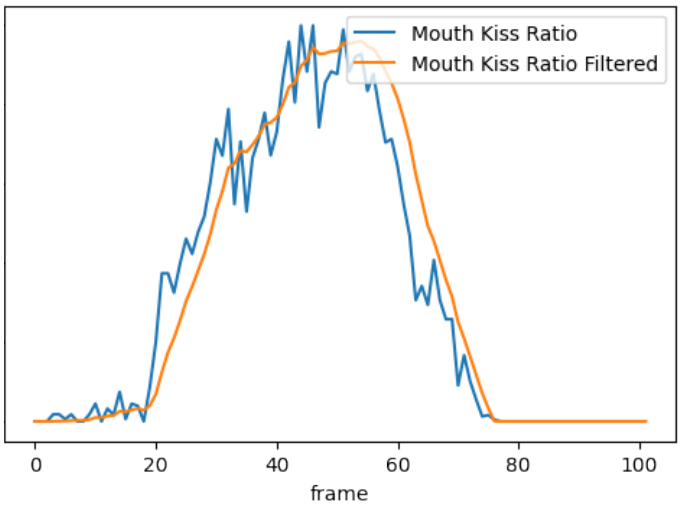

Figura 5.24: Estimación y filtrado del gesto `mkr`.

5.7.4. Pestañeo

Para estimar el gesto de pestañear (`ear`) se hace uso de la relación de aspecto de los ojos, es decir, el ratio altura/anchura. Tal y como se muestra en la figura 5.25 y la ecuación 5.4 donde D_1 , D_3 ,D_4 y D_6 representan la altura de los ojos mientras D_2 y D_5 su altura.

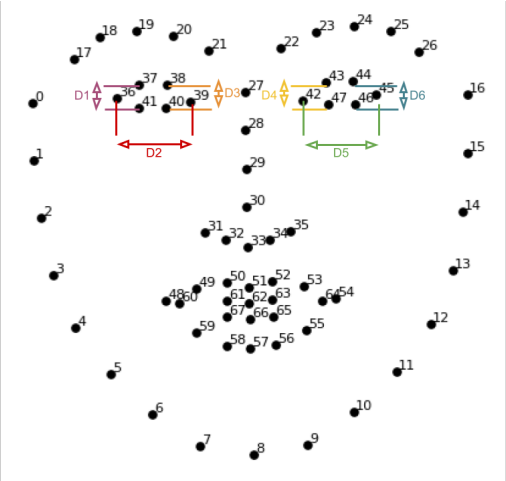

Figura 5.25: Estimación del gesto `ear`. [42]

$$EAR = \frac{\frac{D_1+D_3}{D_2} + \frac{D_4+D_6}{D_5}}{4} \tag{5.4}$$

Una vez calculada la relación de aspecto para cada ojo, se calcula la media entre éstas y posteriormente se procede a su filtrado, tal y como se puede observar en la figura 5.26 donde se analiza un vídeo donde el sujeto realiza múltiples pestañeos. A la izquierda se observa el `ear` para cada ojo y a la derecha la media tras ser filtrada mediante el `stabilizer`.

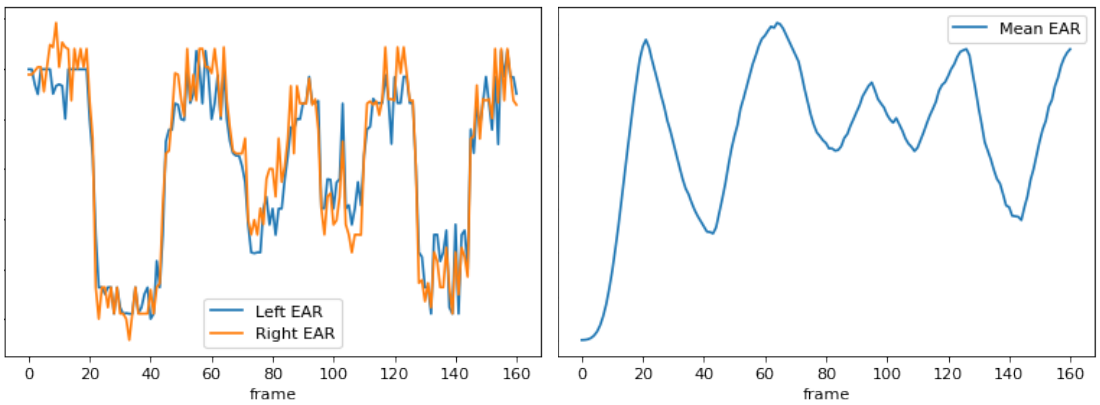

Figura 5.26: Estimación y filtrado del gesto `ear`. [42]

5.7.5. Movimiento vertical de las cejas

Por último, el movimiento de las cejas se estima como el ratio de la distancia entre éstas y el punto medio de la nariz con la longitud de la nariz. Tal y como se puede apreciar en la figura 5.27 y en la ecuación 5.5 donde D_1 y D_2 representan la distancia entre las cejas (izquierda y derecha) con el punto medio de la nariz y D_3 es la distancia de la nariz.

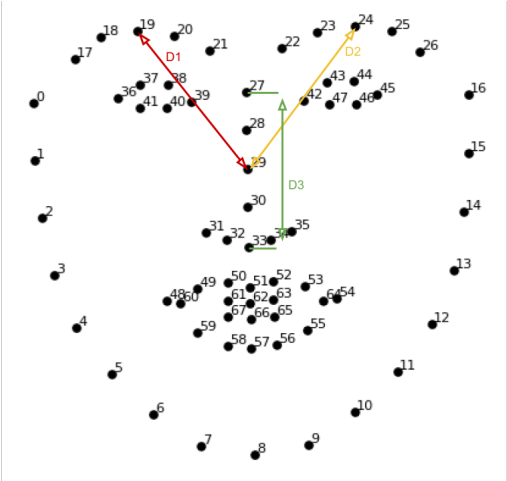

Figura 5.27: Estimación del gesto `ebr`. [42]

$$EBR = \frac{D_1 + D_2}{2D_3} \tag{5.5}$$

Posteriormente se realiza el filtrado de la medición mediante su respectivo `stabilizer`. En la figura 5.28 se puede observar dicho proceso para un vídeo donde el sujeto sube y baja las cejas.

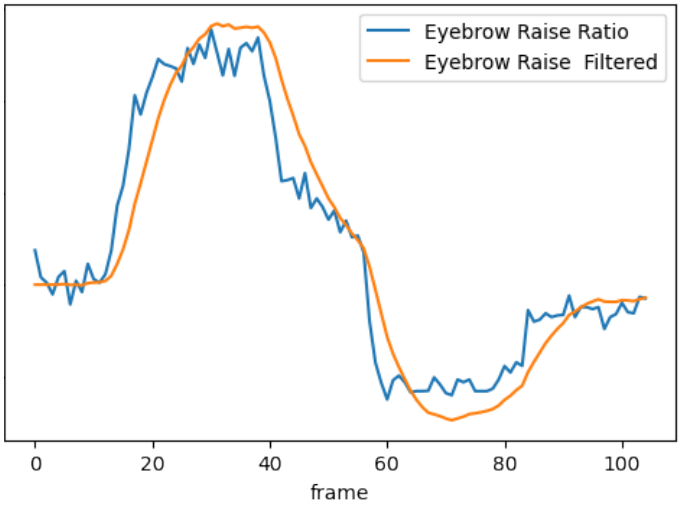

Figura 5.28: Estimación y filtrado del gesto `ebr`.

5.8 Detección de emociones mediante los gestos faciales estimados

Una vez estimados los gestos faciales, podemos detectar diferentes expresiones o emociones mediante estos. Para ello, simplemente asociaremos cada una de las posibles emociones a detectar a una condición de los gestos faciales, tal y como muestra la tabla 5.5.

Expresión	**Condición asociada**
Felicidad	`msr >= 0.7 and ebr >= 0.15`
Ira	`ebr <= 0.05 and msr <= 0.35`
Maldad (Ira con sonrisa)	`ebr <= 0.1 and msr >= 0.6`
Bostezo	`ebr <= 0.15 and mor >= 0.6`
Sorpresa	`ebr >= 0.8 and mor >= 0.6 and ebr >= 0.75`
Neutral	En otro caso

Tabla 5.5: Tabla de condiciones para la detección de expresiones mediante los gestos faciales estimados.

En la figura 5.29 podemos observar la estimación de gestos faciales en función de la región de interés extraída y la estimación de la emoción a través de éstos.

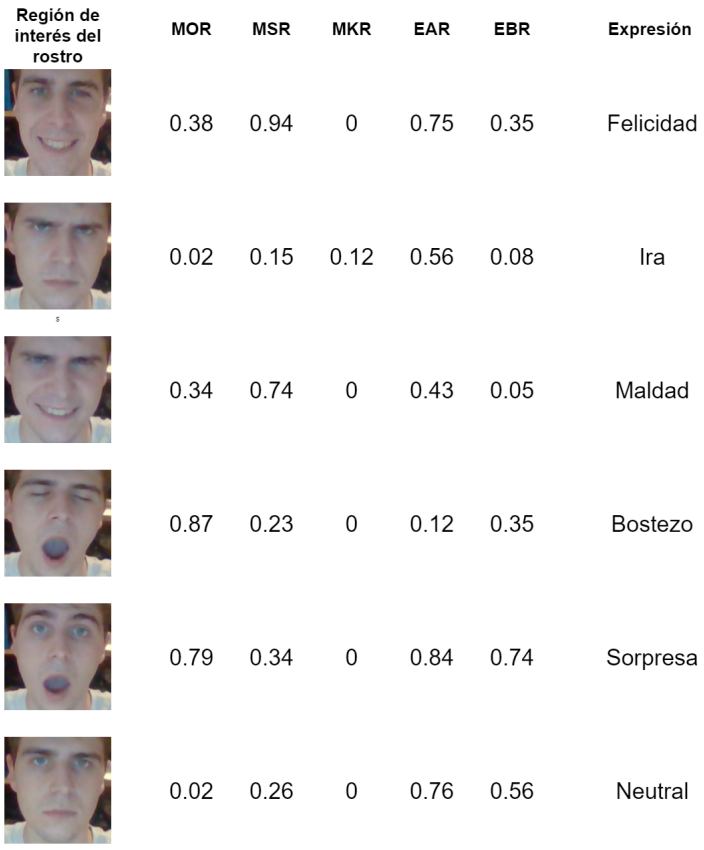

Figura 5.29: Representación visual de la estimación de emociones mediante los gestos faciales.

5.9 Detección de emociones mediante CNN's

Expuestas ya todas las mediciones cuya estimación viene dada por la extracción de los *landmarks*, se procede a la estimación de los parámetros mediante la región de interés del rostro y el uso de CNN's.

En el apartado anterior (5.8), se ha definido el proceso de detección de expresiones mediante el uso de los gestos faciales previamente estimados. El módulo `Emotion Detector CNN`, al contrario que en el caso anterior, hace uso únicamente de la región de interés del rostro detectado y, a partir de ésta, detecta las emociones a través la CNN preentrenada, la cual es capaz de detectar un total de siete emociones distintas (felicidad, ira, disgusto, sorpresa, tristeza, miedo y neutral). Dicha red neuronal convolucional es cargada a través del sub-módulo de `tensorflow` llamado `Keras`, el cual contiene una función (`load-model`) para la carga de modelos preentrenados.

El primer paso a seguir es el preprocesamiento la región de interés del rostro. Para ello, en primer lugar se escala a 64x64x3 dimensiones mediante la función `face-imagetest = cv2.resize(face-roi, (64,64), interpolation= cv2.INTER-CUBIC)`, después se convierte a escala de grises y, finalmente, se normaliza.

Una vez se ha realizado el preprocesamiento, se procede a la predicción mediante la función `model.predict(roi-procesada)` obteniendo así un vector de siete dimensiones, cada una asociada a una de las posibles siete emociones estimables por el modelo. Finalmente, mediante la función `argmax` se obtiene la emoción detectada.

Cabe destacar que, tal y como se expone en el artículo "*Real-Time Facial Expression Recognition Based on CNN*" [28], tras estimar el vector de predicciones, se reemplaza por la media entre éste y el de la última predicción realizada. La finalidad de éste proceso es mejorar la robustez del sistema, tal y como expone Keng-Cheng Liu en el artículo mencionado, dado que la expresión detectada en un *frame* debe de estar, en mayor o menor medida, vinculada a la estimada anteriormente, tal y como muestra la figura 5.30 donde se puede apreciar el proceso de detección de emociones entre un *frame* y otro.

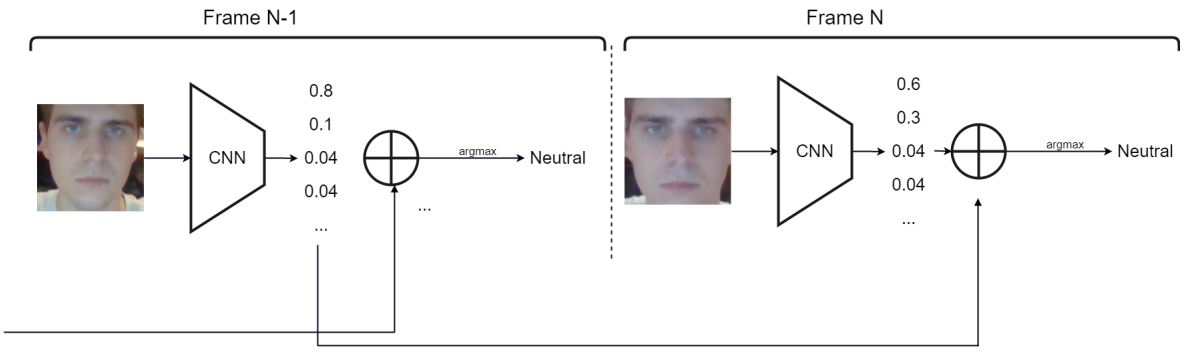

Figura 5.30: Proceso de detección de emociones mediante CNN's.

