Detector de emociones mediante análisis de fotografías

Alumno: Miguel Canteras Cañizares

Dirigido por: Àgata Lapedriza Garcia
David Masip Rodó

Trabajo Final de Máster – Visión por Computador y reconocimiento estadístico de patrones

Máster universitario en Ingeniería Computacional y Matemática
Curso 2015-16, 2º semestre

Hay expresiones de la cara características que son observables para acompañar la cólera, el miedo, la excitación erótica, y todas las otras pasiones.

Aristóteles

Si queremos que los ordenadores sean realmente inteligentes y que interaccionen de forma natural con nosotros, tenemos que otorgarles la capacidad de reconocer, entender e incluso tener y expresar emociones.

Rosalind Picard

FICHA DEL TRABAJO FINAL

Título del trabajo:	Detector de emociones mediante análisis de fotografías
Nombre del autor:	Miguel Canteras Cañizares
Nombre de los consultores:	Àgata Lapedriza Garcia David Masip Rodó
Fecha de entrega:	06/2016
Área del Trabajo Final:	Visión por Computador y reconocimiento estadístico de patrones
Titulación:	Máster universitario en Ingeniería Computacional y Matemática

Resumen del Trabajo:

La especie humana, como animal social que es, es capaz de expresar las emociones de diversas formas. Una de las más importantes es mediante expresiones faciales. Una expresión puede contener codificada una gran cantidad de información referente a las emociones, sensaciones y sentimientos que una persona está experimentando.

Las emociones básicas que se pueden observar en una expresión son alegría, ira, miedo, asco, sorpresa y tristeza. Además existen otras emociones observables, entre ellas desprecio.

Para codificar los diferentes movimientos faciales que componen una expresión existe un sistema estándar conocido como Facial Action Coding System (FACS). Mediante este sistema se puede reconocer cada movimiento mediante un código o Action Unit(AU).

El propósito del presente proyecto es crear un detector que permita identificar, de manera automatizada, la emoción básica que aparece en la fotografía de una cara. Además indicará qué Action Units se encuentran presentes en la cara. El detector deberá identificar la cara dentro de la fotografía, crear un descriptor de características mediante el algoritmo Local Binary Patterns (LBP) y clasificar la emoción presente como una de las básicas mediante el algoritmo de aprendizaje supervisado Support Vector Machines (SVM).

Abstract:

The human species, as a social animal, is able to express emotions in different ways. One of the most important through facial expressions. An expression can code a lot of information about emotions, sensations and feelings that a person is experiencing.

Basic emotions that can be observed in an expression are happiness, anger, fear, disgust, surprise and sadness. In addition there are other observable emotions, including contempt.

To encode different facial movements that form an expression there is a standard system known as Facial Action Coding System (FACS). With this system every movement can be recognized using a code or Action Unit(AU).

The purpose of this project is to develope a detector to identify, in an automated way, the basic emotion appearing in the photograph of a face. It will point also the Action Units present on the face. The detector must identify the face in the photograph, create a feature descriptor using the Local Binary Patterns algorithm (LBP) and classify the emotion found as one of the basics using the Support Vector Machines (SVM) supervised learning algorithm .

Palabras clave:

Emociones, action units, detector, cara, fotografías

Índice de contenidos

Índice de ilustraciones

1. Descripción y plan del proyecto

1.1 Descripción del proyecto

1.1.1 Descripción

Una gran mayoría de personas son capaces de identificar las emociones que transmiten otras personas a través de expresiones faciales. El cerebro humano es una herramienta que, durante miles de años, ha ido desarrollando y evolucionando dicha capacidad hasta el punto de parecer una habilidad muy sencilla de realizar. Pero cuando se intenta implementar dicha capacidad mediante un sistema automatizado la complejidad comienza a emerger, y es necesaria la utilización de técnicas avanzadas para conseguir una tasa de reconocimiento igual o superior a la humana. Estas técnicas suelen pertenecer a campos como la visión por computador, el reconocimiento de patrones, la estadística o la inteligencia artificial.

Un buen reconocedor de emociones es una herramienta informática muy útil para muchos campos, entre ellos la medicina y la psicología (investigación de trastornos como autismo, Parkinson o la epilepsia), la educación (análisis de atención de estudiantes siguiendo una clase online) o el marketing (interés de las personas ante un anuncio o producto).

El presente proyecto consiste en la creación de un detector de emociones ejecutable desde entorno web. Para ello se creará una página web que permitirá, de forma sencilla y amigable, seleccionar una foto desde diferentes fuentes (subida desde ordenador, cámara, enlace a otras webs) y analizarla para identificar las emociones mostradas por la cara que aparece en la misma. Para conseguir la identificación, se implementarán una serie de servicios que ejecutarán diferentes algoritmos para conseguir dicha funcionalidad.

1.1.2 Objetivos

El objetivo de este trabajo es implementar un detector de emociones. Con la consecución de dicho detector se conseguirán los siguientes objetivos.

1.1.2.1 Objetivos generales

- Diseñar e implementar los servicios necesarios para detectar las emociones en una fotografía.
- Diseñar e implementar la web de acceso a dichos servicios por los usuarios.

1.1.2.2 Objetivos específicos

- Analizar la arquitectura necesaria para realizar el detector.
- Seleccionar los lenguajes de programación y librerías a utilizar.
- Implementar la interfaz gráfica de la web.
- Implementar un servicio de detección de puntos característicos en las caras.
- Implementar un servicio de normalizado de la cara detectada.
- Implementar servicios de detección de vectores de características.
- Crear y entrenar, si es necesario por el tipo de clasificador, clasificadores de los vectores de características que permitan clasificar la cara según las emociones detectadas en la misma.
- Analizar los resultados obtenidos por los diversos algoritmos.

1.1.3 Alcance

Se considera dentro del alcance del proyecto:

- Memoria del trabajo.
- Código fuente de todos los componentes.
- Vídeo de la presentación virtual.
- Participación en el debate virtual.

1.2. Organización

1.2.1 Relación de actividades

Las actividades del proyecto se han agrupado en las siguientes entregas. Se indican las horas resultantes del análisis de tareas y el número de hojas de la memoria previstas para cada sección:

PEC1	Plan de trabajo	7 horas	9 hojas
PEC2	Detección de emociones básicas	146 horas	33 hojas
PEC3	Detección de Action Units y clasificación mediante Deep Learning	120 horas	33 hojas
PEC4	Memoria final, presentación y defensa del proyecto	50 horas	15 hojas

1.2.1.1 PEC1

Reunión preliminar		**Horas** 2
Inicio 28/12/15	**Final** 28/12/15	**Hojas** 0
Reunión preliminar con los directores del trabajo para definir el proyecto.		

Elaboración del plan de trabajo		**Horas** 5
Inicio 3/03/16	**Final** 12/03/16	**Hojas** 9
Planificación del proyecto y elaboración del plan de trabajo.		

Revisión del plan de trabajo		**Horas** 2
Inicio 13/3/16	**Final** 15/3/16	**Hojas** 0
Revisión del documento de plan de trabajo, añadiendo las modificaciones apropiadas.		

Entrega PEC1		**Horas** 0
Inicio 16/03/16	**Final** 16/03/16	**Hojas** 0
Entrega de la PEC1.		

1.2.1.2 PEC2

Búsqueda de soluciones tecnológicas		**Horas** 10
Inicio 17/03/16	**Final** 18/03/16	**Hojas** 2
Búsqueda de las soluciones tecnológicas para el desarrollo de la web y los servicios.		

Selección solución tecnológica, arquitectura y fuentes de datos		**Horas** 10
Inicio 19/03/16	**Final** 20/03/16	**Hojas** 3
Selección justificada de las tecnologías a utilizar, la arquitectura completa del sistema y las fuentes de datos de fotos de emociones utilizadas para la implementación del sistema.		

Implementación de la web		**Horas** 20
Inicio 21/03/16	**Final** 25/03/16	**Hojas** 3
Desarrollo de la web que permitirá a los usuarios el acceso al sistema.		

Implementación del detector de Landmarks		**Horas** 12
Inicio 26/03/16	**Final** 28/03/16	**Hojas** 2
Desarrollo del detector de puntos característicos (Landmarks) de la cara en la fotografía.		

Normalizado de la foto		**Horas** 12
Inicio 29/03/16	**Final** 31/03/16	**Hojas** 2
Desarrollo del algoritmo para normalizar la fotografía. Deberá escalar, cortar y centrar la cara, tomando como referencia los ojos.		

Algoritmo LBP		**Horas** 12
Inicio 1/04/16	**Final** 3/04/16	**Hojas** 2
Implementación del algoritmo Local Binary Patterns (LBP) sobre diferentes regiones de la cara para obtener un vector de características.		

Implementación SVM		**Horas** 16
Inicio 4/04/16	**Final** 7/04/16	**Hojas** 2
Desarrollo de Suport Vector Machines (SVM) para clasificar las emociones a partir del vector de características.		

Instalación del servidor		**Horas** 4
Inicio 8/04/16	**Final** 8/04/16	**Hojas** 5
Instalación de todos los resultados anteriores en el servidor de desarrollo		

Clasificación y pruebas emociones básicas		**Horas** 30
Inicio 9/04/16	**Final** 16/04/16	**Hojas** 2
Ejecutar diversos clasificadores variando los parámetros de los diferentes algoritmos.		

Análisis de resultados emociones básicas		**Horas** 10
Inicio 17/04/16	**Final** 19/04/16	**Hojas** 10
Análisis de los resultados obtenidos.		

Revisión PEC2		**Horas** 10
Inicio 20/04/16	**Final** 22/04/16	**Hojas** 0
Revisión del documento de la PEC2, añadiendo las modificaciones apropiadas.		

Entrega PEC2		**Horas** 0
Inicio 23/04/16	**Final** 23/04/16	**Hojas** 0
Entrega de la PEC2.		

1.2.1.3 PEC3

Modificación Action Units		**Horas** 16
Inicio 24/04/16	**Final** 27/04/16	**Hojas** 3
Modificación de los diferentes algoritmos para detectar Action Units en lugar de emociones básicas.		

Clasificación y pruebas Action Units		**Horas** 30
Inicio 28/04/16	**Final** 5/05/16	**Hojas** 5
Ejecutar diversos clasificadores variando los parámetros de los diferentes algoritmos.		

Análisis de resultados Action Units		**Horas** 10
Inicio 6/05/16	**Final** 8/05/16	**Hojas** 10
Análisis de los resultados obtenidos.		

Deep Learning		**Horas** 44
Inicio 9/05/16	**Final** 19/05/16	**Hojas** 5
Implementación de un clasificador sencillo mediante aprendizaje profundo (Deep Learning). Esta tarea, junto con la siguiente, podrá ser cancelada o reducida en caso de necesitar más tiempo por la aparición de un riesgo (ver sección de riesgos).		

Análisis de resultados Deep Learning		**Horas** 10
Inicio 20/05/16	**Final** 21/05/16	**Hojas** 10
Análisis de los resultados obtenidos.		

Revisión PEC3		**Horas** 10
Inicio 22/05/16	**Final** 24/05/16	**Hojas** 0
Revisión del documento de la PEC3, añadiendo las modificaciones apropiadas.		

Entrega PEC3		**Horas** 0
Inicio 25/05/16	**Final** 25/05/16	**Hojas** 0
Entrega de la PEC3.		

1.2.1.4 PEC4

Memoria del proyecto		**Horas** 20
Inicio 26/05/16	**Final** 3/06/16	**Hojas** 25
Finalización de la memoria del proyecto que se ha ido desarrollando durante el trabajo.		

Presentación virtual		**Horas** 20
Inicio 4/06/16	**Final** 12/06/16	**Hojas** 20 (diapositivas)
Creación de la presentación virtual que se presentará al tribunal de evaluación para valorar el proyecto realizado.		

Revisión PEC4		**Horas** 10
Inicio 13/06/16	**Final** 16/06/16	**Hojas** 0
Revisión del documento de la PEC4, añadiendo las modificaciones apropiadas.		

Entrega PEC4		**Horas** 0
Inicio 17/06/16	**Final** 17/06/16	**Hojas** 0
Entrega de la PEC4.		

Defensa virtual del proyecto		**Horas** Por definir
Inicio Por definir	**Final** Por definir	**Hojas** 0
El tribunal formulará una serie de preguntas que se deberán contestar.		

1.2.2 Calendario

La ilustración 1 muestra el diagrama de Gantt del proyecto, con cada una de las tareas definidas y su desarrollo temporal. Para calcular las fechas de inicio y final de cada tarea se ha acordado dedicar 4 horas diarias, trabajando los 7 días de la semana.

	Nombre	Inicio	Terminado
1	Reunión preliminar	28/12/15 8:00	28/12/15 10:00
2	Hito externo 1: Inicio de curso	2/03/16 8:00	2/03/16 8:00
3	**PEC1**	**3/03/16 8:00**	**16/03/16 8:00**
4	Elaboración del plan de trabajo	3/03/16 8:00	12/03/16 17:00
5	Revisión del plan de trabajo	13/03/16 8:00	15/03/16 17:00
6	Entrega PEC1	16/03/16 8:00	16/03/16 8:00
7	**PEC2**	**17/03/16 8:00**	**23/04/16 8:00**
8	Búsqueda de soluciones tecnológicas	17/03/16 8:00	18/03/16 10:00
9	Selección solución tecnológica, arquitectura y fuentes de datos	19/03/16 8:00	20/03/16 17:00
10	Implementación de la web	21/03/16 8:00	25/03/16 17:00
11	Implementación del detector de Landmarks	26/03/16 8:00	28/03/16 17:00
12	Normalizado de la foto	29/03/16 7:00	31/03/16 17:00
13	Algoritmo LBP	1/04/16 7:00	3/04/16 17:00
14	Implementación SVM	4/04/16 7:00	7/04/16 17:00
15	Instalación del servidor	8/04/16 7:00	8/04/16 17:00
16	Clasificación y pruebas emociones básicas	9/04/16 7:00	16/04/16 17:00
17	Análisis de resultados emociones básicas	17/04/16 7:00	19/04/16 17:00
18	Revisión PEC2	20/04/16 7:00	22/04/16 17:00
19	Entrega PEC2	23/04/16 7:00	23/04/16 8:00
20	**PEC3**	**24/04/16 7:00**	**25/05/16 8:00**
21	Modificación Action Units	24/04/16 7:00	27/04/16 17:00
22	Clasificación y pruebas Action Units	28/04/16 7:00	5/05/16 17:00
23	Análisis de resultados Action Units	6/05/16 7:00	8/05/16 17:00
24	Deep Learning	9/05/16 7:00	19/05/16 17:00
25	Análisis de resultados Deep Learning	20/05/16 8:00	21/05/16 17:00
26	Revisión PEC3	22/05/16 8:00	24/05/16 17:00
27	Entrega PEC3	25/05/16 8:00	25/05/16 8:00
28	**PEC4**	**26/05/16 8:00**	**17/06/16 17:00**
29	Memoria del proyecto	26/05/16 8:00	3/06/16 17:00
30	Presentación virtual	4/06/16 8:00	12/06/16 17:00
31	Revisión PEC4	13/06/16 8:00	16/06/16 17:00
32	Entrega PEC4	17/06/16 8:00	17/06/16 17:00
33	Defensa virtual del proyecto	18/06/16 8:00	27/06/16 17:00
34	Hito externo 2: Cierre del proyecto	27/06/16 7:00	27/06/16 8:00

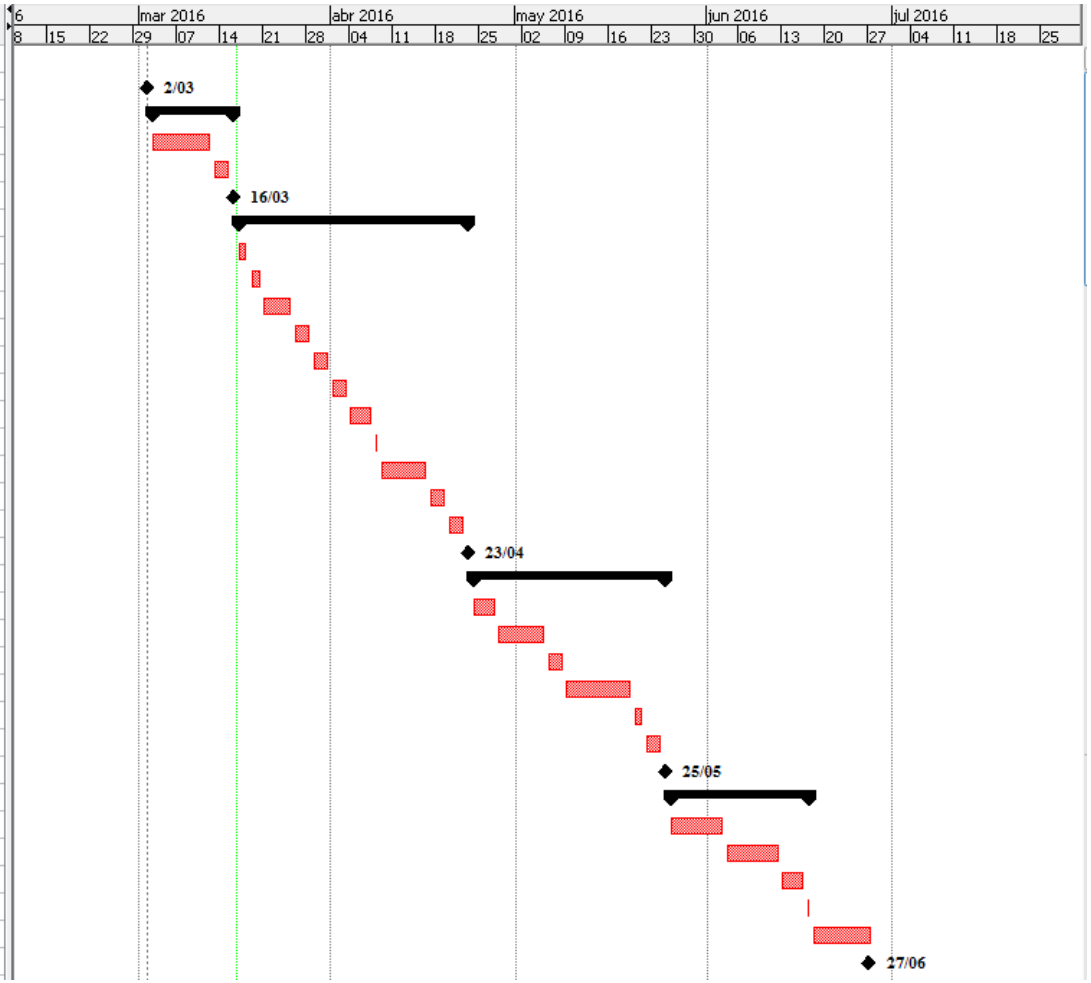


Ilustración 1. Diagrama de Gantt del proyecto

1.2.3 Hitos principales

Los hitos principales del proyecto son:

Fecha	Descripción
2 de febrero de 2016	Hito externo 1: Inicio de curso
16 de marzo de 2016	Entrega PEC1
23 de abril de 2016	Entrega PEC2
25 de mayo de 2016	Entrega PEC3
17 de junio de 2016	Entrega PEC4
27 de junio de 2016	Hito externo 2: Cierre del proyecto

1.3. Evaluación de material

1.3.1 Requerimientos de hardware

Para el desarrollo en local se utilizará un portátil Toshiba Satellite L755, con las siguientes especificaciones. Este equipo ya está disponible, no añadiendo coste alguno.

- CPU Intel® Core® i5-2410M 2.30 Ghz
- 8 GB de memoria RAM
- Disco duro Adata SP900 ATA 500 GB
- Tarjeta gráfica NVIDIA® GeForce GT® 525M, 1GB DDR3
- Conexión a Internet con fibra óptica 300 Mbps

Para la instalación del servicio se ha contratado un servidor virtual a la empresa OVH[1] con las siguientes características. El coste del servicio es de 1,99€/mes, con un coste total de 7,96€ (4 meses).

