UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de
Ingeniería y Sistemas de Telecomunicación

PROYECTO FIN DE GRADO

HERRAMIENTA DE RECONOCIMIENTO FACIAL DE EMOCIONES EN ANDROID

DIEGO ZAPATERO OLMEDILLO

Grado en Ingeniería de Sonido e Imagen

Junio 2016

JUNIO 2016

HERRAMIENTA DE RECONOCIMIENTO FACIAL DE EMOCIONES EN ANDROID

DIEGO ZAPATERO OLMEDILLO

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Herramienta de reconocimiento facial de emociones en Android

AUTOR: Diego Zapatero Olmedillo

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTOR (o Director en su caso): Martina Eckert

DEPARTAMENTO: Teoría de la señal y comunicaciones

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Lourdes López

VOCAL: Martina Eckert

SECRETARIO: José Manuel Díaz López

Fecha de lectura: 30 de Junio de 2016

Calificación:

El Secretario,

AGRADECIMIENTOS

Con este proyecto se acaba una etapa de mi vida cargada de experiencias que me han permitido crecer tanto personalmente como formativamente.

Empecé este trabajo sabiendo que supondría un gran reto. Desconocía por completo la plataforma Android pero quería aplicar todos los conocimientos adquiridos durante la carrera en desarrollar un proyecto que implicara el aprendizaje de algo nuevo y que me llamara la atención.

Quiero mostrar mi agradecimiento a todas las personas que han formado parte de esta etapa, ya que todas y cada una de ellas me han ayudado de alguna manera a llegar al final. En especial gracias a aquellos que me han apoyado en los momentos difíciles mostrándome que en la vida al final lo importante es DISFRUTAR.

En estas líneas debo mencionar a todos los amigos que han logrado que esta etapa sea inolvidable, pues sin ellos esto tendría menos sentido. Gracias a María y Gonxo por los momentos de locura que hacen olvidarnos de todo. Gracias a aquellos que han estado desde el primer día como Alex M. (nuestro Erasmus siempre estará en el recuerdo), Almudena y Sara. Gracias a Sunny, Alex L., Alicia por los momentos vividos con vosotros. Y gracias a aquellos que no han sido nombrados pero que también han sido importantes. Espero que todos vosotros forméis parte de mi vida futura.

Quiero agradecerlo también a las personas que han colaborado activamente en este proyecto. Gracias a Almudena Gil por su ayuda y apoyo mutuo. Gracias a mi tutora Martina por el apoyo y esfuerzo mostrado durante la realización de este trabajo, sobre todo en los momentos complicados, y por otorgarme la posibilidad de investigar este tema.

Por último, GRACIAS a los amigos que siempre han estado cuando se les necesitaba y sobre todo a mis padres por su apoyo incondicional y haberme dado la oportunidad de formarme.

Confía en ti mismo

y dominarás el mundo.

Resumen

Las expresiones faciales no solo representan de manera visual el estado emocional de una persona, sino que constituyen también un papel muy importante a la hora de facilitar la comunicación e interacción entre seres humanos. En las últimas décadas la tecnología ha avanzado de tal manera que se han desarrollado sistemas capaces de reconocer estas emociones de forma automática, y los dispositivos móviles se están convirtiendo en los últimos años en una plataforma en auge para ello.

En este proyecto se desarrolla de una aplicación Android de reconocimiento facial de emociones en tiempo real tomando como base una herramienta implementada en Matlab. Después de emplear técnicas de detección de rostros se localizan las principales regiones de la cara y se extraen los puntos de interés de cada una. A partir de estos puntos se detectan los movimientos musculares de la cara (unidades de acción) y en función de ellos un clasificador reconoce la emoción. Dicho clasificador es entrenado previamente a partir de las combinaciones más frecuentes de unidades de acción en cada una de las emociones básicas (ira, asco, miedo, felicidad, tristeza y sorpresa).

Por último, se presentan pruebas y resultados concluyentes que también van a ser publicadas en el congreso ICCE (*International Conference on Consumer Electronics*) 2016 en Berlín. Además, a partir de ellos se determinan las mejoras necesarias y las líneas de investigación a seguir.

Palabras clave: reconocimiento facial de emociones, Android, dispositivo móvil, unidades de acción, árbol de decisión

Abstract

Facial expressions not only represent the emotional state of a person, they also take an important role in order to facilitate the communication and interaction between humans. During the last decades, technology has made such a huge progresses that automatic emotion recognition systems could be developed, and lately mobile phones are rising as a platform to create these systems.

In this project, an application of facial emotion recognition in real time has been worked out on an Android platform based on a previous work developed in Matlab. It applies a well-known face detection techniques, localizes the main regions in the face and finds the landmarks in each of them. The landmarks are employed to detect facial action units which are fitted into a classifier to recognize the emotion. This classifier is trained before with help of the most relevant combinations of the action units needed to detect the six basic emotions: anger, disgust, fear, happy, sadness and surprise.

Finally, different tests have been performed to obtain the final results which will be published at the ICCE (*International Conference on Consumer Electronics*). According to them, improvements have been proposed and future lines of work have been determined.

Keywords: facial emotion recognition, facial landmarks, Android, action units, decision tree

Índice de contenidos

Índice de figuras

Índice de tablas

Lista de acrónimos

AAM:	Active Appearance Model (Modelo de apariencia activa)
ADABOOST:	Adaptive Boosting
ANN:	Artificial Neural Network (Redes neuronales artificiales)
API:	Application Programming Interface (Interfaz de programación de aplicaciones)
AU:	Action Unit (Unidad de Acción)
CAU:	Combined Action Unit (Unidad de Acción Combinada)
CITSEM:	Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad
CK+:	Extended Cohn-Kanade (Cohn-Kanade extendida)
EBGM:	Elastic Bunch Graph Matching (Combinación de conjuntos de patrones elásticos)
FACS:	Facial Action Coding System (Sistema de Codificación de Acción Facial)
HMM:	Hidden Markov Model (*M*odelo Oculto de Markov)
IDE:	Integrated Development Environment (Entorno de desarrollo integrado)
KNN:	K nearest neighbors (K vecinos más cercanos)
LBP:	Local Binary Patterns (Patrones binarios locales)
LDA:	Linear Discriminant Analysis (Análisis discriminante lineal)
PCA:	Principal Component Analysis (Análisis de componentes principales)
RF:	Requisitos funcionales
RNF:	Requisitos no funcionales
SDM:	Supervised Descent Method (Método de Descenso Supervisado).
STM:	Selective Transfer Machine (Máquina de Transferencia Selectiva)
SVM:	Support Vector Machine (Máquina de soporte vectorial)

1. Introducción

Uno de los campos más estudiados a lo largo de la historia y que cada día cobra más importancia es la comunicación no verbal entre los seres humanos, y actualmente no sólo entre ellos sino también entre personas y las máquinas que les rodean con el objetivo de que en un futuro una máquina sea capaz de actuar en función de lo que expresemos.

Existe un elemento importante a tener cuenta y es el hecho de que nadie puede ver lo que sentimos, sino lo que expresamos, y para ello es indispensable el papel que juega el rostro. Las expresiones faciales son el medio más importante para expresar estas emociones y estados de ánimo. A través de ellas, se puede obtener una mejor compresión de lo que nos intentan comunicar los demás, reforzando o engañando la comunicación. A día de hoy, gracias al desarrollo en estos últimos años del procesado de la imagen, se realizan numerosas investigaciones para hacer posible el reconocimiento de una emoción de manera automática.

El reconocimiento facial de emociones es hoy en día uno de los campos más estudiados debido a la gran cantidad de aplicaciones que puede tener en campos como la medicina (rehabilitación), la psicología, la seguridad vial, robótica, o videojuegos,

1.1 Objetivos

El objetivo principal de este proyecto es desarrollar una herramienta para el reconocimiento de expresiones faciales en el entorno. Dicho proyecto se basa en una herramienta creada en el software Matlab iniciada por un grupo de estudiantes en el Centro de Investigación en Tecnologías de Software y Sistemas Multimedia para la sostenibilidad (CITSEM) y en la cual he participado como becario mediante una Beca de Colaboración concedida por el Ministerio de Educación, Cultura y Deporte durante el curso 2014/15. Dicha herramienta se ha continuado elaborando en dos sucesivos proyectos fin de grado, de los cuales uno ha ido en paralelo a este proyecto. Por lo tanto, tanto en la herramienta de Matlab como en la aplicación Android, está desarrollado el mismo proceso para reconocer la emoción de una persona.

La aplicación, debe de ser una herramienta muy cómoda y útil para los usuarios. Además deberá contar con una interfaz intuitiva de tal manera que pueda ser entendida y utilizada rápidamente por cualquier persona sin dificultad.

En este proyecto se persigue el objetivo de reconocer seis emociones básicas como son: ira, asco, miedo, felicidad, tristeza y sorpresa.

Como primer paso se llevará a cabo un estudio del estado del arte del reconocimiento facial de emociones para conocer de manera general los métodos existentes y compararlos con los utilizados en la herramienta creada en Matlab. Posteriormente, se estudiarán en detalle todos los desarrollos existentes en el ámbito del reconocimiento en la plataforma Android.

Tras esta búsqueda de información se implementará una estructura básica para la aplicación con el objetivo de integrar después todo el proceso de tratamiento de la imagen en ella. El procedimiento constará de varias partes:

- Detección de la imagen (rostro) a analizar
- Extracción de características de la imagen con el fin de poder localizar, analizar y procesar los puntos de interés de la cara.
- Modelo de aprendizaje: se aplicará un algoritmo que sea capaz de realizar un aprendizaje automático con el fin de crear una serie de pautas que ayuden a reconocer la emoción.
- Clasificación de la emoción: la aplicación mostrará qué emoción se ha detectado.

Una vez logrados estos objetivos se llevará a cabo la implementación total y completa de la aplicación de tal manera que sea capaz de reconocer una emoción en tiempo real.

En este proyecto se aplicarán todos los conocimientos adquiridos durante la carrera en el campo del tratamiento digital de la imagen para la parte del reconocimiento facial y de la programación para la parte del desarrollo de la aplicación.

1.2 Estructura de la memoria

En el primer capítulo de este documento se lleva a cabo un análisis del estado del arte en el que se explican los antecedentes en el reconocimiento facial de emociones desde los primeros sistemas creados hasta herramientas diseñadas para dispositivos móviles. Posteriormente se describe la herramienta de Matlab que sirve como base para este proyecto explicando sus características y funcionalidades. En el siguiente capítulo se expone el desarrollo de la aplicación, dividiéndolo en dos partes. En la primera parte se explica el diseño tanto funcional como técnico, las librerías utilizadas y los requisitos mínimos para el correcto funcionamiento de la aplicación. En la segunda parte se detalla el proceso ejecutado para la implementación desde los primeros pasos realizados hasta la versión final. Después se muestran las pruebas y resultados obtenidos en la herramienta final desarrollada. Seguidamente se introducen las conclusiones extraídas a partir de los resultados, y a continuación se explican las posibles mejoras y trabajos futuros que pueden efectuarse a partir de este proyecto. Antes de terminar se encuentra la lista de referencias bibliográficas que se han utilizado a lo largo de este trabajo. Por último se incluye un anexo sobre la configuración del entorno para poder desarrollar esta aplicación.

2.Estado del arte

2.1 Reconocimiento facial de emociones

Las expresiones faciales son una de las maneras más importantes de representar el estado emocional y mental. Una teoría que lo reafirma es la que propuso en 1968 el psicólogo Albert Mehrabian sobre la comunicación no verbal [1]. Según sus experimentos y estudios solamente el 7% del contenido mensaje que se trasmite se realiza a través de las palabras. Un 38% estaría determinado por el tono de la voz y el resto, un 55%, se transfiere a través de las expresiones faciales y corporales.

En cualquier caso, si se retrocede aproximadamente un siglo, ya en el año 1872 Charles Darwin [2] explicaba la importancia de las expresiones y cómo estas afectan de la misma manera a todos los individuos sin importar el género o la raza. Además clasificó los tipos de expresiones parecidas en distintas categorías como sorpresa y asombro o pena y tristeza. De nuevo en el siglo pasado, en la década de 1970, tiene lugar uno de los hitos más importantes en el estudio de las expresiones faciales. Ekman y Friesen [3] definieron seis categorías básicas de expresiones: *Anger* (Ira o enfado), *Disgust* (Asco), *Happy* (Feliz), *Fear* (Miedo), *Sadness* (Tristeza), and *Surprise* (Sorpresa) que proporcionan la base para todos los estudios que se desarrollan hoy en día en el reconocimiento automático de expresiones faciales.

Para continuar con la teoría de Darwin, en 1971 estos dos psicólogos realizaron un estudio sobre personas procedentes de distintas culturas y llegaron a la conclusión de que las expresiones faciales eran prácticamente idénticas en todas ellas, y a pesar de que algunos investigadores como Russell [4] dudaban de ello, se mostraron fuertes evidencias [5] [6] de que era cierto y actualmente es un hecho establecido que las expresiones faciales son universales.

Aparte de estas seis emociones básicas el ser humano es capaz de reconocer otras. En el año 2000 Parrott identificó 136 estados emocionales [7] y actualmente hay numerosos intentos de reconocer emociones además de las definidas por Ekman y Friesen como bien recoge Vinay Bettadapura [8] en el interesante estudio del arte que realizó en 2012, desde expresiones en la cual la persona está posando (*posed expressions*) hasta las microexpresiones (expresiones momentáneas e involuntarias) pasando por las expresiones espontáneas (*spontaneous expressions*), que son expresiones del día al día. Como se puede visualizar en la Figura 1 las expresiones donde el sujeto está posando son más fáciles de detectar y reconocer mientras que las microexpresiones son muy complicadas de capturar al ser instantáneas y por tanto muy difíciles de detectar para hacer pruebas tan siquiera. El objetivo real es detectar las expresiones espontáneas ya que las *posed expressions* pueden parecer artificiales, exageradas en algunas ocasiones y por tanto no sirven para un escenario real. Debido a este hecho, en los últimos años las investigaciones se están centrando en el reconocimiento de estas expresiones espontáneas, pero no es un proceso tampoco sencillo. Requiere rapidez a la hora detectar la expresión y después una gran precisión para analizar cuál es realmente la emoción que se está expresando.

(a)[9] (b) (c)

Figura 1 Expresión de enfado.
(a) Expresión con pose [9] (b) Expresión espontánea [10] (c) Microexpresión[11]

Hasta la actualidad la mayoría de estudios están realizados sobre expresiones con debido a lo comentado anteriormente y, aunque cada vez hay sistemas que reconocen con mayor ratio de eficiencia las emociones, a veces se produce confusión incluso entre algunas de las seis emociones prototipo. Sebe et al. [12] por ejemplo dictaminó que existe la posibilidad de que se confundan las expresiones de ira con asco o la de miedo con sorpresa. En la Figura 2 se puede comprobar como estas posibles confusiones son más probables de lo que en un principio puede parecer. Vinay [8] recoge con más detalle más casos como estos.

Figura 2. Las 6 emociones prototipo descritas por Ekman y Friesen [3]

Hasta finales del siglo XX los estudios eran realizados por psicólogos o gente relacionada con esta especialidad, pero gracias al avance en campos como la robótica, gráficos, animaciones o visión por ordenador otros sectores se empezaron a interesar en el estudio de las expresiones faciales. No fue hasta principios de la década de 1990 cuando se empezaron a desarrollar los primeros sistemas automáticos de reconocimiento facial de expresiones [13] y de hecho en ellos se comenta que no fue posible conseguir los objetivos deseados debido seguramente a los grandes requerimientos de los sistemas para la detección y seguimiento de caras que por aquel entonces carecían de robustez. Sin embargo, gracias a todos los avances comentados anteriormente a partir de esos años se comenzó a estudiar de manera más activa formas de implementar estos sistemas automáticos. En [14] Pantic y Rothkrantz hacen un repaso de todos los sistemas creados desde principios de esa década hasta el 2001.

Estos sistemas automáticos tienen cada vez más importancia en la actualidad puesto que afectan a un gran número de áreas. Se pueden usar en campos donde se estudie todo lo relacionado con el comportamiento humano como la psiquiatría o en otros muy distintos como la robótica o la interacción entre los seres humanos y las máquinas, sin olvidarse de los videojuegos, las animaciones, la realidad virtual, la seguridad en un automóvil o el software educativo.

Los sistemas automáticos de reconocimiento facial de emociones tienen como misión detectar la emoción de una persona en función de sus expresiones faciales y en general, suelen estar divididos en varias etapas representadas en el esquema de la Figura 3. Figura 1Una primera etapa donde se detecta la cara, seguida de otra en la cual se extraen todas las características más significativas y por último una etapa donde un clasificador, que previamente ha sido entrenado, detecta la emoción en función de las particularidades de cada cara.

Este entrenamiento del clasificador constituye precisamente la primera fase de cualquier sistema de reconocimiento. Esta fase, consiste en entrenar el clasificador de tal manera, que a partir de un conjunto amplio de imágenes de las cuales se conoce su emoción, vaya aprendiendo qué características pertenecen a cada una de las emociones. Para ello toma cada imagen y realiza unos pasos parecidos a los descritos anteriormente: detecta la cara de cada imagen, extrae sus características y asocia esas características a la emoción conocida que está expresando esa cara. Una vez que el clasificador ha sido entrenado el sistema ya está preparado para reconocer las emociones.

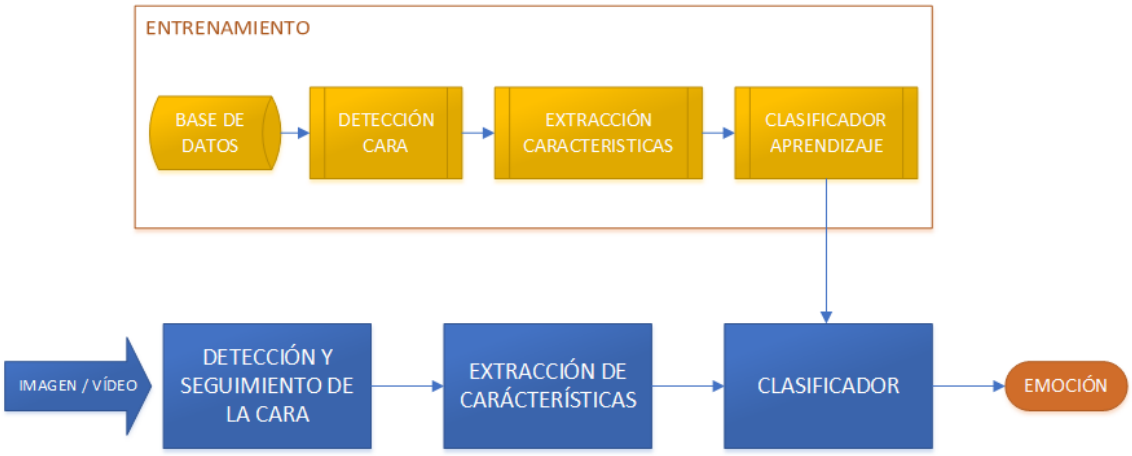


Figura 3 Diagrama de bloques reconocimiento facial de emociones

DETECCIÓN Y SEGUIMIENTO DE LA CARA

Esta etapa tiene como fin separar la imagen en dos partes: la cara y el fondo. Los primeros métodos en la detección y seguimiento de la cara se empezaron a desarrollar a partir de principios de la década de 1990. Hjemal and Low [15] reunieron los métodos de detección facial que existían hasta 2001, dividiéndolos en dos grandes grupos: métodos basados en características y métodos basados en imágenes. A pesar de que investigadores como A. Sharifara hayan propuesto otra clasificación [16] , hoy en día se sigue utilizando la de principios de siglo XXI .H. Hatem [17] la detalla en mayor profundidad, dividendo además estos grupos en subgrupos, derivándose en el esquema mostrado en el esquema de la Figura 4.

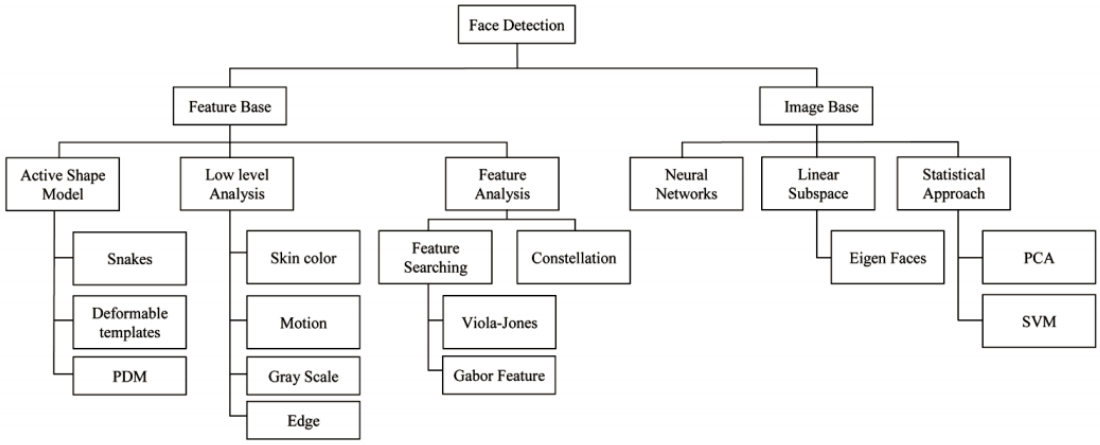
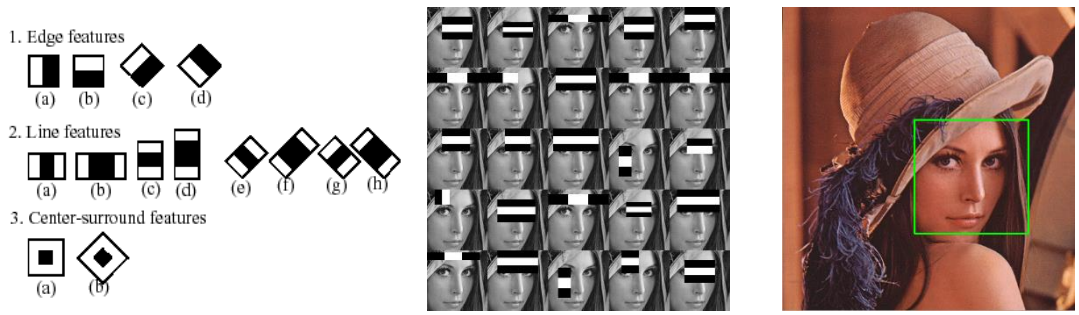


Figura 4 Métodos detección facial [17]

Sin lugar a dudas, el método que ha tenido más éxito desde su creación ha sido el desarrollado por Viola y Jones en 2001 [18] y mejorado en 2004 [19] Este método, englobado dentro de los métodos basados en características y en el subgrupo de análisis de características, ha sido frecuentemente utilizado gracias a su rapidez y a los grandes resultados que proporciona en caras de frente. Se basa en el algoritmo de aprendizaje *AdaBoost* y en la combinación de distintos clasificadores ordenados de menor a mayor complejidad, uno detrás de otro en forma de cascada, que conjuntamente permiten buscar las zonas con mayor cantidad de información útil y descartar las regiones que no son de interés.

Lo primero de todo, como ocurre con todos los clasificadores, han de ser entrenados con un gran número de muestras del objeto que se quiere detectar. Los clasificadores se basan en características llamadas "Haar-like features" como las que se pueden en la Figura 5(a). Los clasificadores van aplicándolas a la imagen para reconocer los bordes, las líneas etc. en un proceso que se vería como en la Figura 5(b) hasta que se consigue localizar la cara Figura 5(c). Dicho proceso está también explicado y reproducido por Adam Harvey [20] en una entrevista que concedió en 2012.



(a) (b) (c)

Figura 5. Algoritmo Viola-Jones

Otro método que últimamente está teniendo más presencia es el LBP (*Local Binary Patterns* o Patrones Binarios Locales) introducido en 2002 por T. Ojala [21] y utilizado en general para detectar todo tipo de objetos basado en texturas, invariante ante cambios de iluminación y capaz de detectar movimiento [22] [23]

Se trata de un sistema capaz de extraer la estructura local en una imagen a partir del entorno de cada uno de los píxeles que la componen. El proceso consiste en ir comparando el valor de cada pixel (considerado como píxel central) con el de sus vecinos distribuidos en una circunferencia a su alrededor. La comparación se hace en base a la Ecuación (1) siendo x_c el valor central y x el valor a evaluar.

$$s(x) = \begin{cases} 1 & if\ (x \geq x_c) \\ 0 & else \end{cases} \quad (1)$$

De tal manera que, si el valor del pixel vecino es mayor que el central entonces el valor será '1', mientras que si es menos será 'o' como se puede ver en el ejemplo de la Figura 6. De este modo, se asocia un valor binario a cada píxel formando un número binario.

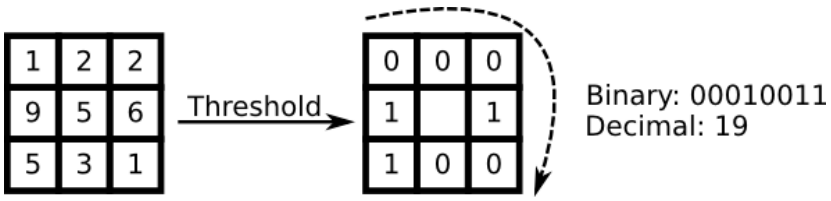


Figura 6 Funcionamiento LBP [24]

Si hay 8 píxeles vecinos se obtendrá un número de 8 dígitos obteniendo así combinaciones posibles, llamadas LBP, y obtenidas matemáticamente a partir de la expresión de la Ecuación (2), donde $i_c$hace referencia a la intensidad del punto central y i_p a la intensidad de cada vecino.

$$LBP\ (x_c, y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \quad (2)$$

Varsha G. y Dipesh S. [25] comparan estos dos métodos comentados hasta ahora para detectar caras explicando las ventajas e inconvenientes de cada uno. El algoritmo de Viola-Jones es prácticamente inigualable en cuanto a precisión y sobre todo al ratio de precisión/velocidad. En cambio, aunque el algoritmo de LBP es menos preciso, es más rápido e invariante frente a cambios de iluminación.

Por último, un método muy diferente a estos dos anteriores utilizado sobre todo para el seguimiento de la cara pero que también está teniendo gran repercusión es el método denominado "Candide" [26] basado en modelos parametrizados. Este método se compone de una máscara o malla que contiene las facciones de un rostro humano divididas en triángulos como se puede apreciar en la Figura 7. La malla se superpone a la cara y se adapta en función del movimiento de las facciones.

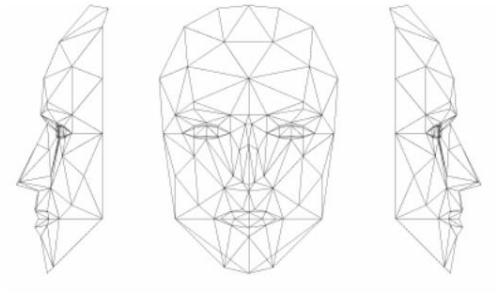


Figura 7. Máscara Candide-3 con 113 vértices y 168 superficies [26]

EXTRACCIÓN DE CARACTERÍSTICAS

Existen muchos mecanismos para la extracción de los componentes principales de la cara. El objetivo es identificar partes de la cara como ojos, boca, nariz, etc. Al tratarse de reconocimiento de partes concretas y comunes en todas las caras se pueden utilizar también los métodos comentados anteriormente para la detección de la cara (y viceversa) pero además de ellos existen otros, y se pueden clasificar por tipos según el esquema de la Figura 8.

