UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

API de Reconocimiento de Microexpresiones Faciales Utilizando Algoritmos de Fisherface y Support Vector Machine

GONZALO ANDRES MARDONES BAEZA

Profesor Guía: RODRIGO PAREDES

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
21 de Diciembre de 2017

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2019

Dedicado a mis padres, hermano, cleo y osita.
"Todo parece imposible hasta que se hace." - Nelson Mandela.

AGRADECIMIENTOS

Agradezco a todas las personas que desde un principio me entregaron su apoyo incondicional, a quienes ya partieron y dejaron un mensaje que me acompañará el resto de mi vida.

En primer lugar, agradecer a quien lejos fue mi cómplice, quien sólo me entrego motivos para seguir adelante a pesar de las adversidades, entre penas y situaciones complicadas, nunca había tiempo para bajar la cabeza. Sin tus consejos y experiencias de vida mamita nada de esto hubiera sido posible, eres la persona que motivo a este loco patiperro a madurar (algo) y ser cada vez mejor persona.

Miguel Ángel, alias *Goma*, compañero y gran amigo que desde mis inicios como scout me impulsaste a ser un individuo activo en la sociedad tanto nacional como internacionalmente. Gracias por tus consejos, críticas y comentarios con alturas de miras y un cariño enorme. Marquitos Mondaca, un amigo de los que hoy en día no existen, con virtudes y defectos solo sumaste en mi etapa universitaria.

A mis primitos Karen y Pablito Baeza, quienes en todo momento me tendieron sus manos, gracias a ustedes aprendí a nunca dejar de perseverar, ser un agradecido de la vida y de las experiencias. Gracias por siempre aconsejarme que las oportunidades hay que aprovecharlas, gracias a ello pude viajar y aprender a cuanto país o rincón de Chile pude.

Finalmente, este párrafo lo reservo para agradecer a un amigo que siempre me escucho, se preocupo y brindo día a día de su tiempo y apoyo, gracias Sergio Flores por todos los buenos momentos compartidos en esta linda etapa.

Marcelita Pacheco, tía Ceci y tía Cristi las quiero mucho, gracias por todo.

Familia y amigos, gracias a todos.

`.-.. --- ... // --.- ..- .. . .-. --- // -- ..- -.-. .... ---`

Buena caza.

RESUMEN

El reconocimiento facial y de microexpresiones se ha convertido en los últimos años en una área de investigación activa, que abarca diversas disciplinas tales como procesamiento de imágenes, reconocimiento de patrones, visión por computadora y redes neuronales.

El objetivo del reconocimiento facial es el siguiente: dada una imagen de una cara "desconocida", o imagen de test, encontrar una imagen de la misma cara en un conjunto de imágenes "conocidas" o imágenes de entrenamiento.

Apple, Google, Facebook, entre otras empresas utilizan estos algoritmos para ayudar a sus usuarios en procesos de reconocimiento y etiquetado de fotos.

Hoy en día el método tradicional de reconocimiento facial que presenta mejores resultados es Fisherface, ya que demuestra un desempeño superior tanto para imágenes con variación de iluminación como de pose.

Machine Learning es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. Support Vector Machine (SVM) es una algoritmo de aprendizaje supervisado que está relacionado con dar solución a los problemas de clasificación y regresión.

Esta memoria desarrollada en el lenguaje de programación Python, implementa una API reconocimiento facial que utiliza el algoritmo de Fisherface y el clasificador de SVM para la clasificación de microexpresiones oculares que permita validar en modo no presencial a la persona que se encuentra en una imagen y además pueda detectar si se encuentra con las cejas arriba o en estado normal.

Para determinar el correcto funcionamiento de la API, se realizan pruebas de exactitud en los modelos de reconocimiento facial y de microexpresiones oculares. La variable que define la precisión del modelo de reconocimiento facial se llama *confianza*. Los resultados obtenidos determinan que si la confianza se encuentra bajo el valor 500, indica que la imagen consultada corresponde a la persona buscada. Asimismo los registros de las personas poseen imágenes con variaciones de iluminación y poses. Las pruebas de coste utilizando un kernel lineal del clasificador SVM, dio por resultado que el modelo de microexpresiones oculares posee una precisión del 92,2 % utilizando una base de datos de 1,106 imágenes.

TABLA DE CONTENIDOS

página

ÍNDICE DE FIGURAS

página

ÍNDICE DE TABLAS

página

1. Introducción

1.1. Descripción del Contexto

Los seres humanos poseen múltiples capacidades que le permiten relacionarse en el mundo. Una de ellas es la visión como la acción de ver, encargada de interpretar nuestro entorno gracias a los rayos de luz que alcanzan los ojos [2]. Es por ello que la luz ingresa por la pupila y se proyecta en la retina. De esta manera el ojo genera sus conexiones con el cerebro permitiendo percibir la banda visible del espectro electromagnético (EEM), ver Figura 1.1. Sin embargo, se pueden utilizar sensores que perciben rango más amplio, que va desde los rayos gamma hasta las ondas de radiales [14].

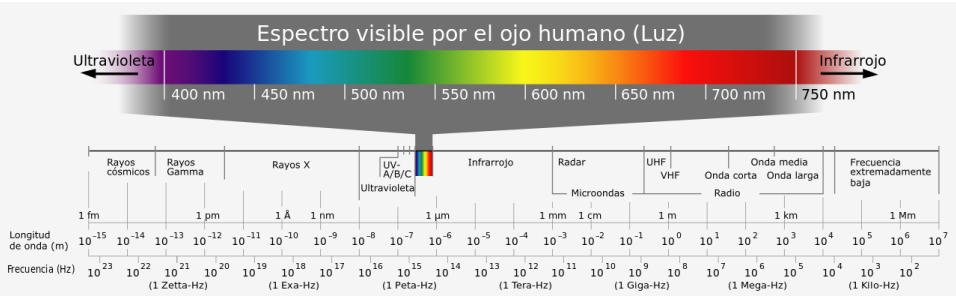


Figura 1.1: Espectro Electromagnético.

Las cámaras de las computadoras poseen un funcionamiento muy similar al del ojo humano, dado que poseen como principio la captura de luz. El ser humano origina el proceso de *percepción*, en donde la luz ingresa al ojo atravesando la córnea para enfocar la imagen. En el caso de las cámaras fotográficas se produce un fenómeno que no es químico, sino electrónico, ya que la luz sensibiliza al sensor de captación de imagen, el cual descompone estos haces de luz en colores rojo, verde y azul.

El humano realiza un proceso de *transformación*, en el cual la luz llega a la retina. En ella se activan las células sensoriales que transforman la luz en energía nerviosa. En

la cámara, las señales son convertidas en cargas eléctricas analógicas por el sensor, que son transferidas a la electrónica del dispositivo.

Posteriormente, se procede al proceso de *transformación*. Este se encarga de enviar los impulsos nerviosos a través del nervio óptico hasta la corteza cerebral; en el caso de las cámaras éstas realizan el procedimiento hacia su procesador para finalmente ser procesadas.

Finalmente, viene la etapa de *interpretación*. En la corteza cerebral se interpretan los impulsos, se reconocen y se procesan para saber lo que vemos. En el caso de este trabajo, el procesador identifica y genera una imagen digital [15, 9].

1.1.1. Introducción al Problema

La empresa *Exe Ingeniería y Software* identifica que actualmente las empresas/ instituciones públicas y privadas en Chile requieren mejorar los tiempos de respuestas con sus clientes o beneficiarios.

La problemática radica que en condiciones específicas (enfermedades o de otra naturaleza), la persona no puede dirigirse a dependencias de la empresa/institución (compañias de seguros, bancos, asociaciones/mutuales de seguridad, etc.) a formalizar algún tipo de cobro de beneficio, realizar algún tramite o solicitud. En estas circunstancias la persona no puede recibir el beneficio simplemente por la imposibilidad física de hacerse presente en dependencias de la entidad.

Hoy en día en Chile, no existe solución costo/beneficio que permita validar a una persona en forma remota con niveles de seguridad que garanticen que la persona "es, quien dice ser".

Las **Interfaz de Programación de Aplicaciones**, abreviada como **API**, de reconocimiento facial (públicas/privadas) existentes no abordan las microexpresiones oculares, se limitan a presentar otros métodos tradicionales de reconocimiento facial: lista de puntos que identifican el contornos facial, si la persona se ubica frente a la cámara, identificar personas en una imagen y algunos presentan los contornos laterales. *Exe* propone desarrollar una API que sea usada por desarrolladores de estas instituciones para dar solución a sus procesos de atención al cliente/beneficiario.

1.2. Objetivos

1.2.1. Objetivo General

Construir una API de reconocimiento facial que utilice el algoritmo de Fisherface y permita la validación y autentificación en modo no presencial de personas por medio de sus microexpresiones oculares utilizando el algoritmo de Support Vector Machine.

1.2.2. Objetivos Específicos

1. Obtener, representar y extraer las características faciales en las imágenes de rostros.
2. Creación de base de datos con imágenes de personas que contengan distintas expresiones faciales para modelar el algoritmo de reconocimiento facial de Fisherface.
3. Creación de base de datos con imágenes de personas que contengan cejas hacia arriba y cejas en estado normal para entrenar el modelo de Support Vector Machine.
4. Definir las funcionalidades que compondrá la API que apoye la búsqueda y reconocimiento de microexpresiones faciales.
5. Verificar local y remotamente el comportamiento de la API.

1.3. Alcances

1.3.1. Sobre la Fotografía

- No debe realizarse la captura de imágenes en espacios con poca o nula luz.
- Solo capturas faciales de una única persona por imagen.
- La persona debe estar en un plano frontal a la cámara para la captura de imágenes.

1.3.2. Sobre la Interfaz de Usuario

- No se considera una interfaz gráfica, en lugar de ello sólo se emplea una interfaz de línea de comando simple dado que es un prototipo funcional de la API.
- La API responde al conjunto de imágenes al que fue entrenada.

1.4. Estructura del Documento

A continuación se describe la estructura del resto de este documento.

El Capítulo de *Antecedente* presenta los fundamentos y conceptos para comprender la ética del proyecto, las microexpresiones faciales, el procesamiento digital de imágenes y los algoritmos de reconocimiento facial y de clasificación.

El Capítulo de *Análisis del Problema* muestra los proyectos similares más populares hoy en día que dan respuesta a la problemática de identificar microexpresiones faciales y oculares. Además, se comenta la propuesta de valor del proyecto y de que manera será abordada la solución del trabajo.

El Capítulo de *Diseño General del Sistema* se muestra los componentes de la API, su arquitectura, tecnologías utilizadas, metodología de desarrollo y justificación del por qué utilizar el algoritmo de reconocimiento facial Fisherface y no otro.

En el Capítulo *Desarrollo* se detallan las etapas para la implementación de la API, el proceso de reconocimiento facial, de microexpresiones oculares y las funcionalidades que posee la API.

El Capítulo de *Pruebas y resultados* presenta los valores obtenidos de los experimentos elaborados y se verifica que el desarrollo del trabajo es adecuado.

El Capítulo *Conclusión* presenta un repaso de los aspectos más importantes en el trabajo, el aspecto teórico y de desarrollo. Por último, se comentan los trabajos a futuro para la API.

2. Antecedentes

El objetivo del capítulo es familiarizar al lector con los fundamentos y conceptos necesarios sobre el procesamiento digital de imágenes y las etapas requeridas para el análisis y extracción de microexpresiones faciales. Finalmente se expone la síntesis de las etapas y conceptos abordados.

2.1. Ética del Proyecto

2.1.1. Respeto a los Derechos y Dignidad de las Personas

El proyecto respeta y promueve el desarrollo de los derechos, la dignidad y privacidad de las personas. El derecho a la privacidad actualmente dota a las personas de "la posibilidad de que los ciudadanos titulares y propietarios de los datos que les conciernen controlen el uso y eventual abuso de los antecedentes que a su respecto sean recopilados, procesados, almacenados y cruzados computacional y telemáticamente" [17].