- Sistema operativo Ubuntu 15.04 Server (en versión 64 bits)
- Espacio en disco 10.7GB
- 1 GB de RAM
- 1 core

1.3.2 Requerimientos de software

A continuación se enumera el software que se usará para el desarrollo del proyecto, la confección de la memoria y el diseño de la presentación virtual. El software ya está disponible o es de código abierto y gratuito, por lo que no repercute en los costes.

1 https://www.ovh.es

- Microsoft Windows 7 SP1, 64 bits
- Ubuntu 14.04 LTS en una máquina virtual
- Ubuntu 15.04 Server (en versión 64 bits) en el servidor de instalación
- VMWare (Máquina virtual)
- LibreOffice Writer 5.1.3.2 (Redacción de documentación)
- LibreOffice Impress 5.1.3.2 (Presentación)
- ProjectLibre (Diagrama de Gantt)
- DIA (Diagrama UML)
- GIMP (Edición de imágenes)
- NGINX (Servidor web)
- JetBrains PyCharm 5.0.3 licencia estudiante (IDE Python)
- Anaconda (Herramientas Python)

1.4. Análisis de riesgos

Los posibles riesgos que pueden aparecer durante el proyecto son:

Riesgo	Problemas de disponibilidad
Descripción	Problemas de tiempo a dedicar al proyecto por enfermedades, viajes de trabajo, etc.
Impacto	Duración de las tareas.
Probabilidad	Alta
Acción	Dedicar más horas el fin de semana o eliminar o reducir tareas de Deep Learning.

Riesgo	Desconocimiento de algunas tecnologías
Descripción	Falta de experiencia con algunos de los algoritmos a utilizar que puede hacer necesitar más tiempo para algunas tareas de programación.
Impacte	Duración de las tareas
Probabilidad	Medio
Acción	Dedicar más tiempo a las tareas o eliminar o reducir tareas de Deep Learning.

Riesgo	Avería del hardware
Descripción	Avería del hardware utilizado para el desarrollo
Impacto	Duración del proyecto
Probabilidad	Baja
Acción	Substituir el hardware y reinstalar el software necesario.

1.5. Descripción de los capítulos de la memoria

Tras la descripción del proyecto realizada en el presente capítulo, el capítulo 2 introducirá el problema de la detección de emociones desde una perspectiva histórica y explicará algunos conceptos fundamentales como qué emociones se pueden detectar y qué son las *Action Units*.

Tras los capítulos introductorios se empezará finalmente con el desarrollo del detector. Lo primero que necesitaremos, y que se muestra en el capítulo 3, es detallar la arquitectura que se utilizará. Para ello se creará un modelo del sistema y se indicarán qué módulos son necesarios desarrollar y qué algoritmos utilizaremos en dichos módulos. También qué bases de datos se utilizarán para entrenar y probar el sistema.

Una vez identificados los algoritmos necesarios, el capítulo 4 explica los mismos en detalle.

Siguiendo la arquitectura y los algoritmos seleccionados se implementará el sistema, proceso que se explica en el capítulo 5.

Una vez terminado el desarrollo, se instalará en un servidor de demostración. El capítulo 6 muestra el proceso de instalación necesario para que el sistema esté operativo. Un proceso similar se debería realizar en la instalación para entornos de producción.

Con el sistema ya operativo, es necesario entrenar los algoritmos de clasificación. En el capítulo 7 se da una explicación de este entrenamiento y los resultados obtenidos por los diferentes clasificadores.

Finalmente el capítulo 8 detalla los resultados y conclusiones obtenidas del trabajo así como posibles ampliaciones futuras.

2. Introducción

2.1 Historia de la observación fisiológica de las emociones

2.1.1 Inicios

El ser humano ha observado cómo detectar las emociones desde la antigüedad. Para ello se han utilizado diversos métodos, siendo uno de los más habituales la observación de expresiones faciales y corporales.

Ya Aristóteles escribió:

> "Hay expresiones de la cara características que son observables para acompañar la cólera, el miedo, la excitación erótica, y todas las otras pasiones".

Después de Aristóteles todavía se tardó unos siglos en dar un impulso final al estudio de las emociones y poder considerar que se convirtió en ciencia. En el siglo XIX, un obrero de los ferrocarriles llamado Phineas Gage[2], tuvo un accidente (el 13 de septiembre de 1848) que le produjo daños severos en el cerebro al ser atravesado su cráneo por una barra de hierro. A pesar de la gravedad sobrevivió, aunque con secuelas que le produjeron cambios en su comportamiento y personalidad. El estudio de estos cambios indicó que lesiones en el lóbulo frontal podían producir cambios en las emociones, la personalidad y la interacción social de los individuos. Este caso es considerado el inicio de los estudios de la base biológica del comportamiento.

Un poco más tarde, en 1872, Charles Darwin escribió "*La expresión de las emociones en hombres y animales*" [1], donde relacionaba las respuestas faciales con estados emocionales idénticos en todos los seres humanos. Esto le permitió intuir un origen evolutivo de ciertas emociones.

En 1884 James y Lange propusieron simultánea pero independientemente una teoría según la cual la corteza cerebral recibe estímulos que provocan respuestas fisiológicas que a su vez provocan las emociones. Así, la acción de llorar provoca tristeza, y no al contrario.

Pero en el siglo XX, Cannon y Bard presentaron una nueva teoría que contradecía la de James-Lange. Según ella, las emociones no eran provocadas por estímulos fisiológicos, sino que ambos se producían simultáneamente. Por tanto, las respuestas emocionales y los sentimientos ocurrirían al mismo tiempo.

En 1937, Klüver y Bucy demostraron, mediante estudios en monos, el papel fundamental del lóbulo temporal en las emociones. Para ello extirparon parte de los lóbulos temporales provocando varios déficits conductuales en los monos. Los mismos síntomas podían provocarse extirpando la amígdala.

2 https://es.wikipedia.org/wiki/Phineas_Gage

También en 1937, Papez postuló un circuito neuronal para las emociones. Este circuito está formado por diferentes estructuras nerviosas involucradas en el control de las emociones.

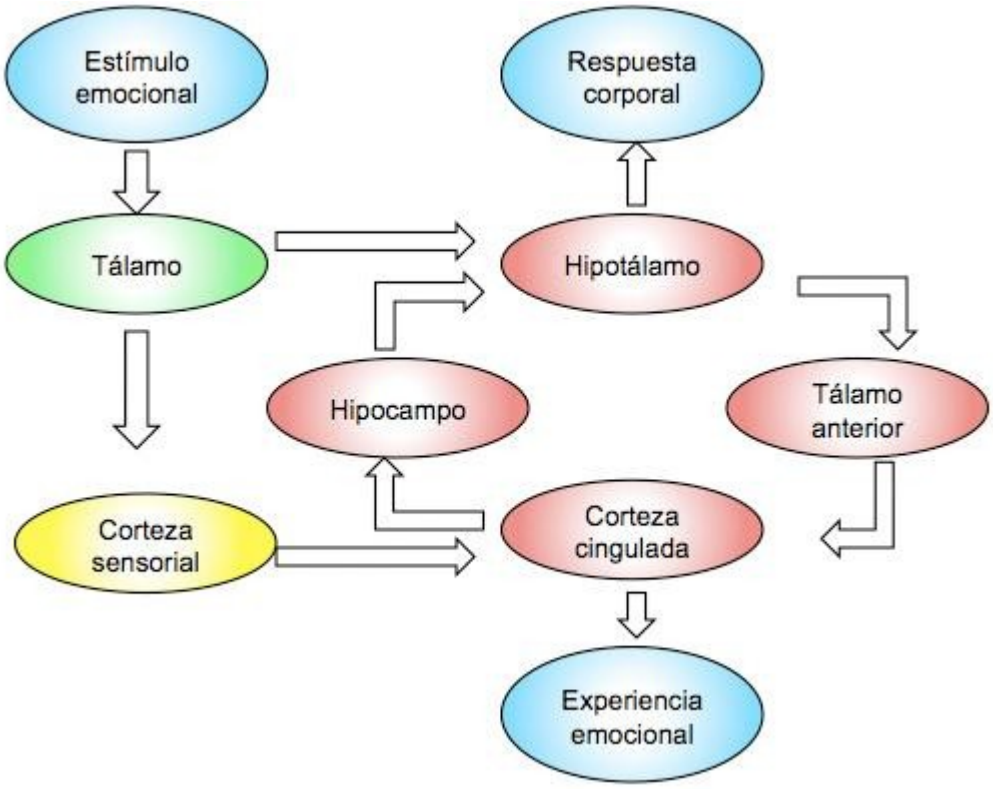


Ilustración 2. Circuito de Papez.
Fuente: **Govaf**. [artículo en línea]
<https://commons.wikimedia.org/wiki/File:Circuito_de_Papez.jpg>

Ampliando los estudios anteriores, MacLean propuso un sistema límbico. Según el mismo el cerebro se divide en tres sistemas:

- El cerebro reptiliano, en el que se observan emociones primitivas (agresión, miedo).
- El cerebro de mamífero, que aumenta las respuestas emocionales del cerebro reptiliano y elabora las emociones sociales.
- El nuevo sistema de mamífero, representa la interfaz de la emoción con la cognición.

El siguiente gran paso lo dieron Schachter y Singer con su teoría de la activación cognitiva. Para ello se basaron en estudios de Gregorio Marañón en los cuales inyectaba epinefrina en pacientes y a continuación les preguntaba por sus sensaciones. Los pacientes indicaron notar ciertas sensaciones, pero ante la falta de estímulos, no fueron capaces de relacionarlos con una emoción concreta. Schachter creía que mediante la reactivación (*feedback*) no podemos determinar qué emoción sentimos, pero que es importante para llegar a una conclusión. Además de la activación física necesitamos una evaluación cognitiva de la situación para determinar qué emoción sentimos mediante un proceso de clasificación.

Estímulo → Activación → Cognición → Emoción

2.1.2 Ekman y las expresiones faciales

Siguiendo la línea de Darwin y James, Paul Ekman pensaba que las expresiones faciales de las emociones tenían un origen biológico y no cultural. Así, personas de diferentes culturas deberían mostrar con las mismas expresiones las mismas emociones, siendo estas expresiones universales.

Para demostrarlo realizó un estudio [2] en el que mostraba diferentes fotografías a integrantes de una tribu aislada de Papúa Nueva Guinea. Dichas fotografías pertenecían a sujetos de otras culturas totalmente diferentes y observó que eran capaces de reconocer las emociones que expresaban con un alto grado de acierto. De este modo concluyó que existe un conjunto de emociones biológicamente universales en la especie humana. En 1972 elaboró una lista de las seis emociones que pensó formaban este conjunto:

- alegría
- ira
- miedo
- asco
- sorpresa
- tristeza

Por tanto, era posible mediante el estudio de las expresiones faciales descubrir cuál de estas emociones estaba sintiendo un sujeto.

Existe una séptima emoción, desprecio, que no fue incluida ya que aunque se encontraron indicios de ser universal, no eran tan claros. Se puede llegar a considerar una mezcla entre ira y asco.

En los años 90, Ekman creó una lista adicional en la que no todas las emociones pueden ser codificadas por expresiones faciales, y en la que incluía el desprecio:

- alivio
- bochorno
- complacencia o contento
- culpa
- diversión
- desprecio o desdén
- entusiasmo o excitación
- felicidad
- ira o rabia
- miedo o temor
- orgullo o soberbia
- placer sensorial
- repugnancia, repulsa, asco o repulsión

- satisfacción
- sorpresa
- tristeza
- vergüenza

2.2 Facial Action Coding System

Continuando con sus estudios, Ekman desarrolló junto a Friesen, basándose en estudios de Hjortsjö [3], un sistema para taxonomizar movimientos faciales humanos llamado *Facial Action Coding System* (FACS) [4]. Mediante el uso de FACS se puede subdividir casi cualquier expresión facial en un conjunto de unidades llamadas *Action Unit* (AU). Existen numerosas AUs (ver apéndice 1) y definen acciones como *Levantamiento interior de ceja* o *Arrugar la nariz*.

Las unidades de acción utilizan una nomenclatura que define:

- Action Units: acciones fundamentales de músculos o grupos individuales de músculos
- Nominadores de acción: movimientos unitarios que pueden implicar las acciones de varios grupos musculares.
- Notación de intensidad: se indican añadiendo letras al final de la AU. Las diferentes intensidades son:
    - A Rastro
    - B Leve
    - C Pronunciado
    - D Severo o extremo
    - E Máximo

Mediante el uso de Action Units se pueden identificar las emociones básicas, siendo cada una de ellas definida por un conjunto de AUs:

Emoción	**Unidades de acción**
Felicidad	6+12
Tristeza	1+4+15
Sorpresa	1+2+5B+26
Miedo	1+2+4+5+7+20+26
Coraje	4+5+7+23
Disgusto	9+15+16
Desprecio	R12A+R14A

Algunos ejemplos de Action Units comunes son:

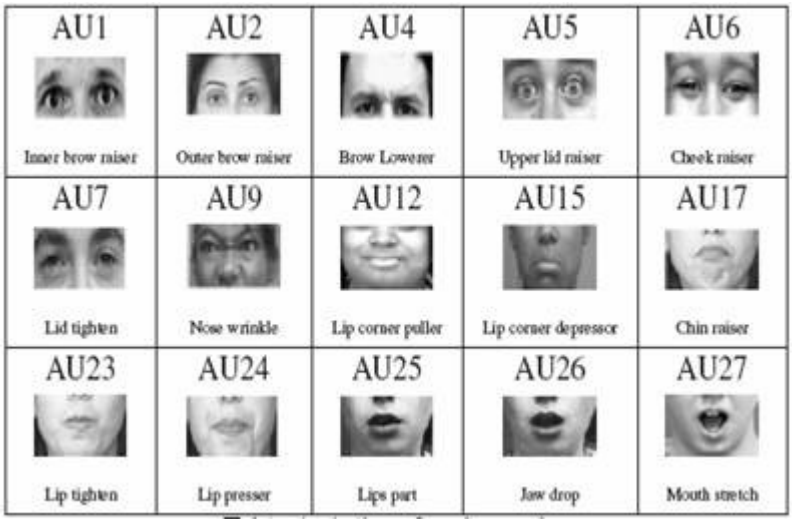


Ilustración 3. Action Units extraídas de la base de datos Cohn-Kanade.
Fuente: [artículo en línea]
<https://www.ecse.rpi.edu/~cvrl/tongy/aurecognition.html>

2.3 Computación afectiva

La computación afectiva es una rama de la inteligencia artificial que intenta diseñar y desarrollar sistemas y dispositivos capaces de reconocer, interpretar y procesar emociones humanas. Para ello utiliza tanto la detección del lenguaje corporal como el habla. Se considera la precursora de la computación afectiva a Rosalind Picard y su artículo “Affective Computing” de 1995 [5].

Una de las áreas fundamentales de la computación afectiva es la detección y reconocimiento de la información emocional, siendo una de las técnicas más importantes la detección de las expresiones faciales. Para ello habitualmente se utiliza la división de las emociones y las AUs desarrolladas por Ekman combinada con diversas técnicas de inteligencia artificial y aprendizaje automático.

2.4 Retos de la detección de expresiones faciales

El principal problema encontrado en la detección de expresiones faciales ha sido la exactitud de las técnicas utilizadas. Con el tiempo la técnica ha ido avanzando y cada vez se consigue mayor exactitud.

Otro problema se encuentra en la falta de bases de datos grandes donde entrenar y evaluar las técnicas. Además las bases de datos disponibles suelen ser de actores posando, de manera que la expresión de la emoción es muy exagerada. Esto puede facilitar el correcto funcionamiento de las

diversas técnicas con la base de datos, pero una dificultad mayor al intentar extrapolar a fotos reales en poses más naturales.

Habitualmente, la detección de emociones funciona muy bien para poses frontales con menos de 20º de rotación, pero cuando se sale de ese rango, la detección deja de funcionar.

2.5 Aplicaciones de la detección de expresiones faciales

Existen diversas aplicaciones para la detección de expresiones faciales. Algunas de ellas son:

- Educación: se puede saber el grado de atención a una clase por las emociones de los alumnos (aburrimiento, interés, frustración) y adaptar el contenido de las lecciones.
- Marketing: en grupos de estudio de interés en un nuevo producto se puede saber la percepción del mismo por parte de los participantes.
- Psicología: en trastornos como el autismo puede ser difícil averiguar el estado emocional de los pacientes sin ayuda.
- Medicina: En pacientes con problemas de comunicación se puede saber si sienten dolor y en qué grado.
- Seguridad vial: detección de estado emocional del conductor, cansancio, etc.

2.6 Críticas

Existen críticas a esta línea de estudio. Por un lado existen psicólogos y antropólogos que no creen en la existencia de emociones universales y piensan que las emociones vienen definidas culturalmente. Hay estudios que demuestran que la forma de interpretar las emociones tiene, al menos, una parte cultural. También la manera de expresar las emociones varía entre culturas. Una cultura puede considerar que una cierta emoción hay que expresarla abiertamente mientras otras creen que hay que reprimirlas.
También existen diferencias idiomáticas, existiendo lenguas donde dos emociones se expresan con la misma palabra o donde directamente no hay forma de expresar una emoción.

2.7 Estado del arte

Desde los años 90 del siglo XX hasta la actualidad ha habido un creciente interés en detectar emociones y rasgos faciales. Se han utilizado diversas técnicas para caracterizar las imágenes a analizar, como Active Shape Models, Gabor wavelets, Historam of Oriented Gradients y Local Binary Patterns.

En esta detección se han utilizado técnicas tanto estáticas como dinámicas. Las diferentes expresiones faciales son muchas veces más identificables mediante el análisis de las transiciones entre diferentes estados, y no tanto en imágenes estáticas como fotografías. Esto se aprecia especialmente en expresiones espontáneas y sin exagerar [13].

Tian et al. [14] desarrollaron métodos para detectar características faciales de imágenes utilizando descriptores geométricos. Valstar et al. [15] utilizaron puntos característicos para describir acciones faciales.

Zhang et al. [16] calcularon un conjunto de filtros Gabor para calcular descriptores, deduciendo que daban mejores resultados que los descriptores geométricos. Aun así estudios posteriores indican resultados contradictorios. Todo parece indicar que la mejor solución sería una combinación de descriptores geométricos y de apariencia.

Jiang et al. [17] utilizaron LBP como descriptores de apariencia estáticos, y LBP-TOP como dinámicos, consiguiendo mejores resultados con los dinámicos. Shan et al. [18] utilizaron LBP como descriptor para detectar emociones.

3. Arquitectura del detector

3.1 Funcionalidades

Las funcionalidades que debe realizar el sistema son las siguientes:

- **Visualizar interfaz de usuario**. Mostrará una web desde donde el usuario podrá interactuar con el detector.
- **Subir fotografía a la web desde el dispositivo**. Permitirá al usuario subir una fotografía desde el equipo que esté utilizado.
- **Cargar fotografía desde una URI de Internet**. Permitirá al usuario indicar una dirección web desde donde bajar una fotografía.
- **Cargar fotografía desde disco del servidor** (fotografías de muestra). El servidor ofrece una pequeña muestra de fotografías para utilizar como ejemplo de funcionamiento del detector.
- **Obtener puntos característicos** (landmarks) de la cara presente en la fotografía, como posición de ojos, nariz, boca, etc.
- **Normalizar la cara** presente en la fotografía. Mediante los puntos obtenidos en el punto anterior, extraerá la cara presente en la fotografía, y la rotará y escalará para normalizarla a un tamaño y posición apropiado para otras funcionalidades.
- **Extraer vector de características** de la fotografía. Obtendrá un conjunto de propiedades que definen las características de la cara normalizada.
- **Clasificar** fotografía. Utilizando el vector de características clasificará la imagen e indicará qué emoción y Action Units pueden observarse en la cara que aparece en la fotografía.