Además, con el fin de aumentar la velocidad del sistema, la detección de emociones se realizará cada 5 *frames* por defecto, a pesar de que se trata de un parámetro que puede modificado en función de las necesidades particulares del usuario. Otra de las causas de esta modificación es que, comúnmente, el sujeto no variará significativamente su expresión facial su expresión en un periodo de 5 *frames*, el cual a 10 FPS equivale a 0.5 segundos.

5.10 Estimación de edad y género mediante CNN's

El proceso para la estimación de la edad y el género dada la región de interés del rostro es similar al descrito anteriormente. En su lugar, el módulo `Age Gender Detector` cuenta con dos CNN's [1] (una para estimar la edad y otra para el género). Ambas redes son cargadas mediante la función `readNet` implementada en el sub-módulo `dnn` de `OpenCV`.

En el caso de la estimación de la edad, la red neuronal convolucional está entrenada para clasificar un total de 8 intervalos ([0,2], [3,6], [7,13], [14,22], [23,35], [36,44], [45,64], [65,100]).

El preprocesamiento de la región de interés del rostro se realiza mediante la función `blobFromImage(face-roi , (227,277))`. *Blob From Image* es una técnica de procesamiento de imágenes ampliamente utilizada en el campo del *Deep Learning* y las CNN's. Normalmente es utilizada para preprocesar una imagen antes de pasarla como entrada a una red neuronal, típicamente una CNN. Este proceso consta de dos fases principales:

1. **Substracción de la media**: Para cada uno de los canales de la imagen (RGB), el algoritmo calcula su respectiva media y la substrae de cada canal.
2. **Normalización**: Tras la substracción de la media para cada canal, éstos se normalizan para representar sus valores entre 0 y 1.

Una vez preprocesada, se inserta como entrada a las CNN's mediante la función `setInput`. Tras ello, el proceso es idéntico al descrito en el apartado anterior (5.9), tal y como muestra la figura 5.31.

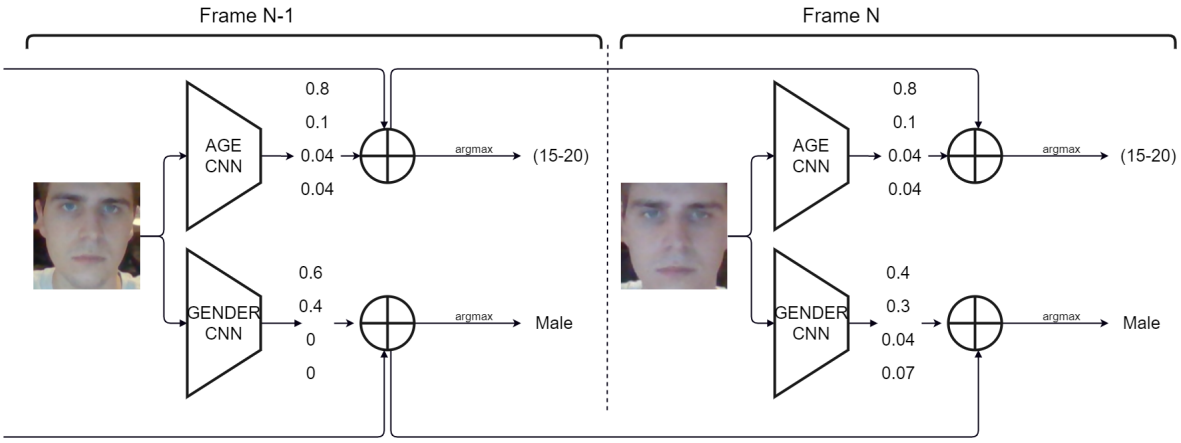

Figura 5.31: Proceso de detección de edad y género mediante CNN's.

Por último, cabe aclarar que, dado que la edad y el género son parámetros invariantes a menos que se detecte un rostro nuevo, por defecto la estimación de estos atributos se realizará cada 30 *frames*, parámetro configurable al igual que en el módulo `Emotion Detector CNN`. Además, tal y cómo se especifica en el diseño del sistema (4), cuando el parámetro `onlyOnce` esté activado (su valor sea `True`), dicha estimación se realizará únicamente tras la primera detección del rostro.

5.11 Identificación facial mediante CNN

Tal y como se expone en la solución propuesta (3.3), el módulo `Face Recognition` está apoyado en la librería `Face Recognition` para la tarea de reconocer rostros.

La base de datos de rostros, cargada y almacenada mediante la librería `pickle`, consiste en una lista de diccionarios donde cada uno de éstos contiene el nombre, una imagen de la región de interés, y la codificación facial del rostro detectado en la forma `[name: "Alvaro Blom", face: *región de interés del rostro* , encodings: *codificación del rostro*, ...]`.

Por defecto, el identificación del rostro se realizará una única vez tras la primera detección de éste (`onlyOnce = True`), en caso contrario se realizará cada 30 *frames* (parámetros configurables).

La codificación del rostro se realiza a través de la función `face-encodings(face-roi)`, con la cual obtenemos el vector de 128 dimensiones (sobre la 128-esfera) que representa al rostro. Una vez obtenido dicho vector podemos compararlo con otros mediante la función `np.linalg.norm(face1encodings-face2encodings)`, la cual calcula la distancia euclídea entre los vectores de codificación relativos a cada rostro. Tal y como especifica la documentación de `face-recognition`, en caso de que dicha distancia sea menor a 0.6 unidades, podemos afirmar que se trata del mismo sujeto. En la figura 5.32 se puede observar la matriz asociada a la comparación entre tres rostros diferentes.

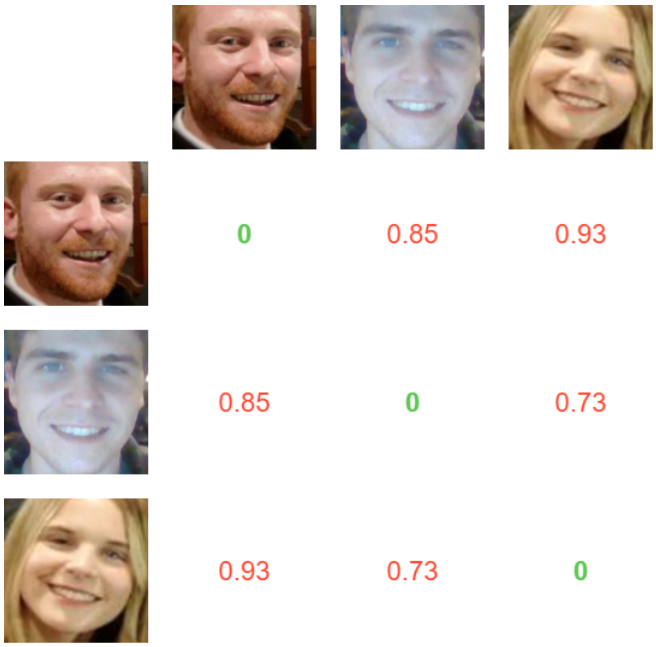

Figura 5.32: Representación gráfica de la distancia entre codificaciones faciales.

A través de este proceso podemos comparar la codificación de la cara detectada con las guardadas en la base de datos. Una vez realizada dicha comparación descartaremos las codificaciones con una distancia menor o igual a 0.6 y, posteriormente, escogeremos la de menor distancia (si es que la hay). De esta forma la función implementada `get-name` es capaz de reconocer el rostro detectado y asociarlo a su respectivo nombre, tal y como muestra la figura 5.33.

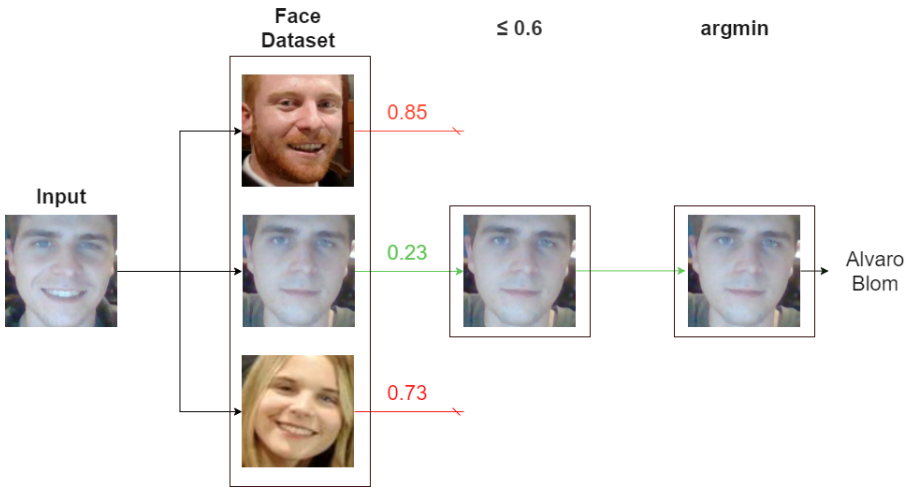

Figura 5.33: Representación gráfica del proceso de identificación facial mediante la base de datos de codificaciones faciales.

5.12 Guardado, cargado y filtrado de las mediciones

Una vez realizado todo el análisis relativo a la monitorización facial, otra de las funcionalidades implementada en el sistema es el guardado, carga y filtrado de las mediciones efectuadas. Para ello, el módulo `Recorder` contiene el atributo (`record`), el cual se trata de una lista donde se almacenarán las mediciones efectuadas. Cada elemento de la lista contendrá las mediciones efectuadas en un *frame* concreto, de tal forma que cada elemento de la lista contendrá los atributos expuestos en la tabla 5.6.

Atributo	**Descripción**
`frame`	Número de *frame*
`image`	Imagen relativa al *frame*
`landmarks`	*landmarks* extraídos
`landmarks-stb`	*landmarks* estabilizadas
`aus`	gestos faciales estimados
`aus-stb`	gestos faciales estabilizados
`gaze`	dirección de la mirada estimada
`gaze-stb`	dirección de la mirada estabilizada
`pose`	pose de la cabeza estimada ([traslación,rotación])
`pose-stb`	pose de la cabeza estabilizada
`emotion`	expresión estimada mediante los gestos faciales
`emotion-cnn`	expresión detectada mediante CNN
`face-roi`	región de interés del rostro detectado
`face-encodings`	codificación facial del rostro detectado
`time`	segundos transcurridos tras la primera detección

Tabla 5.6: Atributos guardados en la lista `record` del módulo `Recorder`.

Cabe destacar que en caso de que el usuario desee estimar únicamente un subconjunto de las posibles mediciones como, por ejemplo, la pose de la cabeza, el resto de mediciones se guardaran automáticamente con el valor `None`. Al igual que en el caso del módulo `Face Recognition`, la carga y guardado de ésta lista se efectuará mediante la librería `pickle`.

Por último, para el filtrado de los datos analizados, se ha implementado una función (`search`), cuyos atributos son funciones *booleanas*. Por ejemplo, si el usuario desea buscar los *frames* dónde el sujeto monitorizado está sonriendo, debe ejecutar la función `search(au-stb = lambda au: au[msr] > 0.75)`. Para ello, la función implementada recorre la lista `record`, definida anteriormente, y realiza las comparaciones necesarias para el filtrado de las mediciones efectuadas. En la figura 5.34 se puede observar una representación gráfica de este proceso.

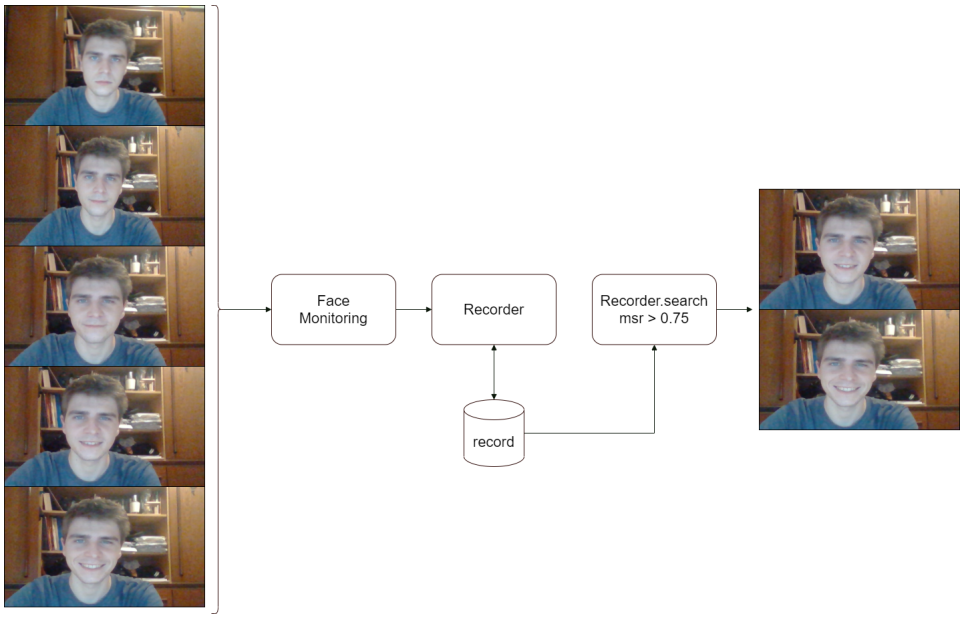

Figura 5.34: Proceso de búsqueda según las mediciones monitorizadas.

5.13 Módulo gestor

`Face Monitoring` es, tal y como se expone en el capítulo 4, el módulo gestor encargado de resolver todas las interdependencias entre los demás módulos. De esta forma, se consigue centralizar el proceso de monitorización facial en una sola clase con una sintaxis sencilla y fácil de comprender.

La función implementada `analyze` es la encargada de realizar todo el análisis resolviendo las interdependencias entre los módulos necesarios. Ésta contiene un total de tres argumentos posibles, tal y como muestra la tabla 5.7.