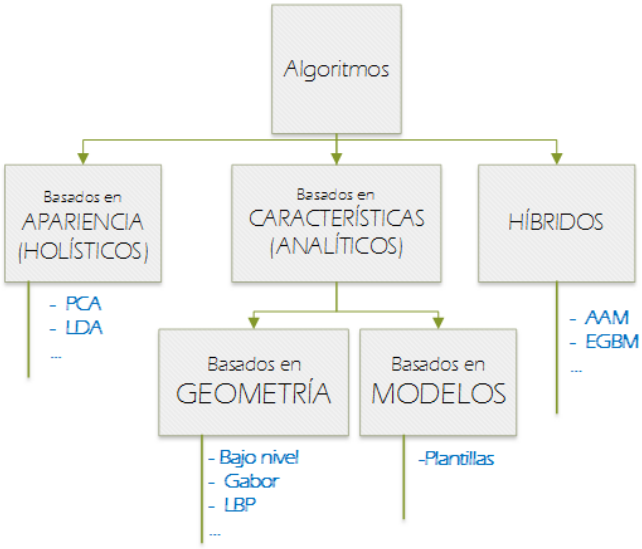


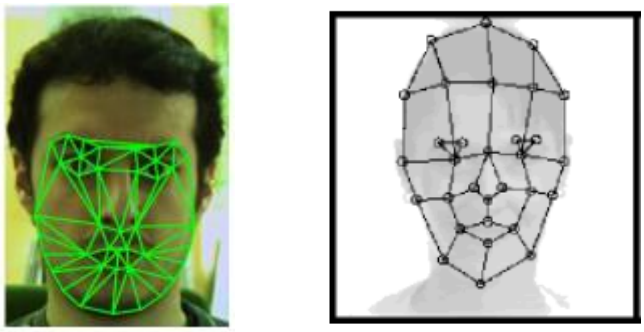
Figura 8 Clasificación métodos extracción de características

Por una parte, en los métodos de extracción basados en la apariencia global se examinan los cambios generales producidos en la cara, obteniendo las características filtrando la imagen o una región de ella como las técnicas basadas en PCA (*Principal Component Analysis* o Analisis de componentes principales) o LDA (*Linear Discriminant Analysis* o Analisis de discriminantes linales) [27].

Por otra parte están los métodos basados en características, que se pueden dividir en técnicas basadas en geometría como LBP (aunque ahora aplicado a regiones y no a la cara completa) o de bajo nivel (utilizando análisis de bordes y segmentaciones) y basados en modelos de características como el descrito en plantillas deformables [28] .

Por último, existen los métodos híbridos y que por tanto se basan en características pero también en apariencia. Entre ellos estaría el AAM (*Active Appearance Model* o modelo de apariencia activa), introducido por varios investigadores en [29]. Está basado en la adaptación de un modelo estadístico creado previamente durante una fase de entrenamiento a partir de una serie de puntos de referencia. Para cada imagen, crea una malla de puntos característicos y adopta la forma y apariencia del objeto de la región de interés. Después, para todas se realiza un análisis PCA para obtener las variaciones y poder obtener la malla modelo que luego se aplicará para la extracción como la que se puede ver en la Figura 9.

Otro método híbrido es EBGM (*Elastic Bunch Graph Matching* o Combinación de conjuntos de patrones elásticos), basado en la teoría de grafos [30] [30] . Como en AAM, se crea un modelo estadístico en una fase previa para comparar posteriormente la diferencia entre el grafo detectado y el grado modelo como se muestra en la Figura 9. Para crear ese grafo, utiliza filtros Gabor [31] que permiten obtener información en frecuencia de las regiones de la imagen.



(a)[32] *(b) [30]*

Figura 9 Métodos hibrídos. AAM(a) EBGM(b)

FACIAL ACTION CODING SYSTEM (FACS)

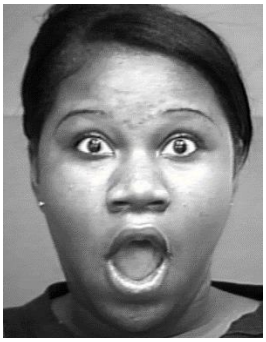
Un método complementario a los anteriormente descritos es el sistema de codificación de movimiento facial desarrollado por Ekman y Friesen en 1978 [33]. Este método se basa en asociar los cambios producidos en las diferentes partes de la cara a la hora de expresar una emoción, con las acciones de los músculos que la generan. Dichas acciones son denominadas AUs (*Action Units* o Unidades de acción).

Se definieron 46 AUs aunque no todas son igual de representativas puesto que unas aportan más información relevante que otras. En la

Tabla 1 se encuentra un ejemplo de algunas AUs. La primera columna se corresponde con el número correspondiente a la AU, la segunda con su representación y la tercera con una breve descripción de ella. En la cuarta columna se puede observar una imagen donde se combinan esas unidades de acción.

En apartados posteriores se hablará en mayor profundidad sobre este método aunque también se puede encontrar información más detallada en el proyecto realizado por S. Gonzalez [34].

Tabla 1 Ejemplos AUs y posible combinación

AU	Imagen	Definición	Combinación
AU 1		Interior de las cejas elevado	
AU 2		Exterior de las cejas elevado	
AU 5		Párpado superior elevado	
AU 27		Boca abierta	

CLASIFICADORES

Una vez que las principales características han sido extraídas, en la última etapa un clasificador se encarga de reconocer la emoción. Como ocurre con las etapas anteriores no existe un proceso único para lograr este objetivo aunque todos tienen ciertos aspectos en común. Como se ha mencionado anteriormente en la introducción (pág. 3), es necesario un aprendizaje previo en el cual el clasificador sea entrenado para poder reconocer la emoción correctamente en el escenario real.

A lo largo de los años distintos investigadores han recopilado bases de datos de imágenes para que los clasificadores pudieran ser entrenados. En general todas las imágenes de las bases de datos con las que se trabaja hoy en día [8][34]son con expresiones posadas, es decir, se ha pedido al sujeto que exprese una emoción determinada. Además, suelen presentar distintas condiciones de iluminación, oclusiones y vistas laterales.

La base de datos mayormente utilizada es la Cohn-Kanade, creada en el año 2000 y mejorada años después denominándose CK+ [35], al ser muy variadas y estar compuesta por hombres y mujeres de distintas nacionalidades y razas. Para cada sujeto hay una imagen con una expresión neutral (sin emoción) y seis imágenes con las emociones básicas comentadas al principio de este capítulo. Este conjunto de imágenes presenta distintas condiciones de iluminación pero sin aparentes oclusiones o diferencias en la pose. Cuenta además con imágenes en blanco y negro y en color. Algunos ejemplos de esta base de datos se pueden observar en la Figura 10.

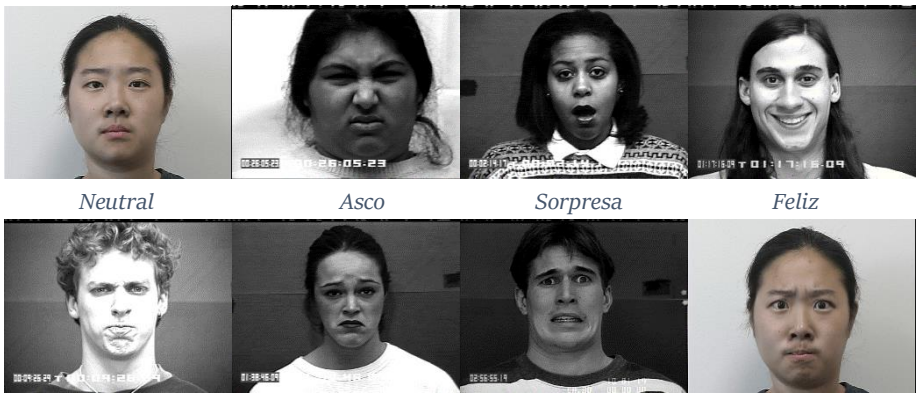


Figura 10 Ejemplos imágenes CK+

Uno de las clasificaciones más frecuentemente usados son las **ANN** (*Artificial neural Networks* o redes neuronales artificiales), un método de aprendizaje basado en el sistema de neuronas del cerebro humano. Una red neuronal se compone de un conjunto de neuronas distribuidas normalmente en tres capas: una capa de entradas, una o varias capas ocultas y una capa de salida. Cada neurona está compuesta por una o varias entradas y emite una o varias salidas. En el ejemplo de la Figura 11 se puede observar una red neuronal con tres entradas, dos salidas y una capa oculta con cuatro neuronas.

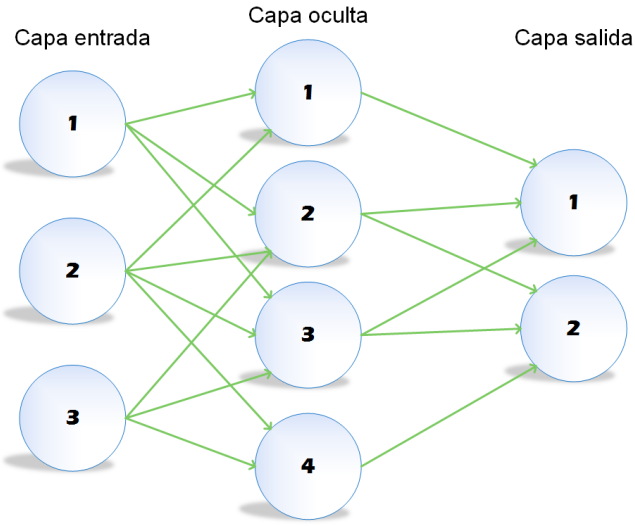


Figura 11 Sistema red neuronal

Cada salida viene dada por tres funciones representadas en el esquema de la Figura 12. Una función de propagación que consiste en el sumatorio de cada entrada multiplicada por el peso de su interconexión, una función opcional de activación que modifica la anterior y una función final de transferencia que se aplica al valor que devuelve la función de la activación en caso de que exista. Su misión es delimitar la salida de la neurona y una de las funciones más usadas como función de transferencia es la función sigmoidea para acotar el valor de salida entre ‘o’ y ‘1’.

En el caso de los sistemas de reconocimiento facial de emociones se puede utilizar como parámetro de entrada el nivel de gris, puntos de referencia o unidades de acción. Se suelen usar además seis nodos de salida, uno para cada emoción y la función sigmoidea para acotar valores.

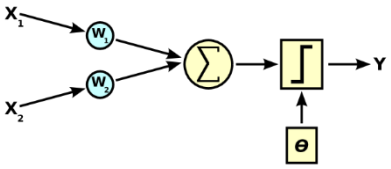


Figura 12 Cálculo de salida a partir de varias entradas. ANN

Las principales ventajas de este sistema son la gran capacidad de auto aprendizaje, una destacable precisión [36] y la rapidez para encontrar una salida, siendo por ello frecuentemente utilizados en aplicaciones en tiempo real [37].

Asimismo, es habitual encontrarse estas redes neuronales combinadas con **árboles de decisión** [38]. Un árbol de decisión es un modelo de predicción basado en la construcción de diagramas lógicos, de manera similar a los sistemas basados en reglas. Al igual que las redes neuronales, se construye a partir de una serie de entradas (también llamadas atributos en este método) y a partir de ellas y de un algoritmo de decisión devuelve una salida. El valor de estas entradas y salidas puede ser discreto o continuo. Si son valores discretos se denomina clasificación (el más común y el utilizado normalmente para predecir emociones) mientras que sin son continuos se llama regresión.

Se denomina árbol de decisión porque en la fase de entrenamiento las decisiones se estructuran en forma de árbol tal y como se puede ver en la Figura 13. Similar a un árbol real, se compone de una raíz, de unas ramas y de unas hojas.

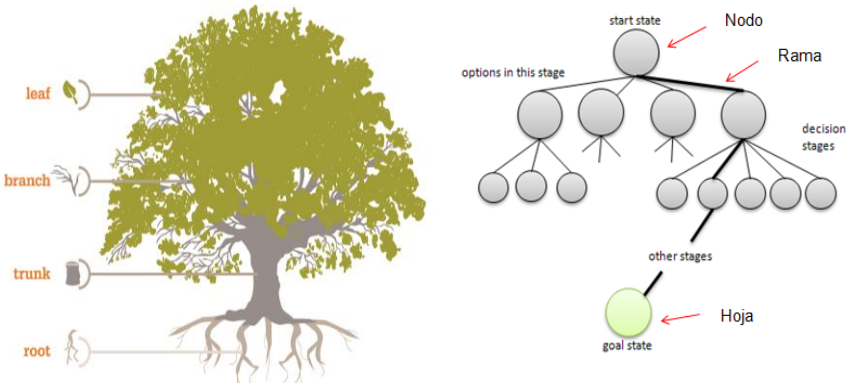


Figura 13 Estructura árbol de decisión

La raíz del árbol es el nodo principal, es decir, la entrada más importante. De él parten las ramas principales que llevan a los nodos secundarios. De esos nodos secundarios partirán más ramas y así sucesivamente hasta llegar a las hojas, que serían las salidas. Los parámetros de entrada pueden ser los mismos que los comentados en las redes neuronales.

Para crear esta estructura de árbol, este método suele usar el algoritmo ID3 [39]. Este algoritmo crea un árbol de manera descendente. La idea general de este algoritmo es identificar cuál es la entrada o atributo más importante y con mayor influencia, es decir, aquel que posea el mayor poder discriminatorio para un conjunto de atributos e ir creando subconjuntos a partir de dicho

atributo. Cuanta más importancia tenga un atributo, más cerca de la raíz del árbol se encontrará. Para calcular la influencia de los atributos, se analiza cada atributo estadísticamente para saber cuánta información aporta.

Un concepto muy importante para ese análisis estadístico es la Entropía (E), que determina la cantidad de incertidumbre de un determinado conjunto de ejemplos y se puede definir de forma general según la Ecuación (3)

$$E\ (S) = -\sum_{i=1}^{N} p_i \log_2 p_i \qquad (3)$$

siendo 'S' un conjunto de ejemplos, 'N' el número de valores posibles que puede tomar el atributo y p_i la probabilidad de los posibles valores.

A mayor entropía, habrá mayor incertidumbre y por tanto más difícil le será al árbol tomar una decisión. En general, un atributo que puede ayudar a discriminar más ejemplos, tiende a reducir la entropía, y por tal motivo, debe ser seleccionado como un nodo de selección para la siguiente subdivisión.

Sin embargo, en general se suele tener en cuenta otro parámetro más preciso para seleccionar el mejor atributo: la ganancia de información, definida en la Ecuación (4), que evalúa la calidad con la que un atributo discrimina los ejemplos. A mayor ganancia de información, más importante será dicho atributo. El algoritmo utiliza esta ganancia calculada para ir eligiendo los mejores atributos en cada paso, crear subconjuntos e ir construyendo el árbol descendentemente.

$$Ganancia\ (atributo) = E\ (S) - Resto\ (atributo) \qquad (4)$$

El resto se puede calcular a partir de la Ecuación (5). Calcula la entropía de cada rama, es decir, de cada subconjunto y realiza una suma proporcional para calcular la entropía del total.

$$Resto\ (atributo) = \sum_{v \in V(atri)} \frac{|S_v|}{|S|} E\ (S_v) \qquad (5)$$

En la cual 'S' es el conjunto de ejemplos y S_v es el subconjunto de 'S' que tiene valor 'v' para el atributo analizado.

Resumiendo, se podría decir que para crear el árbol de decisión este algoritmo realiza los siguientes pasos:

1. A partir de todo el conjunto de ejemplos, se selecciona el mejor atributo entre todos, que corresponde al nodo principal

2. Ese atributo principal divide el conjunto de ejemplos en tantos subconjuntos de ejemplos como valores pueda tomar.

3. A partir de cada subconjunto se vuelve a repetir el mismo proceso (pasos 1 y 2), pero ahora con menos ejemplos y con un atributo menos (el elegido en el paso 1), por lo que cada atributo que se selecciona se descarta para la siguiente prueba.

4. Este proceso se repite sucesivamente hasta que:

    - Todos los ejemplos de un subconjunto tienen el mismo valor y por tanto la salida de esta rama sería ese valor. Este valor sería una de las salidas u hoja del árbol.

    - Se hayan analizado todos los atributos. En este caso el valor de salida estaría determinado por el mayor número de valores iguales dentro de ese subconjunto.

Una vez finalizada la creación del árbol, cuando se tenga que analizar una nueva muestra, el árbol de decisión analizará sus entradas o atributos según el orden que haya creado en la fase de entrenamiento. En función de los valores de sus entradas de la nueva muestra, seguirá un recorrido u otro a lo largo del árbol hasta llegar a una salida concreta.

Este método es destaca por su sencillez a la hora de implementarlo y ofrecer resultados óptimos [40]. También se puede encontrar más en información en [39].

Otro de los métodos clasificadores más utilizados frecuentemente para sistemas de reconocimiento facial de emociones es **SVM** (*Support Vector Machine* o máquinas de vectores de soporte). Se trata de un método de clasificación basado en un entrenamiento supervisado (se conoce cuál es el resultado de cada muestra). A partir de un conjunto de muestras de entrenamiento se pueden etiquetar clases y entrenar un modelo SVM que prediga posteriormente a qué clase pertenece una nueva muestra.

Gráficamente el modelo se puede representar como un conjunto de puntos (muestras) en el espacio, separando los puntos que pertenecen a clases diferentes en conjuntos más pequeños. El modelo SVM construye uno o varios hiperplanos para separar dichas clases. Matemáticamente un hiperplano se puede definir según la Ecuación (6), β es el peso del vector y β_0 la tendencia inicial.

$$f(x) = \beta_0 + \beta^T x \tag{6}$$

En el ejemplo de la Figura 14 se representan dos clases bien distinguidas (cuadrados y círculos) y algunos de los hiperplanos que el modelo constituiría.

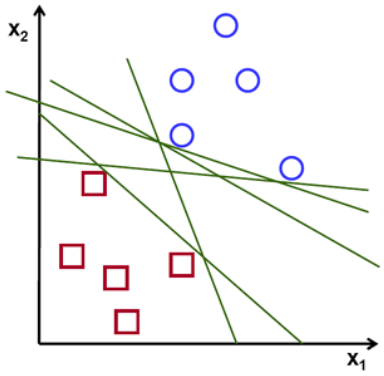


Figura 14 Ejemplo hiperplanos SVM. 2 clases. [41]

Cuando el modelo analiza una muestra nueva, sitúa esa muestra en el espacio y la asocia a una clase u otra en función de la proximidad a ella. Esa proximidad es uno de los puntos clave de este modelo. SVM busca un hiperplano que separe de la forma más óptima posible los puntos de una clase y otra y para ello busca aquel que proporcione la máxima distancia entre las muestras de entrenamiento y el hiperplano en cuestión. Esta distancia máxima se denomina margen. Siguiendo con el ejemplo anterior, el hiperplano ideal sería el representado en la Figura 15. Esto se cumple cuando en la (6, $|f(x)|=1$. Además, al vector que forman los puntos más cercanos al hiperplano se le llama vector de soporte.

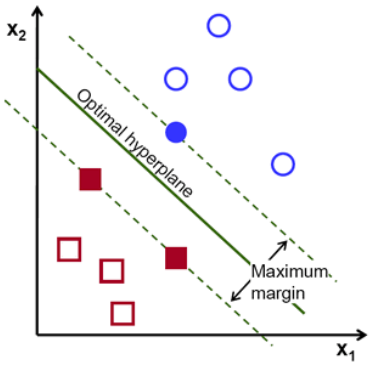


Figura 15 Hiperplano óptimo o canónico [41]

La principal ventaja frente a las redes neuronales es la gran eficiencia y rapidez del entrenamiento. Asimismo, otorga una alta precisión en la clasificación y por ello es tan utilizado en sistemas en tiempo real [42].

Otro método de clasificación supervisada utilizado para el reconcomiendo de emociones es el **KNN** (*K nearest neighbors* o K vecinos más cercanos), basado en la probabilidad de que una muestra pertenezca a una clase u otra en función de la clase a la que pertenezcan las muestras que estén a su alrededor. "K" define el número de vecinos a considerar. A diferencia de SVM en este método las clases no están separadas en conjuntos, si no que se encuentran todas mezcladas.

En la fase de entrenamiento se almacenan todas las muestras y sus clases asociadas. En el ejemplo de la Figura 16 se muestran dos clases principales, una representada por triángulos y otra por cuadrados. En la fase de clasificación se calcula la distancia de la nueva muestra respecto a las muestras almacenadas seleccionando las "k" muestras más cercanas. El nuevo ejemplo es

clasificado con la clase que más se repita entre los "k" vecinos. Normalmente se utiliza la distancia euclidiana.

El valor del parámetro "k" es sumamente importante, ya que si es un valor muy bajo se posee poca información pero si es muy alto pueden incluirse muestras irrelevantes y que generen confusión. Este parámetro por lo tanto debe elegirse en función de los datos de entrada y los resultados de las pruebas realizadas con distintos valores de "k". En el ejemplo comentado, se puede observar que si se elige un K=3, la nueva muestra es clasificada como "triángulo" (hay un cuadrado y dos triángulos) mientras que si K=5 es clasificada como cuadrado (hay tres cuadrados y dos triángulos).

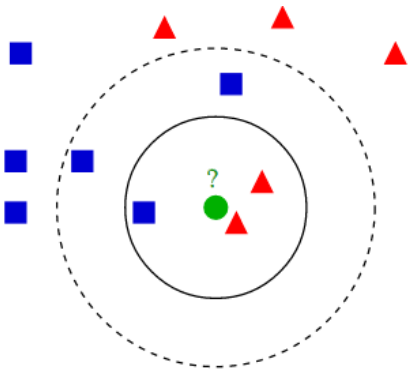


Figura 16 Ejemplo del método KNN [43]

Para corregir el sesgo que se puede producir por muestras irrelevantes se puede ponderar cada muestra con un peso, otorgando mayor peso a las muestras previamente consideradas como más relevantes.

Aunque este modelo no es tan frecuentemente usado como los anteriores también otorga buenos resultados incluso con imágenes con oclusiones [44]

Desde otra perspectiva, existe una aproximación distinta a los modelos anteriores basada en **redes bayesianas**. Una red bayesiana es un modelo de probabilidad que representa un grupo de variables aleatorias y sus dependencias entre sí a través de un grafo [1]. El esquema básico se puede ver representado en la Figura 17.

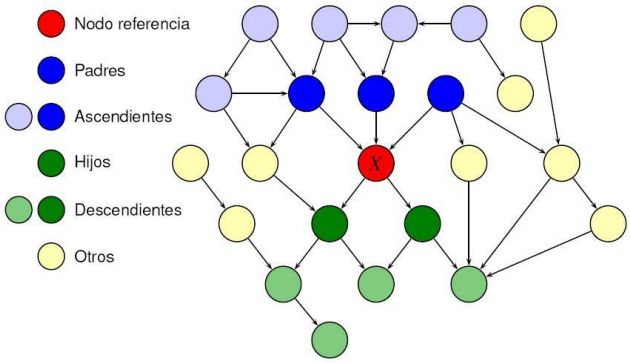


Figura 17 Ejemplo estructura Red Bayesiana [45]

[1] Grafo: representación simbólica de los elementos de un sistema mediante un esquema gráfico.

Cada nodo en una red bayesiana representa una variable y tiene asociado una función de probabilidad que tiene como entrada un conjunto concreto de valores provenientes de otros nodos y devuelve la probabilidad de la variable del nodo evaluado. Este cálculo de la probabilidad está basado en el teorema de Bayes definido en la Ecuación (7). Dado un conjunto de sucesos independientes $\{A_1, A_2, \dots A_i \dots . A_n\ \}$ y tales que la probabilidad de todos ellos sea distinta de '0':

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{7}$$

Siendo B otro suceso del que se conocen las probabilidades de que suceda B en función dé suceso A y donde $P(A_i)$ son las probabilidades a priori, $P(B|A_i)$ es la probabilidad de que suceda B en la hipótesis A_i y $P(A_i|B)$ son las probabilidades a posteriori.

Este método es capaz de realizar un entrenamiento supervisado y no supervisado y obtener buenos resultados a la hora de clasificar más tarde las emociones [46].

Para finalizar este análisis de los métodos de clasificación, uno de los modelos también empleados para el reconocimiento de emociones y muy relacionado con las redes bayesianas son los HMM (*Hidden Markov Model* o modelo oculto de Markov). Un **HMM** puede ser considerado como la red bayesiana más simple. Están basados en la propiedad de Markov (el nombre del modelo es debido a esta propiedad), por la cual la distribución de probabilidad de un proceso del valor futuro de una variable únicamente depende de su valor presente, independientemente de los valores pasados de la misma.

Se trata de un modelo en el cual el estado de cada nodo es desconocido, es decir, está oculto. Este estado oculto se puede determinar a partir de todos los parámetros conocidos del sistema. En este caso, los parámetros son las probabilidades de transición entre un estado y otro. Como ocurre con las redes bayesianas cada estado tiene una distribución de probabilidad sobre las posibles salidas. En la Figura 18 Estructura simple HMM se puede observar un ejemplo de transición de estados en un HMM en el cual "x" serían los estados ocultos, "y" las salidas, "a" las probabilidades de que exista una transición y "b" las probabilidades de salida.

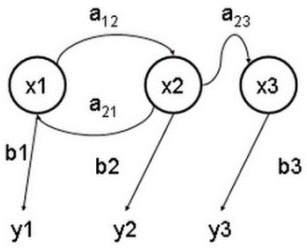


Figura 18 Estructura simple HMM [47]

Se trata de un modelo simple de red bayesiana pero a pesar de ello proporciona también unos resultados óptimos [48]. Su ventaja precisamente frente a las redes bayesianas es su simpleza.

2.2 Reconocimiento facial de emociones en Android

A lo largo de los años los sistemas de reconocimiento facial de emociones se han desarrollado en multitud de plataformas. Actualmente, cada vez más investigadores están invirtiendo recursos en desarrollar estos sistemas para dispositivos móviles. Hoy en día prácticamente todo el mundo dispone de uno capaz de ejecutar las tareas que requieren estos sistemas y no solo eso, sino que además son capaces de soportarlos con gran fluidez y sin apenas diferencia notoria respecto a otros dispositivos como el ordenador.

En un dispositivo móvil estos sistemas son creados en forma de aplicaciones móviles, y dentro de este ámbito existen distintos sistemas operativos que se pueden utilizar para crear estas aplicaciones. En este proyecto se ha decidido implementar una aplicación desarrollada para el sistema operativo Android por las siguientes razones:

- ✓ **SISTEMA ABIERTO:** En primer lugar, se trata de un sistema operativo de *open source,* o dicho en otras palabras, su código es totalmente abierto, gratuito y sin necesidad de adquirir licencias para poder utilizarlo. Además, Android brinda una libertad absoluta para instalar lo que se desee sin limitar al usuario (esto supone una gran ventaja siempre que se haga con conocimiento de ello).

- ✓ **LENGUAJE DE PROGRAMACIÓN:** En segundo lugar, empezar a desarrollar aplicaciones en Android implica menor tiempo que en otros sistemas operativos ya que se programa en Java, uno de los lenguajes de programación orientado a objetos más conocidos y utilizados. Este hecho facilita en gran medida el aprendizaje básico para poder desarrollar una aplicación.

- ✓ **COMUNIDAD DE USUARIOS:** En tercer lugar, existe una gran comunidad de desarrolladores con diversos tutoriales y explicaciones para entender y saber programar algunas funciones de Android. Además, la propia página oficial de Android [49] proporciona una documentación muy completa, así como ejemplos. Gracias a esta comunidad es posible encontrar más fácilmente una solución a problemas que puedan surgir a la hora de crear una aplicación.

- ✓ **MULTIPLATAFORMA:** En cuarto lugar, es un sistema operativo multiplataforma puesto que está diseñado para que cualquiera aplicación pueda ser ejecutada en teléfonos móviles, tablets, relojes inteligentes y otros dispositivos sin apenas tener que tener hacer cambios en ella.

- ✓ **LÍDER EN EL SECTOR:** Por último, es el sistema operativo con mayor cuota de mercado. Actualmente es el sistema operativo más utilizado en dispositivos móviles debido en gran parte a los puntos expuestos anteriormente. Como se puede ver la Figura 19 Android es el sistema operativo líder en ventas de dispositivos constituyendo aproximadamente a finales del año pasado un 80% de las ventas totales. En esta gráfica se puede observar con claridad el extraordinario aumento de las ventas de dispositivos con Android, mientras que el resto de sistemas operativos se han mantenido o han disminuido.