3.2 Modelado

Para detallar las funcionalidades y el sistema, se ha utilizado la metodología UML (*Unified Modeling Language*), ampliamente utilizada en el modelado de sistemas. De todos los diagramas que permite UML, se han realizado los que se consideran más apropiados para las características del proyecto.

3.2.1 Casos de uso

Un diagrama de casos de uso muestra como interacciona el sistema con los usuarios u otros sistemas.

En el sistema desarrollado la interacción del usuario es muy sencilla, puesto que solo interacciona con una web con una única página. En dicha web selecciona una fotografía de las tres fuentes posibles y el sistema automáticamente devuelve el resultado.

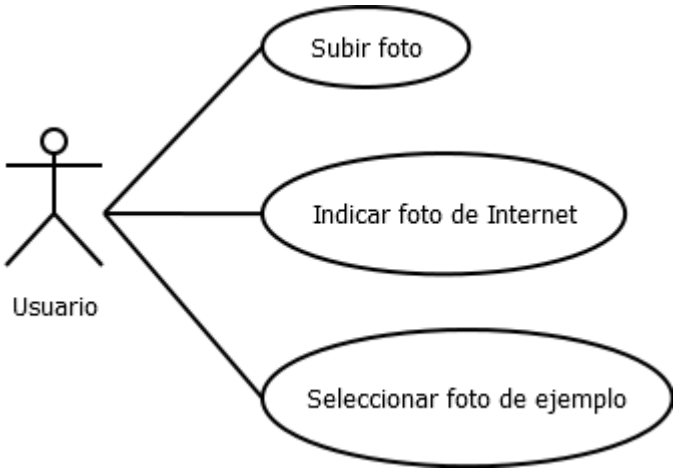


Ilustración 4. Casos de uso usuario.

Existirá un módulo, llamado Emotions, que se encarga de responder al usuario y para ello necesita interactuar con el resto de módulos:

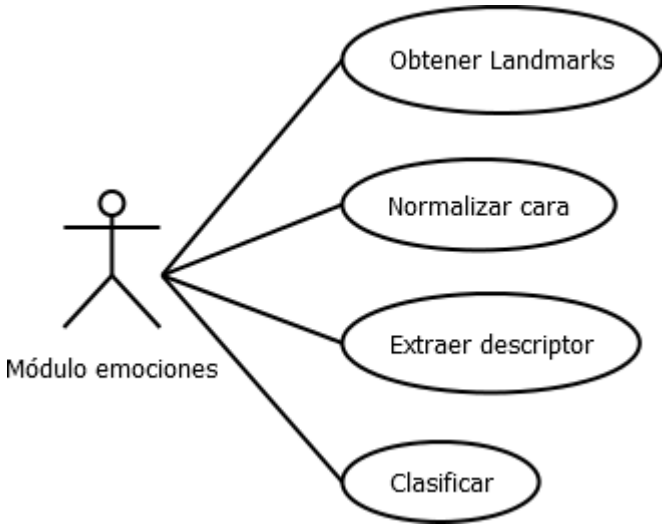


Ilustración 5. Casos de uso módulo Emotions.

3.2.2 Diagrama de actividades

Un diagrama de actividades muestra una secuencia de actividades que suceden en el sistema. El diagrama de actividades para detectar la emoción presente en una fotografía es:

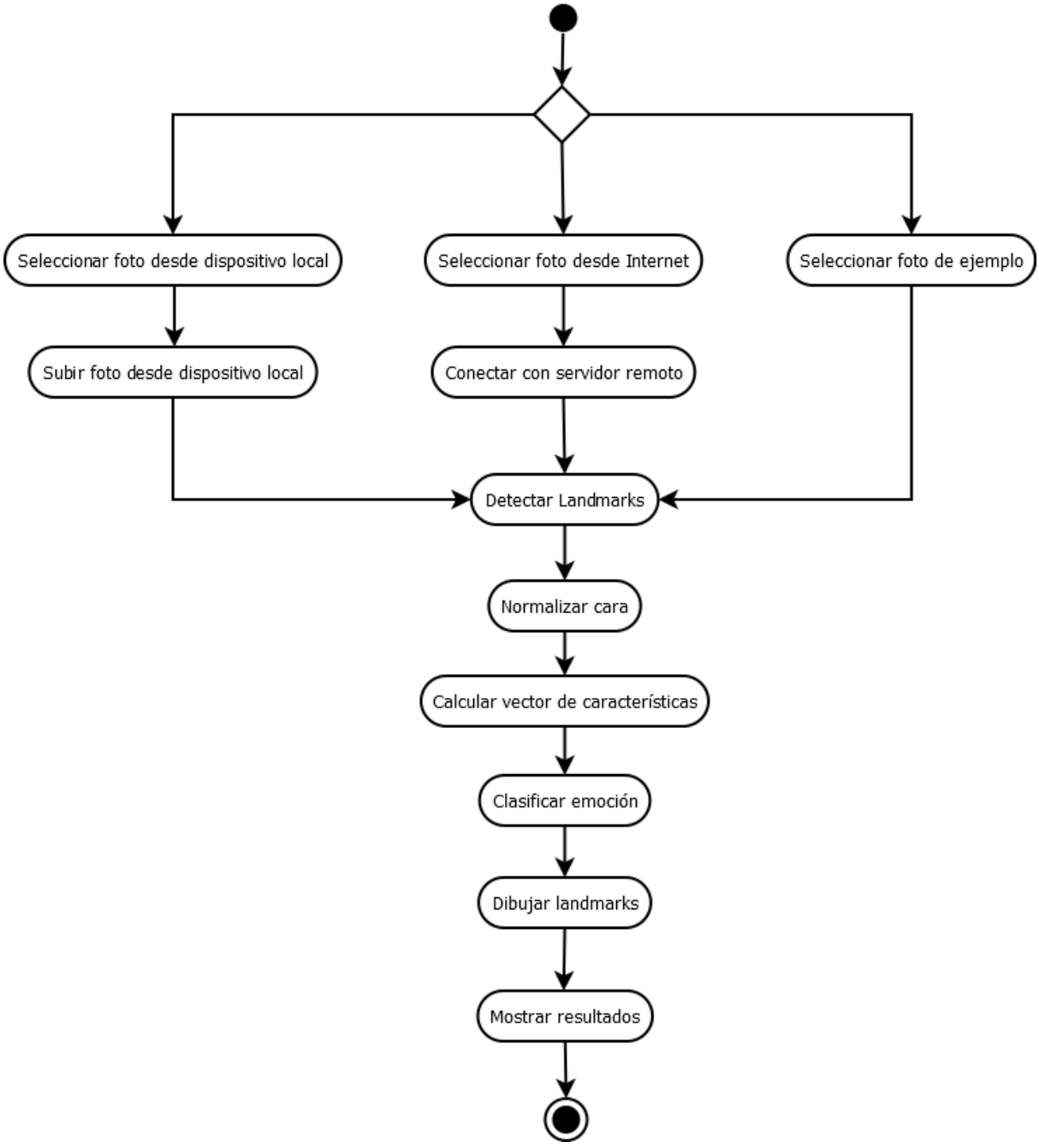


Ilustración 6. Diagrama de actividades.

En el diagrama se muestran todos los casos de uso integrados. Hubiese sido más riguroso hacer un diagrama para cada caso de uso, pero por simplicidad se han integrado todos en uno.

3.2.3 Diagrama de secuencia

Los diagramas de secuencia muestren las interacciones a lo largo del tiempo de una serie de objetos y el intercambio de mensajes que efectúan.

En el sistema desarrollado, los objetos fundamentales son:

- El **usuario**, o persona que interactúa con la página web.
- El **navegador web** que muestra la página de selección y los resultados.
- El **servicio del módulo Emotions**, que gestiona la petición.
- El **servicio del módulo Landmarks**, que calcula los puntos característicos de la cara.
- El **servicio del módulo Face normalization**, que normaliza las caras.
- El **servicio del módulo Features extractor**, que calcula el vector de características.
- El **servicio del módulo Classification**, que clasifica la fotografía en las diferentes emociones.

Los diferentes servicios pueden estar, o no, en un mismo servidor físico.

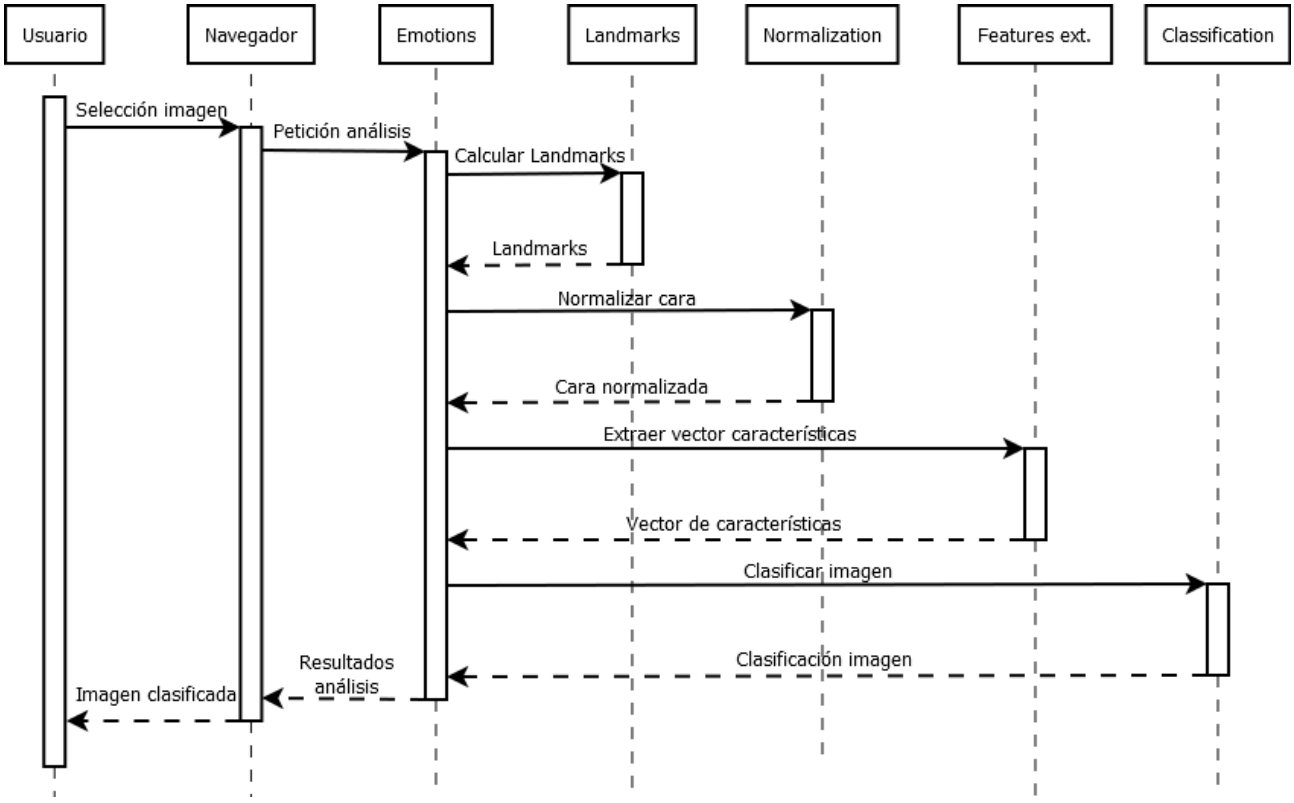


Ilustración 7. Diagrama de secuencia.

3.2.4 Diagrama de despliegue

Los diagramas de despliegue se utilizan para mostrar la disposición física de los diferentes componentes software del sistema en tiempo de ejecución. Como nos basamos en una arquitectura con microservicios (ver punto 3.3), cada módulo podría estar instalado en diferentes servidores físicos. En la instalación demo realizada para el proyecto todos los servicios se han instalado en la misma máquina, y así se muestra en el siguiente diagrama de despliegue.

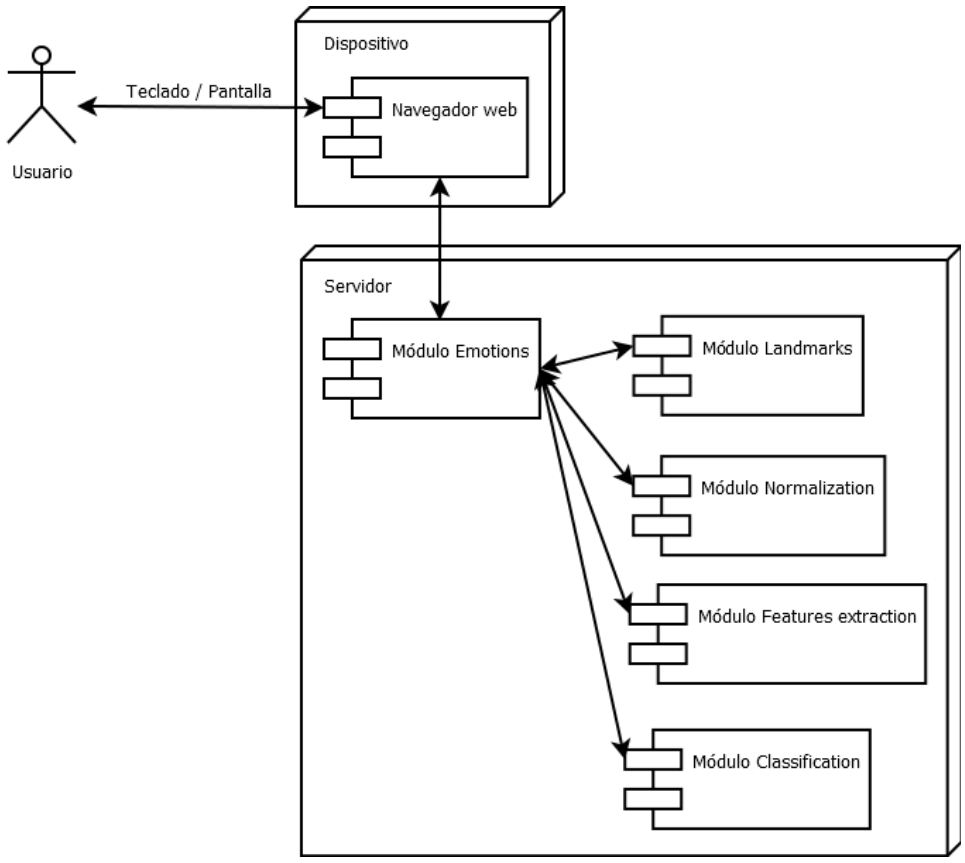


Ilustración 8. Diagrama de despliegue.

3.3 Arquitectura

Para la implementación del servidor del sistema se ha optado por una arquitectura basada en microservicios. De esta forma, cada servicio del sistema implementa una operación específica, correspondiente a cada uno de los módulos indicado en el modelado del detector. La comunicación entre los servicios se realiza utilizando APIs REST predefinidas.

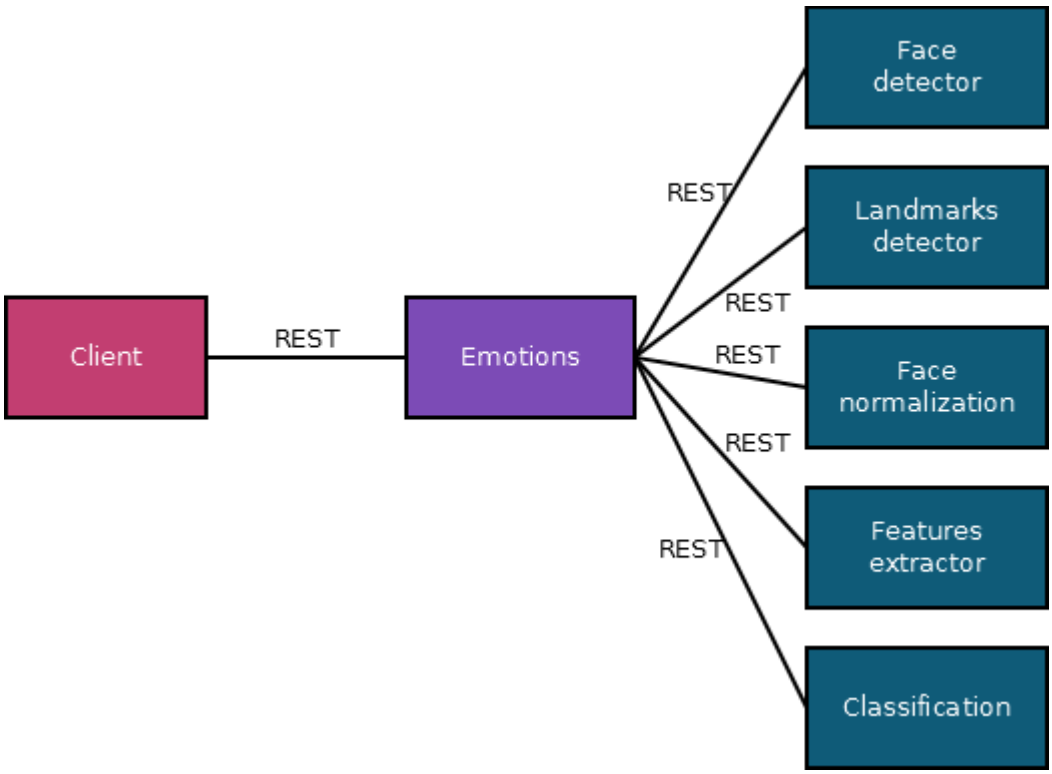


Ilustración 9. Esquema del sistema desarrollado.

Esta arquitectura ofrece una serie de ventajas, pero también de desventajas.

Ventajas:

- *Modularización*. Se divide la aplicación en módulos pequeños que permiten un mantenimiento más sencillo que el de una aplicación monolítica grande.
- *Desarrollo*. Permite que cada módulo pueda realizarlo un grupo de desarrollo independiente y con lenguajes de programación y tecnologías diferentes.
- *Implantación*. Cada módulo se puede instalar y actualizar de manera independiente a la totalidad del sistema.
- *Escalabilidad*. Si un módulo necesita más recursos, se pueden instalar varias instancias en diferentes servidores y, mediante un balanceo de carga, ir repartiendo las ejecuciones.

- *Prototipado rápido*. Nos permite probar un nuevo algoritmo creando un servicio para el mismo e integrándolo en el resto del sistema sin preocuparnos de cómo funciona dicho sistema. Lo único que se tiene que hacer es respetar el API REST de comunicación definida.
- *Módulos comunes*. Se puede utilizar un mismo servicio para distintas aplicaciones. Así, una vez implementado un algoritmo de uso habitual, se puede llamar desde diferentes proyectos, ahorrando tiempo de desarrollo.

Desventajas:

- *Instalación más compleja*. Cada módulo puede tener diferentes requisitos debido a la tecnología utilizada. El servidor deberá cumplir dichos requisitos.
- *Sobrecarga de comunicaciones*. Debido a que no podemos asumir que los datos que necesitan dos módulos están accesibles por ambos desde el mismo recurso (p.e. fichero en disco duro), los datos deben transmitirse entre los servicios a través de la API. Puede darse el caso de que cada módulo esté en un servidor físico diferente o en zonas no accesibles por ambos.

3.3.1 REST

La Transferencia de Estado Representacional (Representational State Transfer) o REST es un tipo de arquitectura para sistemas distribuidos. Se basa en unos principios definidos:

- Un protocolo cliente/servidor sin estado. Cada mensaje contiene toda la información necesaria.
- Un conjunto de operaciones bien definidas. Utiliza las operaciones estándar de HTTP GET, PUT, POST y DELETE (entre otras).
- Una nomenclatura universal de los recursos para identificarlos. Cada recurso es accesible a través de una URI[3].
- Uso de hipervínculos. Es posible navegar de un recurso a otro a través de enlaces.

En los últimos años el uso de REST ha crecido enormemente, siendo usado para la creación de servicios web gracias a su simplicidad.