Argumento	Descripcción	Valor por defecto
`frame`	Imagen de entrada	
`target`	Lista de mediciones que se quieren monitorizar	`['all']`
`record`	Almacenar las mediciones en la lista `record`	`False`

Tabla 5.7: Argumentos de la función `analyze`.

En el caso del argumento `target`, se trata de una lista que contiene los nombres de las mediciones que el usuario desea estimar, tal y como muestra la tabla 5.8.

Atributo	Descripción
`landmarks`	*landmarks* extraídos
`aus`	gestos faciales estimados
`gaze`	dirección de la mirada estimada
`pose`	pose de la cabeza estimada ([traslación,rotación])
`emotion`	expresión estimada mediante los gestos faciales
`emotion-cnn`	expresión detectada mediante CNN
`face-roi`	región de interés del rostro detectado
`face-encodings`	codificación facial del rostro detectado
`all`	Todas las anteriores

Tabla 5.8: Posibles elementos de la lista argumento `target`.

De esta forma, podemos entender la función `analyze` como una "caja negra" que recibe los *inputs* comentados previamente y devuelve un resultado (*result*). Cabe destacar que, en el caso de las mediciones que posteriormente son estabilizadas, la función `analyze` devuelve su estimación estabilizada añadiendo la cadena `'_stb'` a ésta.

Por ejemplo, si el parámetro `target` contiene el elemento `'pose'`, la función devolverá tanto la medición `'pose'` como `'pose_stb'` (pose estabilizada). En la figura 5.35 podemos observar el resultado (`result`) que se obtiene tras ejecutar la función `analyze` para estimar los *landmarks*, la pose y la dirección de la mirada con el parámetro `record = True` y por lo tanto devolverá `None` en todos los parámetros del *output*.

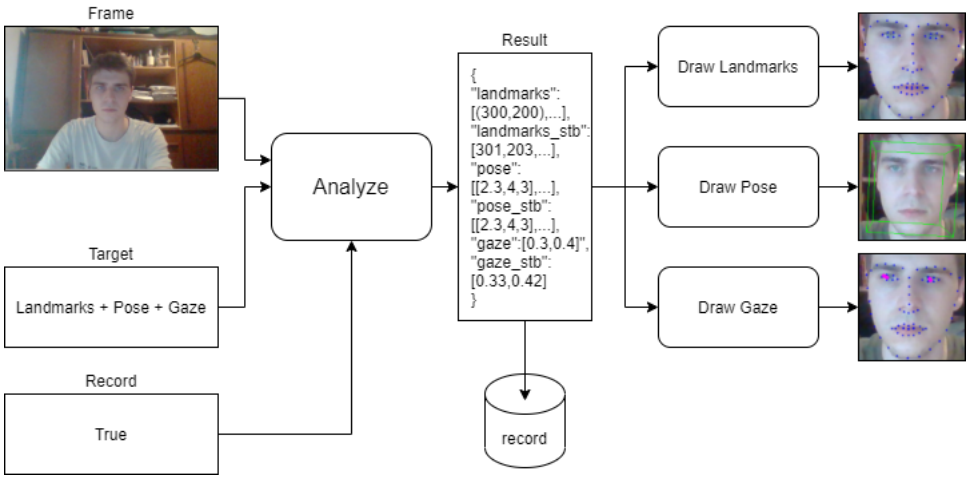

Figura 5.35: Representación gráfica de la función `analyze`.

En el caso de no poder estimar alguna de las mediciones por un fallo en la detección del rostro devolverá `None` exceptuando el caso de las mediciones estabilizadas, las cuales, como se ha expuesto previamente, pueden ser predichas durante un número máximo de *frames*, por defecto cinco y configurable mediante el parámetro `max_frames_without_frames`.

De forma que si, por ejemplo, ejecutamos la función `result = analyze(frame,['pose'])` y a causa de un problema en la detección `result['pose'] = None`, podemos acceder al valor `result['pose_stb']` y ver el estado de la pose predicho tal y como se puede apreciar en la figura 5.36. En caso de que se exceda el número máximo de *frames* de predicción (`max_frames_without_frames`), el sistema interpretará que no hay ningún rostro en la imagen de entrada.

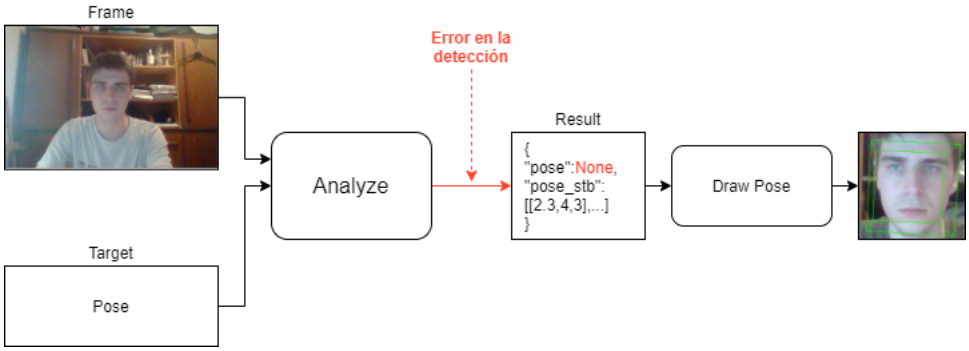

Figura 5.36: Representación gráfica de la función `analyze` ante fallos de detección.

Con el fin de mostrar la facilidad de uso del sistema, en la figura 5.37, se muestra un sencillo programa escrito en que monitoriza, a través de la *webcam*, la pose de la cabeza, las emociones (mediante CNN's) y lo muestra por pantalla mediante el módulo implementado `Face Monitoring`.

```
import FaceMonitoring

fm = FaceMonitoring() # Modulo gestor
fm.select_camera(0) # Selecciona la primera webcam disponible

while True:
    # Extrae un frame de la c mara
    img = fm.get_frame()
    # Si hay un error en la extraccion -> salir
    if img is None: break
    # Analisis de la pose y las emociones (CNN)
    result = fm.analyze(img,target=['pose','emotion-cnn'])
    # Muestra por pantalla la pose estabilizada
    print("pose estabilizada = " + result["pose_stb"])
    # Muestra por pantalla la emocion detectada
    print("emocion detectada mediante cnn : " + result["emotion-cnn"])
```

Figura 5.37: Programa en `Python` para la detección de la pose y emociones.

5.14 Aplicación para la visualización de la monitorización facial

Una vez implementado el sistema, con el fin mostrar visualmente los parámetros estimados, se ha desarrollado una aplicación demostrativa (`realtime`) con su respectiva interfaz mediante la librería `Tkinter` [45]. `Tkinter` forma parte de las librerías estándar dentro del lenguaje de programación `Python`. Con ella es posible desarrollar una interfaz gráfica de forma simple y disponible tanto en platadormas `Unix` como `Windows`. Cabe destacar que el objetivo principal de esta aplicación es visualizar el funcionamiento interno del sistema de monitorización. Por lo tanto el diseño de una interfaz "visualmente atractiva" no forma parte de los propósitos relativos a la creación de ésta.

En primer lugar, el usuario podrá escoger los parámetros faciales a monitorización mediante una ventana inicial, tal y como se muestra en la figura 5.38. Además podrá seleccionar tanto el algoritmo de detección facial como la cámara (entre las disponibles en el equipo).

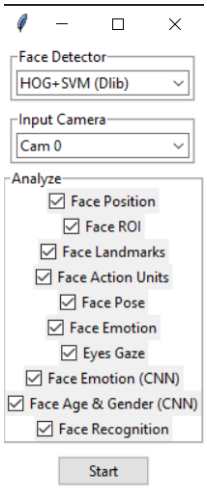

Figura 5.38: Captura de la ventana inicial de `realtime`.

Una vez seleccionadas las configuraciones anteriormente expuestas, se abrirá una nueva ventana mostrando los parámetros monitorizados seleccionados por el usuario. Dicha pantalla contiene un total de 8 cuadros diferentes.

El primer cuadro (*Camera*) mostrará la imagen recogida mediante la *webcam* y mostrará visualmente los *landmarks* extraídos, la posición de la cabeza, la dirección de la mirada y los FPS, tal y como se observa en la figura 5.39.

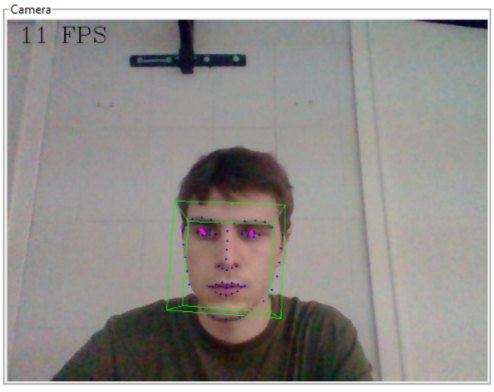

Figura 5.39: Captura del cuadro *Camera* de `realtime`.

El segundo cuadro (*Action Units*) mostrará los gestos faciales estimados mediante el módulo `Action Unit Detector` y posteriormente estabilizados mediante el módulo `Action Unit Stabilizer`, tal y como muestra la figura 5.40.

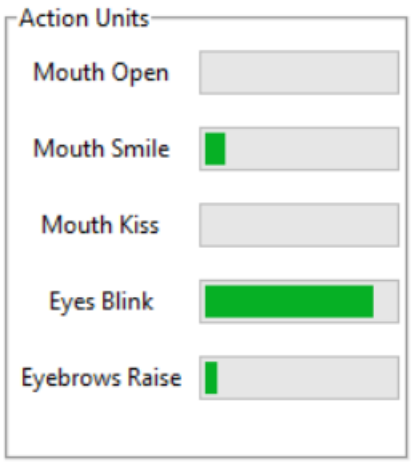

Figura 5.40: Captura del cuadro *Action Units* de `realtime`.

En el tercer cuadro (*Action Unit Based Emotion*), tal y como muestra la figura 5.41, se muestra la emoción clasificada mediante los gestos faciales.

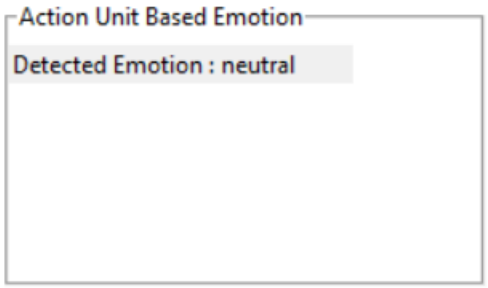

Figura 5.41: Captura del cuadro *Action Unit Based Emotion* de `realtime`.

El tercer cuadro muestra los vectores de traslación y rotación estimados tras estabilización. En el caso del vector de rotación, se ha hecho uso de la función `rotvec2euler` implementada en el módulo `Pose Estimator` para convertir el vector de rotación original a grados, tal y como se muestra en la figura 5.42.

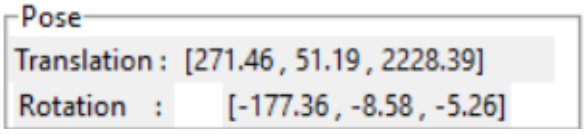

Figura 5.42: Captura del cuadro *Pose* de `realtime`.

El tercer cuadro (*Gaze*), tal y como se observa en la figura 5.43, muestra el procesamiento de la región de interés de los ojos necesario para la estimación de la mirada (*Left Eye Processed* y *Right Eye Processed*). Además tiene dos *sliders* asociados al parámetro *threshold* del procesamiento de cada ojo (*Left Eye Threshold* y *Right Eye Threshold*).

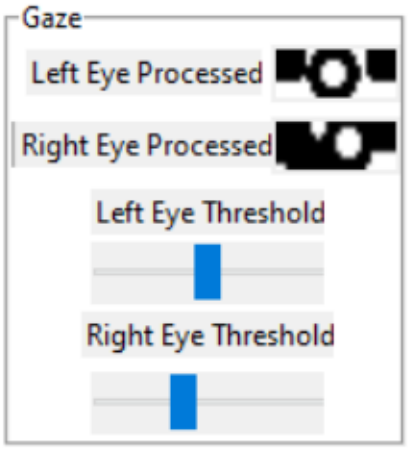

Figura 5.43: Captura del cuadro *Gaze* de `realtime`.

El cuarto cuadro (*CNN Based*) muestra, tal y como indica su nombre, los parámetros estimados mediante el uso de redes neuronales convolucionales. En la figura 5.44 se puede apreciar que dentro de este cuadro se muestra tanto la emoción como el género y la edad estimados mediante CNN's.

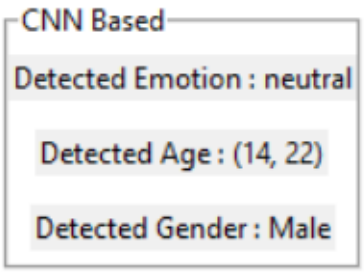

Figura 5.44: Captura del cuadro *CNN Based* de `realtime`.

En quinto lugar nos encontramos con el cuadro *Face ROI*, el cual, tal y como muestra la figura 5.45, muestra la región de interés asociada al rostro detectado.

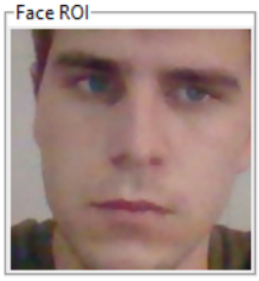

Figura 5.45: Captura del cuadro *CNN Based* de `realtime`.

Por último, el cuadro *Face Recognition* mostrado en la figura 5.46 permite al usuario identificar el rostro detectado mediante el botón *Recognize*. En caso de que la codificación del rostro no esté almacenada en la base de datos de rostros (*Faces Database*), el usuario podrá almacenar dicha codificación con su nombre asociado escribiendo el nombre del sujeto detectado en el *input* definido como *Name* y posteriormente pulsando el botón *Save Encodings*.