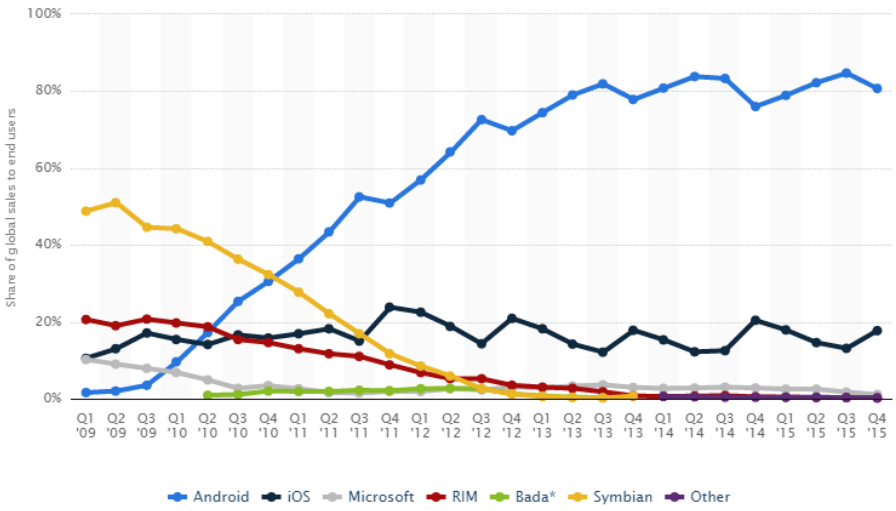


Figura 19 Porcentaje de ventas de los sistemas operativos [50]

En España más concretamente, analizando la Figura 20, se puede advertir más notoriamente el uso de este sistema operativo frente al resto, atrayendo a prácticamente el 90% de la población española que dispone de un terminal móvil. El segundo sistema operativo integrado en dispositivos más móviles más utilizado en España es iOS y se encuentra muy por debajo de esos valores, logrando solamente el 7% de las ventas.

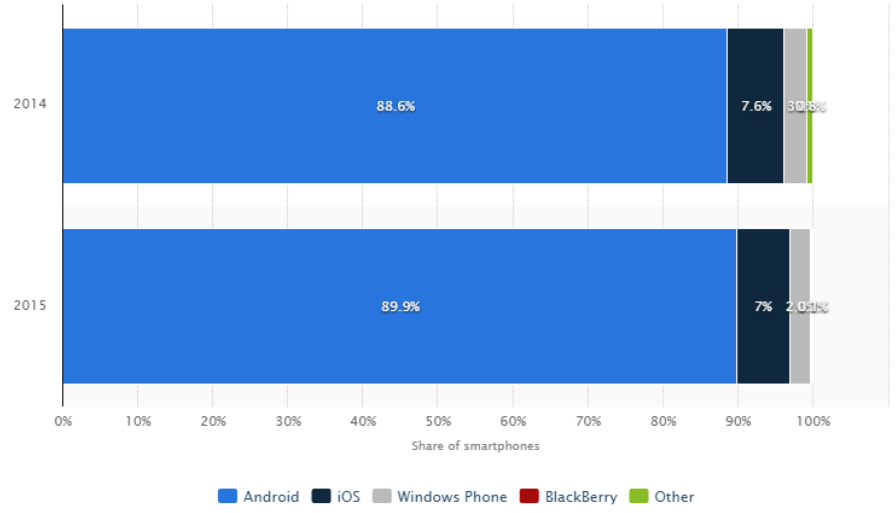


Figura 20 Porcentaje de ventas sistemas operativos en España [50]

Actualmente se pueden encontrar algunas aplicaciones muy de moda que utilizan algunos de los métodos mencionados anteriormente para el reconocimiento facial o la extracción de características.

Una aplicación muy conocida que usa algunos métodos mencionados anteriormente es “Snapchat” [51], una aplicación destinada a la creación y envío de archivos multimedia. La aplicación permite tomar fotografías, realizar vídeos, añadir texto o dibujar, pudiendo incluir incluso añadir filtros a las imágenes. Además, añade una característica muy destacable: la posibilidad de poder aplicar filtros en tiempo real localizando para ellos distintas zonas de la cara y registrando en cada instante su movimiento de tal manera que es capaz de realizar un seguimiento.

Tal y como se puede ver en la Figura 21(a) la aplicación realiza una detección y seguimiento de la cara basándose en la malla Máscara Candide-3 con 113 vértices y 168 superficies [26] explicada en el apartado anterior "Candide. Gracias a esa malla es capaz de registrar todos los movimientos de las facciones de la cara y poder aplicar máscaras sobre ella como el de la Figura 21(b).

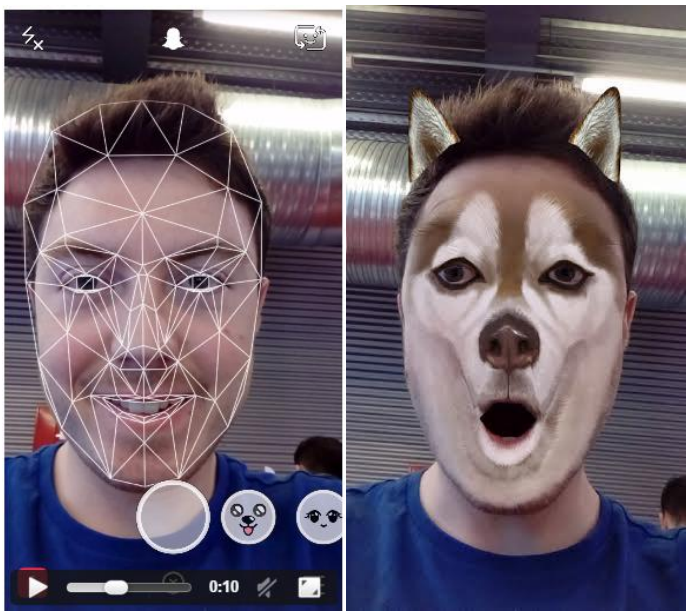


Figura 21 Ejemplo Snapchat. (a) Máscara Candide (b) Máscara aplicada

Otra aplicación que últimamente está ganando muchos adeptos es "MSQRD" o también conocida como "Masquerade" [52]. De manera similar a la anterior, es capaz de reconocer la cara y realizar un seguimiento de ella y todas sus características con el objetivo de poner máscaras sobre ella. En el caso de esta aplicación requiere que se encuadre la cara dentro de un cierto marco como se intuye en la Figura 22(a) y a partir de eso es capaz de encontrar todas las características de la cara. En el ejemplo de la Figura 22(b) se puede ver como se aplican los filtros a la cara una vez encuadrada.

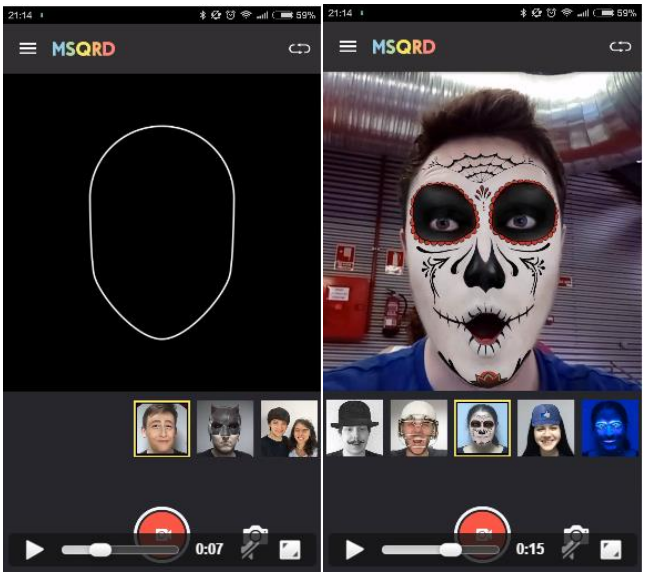


Figura 22 Ejemplo MSQRD. (a)Marco de encuadre (b) Filtro aplicado

Estas aplicaciones comentadas son un ejemplo de aplicaciones que implican detección y seguimiento de la cara, pero no reconocen las emociones como tal. Actualmente, a pesar de que no hay una gran cantidad de aplicaciones dedicadas a este último objetivo, cada vez son más los desarrolladores que se interesan por los dispositivos móviles. En [53] se recogen una variedad de soluciones de software que ofrecen APIs (*Application Programming Interface* o Interfaz de programación de aplicaciones) para el reconocimiento facial de emociones, y algunas de esas API están disponibles para dispositivos móviles.

Un ejemplo de ello es "FacioMetrics", una empresa que propone una solución destinada al reconocimiento facial de emociones en Android llamada IntraFace [54]. Esta aplicación propone un método para la detección, alineación y extracción de características de la cara [55] llamado SDM (*Supervised Descent Method* o método de descenso supervisado).

Lo primero que realiza este método es una estimación de 66 puntos de interés centrando una cara promedio en un cuadrado normalizado como se puede ver en la Figura 23.Una vez lograda esta estimación, se procede a analizar cada una de las imágenes del entrenamiento para poder calcular las diferencias entre los puntos de interés estimados inicialmente y los calculados. Para ello el primer paso que realiza es la detección y seguimiento de la cara. Tras ello, se calculan descriptores sobre zonas de 32x32 píxeles y se aplica la técnica PCA para reducir la cantidad de información.

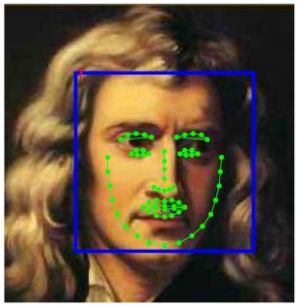


Figura 23 Puntos estimados de interés en SDM

En el apartado de la localización de las características, una de las principales diferencias con otros métodos como AAM es el hecho de que este método realiza una concatenación de regresiones que otorgan una mayor eficacia. Además, a diferencia de este método, SDM no aprende ningún modelo de apariencia, es decir, es un método no paramétrico.

Para el seguimiento y control de los cambios de las características faciales, la idea principal reside en utilizar el método SDM para la detección en cada imagen pero utilizando como puntos estimados los puntos de interés localizados en la imagen inmediatamente anterior.

Se realizaron pruebas con diferentes bases de datos obteniendo resultados óptimos en la mayoría de las imágenes. Estas imágenes, además de ser espontáneas, incluían cambios en la pose y la iluminación y partes de la cara tapadas. En la Figura 24 se pueden ver algunos ejemplos de la utilización de este método.

Figura 24 Ejemplo localización de puntos SDM

Esta aplicación también propone un nuevo método para la detección de unidades de acción (AUs) llamado STM (*Selective Transfer Machine* o máquina de transferencia selectiva) [56]. Está inspirado en el clasificador SVM y su idea principal consiste en otorgar un mayor peso a las imágenes de entrenamiento que tengan más relevancia para la nueva imagen a analizar.

Por ejemplo, en la Figura 25 se puede ver la proyección de las muestras positivas y negativas para la AU 12 (elevación de los extremos de los labios) y como un clasificador lineal puede separarlas. En este caso se usan los mismos sujetos para entrenar y para testear y por tanto se denomina clasificador ideal. Sin embargo, cuando un clasificador es entrenado usando un conjunto de sujetos pero es testeado con un sujeto diferente se denomina clasificador genérico. STM propone un clasificador genérico personalizado como el que se ilustra en la Figura 25 para mejorar la tarea de la detección de las unidades de acción.

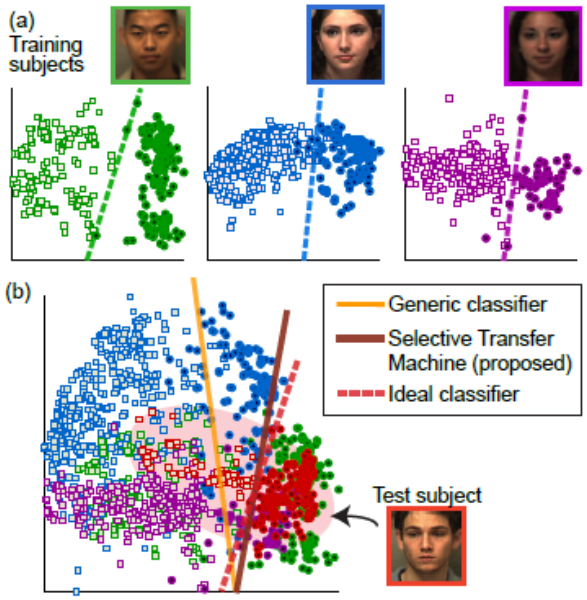


Figura 25 Clasificador STM. (a) clasificador ideal (b)comparación clasificadores[56]

Comparando este método con otros como SVM para la detección de AUs y utilizando la base de datos CK+ se observan mejoras en cuanto al rendimiento como se puede ver en la Tabla 2.

Tabla 2 Comparación en porcentajes (muestras positivas sobre el total) entre los métodos SVM y STM para la detección de AUs

AU	SVM	STM
1	79,8	88,9
2	90,8	87,5
4	74,8	81,1
6	89,7	94,0
7	82,1	91,6
12	88,1	92,8
15	93,5	98,2
17	90,3	96,0
Avg.	**86,1**	**91,3**

Analizando la aplicación (disponible en Google Play) se puede comprobar que es capaz de detectar hasta cuatro emociones distintas además de la neutral como se puede ver en la Figura 26 (tristeza, asco, sorpresa y felicidad). Tras realizar varias pruebas se puede concluir que el ratio de acierto a la hora de reconocer las emociones de felicidad y sorpresa es muy elevado, pero sin embargo confunde con cierta frecuencia asco y tristeza y hay que destacar que no reconoce dos de las emociones más complicadas de detectar y que más confusiones suelen generar: miedo y enfado. Como punto a favor la detección de las emociones es prácticamente invariable con los cambios de iluminación.

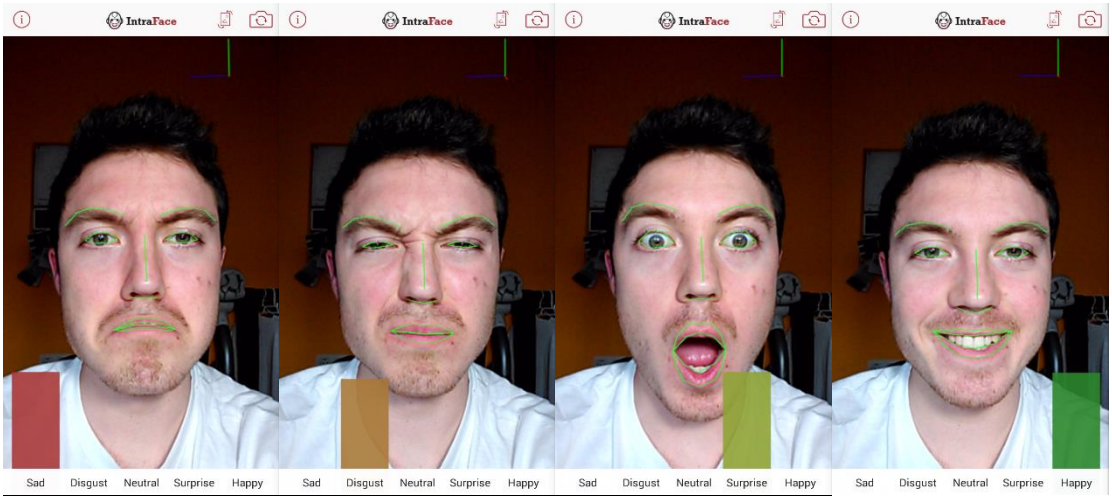


Figura 26 Ejemplos aplicación IntraFace

Otro software dedicado al reconocimiento de emociones en dispositivos móviles es “Affectiva” [57] una plataforma que no solo dispone de una aplicación móvil sino también de una API propia. Otorga la posibilidad a cualquiera de integrar (previo pago de la licencia) las funciones destinadas al reconocimiento en una aplicación.

Su kit de desarrollo de software permite localizar y seguir las caras en una imagen, así como extraer los puntos de interés correspondientes a cada cara utilizando un modelo muy parecido al que se aplica en IntraFace [55].

Tras este paso, este software permite la detección y seguimiento de 15 AUs o acciones faciales, algunas definidas por las FACS y otras personalizadas. En la Figura 27 se ilustra un ejemplo de algunas de ellas. Además, propone un método para la detección de las AUs usando un aprendizaje activo (se entrena el clasificador con un subconjunto de muestras adaptado que garantice el máximo número de muestras positivas) y basándose en un *kernel* no lineal eficiente llamado Nyström [58]. Este *kernel* es parecido al que utiliza SVM y comparando los resultados que se obtienen con ambos logran mejorar la detección de las AUs.

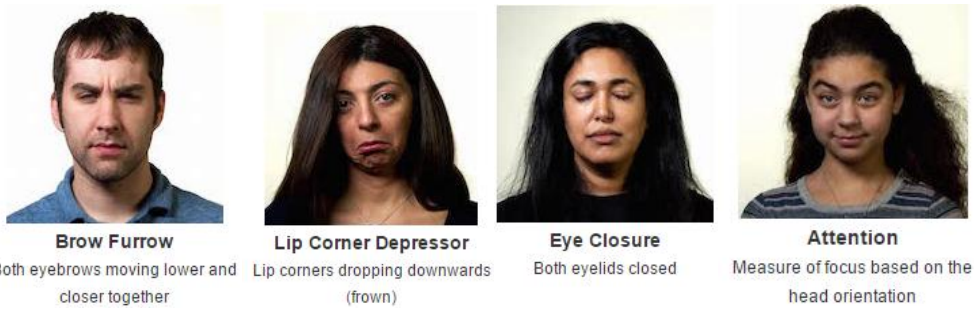


Figura 27 Ejemplos de AUs en Affectiva. Las dos de la izq. son AU4 y AU15 definidas por las FACS mientras que las dos situadas a la dcha. son AUs personalizadas por el software.

Además es capaz de extraer información sobre la apariencia física como el género del sujeto o la existencia de gafas o gafas de sol presentes en el sujeto con un alto nivel de confianza. La Figura 28 ilustra cómo se muestra esta información en pantalla.

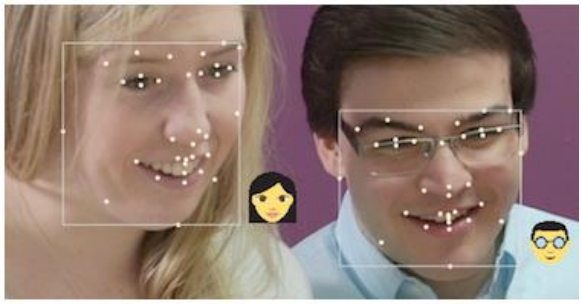


Figura 28 Ejemplo Affectiva mostrando la cara detectad, 33 puntos reconocidos, el género y si lleva gafas

Una vez reconocidos los puntos, es capaz de reconocer siete emociones además de la neutral, las seis básicas y desprecio. Sin embargo, en la aplicación móvil de ejemplo que ofrece, "Affdexme" [59], solamente es posible localizar las seis emociones canónicas. En la Figura 29 se puede observar un ejemplo de las emociones reconocidas. Tras realizar distintas pruebas se comprueba que, al igual que en el caso de IntraFace, tiene una buena respuesta frente a distintas iluminaciones. Respecto al reconocimiento de emociones detecta sorpresa, asco, ira y felicidad sin aparentes problemas pero en cambio el resto de emociones se confunden de vez en cuando. Sin embargo, es destacable el hecho de que la emoción tristeza rara vez es detectada y se confunde casi siempre con asco como ocurría en IntraFace. La emoción de miedo suele confundirla con sorpresa.

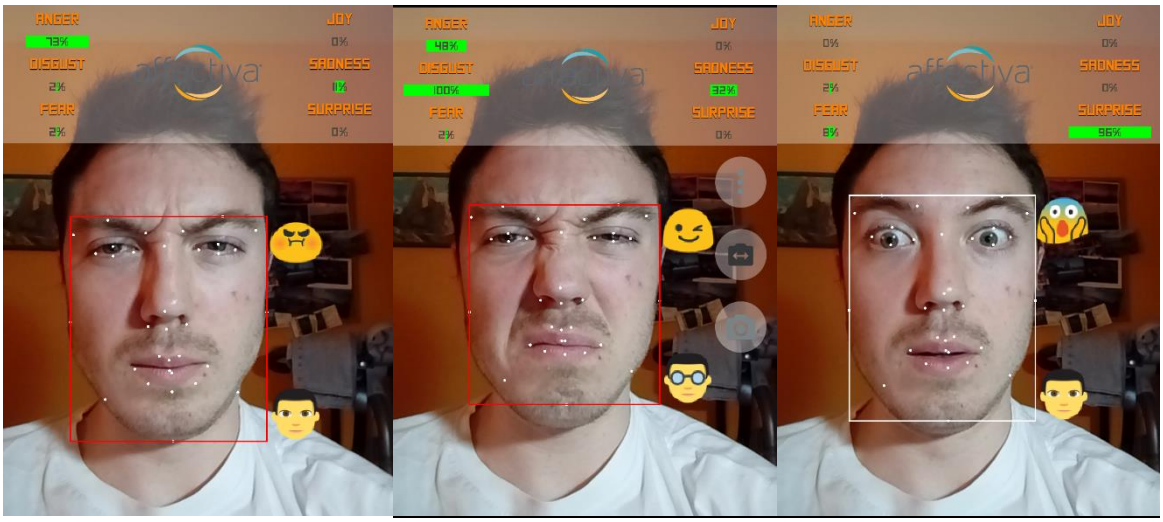


Figura 29 Ejemplo de uso Affdexme. De izquierda a derecha: emoción 'enfado', emoción 'asco' y emoción 'miedo' reconocida como sorpresa.

También ofrecen una base de datos [60] que consiste en 242 vídeos de caras grabadas (168359 imágenes) en condiciones reales, y cada imagen está etiquetada con el género de la persona, las AUs presentes, los movimientos de la cabeza, los fallos a la hora de detectar, etc. También aporta la localización automática de punto de interés e información sobre el rendimiento inicial de los algoritmos de detección aplicados a esta base de datos junto con resultados obtenidos al aplicar un algoritmo de detección de AUs personalizado.

3. Herramienta de referencia: “Interface Faces”

Este proyecto tiene como punto de partida la herramienta desarrollada por S. González [34] para su proyecto final de carrera. Dicha herramienta fue diseñada e implementada en Matlab, el lenguaje para procesamiento de imágenes por excelencia, y permite comparar la eficiencia de diferentes métodos para el reconocimiento facial de emociones pudiendo combinar ciertas técnicas de extracción de características y dos modelos de clasificación distintos.

El procedimiento que sigue la herramienta para reconocer una emoción es el siguiente: primero toma como entrada una imagen a analizar y busca una imagen con expresión neutral del mismo sujeto. Tras ello, analiza las dos imágenes extrayendo los puntos de interés de cada una de ellas, pudiendo elegir dos métodos para hacerlo. Después, a partir de la comparación de los puntos de ambas imágenes el sistema es capaz de clasificar la emoción, pudiendo escoger entre dos modelos de clasificación. A continuación se explica en concreto qué proceso se realiza sobre cada una de las imágenes y cuáles son los posibles modelos a elegir.

En primer lugar, como cualquier sistema de reconocimiento facial, es capaz de detectar la cara. En este caso se utiliza el algoritmo de Viola&Jones descrito en el Estado del arte y que se encuentra implementado en la *Computer Vision Toolbox* de Matlab. El resultado de ello se puede ver en la Figura 30. Una vez localizada la cara, ésta es recortada y redimensionada a 300x300 píxeles, un tamaño que permite realizar después un procesado óptimo en un período razonable de tiempo.

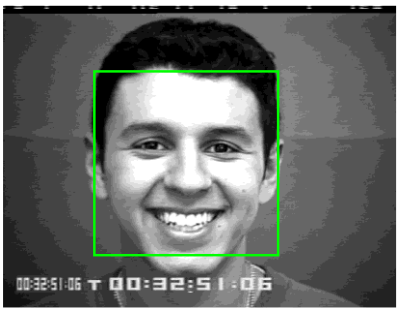


Figura 30 Ejemplo detección cara usando algoritmo Viola&Jones Matlab

El siguiente paso que realiza la herramienta es localizar las distintas zonas de interés de la cara: ojos, cejas, nariz y boca. Para lograr este objetivo se localizan los ojos de forma utilizando un algoritmo similar al de detección de la cara y a partir de ellos se divide la cara en regiones fijas de tal manera que al final quedan totalmente separadas unas de otras como se puede ver en la Figura 31.

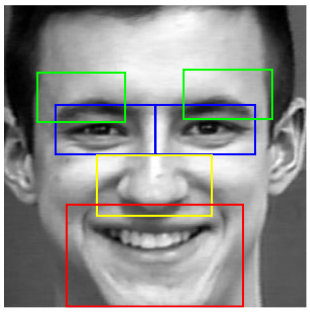


Figura 31 Ejemplo localización regiones de la cara

En tercer lugar se realizar un procesado de cada una de las regiones para intentar localizar los puntos más significativos de ellas. En este sistema es posible localizar hasta 19 puntos en total. El procesado consiste en la aplicación de técnicas de segmentación. Existen dos métodos a elegir para aplicar estas técnicas: método de umbralización y método de detección de bordes. Son métodos diferentes pero generalmente divididos en tres fases: aumento del contraste, detección de contornos y operaciones morfológicas para eliminar ruido.

Una vez segmentada la región se localizan los puntos de interés obteniendo los puntos extremos de cada una de ellas. Concretamente se detectan cuatro puntos para cada ojo, tres para cada ceja, uno para la nariz y cuatro para la boca como se muestra en el ejemplo de la Figura 32.

Figura 32 Ejemplo localización de puntos [34]

Por último, se clasifica la emoción en función de los puntos obtenidos, proponiendo para ello dos métodos. Por una parte, se encuentra el método basado en matrices ideales, que consiste en la comparación de las posiciones de los puntos y sus respectivas variaciones entre la expresión neutral y la emoción a analizar. El procedimiento consiste en crear en la fase de entrenamiento una matriz ideal representativa de cada una de las seis emociones basándose en el hecho de que cada emoción tiene una serie de características en común. Cada matriz es creada realizando el promedio de un gran número de imágenes con la misma emoción. Una vez que el sistema está entrenado, cuando se introduce una nueva emoción a reconocer se crea una matriz de las mismas dimensiones que las ideales y se compara con cada una. La emoción reconocida será aquella que reporte menor diferencia entre su matriz ideal y la matriz de la nueva emoción.

Por otra parte, otro de los métodos propuestos consiste en utilizar unidades de acción detectadas a partir de los puntos reconocidos. Las AUs se detectan comparando los puntos de cada emoción con los puntos de la expresión neutral. Dependiendo de que AU se trate de detectar se evalúan unos puntos u otros. En esta herramienta se utilizan alguna de las AUs definidas por el FACS

pero además se han incorporado otras que se consideraban importantes a la hora de detectar alguna de las emociones. Una vez detectadas estas unidades, se localizan las más relevantes y se realiza un árbol de decisión manual en función de ello. Una vez entrenado el sistema, cuando una nueva imagen entra en el sistema, se localizan los puntos de interés y se comparan con su neutral para poder detectar posteriormente que unidades están presentes. Dependiendo de cuales estén activas se decide clasificar una emoción u otra.

Esta herramienta consta además de una interfaz gráfica en la cual se pueden controlar todos los posibles métodos a aplicar y mostrar todos los resultados. Esta interfaz se puede visualizar en la Figura 33 y dispone de cuatro ventanas claramente diferenciadas.