En el apéndice 2 se puede consultar la API de los servicios REST creados para el sistema.

3 https://es.wikipedia.org/wiki/Identificador_de_recursos_uniforme

3.3.2 Cliente

Puesto que la comunicación se realiza a través de HTTP mediante un API REST, se pueden crear muchos clientes que accedan al sistema desde diferentes plataformas. Así, se pueden crear webs, aplicaciones para móvil o escritorio y utilizarlas desde diferentes sistemas operativos. Como demostración se ha desarrollado una página web. Dicha web es *responsive*, por lo que adapta su formato al tamaño de la pantalla del dispositivo utilizado, y puede ser utilizada desde ordenadores personales hasta móviles o tabletas.

Si abrimos la web en el navegador de un ordenador el resultado es el siguiente:

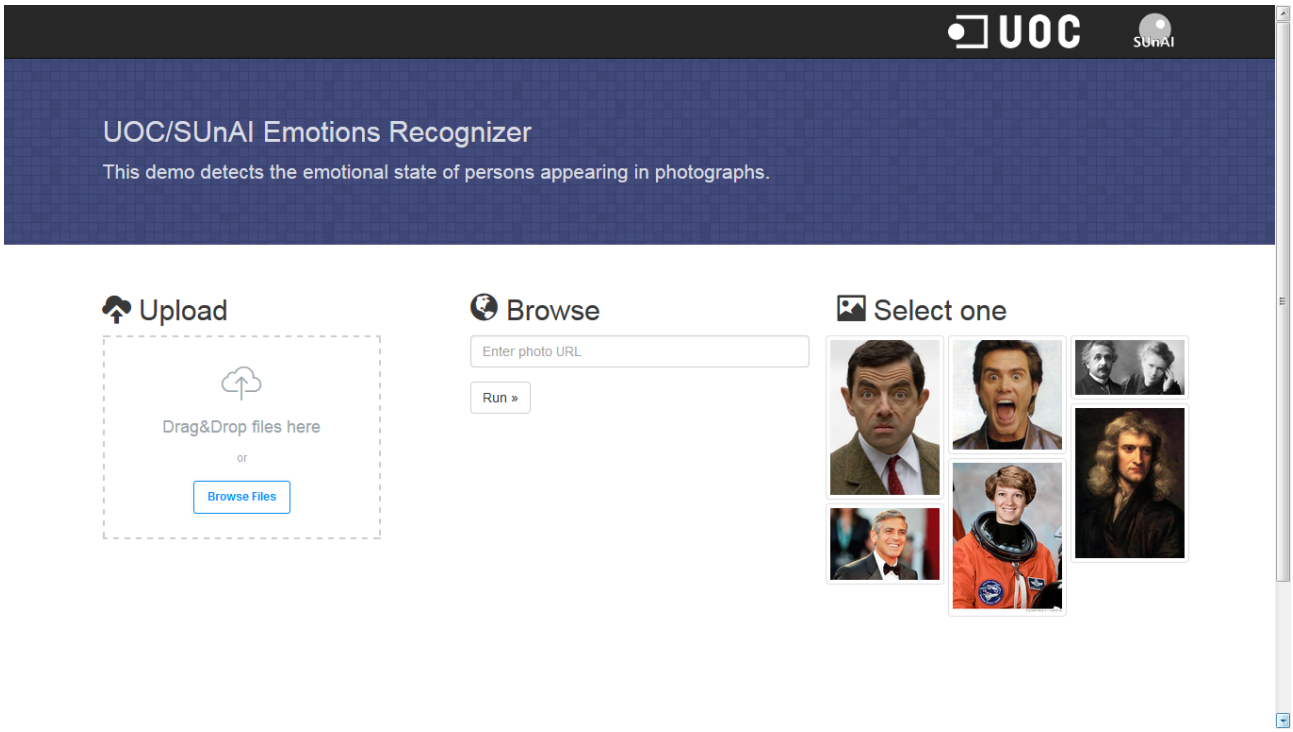


Ilustración 10. Página web del detector.

Se observa que se puede cargar una foto, indicar una dirección web con una foto o seleccionar una de las disponibles de ejemplo. Una vez enviada la foto se procesa y se devuelve el resultado:

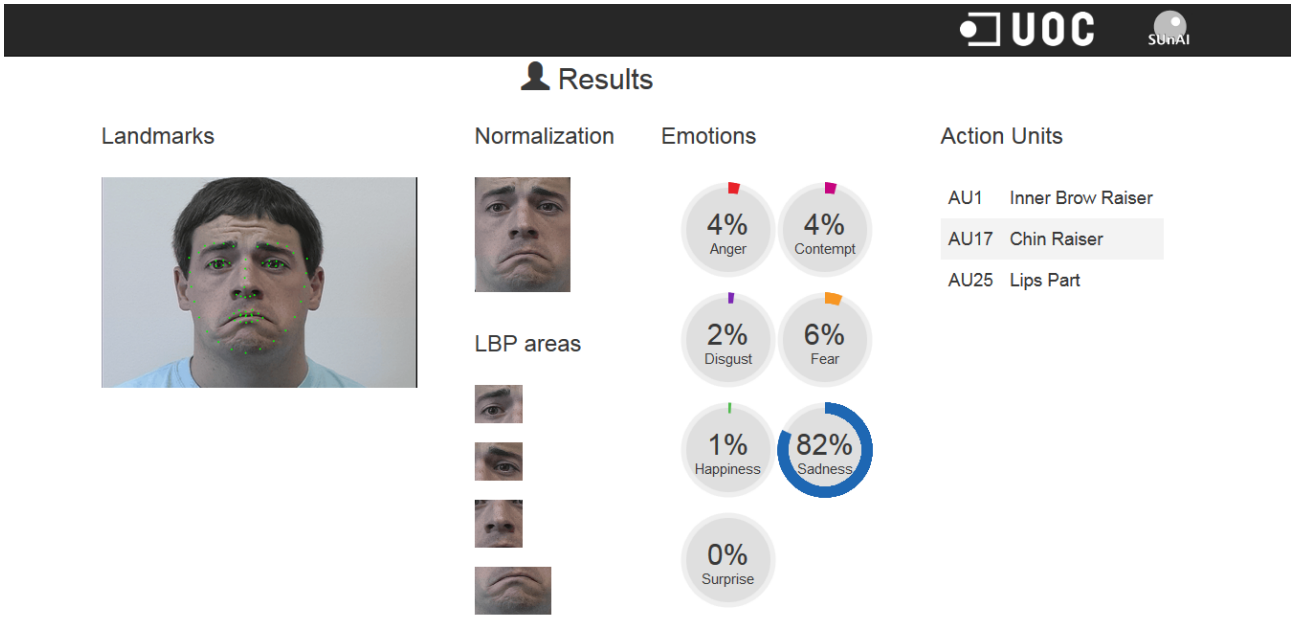


Ilustración 11. Resultado de una detección.

Se muestran los puntos de referencia de las diversas partes de la cara, la foto normalizada, las áreas utilizadas por el algoritmo LBP, la probabilidad de existencia de cada una de las emociones y por último las Action Units detectadas. En este caso el detector indica que la probabilidad más alta es que el sujeto esté triste.

3.3.3 Módulo Emotions

Emotions es el módulo de entrada al sistema. Este módulo recibe las peticiones de los clientes y va llamando a los módulos que implementan los diferentes algoritmos de forma secuencial para conseguir el resultado final a retornar.

Ilustración 12. Imagen original a analizar.
Eileen Collins[4], astronauta de la NASA.
Imagen por cortesía de la NASA

3.3.4 Módulo Face detector

Clase de módulos para implementar detectores de caras. Debido a que el módulo de detección de *Landmarks* creado ya implementa internamente algoritmos de detección de caras, no se ha implementado ninguno para el sistema final.

3.3.5 Módulo Landmarks detector

Clase de módulos para crear detectores de puntos característicos faciales.

En el sistema, se ha creado un módulo que utiliza la última versión disponible en el momento del desarrollo del algoritmo creado por Jason Saragih[5] (ver sección 4.1 para explicación del algoritmo). Actualmente la implementación[6] de este algoritmo la mantiene Kyle McDonald[7].

El módulo recibe como entrada una imagen y devuelve una serie de puntos (66) divididos en las siguientes categorías (número de puntos entre paréntesis):

- left_jaw (8)
- chin (1)
- right_jaw (8)
- left_eyebrow (5)
- right_eyebrow (5)

4 https://en.wikipedia.org/wiki/Eileen_Collins
5 http://jsaragih.org/
6 https://github.com/kylemcdonald/FaceTracker
7 http://kylemcdonald.net/

- nose_bridge (4)
- nose_base (5)
- left_eye (6)
- right_eye (6)
- outer_mouth (12)
- inner_mouth (6)

Debido a la definición de la API REST, es importante que las categorías que devuelva cualquier servidor que implemente un algoritmo distinto sean las mismas, pero el número de puntos de cada categoría puede ser diferente. Así, se podría implementar un algoritmo que retornase una cantidad de puntos mayor, permitiendo identificar las zonas de la cara con mayor resolución.

El módulo solo permite detectar una cara por fotografía aunque existan varias. Una mejora futura sería implementar un módulo con detección múltiple.

Ilustración 13. Imagen con landmarks.

3.3.6 Módulo Face normalization

Clase de módulos para normalizar el posicionado de las caras.

El módulo implementado recibe como parámetros la imagen original, la posición central de los ojos en la imagen, la distancia entre ojos en el resultado final (en píxeles), el ancho y alto de la imagen resultante (en píxeles) y la altura a la que se posicionan los ojos en la imagen resultante (en porcentaje de la altura). Retornará una imagen con los ojos situados en la misma horizontal, anulando cualquier rotación, y del tamaño y posicionado definido. Se utiliza interpolación cúbica para el cálculo de la nueva imagen (ver sección 4.2 para explicación del algoritmo).

Ilustración 14. Imagen normalizada.
Tamaño: 100x120 px
Distancia entre ojos: 50 px
Altura de ojos: 0.25

3.3.7 Módulo Features extractor

Clase de módulos que retornan un descriptor de características de la imagen.

En el sistema implementado se ha generado un descriptor con la concatenación de los resultados de calcular el algoritmo *Local Binary Patterns* (LBP, ver sección 4.3 para explicación del algoritmo) en cuatro zonas de la imagen. Estas zonas corresponden a los dos ojos, la nariz y la boca. El módulo permite utilizar LBP original y LBP uniforme. El primero devuelve un vector de características con 256 valores por cada zona, y por tanto, un descriptor global con 1024 valores. El segundo devuelve 59 valores por zona para un total de 236 valores. Las cuatro áreas tienen parte de la imagen común con respecto a la imagen contigua.

Ilustración 15. Imágenes de las cuatro áreas
para las que se calcula el LBP.

3.3.8 Módulo Classification

Clase de módulos para realizar la clasificación de una imagen mediante su descriptor de características.

En el sistema implementado se ha utilizado una Máquina de Vector de Soporte (Support Vector Machine o SVM, ver sección 4.4 para explicación del algoritmo). Puesto que SVM es un clasificador supervisado, necesitaremos hacer un entrenamiento previo con un conjunto de

imágenes etiquetadas. Este entrenamiento no se ejecutará junto con el detector, puesto que se realiza una sola vez y no cada vez que un usuario hace una petición a la web, ya que sería costoso computacionalmete. En el sistema se instalará un clasificador previamente entrenado. Por tanto, además del detector descrito, se ha tenido que desarrollar el entrenador de los clasificadores.

4. Algoritmos

4.1 FaceTracker

FaceTracker es una librería para el seguimiento de caras. Implementa un algoritmo diseñado por Jason Saragih [6], [7] y [8].

El algoritmo utiliza una técnica similar a la del algoritmo *Active shape model* (ASM), creado por Tim Cootes y Chris Taylor en 1995. Esta técnica consiste en tener un modelo de un objeto definido a base de puntos e ir modificando la posición de los puntos de manera iterativa para que desde el modelo acaben ajustándose a un ejemplo concreto.

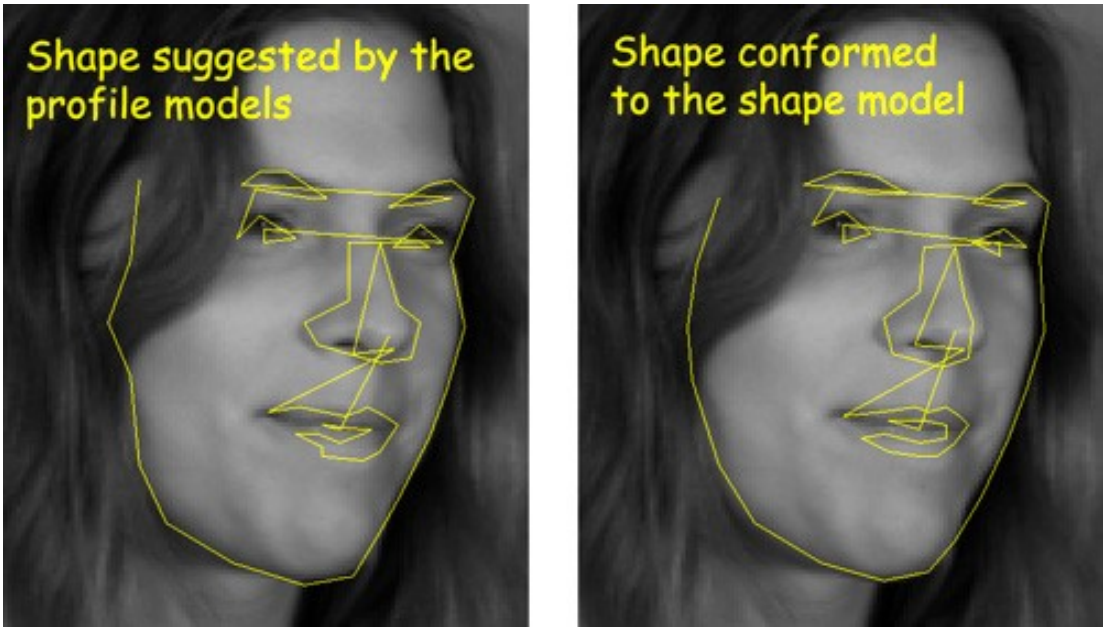


Ilustración 16. Active shape model.
Fuente: Richard Peralta Vinas[artículo en línea]
<https://en.wikipedia.org/wiki/File:Operation_Of_Shape_Model_In_Active_Shape_Model.jpg>

Para calcular el modelo se usa entrenamiento sobre un conjunto de imágenes. Mediante este modelo se puede reconocer si un contorno encontrado en la imagen es un candidato apto a ser el contorno del objeto que modela. El desplazamiento de los puntos en cada iteración del algoritmo viene limitado por el modelo.

En el caso específico del algoritmo diseñado por Saragih, el proceso se divide en varias etapas. La primera consiste en la detección de donde hay una cara en la foto. Para ello utiliza un detector Haar. Este detector se basa en descomponer la imagen en regiones claras y oscuras y entonces comparar las regiones para ver si coinciden con un patrón de cara. El problema es que este detector puede tener problemas detectando caras de piel oscura. Por ello primero se hace una ecualización del histograma de la imagen. Así individuos con diferentes tonos de piel se normalizan y acaban teniendo las mismas diferencias relativas entre regiones.

A continuación se va realizando el ajuste de los puntos siguiendo la técnica tipo ASM. Se van analizando gran cantidad de pequeñas regiones y en estas regiones se observan las diferencias de iluminación y color entre zonas. Dependiendo de estas variaciones se van ajustando los puntos. Los puntos no se ajustan libremente, sino que están relacionados entre ellos. Así, si cambia un punto de una ceja afectará al resto de puntos de la ceja, ya que en el modelo están unidos y relacionados.

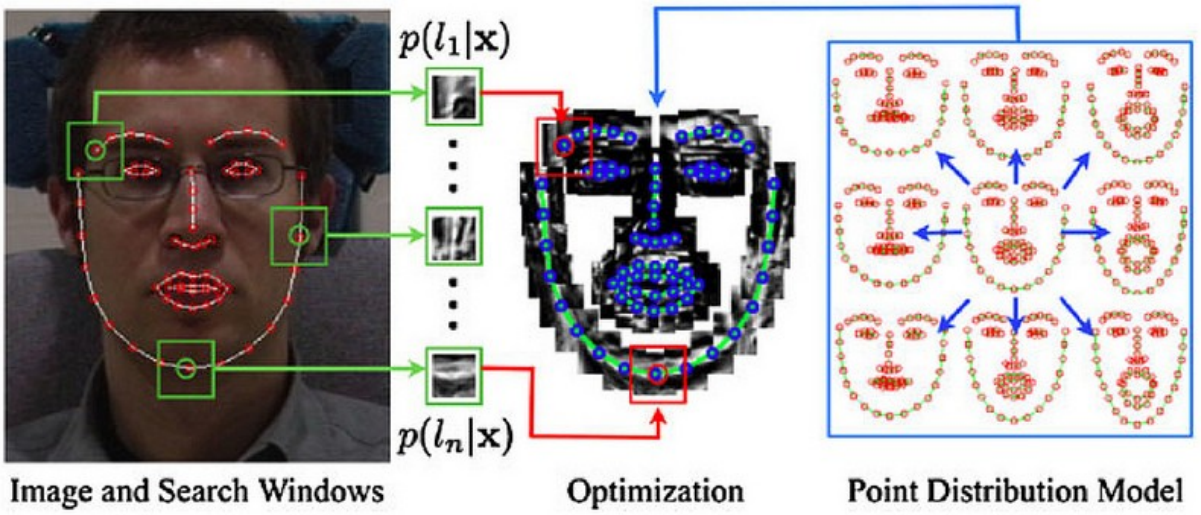


Ilustración 17. Ajuste de puntos en FaceTracker.
Fuente:Makematics.com
<http://makematics.com/research/facetracker/>

Este proceso se va repitiendo hasta tener un resultado adecuado. En algunas situaciones el resultado no llega a ser correcto, por las características de la foto. Así, si la cara aparece parcialmente oculta o en posiciones con mucha rotación en cualquiera de los ejes podría llegar a tener problemas en ajustar los puntos del modelo a la cara real.

4.2 Face normalization

Para normalizar la cara, primero se calculan los centroides de todos los puntos de cada ojo, obtenidos por el algoritmo FaceTracker. Con las posiciones de ambos ojos se utilizan operaciones simples trigonométricas y afines para obtener el ángulo de rotación de la imagen y se aplica a la misma una rotación inversa. De esta forma obtenemos una imagen con los ojos en la misma horizontal.

A continuación se escala la imagen para que la distancia entre los centros de ambos ojos sea la indicada por el parámetro de entrada.

Por último se recorta la imagen para que tenga el tamaño indicado y la horizontal de los ojos esté a la altura adecuada.

4.3 Local Binary Patterns

Local Binary Patterns (LBP) es un operador aplicable a imágenes y texturas, creado originalmente en 1996 por Ojala et al [9]. Sus principales ventajas son su tolerancia a cambios de iluminación y su velocidad de cálculo, que le permite ser usado en aplicaciones en tiempo real.

El operador consiste en evaluar los píxeles de la vecindad alrededor de un píxel central y aplicarles un umbral (*threshold)* al valor del píxel central. Así si el píxel vecino es mayor o igual que el del central, le asignamos un 1 y si no un 0. Con los resultados de todos los píxeles del vecindario generamos un número único, multiplicando cada píxel por una potencia de 2 y sumándolos.