2.1.2. Protección de Datos de Carácter Personal

En consideración a lo anterior, la investigación hace buen uso de los datos personales proporcionados por personas que voluntariamente acceden al estudio. Los datos no son divulgados a personas externas al estudio. Dicho esto, el tratamiento de los datos de carácter personal en la base de datos se encuentra sujeta a las disposiciones de la Ley 19.628 del proyecto de Ley Chileno: *Protección de datos de carácter personal*.

Para ello, se vela por el almacenamiento de los datos sensibles que se refieren a las características físicas de las personas, en específico de capturas fotográficas de caras y expresiones oculares y la conservación de dichos datos en la base de datos generada.

La recolección de datos realizada a través del estudio es avisada a los participantes por medio de un consentimiento informado por escrito (ver Anexo A).

1. La individualización del participante.

2. El motivo y el propósito de la participación en el estudio, y

3. El tipo de dato solicitado.

El propósito y los resultados obtenidos omite las señales que permita la identificación de las personas consultadas.

Dentro de los derechos que tiene la persona que participa del estudio, el consultante puede solicitar información del estado de sus datos y tiene derecho a que se modifiquen o eliminen, lo cual es absolutamente gratuito.

2.2. Microexpresiones Faciales

El psicólogo y científico Norteamericano *Paul Ekman* precursor de la investigación en la comunicación no verbal, nombró a las expresiones faciales como expresiones universales por medio de su método llamado *Facial Action Coding System (FACS)* [12]. Definiendo que la microexpresión es una expresión facial realizada involuntariamente por la persona. Este estudio es aplicado en muchas áreas, tales como la criminología, que utiliza perfiles biométricos [25]; la psicología, que la emplea en la detección de sentimientos; y en medicina, que se utiliza en la identificación de enfermedades; entre algunas disciplinas.

2.2.1. Anatomía Facial

Aún cuando hay otras partes del cuerpo que también aportan al lenguaje corporal, es la cara la que provee de la mayoría de la información para analizar y en un solo lugar. Por lo tanto, las expresiones faciales proporcionan el mejor medio para comunicaciones no verbales de las personas. Los músculos faciales son los más importantes de la cabeza, sin ellos no existirían las expresiones.

El rostro humano esta compuesto de 44 músculos. Estos músculos faciales se dividen en cuatro grupos: *epicraneales, obriculares de los ojos, boca y nasales*, ver Figura 2.1.

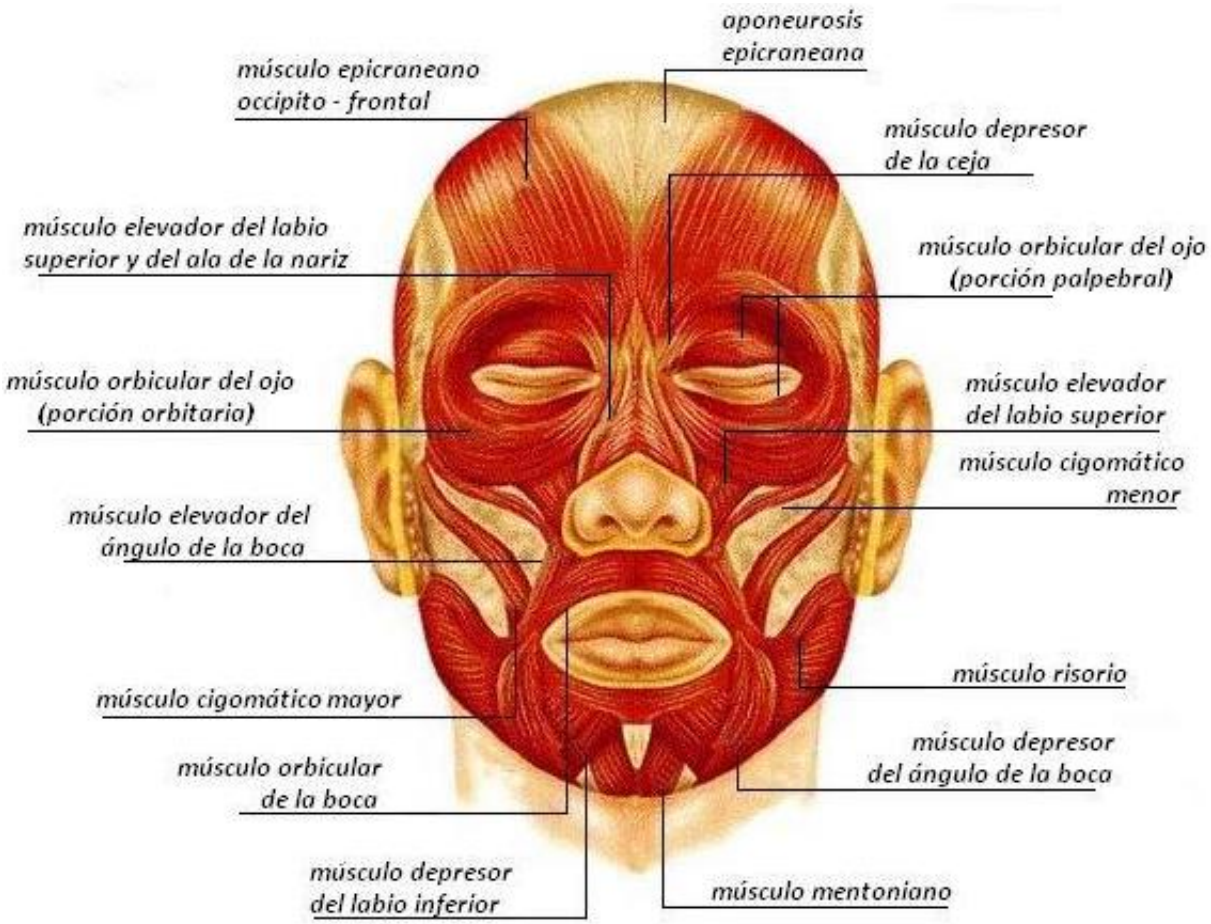


Figura 2.1: Músculos faciales.

Según las microexpresiones de Ekman, existen tres áreas que son responsables de generar los movimientos, las regiones implicadas son:

- Frente / Cejas (frente superior).
- Ojos / Párpados (frente medio).
- Boca / Mentón (frente inferior).

2.2.2. Anatomía Ocular

Los fotosensores de las cámaras se inspiran en las células fotosensibles llamadas **conos**, situadas en la retina de los animales. Los conos son las células responsables de la visión diurna. En el caso humano, son capaces de distinguir longitudes de ondas *cortas, medias*

y *largas*, del inglés *Short (S), Medium (M)* y *Long (L)*. Las células se distribuyen en la retina con densidades diferentes, tal como se muestra en la Figura 2.2.

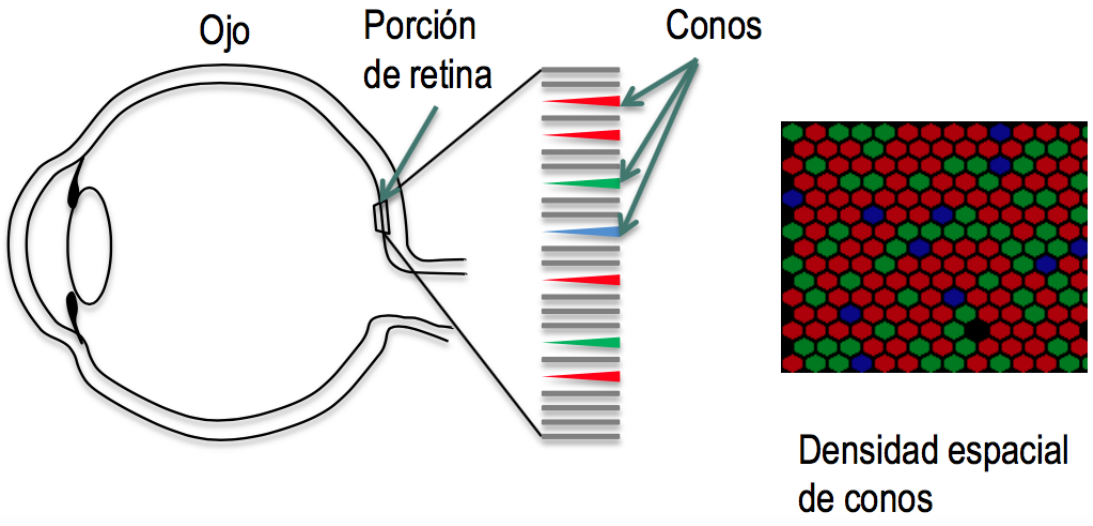


Figura 2.2: Distribución de células conos en la retina.

Existen tambiñen otro tipo de células fotosensibles, los bastones, que son responsables de la visión nocturna, es decir, en condiciones de baja luminosidad.

2.3. Procesamiento Digital de Imágenes

El procesamiento digital de imágenes responde a las técnicas que se aplican a las imágenes digitales para encontrar la información necesaria para resolver un problema en específico. La detección de objetos se produce tanto en imágenes como en vídeos, dado que es una secuencia de imágenes.

Las imágenes dadas por cámaras digitales se componen por matrices de puntos llamados píxeles. El píxel representa la menor unidad de la imagen, ver Figura 2.3.

El color de la luz se caracteriza a partir de sus componentes, que se representan como una onda con una determinada longitud. Por tanto, la luz corresponde a una periodicidad con la que se repite una señal. Por esta razón, el color es una característica que presenta la luz y el ojo humano sólo percibe un subconjunto de colores presentes en la luz.

Las luces con longitudes de ondas más pequeñas, alrededor de 400 nanómetros, son percibidas de color violeta y azul, mientras que las ondas de aproximadamente 700 nanómetros son percibida más bien de color rojo.

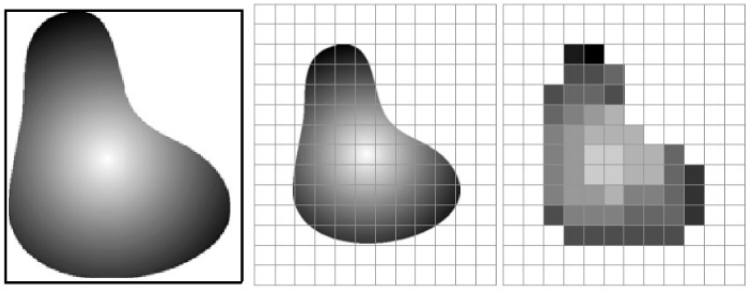


Figura 2.3: Transformación imagen a píxel.

El material de la superficie donde rebota la luz determina las onda que son reflejadas y cuáles son absorbidas por la superficie. La luz incidente viene desde una fuente de luz, que se representa por una función de longitud de onda, llamada $I(\lambda)$. Por ejemplo, una fuente de luz es el sol. La luz reflejada por un material es un porcentaje de la luz incidente se representa como una función de longitud de onda, llamada $S(\lambda)$. Entonces, la luz capturada por la cámara es el producto de estas dos longitudes de onda.

La cámara es sensible a la luz de las mismas longitud de onda percibidas por el humano para las funciones $R(\lambda)$, $G(\lambda)$, $B(\lambda)$. Por medio de estos sensores se cubre el espectro visible por el ojo humano (luz). Los valores que se asignan a cada píxel , son el resultado de la integración del producto de las tres funciones señaladas anteriormente, la función de la luz, de la superficie y del sensor, ver Ecuación (2.1).

$$(R,G,B) = \left(\int I(\lambda)S(\lambda)R(\lambda)\mathrm{d}\lambda, \int I(\lambda)S(\lambda)G(\lambda)\mathrm{d}\lambda, \int I(\lambda)S(\lambda)B(\lambda)\mathrm{d}\lambda\right) \tag{2.1}$$

2.3.1. Imagen Digital

La imagen es el resultado de la combinación de tres factores que intervienen en el proceso de captura de la imagen:

- El color de la luz que ilumina la escena.
- El material de la superficie de los objetos que son capturados y que tendrá propiedades de refracción de la luz diferentes para cada material.
- La sensibilidad de la cámara que se utilice. Generalmente la sensibilidad de la cámara pasa por tres sensores, que son sensibles a diferentes longitudes de onda de la luz que generalmente corresponde a los colores rojo, verde y azul. En general, la cámara intenta cubrir la sensibilidad a todo el espectro visual del humano.