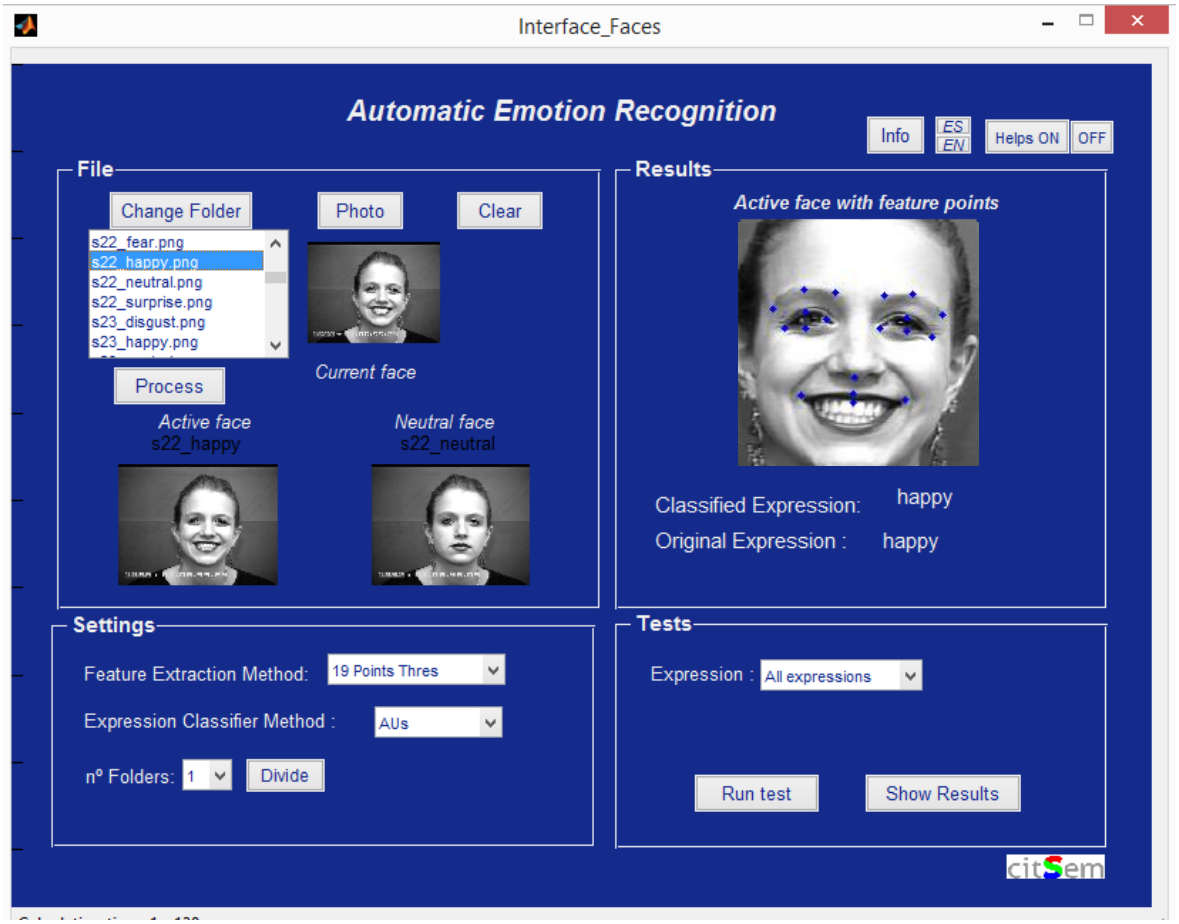


Figura 33 Ejemplo interfaz gráfica herramienta Matlab

En primer lugar, se encuentra la ventana de *Files*. En ella se obtiene la imagen a analizar a partir de una imagen ya almacenada o tomando una captura desde la webcam. Si se ha optado por la primera opción es posible obtener una pre visualización de la misma. Además, una vez seleccionada la imagen se muestra también la imagen con la expresión neutral. Si se elige realizar una captura la herramienta primero solicita una imagen con expresión neutral y después una con la emoción a analizar.

En segundo lugar se encuentra el apartado de *Settings,* en el cual se elige el método de extracción de características deseado y el modelo de clasificación que se quiere emplear. Una vez seleccionados, se puede proceder a reconocer la emoción.

La tercera ventana contiene los resultados. En ella se pueden observar los puntos extraídos de la imagen así como la emoción original y la reconocida por el sistema. En el caso de la Figura 33 la emoción ha sido correctamente reconocida como *happy* o feliz.

Por último, consta de una cuarta ventana utilizada para pruebas. En función de los métodos elegidos en el apartado de *Settings,* este módulo permite obtener el ratio de aciertos/fallos de clasificación de la emoción de serie de imágenes almacenadas en una base de datos. Los resultados los almacena en porcentaje en una matriz de confusión guardada en un archivo Excel.

Esta herramienta cuenta además con mensajes de ayuda con una breve descripción para explicar la funcionalidad de cada uno de los botones y ventanas, siendo también posible seleccionar el idioma en el que se muestra la información (español e inglés).

4. Desarrollo de la aplicación sobre la plataforma Android

Tras haber realizado un estudio del estado del arte hasta la actualidad se ha comprobado que, a pesar de que existen un gran número de herramientas capaces de reconocer emociones, apenas se han implementado para uno de los dispositivos más utilizados hoy en día: el Smartphone. Si bien es cierto que la potencia y velocidad que puede aportar un ordenador es considerablemente mayor, cada vez es más frecuente encontrarse dispositivos móviles capaces de realizar tareas que requieren un gran procesamiento y que años atrás eran impensables.

Con esta solución, se pretende introducir la capacidad de reconocer una emoción de una manera más ágil y cómoda y, aunque que no se obtenga con la misma precisión y fluidez que proporcionan otros equipos, puede ser de gran utilidad para la realización de pruebas o para un uso básico en el cual la rapidez sea lo más importante.

4.1 Análisis y diseño

Para poder desarrollar esta solución se han estudiado algunos aspectos como:

- Los requisitos que debe cumplir la herramienta.
- Las especificaciones de la solución que se propone.
- Los requisitos mínimos del dispositivo móvil para la correcta ejecución.
- El esquema general que va a tener la aplicación.

4.1.1 Análisis de los requisitos

La aplicación a desarrollar debe realizar unas tareas específicas o proporcionar unos servicios concretos. Por una parte se encuentran los requisitos funcionales (RF), que determinan el comportamiento del sistema y cómo debe reaccionar éste ante determinadas situaciones y por otra parte los requisitos no funcionales (RNF), que definen propiedades del sistema que afectan a los servicios o funciones del mismo, tales como restricciones de tiempo, del proceso de desarrollo o estándares a utilizar. A continuación se especifican los requisitos tanto funcionales como no funcionales que debe tener el sistema:

RF1:	La aplicación debe poder ejecutarse en distintos terminales.
RF2:	La aplicación debe ser compatible con la última versión del sistema operativo.
RF3:	La aplicación debe mostrar mensajes de error explicativos cuando afecten al usuario.
RF4:	La aplicación debe reconocer la emoción en tiempo real.
RF5:	La aplicación debe ser capaz de funcionar tanto con la cámara delantera como con la cámara trasera.

RF6: La aplicación debe mostrar o indicar por pantalla la emoción detectada.

RF7: La aplicación debe permitir el guardado de la imagen con la emoción reconocida si se ha elegida la opción de reconocimiento estático.

RF8: La aplicación debe ser capaz de reconocer la emoción esté el móvil con orientación vertical u horizontal.

RF9: La aplicación no debe dejar de funcionar o cerrarse inesperadamente salvo que el usuario lo desee.

RF10: Al dar el botón de "atrás" la aplicación debe volver a la primera pantalla de la aplicación si quiere descartar la imagen neutral después de haberla realizado.

RNF1: La aplicación debe tener una interfaz clara y sencilla.

RNF2: La aplicación debe ser fácil de utilizar e indicar al usuario como proceder en cada paso.

RNF3: La aplicación debe contener solamente lo necesario para reconocer la emoción.

RNF4: La aplicación debe ocupar lo menos posible para no afectar a la experiencia del usuario.

RNF5: La aplicación debe ser fácil de analizar para poder corregir los errores lo ante posible.

RNF6: La aplicación debe de proporcionar la máxima precisión posible a la hora de reconocer la emoción.

RNF7: La aplicación debe tener respuesta lo más cortos posibles para que su funcionamiento sea óptimo en tiempo real.

4.1.2 Especificaciones

Como se ha introducido en el estado del arte, el sistema operativo elegido para el desarrollo de la aplicación es Android debido a las ventajas que ofrece.

Entre las ventajas comentadas destaca la gran cantidad de librerías disponibles, sirviendo de gran ayuda para la implementación. De hecho, la comunidad de desarrolladores de Android facilita una serie de librerías dedicadas a cada una de las versiones que se van desplegando año tras año. En este caso en concreto, se han utilizado las librerías de Android 6.0 *Marshmallow* [49] aunque la aplicación podrá ser ejecutada por la última versión disponible.

También se han utilizado las librerías de código abierto de OpenCV [61] que proporciona una gran cantidad de funciones relacionadas con el tratamiento de la imagen. En este proyecto se ha utilizado la versión 3.0 de OpenCV.

El IDE (*Integrated Development Environment* o Entorno de desarrollo integrado) usado como base para este proyecto es Eclipse [62] un entorno ideal para la programación gracias a su rendimiento y versatilidad. Al mismo tiempo del comienzo de la realización de este proyecto Android ha lanzado su propio entorno de desarrollo llamado Android Studio [63] y en ese momento se

encontraba en fase beta. Actualmente ya dispone de versiones estables, se ha convertido en el IDE oficial de Android desbancado a Eclipse.

En el Anexo I del proyecto se puede encontrar un manual detallado de la instalación de todos los componentes y librerías necesarias para el desarrollo de la aplicación.

Por último, para la realización de pruebas y análisis de las mismas se utiliza un dispositivo externo en lugar de un terminal virtual debido al gran consumo de memoria RAM por parte de éstos últimos y al grado de procesamiento que requiere la aplicación. No obstante, en el anexo también se encuentra una explicación de cómo ejecutar las aplicaciones en un entorno virtual, tanto el facilitado por el propio IDE como otros independientes.

4.1.3 Requisitos mínimos del dispositivo

El dispositivo en disposición de ejecutar la aplicación debe cumplir unos requisitos mínimos con el fin de evitar incompatibilidades o un mal procesamiento. Cuanto más se aleje su dispositivo de estos requisitos con mayor estabilidad, fluidez y precisión funcionará.

Tal y como se ha comentado en el apartado anterior 3.2, Android dispone de varias versiones y en función de la versión que incorpore su terminal se incluirán más o menos características. En este proyecto la versión mínima de Android que se debe tener es la *4.0 Ice Cream Sandwich,* correspondiente a la API 14.

En la Figura 34 ya la Tabla 3 se puede comprobar la distribución porcentual de las diversas versiones que incorporan los dispositivos a principios de 2016, y como se puede observar la mayoría de usuarios Android disponen de una versión igual o superior la mínima requerida. En concreto, solamente un 2,3% tienen una versión instalada inferior y por tanto no compatible y tan solo un 2 % incorpora la versión mínima. El resto de dispositivos, un 95,7%, vienen con una versión más reciente y por tanto superan el mínimo requerido.

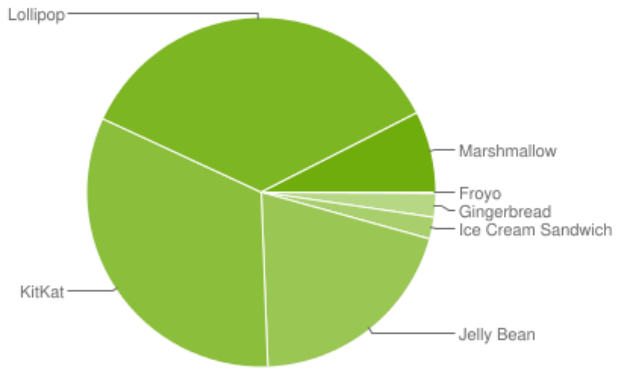


Figura 34 Gráfica versiones Android (Mayo 2016) [64]

Tabla 3 Porcentaje uso versiones Android

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 -2.3.7	Gingerbread	10	2.2%
4.0.3-4.0.4	Ice Cream Sandwich	15	2.0%
4.1.x	Jelly Bean	16	7.2%
4.2.x		17	10.0%
4.3		18	2.9%
4.4	KitKat	19	32.5%
5.0	Lollipop	21	16.2%
5.1		22	19.4%
6.0	Marshmallow	23	7.5%

En general, las especificaciones mínimas que debe tener el terminal móvil para garantizar estabilidad, un procesamiento adecuado y uso de todas las funciones disponibles son:

- Sistema operativo: Android 4.0 Ice Cream Sandwich
- Memoria RAM: 1GB
- Procesador: 1GHz
- Pantalla: 4 pulgadas
- Existencia de cámara frontal: 1,3 megapíxeles

En la Figura 35 se muestran los dispositivos que se han utilizado para la comprobación del correcto funcionamiento y sus principales características.

Xiaomi Redmi 1s Xiaomi Redmi Note 3

Figura 35 Móviles utilizados en el desarrollo de la aplicación [65]

Características:	Xiaomi Redmi 1s	Xiaomi Redmi Note 3
Sistema operativo:	Android 4.4	Android 5.0
GPU:	Adreno 305	PowerVR G6200
Memoria RAM:	1GB	3GB
Procesador:	Qualcomm Snapdragon 400 1,6 GHz	MT6795 X10 Helios 720 2,2 GHz
Pantalla:	4,7 pulgadas (1280x720)	5,5 pulgadas (1920x1080)
Cámara frontal:	1,3 megapíxeles	5 megapíxeles

4.1.4 Esquema general de la aplicación

Como se puede observar en la estructura general en el esquema de la Figura 36 procedimiento que sigue la aplicación para reconocer una emoción es similar al implementado en cualquier sistema de este tipo. En primer lugar se captura y almacena la imagen neutral una vez que el usuario está preparado. En la fase de procesado de la imagen se detecta la cara, se localizan las regiones y se buscan los puntos de interés. Posteriormente el usuario ya puede expresar la emoción a detectar para que la aplicación sea capaz de reconocerla. La aplicación analiza todas las imágenes capturadas en tiempo real y realiza los mismos pasos que para la imagen neutral. A continuación, compara los puntos de la imagen con emoción y la imagen neutral guardada para detectar las unidades de acción presentes en esa imagen. Por último, una vez determinadas esas AUs, se introducen en un clasificador entrenado para que clasifique la emoción. En este proyecto el objetivo es detectar seis emociones básicas: *anger* (ira), *disgust* (asco), *fear* (miedo), *happiness* (felicidad), *sadness* (tristeza) y *surprise* (sorpresa).

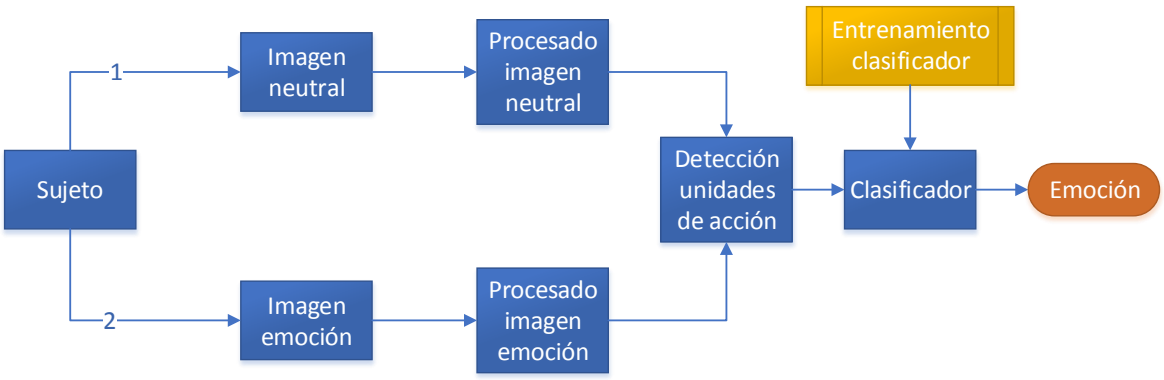


Figura 36 Esquema general reconocimiento facial de emociones

Este proceso se explicará con más detalle en el siguiente apartado, explicando paso a paso todo el proceso que sigue la aplicación para reconocer una emoción. Asimismo, aparte de esta estructura general, se han añadido algunas características que facilitan el uso de la aplicación y la realización de pruebas.

Por una parte se ha incluido un botón que permite conmutar entre la cámara delantera y la cámara trasera del dispositivo. Por defecto la aplicación utiliza la cámara frontal. Cada vez que se pincha sobre el botón la aplicación solicita volver a capturar la imagen con la expresión neutral ya que el sujeto a analizar cambia y por tanto se necesita su expresión neutral.

Por otra parte, se ha añadido un botón que permite reiniciar el proceso de tal manera que si la neutral ha sido incorrectamente tomada se puede volver a capturar.

Por último se han agregado dos botones para pruebas. Uno de ellos permite mostrar las regiones detectadas en la cara y el otro posibilita la opción de visualizar los puntos localizados en la cara, aunque en este último caso aparecerá en una ventana pequeña para que no haya problemas de ralentización.

En la Figura 37 se muestra el diseño básico de la interfaz de la aplicación y en el esquema de la Figura 38 un diagrama de flujo de la aplicación teniendo en cuentas estas características añadidas.

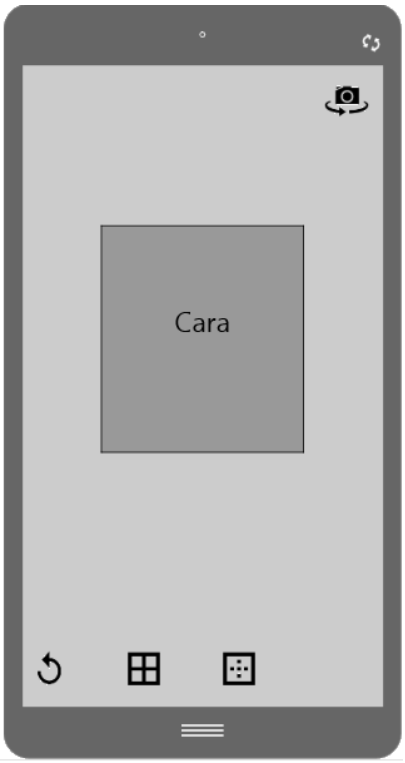


Figura 37 Diseño básico interfaz de la aplicación

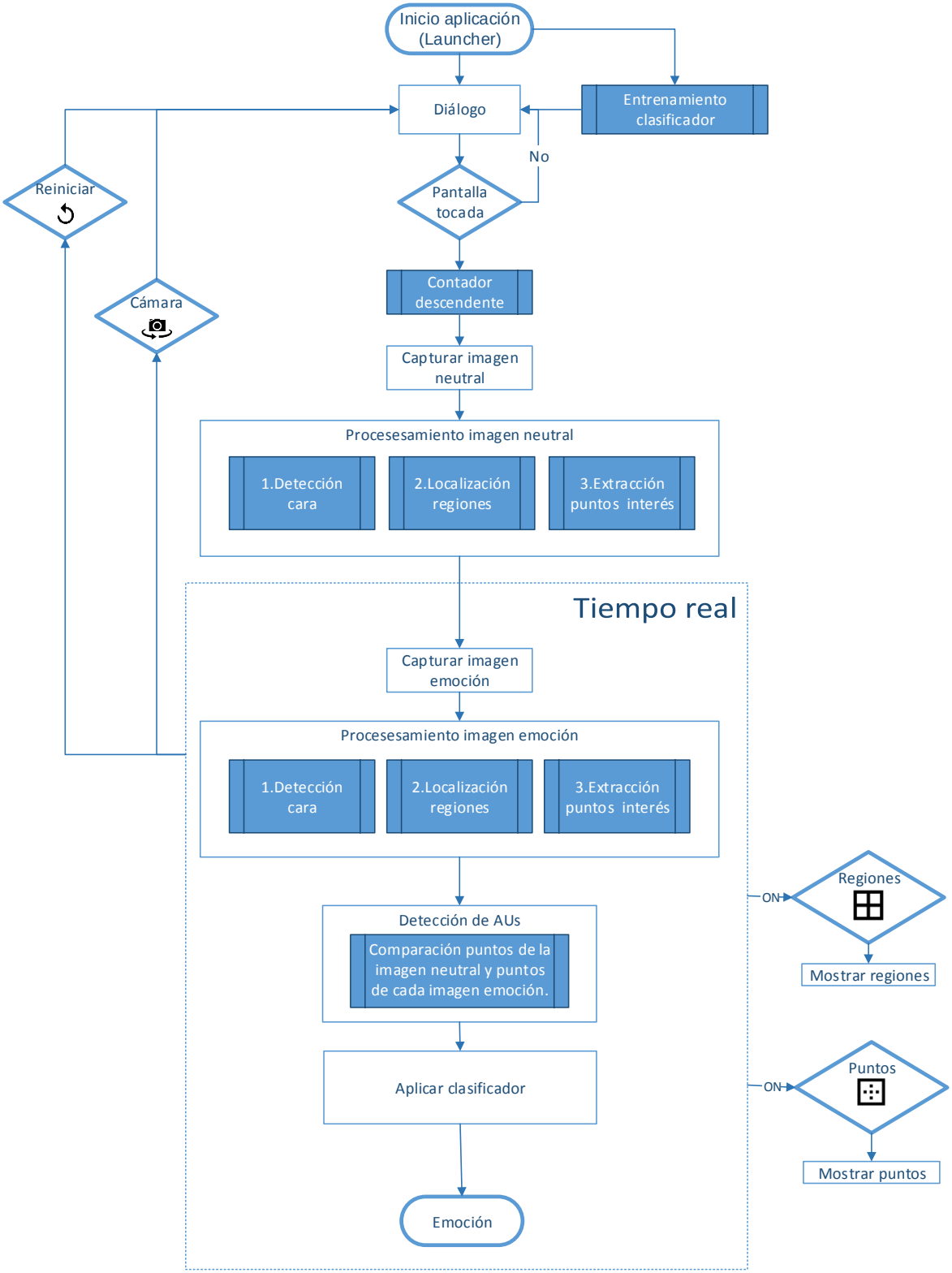


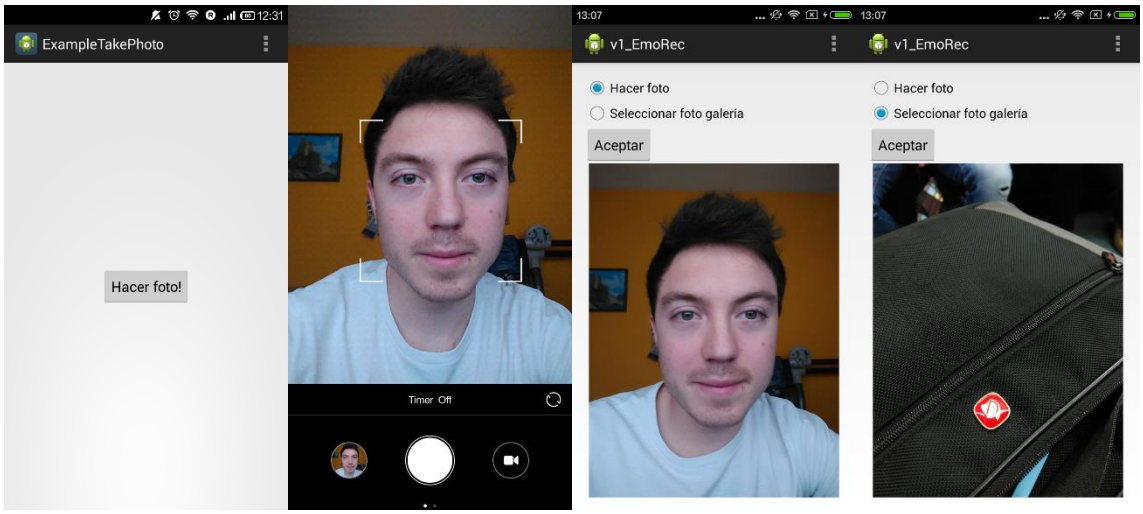
Figura 38 Diagrama de flujo de la aplicación

4.2 Implementación

En este apartado se detalla el progreso que se ha seguido para realizar la aplicación. Primero se describirán los primeros pasos realizados sobre Android, así como los primeros ejemplos realizados como introducción para la versión final de la aplicación. Asimismo, se comentarán los principales problemas encontrados al principio del desarrollo. Más tarde, partiendo de la estructura general de la aplicación (Figura 36) se explicará más detalladamente el procedimiento de cada uno de los pasos en el reconocimiento de una emoción. Se irán comentando las diferencias con la herramienta de Matlab, aclarando las adaptaciones que se han tenido que realizar así como las funciones de la librería usadas para ello. Asimismo, se mostrarán algunas pruebas realizadas durante el proceso.

4.2.1 Primeros pasos

En primer lugar, después de estudiar, analizar e investigar sobre Android se realizaron los primeros ejemplos de aplicaciones con el objetivo de conocer más a fondo la estructura y el funcionamiento de este sistema operativo. Se efectuaron algunos ejemplos como los que se pueden observar en la Figura 39. Las Figura 39 (a) y (b) muestran una simple aplicación que permite pinchar sobre un botón y acceder a la aplicación de la cámara del dispositivo. Las Figura 39 (c) (d) presentan una aplicación capaz de acceder a la cámara o mostrar una imagen previamente almacenada en la galería en función de lo que se haya elegido.



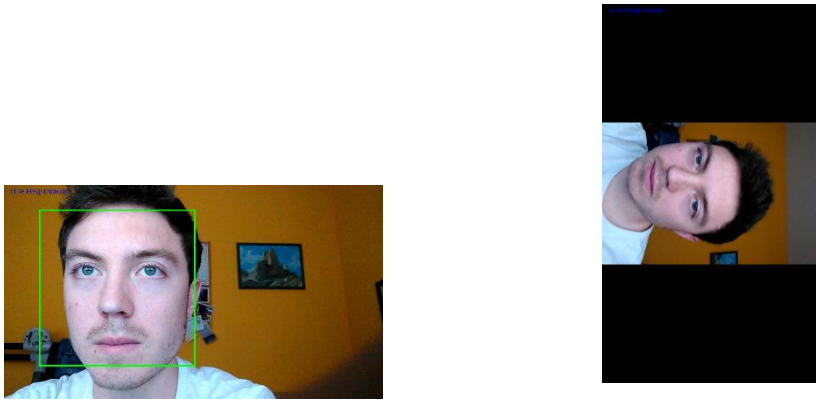
(a) *(b)* *(c)* *(d)*

Figura 39 Ejemplos primeras aplicaciones

El siguiente paso que se realizó fue la integración de la librería OpenCV para poder comunicar la cámara física con la aplicación para poder crear una aplicación propia y no depender de la aplicación que incorpora el dispositivo. Esto facilita la posibilidad de crear una aplicación personalizada y que sea capaz de otorgar mayor rapidez y fluidez. Esta propia librería incluye algunos ejemplos para ello, desde una simple herramienta que muestra lo que se captura por la cámara hasta otras más complejas que son capaces de dar una primera aproximación para la detección de caras.

Tomando ese último ejemplo como referencia se empezó a crear la primera versión de la aplicación desarrollada para este proyecto. Este ejemplo muestra cómo utilizar los algoritmos de detección que vienen integrados en la librería y cómo visualizar después los resultados de la detección. Uno de los problemas principales que este ejemplo tiene y que se tardó en solucionar fue el hecho de que solamente permite reconocer caras cuando el móvil se coloca en horizontal (modo *landscape* de Android), un uso muy poco frecuente y de escasa utilidad cuando el objetivo de la aplicación se centra en la cara (ver Figura 40(a)). El uso más común del dispositivo cuando la aplicación implica una interacción con la cara del sujeto es colocar el dispositivo en vertical y utilizando la cámara delantera como se puede observar también en las aplicaciones analizadas en el capítulo del Estado del arte.

Al existir este problema, cuando se programaba que el móvil funcionara en modo vertical (modo *Portrait* de Android) y usando la cámara delantera ocurría un efecto indeseado como el que se muestra en la Figura 40 (b) y no detectaba la cara correctamente.