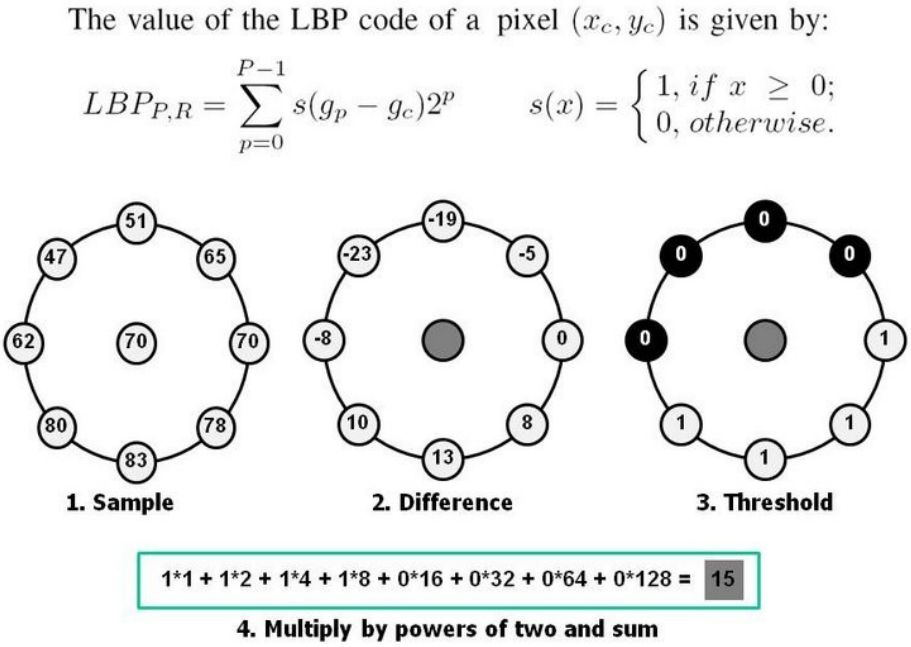


Ilustración 18. Local Binary Patterns.
Fuente: Matti Pietikäinen [artículo en línea]
<http://www.scholarpedia.org/article/File:LBP.jpg>

El operador LBP puede calcularse con diferentes tamaños de vecindario (P) y radio (R). El vecindario es el número de píxeles que se inspeccionan en un círculo del radio indicado. La definición original de LBP solo consideraba vecindarios con P=8. Posteriormente se extendió para admitir vecindarios de múltiples tamaños.

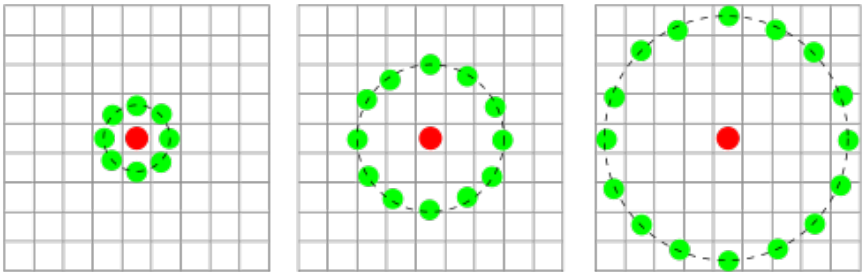


Ilustración 19. Vecindarios LBP.
Fuente: Xiawi [artículo en línea]
<https://commons.wikimedia.org/wiki/File:Lbp_neighbors.svg>

4.3.1 Descriptor Local Binary Patterns

Para crear un descriptor basado en LBP se calcula el operador LBP para cada píxel de la imagen. Con todos los valores se calcula el histograma de frecuencias de aparición de cada valor. Este histograma será el descriptor.

Para un vecindario P=8, el operador LBP puede devolver 256 valores diferentes para cada píxel. Podemos por tanto calcular un histograma con 256 bins y el vector formado por las frecuencias del histograma será el vector de características del descriptor. En caso de tener varias regiones, podemos calcular el descriptor LBP para cada región y concatenarlo creando un único vector de características.

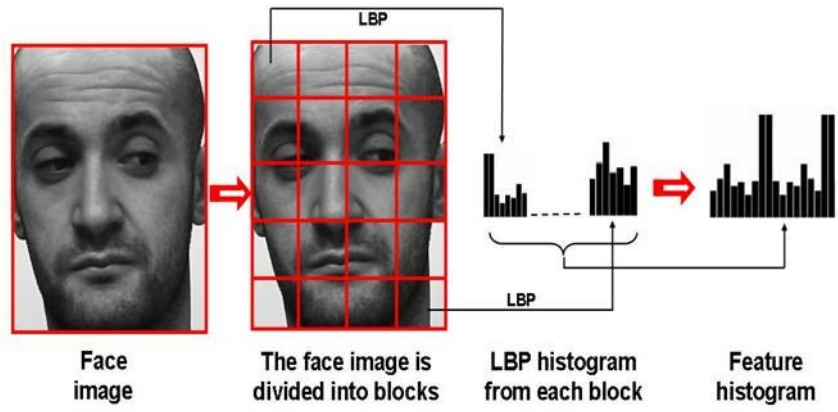


Ilustración 20. Vector de características LBP.
Fuente: Matti Pietikäinen [artículo en línea]
<http://www.scholarpedia.org/article/File:LBP-face.jpg>

4.3.2 Extensiones a Local Binary Patterns

Sobre el operador LBP original se han añadido diversas extensiones. Una de las más importantes es *Uniform LBP*. Su principal ventaja es que consigue reducir el tamaño de los vectores de características obtenidos. Se observó que ciertos patrones binarios se repiten más que otros en el análisis de texturas. Un patrón se llama uniforme si contiene como mucho dos transiciones entre 0 y 1 o viceversa. Así, el patrón 01011001 no lo es por tener 5 transiciones. Tras esta reducción quedan solo 59 patrones posibles (en un vecindario P=8) que se pueden etiquetar con un valor único. Con esto conseguimos que los histogramas pasen de tener 256 bins a 59, reduciendo el tamaño del vector.

Otra extensión llamada LBP-TOP añade un componente temporal a LBP. De esta forma se puede aplicar a secuencias de imágenes, por ejemplo pertenecientes a un vídeo.

4.4 Support Vector Machines

Las máquinas de soporte vectorial o support vector machines (SVM) son algoritmos de aprendizaje supervisado utilizados para clasificar datos y hacer análisis de regresión. Fueron desarrollados por Vladimir Vapnik y su equipo [10].

Una SVM construye un hiperplano o un conjunto de hiperplanos en un espacio de alta dimensionalidad (puede llegar a ser infinita). Este hiperplano separa los puntos a clasificar en dos categorías. Para crear el hiperplano, se necesita un proceso de entrenamiento en el que se pasan puntos etiquetados según la categoría a la que pertenecen. Este hiperplano separa las categorías de puntos de manera óptima, implicando esto buscar la separación mayor entre el hiperplano y los puntos más cercanos de cada categoría.

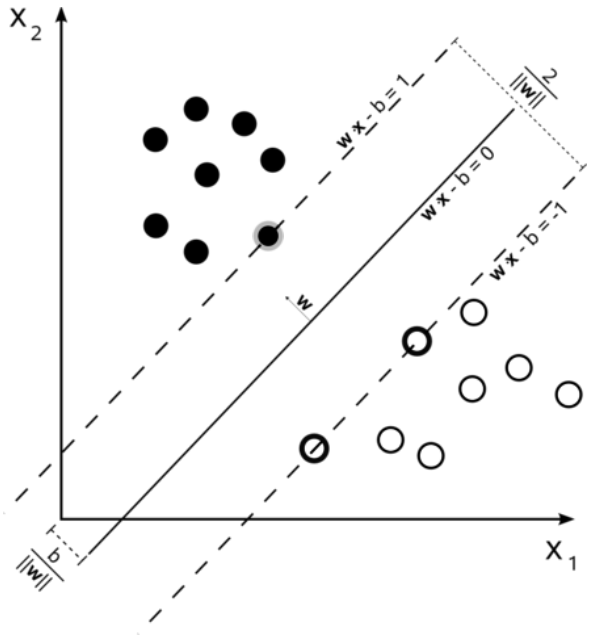


Ilustración 21. Hiperplano de una SVM lineal.
Fuente: Cyc [artículo en línea]
<https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png>

Una vez calculado el hiperplano, para clasificar un nuevo punto se mirará en qué lado del hiperplano se encuentra, y se le asignará la categoría de los puntos de entrenamiento de ese lado.

No siempre se puede lograr una separación perfecta de las categorías y por ello existe un margen que permite errores y los penaliza.

Para poder hacer separaciones no lineales, se pueden utilizar funciones kernel. Mediante ellas se proyectan los datos a un espacio de características transformado, de forma que en ese nuevo espacio se pueda separar los puntos mediante hiperplanos. Algunos de los kernels más comunes son[8]:

- Polinomial-homogénea: $K(x_i, x_j) = (x_i \cdot x_j)^n$
- Perceptron: $K(x_i, x_j) = \| x_i - x_j \|$
- Función de base radial Gaussiana: separado por un hiperplano en el espacio transformado.

$$K(x_i, x_j) = \exp(-(x_i - x_j)^2 / 2(\text{sigma})^2)$$

- Sigmoid: $K(x_i, x_j) = \tanh(x_i \cdot x_j - \theta)$

8 Fuente: https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte

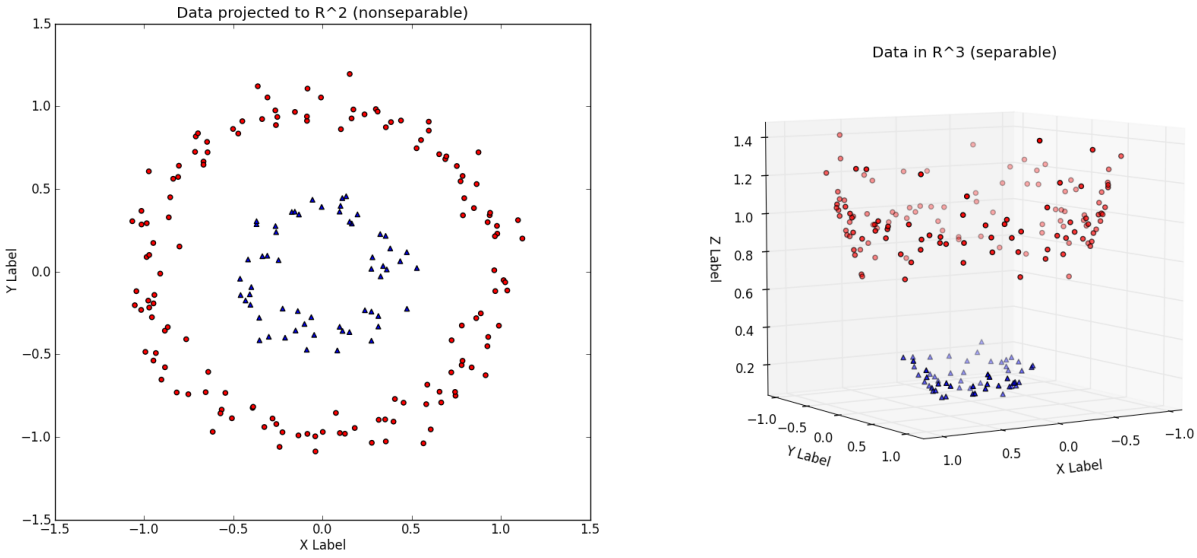


Ilustración 22. Proyección de un espacio 2D a uno 3D.
Fuente: Eric Kim [artículo en línea]
<http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html>

En este ejemplo, los puntos en el espacio 2D no son divisibles por un hiperplano (una línea), pero mediante la transformación T([x, y]) = [x, y, $x^2 + y^2$] pasan a serlo:

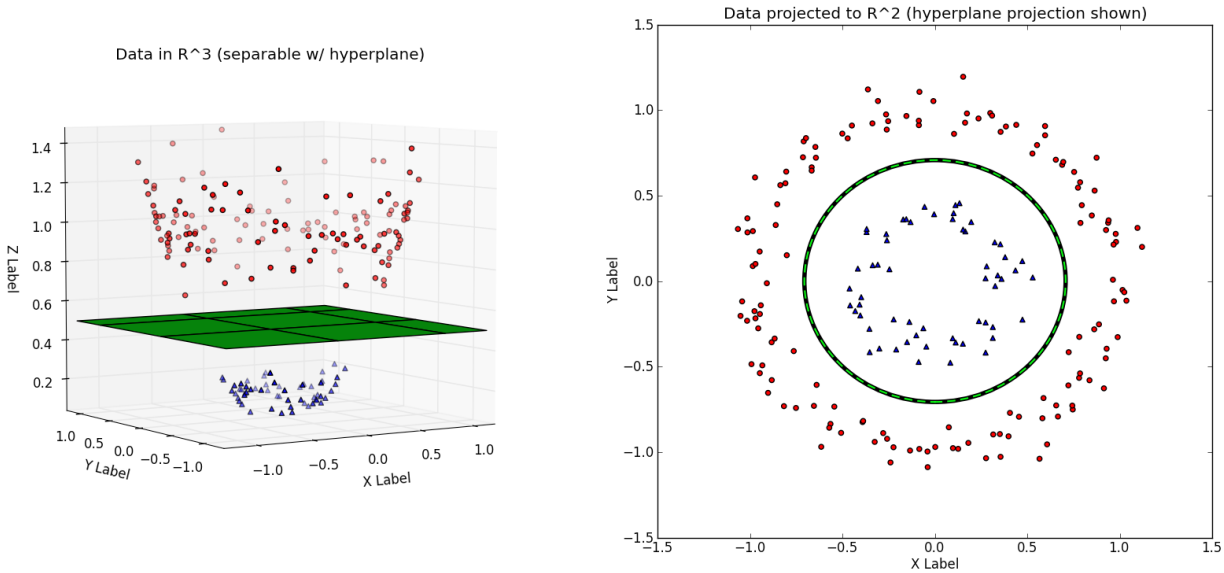


Ilustración 23. Hiperplano espacio transformado.
Fuente: Eric Kim [artículo en línea]
<http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html>

5. Implementación del detector

5.1 Cliente

El cliente del detector es una página web. Se ha programado con HTML, CSS y Javascript, por lo que no es necesario instalar ningún plug-in adicional para ser abierta y es compatible con todo tipo de navegadores, tanto de escritorio como móviles.

Para el desarrollo de la misma se han utilizado las bibliotecas y frameworks JavaScript:

- *Framework* de desarrollo Bootstrap[9], creado por Twitter. Se ha utilizado para la creación de la web de forma *responsive* (licencia MIT).
- jQuery[10], se ha utilizado entre otras cosas para realizar llamadas AJAX al servidor, de forma que se actualiza la información de la página sin necesidad de recargarla (licencia MIT).
- jQuery.filler[11] para subir ficheros al servidor (licencia MIT).
- Masonry[12], usada para la creación de la galería de imágenes de ejemplo (licencia MIT).
- Circliful[13], para añadir controles con forma de círculo porcentual (licencia MIT).

El código de la web se puede encontrar en la carpeta “code/html” que acompaña a esta memoria. Los archivos principales son:

- *index.html*, archivo principal que abre la web. Es el único punto de acceso de los usuarios.
- *css/emotions.css*, archivo css que define el estilo y diseño de la web.
- Carpeta *images*, contiene las imágenes de ejemplo.
- *js/configV1.js*, archivo con la configuración de conexión al servidor del detector. Es el punto de conexión con la API REST del módulo emotions (ver apéndice 2).
- *js/emotionsV1.js*, archivo Javascript que procesa las respuestas del servidor.
- Carpeta *libs*, contiene las librerías Javascript mencionadas antes.
- Carpeta *results*, carpeta donde se almacenan las imágenes temporales que genera el detector. Estas imágenes son las que se ven desde la web en la zona de resultados.
- Carpeta *web-img*, imágenes de diseño de la web, como logos y fondo.

5.1.1 Archivo index.html

El archivo *index.html* es la entrada principal al detector. Genera la web que puede ver el usuario si introduce la dirección del servidor en un navegador. Carga todas las librerías y archivos de diseño necesarios. Como indica su extensión está desarrollado en HTML e incluye algunas funciones Javascript.

9 http://getbootstrap.com/
10 https://jquery.com/
11 http://filer.grandesign.md/
12 http://masonry.desandro.com/
13 https://github.com/pguso/jquery-plugin-circliful

La web está diseñada como una página única (single page application o SPA). Tanto el formulario inicial solicitando la fotografía a procesar como la respuesta al análisis de dicha fotografía se ve en una única página sin recargas de la misma. Para lograrlo se hace uso de llamadas AJAX[14] (Asynchronous JavaScript And XML) mediante jQuery. Cuando el usuario solicita un análisis se hace una llamada AJAX al API REST del servidor y se procesa la respuesta.

Para el diseño se ha utilizado las herramientas de *Grid* de Bootstrap. Bootstrap define la disposición (o layout) de la web mediante una cuadrícula de 12 columnas y tantas filas como sean necesarias. Dependiendo del tamaño de la pantalla con la que se está navegando, a cada elemento se le asignan un número de columnas de esas 12. Si estamos en una pantalla de navegador de escritorio, que es ancha, podemos asignar a un elemento un ancho de, por ejemplo, 3 columnas y disponer varios elementos uno al lado de otro en la misma fila. Pero si esa misma web la intentáramos ver en un móvil, no tendríamos anchura suficiente para todos los elementos y definiríamos un ancho de 12 columnas para cada elemento y se vería uno debajo de otro, en filas separadas. Esto se consigue mediante elementos del tipo:

```
<div class="col-lg-3 col-md-3 col-xs-12">
```

donde col-lg indica el número de columnas del elemento para pantallas grandes, col-md para pantallas medianas y col-xs para pantallas pequeñas. Se puede observar en el código que *index.html* hace un uso exhaustivo de esta técnica.

5.1.2 Archivo emotionsV1.js

El archivo *emotionsV1.js* contiene tres funciones Javascript. Dos de ellas son la captura del evento click, de manera que cuando se selecciona una fotografía se envía a procesar al servidor. La tercera, *changeResults*, es llamada desde las respuestas a los dos eventos anteriores. Esta función extrae los datos de la respuesta del servidor (en formato JSON[15]) y genera la visualización que se mostrará en la página al usuario.

5.2 Servidor

Para implementar los diferentes módulos del servidor se ha utilizado el lenguaje Python. Se ha llegado a esta decisión debido a que es un lenguaje ampliamente utilizado en el mundo científico, por lo que existen disponibles un gran número de bibliotecas que implementan funcionalidades de todo tipo. Algunas de esas funcionalidades, muy especializadas, serán necesarias en el desarrollo de los diferentes módulos y desarrollarlas totalmente llevaría una cantidad de tiempo inasumible. Además, Python es un lenguaje fácil de usar y permite que investigadores sin conocimientos muy especializados de programación puedan añadir nuevos módulos al sistema en un futuro.

14 https://es.wikipedia.org/wiki/AJAX
15 https://es.wikipedia.org/wiki/JSON

Como ya se vio en la arquitectura, los diferentes módulos del servidor se comunican mediante APIs REST. Para implementarlas en Python se ha utilizado el *Framework* Bottle[16]. Bottle permite encaminar una petición REST hacia una función Python que la procese. Para ello, se implementan rutas similares a:

```
@route('/emotions/v1/images', method=['POST', 'OPTIONS'])
    def add_image():
```

En este caso una llamada a la web con la ruta "/emotions/v1/images", siempre y cuando se realice mediante el método POST, se procesa mediante la función add_image.

Además, Bottle facilita acceso a los datos que acompañan a la petición, como campos de formularios, ficheros subidos, etc. Para ello se utiliza la variable *request*.

Como los módulos del detector pueden estar en diferentes sistemas y dominios, las llamadas AJAX desde la web podrían fallar por el mecanismo de seguridad que implementan por defecto los navegadores. Este mecanismo impide abrir recursos o hacer llamadas AJAX desde diferentes dominios al de la petición inicial (en este caso de la página). Para evitarlo, se implementa una función que habilita Cross-origin resource sharing (CORS)[17]. CORS es un mecanismo que permite llamar a estos recursos en otros dominios.

Una vez procesada la petición, todos los módulos responden enviando un fichero JSON con los datos de la respuesta.