Tras la captura de la imagen, se obtiene como resultado una matriz de píxeles de la imagen, ver Figura 2.4.

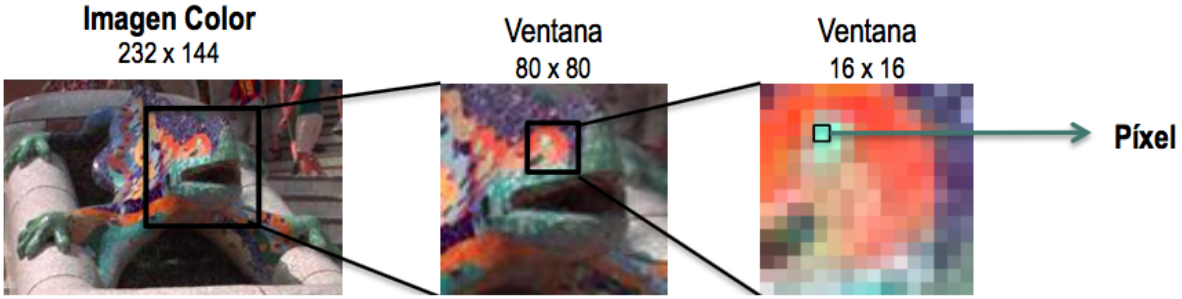


Figura 2.4: Imagen y matriz de puntos.

Las siglas del inglés *RGB* (Red, Green, Blue) corresponde a los colores rojo, verde y azul, respectivamente. Un píxel viene dado por los tres valores numéricos. Estos se encuentran en el rango de 0 y 255, donde el 0 muestra la ausencia de color y por lo tanto a colores oscuros, mientras que 255 indica la representación máxima de ese color en el píxel.

El estudio utiliza la imagen de **intensidad**, que se relaciona con la cantidad de luz que emite cada punto de la fotográfia. A mayores intensidades corresponden a zonas claras de la imagen y a menores intensidades las zonas de color oscuro, ver Figura 2.5.

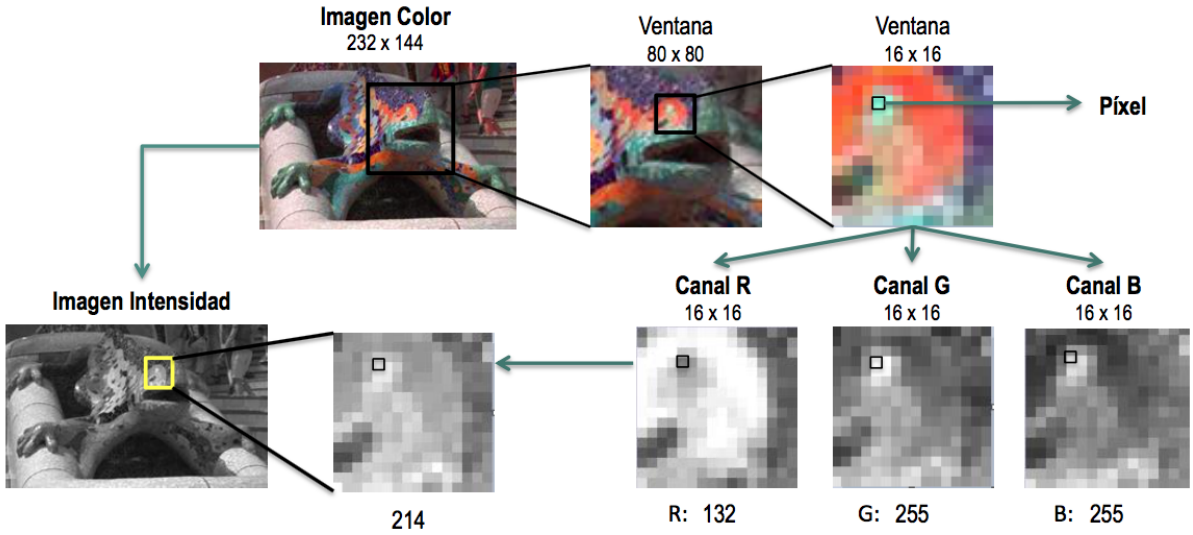


Figura 2.5: Imagen de Intensidad.

Para obtener esta imagen de intensidad se utiliza la ecuación de intensidad que suma los tres canales y se normaliza para obtener el valor máximo para un píxel en particular,

ver Ecuación (2.2):

$$F_i = (R+G+B)\left(\frac{255}{max_{i\in I}(R_i+G_i+B_i)}\right) \tag{2.2}$$

2.4. Extracción de Características

2.4.1. Detección de Características Locales y Filtros

Las características locales o *Local Features* describen las propiedades de un píxel con relación a su entorno. Estas características pueden manifestar diferentes estructuras, por ejemplo los contornos de la imagen, permitiendo identificar los contornos faciales de las personas, debido a que proveen de información clave para el reconocimiento facial. Corresponden a aquellas zonas de la imagen donde existe un alto contraste entre el entorno y la cara.

El objetivo de la detección de características locales es obtener, a partir de una imagen de origen, otra final cuyo resultado sea más adecuado para una aplicación en específica, mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella. Los objetivos que se persiguen con la aplicación de fitro son: suavizar la imagen, eliminación de ruido, realzar bordes y detectar bordes.

2.4.2. Filtros

Un filtro define una serie de pesos que se aplican a cada uno de los vecinos de un píxel determinado. Esos pesos son llamados *kernel, máscara o ventana.* Existen distintos tipos de filtros que pueden ser aplicados a las imágenes. A continuación se mencionan los más utilizados:

- **Filtros de suavizado:** Por medio de este filtro se pierde el foco en la imagen. Permite disminuir las diferencias entre los píxeles vecinos, logrando eliminar el ruido que exista en la imagen.
- **Filtros de realce:** Es lo contrario al anterior, dado que realza las diferencias entre los píxeles vecinos, provocando un efecto de enfoque, aumentando de la definición y logrando que las tonalidades aumenten.
- **Filtro de detección de bordes:** Permite localizar las zonas donde se producen transiciones bruscas de intensidad. Por medio de la detección de bordes es posible identificar fronteras. Estás fronteras pueden ser restringidas a una dirección

predeterminada, como la horizontal, vertical o diagonales. Uno de los filtros más utilizados para la detección de bordes de objetos es el *Filtro de Prewitt*.

Para aplicar los filtros se recurre a la convolución entre dos funciones, f y g, la que produce una nueva función que entrega el grado de solapamiento de la función g sobre la función f, ver Ecuación (2.3):

$$(f * g)(x, y) = \sum_{x',y'} g\left(x', y'\right) * f\left(x - x', y - y'\right) \tag{2.3}$$

Donde f representa la imagen y la función g es el filtro que representa la característica local de la se quiere medir el solapamiento con la imagen original, para lo cual k es el kernel o filtro e I es la imagen, dando como resultado la Ecuación (2.4):

$$r(x, y) = I(x, y) * k(x, y) = \sum_{x',y'} k\left(x', y'\right) * I\left(x - x', y - y'\right) \tag{2.4}$$

2.5. Máquinas de Soporte Vectorial (SVM)

Las *Máquinas de Soporte Vectorial* **(SVM)**, son una generalización de un clasificador simple e intuitivo llamado clasificador de margen máximo [16]. Es una técnica de aprendizaje automático supervisado, cuya idea principal consiste en que dado un conjunto de datos de entrenamiento, es posible etiquetar las clases y construir un modelo que prediga la clase de una nueva muestra.

Consiste en un modelo de clasificación que representa a los puntos de la muestra en el espacio, separando las clases en espacios lo más amplio posible mediante un hiperplano de separación. Existen puntos que definen el margen de separación entre las clases y el hiperplano. Estos puntos que forman un vector reciben el nombre de **vectores de soporte**.

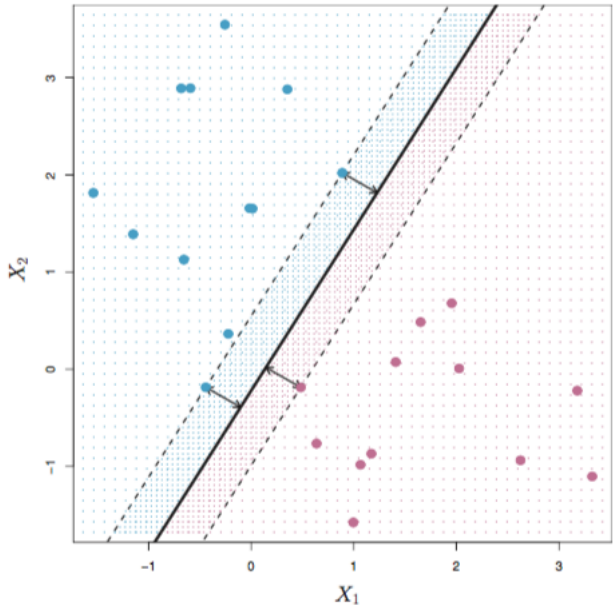


Figura 2.6: Support Vector Machine Lineal

En la Figura 2.6 hay dos clases de observaciones que se muestran en azul y purpura. El hiperplano de margen máximo se muestra como una línea continua. El **margen** es la distancia desde la línea continua a cualquiera de las líneas punteadas. Los dos puntos azules y el punto violeta que se encuentran en las líneas punteadas son los vectores de soporte, y la distancia desde esos puntos hasta el margen se indica mediante flechas.

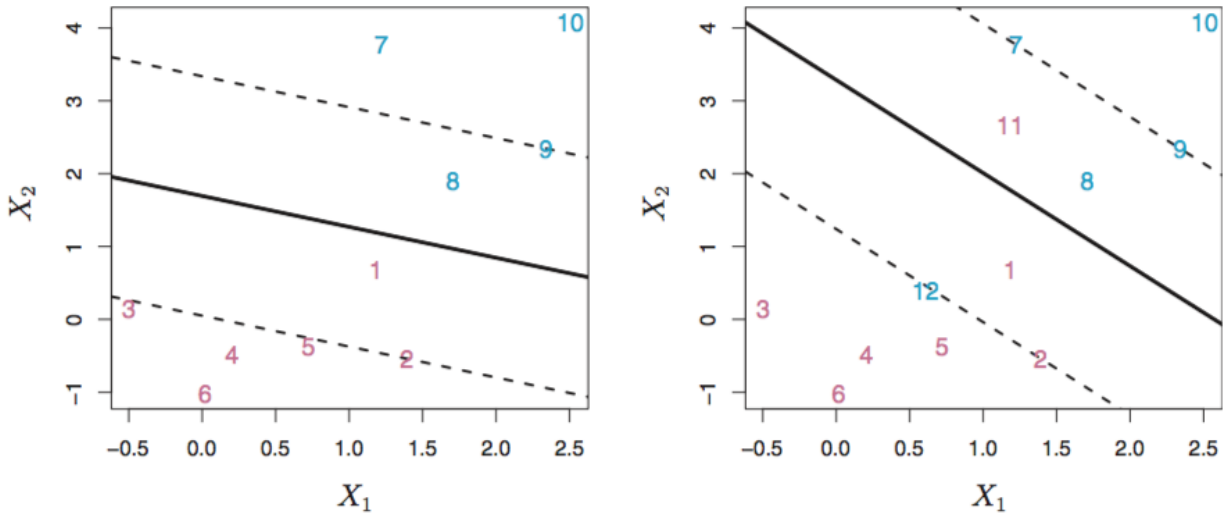


Figura 2.7: Comportamiento de observaciones en hiperplanos de SVM.