(a) *(b)*

Figura 40 Problema integración librería OpenCV

Este problema ocasionó una demora considerable en el desarrollo del proyecto ya que se trata de un bug de la librería. Se trató de girar la imagen para que mostrara la correcta visualización pero la aplicación seguía procesando internamente la imagen inicial por lo que tampoco era válido este proceso (ver Figura 41). Tras numerosas pruebas e intentos de tratar de resolver esta cuestión, al final se solucionó creando una nueva clase llamada *PortraitCameraView* basada en una clase que utiliza la librería para comunicar la cámara con la aplicación. Se tuvieron que modificar parámetros, crear nuevos métodos y realizar una nueva estructura. Una vez creada esta clase la aplicación ya es capaz de obtener la imagen de la cámara en vertical y con la cámara delantera.

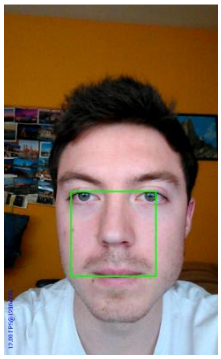


Figura 41 Ejemplo detección cara al girar la imagen

4.2.2 Procesado de la imagen

Una vez solucionado este error se pudo empezar a desarrollar la primera versión de la aplicación para el reconocimiento facial de emociones. Siguiendo el esquema de la Figura 42, se tienen que procesar dos imágenes. En primer lugar una imagen con expresión neutral y en segundo lugar la imagen con la emoción a detectar. El proceso que sufren las dos imágenes es idéntico y se divide en varias etapas representadas en el Figura 42

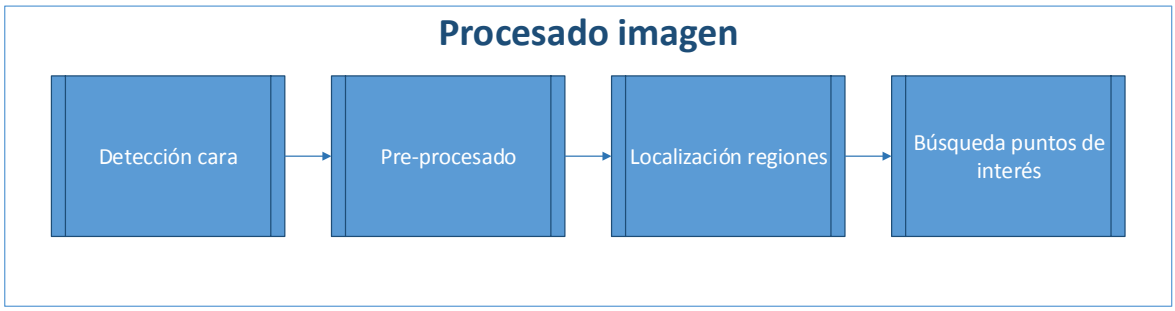


Figura 42 Etapas proceso de la imagen

Detección de la cara

En la primera etapa se realiza la detección de la cara. Para ello se pueden utilizar métodos como los expuestos en el Estado del arte. Por ejemplo, gracias a la librería OpenCV es posible realizar la detección con el algoritmo de Viola&Jones basado en características Haar o usando LBP. El primero paso dentro de esta etapa consiste en cargar previamente ese algoritmo de un archivo XML.

Una vez cargado el algoritmo, como paso previo para la detección hay que convertir la imagen a escala de grises en caso de que no lo esté. Después, para llevar a cabo la detección se pueden modificar una serie de parámetros que ayudan a limitar y ajustar la detección como:

- Factor de escala: parámetro para determinar cuánto se reduce la imagen para que pueda detectar la cara. A menor factor de escala mayor precisión pero mayor retardo también. En este caso se emplea un factor de escala de 1,1, es decir, reducción del 10%.
- Número de vecinos: especifica el mínimo número de píxeles vecinos a tener en cuenta. A menor número menor precisión pero más detecciones. En este caso se ha estimado que tres vecinos es el número óptimo.
- Tamaño mínimo y máximo: determinan cuales pueden ser el tamaño mínimo y máximo de la cara. En este caso se ha determinado que el tamaño mínimo tenga una altura al menos del 20% de la altura de la imagen

Además, hay distintas versiones de los algoritmos. Se han comparado, tanto en eficiencia como en rapidez, los dos métodos y utilizando todos los algoritmos disponibles. En general todos proporcionan resultados favorables en cuanto a eficiencia, sin embargo si se aprecian diferencian respecto a la rapidez. A continuación, en la Tabla 4 se encuentra un resumen de la rapidez de cada uno de los algoritmos bajo diferentes condiciones de iluminación y en la Figura 43 una comparación de la detección de la cara proporcionada por algunos de los algoritmos probados.

Tabla 4 Comparación algoritmos detección de cara en función de la luminosidad y la velocidad de detección

Algoritmo	Luminosidad (lx[1])	Velocidad media (ms)
haarcascade_frontalface_alt_tree	40	36,4
	200	40,9
	450	37,3
haarcascade_frontalface_alt	40	35,3
	200	44,6
	450	34,2
haarcascade_frontalface_alt2	40	41,1
	200	45,9
	450	41,7
haarcascade_frontalface_default	40	56,3
	200	64,3
	450	59,2
lbpcascade_frontalface	40	14,7
	200	15,8
	450	16,3

(a) Original (b) Haar_default (c) Haar_alt2 (d) LBP

Figura 43 Ejemplo detección cara utilizando distintos algoritmos

Como se puede comprobar cada uno de ellos proporciona una rapidez similar independientemente de la iluminación. Sin embargo, la detección utilizando el algoritmo de LBP es considerablemente más rápida que las detecciones con Viola&Jones. Además, ofrecen una detección parecida (ver Figura 43). Debido a estas razones, y a diferencia del sistema implementado en Matlab, el algoritmo elegido ha sido el LBP. En la se muestra un ejemplo de la detección realizada por la aplicación utilizando el método escogido.

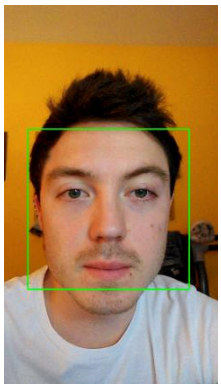


Figura 44 Ejemplo cara detectada

[1]Lx: el lux es la unidad del Sistema Internacional para medir luminancia o nivel de iluminación

Pre procesado

La siguiente etapa consiste en la realización de un pre procesado de la imagen. En esta fase se recorta la cara detectada para analizarla de manera independiente y se redimensiona a un tamaño de 300x300 como estaba establecido en la herramienta de Matlab obteniendo como resultado la Figura 45 Ejemplo cara detectada y pre procesada.

Figura 45 Ejemplo cara detectada y pre procesada

La tercera etapa del procesado de la imagen corresponde a la localización de las regiones principales de la cara: ojos, cejas, nariz y boca. Con el fin de lograr este objetivo, primero se localizan los ojos usando un algoritmo de Viola&Jones que realiza un proceso análogo al utilizado en la cara pero en este caso aplicado a la detección ocular.

Tras la importación del algoritmo, como ocurre con la detección facial, OpenCV facilita una función cuyos parámetros se pueden modificar para realizar una búsqueda más precisa. Se aplica la misma configuración de los parámetros exceptuando el valor aplicado al tamaño mínimo y máximo a detectar. En este caso, el tamaño mínimo es 180x45 (60% del ancho de la cara y 15% del alto) y el tamaño máximo 210x60 (70% del ancho de la cara y 20% del alto). Una vez localizados los ojos se divide el área en dos para poder analizar por separado el ojo izquierdo y el ojo derecho.

A partir de la localización de los ojos se determinan el resto de regiones basándose en los valores utilizados en la herramienta de Matlab y en la realización de pruebas con distintas imágenes. Consecuentemente, como segundo paso se localizan las cejas en función del área que encuadra los ojos haciendo coincidir el borde inferior del área de las cejas con la parte superior de los mismos. Asimismo, el ancho del área será mayor que el ancho de los ojos de tal manera que sea capaz de localizar cualquier ceja pero sin llegar a captar parte del pelo del sujeto. La altura sin embargo será la misma que la de los ojos con el fin de poder encontrar también las cejas cuando éstas se eleven. Como ocurre con los ojos, se divide el área también en dos.

Localización regiones

El tercer paso consiste en localizar la región de la nariz. En este caso el ancho de su área será menor que el ancho de los ojos pero la altura será mayor para poder realizar una delimitación lo más ajustada posible. Por último, en cuarto lugar, se determina el área de la boca. Su altura será también mayor que las de los ojos mientras que su ancho será menor. En esta última localización se hace coincidir siempre el borde inferior del área de la boca con el borde inferior de la cara.

En la Figura 46 se puede ver un esquema de la localización de las regiones junto con los valores empleados.

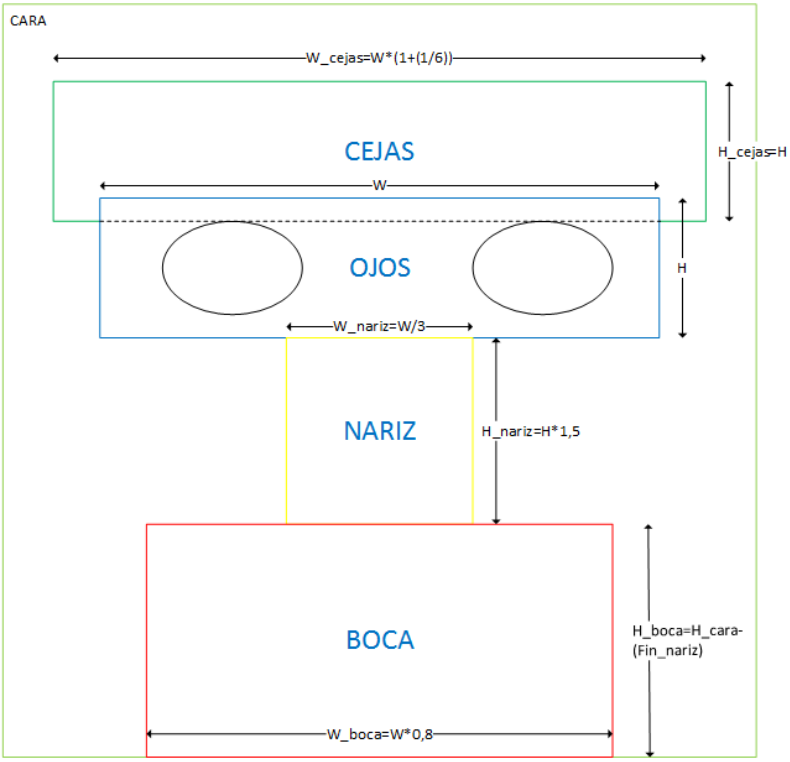


Figura 46 Esquema delimitación regiones

Aplicando esta serie de patrones en la Figura 47 se puede observar un ejemplo de una localización real de las regiones.

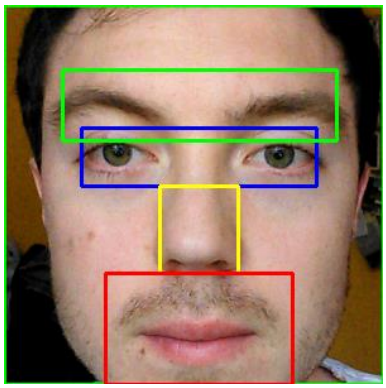


Figura 47 Ejemplo localización real de regiones

Búsqueda de los puntos de interés

La cuarta y última etapa del procesado de la imagen consiste en la búsqueda de los puntos de interés del interior de cada región. En la herramienta de matlab como se ha comentado en el capítulo de la herramienta de referencia existen dos métodos para extraer los puntos de interés pudiendo elegir cualquiera de ellos. En el caso de este proyecto se ha elegido el método más adecuado en función del objeto que se quiere detectar. El proceso que se lleva a cabo en esta etapa en cada uno de los métodos se puede dividir en varias fases representadas en la Figura 48.

En el caso del método de umbralización se realiza en primer lugar una ecualización de la imagen para mejorar el contraste y poder distinguir los objetos (ojo, ceja nariz o boca) con mayor facilidad. En segundo lugar se convierte la imagen de grises a binaria tomando como referencia un determinado umbral. En tercer lugar, se llevan a cabo operaciones morfológicas (como la erosión o la dilatación) con el fin de eliminar ruido y objetos irrelevantes. En cuarto lugar, se detectan los bordes del objeto que se quiere detectar.

En el otro caso, para el método de detección de bordes en primera instancia se difumina la región con el fin de que la imagen sea más uniforme y los objetos no presenten tantos detalles. En segunda instancia se aplica la detección de bordes *Canny* [66]. Una vez detectados los bordes es necesario realizar en tercera instancia operaciones morfológicas por si hubiera algún defecto a la hora de la detección.

El último paso de la búsqueda de los puntos de interés consiste precisamente en la localización de los extremos de los objetos detectados, y en algunos casos es preciso optimizar dicha localización.

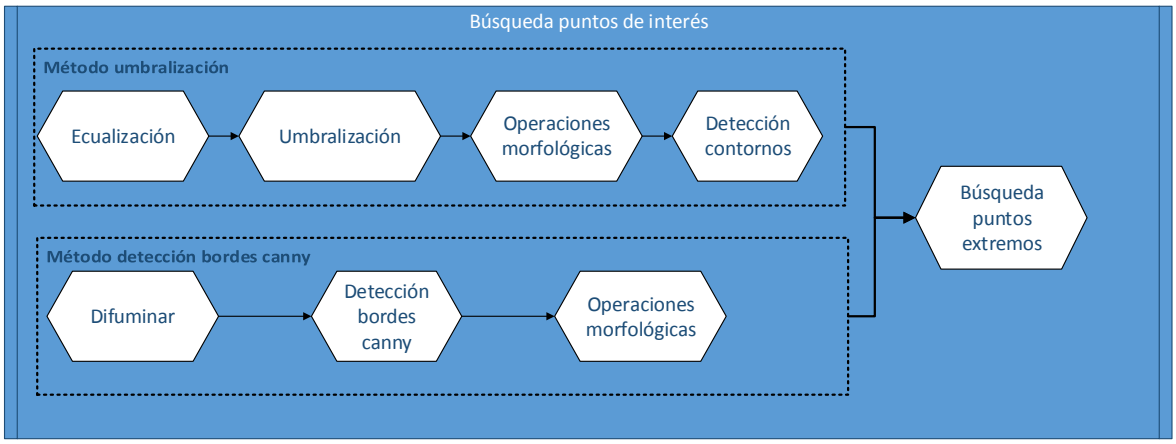


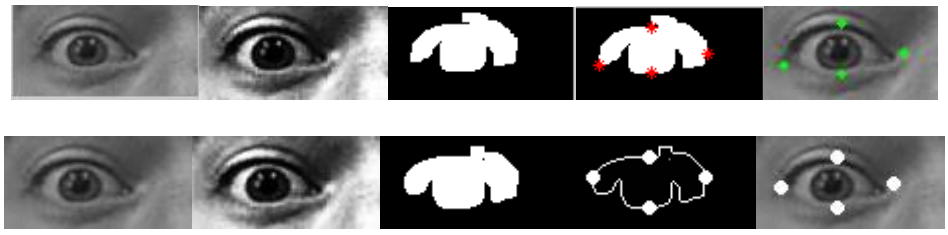
Figura 48 Fases búsqueda de los puntos de interés de cada región

A continuación se expone el proceso detallado de cada una de las regiones de la cara para la búsqueda de los puntos más significativos explicando las diferencias entre cada uno de las etapas y con respecto a la herramienta de Matlab. Como ocurre en el sistema en la que se basa este proyecto se detectan 19 puntos: cuatro en cada uno de los ojos, tres en cada ceja, uno en la nariz y cuatro en la boca.

Para lograr este fin se han realizado pruebas sobre 10 imágenes pertenecientes a la base de datos CK+, introduciéndolas en Matlab y Android. Este conjunto de imágenes contiene sujetos de diferentes nacionalidad, razas y con distintas condiciones de iluminación. En esta memoria se mostrarán las imágenes de cada una de las fases de esta búsqueda de puntos en ambos sistemas correspondientes a un mismo sujeto. También comentarán los problemas y dificultades que pueden encontrarse durante el proceso mostrando un ejemplo ilustrativo.

La primera región que se analiza son los **ojos**. Como se ha comentado anteriormente, la región se divide en dos para tratar cada ojo de forma independiente. Las pruebas se han hecho solamente con un solo ojo puesto que el proceso es idéntico para ambos. En la Figura 49 se puede observar

cada uno de los pasos anteriormente comentados y su comparación con lo realizado por la herramienta de Matlab.



(a) (b) (c) (d) (e)

Figura 49 Comparación proceso detección puntos de interés ojo. Fila superior: Matlab. Fila inferior: Android. (a) imagen del ojo recortada y preparada para ser analizada. (b)Imagen después de ser ecualizada. (c) Imagen después de la umbralización y las operaciones morfológicas. (d) Puntos extremos localizados sobre el contorno (e) Puntos sobre la imagen real del ojo.

En este caso, para los ojos, se ha escogido el método de umbralización como en el caso de la herramienta. La umbralización se basa en convertir la imagen de grises (imagen cuyos píxeles toman valores entre 0 y 255, siendo 0 el color negro y 255 el color blanco) a binaria (píxeles con valores 1 ó 0) en función de un umbral establecido. Aquellos píxeles cuyo valor superen el del umbral se les asociará un valor binario determinado. Para los ojos se ha escogido un umbral de 30, un valor proporcional al utilizado en Matlab (en esta plataforma los valores oscilan entre 0 y 1 en lugar de entre 0 y 255) y aceptado tras la realización de las pruebas correspondientes.

A la de ejecutar las operaciones morfológicas se ha seguido también el mismo procedimiento. Estos pasos se pueden ver ejemplificados en la Figura 50.Prmero se realiza una dilatación para ampliar el objeto (Figura 50(b)), después una combinación de erosión+dilatación (llamado también *opening*. Figura 50(c)) con el fin de eliminar el ruido que pueda haber y por último otra dilatación para volver a ampliar el objeto al poder verse perjudicado en el paso anterior (Figura 50(d)) .

Las diferencias al aplicar este método entre Android y Matlab son principalmente a la hora de introducir los parámetros para la parte morfológica, ya que en Android no se puede introducir por ejemplo el número de píxeles exactos que se quieren eliminar a la hora de hacer el *opening*.

(a) (b) (c) (d)

Figura 50 Operaciones morfológicas aplicadas para localizar los ojos. (a) Imagen tras aplicar la umbralización. (b) Imagen después de aplicar la primera dilatación. (c) Imagen tras aplicar el opening (d) imagen tras aplicar la segunda dilatación. Corresponde a la imagen de la *Figura 49*(c).

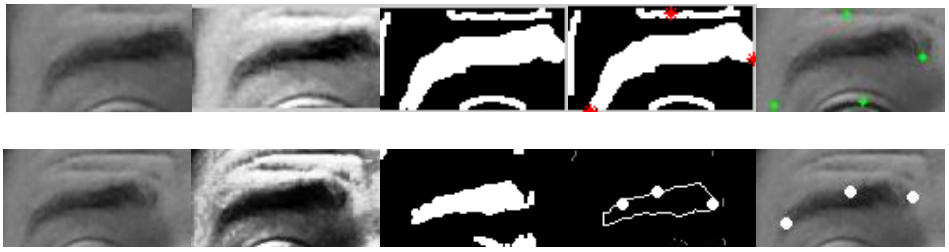
Uno de los problemas que puede surgir a la hora de aplicar este método es que el ojo sea confundido con arrugas o sombras como ocurre en ejemplo de la Figura 51 o de la Figura 52. Este hecho limita la detección correcta de los puntos de interés y genera una dependencia de la iluminación en la cara.

Figura 51 Ejemplo incorrecta detección del punto superior del ojo por la aparición de una arruga. De izq. a dcha.: región original, región tras aplicar umbralización y operaciones morfológicas y región con los puntos detectados.

Figura 52 Ejemplo incorrecta detección punto superior del ojo debido a malas condiciones de iluminación. De izq. a dcha.: región original, región tras aplicar umbralización y operaciones morfológicas y región con los puntos detectados.

La segunda región a analizar son las **cejas**. Como ocurre con los ojos, se divide en dos y las pruebas se realizan solamente sobre una de cejas ya que el procedimiento es igual para ambas. Como se ha comentado anteriormente, la región de las cejas posee un ancho mayor que el de los ojos para poder ser detectadas con la mayor precisión posible.

Para esta región también se ha aplicado el método de la umbralización. En la Figura 53 se puede observar un ejemplo de todo el proceso. El proceso de segmentación en las cejas es similar al ejecutado en los ojos. Se ha utilizado el mismo valor de umbral, pero sin embargo se ha empleado una secuencia diferente de operaciones morfológicas. En este caso se ha aplicado solamente una dilatación y después un *opening*.



(a) *(b)* *(c)* *(d)* *(e)*

Figura 53 Comparación proceso detección puntos de interés ceja. Fila superior: Matlab. Fila inferior: Android. (a) imagen de la ceja recortada y preparada para ser analizada. (b)Imagen después de ser ecualizada. (c) Imagen después de la umbralización/detección de bordes y las operaciones morfológicas. (d) Puntos extremos localizados sobre el contorno (e) Puntos sobre la imagen real de la ceja.

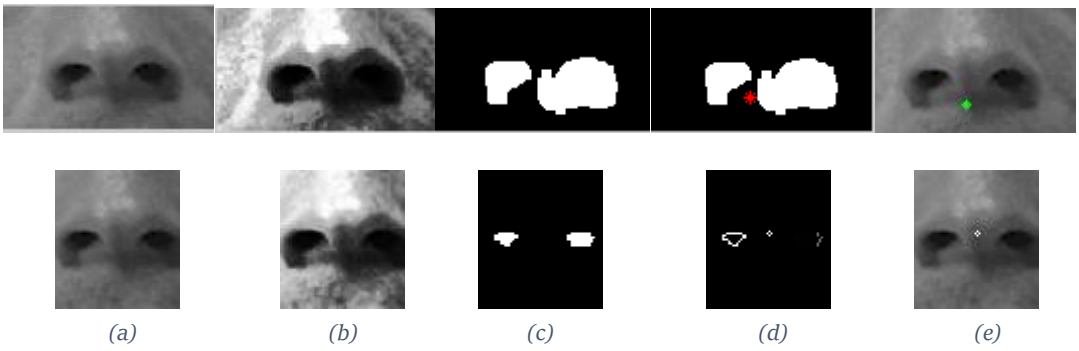
Como se puede observar en la Figura 53 el método empleado en Android es diferente al implementado en Matlab. Mientras que en esta última plataforma se opta por utilizar el método de detección de bordes canny, en este proyecto se utiliza el método de la umbralización. Este método ha sido elegido porque en Matlab se usan funciones morfológicas como *bridge* (función para unir bordes que están separados) o *holes* (función que permite rellenar objetos) no disponibles en Android. Además, el método usado proporciona una detección de puntos más fiable, ya que como se puede comprobar en la Figura 53(e) en Matlab se detectan objetos que no se deberían detectar.

Los problemas que surgen al localizar los puntos en esta región están relacionados con la aparición del cabello sobre la cara. Esto ocurre en ambas plataformas y dificulta el proceso. Además, como ocurre en el caso de los ojos, las arrugas pueden ser otro foco de problemas. En la Figura 54 se muestra un ejemplo de ello.

Figura 54 Ejemplo incorrecta detección puntos ceja debido a arrugas. De izq. a dcha.: región original, región tras aplicar umbralización y operaciones morfológicas y región con los puntos detectados.

La tercera región que se analiza es la **nariz**. La idea consiste en buscar los orificios nasales a través del método de la umbralización. Utilizando un umbral bajo es posible detectarlos gracias a la diferencia que existe con el tono de la piel incluso en personas de raza oscura.

En este caso se utiliza un umbral muy bajo, 8, para poder distinguir con mayor facilidad los orificios. Tras este paso se aplica una operación combinada de erosión+dilatación (también llamada *closing*) para rellenar los posibles huecos que pudieran existir. Después, el punto de interés de la nariz se obtiene como el punto medio entre los orificios detectados.



(a) *(b)* *(c)* *(d)* *(e)*

Figura 55 Comparación proceso detección puntos de interés de la nariz. Fila superior: Matlab. Fila inferior: Android. (a) Imagen de la nariz recortada y preparada para ser analizada. (b)Imagen después de ser ecualizada. (c) Imagen después de la umbralización/detección de bordes y las operaciones morfológicas. (d) Punto localizado (e) Punto sobre la imagen real de la nariz.

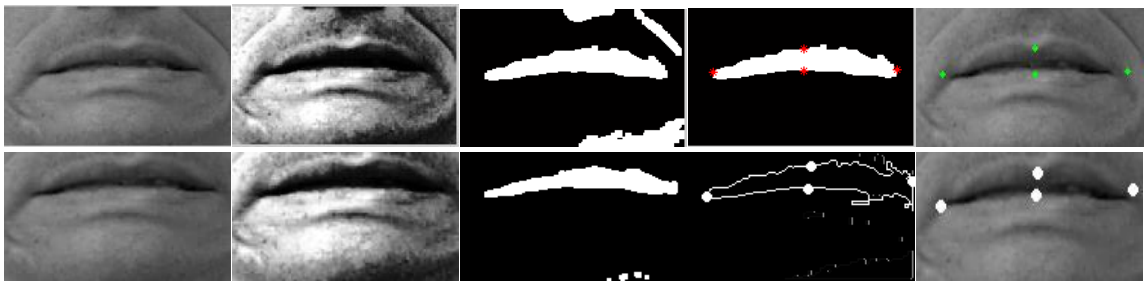
Comparando la detección de puntos realizada por las dos herramientas en la Figura se pueden observar diversas diferencias. La primera de ellas está relacionada con el tamaño de la región. En Matlab el ancho es mayor y esto puede suponer detectar arrugas indeseadas. En Android sin embargo se ajusta la región al ancho de la nariz para evitar este posible caso y mejorar la rapidez y precisión a la hora de localizar los orificios nasales.

La segunda diferencia se encuentra en las operaciones morfológicas realizadas durante la segmentación. En la herramienta de Matlab se aplican tres operaciones morfológicas para detectar los orificios mientras que en este proyecto se aplica solamente un *closing*.

La última región que se analiza es la **boca**. De las cuatro regiones que se localizan esta es la que más problemas genera puesto que supone una dificultad mayor para encontrar un patrón para poder detectarla. En este proyecto se han realizado varias aproximaciones. En primer lugar se intentó realizar una segmentación por color de tal manera que se pudiera detectar la boca a partir de los tonos rojizos de la misma pero se descartó por no aportar los resultados esperados.

Una segunda aproximación fue intentar aplicar el mismo método utilizado en Matlab. Al realizar las pruebas sobre las imágenes preseleccionadas de la base de datos CK+ todo parecía correcto y la detección era precisa como se puede comprobar en la Figura 56. Sin embargo, generaba problemas cuando la boca se abría o presentaba malas condiciones de iluminación. Esto se puede

comprobar en el ejemplo de la Figura 57. La solución a esto fue intentar realizar otra detección como tercera aproximación aplicando el método de bordes *Canny*. Al aplicar este método sobre la misma imagen que generaba problemas se corrigen los problemas que surgían como se muestra en la Figura 57.



(a) *(b)* *(c)* *(d)* *(e)*

Figura 56 Comparación proceso detección puntos de interés de la boca. Fila superior: Matlab. Fila inferior: Android. (a) Imagen de la boca recortada y preparada para ser analizada. (b)Imagen después de ser ecualizada. (c) Imagen después de la umbralización/detección de bordes y las operaciones morfológicas. (d) Puntos localizados (e) Punto sobre la imagen real de la boca.

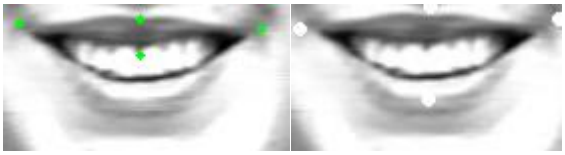


Figura 57 Comparativa detección de puntos de la boca entre Matlab (izquierda) y Android (derecha) en una imagen de la base de datos CK+

Además, se han realizado pruebas de detección con imágenes en tiempo real obteniendo los mismos problemas que los encontrados en las imágenes de la base de datos analizada. Al aplicar el método de detección de bordes *Canny* se solventan estos problemas y se detectan los puntos con mayor precisión como se puede ver en la Figura 58.