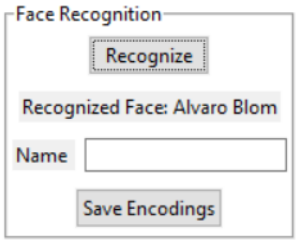

Figura 5.46: Captura del cuadro *CNN Based* de `realtime`.

En la figura 5.47 se muestra la visualización completa de la interfaz desarrollada. En ella se puede apreciar que, realizando todas las mediciones al mismo tiempo y mostrando una representación gráfica de éstas (lo cual también supone un coste computacional), la aplicación es capaz de llegar a los 12 FPS.

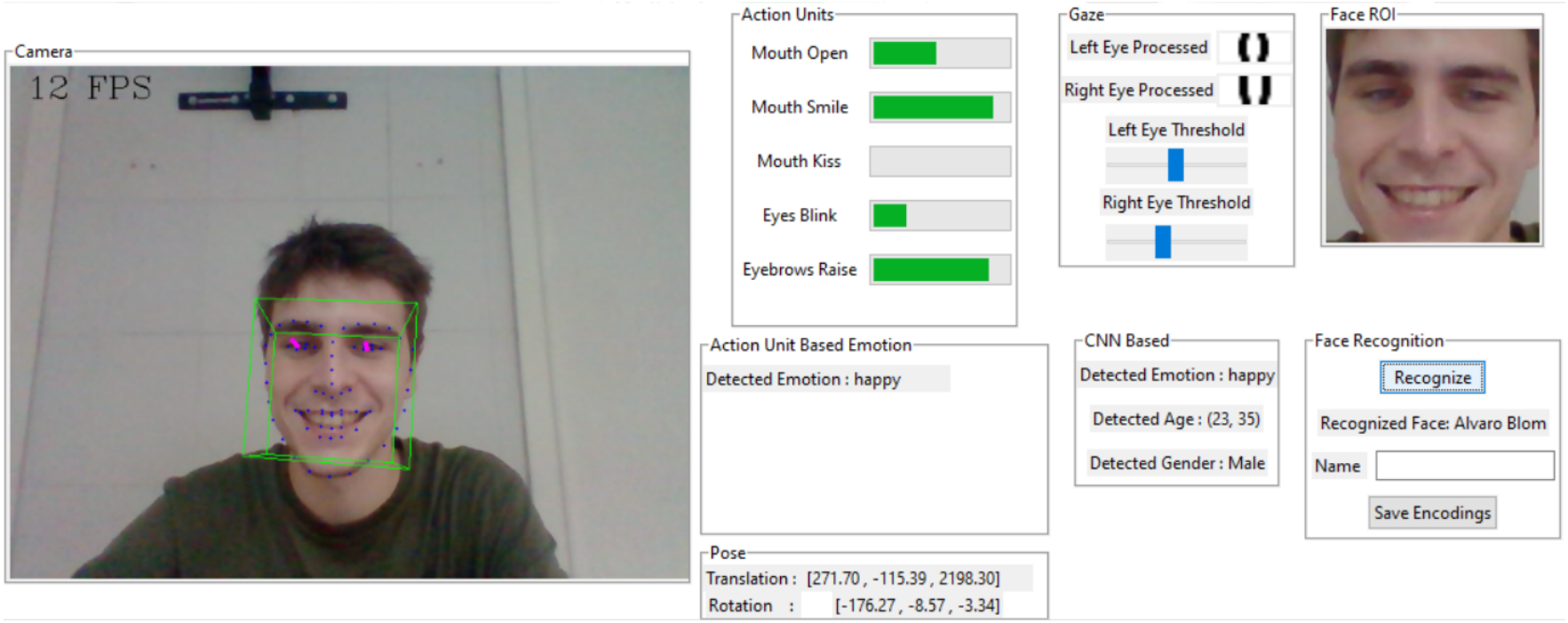

Figura 5.47: Visualización completa de la interfaz `realtime`.

CAPÍTULO 6
Pruebas

Una vez expuesto el desarrollo del sistema, es preciso realizar una serie de pruebas para el análisis de la robustez y la eficiencia de éste. En este capítulo se abordará dicha cuestión realizando en primer lugar pruebas para encontrar posibles fallos o elementos a mejorar y, por último, realizando pruebas acerca del uso de recursos del equipo y tiempos obtenidos.

6.1 Extracción de los *landmarks*

En primer lugar, cabe aclarar que las pruebas sobre la detección de rostros han sido previamente expuestas en el análisis de las soluciones posibles 3.2.1. A partir de éste, se procede a las pruebas sobre la extracción de los *landmarks*, tarea que va asociada también a la detección facial, dado que ésta es necesaria para la posterior extracción de los puntos de referencia.

La prueba consiste en analizar el efecto del algoritmo de detección facial utilizado en la extracción de los *landmarks*. En este caso, se evaluaré dicho efecto para los algoritmos HOG+SVM `Dlib` y CNN `cvlib`, dado que son los dos capaces de funcionar correctamente a tiempo real.

En la figura 6.1 se muestran distintas extracciones empleando el algoritmo de detección HOG+SVM. En ella podemos observar que para las imágenes 4 y 8 el algoritmo de detección no es capaz de detectar el rostro ante rotaciones de la cabeza grandes (mirando hacia el lado y hacia abajo). Además se puede apreciar en la imagen 6 que cuando el sujeto mira hacia arriba los *landmarks* asociados a la zona de la boca no se extraen de manera adecuada.

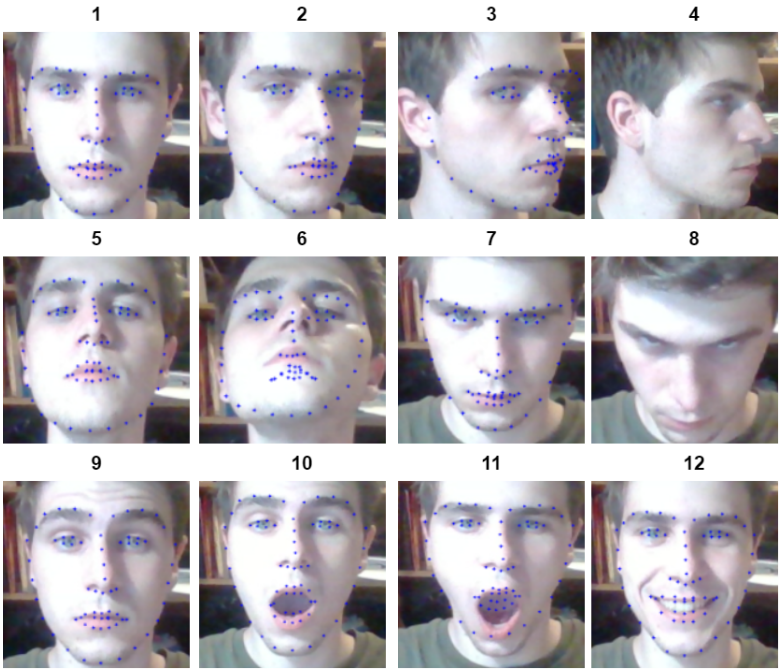

Figura 6.1: Extracción de *landmarks* mediante el algoritmo de detección HOG+SVM.

En el caso del algoritmo de detección CNN (`cvlib`) se puede apreciar en la figura 6.2 que el algoritmo es capaz de detectar el rostro en todas las ocasiones. Sin embargo, tal y como se puede apreciar en las imágenes 3 y 4, esto lleva a una incorrecta extracción de los *landmarks*. Éste problema puede deberse a que el modelo de extracción de *landmarks* pertenece a la librería `Dlib`, por lo que éste ha sido entrenado con el algoritmo de detección HOG+SVM (`Dlib`). Una de las posibles soluciones a este problema pasaría por reentrenar al modelo con el algoritmo CNN de `Dlib`, el cual sí puede detectar rostro en ángulos de rotación mayores.

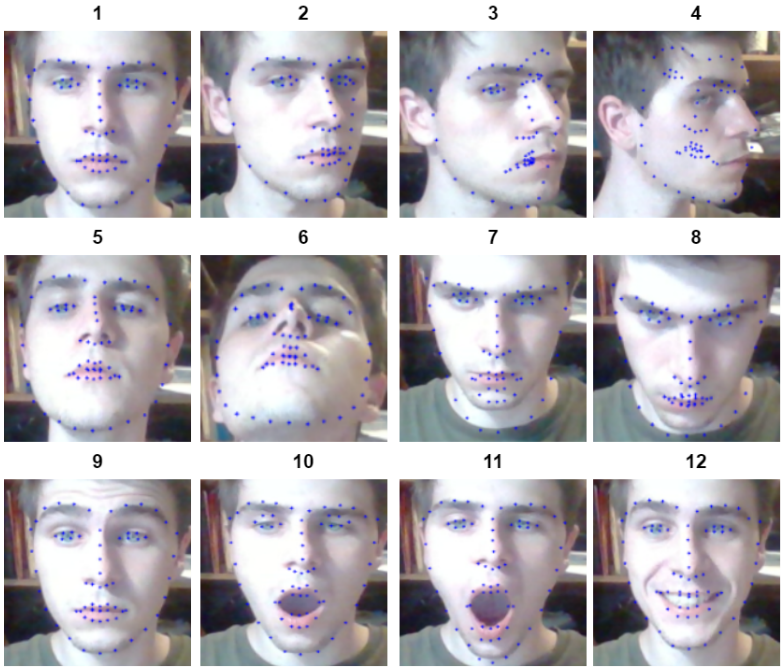

Figura 6.2: Extracción de *landmarks* mediante el algoritmo de detección CNN (`cvlib`).

6.2 Estimación de la pose de la cabeza

En lo respectivo a la estimación de la pose de la cabeza, podemos observar en la figura 6.3, que el cambio de los *landmarks* usados para la estimación de la pose ha dado buenos resultados. En las imágenes 3, 5 y 6 se puede apreciar que ante ángulos de rotación grandes el sistema es capaz de estimar la pose de forma precisa. Además, tal y como muestran las imágenes 7 y 8, la estimación de la pose es robusta ante muecas o gestos faciales.

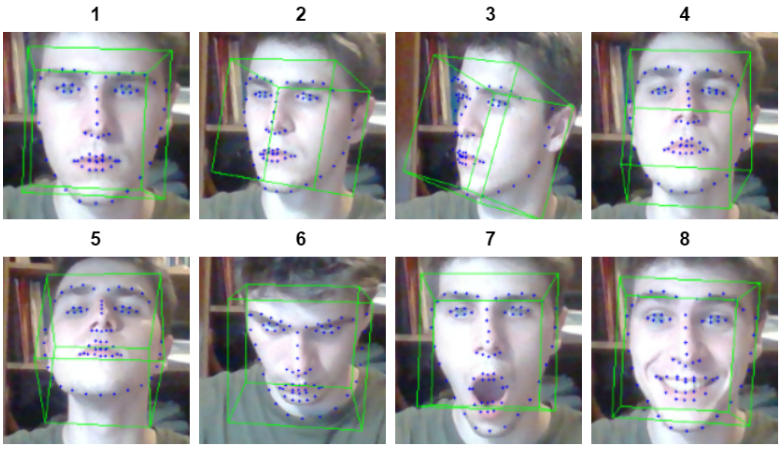

Figura 6.3: Estimación de la pose de la cabeza para distintos ángulos y gestos faciales.

6.3 Estimación de la dirección de la mirada

En cuanto a la dirección de la mirada, se puede apreciar en la figura 6.4 que la estimación se realiza de forma relativamente correcta. En la imagen 6 se puede observar que la estimación de la dirección de la mirada no se efectúa de forma correcta. Esto se debe a que, al estar mirando hacia abajo, el sistema en incapaz de detectar de forma correcta la pupila, ya que las cejas cubren la región de interés de los ojos.

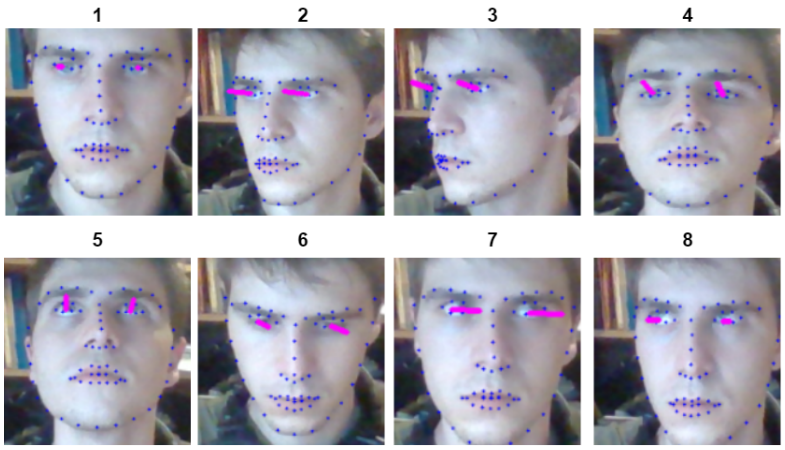

Figura 6.4: Estimación de la dirección de la mirada para distintos ángulos.

6.4 Estimación de los gestos faciales

En la figura 6.5 se muestra la estimación de gestos faciales para distintas imágenes de entrada. En cuanto a la apertura de la boca (*Mouth Open*), tal y como se puede observar en las imágenes 1, 2 y 4, el sistema es capaz de predecir de manera correcta este gesto y, además, dicha estimación no se ve perturbada por cambios de rotación como los mostrados en 9, 10, 11 y 12. Al igual que la apertura de la boca, el ratio de sonrisa (*Mouth Smile*) no se ve afectado por las rotaciones y se estima correctamente (1 ,2, 3 y 4). La estimación del gesto de beso (*Mouth Kiss*) se efectúa de forma correcta, como era de esperar, dado que éste se calcula mediante los dos anteriormente descritos. Tal y como muestran las imágenes l y 6, el gesto de pestañeo (*Eyes Blink*) se estima correctamente. Sin embargo, se puede apreciar en la imagen 9 que al levantar la cabeza puede producirse una estimación inferior a lo esperado. Por último, el gesto de levantar o subir las cejas (*Eyebrows Raise*), es el que presenta una mayor variabilidad ante cambios de rotación. Por ejemplo, en la imagen 10, 11 y 12 se puede apreciar que la rotación horizontal de la cabeza produce un cambio de estimación inesperado, por lo que podríamos concluir que es el gesto más variable y, por tanto, menos fiable.