En la Figura 2.7, se muestra a la izquierda que un clasificador de SVM se ajusta a un pequeño conjunto de datos. El hiperplano se muestra como una línea continua y los márgenes se muestran como líneas punteadas. Observaciones púrpuras: las observaciones $3, 4, 5$ y 6 están en el lado correcto del margen, la observación 2 está en el margen y la observación 1 está en el lado incorrecto del margen. Observaciones azules: las observaciones 7 y 10 están en el lado correcto del margen, la observación 9 está en el margen y la observación 8 está en el lado incorrecto del margen. No hay observaciones en el lado incorrecto del hiperplano. La imagen de la derecha, es igual a la del panel izquierdo salvo que incluye dos puntos adicionales, 11 y 12. Estas dos observaciones están en el lado incorrecto del hiperplano y en el lado incorrecto del margen.

Con el fin de permitir flexibilidad, SVM posee un parámetro ajustable C, que controla la compresión entre los errores de entrenamiento y márgenes, creando un margen blando que permite algunos errores en la clasificación, como también otros más rígidos.

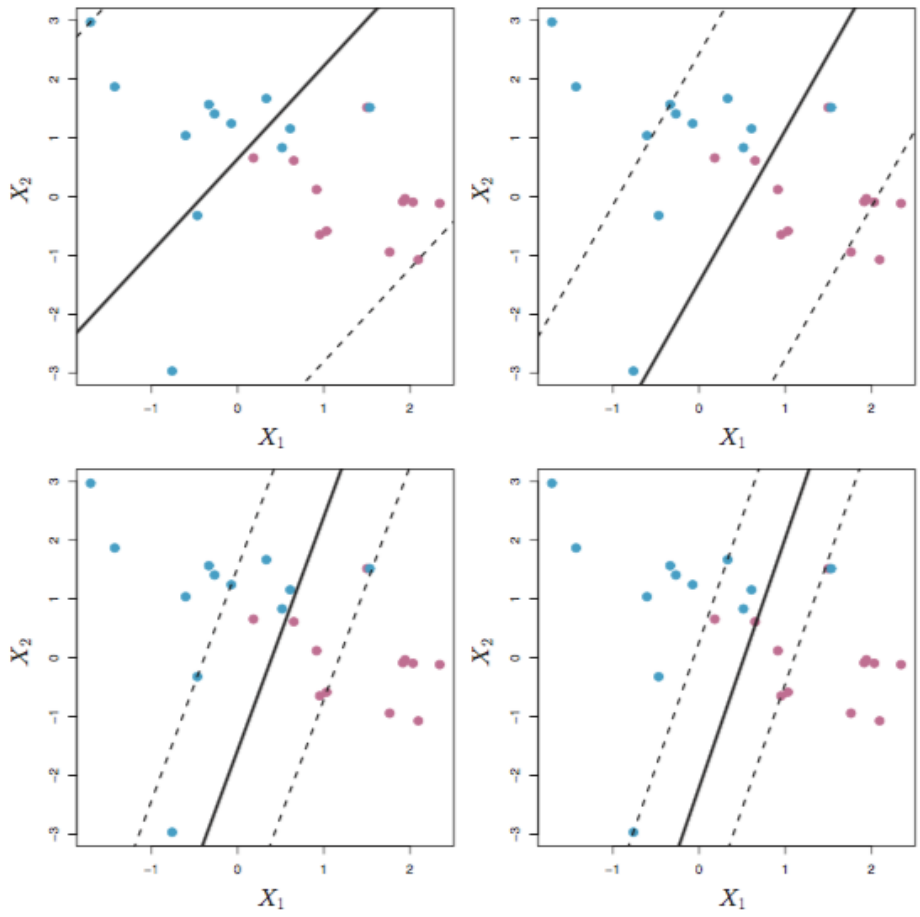


Figura 2.8: Comportamiento de SVM utilizando distintos valores de parámetro C.

En la etapa de entrenamiento, un valor grande de C equivale a un mayor ingreso de valores errores, mientras a menor valor, el margen se vuelve más tolerante al ingreso de valores erróneos, por ejemplo, en la Figura 2.8, el valor más grande de C se utilizó en el panel superior izquierdo, y los valores más pequeños se usaron en los paneles superior derecho, inferior izquierdo e inferior derecho. Cuando C es grande, existe una gran tolerancia para que las observaciones estén en el lado equivocado del margen, por lo que el margen será grande. A medida que C disminuye, la tolerancia para que las observaciones estén en el lado incorrecto del margen disminuye, y el margen se estrecha.

En la mayoría de los casos, el conjunto de entrada no es linealmente separable, y por lo tanto es necesario hacer una transformación a un espacio que sea linealmente separable, ver Figura 2.9.

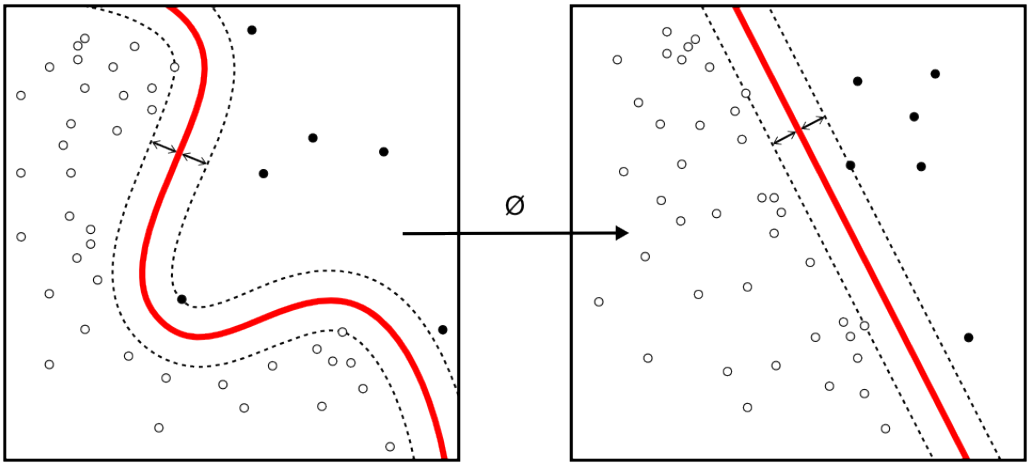


Figura 2.9: Transformación espacio linealmente no separable a espacio linealmente separable.

En varias ocasiones SVM trata con más de dos variables predictoras (clases), curvas no linealmente separables, clasificaciones en más de dos categorías, entre otros casos. Para esto, se utiliza la función **kernel** que mapea el espacio de entradas X a un nuevo espacio de características de mayor dimensionalidad $F = \{\phi(x)|x \in X\}$, donde $x = \{x_i, x_2, ..., x_n\} \rightarrow \phi(x) = \{\phi 1(x), \phi 2(x), ..., \phi n(x)\}$. El algoritmo sólo depende de los datos de entrada, para un conjunto de datos de entrenamiento $\{x_i, y_i\}$ con $i = 1, ...l, y_i \in \{-1, 1\}$ y $x_i \in \mathbb{R}^d$ a través de los productos del kernel $\phi(x_i) : \phi(x_j)$. Es decir, se busca la máxima separación entre clases. Esta función, cuando es traída de regreso al espacio de entrada, puede separar los datos en todas las clases distintas, cada una formando un agrupamiento [3] tal que

cumple:

$$K(x_i, x_j) = \phi(x_i) : \phi(x_j) \tag{2.5}$$

Tipos de kernel SVM

Tipo de SVM	Kernel	Comentarios
Polinómico	$(x^T y + 1)^p$	p especificado por el usuario.
RBF (Radial Basic Function)	$\exp^{-\frac{1}{2\sigma^2}\|x-x_i\|^2}$	σ^2 especificado por el usuario y común a todos los kernel.

Cuadro 2.1: Tipos de kernel para conjuntos no linealmente separables.

2.5.1. Comparativa RNA vs SVM

Comparación RNA vs SVM

RNA	**SVM**
Capas ocultas transforman a espacios de cualquier dimensión.	Kernel transforma a espacios de dimensión muy superior.
El espacio de búsqueda tiene múltiples mínimos locales.	El espacio de búsqueda tiene sólo un mínimo global.
El entrenamiento es costoso.	El entrenamiento es muy eficiente.
La clasificación es muy eficiente.	La clasificación es muy eficiente.
Se diseña el número de capas ocultas y nodos.	Se diseña la función kernel y el parámetro de coste C.
Muy buen funcionamiento en problemas típicos.	Muy buen funcionamiento en problemas típicos.
	Extremadamente robusto para generalización, menos necesidad de heurísticas para entrenamiento.

Cuadro 2.2: Comparación entre redes neuronales artificiales (*RNA*) y Support Vector Machine (*SVM*).

En muchas aplicaciones, los clasificadores SVM han mostrado tener un gran desempeño, más que las máquinas de aprendizaje tradicional tales como las redes neuronales

[5] y han sido introducidas como herramientas poderosas para resolver problemas de clasificación. El Cuadro 2.2 muestra la comparativa entre RNA y SVM [6].

2.6. Algoritmo de Reconocimiento de Caras

El proceso de reconocimiento facial consiste en tomar una imagen de dos dimensiones, a filas y b columnas, la que se transforma en un vector unitario contenido en un espacio de imágenes n-dimensionales ($n = a \times b$). Posteriormente, se obtiene la imagen promedio y se proyecta el vector resultante en un subespacio de menor dimensión utilizando uno de los métodos de reducción de dimensión (extracción de características). Esta proyección es comparada con la proyección de un conjunto de imágenes de una base. La clase del vector más similar es el resultado del proceso de reconocimiento [22].

2.6.1. Eigenfaces

Eigenfaces realiza una proyección lineal del espacio de imágenes a un espacio de características de menor dimensión. Esta reducción se realiza utilizando la técnica de *Análisis de Componentes Principales* (PCA), la cual consigue una reducción de la dimensión proyectando los datos sobre la dirección que maximice la distribución total sobre todas las clases.

En primer lugar se considera un conjunto de N imágenes con valores en el espacio de imágenes n-dimensional:

$$\{x_i\} \qquad i = 1, 2, ..., N \tag{2.6}$$

Se asume además que cada una de las imágenes pertenece a una de las c clases $\{X_1, X_2, ..., X_c\}$. Asimismo se considera una transformación lineal que lleva al espacio de imágenes original de n dimensiones al espacio de características de dimensión m, donde $m < n$. Los nuevos vectores de características $y_k \in \mathbb{R}^m$ son definidos por la siguiente transformación lineal:

$$y_k = W^T x_k \qquad k = 1, 2, \ldots, N \tag{2.7}$$

donde $W \in \mathbb{R}^{n \times m}$ es una matriz de columnas ortogonales. Se define además la matriz de distribución total S_T como:

$$S_T = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T \tag{2.8}$$

donde $\mu \in \mathrm{R}^n$ es la media de todas las imágenes de la Ecuación (2.6). Luego de aplicar la transformación lineal W^T, la distribución de los vectores de características $\{y_1, y_2, ..., y_N\}$ es $W^T S_T W$. Se toma aquella proyección W_{opt} que maximiza el determinante de la distribución total de la matriz de las imágenes proyectadas, esto es

$$W_{opt} = \underset{\mathrm{W}}{\operatorname{argmax}} |W^T S_T W| \tag{2.9}$$

$$= [w_1 \, w_2 \, \ldots \, w_m]$$

donde $\{w_i \,|\, i = 1, 2, \ldots, m\}$ es el conjunto de vectores propios n-dimensionales de S_T correspondiente a los mayores m vectores propios. Dichos vectores propios tienen la misma dimensión que las imágenes originales y se les denomina eigenfaces.