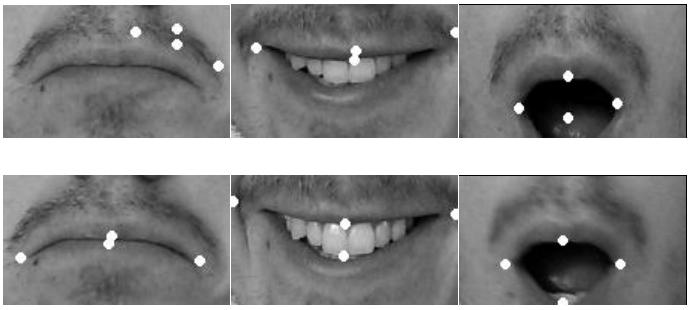


Figura 58 Comparativa detección de puntos de la boca entre Matlab (fila superior) y Android (fila inferior)

Una vez analizadas las cuatro regiones para la detección de los puntos de interés el resultado sería el obtenido en la Figura 59 para la imagen de la base de datos utilizada para la explicación de los métodos aplicados. La Figura 60 muestra el resultado en una imagen en tiempo real.

Figura 59 Puntos detectados Matlab (izq.) y Android (derecha) en una imagen de la base de datos CK+

Figura 60 Puntos detectados Matlab (izq.) y Android (derecha) en una imagen en tiempo real

Como se ha comentado anteriormente este proceso es realizado tanto para la imagen con expresión neutral como para la imagen con la emoción a detectar. Una vez detectados los puntos en ambas imágenes el siguiente paso realizado, como se indica en el esquema general de la Figura 36 Esquema general reconocimiento facial de emociones consiste en comparar los puntos localizados en cada imagen con el fin de detectar las unidades de acción presentes. Este paso intermedio antes de clasificar la emoción se ha introducido brevemente en el Estado del arte.

4.2.3 Detección unidades de acción

La idea es detectar los cambios musculares que se producen en la cara con el fin de facilitar la tarea al clasificador para reconocer la emoción puesto que cada emoción posee unas expresiones faciales distintas e involucran músculos diferentes. En función de las unidades de acción presentes en una imagen a analizar el clasificador decidirá reconocer una emoción u otra.

La herramienta de Matlab en la que se ha basado este trabajo contenía un primer acercamiento a las unidades de acción, pero al mismo tiempo que se realizaba este proyecto se ha realizado una segunda versión de dicha herramienta. Esta versión ha sido realizada por A.Gil [67] en su proyecto final de grado “Integración de algoritmos basados en Unidades de Acción en una herramienta de análisis de reconocimiento de emociones”. En él se ha realizado un estudio en profundidad para mejorar la integración de las AUs en el proceso del reconocimiento facial de emociones. En este proyecto se ha intentado trabajar siempre basándose en la última versión de la herramienta desarrollada en Matlab por lo que en este caso se ha incluido también esta nueva aproximación sobre las unidades de acción.

En la investigación llevada a cabo en paralelo se realiza un análisis de las AUs con mayor importancia y sus posibles combinaciones a la hora de reconocer una emoción. En primer lugar se han estudiado las publicaciones que informan sobre las unidades de acción influyen en cada emoción y de qué manera. En segundo lugar se ha elaborado un análisis subjetivo análogo observando imágenes de la base de datos CK+. Con el fin de llegar a una conclusión final, se han ejecutado en ambos estudios los siguientes pasos para cada emoción:

1. Registrar todas las unidades de acción que pueden estar presentes.
2. Anotar las posibles combinaciones entre ellas.
3. Realiza un proceso estadístico para detectar qué AUs tienen mayor relevancia.
4. Descartar las AUs con menor relevancia.
5. Eliminar aquellas que no se puedan detectar con los 19 puntos detectados.
6. Volver a apuntar las posibles combinaciones tras haber realizado los pasos 3,4 y 5.

Una vez registradas las combinaciones posibles para los dos estudios se ha llevado a cabo un análisis final comparando ambos y generando combinaciones finales. Tras el estudio completo y en una primera aproximación se ha decidido que las AUs con mayor importancia para la reconocer cada una de las emociones son las que se exponen en la Tabla 5 y las que se han utilizado en este proyecto. Como se puede observar ciertas unidades de acción pueden ser combinadas entre ellas además para crear CAUs (*Combinated Action Units* o Unidades de Acción Combinadas). Una CAU es el conjunto de dos o más AUs que forman parte de la misma expresión facial y se producen al mismo tiempo en alguna emoción. Por ejemplo, la CAU2 es la combinación de las AUs 9 y 10 porque siempre que se expresa una emoción de asco siempre se frunce el ceño y se elevan los labios.

Tabla 5 Selección de AUs y CAUs [67]

	AU	CAU	Descripción de la AU	Imagen
Parte posterior del rostro	1	1	Interior de las cejas elevado	
	2		Exterior de las cejas elevado	
	5		Párpado superior elevado	
	4		Cejas bajadas	
	6		Mejillas elevadas	
	7		Párpados tensos	
Parte inferior	9	2	Nariz arrugada	
	10		Labio superior elevado	

	12		Comisuras de los labios elevados	
	15	3	Comisuras de los labios hacia abajo	
	17		Barbilla elevada	
	23	4	Labios tirantes, tensos	
	24		Labios presionados	
	26		Boca entreabierta	
	27		Boca abierta	
	H		Boca sonriente abierta	

Una vez seleccionadas las AUs y CAUs presentes en las emociones a reconocer se puede proceder a la detección de las mismas. Este proceso se ha realizado en paralelo en Matlab y Android y en ambas plataformas puede ser origen de problemas. ¿Cuánto tienen que subir las cejas para determinar que las cejas se han elevado?, ¿Cuánto debe abrirse la boca para decidir que corresponde a una expresión de sorpresa? Estas preguntas son difíciles de responder pero el sistema debe ser lo suficientemente preciso como para distinguir las unidades de acción entre ellas.

Por este motivo se han diseñado e implementado un conjunto de criterios con el fin de poder determinar con la mayor certeza posible si una AUs está presente. Para la detección se pueden usar descriptores como áreas, distancias entre puntos, vectores de movimiento o ángulos. En este caso se han utilizado solamente los tres primeros. Las áreas utilizadas son las representadas en la Figura 61 y se describen de la siguiente manera:

- LU (left-up), RU (right-up): triángulo superior del ojo izquierdo y derecho, formados por los puntos (7, 8, 9) y (11, 12, 13) respectivamente.
- LD (left-down), RD (right-down): triángulo inferior del ojo izquierdo y derecho, formados por los puntos (7, 9, 10) y (11, 13, 14) respectivamente.
- LI (left-inner), RI (right-inner): triángulo interior del ojo izquierdo y derecho, puntos (3, 8, 9) y (4, 11, 12) respectivamente.
- UL (up-left), UR (up-right): triángulo superior de la boca para el lado izquierdo y derecho de la misma, formado por los puntos (15, 16, 19) y (16, 17, 19) respectivamente.

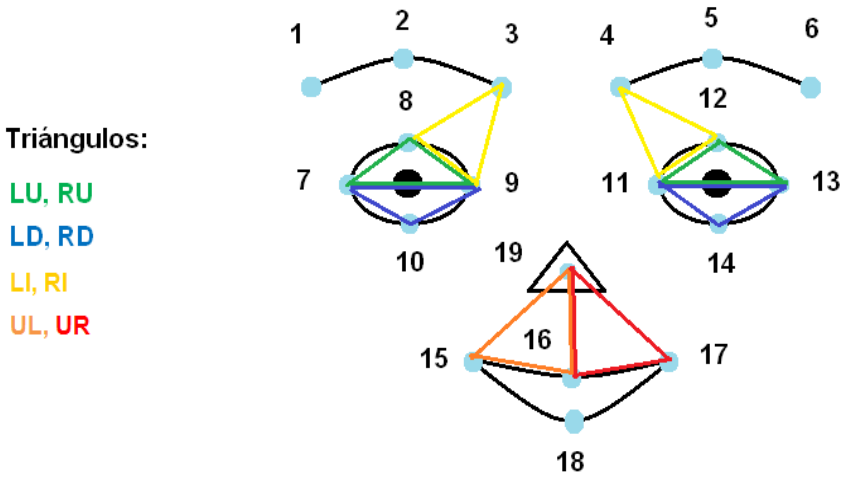


Figura 61 Triángulos empleados para la detección de las AUs [67]

Un ejemplo de aplicación puede ser la detección de la AU4, en la cual el área del triángulo LI y RI debe ser menor en la emoción a reconocer que en la imagen con expresión neutral ya que la parte interior de las cejas se junta con los ojos.

En la Figura 61 se puede observar además la numeración de los puntos localizados. Un ejemplo de detección de AUs aplicando distancia entre puntos como descriptor puede ser la CAU3. Esta CAU está presente siempre y cuando los puntos 15 y 17 se sitúen por debajo del punto 18 ya que indicará que el sujeto tendrá las comisuras de los labios bajadas. Por último, los vectores de movimiento se emplean comparando el movimiento de puntos entre la imagen con emoción y la imagen neutral. Un ejemplo de ello podría ser la detección de la CAU5, en la cual los puntos 16 y 18 deben desplazarse y alejarse uno del otro.

Las combinaciones finales de estas AUs y CAUs resultado del estudio realizado por A.Gil se describen en la Tabla 6.

Tabla 6 Combinaciones finales tras el estudio completo [67]

	Ira	Asco	Miedo	Felicidad	Tristeza	Sorpresa
Combinaciones finales	AU4+CAU4	AU4+CAU2	AU4+AU27	AU6+AU12	AU4+CAU3	CAU1+CAU5
	AU4+AU7+CAU2	CAU2	CAU1+AU26	AU6+AUH	AU6+CAU3	CAU1
	AU4+AU7+CAU4	AU4+CAU2+ CAU3	AU26	AU4+AU6+ AU12		CAU5
	AU4+AU7+CAU5	CAU2+CAU3		AU4+AU6+ AUH		
	AU4+AU7			AU12		
	CAU4			AUH		

Este conjunto de combinaciones es el que usará el clasificador en el apartado siguiente para realizar el entrenamiento. Como se ha comentado en el apartado de clasificadores del estado del arte es necesario un entrenamiento previo antes de que el clasificador sea capaz de reconocer una emoción.

4.2.4 Clasificador

El último paso para reconocer una emoción tras detectar las unidades de acción es aplicar un clasificador capaz de reconocer la emoción a detectar. Para que este clasificador pueda realizar dicha tarea es necesario realizar que realice un aprendizaje previo.

El clasificador elegido para este proyecto ha sido el árbol de decisión por ser el utilizado en la herramienta de Matlab, pero al haber realizado una implementación modular en Android se ha incluido la opción también de usar otros clasificadores como ANN, SVM o redes bayesianas. En principio sólo se aplicará el método de clasificación elegido, pero en cualquier caso como trabajo futuro siempre queda la oportunidad de realizar una comparación entre los diversos métodos posibles a utilizar. Estos métodos se han integrado en el proyecto gracias a la librería de WEKA [68], una plataforma especializada en módulos de aprendizaje y que facilita una serie de funciones para poder utilizar estos clasificadores.

WEKA propone una solución llamada J48, una implementación de código abierto desarrollada para java para la creación de árboles de decisión. Se basa en el algoritmo C4.5 [69] , que es una extensión del algoritmo ID3 descrito en el capítulo del estado del arte. Algunas de las mejoras que se introducen en este algoritmo son:

- ✓ Mejora en cuanto a rapidez y uso de memoria
- ✓ Manejo de datos de entrada con valores de atributos desconocidos. Estos valores simplemente no se utilizan en los cálculos de entropía y ganancia de información.
- ✓ Asignación de diferentes costes a cada uno de los atributos.
- ✓ Poda del árbol de decisión. Una vez creado, el algoritmo C4.5 recorre el árbol desde las hojas hasta el nodo raíz eliminado los atributos que no ayudan.

Asimismo, esta librería permite configurar distintos parámetros a la hora de generar el árbol de decisión final. Por ejemplo, se pueden modificar las siguientes variables:

- **Árbol sin poda**: permite crear el árbol sin eliminar atributos que no aportan información
- **Confianza de poda**: se puede establecer un determinado umbral de confianza (por defecto 0,25)
- **Número mínimo de muestras**
- **Divisiones binarias**: si esta opción está activada los nodos aplicarán solamente divisiones binarias
- **Número de particiones** del conjunto de muestras. Con el fin de comprobar la precisión del árbol de decisión generado se puede utilizar la técnica de la validación cruzada. que consiste en utilizar una parte del conjunto total para entrenar y crear el árbol y el resto para poner a prueba ese árbol creado. La idea consiste en dividir el conjunto de datos de entrada en K subdivisiones, de tal manera que se utiliza una subdivisión para pruebas y K-1 subdivisiones para el entrenamiento. Este proceso es repetido K veces, una vez por cada subdivisión. En la Figura 62 se muestra un esquema de este proceso si por ejemplo K=10.

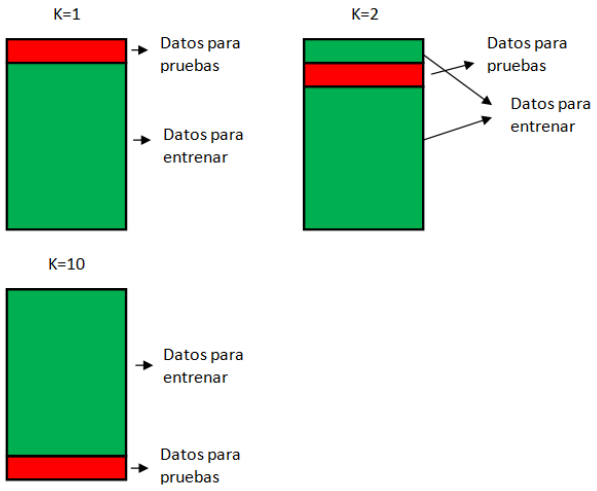


Figura 62 Proceso validación cruzada

En este caso, el aprendizaje se completa realizando un entrenamiento supervisado, es decir, a partir de un conjunto de ejemplos ya clasificados. Concretamente, el clasificador recibe como entrada diversas muestras con una serie de atributos y, conociendo la emoción de dichas muestras, el árbol aprende y elabora una serie de reglas y decisiones en función de la información recibida.

En esta ocasión, el grupo de ejemplos de entrenamiento son las combinaciones finales obtenidas en el apartado anterior. Estas combinaciones se expresan en forma de la matriz de tal manera que el clasificador sea capaz comprender las muestras de entrada. Las filas de esa dicha matriz son cada una de las posibles combinaciones para cada emoción mientras que las columnas se componen de todos atributos que pueden poseer las muestras. Es importante que también exista una columna con la emoción de cada una de las filas para que el aprendizaje sea supervisado. Un ejemplo de matriz de entrenamiento se encuentra en la Tabla 7. En ella se puede observar que los atributos son las AUs o CAUs que se han elegido como relevantes.

La primera fila correspondería a la primera muestra. Cada atributo, desde la CAU1 hasta la AUH, posee un valor determinado (en esta herramienta siempre serán valores binarios). En este ejemplo, la combinación estaría formada por la CAU4 y la AU4. En la última columna se coloca la emoción que se detecta con esta combinación. El resto de la matriz seguiría el mismo patrón hasta completar todas las combinaciones posibles.

Tabla 7 Ejemplo matriz de entrenamiento

Combinaciones	AUs/CAUs								Emoción real
	CAU 1	CAU2	CAU 3	CAU 4	CAU 5	AU 4	AU 6	…	
	0	0	0	1	0	1	0		Ira
	0	1	0	0	0	1	0		Asco
	0	0	0	0	1	1	0		Miedo
	0	0	0	0	0	0	1		Felicidad
	0	0	1	0	0	1	0		Tristeza
	1	0	0	0	1	0	0		Sorpresa
	…								…

En este caso los atributos son las unidades de acción, por lo que el algoritmo debe decidir en primer lugar cuál es la AU que aporta mayor relevancia a la hora de clasificar una emoción.

Como primera aproximación, la matriz resultante al aplicar las combinaciones finales conseguidas posee 30 filas (30 posibles combinaciones) y 11 columnas (11 AUs o CAUs). El árbol de decisión que se obtiene con la información aportada por dicha matriz es el que ilustra la Figura 63.

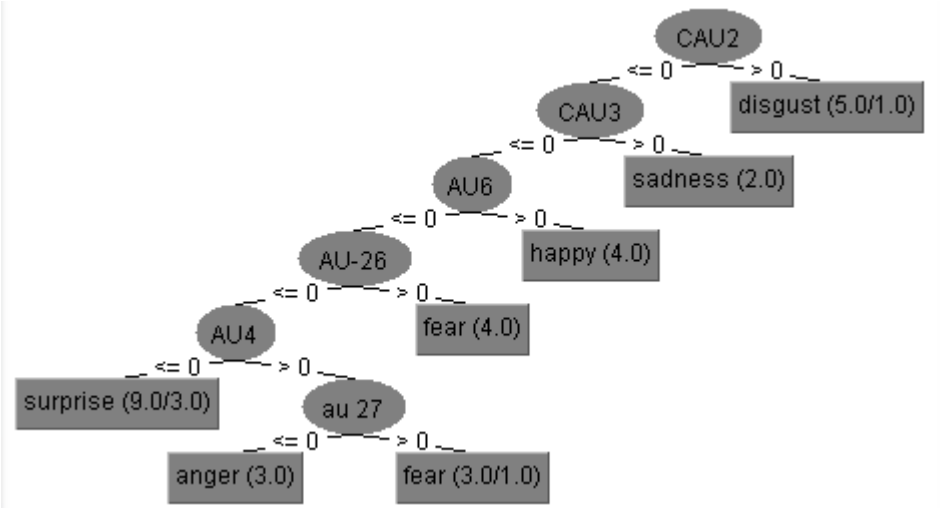


Figura 63 Árbol de decisión resultante

Al observar el árbol de decisión creado se comprueba que el algoritmo decide que el atributo con mayor ganancia de información es la CAU2, seguido de la CAU3, continuando con la AU6 y así hasta la AU27. La primera pregunta que surge al ver este árbol es, ¿Qué ha ocurrido con el resto de atributos? La respuesta es simplemente que el algoritmo determina que con esos atributos es suficiente para clasificar una emoción y que los ausentes no aportan la información precisa como para ser decisivos. Otra duda que puede producirse es ¿Será el árbol capaz de realizar una correcta clasificación teniendo en cuenta la escasa redundancia a la hora de decantarse por una emoción u otra? La contestación a esta cuestión es que sí debería. Precisamente, uno de los objetivos de este clasificador consiste en llegar a la solución final por el camino más corto posible y así obtener la emoción con mayor brevedad.

.Un árbol de decisión requiere un conjunto amplio de muestras para poder ser creado contemplando todas las posibles combinaciones de atributos que puedan llegar a existir. Utilizar una cantidad de muestras por encima de lo debido puede provocar un sobre entrenamiento de este árbol y desencadenar en una incorrecta clasificación. De hecho, el árbol de decisión trata de evitar estos posibles problemas llevando a cabo técnicas de recorte (*pruning*) que eliminan ramas del árbol para evitar decisiones que no tengan suficiente evidencia, pero usar un conjunto pequeño de muestras también puede llevar a una mala clasificación al no aportar suficientes datos. En el capítulo de resultados se mostrarán ejemplos que harán plantearse si la matriz de entrenamiento elaborada es la más adecuada para este clasificador.

Una vez que el clasificador ha realizado el aprendizaje está preparado para ejecutar la clasificación de la emoción. Esa emoción a reconocer se introduce en el al árbol como una nueva muestra con

un valor determinado en cada uno de los atributos. En función de dichos valores se recorre el árbol de decisión por unas ramas u otras hasta llegar a la decisión final. Se procede a analizar un ejemplo para un mejor entendimiento. Tras aplicar el procesado de la imagen neutral del sujeto y de esta imagen con emoción se han detectado las siguientes AUs y CAUs utilizando el algoritmo explicado en el apartado anterior:

CAU1	CAU2	CAU3	CAU4	AU27	AU4	AU6	AU7	AU12	AUH	AU26	Emoción
1	0	0	0	0	0	0	0	0	0	1	¿?

Fijándose en el árbol creado en la Figura 63, a la hora de analizar esta nueva muestra el primer atributo que se evalúa es la CAU2. Si es '1' la emoción detectada será de asco. Como es '0' se desciende el árbol por otra rama y se procede a analizar otro atributo como se muestra en la Figura 64.

Figura 64 Primera decisión del árbol. Nodo raíz

El siguiente que se analiza es la CAU3. Si éste es '0' la emoción detectada es tristeza, pero como es '1' se sigue descendiendo el árbol hasta encontrarse con el siguiente atributo como se indica en la Figura 65.

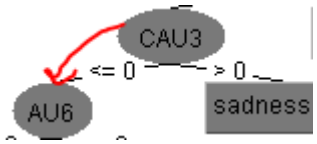


Figura 65 Segunda decisión del árbol. Desciende por la rama de la izquierda

El siguiente atributo es la AU6. Como no es '1' no se puede determinar que la emoción es felicidad y por tanto hay que continuar el análisis (Figura 66). El próximo atributo que se examina es la AU26.

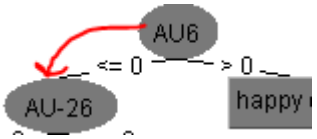


Figura 66 Tercera decisión del árbol.

En este caso, como el valor es '1', ya es posible llegar a una solución final y decidir que la emoción a detectar es miedo como se aprecia en la Figura 67

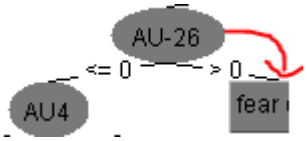


Figura 67 Ultima decisión del árbol. Emoción reconocida: fear (MIEDO)

CAU1	CAU2	CAU3	CAU4	AU27	AU4	AU6	AU7	AU12	AUH	AU26	Emoción
1	0	0	0	0	0	0	0	0	0	1	MIEDO

De esta manera, siguiendo este procedimiento se puede evaluar el árbol de decisión y comprobar si realmente ha realizado la clasificación correctamente. En el siguiente capítulo de Pruebas y resultados se lleva a cabo este proceso. Sabiendo previamente cuál es la emoción que debe reconocer el árbol, se realizarán pruebas con el fin de verificar si la emoción ha sido correctamente reconocida.

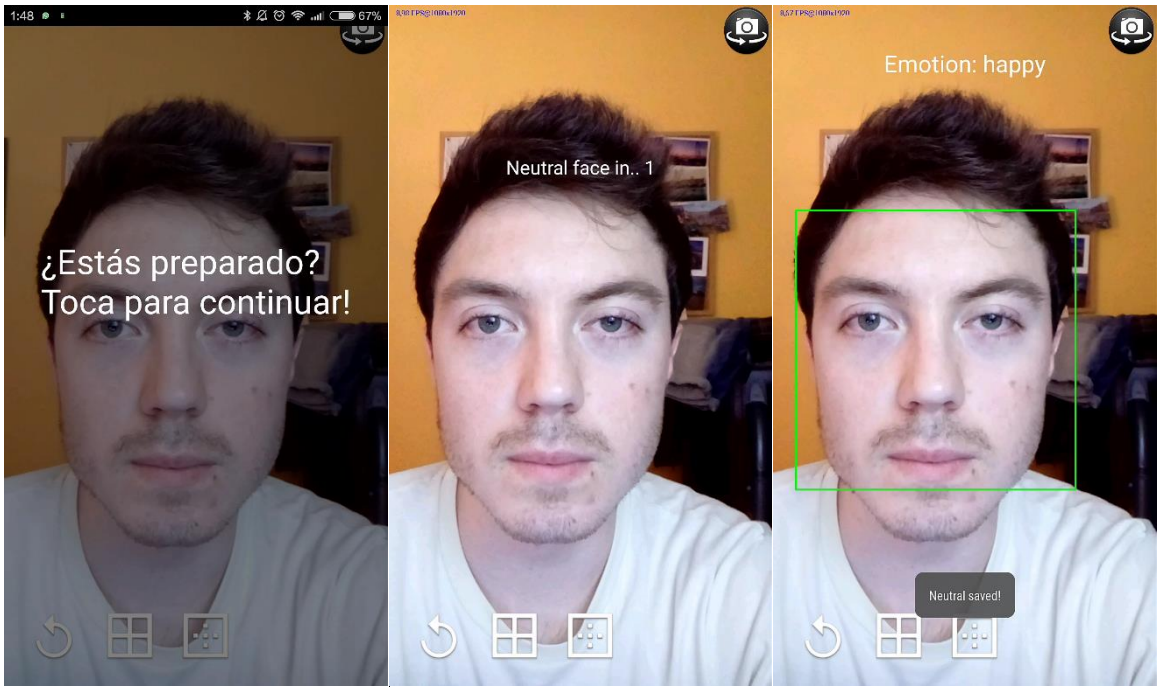
5.Pruebas y resultados

A lo largo de la sección anterior se han ido evaluando las distintas partes del procesado de una imagen. Ahora, en este capítulo se muestran los resultados de las pruebas realizadas una vez concluida la aplicación.

En primer lugar se expondrán capturas del proceso de reconocimiento facial que realiza la aplicación, desde que se inicia hasta que finalmente reconoce la emoción además de enseñar las opciones adicionales que se han incorporado. Posteriormente se adjuntarán los resultados de la velocidad de reconocimiento y de la precisión a la hora de detectar una emoción para distintas condiciones de iluminación. Por último, se hará experimentará el reconocimiento facial de emociones en condiciones extremas.

5.1 Proceso de reconocimiento en la aplicación desarrollada

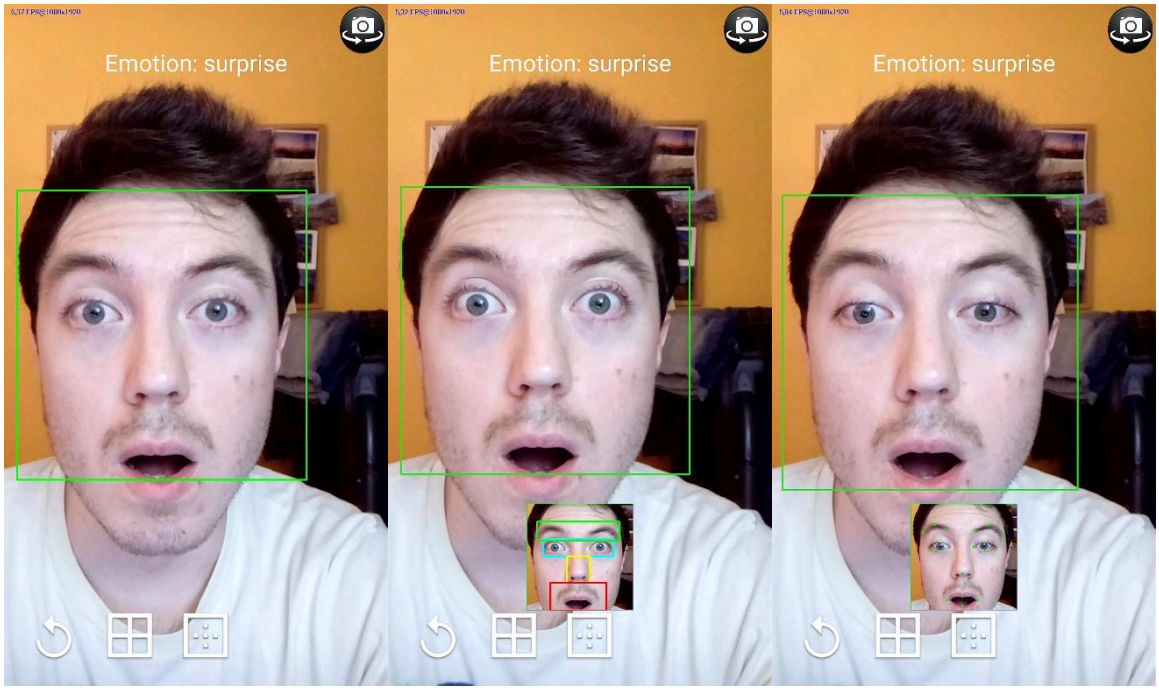
Nada más arrancar la aplicación salta una pantalla de diálogo como el de la Figura 68(a) para esperar a que el usuario se prepare. Para quitar esta ventana basta con toca en cualquier parte de ella y desaparecerá empezando justo después un contador descendente de tres segundos como el de la Figura 68(b). Cuando el contador llegue a cero se capturará la imagen con expresión neutral y se mostrará un mensaje de que se ha guardado (Figura 68(c)).