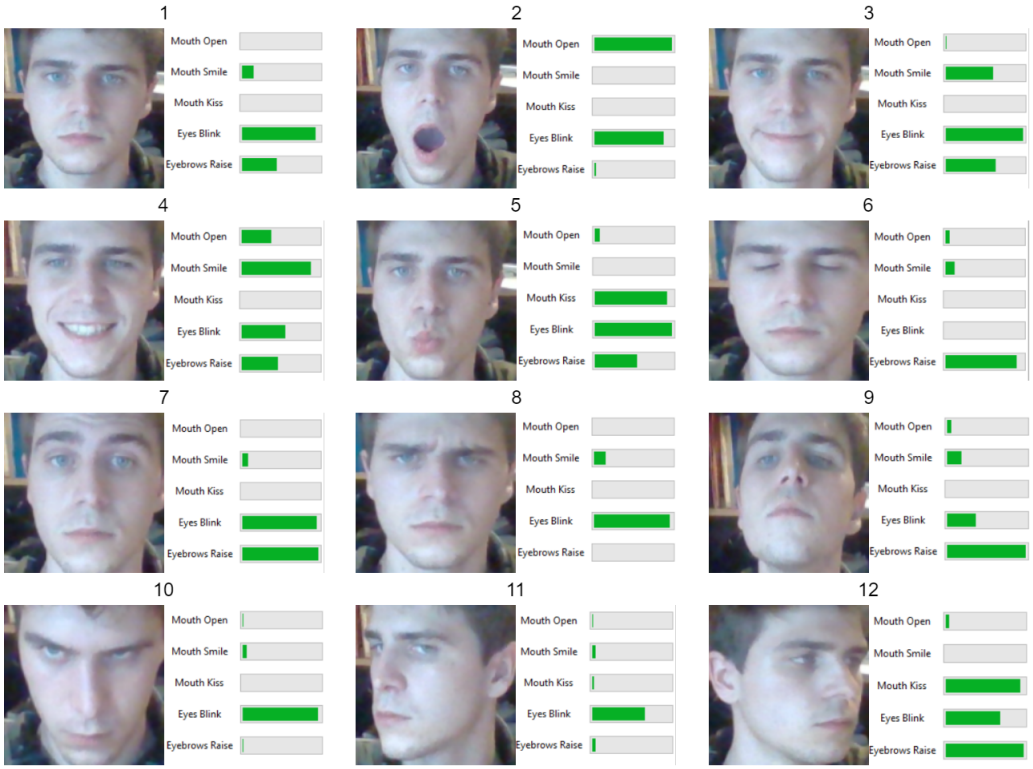

Figura 6.5: Pruebas de estimación de gestos faciales.

6.5 Estimación de emociones mediante los gestos faciales

En lo respectivo a la estimación de las emociones o expresiones mediante los gestos faciales podemos observar en la figura 6.6 que, en general, se estima correctamente (recuadros verdes). Sin embargo, en las imágenes 5 y 12 podemos observar un fallo en la estimación. Este fallo puede deberse a que la fallida estimación viene dado por un error en la estimación del gesto de bajar las cejas, el cual, como se ha expuesto en el anterior apartado, resulta el más variable. Como se puede observar en dichas imágenes, el sujeto está bajando las cejas, pero el sistema no lo estima así dado que, en caso contrario, la estimación de la expresión sería correcta.

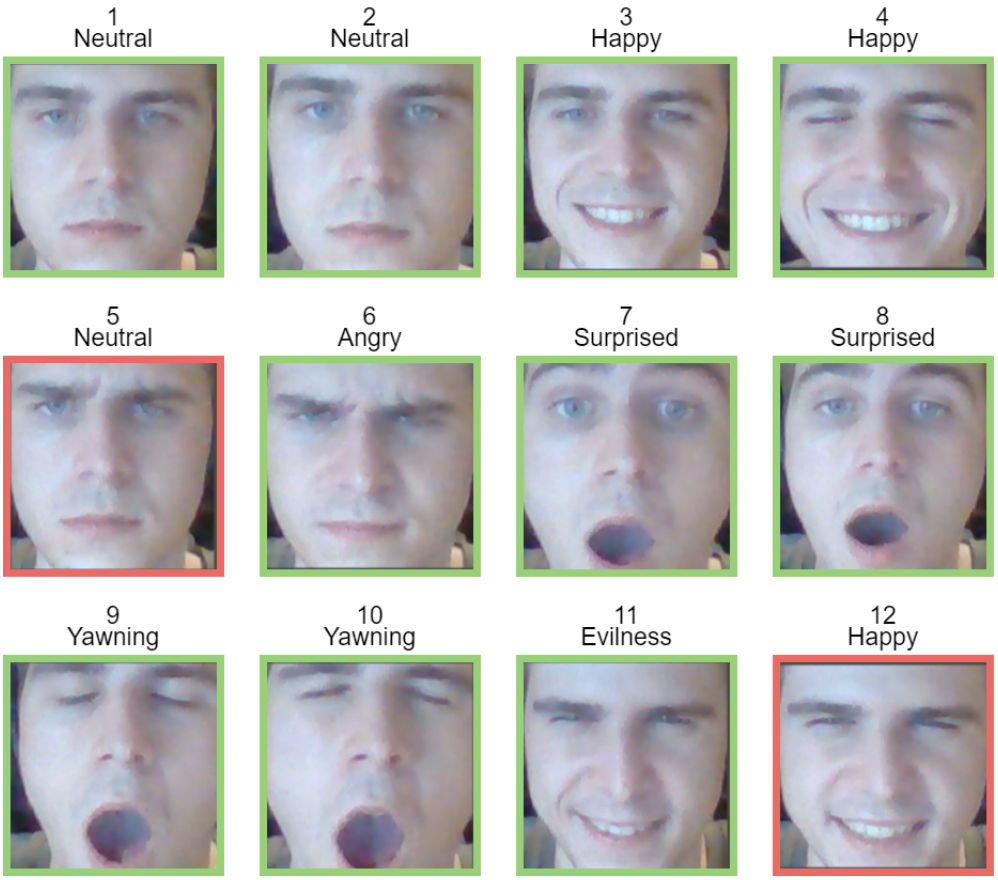

Figura 6.6: Pruebas de estimación de emociones mediante gestos faciales.

6.6 Estimación de emociones mediante CNN's

En el caso de las emociones estimadas mediante CNN's, podemos observar en la figura 6.7 que presenta algunos problemas de estimación. En la imagen 2 podemos observar que ha clasificado una imagen de un rostro neutral como triste, en la 12 se ha clasificado el disgusto como enfado y en la 10 el miedo como sorpresa. Sin embargo, cabe destacar que se trata de expresiones que pueden generar confusión. Por ejemplo, el disgusto puede ser interpretado como una forma de enfado y el miedo como una forma de sorpresa. Además, la capacidad de actuación del sujeto puede influir notablemente en el resultado del experimento. Por esta razón, tal y como se expondrá en el capítulo 8, se propondrá la evaluación del modelo mediante un gran conjunto de datos, el cual podría aportar más luz acerca de la eficacia del modelo de *Deep Learning* utilizado para esta tarea.

Figura 6.7: Pruebas de estimación de emociones mediante CNN.

6.7 Estimación del género y la edad

Para comprobar la efectividad de la estimación de la edad y el género se ha hecho uso del conjunto de datos (*dataset*) *All-Age-Faces* (AAF) [3]. El conjunto de datos AAF contiene un total de 13332 imágenes de rostros con su edad y género asociados incluyendo 7381 hombres y 5941 mujeres. Siendo la edad de la mayoría de ellos entre los 25 y los 35 años, tal y como se puede apreciar en la figura 6.8, donde se muestra la distribución de edad dentro del conjunto de datos. Tras evaluar el sistema con el conjunto de datos AAF se obtiene un porcentaje de acierto en la estimación de la edad del 63,47 %.

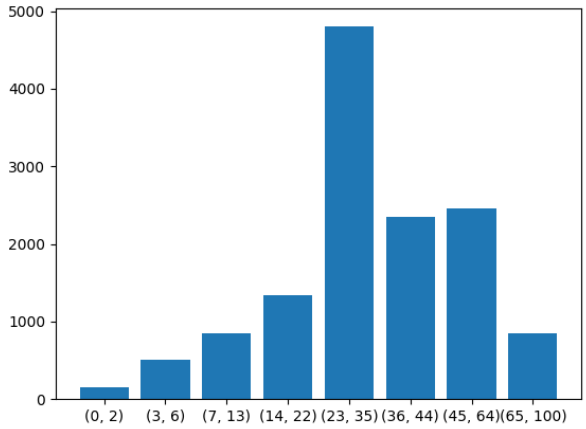

Figura 6.8: Distribución de la edad en el *dataset* AAF.

En lo respectivo a la estimación de la edad se han tomado dos métricas distintas para el cálculo del error. En la primera métrica, a la que nos referiremos como "error1", se estima que en caso de que la edad real se encuentre dentro del intervalo predicho el error es 0 y, en caso contrario, el error es la distancia absoluta de la edad real a la media del intervalo predicho. En la segunda métrica, a la que nos referiremos como "error1", en caso de que la edad real se encuentre fuera del intervalo predicho, el error se calculará como la distancia absoluta de la edad real al límite más cercano del intervalo.

Teniendo en cuenta la primera métrica, la estimación de la edad tiene un error asociado de $\pm$ 7.2 años y, en el caso de la segunda métrica, la estimación tiene un error asociado de $\pm$ 5.3 años de edad. En la figura 6.9 podemos observar que la distribución del error es equitativa en todos los intervalos de edad tanto para el "error1" (izquierda), como en el "error2" (derecha).

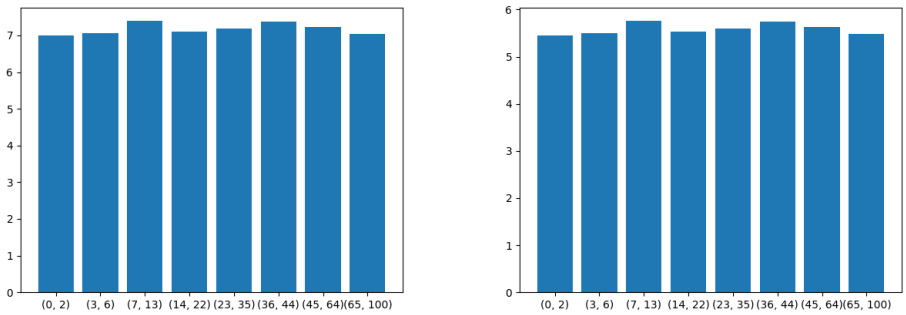

Figura 6.9: Distribución del error asociado a la estimación de la edad.

Por último, en la matriz de confusión asociada normalizada a la estimación de la edad, mostrada en la figura 6.10, podemos observar que el sistema tiende a clasificar los rostros de entre (12-22) y (36-44) años de edad como (23-35).

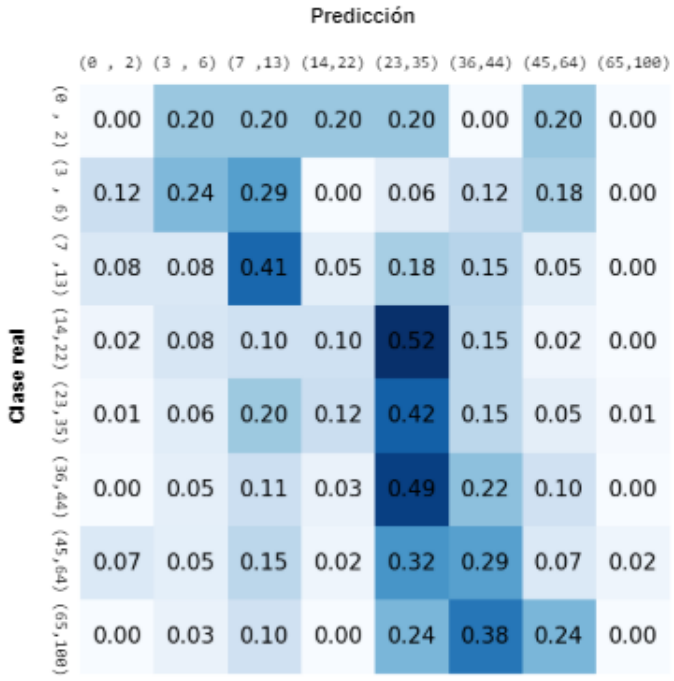

Figura 6.10: Matriz de confusión asociada a la clasificación de edad evaluada en el *dataset* AAF.

Teniendo en cuenta lo anteriormente expuesto, podemos afirmar que la estimación de la edad y el género es el elemento a monitorizar en el que se obtiene mayor variabilidad y, por tanto, el menos fiable.

6.8 Identificación facial

Para estimar la robustez del subsistema de identificación facial se han recopilado un total de 11 pares fotos de sujetos distintos. Cada par contiene una foto del sujeto con diferentes expresiones.

Posteriormente se ha calculado la distancia euclídea entre cada par de imágenes para obtener una comparación de los rostros detectados. Cuando dicha distancia es inferior a 0.6 unidades podemos afirmar que los rostros pertenecen al mismo sujeto.