2.6.2. Fisherfaces

Fisherfaces utiliza el *Discriminante Lineal de Fisher* (FLD) para la reducción de dimensiones. Busca maximizar la varianza de las muestras entre clases (entre personas) y minimizarla entre muestras de la misma clase (de la misma persona). Este método selecciona el W de la Ecuación (2.6) de manera que los cocientes entre la distribución entre las clases y la distribución intra-clases sea máxima. Para esto se define la matriz S_B de distribución entre clases como:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{2.10}$$

y la matriz S_W de distribución intra-clases

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} N_i (x_k - \mu_i)(x_k - \mu)^T \tag{2.11}$$

donde μ_i es la imagen media de la clase X_i y N_i es el número de imágenes en la clase X_i. Si la matriz S_W es no singular, la proyección W_{opt} se elige como la matriz con columnas ortonormales que maximiza el cociente del determinante de la matriz de distribución entre clases de las imágenes proyectadas y el determinante de la matriz de la distribución intra-clases de las imágenes proyectadas, esto es

$$W_{opt} = \underset{\mathrm{W}}{\operatorname{argmax}} \frac{|W^T S_B W|}{|W^T S_W W|} \tag{2.12}$$

$$= [w_1 \, w_2 \, \ldots \, w_m]$$

donde $\{w_i \,|\, i = 1, 2, \ldots, m\}$ es el conjunto de valores propios de S_B y S_W correspondiente a los m mayores valores propios $\{\lambda_i | i = 1, 2, \ldots, m\}$, esto es

$$S_B w_i = \lambda_i S_w w_i \qquad i = 1, 2, \ldots, m \tag{2.13}$$

Se observa entonces, que a lo sumo se tienen $c-1$ valores propios distintos de cero, y por lo tanto el límite superior de m es $c-1$, donde c es el número de clases. Para el problema de reconocimiento de caras, se tiene que la matriz $S_W \in \mathrm{R}^{n \times n}$ es siempre singular, dado que el rango de S_W es a lo sumo $N-c$, y en general, el número de imágenes de entrenamiento N, es mucho más chico que el número de píxeles de cada imagen n. Por lo tanto puede ser posible elegir una matriz W tal que la distribución intra-clases de las imágenes proyectadas pueda ser exactamente cero. Como alternativa entonces, al criterio establecido en la Ecuación (2.12), se proyecta el conjunto de imágenes a un espacio de menor dimensión, de manera que la matriz resultante de la distribución intra-clases S_W sea no singular. Utilizando *PCA* se realiza la reducción de dimensiones del espacio de características a $N-c$ y luego, aplicar *FLD* definido en la Ecuación (2.12) para reducir la dimensión a $c-1$. De esta manera W_{opt} es dado por

$$W_{PCA}^T = W_{FLD}^T W_{PCA}^T \tag{2.14}$$

$$W_{PCA} = \underset{\mathrm{W}}{\mathrm{argmax}} |\mathrm{W^T S_T W}|$$

$$W_{FLD} = \underset{\mathrm{W}}{\mathrm{argmax}} \frac{|\mathrm{W^T W_{PCA}^T S_B W_{PCA} W}|}{|\mathrm{W^T W_{PCA}^T S_W W_{PCA} W}|}$$

A las columnas de esta matriz se les refiere como Fisherfaces.

3. Análisis del Problema

3.1. Proyectos Similares

Hasta la fecha existen alrededor de 50 APIS de reconocimiento facial, la mayoría de pago, las que son gratuitas en cambio, presentan grandes limitantes. A continuación se listan las más populares en el mercado:

- **FaceR**: De la empresa *Animetrics*, esta API permite realizar una reducida cantidad de operaciones. La obtención de la *key* para consumir la API es lenta por parte del proveedor (alrededor de tres semanas).
- **Face++**: Desarrollada por *Megvii*, proporciona planes de pago y uno gratuito, este último con la limitante de poseer mil imágenes en la base de datos, sin la posibilidad de ser ampliada.
- **SkyBiometry**: Empresa del mismo nombre que su API, dispone de una documentación robusta e integración para diversos lenguajes de programación, sean estos *Python, C#, Ruby, PHP, JavaScript, IOS*, entre otros.
- **Lambda API**: De *Lambda Labs*, se limita a entregar una documentación con la respuesta de los métodos, no así, formas de conexión, no muestra cuando fue su última actualización, posee planes de pago y planes gratuitos.

Las APIS comerciales no abordan la detección de microexpresiones oculares. A continuación, el Cuadro 3.1 muestra el resumen de las principales características que debe poseer una API de reconocimiento facial. El resumen integra las APIS descritas anteriormente, destacando los componentes que se integran en la API propuesta en este trabajo, llamada **FME_API** (Facial Microexpressions API):

Acciones	**FaceR**	**Face ++**	**SkB**	**Lambda Labs**	**FME_API**
Contornos Facial Frontal.	SI	SI	SI	SI	SI
Contorno Facial Lateral.	NO	SI	NO	NO	NO
Documentación	SI	SI	SI	SI	SI
Detección Microexpresiones Oculares.	NO	NO	NO	NO	SI
Reconocimiento Facial	SI	SI	SI	SI	SI
Precisión (Confianza) Reconocimiento	NO	NO	NO	SI	SI

Cuadro 3.1: Comparación de APIS de reconocimiento facial.

3.1.1. Propuesta de Solución

Dada la problemática comentada en la Sección 1.1.1, se ha definido el uso de una API para que los programadores puedan hacer uso de este servicio a fin de desarrollar soluciones en la autentificación de personas en modo no presencial de manera remota.

FME_API integra desde el reconocimiento facial hasta la detección de microexpresiones oculares.

La solución del problema está dividida en tres etapas: Reconocimiento facial, reconocimiento de microexpresiones oculares y la integración de estos módulos en una API.

FME_API es capaz de recibir imágenes de rostros de personas y determinar (en caso que se encuentre registrada en la base de datos) el nombre de la persona, además de indicar si se encuentra con las cejas arriba o las cejas en un estado normal, estos aspectos son abordados en los siguientes capítulos.

4. Diseño General del Sistema

El capítulo tiene por objetivo dar a conocer de forma global el funcionamiento de la API junto a la arquitectura, componentes y subcomponentes. Posteriormente, se describen las tecnologías utilizadas en la implementación. Por último, se describe la metodología de desarrollo de software utilizada y los fundamentos de la elección del algoritmo de reconocimiento facial.

4.1. Componentes de API REST

La **Interfaz de Programación de Aplicaciones**, abreviada como **API** del inglés *Application Programming Interface*, es un conjunto de funciones y procedimientos para ser utilizado por otro software como una capa de abstracción.

Describe la forma en que los programas y sitios webs intercambian datos, normalmente en formato **JSON** (JavaScript Object Notation) o **XML** (eXtensible Markup Language).

Se utiliza para ofrecer datos a aplicaciones y permite a los desarrolladores simplificar la programación, ya que tiene un formato estándar, además de poder consumir datos de otras aplicaciones [7].

La **Transferencia de Estado Representacional**, abreviada como **REST** del inglés *Representational State Transfer*. Es un tipo de arquitectura de desarrollo web que se apoya totalmente en el protocolo HTTP. Se compone de una lista de reglas que se deben cumplir en el diseño de la arquitectura de la API.

REST no es un estándar ni un protocolo, sino una serie de principios de arquitectura [21]. Las peticiones realizadas son independientes, o sea, que cada petición debe llevar todo el contenido necesario para ser realizadas, y es el usuario el encargado de mantener el estado en su propia aplicación en base a lo retornado por la API. El funcionar bajo un protocolo sin estado permite a los servidores ahorrar recursos al no ser necesario almacenar información de la sesión de usuario y facilita las labores de desarrollo.

4.1.1. Diseño de API

Joshua Bloch (Ingeniero de Software en Google) menciona en su publicación [4] que un buen diseño de API tiene que tener las siguientes características:

- Fácil de aprender.
- Fácil de usar, incluso sin documentación.
- Difícil de mal usar.
- Fácil de leer y mantener el código que utiliza.
- Fácil de expandir.
- Apropiada para la audiencia.

El proceso más importante en el desarrollo de software es el diseño, que consiste en *"El proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o componente que resulte de este proceso"* [IEEE610.12-90]. La Figura 4.1 muestra la arquitectura que se utiliza en el desarrollo de FME_API. Destaca el componente de **Reconocimiento Facial**, responsable de reconocer la imagen de la persona en alguna persona registrada en la base de datos.

Para validar que la persona sea quien dice ser, es necesario someter a la persona a pruebas de microexpresiones oculares, es aquí donde el componente **Microexpresiones Oculares** determina si la persona se encuentra con sus cejas arriba o en estado normal. El usuario, por medio de peticiones en formato tipo JSON obtiene respuesta a cualquiera de los componentes de la API, ver Figura 4.1.

4.2. Python

Es un lenguaje de programación creado a finales de los años 80 por *Guido van Rossum* en el *CWI, Centrum Wiskunde and Informatica* (Centro para las matemáticas y la Información) [1]. El nombre del lenguaje proviene de la afición de su creador a los humoristas británicos *Monty Python* [13].

Es un lenguaje multiparadigma dado que es *orientado a objetos* e *imperativo*, ya que son un conjunto de instrucciones que le indican al computador cómo realizar una tarea. Es un lenguaje de programación *procedural*, se basa en el uso de funciones matemáticas y que enfatiza los cambios de estado mediante la mutación de variables.

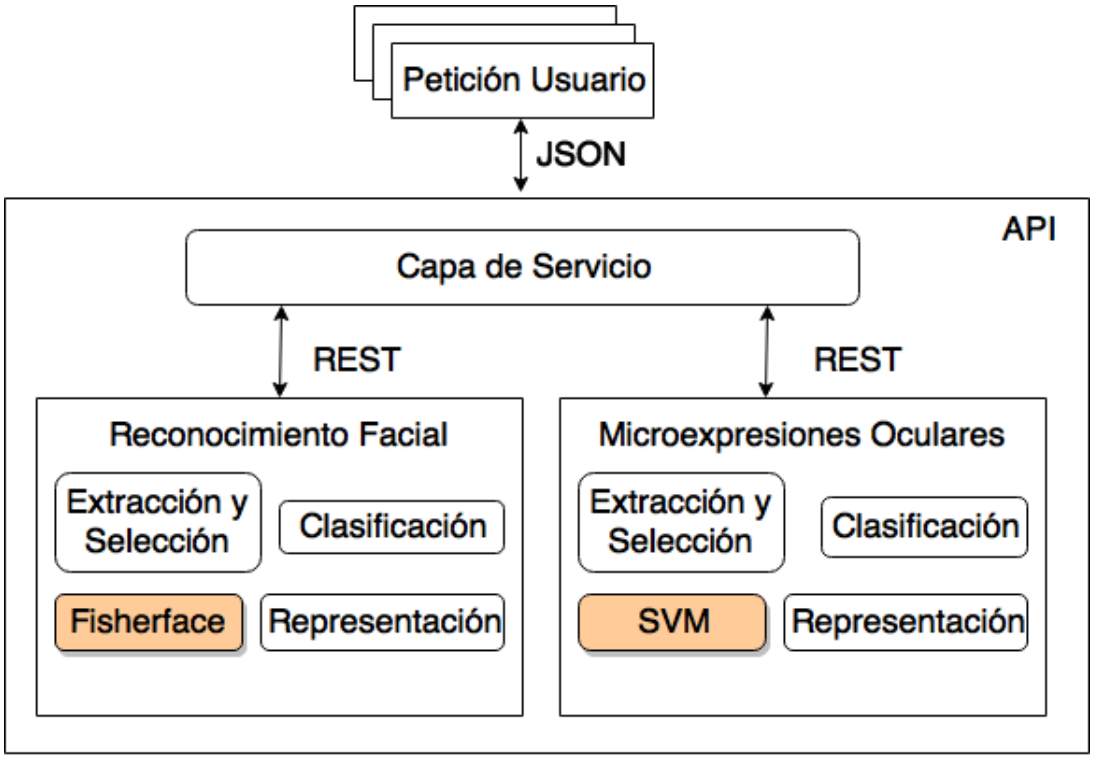


Figura 4.1: Arquitectura de la API.