5.2.1 Módulo emotions

Este módulo sirve para ir llamando a los otros módulos. Su código se puede encontrar en la carpeta "code/modules/emotions". El fichero principal es "emotionsV1.py", siendo el resto de apoyo al mismo.

Inicialmente, el módulo recibe mediante Bottle la información de la imagen a analizar. Esta información incluye el tipo de fuente de la imagen (url, local o enviada con el POST) y los datos necesarios para recuperarla. Una vez recuperada la imagen se envía al módulo de detección de landmarks.

Con los landmarks obtenidos, *emotions* calcula las coordenadas del punto central de cada ojo y las envía al módulo de normalización junto con la imagen original. Además incluye los siguientes parámetros:

- Distancia entre ojos = 50
- Ancho de la imagen normalizada = 100
- Alto de la imagen normalizada = 120
- Altura de los ojos en la imagen = 25%

16 http://bottlepy.org
17 https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

A continuación, el módulo emotions extrae las cuatro áreas de interés (ver ilustración 15) de la foto normalizada y ejecuta cuatro veces el módulo features_LBP, una por área. Los cuatro vectores de características retornados por features_LBP son concatenados y se pasan a los módulos de clasificación.

Finalmente el módulo devuelve el resultado de la clasificación y una imagen de la fotografía original con los landmarks dibujados en la misma.

5.2.2 Módulo landmarks_Saragih

El código del módulo se puede encontrar en la carpeta “code/modules/landmarks_Saragih”. El fichero inicial es “landmarks.py”.

El módulo está implementado en Python, mientras que la biblioteca que implementa el algoritmo de Saragih (FaceTracker) lo está en C++. Para poder comunicarlos, se ha implementado un envoltorio o *wrapper* mediante la librería Boost Python[18]. Existe un *wrapper* disponible[19], pero no está actualizado desde hace tiempo (16-07-2014). Además, así se ha desarrollado específicamente para los requerimientos de la aplicación. El código de la biblioteca y el *wrapper* está disponible en “code/FaceTracker”, incluyendo el fichero “face_tracker.cc”, que consiste en una versión modificada con el *wrapper* Boost.

El primer paso necesario es compilar la biblioteca FaceTracker. Para ello es necesario instalar las bibliotecas OpenCV2. Como se está utilizando como sistema operativo un Ubuntu 15.04 Server, bastará con ejecutar:

```
sudo apt-get install libcv-dev libopencv-dev
```

A continuación se clona el repositorio de Kyle McDonald donde está alojada la biblioteca:

```
git clone git://github.com/kylemcdonald/FaceTracker.git
```

Se configura la ruta donde se ha instalado OpenCV en el fichero Makefile y se compila el proyecto mediante la orden *make*.

Con el código del módulo se distribuye la biblioteca compilada (archivo “face_tracker.so”). Esta biblioteca incluye el código original de Saragih/McDonad junto con el *wrapper* de implementación propia. La biblioteca necesita un modelo que se puede encontrar en la carpeta:

“code/modules/landmarks_Saragih/model”

Este es el modelo que define los puntos que existen en una cara y las relaciones entre ellos.

18 http://www.boost.org/doc/libs/1_60_0/libs/python/doc/html/index.html
19 https://bitbucket.org/amitibo/pyfacetracker

5.2.3 Módulo normalization_face

El código del módulo se puede encontrar en la carpeta “code/modules/normalization_face”. El fichero principal es “normalization.py”.

El módulo inicialmente carga la imagen y los parámetros enviados por el módulo emotions. Utilizando la posición de los ojos calcula la rotación de la fotografía y aplica la rotación inversa. A continuación hace un escalado y recorta la fotografía. Por último retorna la ubicación de la imagen resultante.

5.2.4 Módulo features_LBP

El código del módulo se puede encontrar en la carpeta “code/modules/features_LBP”. El fichero principal es “lbp.py”.

El módulo calcula un vector de características mediante el algoritmo LBP (ver punto 4.3).

Tras cargar la imagen pasada se cargan los parámetros del algoritmo Estos parámetros son:

- n_points, número de píxeles del vecindario LBP.
- radius, radio para el algoritmo LBP.
- method, método a utilizar al ejecutar el algoritmo LBP.Puede ser:
  - original, algoritmo LBP original.
  - uniform, algoritmo LBP uniforme.

Tras obtener los parámetros, se transforma la imagen a escala de grises con la función de scikit-image[20]:

```
image = color.rgb2gray(image)
```

A continuación se calcula el descriptor LBP mediante las funciones de scikit-learn[21]:

```
lbp = local_binary_pattern(image, int(points), float(radius), 'default')
values, bins = numpy.histogram(lbp, bins=numpy.arange(257))
```

La primera función calcula el algoritmo LBP para cada píxel de la imagen, mientras que con la segunda se calcula el histograma de frecuencias del resultado.

En *values* se encontrará el descriptor buscado, siendo devuelto como resultado.

20 http://scikit-image.org/
21 http://scikit-learn.org/

5.2.5 Módulo features_HOG

El código del módulo se puede encontrar en la carpeta “code/modules/features_HOG”. El fichero principal es “hog.py”.

Este módulo implementa un descriptor diferente mediante el algoritmo Histogram of oriented gradients (HOG)[22] [11]. Se ha implementado para probar mejoras en la clasificación con un descriptor que mezclara LBP y HOG. Los resultados no han sido los esperados, por lo que no se ha utilizado en la versión final. A pesar de ello se incluye el código como demostración de la modularidad del sistema, y la posibilidad de crear nuevos descriptores de forma sencilla y rápida.

El funcionamiento es muy similar al módulo features_LBP. Se obtiene la fotografía y los siguientes parámetros:

- orientations
- cell_x
- cell_y
- block_x
- block_y

Estos parámetros son utilizados en la función de scikit-learn:

```
hog = hog(image, orientations=int(orientation), pixels_per_cell=(int(cell_x), int(cell_y)),
        cells_per_block=(int(block_x), int(block_y)))
```

Después se crea un histograma de frecuencias y se devuelve como vector de características.

5.2.6 Módulo classification_emotions

El código del módulo se puede encontrar en la carpeta “code/modules/classification_emotions”. El fichero principal es “classify.py”.

El módulo carga, al arrancarse, el modelo del clasificador entrenado (ver capítulo 7). De esta forma se evita tener que cargarlo en cada petición de clasificación posterior. Para cada una de estas peticiones, se lee el vector de características como valor de entrada y con este vector se clasifica la imagen mediante la función de scikit-learn:

```
svm_pred = clf_prob.predict_proba(current_feature)
```

Como resultado obtenemos un vector de probabilidades. Este vector nos indica la probabilidad de que la fotografía pertenezca a cada una de las emociones. Este vector se devuelve como resultado.

22 https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

5.2.7 Módulo classification_aus

El código del módulo se puede encontrar en la carpeta “code/modules/classification_aus”. El fichero principal es “classify.py”.

Este módulo funciona de forma similar al anterior. La diferencia principal es que en lugar de un único modelo de clasificador entrenado existe uno por cada Action Unit clasificada. Estos modelos se cargan al inicializarse el módulo. Posteriormente, para cada petición de clasificación, se clasifica cada AU de manera individual con su clasificador correspondiente. Esto nos devuelve un verdadero o falso a la presencia de cada una de las AUs. Esta información es la que devuelve el módulo.

6. Instalación del detector

En el servidor se ha instalado Ubuntu 15.04 Server de 64 bits. Para servir la página web se utiliza un servidor Nginx[23]. Los archivos de la web cliente se copian en la carpeta "/var/ww/html".

Como los módulos están programados en Python se debe instalar junto con uWSGI[24] (para que Nginx se comunique con los módulos Python). Todo esto se puede instalar con una orden como:

```
sudo apt-get install python2.7 python-pip uwsgi uwsgi-plugin-python nginx
```

Los módulos Python desarrollados se copian en "/usr/share/nginx/www".

Una buena práctica al instalar aplicaciones Python en un servidor es instalar un entorno virtual ("virtualenv") para cada aplicación. Así , si diferentes aplicaciones necesitan diferentes versiones de Python o de alguna de las bibliotecas no habrá conflictos. Para instalar *virtualenv* ejecutamos:

```
sudo pip install virtualenv
```

Para cada módulo activaremos el entorno con:

```
cd /usr/share/nginx/www/carpeta_del_modulo
sudo virtualenv venv-nombre_del_modulo
```

Y se instalarán las bibliotecas necesarias en ellos:

- scikit-leran
- scikit-image
- numpy
- opencv
- libboost
- bottle

Cada módulo instalará las bibliotecas necesarias por el mismo de entre las anteriores. Para ello se puede usar pip, como por ejemplo en el caso de Bottle:

```
sudo venv-nombre_del_modulo/bin/pip install bottle
```

El siguiente paso es configurar uWSGI. Para ello se creará un fichero para cada módulo en la carpeta "/etc/uwsgi/apps-available" y un enlace a los mismos en la carpeta "/etc/uwsgi/apps-enabled". Los ficheros se pueden ver en la carpeta "configuration\uwsgi" que acompaña a esta memoria. Como ejemplo mostramos el fichero para el módulo *emotions*:

```
[uwsgi]
socket = /run/uwsgi/app/emotions/socket
chdir = /usr/share/nginx/www/emotions
master = true
plugins = python
file = emotionsV1.py
```

23 https://nginx.org/

24 https://uwsgi-docs.readthedocs.io/en/latest/

```
virtualenv = venv-emotions
uid = www-data
gid = www-data
vacuum = true
```

Para terminar es necesario configurar Nginx. Para ello se crea el fichero “/etc/nginx/sites-available/emotions” (y un enlace al mismo en “/etc/nginx/sites-enabled/emotions”) con el siguiente contenido:

```
server {
    listen 4040;
    server_name 92.222.29.136;

    location / {
   root  /var/www/html;
   index  index.html;
    }

    location /emotions/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/emotions/socket;
    }

    location /landmarks/v1/saragih/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/landmarks_Saragih/socket;
    }

    location /normalization/v1/face/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/normalization_face/socket;
    }

    location /features/v1/LBP/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/features_LBP/socket;
    }

    location /features/v1/HOG/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/features_HOG/socket;
    }

    location /classification/v1/emotions/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/classification_emotions/socket;
    }

    location /classification/v1/aus/ {
        include uwsgi_params;
        uwsgi_pass unix:/run/uwsgi/app/classification_aus/socket;
    }

}
```

7. Entrenamiento de los clasificadores

Una vez instalado el detector es necesario suministrar unos modelos a los clasificadores para que puedan usarlos de referencia para clasificar correctamente las fotografías. Puesto que los clasificadores utilizados se basan en SVM (ver punto 4.4), que es un algoritmo de aprendizaje supervisado, necesitaremos un conjunto de muestras de entrenamiento etiquetadas con las que entrenar el clasificador y crear estos modelos. En el caso del detector, será necesaria una base de datos de fotografías que nos indique, para cada fotografía, qué emoción aparece y qué AUs están activas.

7.1 Bases de datos

Para entrenar a los clasificadores se ha utilizado la base de datos Cohn-Kanade Extendida (CK+) [12].

Esta base de datos consta de 593 secuencias de fotos de 123 sujetos. El tamaño de las secuencias varía en número de fotos. Los sujetos están entre 18 y 50 años, hay un 69% de mujeres, un 81% de euroamericanos, un 13% de afroamericanos y 6% de otros grupos.

Cada secuencia empieza en una pose neutra y termina en el pico de expresividad. Las imágenes son de la cara y frontales.

Cada secuencia incluye un archivo con la especificación de qué Action Units se encuentran en el gesto final.

En 327 secuencias se dispone de información de qué emoción se muestra. Las emociones vienen codificadas como:

- 0=neutral
- 1=anger (ira)
- 2=contempt (desprecio)
- 3=disgust (asco)
- 4=fear (miedo)
- 5=happy (alegría)
- 6=sadness (tristeza)
- 7=surprise (sorpresa)

Ilustración 24. Imágenes de la base de datos Cohn-Kanade.

7.2 Obtención del vector de características

Para no tener que calcular cada vez que queremos entrenar un clasificador el vector de características de todas las muestras de entrenamiento, se ha creado un script que lo calcula y vuelca el resultado a disco. De esta forma solo será necesario cargar este volcado, proceso mucho más rápido. El código de dicho script se encuentra en “code/training/get_features_vector.py”.

Este script, aunque se ejecuta en local, utiliza los módulos ya instalados en el servidor de forma que aseguramos que los descriptores generados son los mismos que tendremos en producción. Para ello se hacen llamadas a los módulos del servidor mediante el API REST de los mismos. Es por ello que para esta fase de entrenamiento es necesario haber hecho previamente la instalación del servidor.

Hay que tener en cuenta que si se quieren variar los datos del vector de características será necesario volver a ejecutar el script. Si bien el proceso hasta la normalización de la cara ha permanecido igual en los diferentes entrenamientos, los parámetros del LBP sí que se han modificado. También se han hecho pruebas con descriptores diferentes como HOG+LBP.

7.3 Entrenamiento de los clasificadores de emociones

Para entrenar los diferentes clasificadores se ha utilizado un 90% de imágenes en el conjunto de entrenamiento y un 10% en el conjunto de test, cogiendo la imagen pico de expresión de cada secuencia de la base de datos Cohn-Kanade. Como el número de muestras de la base de datos es pequeño, es necesario utilizar el máximo posible en el conjunto de entrenamiento. Si se probaba con un ratio de 80% de muestras en el conjunto de entrenamiento, la calidad del clasificador bajaba.

Inicialmente se ha intentado usar conjuntos de entrenamiento con el mismo número de muestras en cada emoción, para que las categorías estén balanceadas, pero debido al escaso número en algunas emociones los resultados no son adecuados. Al tener contempt únicamente 18 muestras totales, 16 de entrenamiento al coger el 90%, fuerza al resto de emociones a usar igualmente 16 muestras o menos, algo totalmente insuficiente para el entrenamiento. El número total de muestras de cada emoción disponible en la base de datos es:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
samples	45	18	59	25	69	28	83

Como ejemplo de este problema se muestran los resultados cogiendo 15 muestras de cada emoción para el entrenamiento, con un vector de características construido con LBP(P=8, R=1):

Exactitud	68.02%
Training elements	105 (32.11%)
Test elements	222

Muestras por emoción:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Training	15	15	15	15	15	15	15
Test	30	3	44	10	54	13	68

Matriz de confusión:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	12	1	3	2	3	8	1
Contempt	0	3	0	0	0	0	0
Disgust	7	0	30	1	3	3	0
Fear	1	2	1	4	1	1	0
Happy	0	1	6	2	44	0	1
Sadness	6	2	0	0	0	5	0
Surprise	5	2	4	1	1	2	53

Matriz de confusión en porcentajes:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	40	3.33	10	6.67	10	26.67	3.33
Contempt	0	100	0	0	0	0	0
Disgust	15.9	0	68.19	2.27	6.82	6.82	0
Fear	10	20	10	40	10	10%	0
Happy	0	1.85	11.11	3.7	81.48	0	1.85
Sadness	46.15	15.38	0	0	0	38.46	0
Surprise	7.35	2.94	5.88	1.47	1.47	2.94	77.94

Como se puede observar, la exactitud se queda en un 68%, dando un clasificador pobre y bastante lejos de valores deseados. Además el ratio de exactitud varía mucho entre emociones, pasando de un 100% para contempt a un 38% para sadness.

Para solucionar este problema se ha optado por no coger categorías balanceadas.

Otra forma de entrenar puede ser seleccionar un 90% de muestras para el conjunto de entrenamiento de forma aleatoria, sin tener en cuenta a qué emoción etiquetan. Se han utilizado las funciones de scikit-learn:

```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, Y, test_size=0.1, random_state=0)
```

Y:

```
scores = cross_validation.cross_val_score(clf, X, Y, cv=10)
```

Para realizar 10 pruebas sobre las muestras con *cross validation*, con el resultado:

```
[ 0.77142857, 0.94285714, 0.8, 0.82352941, 0.73529412, 0.65625, 0.75, 0.90625, 0.8, 0.75 ]
Accuracy: 0.79 (+/- 0.16)
```

La exactitud ha aumentado a un 79%, pero sigue siendo baja. Si observamos los resultados con detalle, vemos que la desviación es de +/- 0.16, dando resultados desde 94% a 66% de exactitud. Por tanto, dependiendo de las muestras seleccionadas para el entrenamiento la variabilidad es muy grande.

Juntando estos datos con los obtenidos al entrenar con categorías de emociones balanceadas se observa que hay emociones que se detectan más fácilmente. Si al seleccionar las muestras de forma aleatoria todas son de las emociones más fáciles de detectar, tendremos una exactitud muy alta, y al revés en el caso contrario. Esto puede justificar los resultados del *cross validation*. El problema es que en el caso con mayor exactitud puede que haya emociones que no aparezcan en el conjunto de entrenamiento y que nunca puedan ser detectadas.

Necesitamos por tanto asegurar que todas las emociones aparezcan lo suficiente en el conjunto de entrenamiento. Para ello se ha utilizado en el entrenamiento un 90% de muestras de cada emoción.
El código para este entrenamiento se puede encontrar en el fichero "code/training/training_1_img_per_seq.py". Si se entrena el clasificador de esta manera para un vector de características construido con LBP(8, 1), es decir, con un vecindario de 8 píxeles y un radio de 1, tenemos:

Descriptor	LBP(8, 1)
Exactitud	65.62%
Training elements	295
Test elements	32

Muestras por emoción:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Training	41	16	53	23	62	25	75
Test	4	2	6	2	7	3	8

Matriz de confusión:

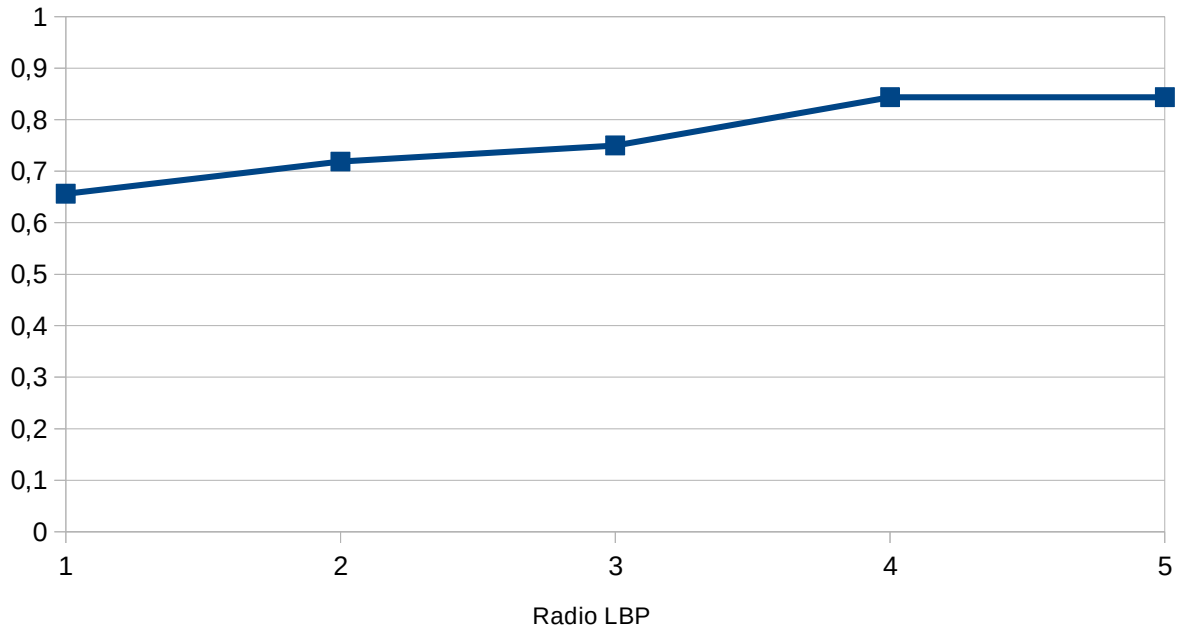
	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	0	2	0	1	0	1	0
Contempt	0	1	0	1	0	0	0
Disgust	1	0	5	0	0	0	0
Fear	0	1	0	1	0	0	0
Happy	1	0	0	0	6	0	0
Sadness	0	1	0	2	0	0	0
Surprise	0	0	0	0	0	0	8

Matriz de confusión en porcentajes:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	0	50	0	25	0	25	0
Contempt	0	50	0	50	0	0	0
Disgust	16.7	0	83.3	0	0	0	0
Fear	0	50	0	50	0	0	0
Happy	14.29	0	0	0	85.71	0	0
Sadness	0	33.3	0	66.67	0	0	0
Surprise	0	0	0	0	0	0	100

A pesar de que ahora podemos garantizar que todas las emociones están presentes en el entrenamiento, y que por tanto el clasificador servirá para cualquier emoción, el resultado ha bajado hasta el 65.62% de exactitud, en el rango bajo del *cross validation*. Además hay emociones que llegan a tener un ratio de exactitud del 0%.