De esta forma se ha construido la matriz de la figura 6.11 donde se puede apreciar que la diagonal principal es la única que contiene valores menores a 0.6 unidades. Por lo que podemos concluir que el identificación facial aportado por la librería `Face Recognition` aporta un buen resultado al sistema. En la figura 6.12 se puede apreciar como queda la matriz tras ser filtrada. De forma que los valores menores a 0.6 son azules y los mayores o iguales blancos.

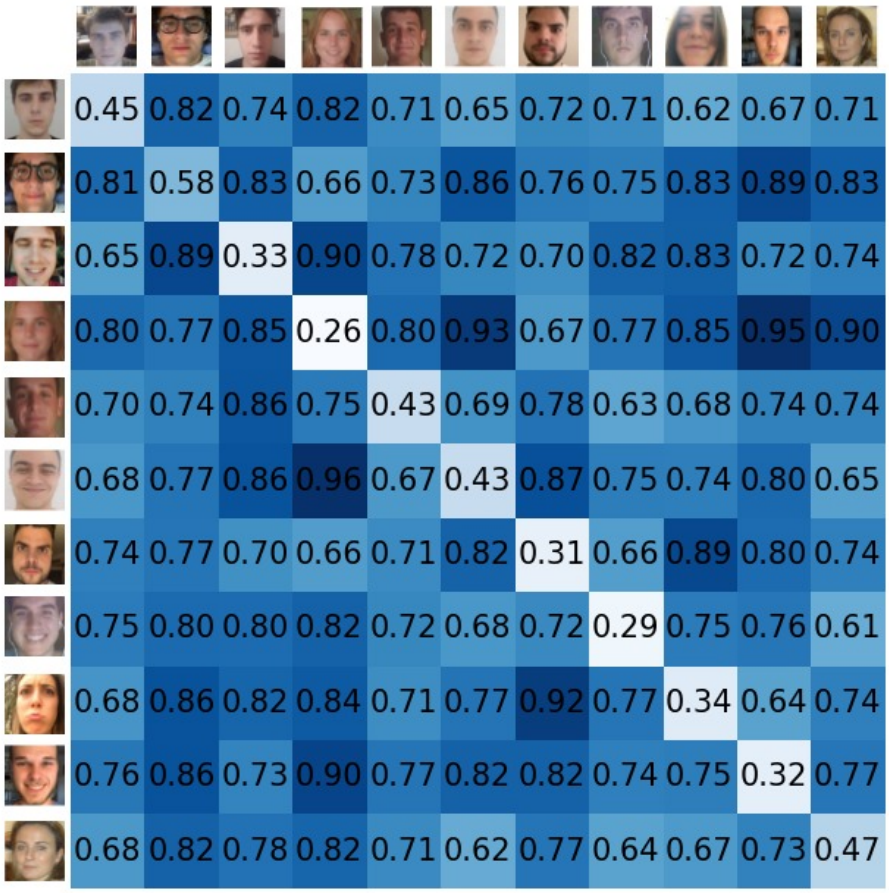

Figura 6.11: Matriz de distancias entre las codificaciones faciales.

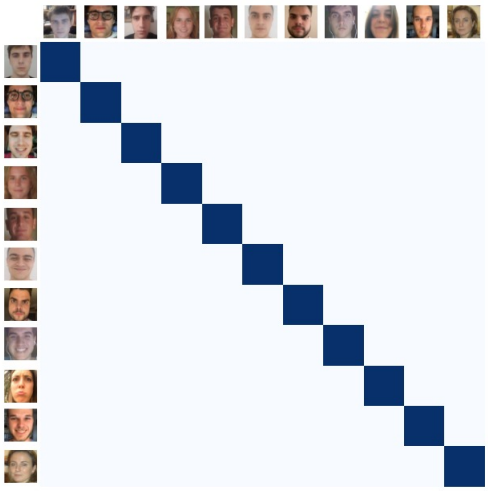

Figura 6.12: Matriz de distancias entre las codificaciones faciales filtrada por la condición `valor <= 6`.

El segundo experimento realizado consiste en comparar fotos de sujetos en distintas edades. Para ello se han recopilado 4 pares de imágenes donde cada par contiene una foto del sujeto del año 2003 y otra del 2021. En la figura 6.13 se puede observar que en todos los casos la distancia supera las 0.6 unidades además de que los elementos de la diagonal principal no son estrictamente menores a los demás, por lo que podemos concluir que el sistema de identificación es incapaz de reconocer al sujeto con el paso de los años. Sin embargo, esta característica resulta normal dentro del campo de la identificación facial, dado que en los sistemas de identificación facial es común actualizar periódicamente la imagen que representa a una persona.

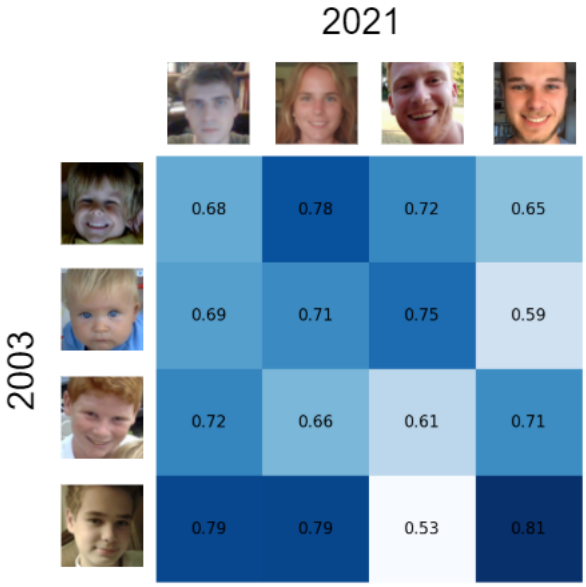

Figura 6.13: Matriz de distancias entre las codificaciones faciales (2003-2021).

6.9 Pruebas de rendimiento

Una vez analizada la robustez del sistema, es preciso realizar un análisis de rendimiento. En esta sección se evaluarán mediciones como tiempos de carga, tiempos de cómputo, FPS, uso de memória, etc.

En cuanto al tiempo de carga del sistema, es decir, del modulo `Face Monitoring`, se obtiene tiempo medio de 4.128 segundos. Esto se debe en gran medida a la carga de los modelos de *Deep Learning* (CNN's) implementados en el sistema, ya que la carga de éstos supone un 93.6 % del tiempo total de carga del sistema.

En la tabla 6.1 podemos observar los FPS de media que se obtienen al monitorizar a un sujeto mediante la *webcam* del ordenador en función de los parámetros a estimar (variable `target`). En ella, se puede apreciar como la estimación de la dirección de la mirada (gaze) y la codificación facial (face-encodings) resultan ser los procesos más costosos. En el caso de la estimación de la dirección de la mirada se debe al procesamiento de la región de interés del ojos, la cual supone un 84.7 % del tiempo de estimación. En el caso de la codificación facial, se debe al que el procesamiento de la CNN encargada de dicha tarea, ya que supone el 97.5 % del tiempo de cómputo total para la codificación.

Sin embargo, cabe destacar que la estimación de la dirección de la mirada se realiza en todos los *frames* mientras que la codificación facial se está realizando en este experimento cada 30 *frames*. Teniendo en cuenta que cada opción se ha evaluado con un total de 100 *frames*, se infiere que la codificación facial se ha realizado únicamente tres veces, lo cual supone un coste considerable a tener en cuenta. Por esta razón el módulo `Face Recognition` tiene habilitada la opción `onlyOnce` por defecto para realizar la codificación una única vez tras la primera detección del rostro.

Target	FPS
['pose']	20.306
['action-units']	20.595
['gaze']	15.921
['emotion']	21.450
['emotion-cnn']	20.132
['age-gender']	22.187
['face-encodings']	16.859
['all']	11.287

Tabla 6.1: *Frames* por segundo para distintos valores del parámentro `target`.

En lo respectivo al modelo de detección utilizado, realizando todas las mediciones al mismo tiempo (`targe = ['all']`), con el modelo de `Dlib` obtiene una media de 11.041 FPS, mientras que el CNN de `cvlib` obtiene 12.89 FPS. Sin embargo cabe destacar que, tal y como se ha expuesto en las pruebas de extracción de *landmarks* (6.1), la detección del rostro mediante este segundo modelo puede suponer una pérdida de robustez a la hora de extraer los puntos de referencia faciales. Por lo que se recomienda el uso del modelo de detección CNN siempre que los parámetros a estimar no dependan de los *landmarks* estimados, como la codificación facial o la detección del género y la edad.

En la figura 6.14 se muestra el porcentaje de CPU y memoria utilizado por el equipo mientras se ejecuta la función `analyze` con el parámetro `target = ['all']`, es decir, realizando todas las mediciones tal y como el sistema las realiza por defecto (sin mostrar nada por pantalla). En ella podemos observar que tras la inicialización del programa el uso de CPU se eleva de aproximada el 5 % a el 30 %, teniendo su máximo valor en 31 % de uso de CPU. Sin embargo, no se aprecia una subida significante del uso de memoria.

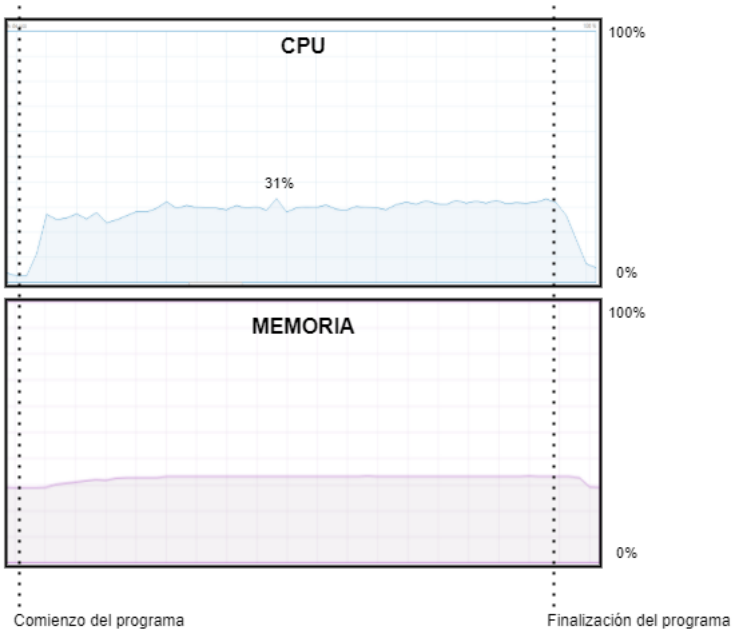

Figura 6.14: Uso de CPU y memoria realizando el análisis de todas las mediciones.

Sin embargo, realizando un análisis más exhaustivo podemos observar, tal y como muestra la figura 6.15 donde se muestra el uso de cada procesador lógico, que la carga del cómputo la está soportando un sólo núcleo. De tal forma que, tal y como se expresará más adelante en los trabajos futuros, podría resultar interesante tratar de optimizar el programa mediante el uso de multiprocesamiento.

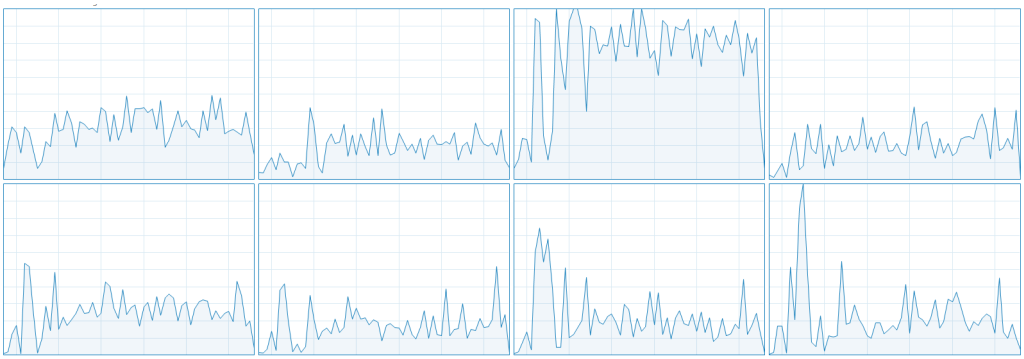

Figura 6.15: Uso de procesadores lógicos realizando el análisis de todas las mediciones.

CAPÍTULO 7
Casos de uso del sistema

En este capítulo se mostrarán diversas aplicaciones demostrativas y casos de uso posibles del sistema con el fin de mostrar las posibles aplicaciones que éste puede tener en diversos sectores del mercado. Además, a través de éstas se mostrará la facilidad de uso del sistema y,por tanto, el potencial que éste tiene como una posible futura librería para `Python` especializada en el reconocimiento facial.

7.1 Pasa páginas automático

Actualmente, una gran cantidad de músicos hacen uso de sus dispositivos electrónicos para la visualización de partituras mientras se interpreta un obra. Sin embargo, uno de los principales problemas que encuentran es el de pasar las páginas mientras se interpreta la obra. Aprovechando el hecho de que en la actualidad la mayoría de dispositivos tienen integrada una cámara frontal, se propone la creación de un "pasa páginas" automático mediante el sistema desarrollado.

De esta forma, podemos utilizar la monitorización la pose de la cabeza de forma que, inclinando rápidamente la cabeza hacia la derecha la aplicación pase de página automáticamente y, inclinándola en el sentido contrario, vuelva a la página anterior.

Para ello, en primer lugar estimaremos la pose de la cabeza mediante la función implementada `analyze(image,target=['pose'])`. Una vez estimada, pasará la rotación estabilizada a grados mediante la función `rotvec2euler(result["pose-stb"][1])`. Accediendo al segundo elemento de éste vector obtendremos la inclinación de la cabeza en grados.

En segundo lugar, una variable entera (`state`), definirá el estado de la cabeza teniendo como posibles valores los mostrados en la tabla 7.1.