4.2.1. Flask

Es un microframework de *Python* desarrollado por *Armin Ronacher* que posibilita crear aplicaciones web en reducidas líneas de código. Es *Open Source* y esta bajo licencia *BSD* (Berkeley Software Distribution) y permite ser usada libremente para propósitos comerciales y académicos [23].

Flask, a diferencia de *Django* y *Pyramid* (framework de Python para desarrollo web), reduce la cantidad de módulos para tareas comunes en el desarrollo web.

Flask incluye un servidor web de desarrollo sin la necesidad de instalar *Apache* o *Nginx*. Proporciona un depurador y soporte integrado para pruebas unitarias. Al no necesitar de dependencias, es ideal para el desarrollo ágil.

4.2.2. Scikit-Learn

Es una librería de Machine Learning gratuita para *Python*, desarrollada por *David Cournapeau* (ingeniero de Google) el 2007 y publicada el 1 de Febrero de 2010. Además es de distribución BSD [8].

Scikit-learn posee módulos de clasificación (*SVM, Naive Bayes, KNeighbors, Random*

Forest, entre otros), clustering (*KMeans, MeanShift, Spectral Clustering*, entre otros), regresión (*SGD Regressor, Lasso, SVR*, entre otros.) y reducción de dimensión (*Randomized PCA, Isomap, LLE*, entre otros).

4.3. OpenCV

OpenCV es una librería libre de visión artificial de distribución BSD. Posee interfaces para *C++, C, Python* y *Java* [26].

Es multiplataforma, se encuentra disponible para *GNU/Linux, Mac OS X, iOS, Android* y *Windows*. Esta disponible en el sitio web oficial del proyecto: `http://opencv.org`.

Contiene más de 500 funciones que abarcan una gran área de visión por computadora, tales como reconocimiento de objetos, reconocimiento facial, calibración de cámaras, visión estérea y visión robótica. La documentación se encuentra disponible en `http://docs.opencv.org/2.4/index.html`.

4.4. Dlib

Es un conjunto de herramientas de C++ que contiene algoritmos de Machine Learning, de distribución BSD. Posee una documentación completa y precisa de todas las clases y funciones [19]. Provee una gran cantidad de programas de ejemplo.

Es multiplataforma, se encuentra disponible para *GNU/Linux, Windows* y *Mac OS X*. No requiere otros paquetes para utilizar la biblioteca.

4.5. Metodología de Desarrollo

Scrum es el nombre que recibe la metodología de desarrollo de software ágil introducido por *Ikujiro Nonaka* e *Hirotaka Takeuchi* a principio de los años 80.

Scrum define roles y prácticas para definir el proceso de desarrollo que se ejecutará durante el proyecto.

El periodo completo de desarrollo se divide en etapas cortas de una a cuatro semanas, las que se denominan *Sprint*, en cada periodo se produce un producto potencialmente entregable, de manera que cuando el cliente (*Product Owner*) lo solicite sólo sea necesario un esfuerzo mínimo para que el producto este disponible para ser utilizado.

El principio clave de Scrum es el reconocimiento que durante un proyecto, los clientes pueden cambiar de idea sobre lo que quieren y necesitan [18]. Tras los requerimientos

obtenidos por el cliente se generá el documento de requisitos, ver Anexos A.

4.6. Elección Algoritmo de Reconocimiento Facial

Guillermo Ottado en su publicación [22] menciona los dos experimentos realizados para evaluar el desempeño de los algoritmos de reconocimiento facial de *Eigenfaces* y *Fisherfaces*. Planea que a través de aplicar variaciones en la iluminación en función del ángulo de la fuente de luz y asumiendo el perfil frontal de la cara. Se obtiene por resultado que las imágenes presentan peor iluminación cuando mayor sea este ángulo.

El primer experimento evalúa la tasa de reconocimiento obtenido para distintos valores de m (número de características faciales de la imagen, por ejemplo ojos, nariz, boca, etc.), tomando como resultado el promedio de reiterar este procedimiento 5 veces de manera de asegurar que el resultado obtenido no responda a una elección particular de los conjuntos de entrenamiento y prueba. Los resultados se muestran en Anexo B.1.

El resultado indica que *Fisherface* es superior a los otros dos métodos (*Eigenfaces* y Eigenface con variaciones de iluminación denominado *Eigenfaces s/3*), haciéndose esta diferencia más evidente cuando menor sea el número de características que se consideren.

El segundo experimento utilizó imágenes de caras en distintas poses y se evalúa la tasa de reconocimiento para distintos valores de m. Los resultados se muestran en el Anexo B.2. Donde, *Fisherface* es superior al resto, aunque tanto este método como *Eigenfaces* obtienen un buen desempeño, superior al 90 %. Se observa además, que cuanto mayor es el número de características que se consideran, el desempeño de los algoritmos es similar.

5. Desarrollo

En este capitulo se detallan las etapas para la implementación de la API:

- Reconocimiento facial.
- Reconocimiento de microexpresiones oculares.
- Desarrollo de API.

El capítulo detalla los procesos del algoritmo de *Fisherface* y del clasificador *SVM*. También se muestra cómo se extraen las características faciales de la imagen, cómo se entrenan los modelos, su estructura y de qué forma predicen estos ante imágenes de prueba.

Se explican además, las características que poseen las base de datos de ambos modelos, la distribución de directorios y archivos.

Por último, se indican las propiedades de la API, funcionamiento y métodos. Esto es, cómo el usuario puede realizar las peticiones y los distintos formatos de salida que la API entrega.

5.1. Computador Servidor

A continuación se listan las características del computador servidor utilizado para las pruebas independientes para el reconocimiento facial y de microexpresiones oculares, asímismo de las tecnologías utilizadas para la implementación de la API.

Se utilizó el software **VMware Fusion** v. pro `8.5.7` (5528452) como virtualizador, el cual fue configurado con las siguientes tecnologías:

- **Sistema Operativo**: Linux - Fedora v. 25.
- **Nombre Procesador**: Intel Core i5.
- **Velocidad Procesador**: 2,5 GHz.
- **Cantidad Procesadores**: 1.
- **Cantidad total de núcleos**: 2.
- **Memoria**: 8 GB 1600 MHz DDR3.
- **Disco de Arranque**: Macintosh SSD.

  Python: v. `2.7.13`.

  - PIP: v. `9.0.1`.
  - Sttckit-learn: v. `0.19.1`.
  - Flask: v. `0.12.2`.
- **OpenCV**: v. `2.4.13`.
- **Dlib**: v. `19.7`.
- **Ambiente de programación**: Sublime Text: v. `3.0`.
- **Controlador de versiones**: Para mantener la privacidad del código, se utiliza un repositorio privado online en GIT: `www.bitbucket.com`.

El detalle completo de las dependencias de la API se encuentran en Anexo C.

5.2. Captura de Imágenes

5.2.1. Condiciones del Entorno y Características de la Cámara

Los espacios donde se captura las imágenes pueden incluir variaciones de brillo o sombra, por lo que las capturas son tomadas en un ambiente controlado (pared de tonos claros y sin fotografías de fondo).

Al momento de estar tomando fotografías, la persona no puede realizar movimientos bruscos, sean estos giros rápidos de cabeza hacia la izquierda, derecha o en algún ángulo, ya que el proceso de *extracción de características faciales* no puede identificar los puntos

de referencias faciales necesarios para que los modelos de reconocimientos puedan ser entrenados adecuadamente.

Las capturas deben ser tomadas en un plano frontal para obtener la totalidad de los puntos de referencias faciales.

Se utiliza la cámara web que viene integrada al computador que posee una resolución HD de 720p.

Las capturas de imágenes son convertidas a escala RGB, las que son almacenadas con extensión **PGM** (*Portable Graymap*). PGM es un formato en escala de grises que utiliza 8 bits por píxel, donde el valor máximo de gris es 255 [20].

Para el reconocimiento facial, las capturas realizadas se almacenan en un directorio que tiene como nombre, el nombre de la persona de quien fueron capturadas las imágenes. De esta manera es posible diferenciar las imágenes de una u otra persona.

En el caso de la detección de microexpresiones oculares, existen dos directorios de imágenes que contienen las microexpresiones oculares de personas adultas de todos los continentes con cejas arriba y en estado normal [10].

El directorio de `Cejas_Arriba` contiene 584 imágenes y el directorio `Cejas_Normal` contiene 522 imágenes.

5.3. Base de Datos

La API necesita dos bases de datos, una de ellas destinada al reconocimiento facial y otra para el reconocimiento de microexpresiones oculares.

5.3.1. Reconocimiento Facial

Se cuenta con un directorio que actua de base de datos de personas, dentro de él existen sub-directorios que poseen como nombre, el nombre y apellido de la persona, para comprender la organización de la base de datos, véase el Anexo D.1. La documentación de OpenCV menciona que como mínimo se necesita 10 imágenes por persona para utilizar el algoritmo de Fisherface, no asegura que la persona es reconocida usando una baja cantidad de fotografías. No hay claridad de los ambientes y reglas claras para obtener un mejor rendimiento en el reconocimiento facial. Se menciona también que el modelo de entrenamiento de Fisherface requiere de imágenes que posean como mínimo una resolución de 10x10 píxeles en formato .pgm y en la base de datos debe haber como mínimo 2 registros de personas en la base de datos [27].

Hay que señalar que el número de imágenes en los directorios de las personas no asegura un mejor rendimiento del algoritmo. Este va a depender del ángulo de la cara al momento de capturar la fotografía, de la pose, iluminación, expresiones registradas y el ambiente de trabajo usado para el registro de las imágenes.

5.3.2. Microexpresiones Oculares

Se cuenta con una base de datos de microexpresiones oculares, la que considera expresiones de personas con cejas arriba y cejas en estado normal que son almacenadas en formato .jpg y .png Para ver la organización de la base de datos, revisar el Anexo D.2.

La base de datos utilizadas es *Chicago Face Database* [10]. Además se incluyen imágenes capturadas localmente para poder poblar la base de datos de microexpresiones oculares.

Los directorio de *Cejas_Arriba* y *Cejas_Normal* poseen imágenes de la forma, mostrada en la Figura 5.3.2.

(a) Cejas Normal. (b) Cejas Arriba.

Figura 5.1: Imágenes de la base de datos microexpresiones oculares.

5.4. Etapa 1: Reconocimiento Facial

El reconocimiento facial consta de diversos procesos que se muestran en el siguiente diagrama, de la Figura 5.2.

- Dada una imagen, se detecta que hay una cara sin identificarla. Si se trata de un vídeo, se evalua imagen por imagen. Se realiza el seguimiento de la cara, entregando la localización y la escala a la que se encuentra.