Tras diversas pruebas se ha observado que el radio utilizado en el descriptor LBP influye mucho en el resultado. Por ello se hacen pruebas con diferentes radios. Los ratios de exactitud conseguidos dependiendo del radio LBP(8, R) se muestran en la siguiente gráfica:



Utilizando un LBP(8, 4) y LBP(8,5) llegamos a tener una exactitud de 84.38%, casi 20 puntos por encima del LBP(8, 1) usado anteriormente. Los resultados para LBP(8, 4) son:

Descriptor	LBP(8, 4)
Exactitud	84.38%
Training elements	295
Test elements	32

Muestras por emoción:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Training	41	16	53	23	62	25	75
Test	4	2	6	2	7	3	8

Matriz de confusión:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	3	0	0	0	0	1	0
Contempt	0	1	0	0	0	1	0
Disgust	0	0	6	0	0	0	0
Fear	0	0	0	2	0	0	0
Happy	0	0	0	1	6	0	0
Sadness	1	1	0	0	0	1	0
Surprise	0	0	0	0	0	0	8

Matriz de confusión en porcentajes:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	75	0	0	0	0	25	0
Contempt	0	50	0	0	0	50	0
Disgust	0	0	100	0	0	0	0
Fear	0	0	0	100	0	0	0
Happy	0	0	0	14.29	85.71	0	0
Sadness	33.33	33.33	0	0	0	33.33	0
Surprise	0	0	0	0	0	0	100

Se ve que todas las emociones han pasado a tener un ratio por encima de 50% excepto sadness. Si calculamos los porcentajes para LBP(8, 5):

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	75	0	0	0	0	25	0
Contempt	0	50	0	0	0	50	0
Disgust	0	0	100	0	0	0	0
Fear	0	0	0	100	0	0	0
Happy	0	0	0	28.57	71.43	0	0
Sadness	0	33.33	0	0	0	66.67	0
Surprise	0	0	0	0	0	0	100

Vemos que el resultado es parecido, bajando un poco happy y subiendo sadness, teniendo ahora todas las emociones por encima del 50% de exactitud.

Si observamos las tablas anteriores, vemos que las emociones con pocas muestras de entrenamiento suelen dar peores porcentajes. Puesto que Cohn-Kanade no tiene más secuencias etiquetadas, se opta por coger más de una fotografía por secuencia para aumentar el número de muestras. Hasta ahora se cogía solo la imagen con el pico de representación de la emoción de cada secuencia. Se eligen ahora las 3 últimas imágenes de la secuencia, la del pico y las dos anteriores. Se eligen estas por ser las que contienen una mayor expresividad de la emoción. Este método nos permitirá aumentar el número de muestras con el inconveniente de que habrán muestras muy parecidas unas con otras (las de la misma secuencia). Esto podría llevarnos a un sobreajuste (overfitting) del clasificador para la base de datos. El código para estas pruebas puede encontrarse en "code/trainig/training_3_img_per_seq.py".

Con estas muestras, el resultado para el entrenamiento con LBP(8, 4) es:

Descriptor	LBP(8, 4)
Exactitud	85,57%
Training elements	884
Test elements	97

Muestras por emoción:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Training	122	49	159	68	186	76	224
Test	13	5	18	7	21	8	25

Matriz de confusión:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	7	3	0	0	0	3	0
Contempt	0	2	0	0	0	1	2
Disgust	0	0	18	0	0	0	0
Fear	0	0	0	7	0	0	0
Happy	0	0	0	3	18	0	0
Sadness	0	0	0	0	0	8	0
Surprise	0	0	2	0	0	0	23

Matriz de confusión en porcentajes:

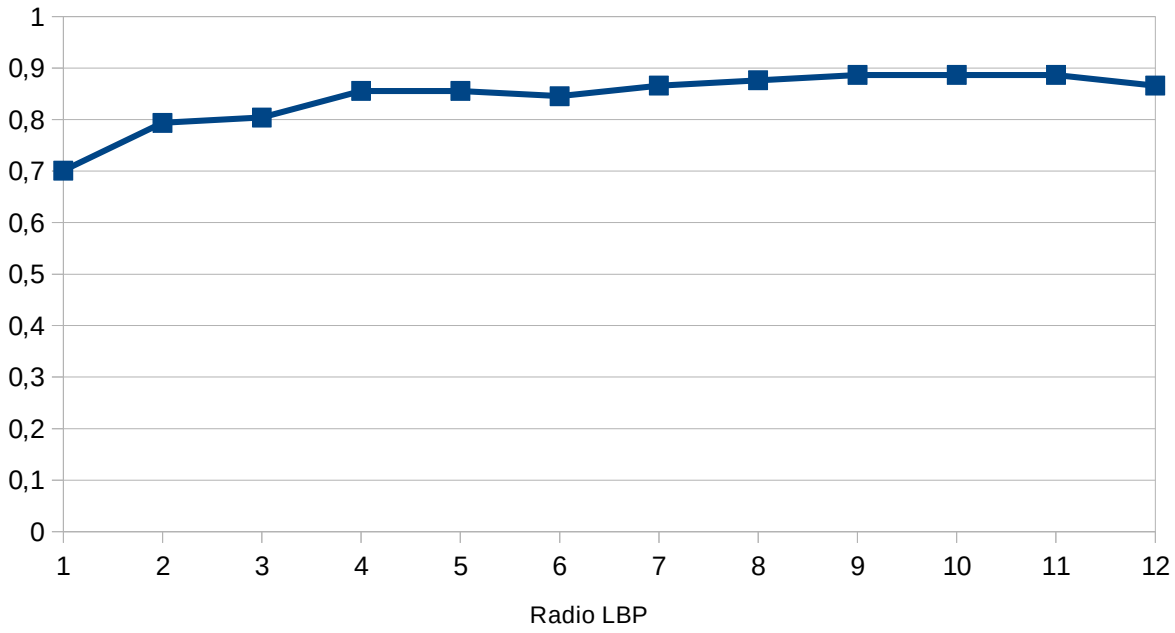
	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	53.85	23.08	0	0	0	23.08	0
Contempt	0	40	0	0	0	20	40
Disgust	0	0	100	0	0	0	0
Fear	0	0	0	100	0	0	0
Happy	0	0	0	14.29	85.71	0	0
Sadness	0	0	0	0	0	100	0
Surprise	0	0	8	0	0	0	92

Se ha mejorado el resultado respecto al ejemplo con una sola imagen por secuencia en un punto porcentual (1.19%).

Los porcentajes para LBP(8, 5) son:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	75	0	0	0	0	25	0
Contempt	0	50	0	0	0	50	0
Disgust	0	0	100	0	0	0	0
Fear	0	0	0	100	0	0	0
Happy	0	0	0	28.57	71.43	0	0
Sadness	0	33.33	0	0	0	66.67	0
Surprise	0	0	0	0	0	0	100

Como ya se había observado, el radio del algoritmo LBP tiene una influencia grande. Utilizando el nuevo conjunto de muestras, con 3 imágenes por secuencia, podemos calcular la exactitud para diferentes radios:



Los radios con mejores resultados son 9, 10 y 11, con una exactitud del 88.66%. Los resultados para uno ellos, cogiendo como referencia el LBP(8,11) son:

Descriptor	LBP(8, 11)
Exactitud	88,66%
Training elements	884
Test elements	97

Muestras por emoción:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Training	122	49	159	68	186	76	224
Test	13	5	18	7	21	8	25

Matriz de confusión:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	13	0	0	0	0	0	0
Contempt	0	2	0	0	0	1	2
Disgust	0	0	18	0	0	0	0
Fear	0	0	0	7	0	0	0
Happy	0	0	0	3	15	0	3
Sadness	0	0	0	0	0	8	0
Surprise	0	0	2	0	0	0	23

Matriz de confusión en porcentajes:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	100	0	0	0	0	0	0
Contempt	0	40	0	0	0	20	40
Disgust	0	0	100	0	0	0	0
Fear	0	0	0	100	0	0	0
Happy	0	0	0	14.29	71.4	0	14.29
Sadness	0	0	0	0	0	100	0
Surprise	0	0	8	0	0	0	92

Si se observan los diversos clasificadores anteriores, se aprecia que una gran parte del error se encuentra en la emoción contempt. Esto puede deberse a que es la emoción con menos secuencias. También se observa que en muchos estudios la emoción comtempt no aparece incluida. Como se indicó en el punto 2.1.2, contempt es la única de las emociones que se están analizando que no pertenece a las básicas según Ekman. Por ello se hace una prueba con solo 6 emociones sin considerar a contempt, con 3 imágenes por secuencia. Tanto LBP(8,4) como LBP(8,11) han dado el mismo porcentaje de exactitud. Se muestra el resultado para LBP(8, 4).

Descriptor	LBP(8, 4)
Exactitud	91.3%
Training elements	835
Test elements	92

Muestras por emoción:

	Anger	Disgust	Fear	Happy	Sadness	Surprise
Training	122	159	68	186	76	224
Test	13	18	7	21	8	25

Matriz de confusión:

	Anger	Disgust	Fear	Happy	Sadness	Surprise
Anger	10	0	0	0	3	0
Disgust	0	18	0	0	0	0
Fear	0	0	7	0	0	0
Happy	0	0	3	18	0	0
Sadness	0	0	0	0	8	0
Surprise	0	2	0	0	0	23

Matriz de confusión en porcentajes:

	Anger	Disgust	Fear	Happy	Sadness	Surprise
Anger	76.92	0	0	0	23.08	0
Disgust	0	100	0	0	0	0
Fear	0	0	100	0	0	0
Happy	0	0	14.29	85.71	0	0
Sadness	0	0	0	0	100	0
Surprise	0	8	0	0	0	92

Se comprueba que sin contempt se llega a un 91.3% de exactitud, siendo la emoción con peor porcentaje anger.

Además del radio de LBP, otro parámetro que se ha tenido en cuenta es el kernel utilizado en las SVM. Los kernels que mejor han funcionado son el lineal y el polinómico. Si ejecutamos el clasificador con diferentes grados, obtenemos las siguientes exactitudes:

Lineal	82,47%
Grado 2	85,57%
Grado 3	86,6%
Grado 4	87,63%
Grado 5	87,63%
Grado 6	88,66%
Grado 7	88,66%
Grado 8	88,66%
Grado 9	87,63%
Grado 10	87,63%

Para los entrenamientos previos, que nos daban los mejores resultados, se ha utilizado un kernel de grado 6 con los parámetros:

```
params['kernel'] = 'poly'
params['degree'] = 6
params['C'] = 1.0
params['coef0'] = 100.0
params['gamma'] = 0.001
params['class_weight'] = 'balanced'
params['decision_function_shape'] = 'ovr'
```

7.3.1 Comparativa

Para evaluar los resultados obtenidos, se ha comparado el detector creado con estudios de otros investigadores que también usan como técnica LBP+SVM y la base de dato CK+.

	Nuestro detector	Shan et al. [19] LBP	Sadeghi et al. [20]
Anger	76.2	89.7	92.59
Disgust	100	97.5	97.70
Fear	100	73	88.89
Happy	85.71	97.9	96.14
Sadness	100	83.5	88.10
Surprise	92	98.7	96.39
Total	91.3	92.6	94.68

Se puede ver que los resultados totales son bastante similares, siendo en nuestro método un poco

más bajo. Esto es debido a que los dos estudios utilizan métodos dinámicos, usando todas las imágenes de las secuencias, mientras que nuestro detector utiliza métodos estáticos, utilizando únicamente una fotografía para clasificar. Como ya se vio en estado del arte, los métodos dinámicos ofrecen mejores resultados, pero como el objetivo del trabajo era crear un detector en imágenes estáticas no podían utilizarse.

7.4 Entrenamiento de los clasificadores de Action Units

Para clasificar AUs se han seleccionado las AUs presentes en 75 o más muestras, descartándose el resto. De esta forma se asegura un mínimo de muestras para poder hacer el entrenamiento. Las AUs que cumplen este requisito son:

Código Action Unit	Acción
1	Levantamiento interior de ceja
2	Levantamiento exterior de ceja
4	Bajar cejas
5	Levantamiento del párpado superior
6	Levantamiento de mejillas
7	Apretar parpado(s)
9	Arrugar la nariz
12	Tiramiento labial esquinal
15	Depresión labial esquinal
17	Levantamiento de mejillas
20	Apretar los labios
25	Deslizamiento labial
27	Apretamiento bucal

Para cada una de estas AU se ha creado un clasificador binario que nos indique si la acción está presente (1) o no (0). Cada muestra de los conjuntos de entrenamiento y test puede tener varias AUs activas (de hecho esto es lo normal). El código de este entrenamiento se puede encontrar en el fichero “code/training/training Aus.py”.

Para entrenar los clasificadores se ha utilizado la imagen pico de cada secuencia y un descriptor con LBP(8, 4). El resultado para cada clasificador es el siguiente:

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
1	81.67%	374	159	42	18

Unidades	0	1
0	32	10
1	1	17

%	0	1
0	76.19	23.81
1	5.55	94.44

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
2	83.33%	426	107	50	10

Unidades	0	1
0	45	5
1	5	5

%	0	1
0	90	10
1	50	50

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
4	70%	362	171	37	23

Unidades	0	1
0	30	7
1	11	12

%	0	1
0	81.08	18.92
1	47.83	52.17

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
5	78.33%	448	85	43	17

Unidades	0	1
0	41	2
1	11	6

%	0	1
0	95.35	4.65
1	64.71	35.29

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
6	95%	415	118	55	5

Unidades	0	1
0	54	1
1	2	3

%	0	1
0	98.18	1.82
1	40	60

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
7	86.67%	415	118	57	3

Unidades	0	1
0	52	5
1	3	0

%	0	1
0	91.23	8.77
1	100	0

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
9	98.33%	460	73	58	2

Unidades	0	1
0	58	0
1	1	1

%	0	1
0	100	0
1	50	50

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
12	83.33%	414	119	48	12

Unidades	0	1
0	45	3
1	7	5

%	0	1
0	93.75	6.25
1	58.33	41.67

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
15	85%	452	81	47	13

Unidades	0	1
0	45	2
1	7	6

%	0	1
0	95.74	4.26
1	53.85	46.15

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
17	76.67%	352	181	39	21

Unidades	0	1
0	30	9
1	5	16

%	0	1
0	76.92	23.08
1	23.81	76.19

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
20	86.67%	462	71	52	8

Unidades	0	1
0	47	5
1	3	5

%	0	1
0	90.38	9.62
1	37.5	62.5

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
25	68.33%	233	300	36	24

Unidades	0	1
0	23	13
1	6	18

%	0	1
0	63.89	36.11
1	25	75

AU	Exactitud	Training elems. 0	Training elems. 1	Test elems. 0	Test elems. 1
27	98.33%	458	75	54	6

Unidades	0	1
0	54	0
1	1	5

%	0	1
0	100	0
1	16.67	83.33

La exactitud total del sistema es de un 83.97%, similar a su equivalente en los entrenamientos de emociones (LBP(8, 4) con una imagen por secuencia) que tenía una exactitud de 84.38%. Este resultado puede ser engañoso, ya que el hecho de tener muchas más muestras 0 que 1 en cada clasificador parece estar desviando los resultados hacia el 0. En general, los clasificadores tienden a clasificar como 0.

7.5 Pruebas adicionales

Además de entrenar los clasificadores se han hecho pruebas adicionales que finalmente no se han utilizado en el detector. Esto puede ser bien porque no han dado los resultados esperados, bien porque eran pruebas de validación.

Para intentar mejorar el descriptor, se ha utilizado un vector de características con la unión de los algoritmos HOG y LBP. El resultado en cuanto a exactitud ha sido prácticamente igual a utilizar únicamente LBP. Por ello se ha descartado su uso, puesto que supondría un mayor tiempo de cómputo sin aportar ningún beneficio.

Como se observa que varios clasificadores SVM dan pequeñas diferencias en el porcentaje de exactitud para cada emoción, se ha probado a realizar un conjunto (*ensemble*) de clasificadores. Para ello se ha usado un conjunto por votación mayoritaria (*majority vote ensemble*). Consiste en clasificar una muestra con varios clasificadores y elegir como resultado la clasificación que le asignan un mayor número de ellos.

Como ejemplo se ha clasificado un conjunto de test con 8 clasificadores SVM dando como resultado las siguientes categorías para 32 muestras:

```
[3 3 7 1 7 3 3 5 7 7 5 5 3 7 5 7 1 7 5 7 5 6 2 4 2 4 4 4 4 2 2 2]
[3 3 7 3 7 3 3 5 7 7 5 5 1 7 5 7 5 7 5 7 5 6 3 2 1 2 4 4 4 2 2 4]
[3 3 7 1 7 3 3 5 7 7 5 5 3 7 5 7 5 7 4 7 5 6 1 2 1 1 2 4 4 6 1 2]
[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 5 7 4 7 5 6 1 2 1 1 2 4 6 6 1 4]
[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 4 7 4 7 5 6 1 2 6 1 2 4 6 6 1 4]
[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 4 7 4 7 5 6 1 2 6 1 2 4 1 6 1 4]
[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 4 7 4 7 5 6 1 2 6 1 2 4 1 6 1 4]
[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 5 7 5 7 5 6 1 2 1 1 2 4 7 1 1 4]
```

Si calculamos el voto mayoritario, obtenemos la siguiente clasificación para las muestras:

[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 5 7 4 7 5 6 1 2 1 1 2 4 4 6 1 4]

El valor real de las muestras es el siguiente:

[3 3 7 3 7 3 3 5 7 7 5 5 3 7 5 7 5 7 5 7 5 1 1 6 6 1 2 4 6 2 1 4]

Por lo que se puede ver que existen 6 muestras mal clasificadas, para una exactitud del 81.25%. En general la exactitud ha sido muy similar a la obtenida con los mejores clasificadores individuales. Debido a que SVM busca la separación óptima tiene sentido que no tengamos mejora o muy poca. Por ello, no es rentable aplicar el ensemble ya que el resultado es equivalente a un solo clasificador y el tiempo de cálculo sube al tener que clasificar varias veces.