(a) (b) (c)

Figura 68 Pantallas aplicación. Parte 1. (a) Fase inicial de espera (b) Contador descendente (c) Imagen neutral guardada

Una vez almacenada la neutral, la aplicación ya puede proceder a reconocer la emoción que se desee. En la pantalla se muestra un recuadro verde que muestra la detección de la cara para que sirva de guía. Además, se mostrará la emoción reconocida pudiendo ver fácilmente si se detecta correctamente. Esto se puede ver en la Figura 69(a). Por último, en cualquier momento del reconocimiento se pueden ver las regiones o los puntos localizados de la cara como se muestra en las Figura 69(b) y Figura 69(c). Asimismo, también existe la opción de reiniciar el proceso de nuevo pinchando en el botón de reinicio.



(a) *(b)* *(c)*

Figura 69 Pantallas aplicación. Parte 2. (a) Emoción reconocida (b) Ver regiones (c) Ver puntos

5.2 Pruebas velocidad y precisión

En este apartado se evalúa la rapidez a la hora de reconocer una emoción y la precisión del reconocimiento. Se realizan pruebas para iluminaciones menores de 100lx, para iluminaciones entre 100lx y 1000lx y para iluminaciones mayores de 1000lx.

Caso 1: Iluminación menor que 100lx

Tabla 8 Velocidad media para cada emoción caso 1

Emoción	Velocidad media (ms)
Ira	100,75
Asco	105,90
Miedo	103,56
Felicidad	99,67
Tristeza	86,44
Sorpresa	85,51
	96,97

Observando la Tabla 8 se puede comprobar que la velocidad es parecida para todas las emociones salvo pequeñas diferencias. Estas diferencias son justamente en las emociones más difíciles de reconocer al presentar mayores dificultades en la localización de los puntos de la boca por lo que este último motivo puede ser la causa de esa desviación.

En este caso, para realizar las pruebas de precisión se han establecido las condiciones ideales de iluminación y de exposición del sujeto, logrando así altos porcentajes en casi todas las emociones. Las emociones más distintivas son felicidad tristeza y sorpresa puesto que la expresión de la cara de cada una de ellas es prácticamente única y no suelen generar confusiones si las AUs y CAUs son detectadas correctamente. En la Tabla 9 se puede ver un resumen de la precisión de cada emoción junto con una media de la precisión general de la aplicación para estas condiciones.

Tabla 9 Precisión reconocimiento de emoción caso 1

Emoción	Precisión (%)
Ira	61,9
Asco	58,6
Miedo	75,0
Felicidad	79,6
Tristeza	75,0
Sorpresa	93,4
	73,9

Las emociones que poseen un mayor porcentaje de acierto son las tres mencionadas junto con miedo. Esta última emoción es reconocida correctamente en gran parte por la detección precisa de la AU26 que permite que se distinga de otras emociones como sorpresa. Fijándose en el árbol creado de la Figura 63 se comprende la importancia de esta AU para miedo. Si esta unidad de acción no es detectada correctamente lo más probable es que no se detecte la emoción.

Las emociones de ira y asco tienen porcentajes menores debido a que son confundidas a veces una con la otra, ya que la expresión de la parte superior del rostro es similar y es complicado distinguir la parte inferior.

Caso 2: Iluminación entre 100lx y 1000lx

En el caso 2 se han realizado pruebas en condiciones reales para saber cuál es la precisión verídica de la aplicación. Respecto a la velocidad, como en el caso anterior, analizando la Tabla 10 todas las emociones proporcionan un tiempo de reconocimiento similar y no se aprecian diferencias destacables.

Tabla 10 Velocidad media para cada emoción caso 2

Emoción	Velocidad media (ms)
Ira	92,32
Asco	109,28
Miedo	87,9
Felicidad	104,34
Tristeza	94,16
Sorpresa	113,23
	100,21

En cuanto a la precisión ahora en la Tabla 11 los porcentajes no son tan elevados al no haber condiciones ideales y por tanto presentar problemas a la hora de reconocer algunas emociones. La emoción de sorpresa cuenta con el porcentaje más alto ya que si se detectan sus unidades de acción características el árbol de decisión determina la reconoce correctamente. La emoción de ira aunque su porcentaje sea algo mayor que para el caso ideal normalmente es confundida con felicidad por el hecho de que se detecta la AU6 en lugar de la AU7 (dos unidades de acción muy similares) y el árbol de decisión considera más importante la primera de ellas o con asco por lo comentado en el caso 1.

Precisamente la emoción de asco baja su porcentaje por este motivo. Es confundida frecuentemente con ira ya que si el algoritmo no es capaz de detectar el CAU2 no hay otra posibilidad para detectar esta emoción. A su vez, esta CAU puede ser detectada a veces en miedo si el sujeto presenta una expresión de la boca que haga que se eleve y provoque una confusión. El porcentaje en medio ha descendido en parte por este motivo.

En cuanto a la emoción de tristeza el descenso de la precisión tiene que ver con otro problema como la detección de los puntos de la boca. En el capítulo de la implementación ya se han comentado estas dificultades.

Tabla 11 Precisión reconocimiento de emoción caso 2

Emoción	Precisión (%)
Ira	63,8
Asco	31,1
Miedo	43,8
Felicidad	70,7
Tristeza	56,3
Sorpresa	83,3
	58,2

Caso 3: Iluminación mayor que 1000lx

En este último caso se han realizado pruebas en la calle a plena luz del día. Aunque este hecho haga pensar que al haber mayor luminosidad el reconocimiento será mejor no es cierto, ya que la cara estará sobre iluminada y por tanto apenas habrá contrastes entre las diferentes regiones de

la cara por lo que puede resultar más complicado detectar los puntos de interés. En la Tabla 12 se muestran las velocidades de reconocimiento obtenidas y se puede recalcar que ha descendido respecto a los anteriores casos debido en gran parte a la rapidez con la que se detecta la cara.

Tabla 12 Velocidad media para cada emoción caso 3

Emoción	Velocidad media (ms)
Ira	89,36
Asco	94,38
Miedo	104,44
Felicidad	97,78
Tristeza	98,16
Sorpresa	97,7
	96,97

En cuanto a la precisión obtenida bajo estas condiciones en general descienden todos los porcentajes comparados con los casos predecesores. La emoción de ira es confundida de nuevo con asco y en este caso es más notable debido a la peor detección de los puntos de interés.

La emoción de miedo ha subido en esta prueba puesto que en la mayoría de veces ha reconocido correctamente la AU26 y por tanto es la única emoción posible con esta unidad de acción. Sin embargo, una emoción que ha descendido en gran medida es felicidad. Este caso es importante comentarlo ya que aquí pueden verse algunas de las carencias relacionadas con el árbol de decisión creado. Por error se detecta la CAU1 a veces, un atributo más propio de emociones como miedo o sorpresa y esto hace que a pesar de reconocer correctamente la AU12 o la AUH (unidades de acción características de felicidad) la emoción reconocida sea errónea ya que felicidad sólo será reconocida si la AU6 está activa (ver árbol de decisión creado en la Figura 63).

Por último, sorpresa también pierde precisión por una cuestión parecida al caso de felicidad. Si la boca no es correctamente localizada es posible que el punto inferior de la boca se detecte de manera incorrecta y produzca que se detecten unidades de acción como la CAU3 que provoque se en lugar de reconocer sorpresa reconozca tristeza.

Tabla 13 Precisión reconocimiento de emoción caso 3

Emoción	Precisión (%)
Ira	55,9
Asco	55,0
Miedo	65,5
Felicidad	25,0
Tristeza	49,1
Sorpresa	72,0
	58,25

Conclusiones de pruebas de iluminación

Observando los valores medios de la velocidad en la gráfica resumen de la Figura 70 se puede concluir que la iluminación no condiciona en la rapidez de la detección, ya que para todos los casos se sitúa en valores entre 80 y 100 milisegundos.

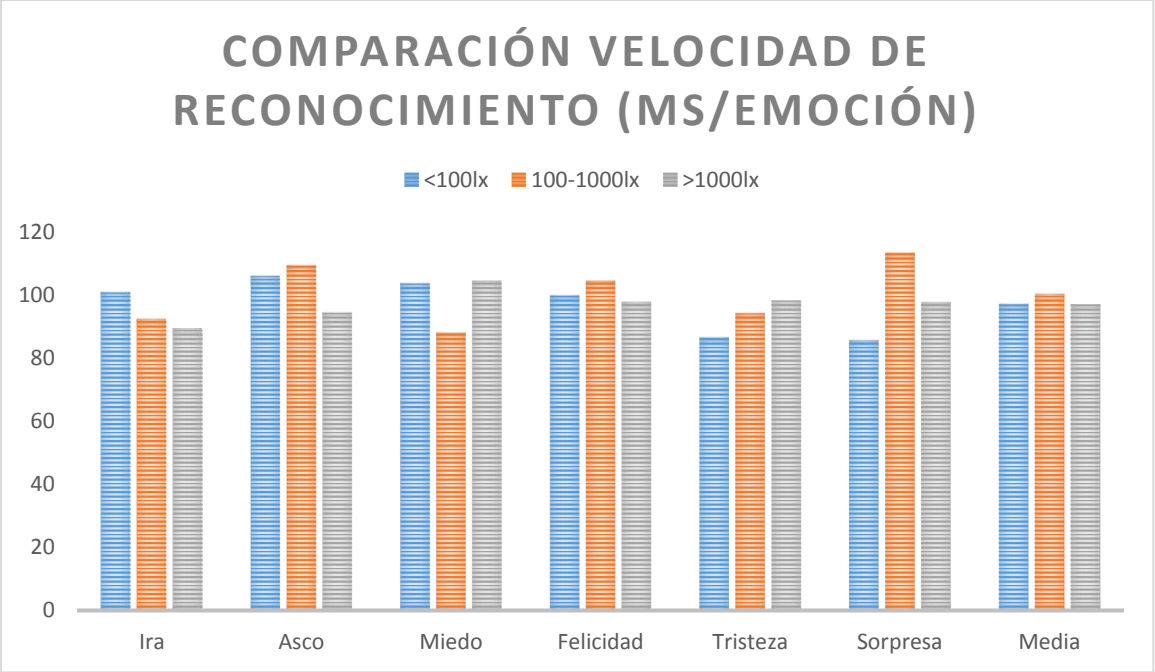


Figura 70 Comparación velocidad de reconocimiento

Sin embargo, sí que influye a la hora de reconocer la emoción correctamente como se muestra en la Figura 71 Los motivos pueden ser:

- Mala localización de los puntos, sobretodo de los de la boca. Esto afecta a la posterior detección de las unidades de acción.
- Incorrecta detección de las unidades de acción, pudiendo tener origen en la causa anterior o en el algoritmo de detección de dichas AUs.
- Reconocimiento de la emoción indeseada. Debido a las dimensiones del árbol y las unidades de acción que influyen en él, es probable que el clasificador llegue a una solución errónea.

De cualquier manera, se ha comprobado que en condiciones ideales los puntos son detectados correctamente por lo que los fallos son originados únicamente por el árbol de decisión. Como se ha explicado en alguno de los casos, la razón de ello es el hecho de que tenga en cuenta sólo algunos atributos y que algunos de ellos sean los menos característicos según el estudio realizado y tengan más importancia que otros. En este árbol por ejemplo no aparece la Au12 o AuH para detectar felicidad, dos unidades de acción únicas de esta emoción. Sin embargo, la Au6 aparece en el nodo decisor y por tanto llevar a confusión ya que puede estar presente en otras emociones también.

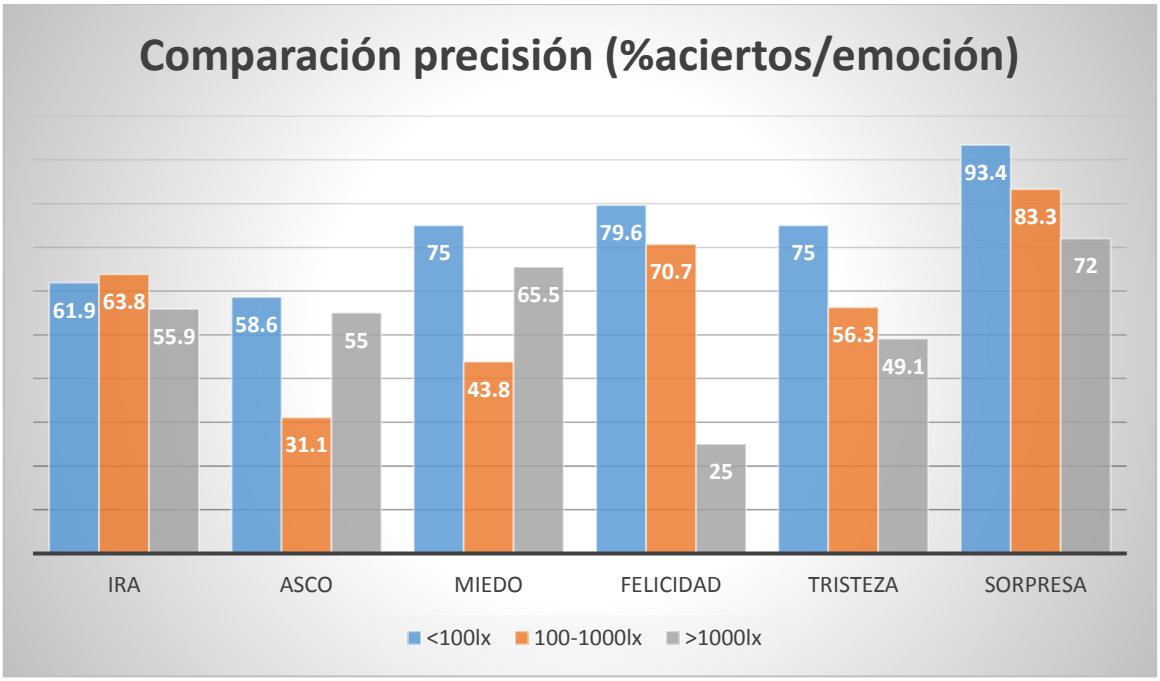


Figura 71 Comparación precisión en función de iluminación

Debido a lo anteriormente comentado se llega a la conclusión de que la matriz posee escasa información y el árbol no es capaz de tomar decisiones fiables. Es necesario cambiar la matriz de entrenamiento de tal modo que logre colocar en sus nodos principales los atributos más característicos. En el capítulo de mejoras futuras se incluyen algunas ideas para mejorar esta matriz de entrenamiento y consecuentemente aumentar la precisión del reconocimiento.

5.3 Pruebas diferenciación emociones efusivas y no efusivas

En este apartado se lleva a cabo un análisis de los problemas que supone para el árbol de decisión no disponer de la información suficiente para clasificar de manera correcta las emociones. La matriz de entrenamiento que se ha utilizado para generar este clasificador tiene un número de muestras muy bajo con respecto al número de emociones a detectar, puesto que existen 30 muestras solamente frente a las seis emociones posibles. Por este motivo, y a tenor de los resultados obtenidos en el apartado anterior se puede considerar que el árbol está sub-entrenado.

Para comprobar estas conclusiones se propone una solución en la cual el árbol de decisión solamente tiene que decidir entre dos posibles emociones: efusividad (*fun* o diversión) y sin efusividad (*suffering* o sufrimiento). Como se indica en Tabla 15, por un lado en las emociones efusivas estarían englobadas felicidad y sorpresa ya que expresan de manera vehemente una reacción. Por otro lado, el resto de las seis emociones (ira, asco, miedo y tristeza) estarían encuadradas como no efusivas.

Tabla 14 Emociones básicas vs emociones simplificadas

Emoción	Emoción simplificada
Ira	No efusiva (Sufrimiento)
Asco	No efusiva (Sufrimiento)
Miedo	No efusiva (Sufrimiento)
Tristeza	No efusiva (Sufrimiento)
Felicidad	Efusiva (Diversión)
Sorpresa	E Efusiva (D1iversión)

Aplicando la misma matriz de entrenamiento el árbol de decisión generado es el que se muestra en la Figura 72, apreciando en este caso que las unidades de acción en principio más características se sitúan en la raíz del árbol. En este árbol las decisiones son tomadas con mayor fiabilidad puesto que solamente tiene que decidir entre dos salidas.

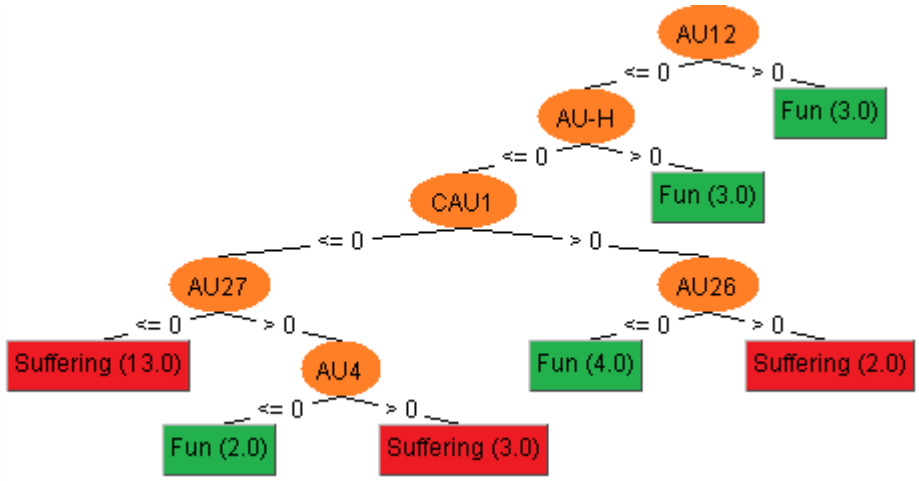


Figura 72 Árbol de decisión emociones simplificadas

Con este árbol de decisión creado parece sencillo diferenciar las dos emociones propuestas. Los resultados de esta solución se pueden encontrar en el documento presentado oficialmente [70].
En él se propone un sistema rápido de reconocimiento facial de emociones que sea capaz de tomar consciencia de la emoción de un sujeto y realizar modificaciones en función de ello. Un ejemplo de aplicación de este sistema pueden ser los *exergames* (juegos de deporte interactivo entre la máquina y un usuario).

La idea consiste en ejecutar esta herramienta en segundo plano del juego de tal modo que se hagan cambios en el mismo en función de la emoción detectada. Por ejemplo, si una persona se está divirtiendo con un juego de tipo *exergame* dedicado al baile se podría aumentar la intensidad del ejercicio físico si se detecta que la persona está contenta o disminuir en caso que se detecte que la persona no muestra efusividad a la hora de jugar.

En la Tabla 15 se pueden comprobar los resultados obtenidos al utilizar solamente estas dos emociones aplicándolos a las imágenes de la base de datos CK+.

Tabla 15 Resultados obtenidos diferenciando dos emociones simplificadas con la base de datos CK+

	Emoción probada					
Emoción detectada	Ira	Asco	Miedo	Felicidad	Tristeza	Sorpresa
Diversión	13	5	38	**61**	41	**90**
Sufrimiento	**87**	**95**	**63**	39	**59**	10

En la Tabla 16 se muestran los resultados obtenidos al aplicar esta propuesta en la aplicación Android desarrollada.

Tabla 16 Resultados obtenidos diferenciando dos emociones simplificadas en Android en tiempo real

	Emoción probada					
Emoción detectada	Ira	Asco	Miedo	Felicidad	Tristeza	Sorpresa
Diversión	9	4	16	**92**	32	**97**
Sufrimiento	**91**	**96**	**84**	8	**68**	3

Como se puede observar en ambas tablas, los resultados obtenidos al utilizar como salida del árbol solamente dos emociones posibles son positivos, obteniendo en los dos casos altos porcentajes. Este hecho reafirma las conclusiones que se habían obtenidos en el apartado anterior tras hacer las pruebas, el árbol de decisión está sub entrenado y no es capaz de tomar decisiones fiables si no dispone de suficiente información para clasificar un número determinado de salidas. En este caso, al haberse aplicado la misma matriz de combinaciones (30 filas x 10 columnas) pero solamente clasificar dos salidas el árbol determina las decisiones con mayor seguridad al tener un mayor número de datos que puede analizar para cada salida.

6. Conclusiones

En este proyecto se ha llevado a cabo la implementación de una aplicación Android de reconocimiento facial de emociones tomando como referencia la herramienta desarrollada en Matlab. Ambos sistemas se basan en la comparación de un rostro con la emoción que se desea reconocer y la misma cara con una expresión neutral.

Se han cumplido los objetivos propuestos mejorando incluso la eficiencia de la herramienta base, ya que el sistema desarrollado no sólo es capaz de reconocer una emoción con mayor rapidez, sino además es capaz de hacerlo en tiempo real. Esto ha sido posible gracias al uso de OpenCV, un conjunto de librerías que permiten manipular la cámara del dispositivo móvil y realizar un procesado de la imagen capturada.

En primer lugar se hizo un análisis teórico de la aplicación contemplando los requisitos que debía cumplir, las especificaciones, los requerimientos mínimos del dispositivo móvil o el diseño de la interfaz gráfica teniendo en cuenta el procedimiento que se realiza en el reconocimiento de emociones.

Los primeros pasos prácticos de este trabajo estuvieron marcados por el estudio y análisis del sistema operativo Android. Para entender mejor su estructura y funcionamiento se implementaron una serie de aplicaciones sencillas. Durante este proceso se fueron solucionando todos los problemas y dificultades surgidos. El punto de inflexión que permitió avanzar al siguiente paso fue la corrección de la orientación de la imagen capturada por la cámara.

Una vez solucionado este error, y partiendo de un ejemplo proporcionado por las librerías usadas, se empezó a implementar la primera fase del reconocimiento, la detección de la cara. Se compararon los distintos modelos a usar para ello obteniendo que LBP era el que mayor rapidez proporcionaba independientemente de las condiciones de iluminación.

El siguiente paso consistía en extracción de las características más importantes. Para completar este proceso se tuvieron en cuenta las fases que se llevan a cabo en la herramienta de Matlab. Estas etapas son la división en regiones de la cara, el pre procesado de cada una de ellas, la segmentación realizada para encontrar las partes de interés (ojos, cejas boca y nariz) y la búsqueda final de los puntos de interés. El proceso de segmentación requirió un amplio tiempo de estudio y pruebas puesto que a pesar de utilizar métodos parecidos a los empleados en el sistema de referencia se tuvieron que realizar una gran cantidad de modificaciones y correcciones que permitieran la correcta detección. Sin embargo, esta circunstancia ha provocado que la búsqueda de los puntos de interés sea mucho más precisa en esta herramienta desarrollada.

Con la localización de los puntos de interés de la imagen neutral y la imagen con la emoción a reconocer se dispone de información suficiente para detectar los cambios musculares producidos en la cara y que corresponderían con las unidades de acción. Tras un profundo estudio realizado en paralelo a este proyecto se obtuvieron las AUs más relevantes y sus posibles combinaciones, que son las que utiliza el árbol de decisión empleado en este proyecto para su entrenamiento.

Este clasificador se implementó gracias a la librería de código abierto WEKA. La idea consistía en aplicar realizar un entrenamiento con las combinaciones ideales para intentar que el árbol de decisión llegara a las posibles soluciones de la manera más rápida posible, pero el árbol resultante no fue tan óptimo como en un principio se esperaba. Situaba en los nodos más importantes unidades de acción que no son las más características para distinguir las emociones.

Por último, en la última fase del proyecto se realizaron pruebas para ver la eficiencia y velocidad de la aplicación en función de la iluminación. Se llegó a la conclusión de que la velocidad de reconocimiento es aproximadamente 100s independientemente de la iluminación, pero en cambio la precisión del sistema disminuye si existen malas condiciones puesto que no es capaz de detectar los puntos eficientemente.

Aún en condiciones ideales y obteniendo una buena detección de puntos de interés a veces la aplicación no reconoce la emoción correcta. Esto es debido al árbol de decisión que se generaba, comprobándose que estaba sub entrenado y no disponía de datos suficientes para clasificar con fiabilidad las seis emociones. Además, se confirmó al hacer pruebas reduciendo el número de emociones a reconocer. Se dividieron las seis emociones básicas en dos grandes grupos, emociones efusivas que implicaban diversión y emociones no efusivas que podían indicar sufrimiento. Tras realizar el entrenamiento del árbol con las mismas combinaciones y solo dos posibles salidas se obtuvieron unos resultados considerablemente mejores.

Como extras añadidos, se incorporaron opciones en la aplicación para mostrar las regiones y puntos detectados de tal manera que se pueda observar en cualquier momento si está realizando una localización correcta de las regiones de interés de la cara.

Aunque se han cumplidos los objetivos y requisitos definidos al inicio del proyecto, los resultados tienen aún margen de mejora a la hora de poder reconocer y distinguir correctamente las seis emociones básicas. Por esta razón en el siguiente capítulo se propondrán algunas ideas de líneas futuras que se pueden seguir para mejorar esta herramienta.

7. Trabajo futuro

Con el fin de seguir evolucionando y mejorando la herramienta desarrollada se proponen una serie de mejoras futuras.

Siguiendo la línea de trabajo se podrían implementar nuevas técnicas para la extracción de características. Como se ha comentado a lo largo del proyecto la correcta detección de los puntos de interés está limitada por la presencia de arrugas, oclusiones, malas condiciones de iluminación, sombras, etc. Una posible línea de trabajo a seguir podría ser la introducción de métodos basados en mallas como AAM que permita crear una serie de puntos de referencia. Otras aplicaciones explicadas en el estado del arte ya utilizan técnicas similares.

Una mejora inmediatamente directa que surge a partir del análisis de los resultados obtenidos es ampliar el conjunto de combinaciones de unidades de acción de tal manera que el clasificador tenga más información para poder realizar el entrenamiento y así ser capaz de tomar decisiones fiables para las seis emociones. Asimismo, se pueden razonar otras técnicas y algoritmos para detectar esas AUs.

Otra línea de investigación que se puede seguir es utilizar otros métodos de clasificación y compararlos con los árboles de decisión. De hecho, esta herramienta se ha desarrollado de forma modular de tal manera que se puedan incorporar nuevos métodos de aprendizaje sin problema. En este caso además se ha decidido incluir ya la posibilidad de seleccionar otros modelos de clasificación como son ANN, SVM o las redes bayesianas.

Respecto a la funcionalidad de la aplicación, es necesario otorgar cierta estabilidad a la hora de reconocer emociones y mostrarlas por pantalla, ya que actualmente es un sistema con variaciones constantes sin apenas ejercer cambio en los músculos de la cara.

Otra mejora de funcionalidad podría ser incorporar una detección de la iluminación detectada por los sensores del móvil con el objetivo de adecuar el reconocimiento en función de ello.

8.Referencias

[1] A. Mehrabian,“Communication without words“, *Psychology Today*, vol.2, no.4, pp 53–56, 1968

[2] C. Darwin,“The Expression of the Emotions in Man and Animals“,. John Murray, London, pp. 88–144, 1872

[3] P. Ekman, W.V. Friesen, “Constants across cultures in the face and emotion.“ *Journal of Personality and Social Psychology*, vol.17, no. 2, pp. 124–129, 1971

[4] J.A. Russell, “Is there universal recognition of emotion from facial expressions? A review of the cross-cultural studies,” *Psychological Bulletin,* vol.115, no. 1, pp. 102-141, Jan.1994.

[5] P. Ekman, “Strong evidence for universals in facial expressions: A reply to Russell's mistaken critique,” *Psychological Bulletin,* vol.115, no. 2, pp. 268-287, Mar.1994.

[6] C.E. Izard, “Innate and universal facial expressions: Evidence from developmental and cross-cultural research,” *Psychological Bulletin,* vol. 115, no. 2, pp. 288-299, Mar. 1994.