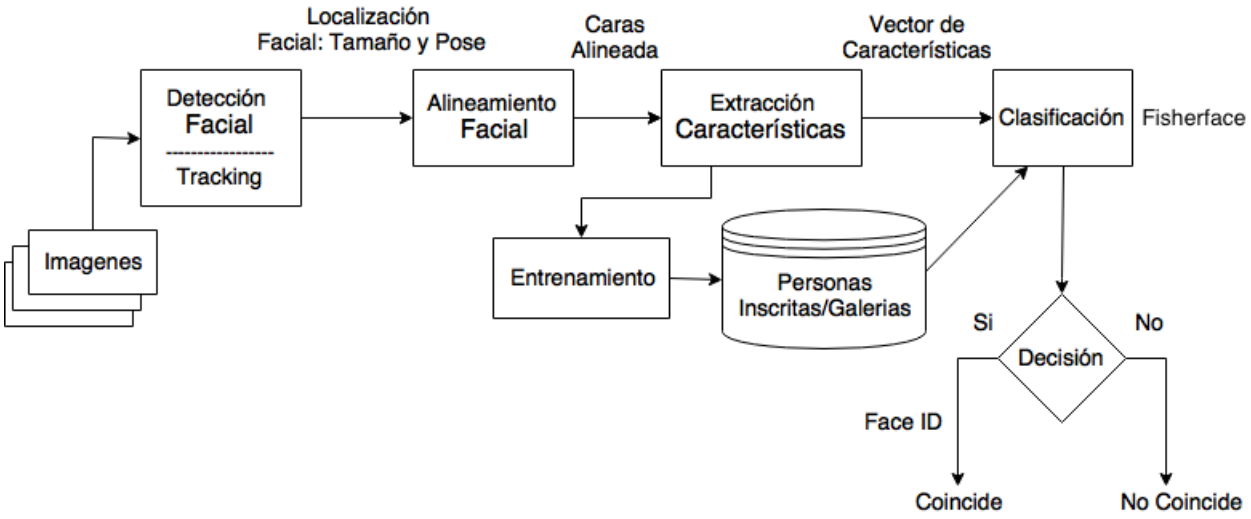


Figura 5.2: Proceso de reconocimiento facial.

- De la cara, se extraen los puntos de referencias faciales (son los puntos de tipo (X, Y) que definen el contorno de la cara, ojos, cejas nariz, boca y labio), se almacena en una matriz de $68x2$ de tipo *float*. Cada fila del arreglo esta compuesto por la ubicación de cada punto en la imagen. Por medio de los puntos de referencia faciales es posible identificar sectores de la cara. esto se ilustra en la Figura 5.3.

  Donde:

  - **Contorno de la cara:** puntos $0 - 16$.
  - **Ceja izquierda:** puntos $17 - 21$.
  - **Ceja derecha:** puntos $22 - 26$.
  - **Nariz:** puntos $27 - 35$.
  - **Ojo izquierdo:** puntos $36 - 41$.
  - **Ojo derecho:** puntos $42 - 47$.
  - **Labio superior:** puntos $48 - 59$.
  - **Labio inferior:** puntos $60 - 67$.

- Los puntos de referencia facial cambian a medida que la cara se mueve a diferentes partes de la cámara. Podría estar realizando la misma expresión en la parte superior izquierda de la imagen como en la parte inferior derecha en otra imagen. La relación entre las coordenadas son invariantes a la ubicación, lo que significa que no importa en que parte del cuadro se encuentre la cara detectada.

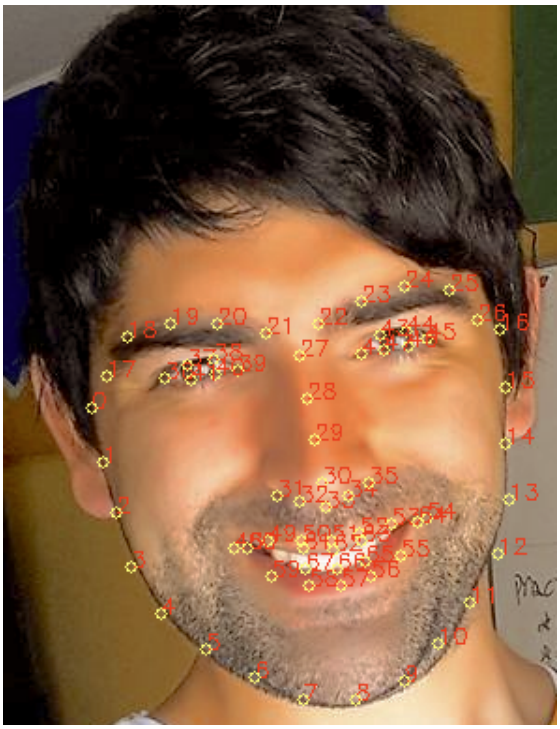


Figura 5.3: Puntos de referencias faciales.

Para alinear la cara, se calcula la media de los dos ejes, lo que resulta en una especie de "centro de gravedad" de todos los puntos de referencia facial. Entonces se obtiene la posición de todos los puntos relativos a este punto central, ilustrada en la Figura 5.4.

Cada línea de color rojo tiene una magnitud (distancia entre los puntos) y una dirección (ángulo con respecto a la imagen horizontalmente, es decir un vector).

No es posible tomar la punta de la nariz como punto de referencia, ya que existen narices cortas o con deformaciones, por lo cual habría que agregar una varianza extra. El centro de gravedad se desplaza cuando la cabeza se aleja o acerca de la cámara.

Para disminuir el uso de recursos computacionales se utiliza escala de grises y se reduce la resolución de las imágenes.

(a) Añade "centro de gravedad" en la cara, se muestra con un punto de color azul.

(b) Se trazan lineas de color rojo ente el punto central y cada punto de referencia facial.

Figura 5.4: Normalización de imagen facial.

- Una vez alineada la cara, es necesario extraer sus características, que son pequeños fragmentos de información que la describen (nariz, boca, ojos, etc.), las cuales proporcionan datos para distinguir entre las caras de diferentes personas según variaciones geométricas.

- Cada persona posee un identificador asignado por el modelo de Fisherface. Este inicia del número 0 en adelante. La siguiente muestra posee 3 directorios, con distinta cantidad de imágenes. El modelo resultante denominado `modelo_reconocimiento.pkl` indica la matriz de identificadores (etiquetas) de la forma, mostrada en la Figura 5.5:

- El vector de características extraído de una imagen se compara con los vectores de características de cada una de las personas que existe en la base de datos. El porcentaje de similitud entre vectores de características se llama **confianza**.

```
labels: !!opencv-matrix
rows: 490
cols: 1
dt: i
data: [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   ...,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
   1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2,
   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
   ...,
   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
   2, 2, 2, 2, 2, 2, 2 ]
```

Figura 5.5: Modelo de reconocimiento facial.

5.5. Etapa 2: Reconocimiento Microexpresiones Oculares

Una vez reconocida a la persona en la etapa de reconocimiento facial, y con el objetivo de validar de que se trata realmente de la persona, se aplican pruebas de levantamiento de cejas, para lo que se utiliza un clasificador SVM con kernel lineal con el fin de predecir si la persona levanta o no las cejas. El diagrama de la Figura 5.6 presenta el proceso que responde a este objetivo.

5.5.1. Preparación de Datos y Entrenamiento

La base de datos se compone de imágenes con cejas arriba y cejas en estado normal. Para crear y entrenar el modelo, se utiliza un clasificador SVM con kernel lineal y parámetro de coste $C = 1{,}0$.

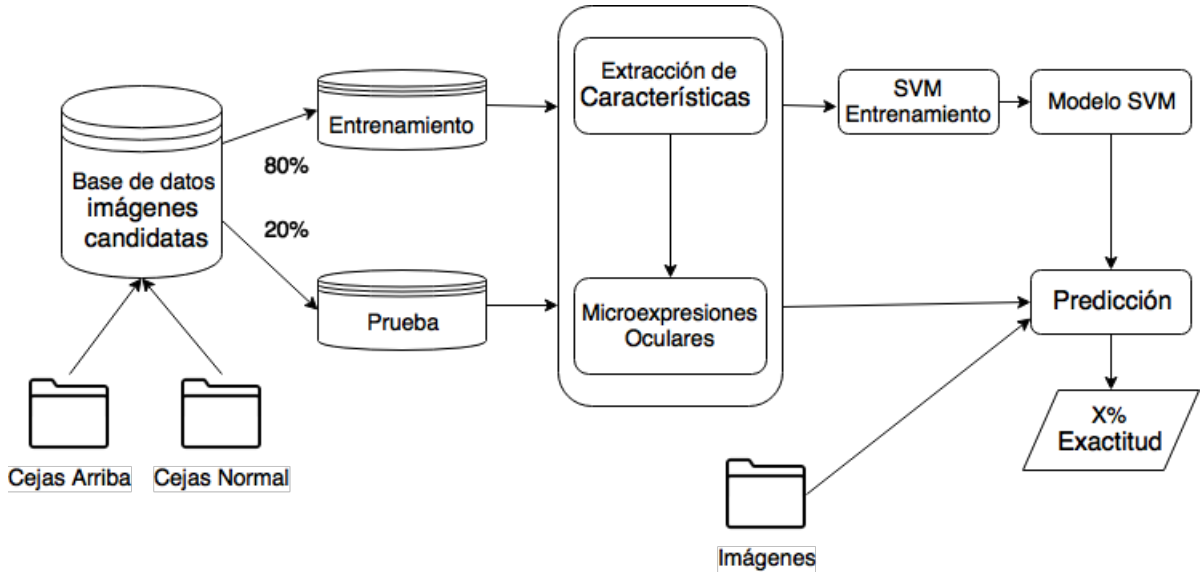


Figura 5.6: Proceso de reconocimiento microexpresiones oculares.

5.5.2. Extracción de Microexpresiones Oculares

Una vez obtenidos los datos de entrenamiento se realiza el proceso de extracción de características. Para ello se utiliza el método visto para el reconocimiento facial. Los puntos de referencias faciales se centran en el sector de ojos y cejas, obteniendo las microexpresiones oculares de cada una de las imágenes.

5.5.3. Entrenamiento, Clasificación y Predicción SVM

El proceso de entrenamiento considera las imágenes como puntos en el hiperplano de separación. *Scikit-learn* define por defecto las clases de los objetos que son entrenados, otorgando a los puntos del directorio *Cejas_Arriba* la clase 0 mientras que los puntos del directorio *Cejas_Normal* la clase 1. SVM construye un modelo capaz de predecir si una imagen nueva (cuya categoría se desconoce) pertenece a una u otra clase.

Para determinar la exactitud del modelo de clasificación se utiliza validación cruzada (*cross-validation* por sus siglas en ingles). Esta técnica se utiliza en entornos donde el objetivo principal es la predicción cuando se quiere estimar la precisión de un modelo [11]. Se utiliza un 80 % de las imágenes de la base de datos para entrenar el modelo y 20 % para probar y determinar la precisión del modelo.

Las Figuras 5.7 y 5.8 muestran las predicciones del clasificador para cada FPS (cuadros por segundo) que la cámara captura en tiempo real, indicando que se trata de cejas arriba y de cejas en estado normal.

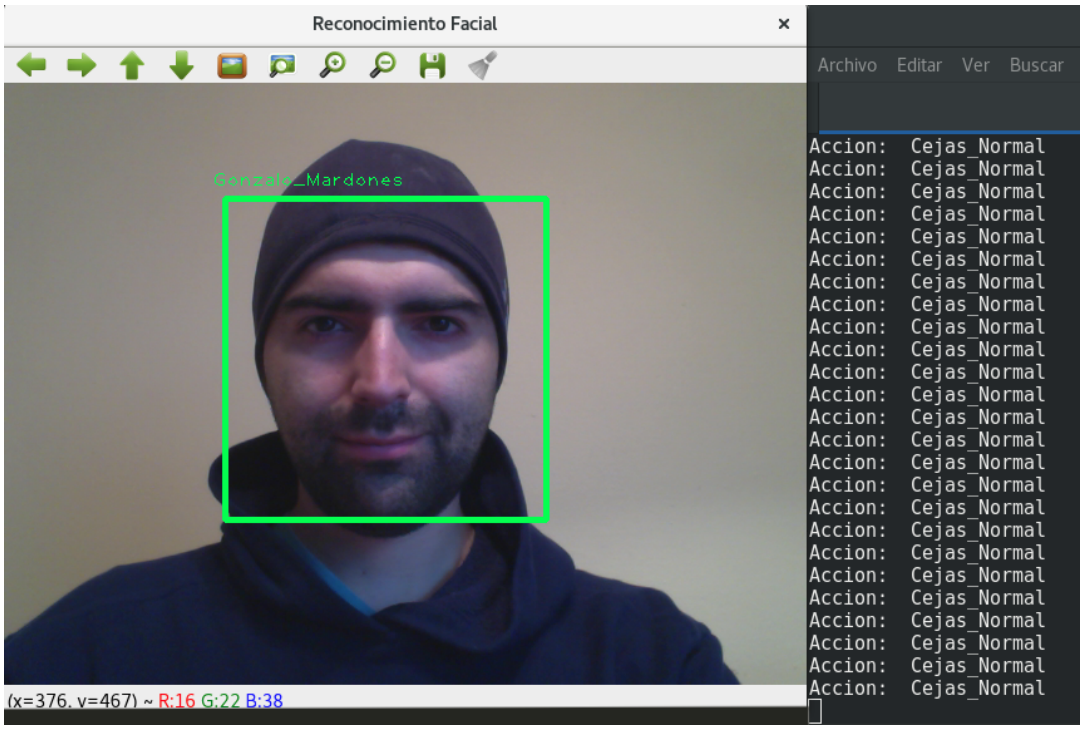


Figura 5.7: Predicción clasificador SVM para rostros con cejas en estado normal.