Valor	Descripción
-1	la cabeza está inclinada con un ángulo menor a -20 grados
0	la cabeza tiene una inclinación entre -15 y +15 grados (se estimará como no inclinada)
1	la cabeza está inclinada con un ángulo mayor a +20 grados

Tabla 7.1: Tabla de posibles valores de la variable `state`.

Figura 7.1: Representación visual de la variable `state` en función de la inclinación de la cabeza.

De esta forma, cuando la variable `state` pase de 0 a 1 y vuelva a 0 en un tiempo menor a 0.8 segundos, se interpretará que el usuario desea pasar de página. Por el contrario, cuando `state` pase de 0 a -1 y vuelva a 0, se interpretará que el usuario desea volver atrás.

En la figura 7.2 se puede observar que, mediante el sistema implementado, el desarrollo de este caso de uso sólo supone 30 líneas de código.

```
from glob import glob
import cv2
from FaceMonitoring import FaceMonitoring
from time import time as now
fm = FaceMonitoring() # modulo gestor
fm.select_camera(0) # selecciona la primera camara disponible
images_names = glob(r"ruta\al\*.jpg") # carga los nombres de las imagenes
images = [cv2.imread(image_name) for image_name in images_names ]
idx , state , max_angle, min_angle = 0 , 0 , 20 , 15
t0 , min_time = now() , 0.8
while True:
    frame = fm.get_frame()
    result = fm.analyze(frame,target=['pose'])
    if result["pose"] is not None:
        trans , rot = result["pose_stb"]
        euler = fm.POSE_ESTIMATOR.rotvec2euler(rot)
        angle = euler[2]
        if state == 0 and angle >= max_angle: state = 1
        elif state == 0 and angle <= -max_angle: state = -1
        if state == 1 and -min_angle <= angle <= min_angle:
            if idx <= len(images) -1 and (now()-t0) < min_time:
                idx += 1
            state , t0 = 0 , now()
        if state == -1 and -min_angle <= angle <= min_angle and (now()-t0) <
            min_time:
            if idx > 0:
                idx -= 1
            state , t0 = 0 , now()
        if state == 0: t0 = now()
    cv2.imshow("",images[idx])
    cv2.waitKey(1)
```

Figura 7.2: Código de la aplicación `pasa páginas`.

7.2 Monitorización del sueño en la conducción

Tal y como afirma FESVIAL (Fundación para la Seguridad Vial), el sueño y la fatiga son la causa de hasta el 30 % de los accidentes de tráfico [45]. Por ello, otro de los posibles casos de uso del sistema sería la monitorización del sueño y la fatiga en la conducción.

Para ello, se ha desarrollado una simple aplicación demostrativa. La cual, mediante los gestos faciales estimados y la expresiones a través de éstos, es capaz de detectar el tiempo que el conductor cierra los ojos o bosteza.

Para ello, tal t como muestra la figura 7.3, se obtienen los gestos faciales y las emociones estimadas a través de éstos mediante la función `analyze( image , target = [ 'emotion' , 'action-units' ])`. Una vez estimados se calcula el intervalo de tiempo donde `result [ "action-units_stb" ][ "ear" ] < 0.35`, es decir, el intervalo de tiempo donde el conductor tiene los ojos cerrados en un 62 %. Posteriormente se comprueba si la expresión estimada es la de bostezar (`result["emotion" ]=='yawning'`). En este programa se muestra en la imagen un texto de alarma a modo de demostración. Sin embargo, en una aplicación real podría alertar al conductor mediante sonidos o incluso realizar una llamada de emergencia automática.

```
from FaceMonitoring import FaceMonitoring
import cv2
from time import time as now
fm = FaceMonitoring()
fm.select_camera(0)
time_with_closed_eyes_ini, max_time_with_closed_eyes = 0 , 1
eyes_closed = False
time_with_closed_eyes = 0
while(True):
    image = fm.get_frame()
    result = fm.analyze(image,target=['emotion','action-units'])
    if result["action-units"] is not None:
        if result["action-units_stb"]["ear"] < 0.35:
            if not eyes_closed:
                time_with_closed_eyes_ini = now()
                eyes_closed = True
            else:
                time_with_closed_eyes = now() - time_with_closed_eyes_ini
                if time_with_closed_eyes >= max_time_with_closed_eyes:
                    cv2.putText(image,"DANGER! WAKE UP!!",(50,50),cv2.
                        FONT_HERSHEY_COMPLEX,1,(0,0,255))
        else:
            time_with_closed_eyes_ini = now()
            time_with_closed_eyes = 0
            eyes_closed = False
    if result["emotion"] is not None:
        if result["emotion"] == 'yawning':
            cv2.putText(image,"DANGER! WAKE UP!!",(50,50),cv2.
                FONT_HERSHEY_COMPLEX,1,(0,0,255))
        print(result["emotion"])
    cv2.imshow("Asistente de conducci n" , image)
    cv2.waitKey(1)
```

Figura 7.3: Código de la aplicación de monitorización del sueño en la conducción.

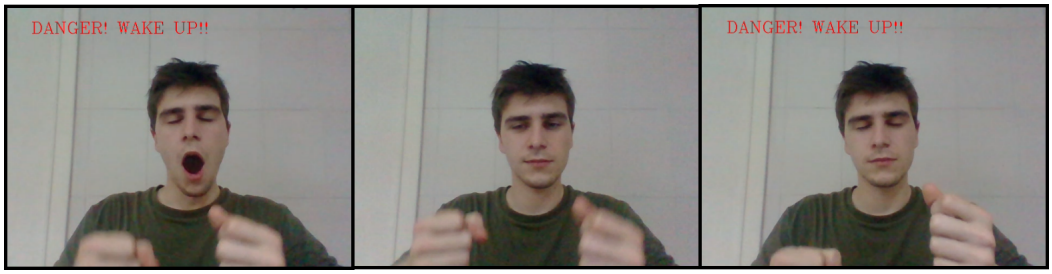

Figura 7.4: Visualización del programa definido en la figura 7.3

7.3 Separación de conjuntos de imágenes por identificación facial

Actualmente, multitud de dispositivos, incluyen la funcionalidad de separar la galería de fotos mediante algoritmos de identificación facial de forma que el conjunto de imágenes se separe por el sujeto que aparezca en éstas.

El sistema desarrollado también será capaz de implementar esta utilidad mediante el uso del módulo `Face Monitorning` en tan solo 27 líneas de código, tal y como se muestra en la figura 7.5. Para ello, en primer lugar creamos dos listas vacías (línea 9). La primera de ellas (`codificaciones`), guardará las nuevas codificaciones encontradas mientras que en la segunda (`rostros_divididos`) se almacenarán las imágenes de entrada ya divididas, por lo que al final de la ejecución la lista `rostros_divididos` será un lista de listas de imágenes divididas por los rostros identificados.

En segundo lugar debemos hacer que el módulo de identificación facial estime la codificación del rostro detectado en todos los *frames* (líneas 7-8). Tras ello, recorremos todas las imágenes obteniendo en cada una su respectiva codificación (línea 12).

Una vez tengamos la codificación facial (`codificacion`) recorreremos la lista de codificaciones faciales encontradas hasta el momento y las compararemos con la actual (línea 16) obteniendo la distancia entre éstas (`dist`). Si la distancia es menor a 0.6 unidades podemos afirmar que se trata del mismo sujeto por lo que, posteriormente, añadiremos la imagen a la sublista de imágenes del sujeto (línea 20). En caso contrario crearemos una sublista nueva que contenga la imagen actual y añadiremos la codificación facial a la lista `codificaciones` (líneas 22-23).

Por último, se guardarán las imágenes separadas por identificación facial en carpetas distintas (una para cada sujeto), tal y como muestran las líneas 24-27.

```
from FaceMonitoring import FaceMonitoring
import glob, os, cv2

relative_path = os.path.dirname(__file__)
images = glob(relative_path + 'ruta/a/tu/carpeta/de/imagenes/*.*')
fm = FaceMonitoring()
fm.FACE_RECOGNITION.onlyOnce = False
fm.FACE_RECOGNITION.frame_skip = 1
codificaciones , rostros_divididos= [] , []
for image in images:
    img = cv2.imread(image)
    codificacion = fm.analyze(img,target=['face-encodings'])["face-encodings"]
    if codificacion is not None:
        encontrado = False
        for idx , cod in enumerate(codificaciones):
            dist = fm.FACE_RECOGNITION.compare(cod,codificacion)
            if dist < 0.6:
                encontrado = True
                break
        if encontrado: rostros_divididos[idx].append(img)
        else:
            codificaciones.append(codificacion)
            rostros_divididos.append([img])
for idx , rostro_dividido in enumerate(rostros_divididos):
    os.mkdir(f"{relative_path}/rostros_divididos/{idx}")
    for idx2, rostro in enumerate(rostro_dividido):
        cv2.imwrite(f"{relative_path}/rostros_divididos/{idx}/{idx2}.jpg" ,
            rostro)
```

Figura 7.5: Código de la aplicación de separación de conjuntos de imágenes por identificación facial.

Cabe destacar que, mediante una aproximación similar, podrían separarse conjuntos de imágenes por otras características faciales. Por ejemplo, podríamos separar las fotos dónde el sujeto mira a la izquierda o a la derecha con la posición de la cabeza y la dirección de la mirada, separar por emociones, gestos, edad o género. De forma que esta utilidad podría resultar interesante para la creación de conjuntos de datos con una distribución uniforme de poses, gestos faciales, edad, etc.

Además, también podrían clasificarse vídeos según el porcentaje de felicidad (tiempo en el que la emoción estimada sea felicidad) u otros parámetros distintos.

7.4 Otros casos de uso

7.4.1. Porcentaje de satisfacción

En numerosos sectores del mercado, la satisfacción del cliente es uno de los principales objetivos. En este sentido, un posible caso de uso del sistema podría ser la monitorización de las emociones de los clientes para conocer el "porcentaje de satisfacción" que obtiene el producto.

Por ejemplo, la reacción del público ante un anuncio publicitario, un escaparate o a lo largo de una película o serie podría ser monitorizada mediante el uso del módulo `Emotion Detector CNN` para aproximar el grado de felicidad, asombro o intriga que genera el producto en cuestión.

7.4.2. Aplicaciones para personas con discapacidad funcional

Otra de las posibles aplicaciones del sistema sería la creación de herramientas útiles para las personas con algún tipo de discapacidad funcional. Por ejemplo, podría crearse una aplicación para mover el cursor mediante la pose de la cabeza y hacer *click* mediante algún gesto como pestañear dos veces o subir las cejas. También podría ser útil crear una interfaz para escribir mediante el uso de gestos faciales y el auto-completado de palabras.

CAPÍTULO 8
Conclusiones y trabajos futuros

Tal y como se expone en la sección de objetivos del proyecto (1.2), el propósito principal de este proyecto era la creación de un sistema de monitorización facial a tiempo real y robusto a modo de prototipo con el fin de explorar las diferentes herramientas y técnicas disponibles actualmente para la resolución de este problema. Además, la idea principal subyacente al desarrollo del sistema era la de extender la cantidad de información que se puede extraer de un rostro integrando diferentes tecnologías de campos como el *Machine Learning*, el *Deep Learning* o la Visión por Computador.

En este sentido, podemos afirmar que el desarrollo del sistema ha llevado a una solución que cumple los requisitos propuestos y que, además, unifica distintos algoritmos y técnicas del Estado del Arte. A través de éste, se ha podido llegar a diversas conclusiones acerca de la efectividad de los distintos algoritmos existentes para la detección, identificación y extracción de diversos parámetros relativos al rostro humano.

Una de las conclusiones que hemos podido observar durante el estudio y desarrollo del proyecto es que el uso de técnicas de *Deep Learning* para la monitorización facial en tiempo real puede suponer un coste computacional elevado a considerar. Sin duda el campo del *Deep Learning* supone un importante avance en el campo aportando resultados cada vez más sorprendentes y resolviendo problemas complejos con precisión.

Sin embargo, la otra cara de la moneda muestra que todavía queda mucho camino por recorrer en lo relativo al coste temporal que supone el uso de modelos de este campo como las redes neuronales convolucionales o CNN's dado que, como se ha podido observar a lo largo de la memoria, el uso de CNN's puede suponer una caída de FPS considerable frente a otro tipo de modeles de Aprendizaje Automático como los Árboles de regresión o las SVM, los cuales sí obtienen buenos resultados a tiempo real. Tal vez, el avance, tanto del campo Aprendizaje Profundo como del nuevo *Hardware* desarrollado, permita en un futuro cercano que el uso de este tipo de modelos no suponga un coste temporal tan significativo en CPU.

En lo respectivo a la robustez del sistema, podemos concluir que el uso de el filtro de Kalman para la estabilización ha aportado buenos resultados solucionando así parte del problema de la alta variabilidad relativa a la extracción de los *landmarks*. Además, la funcionalidad añadida de predecir nuevos estados de las mediciones cuantitativas ante fallos en la detección del rostro a aportado un extra al sistema a considerar y, desde nuestra humilde opinión, a tener en cuenta para el desarrollo de sistemas de monitorización a tiempo real. De hecho, una de las funcionalidades que, tras lo aprendido, consideramos más importantes a la hora de implementar un sistema de monitorización a tiempo real, es el uso de técnicas clásicas o de procesamiento de imágenes como "segunda opción" ante fallos imprevistos de los sub-sistemas basados en *Machine Learning*. Ésto aporta un extra de robustez al sistema que puede ser interesante para el problema que nos concierne.

De hecho, en el artículo "*Real-Time Head Pose Estimation by Tracking and Detection of Keypoints and Facial Landmarks*" [2], se propone un sistema de estimación de la pose de la cabeza que fusiona el uso de los *landmarks*, los filtros de Kalman y el *Optical Flow* que resulta interesante dado que en caso de un fallo en la detección del rostro y, por tanto, en la extracción de los *landmarks*, el sistema es capaz de seguir estimando la pose siguiendo los puntos de referencia mediante el algoritmo de *Optical flow*, por lo que puede resultar una opción a considerar para futuros trabajos relacionados con la monitorización facial.