Para comparar con otros métodos se ha programado una red neuronal muy sencilla mediante la librería ConvnetJS[25]. El código está disponible en la carpeta “code/nnem”. Se ha entrenado con los mismos conjuntos que el clasificador SVM con LBP(8, 4) y una sola imagen por secuencia. La red neuronal se crea con dos capas ocultas de 20 neuronas cada una. Se entrena con el algoritmo *adadelta* con l2_decay = 0.0001 y batch_size = 10. La capa de salida usa algoritmo *softmax*. Los resultados obtenidos han sido:

exactitud SVM: 85.57%
exactitud red neuronal: 86.6%

25 http://cs.stanford.edu/people/karpathy/convnetjs/

La ejecución de la red neuronal después de 1480 pasos es:

Current epoch: 1480

Confusion matrix:

	Anger	Contempt	Disgust	Fear	Happy	Sadness	Surprise
Anger	6	3	1	0	0	3	0
Contempt	0	2	0	0	0	0	3
Disgust	0	0	18	0	0	0	0
Fear	0	0	0	7	0	0	0
Happy	0	0	0	1	20	0	0
Sadness	0	0	0	0	0	8	0
Surprise	0	0	2	0	0	0	23

Loss:
L2 decay loss: 0.00006499660825826706
Cost loss: 0.0000055282842728506 63
Time series:

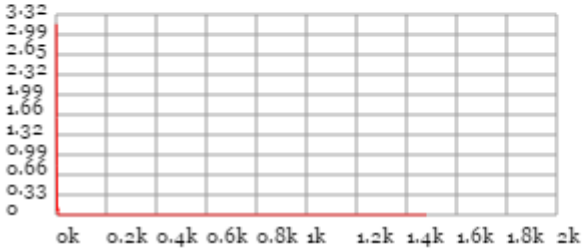


Ilustración 25. Resultados de la red neuronal tras 1480 iteraciones.

Se puede ver que los porcentajes de exactitud son prácticamente iguales en los dos tipos de clasificación. También las matrices de confusión se asemejan mucho, con pequeñas variaciones.

En cuanto a la clasificación de Action Units, se ha intentado que todos los clasificadores individuales estén balanceados, para que tengan el mismo número de muestras con y sin la AU en el conjunto de entrenamiento. Tras probarlo se ha descartado, ya que el porcentaje de exactitud bajaba mucho, generando un peor clasificador. Muy probablemente se deba a la falta de muestras suficientes en el conjunto de entrenamiento.

8. Conclusiones

El objetivo principal del trabajo era crear un detector de emociones funcional accesible vía web. El detector no solo consiste en la implementación de la interfaz web, sino en la utilización de algoritmos que permitan analizar las fotografías que se le pasan y clasificar dichas fotografías según las emociones y Action Units presentes en las mismas. Este análisis y clasificación era la parte más compleja del proyecto.

Inicialmente se ha desarrollado una página web que permita al usuario utilizar el detector. Esta web se ha realizado cumpliendo unos requisitos de simplicidad y accesibilidad desde diferentes dispositivos. Para ello se ha creado una página *Single Page Application* (SPA), de forma que desde una única página se utilice todo el sistema. Esta misma página sirve para mostrar los resultados finales. Además la página es *responsive*, lo que permite adaptar la visualización de su contenido a diversos dispositivos, como ordenadores de escritorio, móviles o tabletas. Esta página se ha creado con tecnologías web estándar y se ha podido realizar sin que aparecieran grandes complejidades.

En una segunda fase se ha desarrollado un servidor que implementa el detector propiamente dicho. Se ha realizado utilizando una arquitectura de microservicios, que permitirá ampliar el sistema mediante la utilización de módulos. Para poder comunicar los diferentes componentes se ha diseñado una API REST propia. La implementación con estas arquitecturas eran desconocidas por el alumno en el momento de iniciar el trabajo, por lo que ha servido para aprender de manera práctica su uso.

La parte principal del proyecto ha sido la implementación y parametrización de los métodos y algoritmos que permiten clasificar las fotografías. Se han desarrollado módulos para implementar los algoritmos:

- Detección de puntos característicos. Se ha utilizado la librería de acceso público *FaceTracker* escrita en C++. El principal trabajo ha sido la compilación y creación de un envoltorio para poder usarla desde Python.
- Normalización de la cara. Este módulo ha sido el más sencillo de desarrollar, ya que se compone de transformaciones afines simples.
- *Local Binary Patterns.* La implementación del módulo ha sido sencilla, puesto que skimage ya ofrece funciones con el algoritmo. Lo más importante de este módulo es su parametrización.
- *Support Vector Machines*, ha sido el método implementado para clasificación. Se ha desarrollado también una red neuronal simple, pero SVM es el método usado en el detector final.

La parametrización de los algoritmos ha sido fundamental para conseguir mejorar la exactitud del detector y ha sido una de las partes donde más tiempo se ha utilizado. Hay que hacer muchas pruebas con diferentes valores para poder compararlos y algunas de estas pruebas son costosas temporalmente. A base del estudio de los diferentes parámetros y de cómo afectan a los resultados se ha conseguido aumentar la exactitud de un 65.62% inicial a un 88.66% en el caso de la clasificación de las siete emociones estudiadas. Si quitamos de la clasificación el desprecio (contempt) y clasificamos estrictamente las 6 emociones básicas identificadas por Ekman, se llega a una exactitud del 91.3%.

Los parámetros que más afectan a la exactitud final del detector son los asociados al algoritmo LBP. El tamaño del vecindario más adecuado es 8 y cuando se subía a valores como 12 o 16, la exactitud bajaba. Pero el parámetro que ha resultado crítico ha sido el radio. Solo cambiando este parámetro se ha aumentado en un 20% la exactitud global.

También el kernel seleccionado en las SVM tiene repercusión en los resultados, pero en este caso en menor medida que el radio LBP. El kernel utilizado finalmente, por ser de los que mejor resultado daba, ha sido el polinómico de grado 6.

La clasificación de Action Units ha resultado más difícil. Los clasificadores tienen tendencia a clasificar hacia el 0 (no activación de la AU), probablemente debido a una falta de muestras en el conjunto de entrenamiento. El clasificador ha dado una exactitud del 83,97%, pero en gran medida por dicha desviación hacia el 0.

El tamaño de la base de datos ha sido una de las principales dificultades encontradas, tanto al clasificar emociones como AUs. Como ejemplo, al aplicar un 90% de muestras al conjunto de entrenamiento, quedan únicamente 16 secuencias para la emoción desprecio, número demasiado bajo para poder realizar un entrenamiento eficaz.

Otra dificultad es el hecho que las fotografías que aparecen en la base de datos Cohn-Kanade son actuadas y frontales. Esto hace que al intentar clasificar fotografías del mundo real, más naturales y en diversos ángulos, el clasificador encuentre dificultad en obtener una detección adecuada.

Este punto sería importante intentar mejorarlo como trabajo futuro, intentando conseguir o generando una base de datos más completa.

Otra mejora sería intentar localizar un descriptor diferente al formado únicamente con LBP que diera mejores clasificaciones. Puesto que LBP opera sobre la textura, podría ser interesante combinarlo con un descriptor geométrico. En algunos casos el descriptor LBP puede verse afectado por imágenes con ruido, como el generado al interpolar la fotografía para normalizar la cara. En este caso el descriptor geométrico podría ayudar a crear un mejor descriptor.

En el caso del clasificador, SVM ha dado un resultado bueno, pero las pruebas con redes neuronales indican que es una posibilidad a estudiar. Se ha desarrollado una red muy sencilla y con muy pocos ajustes de parámetros y, a pesar de ello, se ha obtenido un resultado parejo al clasificador con SVM. Ello hace pensar que un desarrollo con una red neuronal bien ajustada al problema puede dar unos resultados prometedores.

El detector solo es capaz de clasificar una cara por imagen y una mejora inmediata debería ser poder trabajar con fotografías con múltiples caras.

Ya saliendo de mejoras del sistema desarrollado, otra posible línea de futuro es el desarrollo de un detector capaz de operar en vídeos en tiempo real. Como ya se vio en el estado del arte, los estudios parecen indicar que es más fácil detectar rasgos faciales en secuencias de imágenes y no en imágenes estáticas. Para ello habría que optar por otro tipo de arquitectura, no basada en un servidor central. Sería interesante el desarrollo de una librería en algún lenguaje de alto rendimiento, como C++, y utilizando aceleración por hardware. De esta manera podría integrarse en diversas aplicaciones.

Apéndice 1 - Lista de Action Units

Lista de Unidades de acción y descripciones de acción (con los músculos faciales subyacentes) extraída del artículo “Sistema de Codificación Facial”[26].

Códigos Principales

Número de acción	Nombre de acción
0	Rostro Neutral
1	Levantamiento interior de ceja
2	Levantamiento exterior de ceja
4	Bajar cejas
5	Levantamiento del párpado superior
6	Levantamiento de mejillas
7	Apretar parpado(s)
8	Labios encimados uno de otro
9	Arrugar la nariz
10	Levantamiento del labio superior
11	Profundidad nasolabial
12	Tiramiento labial esquinal
13	Tiramiento labial frontal
14	Hoyuelo facial
15	Depresión labial esquinal
16	Depresión labial frontal
17	Levantamiento de mejillas
18	Arruga labial
19	Muestro de lengua
20	Apretar los labios
21	Apretamiento de cuello
22	Embudo labial
23	Morder labios
24	Presión Labial
25	Deslizamiento labial
26	Caída de la mandíbula
27	Apretamiento bucal
28	Lamido labial
29	Tracción de la mandíbula
30	Deslizamiento de mandíbula
31	Contracción mandibular
32	Mordida labial

26 https://es.wikipedia.org/wiki/Sistema_de_Codificación_Facial

Número de acción	Nombre de acción
33	Succión de mejillas
34	Inflar mejillas
35	Soplido de mejillas
36	Protuberancia de lengua
37	Limpieza labial
38	Dilatado nasal
39	Compresión nasal
41	Bajeza de ojeras
42	Contracción retinal
43	Ojos cerrados
44	Recolector retinal
45	Parpadeo
46	Guiño

Códigos motores de cabeza

Número de acción	Nombre de acción
51	Girar cabeza a la izquierda
52	Girar cabeza a la derecha
53	Alzar la cabeza
54	Bajar la cabeza
55	Inclinación de la cabeza hacia la derecha
M55	Inclinación de la cabeza a la izquierda
56	Leve inclinación
M56	Inclinación derivada en el cuello
57	Head Forward
M57	Empujar cabeza hacia adelante
58	Empujar hacia atrás
M59	Agitar cabeza hacia arriba y abajo
M60	Agitar cabeza de lado a lado
M83	Inclinación diagonal

Códigos de movimiento ocular

Número de acción	Nombre de acción	
61	Mover ojos hacia la izquierda	
M61	Ojos a la izquierda	el cerebro tiende a recordar
62	Mover ojos a la derecha	
M62	Ojos a la derecha	El cerebro tiende a crear o mentir.

Número de acción	Nombre de acción	
63	Ojos hacia arriba	Demuestran necedad, o falta de interés
64	Ojos hacia abajo	Muestran miedo, depresión, negación
65	Leucoma	
66	Estrabismo	
M68	Deslizamiento ocular	
69	Fijar los ojos en otra persona	
M69	Posicionamiento ocular y frontal	

Códigos de visibilidad

Número de acción	Nombre de acción
70	Frente y cejas no visibles
71	Ojos no visibles
72	Mentón no visible
73	Rostro sin movimiento
74	Indescifrable

Códigos de comportamiento bruto

Estos códigos están reservados para el registro de información sobre conductas manifiestas que pueden ser relevantes para las acciones faciales a calificar.

Número de acción	Nombre de acción
40	Estornudo
50	Discurso
80	Tragar
81	Masticar
82	Encogimiento de hombros
84	Sacudir cabeza
85	Asentir con la cabeza
91	Movimiento instantáneo
92	Movimiento instantáneo parcial
97*	Temblar
98*	Barrido ocular

Apéndice 2 - APIs REST

Emotions

Esta API es la que utilizan los clientes para acceder al servicio. Se accede mediante la URI[27]

http://<server address>/emotions/v1

Método	HTTP request	Descripción
POST	/images	Crea un proceso de análisis de una imagen. Parámetros, por orden de prioridad (al menos uno): • *url*: Dirección de una imagen a descargar por el servidor. • *serverId:* Id de la imagen de ejemplo ofrecida por el servidor • *files*: Imagen a analizar subida mediante POST Resultado: • Fichero JSON.

Landmarks

Esta API es la que exponen los módulos de clase *Landmarks detector.* Se accede mediante la URI[28]

http://<server address>/landmarks/v1

Método	HTTP request	Descripción
POST	/<*algorithm*>/landmarks	Procesa una imagen mediante el algoritmo indicado en <algorithm>. Parámetros: • *files*: Imagen a analizar subida mediante POST. Resultado: • Fichero JSON con la estructura: {"landmarks": {"nose_bridge": {"1": {"x": 70.6, "y": 125.9}, … }}}

Los algoritmos (<algorithm>) disponibles son:

- Jason Saragih y Kyle McDonald.

27 Identificador de recursos uniforme
28 Identificador de recursos uniforme

Ejemplo de uso desde HTML:

```
<!DOCTYPE html>
<html>
<body>

<form action="http://<server address>/landmarks/v1/saragih/landmarks"
method="post" enctype="multipart/form-data">
  <input type="file" name="files" id="local_photo">
  <input type="submit">
</form>

</body>
</html>
```

Ejemplo de uso desde Python:

```
import requests

fd = open(file_path, 'rb')
files = {'files': fd}
r = requests.post(url, files=files)
landmarks = r.json()
fd.close()
```

Face normalization

Esta API es la que exponen los módulos de clase *Face normalization*. Se accede mediante la URI[29]

http://<server address>/normalization/v1/face

Método	HTTP request	Descripción
POST	/images	Normaliza una imagen. Parámetros: • lefteye_x: posición X del centro del ojo izquierdo. • lefteye_y: posición Y del centro del ojo izquierdo. • righteye_x: posición X del centro del ojo derecho. • righteye_y: posición Y del centro del ojo derecho. • files: Imagen a analizar subida mediante POST. • *eye_distance (optativo)*: distancia entre ojos. 50 píxeles por defecto. • *width (optativo)*: ancho de la imagen resultante. 100 píxeles por defecto.

29 Identificador de recursos uniforme

Método	HTTP request	Descripción
		• *height (optativo)*: dalto de la imagen resultante. 120 píxeles por defecto. • *eye_height (optativo)*: altura de los ojos respecto a la altura de la imagen en porcentaje (0.0 – 1.0). 0.25 por defecto. Resultado: • Fichero JSON.

Features extractor

Esta API es la que exponen los módulos de clase *Features extractor.* Se accede mediante la URI[30]

http://<server address>/features/v1

Método	HTTP request	Descripción
POST	/LBP/images	Extrae características de una imagen mediante el algoritmo LBP. Parámetros: • *n_points*: número de puntos de la vecindad. • *radius*: radio • *method*: Puede ser uno de: ◦ *original*: algoritmo LBP original. ◦ *uniform*: algoritmo LBP uniforme • files: Imagen a analizar subida mediante POST. Resultado: • Fichero JSON con la estructura: {"LBP": {"0": 1928, "1": 847, "2": 69, "3": 312, … }}

30 Identificador de recursos uniforme

Glosario

Action Units (AUs): unidades específicas de acción. Consisten en la codificación de movimientos musculares faciales como una contracción o relajación.

Algoritmo: es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos.

Clasificador: algoritmo utilizado para asignar un elemento entrante no etiquetado a una categoría concreta conocida.

Computación afectiva: estudio y el desarrollo de sistemas y dispositivos que pueden reconocer, interpretar, procesar y estimular las emociones humanas.

C**onjunto (ensemble) de clasificadores**: utilización de múltiples clasificadores de manera conjunta para mejorar los resultados de la clasificación.

Descriptor: conjunto de características que definen un objeto.

Detector: sistema que permite descubrir la existencia de algo que no era patente.

Emoción: alteración del ánimo intensa y pasajera, agradable o penosa, que va acompañada de cierta conmoción somática.

Expresión facial: gesto producido por los movimientos de ls músculos de la cara. Permite expresar emociones y estados de ánimo.

Facial Action Coding System (FACS): sistema para taxonomizar los movimientos faciales humanos por su apariencia en la cara.

Funciones kernel: funciones matemáticas utilizadas en las Support Vector Machines para convertir un problema no lineal en lineal en un espacio dimensional mayor.

Hiperplano: análogo al plano (de dos dimensiones) en el espacio n-dimensional.

Histograma de frecuencias: distribución por frecuencia de datos numéricos.

Identificador de recursos uniforme (URI): es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

Local Binary Patterns (LBP): operador aplicable a imágenes y texturas.

Microservicios: arquitectura de desarrollo basada en la creación de servicios pequeños y que suelen corresponder a un área de negocio del sistema.

Normalización: estandarización, hacer comparables resultados que no lo son directamente.

Puntos característicos (landmarks): puntos que identifican rasgos específicos.

Representational State Transfer (REST): estilo de arquitectura software para sistemas distribuidos.

Single page application (SPA) o aplicación de página única: es una aplicación o sitio web que cabe en una sola página.

Sobreajuste (overfitting): efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado.

Support vector machines (SVM): conjunto de algoritmos de aprendizaje supervisado.

Unified Modelling Language (UML): lenguaje de modelado de sistemas software.

V**ector de características**: conjunto de características que definen un objeto.

Bibliografía

[1] **Charles Darwin** (1872). "La expresión de las emociones en hombres y animales".

[2] **Paul Ekman** (1969). "Pan-Cultural Elements in Facial Display of Emotions". Revista Science.

[3] **Carl-Herman Hjortsjö** (1970). "Man's face and mimic language".

[4] **P. Ekman y W. Friesen** (1978). "Facial Action Coding System: A Technique for the Measurement of Facial Movement". Consulting Psychologists Press, Palo Alto.

[5] **Rosalind Picard** (1995). "Affective computing". MIT Media Laboratory.

[6] **J. M. Saragih, S. Lucey, and J. F. Cohn**. Face Alignment through Subspace Constrained Mean-Shifts. International Conference of Computer Vision (ICCV), September, 2009.

[7] **Jason Saragih et al**. "Deformable Model Fitting by Regularized Landmark Mean-Shift". Springer.

[8] **Kyle McDonald**. Entrevista en Makematics

[9] **Ojala, Pietikäinen y Harwood** (1996). "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51-59.

[10] **Vladimir Vapnik** (1995). "The Nature of Statistical Learning Theory". Springer-Verlag.

[11] **N. Dalal and B. Triggs** (2005). "Histograms of Oriented Gradients for Human Detection". INRIA.

[12] **P. Lucey et al** (2010). "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression". Proceedings of IEEE workshop on CVPR for Human Communicative Behavior Analysis.

[13] **Ambadar Z, Schooler JW, Cohn JF** (2005). "Deciphering the enigmatic face: The importance of facial dynamics in interpreting subtle facial expressions". Psychological Science 16(5):403–410

[14] **Y.l. Tian, T. Kanade, J.F. Cohn** (2001). "Recognizing action units for facial expression analysis". Pattern Anal. Mach. Intell. IEEE Trans. 23 97–115.

[15] **M.F. Valstar, I. Patras, M. Pantic**, (2005). "Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data". IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops. CVPRWorkshops, IEEE, p. 76.

[16] **Z. Zhang, M. Lyons, M. Schuster, S. Akamatsu** (1998). **“**Comparison between geometrybased and gabor-wavelets-based facial expression recognition using multi-layer perceptron, Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on, IEEE, pp. 454–459.

[17] **B. Jiang, M.F. Valstar, M. Pantic** (2011). “Action unit detection using sparse appearance descriptors in space-time video volumes”, Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on, IEEE, pp. 314–321.

[18] **C. Shan, S. Gong, P.W. McOwan** (2005). “Robust facial expression recognition using local binary patterns”, IEEE International Conference on Image Processing. ICIP, IEEE, pp. II–370.

[19] **Caifeng Shan, Shaogang Gong, Peter W. McOwan** (2009). “Facial expression recognition based on Local Binary Patterns: A comprehensive study”. Image and Vision Computing 27 pp. 803–816

[20] **Sadeghi, Raie, Mohammadi** (2014). “Facial Expression Recognition Using Texture Description of Displacement Image”. Journal of Information Systems and Telecommunication, Vol. 2, No. 4

[21] **D. Sanchez-Mendoza, D. Masip, A. Lapedriza** (2015). “Emotion recognition form mid-level features”. Pattern Recognition Letters 67, pp. 66-74.