[7] W.G. Parrott, *“Emotions in Social Psychology,” Psychology Press*, Philadelphia, Oct. 2000.

[8] V. Bettadapura*, “Face Expression Recognition and Analysis: The State of the Art“.* ,College of Computing, Georgia Institute of Technology*, pp. 8-9, 2012*

[9] T. Kanade, J. Cohn, and Y. Tian, “Comprehensive Database for Facial Expression Analysis*,” Proc. IEEE Int’l Conf. Face and Gesture Recognition (AFGR ’00),*pp. 46-53, 2000.

[10] S. Du, A.M. Martinez, “Compound facial expressions of emotion: from basic research to clinical applications”, *Dialogues Clin Neurosci*, vol.17, no. 4, pp. 443-455., 2015

[11] D. Espinoza, “Micro expresiones para conocer mejor a tu interlocutor“, 16 Agosto 2014 [En línea], Disponible: http://www.saocupsi.com/2014/08/microexpresiones-para-conocer-mejor-tu.html, [Último acceso: 02/05/2016]

[12] I. Cohen, N. Sebe, A. Garg, L.S. Chen, and T.S. Huang, “Facial Expression Recognition From Video Sequences: Temporal and Static Modeling”, *Computer Vision and Image Understanding,* vol. 91, pp. 160-187, 2003.

[13] A. Samal and P.A. Iyengar, “Automatic Recognition and Analysis of Human Faces and Facial Expressions: A Survey,” *Pattern Recognition,* vol. 25, no. 1, pp. 65-77, 1992.

[14] B. Fasel and J. Luttin, “Automatic Facial Expression Analysis: a survey,” *Pattern Recognition,* vol. 36, no. 1, pp. 259-275, 2003.

[15] E. Hjelmas, B. Kee Low, “Face Detection: A Survey,” *Computer Vision and Image Understanding,* 83, 236-274 April 2001.

[16] A. Sharifara, M.S.M. Rahim and Y. Anisi, “A General Review of Human Face Detection Including a Study of Neural Networks and Haar Feature-based Cascade Classifier in Face Detection”, *International Symposium on Biometrics and Security Technologies (ISBAST)*, Universiti Teknologi Malaysia, 2014

[17] H. Hatem, Z. Beiji and R. Majeed, “A Survey of Feature Base Methods for Human Face Detection”, *International Journal of Control and Automation*, Vol.8, no.5, pp.61-78,2015

[18] P. Viola y M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition. CVPR 2001*, vol. 1, págs. I–511–I–518, 2001.

[19] P. Viola and M.J. Jones, “Robust real-time object detection,”*Int. Journal of Computer Vision,* vol. 57, no. 2, pp. 137-154, Dec. 2004.

[20] A. Harvey, Interviewee, *Adam Harvey Explains Viola-Jones Face Detection.* [Entrevista]. 2012

[21] T. Ojala, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on* vol. 24, no. 7, págs. 971–987, 2002.

[22] S. Moore, R. Bowden, “Local binary patterns for multi-view facial expression recognition”, *Computer Vision and Image Understanding,* vol. 115, pp. 541-558, 2011

[23] J. Chang-yeon, “Face Detection using LBP features”, Final Project Report, Dec. 2008

[24] Opencv dev team «OpenCV,» 2011-2014. [En línea]. Available: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms [Último acceso: 11 de Mayo 2016].

[25] V.Gupta, D.Sharma, “A Study of Various Face Detection Methods”, *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 3, no. 5, May. 2014

[26] J. Ahlberg, “CANDIDE-3 - An Updated Parameterized Face”, Image Coding Group, Linköping University, Sweden, 2001

[27] J. Eslava Rios, “Reconocimiento facial en tiempo real”, PFC Universidad Autónoma de Madrid, Jul.2013

[28] Yuille, A. L., Cohen, D. S., and Hallinan, P. W., "Feature extraction from faces using deformable templates", Proc. of CVPR, 1989.

[29] T. Cootes, G. Edwards, y C. Taylor, “Active appearance models,” *IEEE Trans. pattern analysis and machine intelligence,* vol. 23, no. 6, págs. 681–685, 2001.

[30] L. Wiskott y J. Fellous, “Face recognition by elastic bunch graph matching,” *Pattern Analysis and machine intelligence,* págs. 1–23, July, 1997.

[31] S. P. Khandait, R. C. Thool, y P. D. Khandait, “ANFIS and BPNN based Expression Recognition using HFGA for Feature Extraction,” vol. 2, no. 1, págs. 11–22, 2013.

[32] L. Blazquez, “Reconocimiento Facial Basado en Puntos Característicos de la Cara en entornos no controlados,” Proyecto fin de Carrera, Universidad Autónoma de Madrid, Madrid, En. 2013.

[33] P. E. Group, “FACS Archives - Paul Ekman Group, LLC.” [Online]. Available: http://www.paulekman.com/product-category/facs/.

[34] S.Gonzalez, "Desarrollo de una herramienta para analizar métodos de reconocimiento de emociones", Proyecto Fin de Grado, May. 2015

[35] P. Lucey, J. Cohn, y T. Kanade, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," *Comput. Vision and Pattern Recognition Workshops (CVPRW),* Jul. 2010.

[36] D. Myint and N. Aye, "Automatic Facial Expression Recognition System using Orientation Histogram and Neural Network*", International Journal of Computer Applications (0975 – 8887),* Vol. 63–no.18, Feb. 2013

[37] A.N. Martinez-Gonzalez and V. Ayala-Ramirez, "Real Time Face Detection Using Neural Networks", Mexican International Conference on Artificial Intelligence, 2011

[38] Y. Xiao, L. Ma, K. Khorasani "A New Facial Expression Recognition Technique Using 2-D DCT and Neural Networks Based Decision Tree", *International Joint Conference on Neural Networks*, Jul. 2006

[39] Tom M. Mitchell, "Machine Learning", McGraw-Hill Science, pp. 52-78, 1997

[40] S. Sriram, X. Yuan, "An Enhanced Approach for Classifying Emotions using Customized Decision Tree Algorithm", School of Computer Science, 2012

[41] Opencv dev team «OpenCV,» 2011-2014. [En línea]. Available: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html [Último acceso: 22 de Mayo 2016].

[42] P. Michel, R. El Kaliouby, "Real Time Facial Expression Recognition in Video using Support Vector Machines*", International Conference on Multimodal Interaction*, Nov. 2003

[43] Wikipedia- Enciclopedia libre "K-nearest neighbors algorithm,". [En línea]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm. [Último acceso: 22 de Mayo 2016].

[44] R. Azmi, S. Yegane, "Facial Expression Recognition in the Presence of Occlusion Using Local Gabor Binary Patterns", *20th Iranian Conference on Electrical Engineering*, May. 2012

[45] J. P. Ordonez Lopez, "Redes Bayesianas", [En línea]. Available: https://jpordonez.wordpress.com/2008/08/08/redes-bayesianas/. [Último acceso: 22 de Mayo 2016].

[46] I. Cohen, N. Sebe, F.G. Cozman, M.C. Cirelo, T.S. Huang, "Learning Bayesian Network Classifiers for Facial Expression Recognition using both Labeled and Unlabeled Data", *Computer Society Conference on Computer Vision and Pattern Recognition,* 2003

[47] Wikipedia- Enciclopedia libre "Modelo oculto de Markov,". [En línea]. Available: https://es.wikipedia.org/wiki/Modelo_oculto_de_M%C3%A1rkov. [Último acceso: 22 de Mayo 2016].

[48] M. Pardàs, A. Bonafonte, "Facial animation parameters extraction and expression recognition using Hidden Markov Models", Signal Processing: Image Communication, vol.17, pp. 675-688, 2002

[49] Android, Android Developers, [En línea]. Available: https://developer.android.com/index.html, [Último acceso: 22 de Mayo 2016]

[50] Statista, "Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 4th quarter 2015", [En línea]. Available: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, [Último acceso: 22 de Mayo 2016]

[51] Snapchat, Inc., Snapchat (Versión 9.30.5.0) [Aplicación Móvil]. Descargado de: https://play.google.com/store/apps/details?id=com.snapchat.android

[52] Masquerade Technologies, Inc, MSQRD (Version 1.5.1), [Aplicación Móvil]. Descargado de: https://play.google.com/store/apps/details?id=me.msqrd.android

[53] Nordic APIs, "20+ Emotion Recognition APIs That Will Leave You Impressed, and Concerned", dic. 2105, [En línea]. Available: http://nordicapis.com/20-emotion-recognition-apis-that-will-leave-you-impressed-and-concerned/

[54] Human Sensing Laboratory and Affect Analysis Group, Intraface (Version 1.16), [Aplicación Móvil]. Descargado de: https://play.google.com/store/apps/details?id=com.intraface.intraface

[55] X. Xiong and F. De la Torre. "Supervised descent method and its applications to face alignment". *In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 532–539. IEEE, 2013.

[56] W.-S. Chu, F. De la Torre and J. F. Cohn, "Selective Transfer Machine for Personalized Facial Action Unit Detection", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2013,

[57] AFFECTIVA, INC (2016), Affectiva, [En línea]. Available: http://www.affectiva.com/ [Último acceso: 22 de Mayo 2016]

[58] T. Senechal, D. McDuff and R. el Kaliouby, " Facial Action Unit Detection using Active Learning and an Efficient Non-Linear Kernel Approximation", Affectiva

[59] Affectiva, Inc., Affdexme (Version 3.0.1-103), [Aplicación Móvil]. Descargado de: https://play.google.com/store/apps/details?id=com.affectiva.affdexme&hl=es

[60] D. McDuff, R. El Kaliouby, T. Senechal, May Amr, J. Cohn, R. Picard, Affectiva MIT Facial Expression Dataset (AM-FED): Naturalistic and Spontaneous Facial Expressions Collected "In the Wild", *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013

[61] Itseez-OpenCV Developers Team, OpenCV, [En línea]. Available: http://opencv.org/about.html [Último acceso: 01 de Junio de 2016]

[62] The Eclipse Foundation, Eclipse, [En línea]. Available: https://eclipse.org/ [Último acceso: 01 de Junio de 2016]

[63] Android, Android Developers [En línea]. Available: http://developer.android.com/intl/es/tools/studio/index.html [Último acceso: 01 de Junio de 2016]

[64] Android, Android Developers [En línea]. Available: http://developer.android.com/intl/es/about/dashboards/index.html [Último acceso: 01 de Junio de 2016]

[65] Xiaomi Global Community, Xiaomi Global, [En línea]. Available: http://xiaomi-mi.com/, [Último acceso: 02 de Junio de 2016]

[66] Opencv dev team «OpenCV,» 2011-2014. [En línea]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html [Último acceso: 09 de Junio 2016].

[67] A. Gil, "Integración de algoritmos basados en Unidades de Acción en una herramienta de análisis de reconocimiento de emociones", Proyecto Fin de Grado, Jun. 2016

[68] R. R. Bouckaert, E. Frank, M. A. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. "WEKA-experiences with a java open-source Project". *Journal of Machine Learning Research*, 11:2533-2541, 2010.

[69] J. R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993.

[70] M. Eckert, A.Gil, D. Zapatero, J. Meneses, J. F. Martínez, "Fast facial expression recognition for emotion awareness disposal", Research Center on Software Technologies and Multimedia Systems for Sustainability (CITSEM), Jun. 2016

9.Anexo: Configuración del entorno

En primer lugar se instalan los componentes necesarios para poder ejecutar una aplicación Android y empezar a conocer el funcionamiento básico y todas sus características. Después se instalan los componentes necesarios para el tratamiento de imagen para poder realizar el reconocimiento facial de emociones. En resumen, los componentes necesarios para la configuración del entorno son los siguientes:

1. JDK (*Java Development Kit)*
2. Entorno de desarrollo (Eclipse/Android Studio)
3. SDK Android (*Software Development Kit*)
4. Plug-in ADT (*Android Development Tools*) para Eclipse
5. NDK (*Native Development Kit*) y CDT (*C/C++ Development Tools*)
6. Librerías OpenCV para Android

PASO 1. DESCARGA DE JDK

Descargar e instalar el JDK (*Java Development Kit*) adecuado para nuestro sistema operativo. Es el software que provee las herramientas de desarrollo necesarias para la creación de programas en Java. Se puede encontrar en la página oficial de Oracle [1]

Se puede comprobar si se ha instalado correctamente el JDK en el símbolo del sistema como se indica en la Figura 73 Comprobación versión JAVA (*Inicio >Ejecutar >'cmd'*) introduciendo la siguiente línea de código:

```
C:\Users>java -version
```

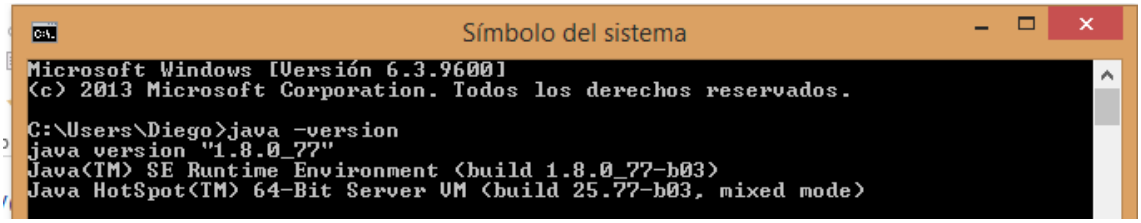


Figura 73 Comprobación versión JAVA

Para instalar correctamente el compilador de Java es necesario añadirlo a las variables de entorno de Windows. Esto se añade en configuración avanzada del sistema (*Inicio>Panel de Control>Sistema*) en el apartado de variables del entorno (ver Figura 74)

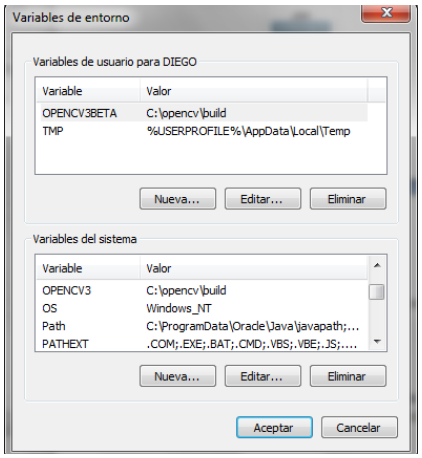


Figura 74 Variables de entorno

Se crea una nueva variable pinchando en 'nueva' con nombre 'JAVA_HOME' e introduciendo la ruta donde se encuentre el JDK descargado en el campo de 'valor de variable' (en mi caso por ejemplo: C:\Program Files\Java\jdk1.8.0_40). Una vez creada esta variable, hay que editar la variable 'Path' y al final de toda la línea hay que introducir ';%JAVA_HOME%'.

PASO 2. DESCARGA DE ECLIPSE

Descarga del entorno de desarrollo Eclipse de su página oficial [2]. No requiere instalación. Una vez descargado, se descomprime, y se guarda la carpeta deseada (si se ha descargado también Android Studio es recomendable ubicarlo en la misma carpeta para una mejor organización). Para abrirlo, simplemente ejecutar 'eclipse.exe' y seleccionar el espacio de trabajo donde se irán guardando todos los proyectos creados.

PASO 3. DESCARGA DEL SDK

Descargar el SDK (*Software Development Kit*, un kit de desarrollo de software que contiene las herramientas necesarias para compilar y empaquetar nuestras aplicaciones.) de Android de la página oficial [3] . Se puede descargar además junto con el nuevo entorno oficial de desarrollo de aplicaciones Android, Android Studio. Para instalar este entorno simplemente es necesario seguir los pasos de instalación. En este caso se explicará la integración del SDK en el entorno de Eclipse ya que ha sido el utilizado en el desarrollado de esta aplicación.

Una vez descargado el SDK, se descomprime y se procede a su instalación y configuración (si se ha instalado junto con Android Studio se puede hacer desde el propio). Para configurar el SDK, se abre el 'SDK Manager' en Eclipse desde la pestaña *Window>Android SDK Manager*. Como se muestra en la Figura 75 En él, se puede ver todos los componentes que se han instalado y cuales están aún por instalarse (herramientas, extras, versiones de Android, etc).

Es recomendable descargarse siempre la última versión y todas las necesarias para las cuales la aplicación se pueda ejecutar. Dentro de cada versión se pueden instalar diferentes componentes. Además del SDK, un componente importante es el emulador (Arm o Intel) que permite simular un dispositivo Android para poder probar en él las aplicaciones creadas.

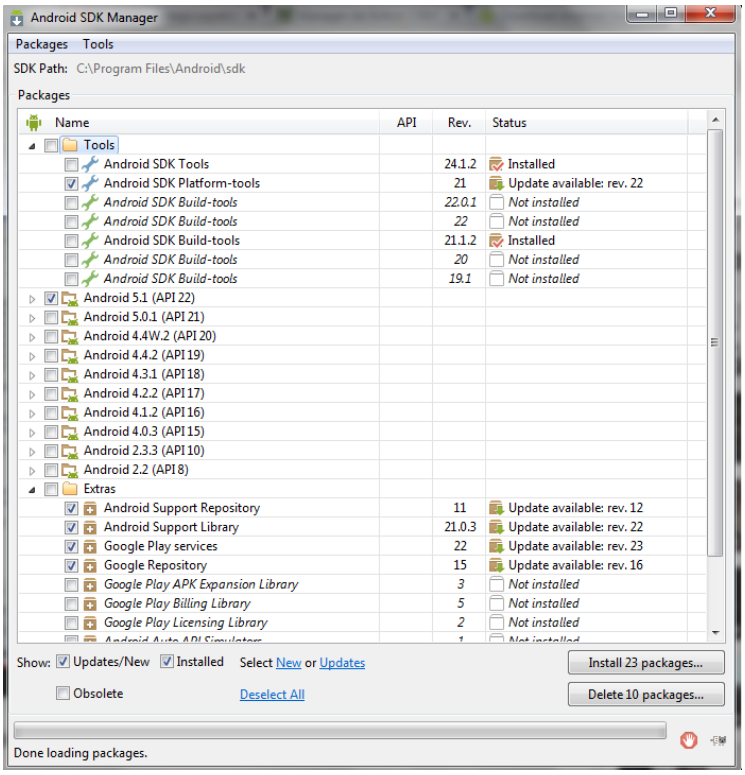


Figura 75 SDK Manager

De los extras es recomendable instalar el 'Inter x86 Emulator Accelerator (HAXM)' (solo para procesadores Intel) para que el uso del emulador sea mucho más fluido.

Hay que referenciar el SDK para que Eclipse conozca donde se ubica. Para ello en la pestaña *Window>Preferences>Android* se pone la ruta donde se encuentre la carpeta del SDK.

PASO 4. DESCARGA DEL PLUG-IN ADT

Hay que descargar también los plug-ins ADT (*Android Development Tools*), un plug-in para el IDE Eclipse que amplía las capacidades para que pueda configurar rápidamente nuevos proyectos para Android, crear una interfaz de usuario de la aplicación, agregar paquetes basados en la API de Android Framework, depurar sus aplicaciones usando el SDK de Android herramientas, e incluso exportar firmado archivos con el fin de distribuir la aplicación. La forma más sencilla de instalar este plug-in es introduciendo desde Eclipse el enlace proporcionado por la página oficial de Android y seguir los pasos indicados [4]

Tras su instalación, es necesario reiniciar Eclipse. Para instalar futuras actualizaciones en la pestaña Help>software updates se puede comprobar si existen actualizaciones.

PASO 5. DESCARGA DEL NDK

Una vez realizadas las pruebas básicas, se procede a la instalación de los componentes necesarios para la utilización de la librería de OpenCV en la parte del procesamiento de la imagen. Al estar estas librerías desarrolladas en C++ es necesario instalar las herramientas necesarias para poder tratar este lenguaje y que Android pueda interpretarlas. Para ello se requiere la instalación del NDK (*Native Development Kit*) necesario para poder ejecutar procesos en C++ y el plug-in CDT

(*C/C++ Develpment Tools*) que proporciona ciertas características para el desarrollo en dicho lenguaje de programación. Por un lado, el NDK se puede descargar directamente de la página oficial de Android [5]

Hay que referenciar este Kit en Eclipse y en las variables del sistema. Para ello, se hace un proceso idéntico al realizado cuando se instaló el JDK. Primero se crea una nueva variable con nombre 'NDKROOT' y valor la ubicación del directorio donde se encuentre el NDK (Figura 76) y después se actualiza la variable del entorno 'PATH' añadiendo en el campo 'Valor de la Variable' la variable creada tal y como se muestra en la Figura 76

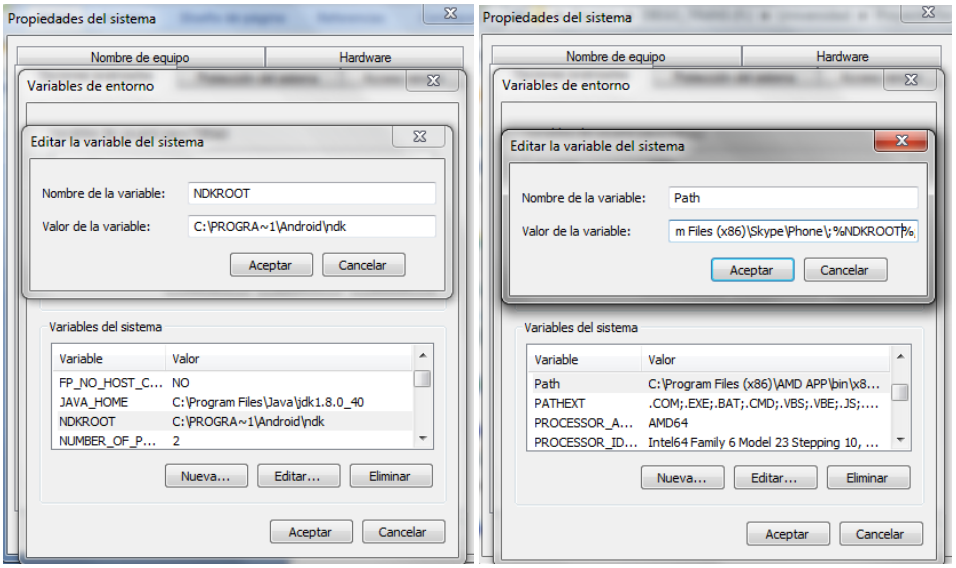


Figura 76 Variable entorno NDK

Si aparece algún error, realizar también los siguientes pasos en Eclipse:

Window>Preferences>C/C++>Build>Build Variables> Add... y seleccionar la variable NDKROOT que contiene ruta específica.

Window>Preferences>C/C++>Build>Environment> Select y seleccionar NDKROOT

Además, Eclipse debe saber la ruta específica donde se encuentra el NDK. Esto se realiza en *Windows> Preferences>Android>NDK* (IMPORTANTE: La ruta NO puede contener espacios, así que hay que ubicar la carpeta en una ruta que no contenga espacios)

Por otro lado, el plug-in CDT debería estar instalado si cuando se instaló el plug-in ADT se seleccinó también la instalación de 'NDK plug-ins'. Esto se puede comprobar en Eclipse yendo a *Help>Installation Details*. En la pestaña Plug-ins se puede comprobar qué plug-ins se encuentran instalados.

Si no está instalado, basta con ir a *Help>Install New Software* y en el apartado de 'Work with' seleccionar 'All available sites' para que busque todos los componentes disponibles para instalar que no estén ya instalados. Seleccionar el que nos falte.

Otra opción es descargarlo directamente y luego referenciarlo con el botón *Add..>Archive* y seleccionar el CDT descargado.

PASO 6. DESCARGA E INSTALACIÓN LIBRERÍAS OPENCV

El siguiente y último paso es proceder a la instalación de las librerías OpenCV [6]. Una vez descargado el archivo, se descomprime y se coloca donde el usuario desee. La carpeta, además de contener las librerías contiene una serie de ejemplos. Para usar las librerías en un proyecto es necesario que esta se encuentre en el mismo workspace de Eclipse:

File> Import> Existing Android Project into Workspace> sdk->java

Comprobar que esta tratado como una librería (*Botón derecho sobre el Proyecto>Propiedades> Android*) y mirar que esté marcada la opción de *IsLibrary*). Después, para importar los ejemplos:

File> Import> Existing Android Project into Workspace> sdk->samples

Cuando se cree un proyecto nuevo y se quieran utilizar estas librerías (ya previamente ubicadas en el mismo *workspace*) habrá que importarlas. Para ello, hacer clic con el botón derecho del ratón encima del proyecto e ir a la sección de Propiedades>Android>Add> Library. Se puede encontrar más información al respecto en [7][8].

Se puede probar cualquiera de los ejemplos de la misma manera que se realiza con un proyecto normal, pero al usar la librería OpenCV el dispositivo o emulador solicitará la instalación previa de la aplicación OpenCV Manager, herramienta que integra las librerías necesarias. En el caso de que no se desee instalar esta aplicación (es recomendable su instalación por parte del programador) basta con copiar las librerías <SDK opencv>/sdk/native/libs/<target_arch> al directorio del proyecto en la carpeta libs/<target_arch>.

PASO 7. EJECUCIÓN Y PRUEBA APLICACIÓN

Para probar la aplicación en un dispositivo móvil es necesario configurarlo previamente. En primer lugar, para poder realizar un debug de la aplicación hay que habilitar las opciones de desarrollo. En función de la versión de Android se realiza el proceso de manera diferente pero en general se suele encontrar el Ajustes>Opciones de desarrollador>Depuración USB (Figura 77). Además, para poder instalar la aplicación se debe permitir la instalación desde fuentes desconocidas. Esta opción normalmente se encuentra en Ajustes>Opciones Avanzadas>Seguridad>Fuentes desconocidas (Figura 77). Una vez configurado ya se puede conectar al ordenador y ejecutar la aplicación desde Eclipse.

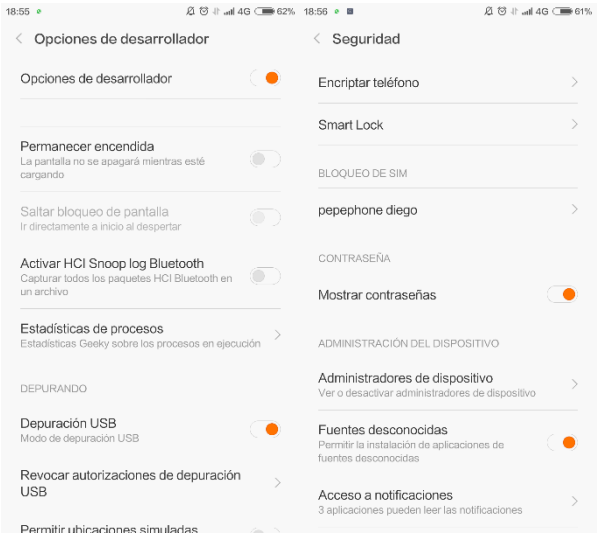


Figura 77 Configuración dispositivo móvil

REFERENCIAS

[1] Oracle, Oracle [En línea]. Available: http://www.oracle.com/technetwork/es/java/javase/downloads/index.html[Último acceso: 22 de Mayo 2016].

[2]The Eclipse foundation, Eclipse [En línea]. Available: https://eclipse.org/downloads/packages/eclipse-ide-eclipse-committers-442/lunasr2 [Último acceso: 22 de Mayo 2016].

[3] Android Developers, Android [En línea]. Available: http://developer.android.com/sdk/index.html [Último acceso: 22 de Mayo 2016].

[4] Android Developers, Android [En línea]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html [Último acceso: 22 de Mayo 2016].

[5] Android Developers, Android [En línea]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html [Último acceso: 22 de Mayo 2016].

[6]Opencv dev team «OpenCV,» [En línea]. Available: http://sourceforge.net/projects/opencvlibrary/files/opencv-android/ [Último acceso: 22 de Mayo 2016].

[7]Opencv dev team «OpenCV,» [En línea]. Available: http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html [Último acceso: 22 de Mayo 2016].

[8]Opencv dev team «OpenCV,» [En línea]. Available: http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html[Último acceso: 22 de Mayo 2016].