En cuanto a la estimación de la mirada, uno de los principales puntos débiles del subsistema encargado de dicha tarea es el de que el parámetro *threshold* deba ser configurado manualmente por el usuario. Lo cual nos hace reflexionar que, en futuros trabajos, el uso de técnicas de auto *thresholding* como el algoritmo de Otsu's [35] es una cuestión a considerar y estudiar.

Otro de los aspectos a considerar es el del efecto del algoritmo de detección utilizado en la posterior extracción de los *landmarks*. Hemos podido observar que el algoritmo CNN `cvlib` obtiene mejores resultados que el HOG+SVM `Dlib` ante ángulos de rotación de la cabeza grandes. Sin embargo, la extracción de los *landmarks* resulta menos efectiva dado que el algoritmo de extracción basado en Árboles de Regresión ha sido entrenado junto al algoritmo de detección de `Dlib`. Por lo que otro de los trabajos futuros a considerar es el de entrenar el modelo de extracción de *landmarks* junto al algoritmo de detección de `cvlib` y estudiar la calidad del resultado. En caso de obtener una buena efectividad en dicho estudio podría suponer un avance de robustez del sistema dado que la totalidad de los parámetros faciales a estimar requiere de la previa detección del rostro.

En cuanto a la estimación de emociones mediante el uso de redes neuronales convolucionales, resultaría de gran interés la evaluación de ésta mediante algún *dataset* disponible en la *web*, por ejemplo el "Emotion Detection From Facial Expressions" de la plataforma *Kaggle* [16]. En el caso de la estimación de emociones mediante los gestos faciales, podemos concluir que la estimación mediante los *landmarks* puede resultar una aproximación interesante. En el artículo "Facial Expression Recognition Using Facial Landmarks and Random Forest Classifier" [31], se muestra como, mediante la extracción de características a partir de los *landmarks* extraídos y su posterior análisis mediante modelos de *Machine Learning*, se obtiene hasta un 90 % de eficacia en la estimación de emociones, por lo que dicha aproximación podría resultar de gran interés en futuros trabajos.

Por último, como se ha podido apreciar en el capítulo 7, el desarrollo del sistema ha llevado de forma natural a la creación de un conjunto de módulos `Python` de alto nivel, altamente configurables y de fácil uso. Por lo que uno de los principales futuros trabajos propuestos es el la creación de una librería específica para la monitorización facial en tiempo real y la extracción de información relativa al rostro partiendo del sistema de módulos ya implementado aportando nuevas y mejoradas técnicas útiles para dicho problema e incluso portar el sistema a un lenguaje de programación compilado como `C++`.

8.1 Relación del trabajo desarrollado con los estudios cursados

No cabe duda de que lo aprendido en el grado de Ingeniería Informática ha resultado de gran ayuda para el desarrollo y el estudio de la temática propuesta. La asignaturas de Aprendizaje Automático y Técnicas, entornos y aplicaciones de inteligencia artificial han aportado una introducción teórica al campo de gran ayuda para la comprensión de conceptos tan complejos como las SVM, las redes neuronales y otros modelos de *Machine Learning* además de la evaluación de éstos y para discernir sus diferencias y peculiaridades. Por otro lado, la asignatura de Percepción ha sido de gran ayuda para la comprensión de diversas técnicas de procesamiento de datos y concretamente de imágenes. En general, el conjunto de todas las asignaturas cursadas ha aportado una perspectiva más completa a la comprensión del campo de la informática.

Bibliografía

[1] *Age and Gender Classification Using Convolutional Neural Networks*. URL: https://talhassner.github.io/home/publication/2015_CVPR.

[2] Jilliam Maria Diaz Barros. *Real-Time Head Pose Estimation by Tracking and Detection of Keypoints and Facial Landmarks*. URL: https://www.researchgate.net/publication/334641917_Real-Time_Head_Pose_Estimation_by_Tracking_and_Detection_of_Keypoints_and_Facial_Landmarks.

[3] Jing chun Cheng. *All Age Faces Dataset*. URL: https://github.com/JingchunCheng/All-Age-Faces-Dataset.

[4] *Classification and regresion trees*. URL: https://www.routledge.com/Classification-and-Regression-Trees/Breiman-Friedman-Stone-Olshen/p/book/9780412048418.

[5] *CNN Based Face Recognition in Dlib*. URL: http://dlib.net/cnn_face_detector.py.html.

[6] *Contours in OpenCV*. URL: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html.

[7] *cvlib*. URL: https://www.cvlib.net/.

[8] Davisclick. *Haar Filters*. URL: https://es.wikipedia.org/wiki/Archivo:Aplicaci%5C%C3%5C%B3n_de_filtros_de_Haar.png.

[9] *Deep Face Documentation*. URL: https://pypi.org/project/deepface/.

[10] *Definición de monitorización según la RAE*. URL: https://dle.rae.es/monitorizar.

[11] Jonathan DEKHTIAR. *Why convolutions always use odd-numbers as filter$_s$ize*. URL: https://datascience.stackexchange.com/questions/23183/why-convolutions-always-use-odd-numbers-as-filter-size.

[12] *Demonstration of Facial Emotion Recognition on Real Time Video Using CNN : Python Keras*. URL: https://appliedmachinelearning.blog/2018/11/28/demonstration-of-facial-emotion-recognition-on-real-time-video-using-cnn-python-keras/.

[13] *Desarrollo de un sistema de reconocimiento facial utilizando Deep Learning con OpenCV*. URL: https://riunet.upv.es/handle/10251/156694.

[14] *Dlib HOG+SVM Detector*. URL: http://dlib.net/train_object_detector.py.html.

[15] *Dlib Landmark Predictor*. URL: http://dlib.net/face_landmark_detection.py.html.

[16] *Emotion Detection From Facial Expressions*. URL: https://www.kaggle.com/c/emotion-detection-from-facial-expressions/data.

[17] *Eye Aspect Ratio(EAR) and Drowsiness detector using dlib*. URL: https://medium.com/analytics-vidhya/eye-aspect-ratio-ear-and-drowsiness-detector-using-dlib-a0b2c292d706.

[18] *Face recognition and Face detection using the OpenCV*. URL: https://www.javatpoint.com/face-recognition-and-face-detection-using-opencv.

[19] *Face Recognition Python*. URL: https://pypi.org/project/face-recognition/.

[20] *FaceNet Python*. URL: https://github.com/timesler/facenet-pytorch.

[21] *Google Colab*. URL: https://colab.research.google.com/notebooks/welcome.ipynb?hl=es.

[22] Justin Güse. *Detección de rostros basada en TensorFlow y MTCNN*. URL: https://programmerclick.com/article/1680569525/.

[23] *Head Pose Estimation*. URL: https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/.

[24] *Histograms of Oriented Gradients for Human Detection*. URL: N%20Dalal.

[25] T S Huang. *Computer Vision: Evolution and Promise*. URL: https://cds.cern.ch/record/400313/files/p21.pdf.

[26] *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks*. URL: https://arxiv.org/abs/1604.02878.

[27] Pavel Korshunov. *Speaker Inconsistency Detection in Tampered Video*. URL: https://www.researchgate.net/figure/The-68-landmarks-detected-by-dlib-library-This-image-was-created-by-Brandon-Amos-of-CMU_fig2_329392737.

[28] Keng-Cheng Liu. *Real-Time Facial Expression Recognition Based on CNN*. URL: https://ieeexplore.ieee.org/document/8823409.

[29] Satya Mallick. *Speeding up Dlib's Facial Landmark Detector*. URL: https://learnopencv.com/speeding-up-dlib-facial-landmark-detector/.

[30] AZMATH MOOSA. *HOW FACIAL RECOGNITION SYSTEMS WORK*. URL: https://www.baseapp.com/computer-vision/how-face-recognition-systems-work/.

[31] Nuwan Munasinghe. *Facial Expression Recognition Using Facial Landmarks and Random Forest Classifier*. URL: https://www.researchgate.net/publication/325674764_Facial_Expression_Recognition_Using_Facial_Landmarks_and_Random_Forest_Classifier.

[32] T. Ojala. *Performance evaluation of texture measures with classification based on Kullback discrimination of distributions*. URL: https://ieeexplore.ieee.org/document/576366/authors#authors.

[33] *One Millisecond Face Alignment with an Ensemble of Regression Trees*. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Kazemi_One_Millisecond_Face_2014_CVPR_paper.pdf.

[34] *OpenCV Cascade Classifier*. URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[35] *Otsus Binarization*. URL: https://docs.opencv.org/4.5.2/d7/d4d/tutorial_py_thresholding.html.

[36] *pickle documentation*. URL: https://docs.python.org/3/library/pickle.html.

[37] *Pinhole Camera Model OpenCV*. URL: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html.

[38] *Real-time Age, Gender and Emotion Prediction from Webcam with Keras and OpenCV*. URL: https://towardsdatascience.com/real-time-age-gender-and-emotion-prediction-from-webcam-with-keras-and-opencv-bde6220d60a.

[39] *Real-time eye tracking using OpenCV and Dlib*. URL: https://towardsdatascience.com/real-time-eye-tracking-using-opencv-and-dlib-b504ca724ac6.

[40] *Real-time eye tracking using OpenCV and Dlib*. URL: https://towardsdatascience.com/real-time-eye-tracking-using-opencv-and-dlib-b504ca724ac6.

[41] *Real-Time Head Pose Estimation With OpenCV and Dlib*. URL: https://medium.com/analytics-vidhya/real-time-head-pose-estimation-with-opencv-and-dlib-e8dc10d62078.

[42] Christos Sagonas. *Facial point annotations*. URL: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/.

[43] Kelvin Salton. *Implement Local Binary Pattern Descriptor from scratch*. URL: https://fairyonice.github.io/implement-lbp-from%5C%20scratch.html.

[44] *The Seminal Kalman Filter Paper (1960)*. URL: https://www.cs.unc.edu/~welch/kalman/kalmanPaper.html.

[45] *Tkinter documentation*. URL: https://docs.python.org/es/3/library/tkinter.html.

[46] *Topological Structural Analysis of Digitized Binary Images by Border Following*.

[47] Melda Ulusoy. *Understanding Kalman Filters Part 3: An Optimal State Estimator*. URL: https://es.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator--1490710645421.html.

[48] Paul Viola y Michael Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*. URL: https://ieeexplore.ieee.org/document/990517.

[49] Wang Xun. *Detecting Worms via Mining Dynamic Program Execution*. URL: https://www.researchgate.net/publication/4344800_Detecting_Worms_via_Mining_Dynamic_Program_Execution.

APÉNDICE A
Especificaciones de la máquina utilizada

Comptadora	OMEN HP LAPTOP-2TKKT7BG
Procesador	Intel Core i5-8330H CPU @ 2.3GHz 2.3GHz
RAM	16 GB
Sistema operativo	Windows 10 Home (2004)
Gráfica	NVIDIA GFORCE GTX 1050

Tabla A.1: Especificaciones de la máquina utilizada.

APÉNDICE B
Procesamiento de imágenes

El procesamiento de imágenes es un pilar fundamental dentro del campo de la visión por computador. Se trata de un conjuto de técnicas aplicadas a imágenes digitales con el fin de extraer información de éstas o de facilitar su posterior análisis. Existe una gran cantidad de técnicas de procesamiento de imágenes, aplicadas tanto en el dominio del espacio, es decir, de la estructura, como en el dominio de la frecuencia, el cual hace referencia a los canales (colores) de una imagen. Si embargo, en este apartado se comentarán brevemente algunas de las técnicas utilizadas para el desarrollo de este proyecto. Las cuales resultaron especialmente útiles en la tarea de estimar la dirección de la mirada.

B.0.1. *Thresholding*

El proceso de *thresholding* se aplica para convertir imágenes de un sólo canal en una imagen binaria,es decir, cuyos píxeles tienen dos estados posibles (blanco y negro). Para ello simplemente se aplica un parámetro (*threshold* o umbral) de forma que los píxeles que se encuentren por debajo de esa intensidad pasarán a negro mientras que los que se encuentren por encima pasarán a ser blancos.

B.0.2. Erosión

El proceso de erosión es típicamente aplicado sobre imágenes binarias. El efecto de este proceso sobre una imagen de estas características es el de, como su nombre indica, erosionar los límites de las regiones de píxeles activos.

B.0.3. Dilatación

Este proceso es, básicamente, el inverso que el de erosión. Al contrario que éste, su efecto es el de aumentar los límites de las regiones de píxeles activos.

B.0.4. Desenfoque por filtro mediano

Este proceso es utilizado comúnmente para eliminar ruido de una imagen o señal. Para ello, el algoritmo reemplaza cada uno de los píxeles por la mediana de sus píxeles vecinos. El patrón que describe dichos vecinos viene dado por un parámetro definido como ventana.

B.0.5. Detección de contornos

La detección de contornos es un proceso aplicado sobre imágenes digitales por el cual se extraen las regiones de píxeles con el mismo color o intensidad. Para ello, este proceso hace uso del algoritmo presentado en el paper *Topological Structural Analysis of Digitized Binary Images by Border Following* [46] implementado en la librería `OpenCV` [6]

B.0.6. *Blob From Image*

Blob From Image es una técnica de procesamiento de imágenes ampliamente utilizada en el campo del *Deep Learning* y las CNN's. Normalmente es utilizada para preprocesar una imagen antes de pasarla como entrada a una red neuronal, típicamente una CNN.

Este proceso consta de dos fases principales:

1. **Substracción de la media**: Para cada uno de los canales de la imagen (RGB), el algoritmo calcula su respectiva media y la substrae de cada canal.
2. **Normalización**: Tras la substracción de la media para cada canal, éstos se normalizan para representar sus valores entre 0 y 1.