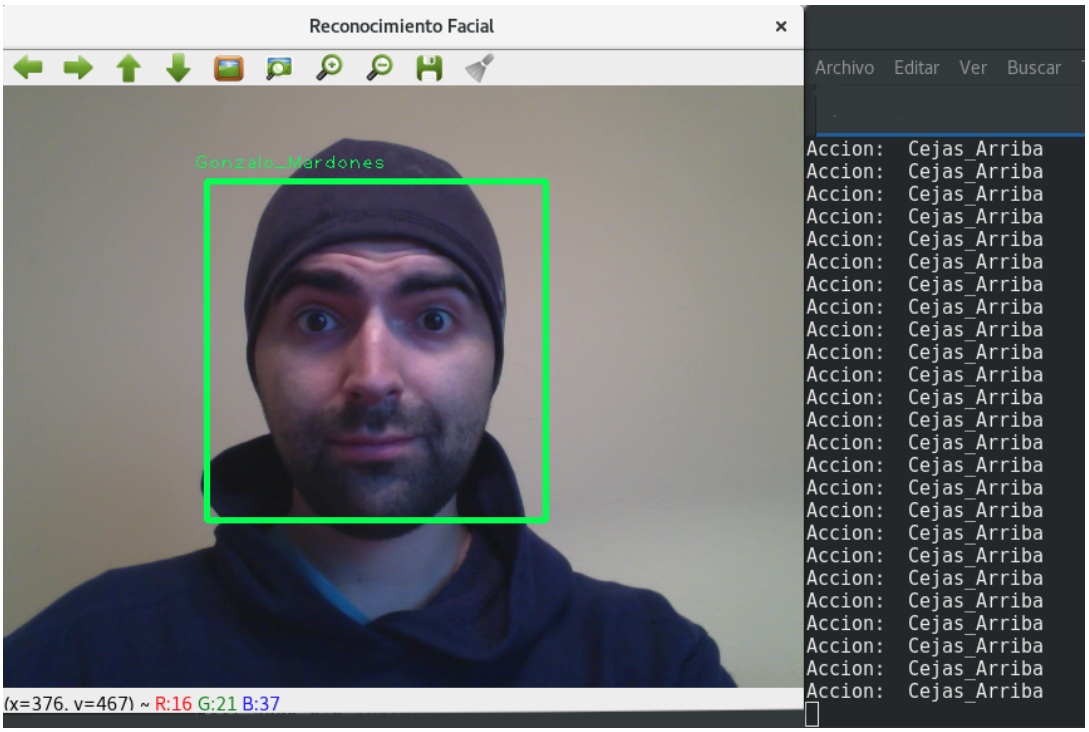


Figura 5.8: Predicción clasificador SVM para rostros con cejas arriba.

5.6. Etapa 3: Desarrollo API

Se presenta la implementación para generar la capa de comunicación entre los módulos de reconocimientos con los usuarios que consuman el servicio de la API.

Esta solución consiste en desarrollar una API utilizando el micro-framework *Flask*. Las dependencias necesarias para la instalación de *Flask* están disponible en Anexo C.

5.6.1. Usuarios

Para generar la comunicación con el computador servidor descrito en la Sección 5.1, es necesario la contar de un usuario que pueda conectarse local y remotamente, que envié peticiones válidas y que pueda también obtener las respuestas brindadas por el servidor. Para ello, se utiliza *cURL* que es un interprete de comandos orientado a la transferencia de archivos [24].

5.6.2. Funcionalidades

En el Cuadro 5.1se presentan los métodos principales de FME_API. Por medio de cURL se realiza las peticiones para el reconocimiento facial y de microexpresiones oculares.

#	**Método**	**Funcionalidad**
1	`entrenar_modelo_facial(folder)`	Retorna el modelo de reconocimiento facial.
2	`reconocer_microexpresion()`	Retorna microexpresión ocular detectado del rostro de la persona.
3	`reconocer_cara()`	Retorna la precisión, nombre e indice del directorio donde se encuentra la persona consultada.
4	`not_found(error)`	Mensaje de error 404.

Cuadro 5.1: Lista de funcionalidades FME_API.

Para iniciar el servicio de API y visualizar de mejor manera el comportamiento, se utiliza de modo local la dirección IP de *localhost* (dirección que apunta al computador que se está utilizando). Para conexiones remotas es necesario ingresar la IP y PUERTO de conexión del computador servidor. Es necesario iniciar una consola o terminal e ingresar los siguientes parámetros:

```
[gmardones@GM-pc FME_API]$ export FLASK_APP = fme_api.py
[gmardones@GM-pc FME_API]$ python -m flask run --host = 0.0.0.0

 * Serving Flask app "fme_api"
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Una vez iniciado el servicio, es posible realizar peticiones por medio de cURL. Las respuestas de FME_API al usuario son por medio del formato de intercambio de datos JSON. El Cuadro 5.2 muestra las peticiones en cURL y las respuesta en JSON.

En la fila 1 se aprecia la petición de cURL a la URL `/facial/recognition` que invoca al método `reconocer_cara()`. El usuario envía por método POST la imagen que quiere reconocer. Flask almacena localmente la imagen para que el modelo de reconocimiento facial pueda predecir. Note que es posible almacenar este registro temporal o permanentemente. La respuesta JSON retorna la `confianza` de tipo *float* obtenida por el modelo. El `indice_prediccion` de tipo *int*, indica a qué directorio de la base de datos pertenece la imagen. Por su parte `nombre` de tipo *string* indica de la persona identificada.

La fila 2 muestra la forma de como entrenar el modelo de reconocimiento facial. Es un método de tipo GET, por medio de la petición cURL a `/facial/train_model` que invoca al método `entrenar_modelo_facial()`. Se retorna un valor de tipo *boolean*: `true` en caso de que se entreno el modelo exitosamente, o `false` en caso contrario.

Finalmente, la fila 3 muestra la petición cURL a `/microexpression`, de tipo POST, que invoca al método `reconocer_microexpresion()`. El usuario envía la imagen y la API retorna un objeto JSON indicando si la persona posee las cejas arriba (`Cejas_Arriba`) o cejas en estado normal (`Cejas_Normal`).

#	**Peticiones**
1	`$ curl http://IP:PORT/api/v1.0/facial/recognition`
	POST: imagen.jpg/png

	`{`
	`  "confianza": float,`
	`  "indice_prediccion": int,`
	`  "nombre": string`
	`}`

2	`$ curl http://IP:PORT/api/v1.0/facial/train_model`

	`{`
	`  "modelo_facial_entrenado": boolean`
	`}`

3	`$ curl http://IP:PORT/api/v1.0/microexpression`
	POST: imagen.jpg/png

	`{`
	`  "Accion": string`
	`}`

Cuadro 5.2: Peticiones cURL y respuestas tipo JSON.

6. Pruebas y Resultados

El capítulo detalla las pruebas realizadas para verificar el correcto funcionamiento de los métodos implementados en la API. Para ello se realizan pruebas sobre los módulos de reconocimiento facial y de reconocimiento de microexpresiones oculares.

6.1. Reconocimiento Facial

La primera etapa de desarrollo de la API cuenta con el modelo predictorio de reconocimiento facial, tiene como objetivo identificar a la persona que se encuentra frente de la cámara devolviendo con su nombre y precisión.

6.1.1. Escenarios de Prueba

Las pruebas fueron realizadas en un ambiente controlado, donde cada imagen procesada se encontraba sin elementos distractores para la cámara web. Es decir, sin fotografías en las paredes con iluminación artificial (de preferencia luz de tipo led), sin la participación de más personas al momento de realizar las capturas y donde que las paredes están pintadas de color claro.

6.1.2. Confianza y Reconocimiento

A menor valor de la variable confianza, mejor es la respuesta del modelo de Fisherface. El Cuadro 6.1 muestra que a mayores variaciones de iluminación, pose, expresiones y cantidad de imágenes que posea la persona inscrita en la base de datos, mejor es la precisión del modelo para predecir imágenes. Las pruebas de confianzas determinan los rangos para indicar la predicción del modelo (no reconoce, reconoce pero imagen difusa, reconoce). Se evaluaron 100 fotografías y se obtuvo la confianza promedio, determinando de esta manera los rangos de reconocimiento.

#	Imagenes	Confianza	Resultado	Observaciones
1	10	736,6798	No reconoce	Luz constante, sin variación de pose, expresión normal, distancia de la cámara: 55 cm.
2	50	670,8243	No reconoce	Luz constante, giro suave derecha e izquierda, expresión normal, distancia de la cámara: 55 cm.
3	100	660,8124	No reconoce	Luz constante, giro suave derecha e izquierda, expresión normal, sonrisa, distancia de la cámara: 55 cm.
4	150	554,9345	Reconoce, pero no estable	Luz constante, luz tenue, giro suave derecha e izquierda, expresión normal, sonrisa, enojo, distancia de la cámara: 55 cm.
5	200	533,0163	Reconoce, pero no estable	Luz constante, luz tenue, luz localizada lado derecho, giro suave derecha e izquierda, expresión normal, sonrisa, enojo, ojos cerrados, distancia de la cámara: 55 cm.
6	250	446,8639	Reconoce	Luz constante, luz tenue, luz localizada lado derecho, giro suave derecha e izquierda, expresión normal, sonrisa, enojo, ojos cerrados, distancia de la cámara: 30 cm.
7	350	326,8757	Reconoce	Luz constante, luz tenue, luz localizada lado derecho e izquierdo, giro suave hacia la derecha, izquierda, arriba, abajo, en circulos, expresión normal, sonrisa, enojo, ojos cerrados, distancia de la cámara: 55 cm.

Cuadro 6.1: Variaciones reconocimiento facial.

6.2. Reconocimiento Microexpresiones Oculares

La librería de Python *Scikit-learn* provee de clasificadores de tipo SVM. Estos poseen kernel de tipo *LINEAL, RBF* y *POLY*. Para el caso del kernel *LINEAL* es posible configurar su valor de coste, esto es el margen del hiperplano. El kernel *RBF* configura el valor *gamma* que permite ampliar o reducir la zona de influencia de los vectores de soporte. El kernel *POLY* permite variar el valor *degreee* que corresponde a la cantidad de clases que participan del entrenamiento. A continuación se presentan distintas pruebas para determinar el mejor clasificador frente a la base de datos comentados en el capitulo anterior.

6.2.1. Base de Datos

Las pruebas de base de datos son desarrolladas con un clasificador estándar SVM con kernel LINEAL, coste = 1,0 (coste por defecto de Sckit-learn) y 10 iteraciones para determinar un promedio más ajustado en la etapa de entrenamiento del modelo. El tamaño de la base de datos es variable y corresponde a la cantidad de imágenes en los directorios **CN: Cejas_Normal** y **CA: Cejas_Arriba**, el tiempo de entrenamiento del modelo es medido en segundos. Se determina la exactitud del modelo predictor SVM y la exactitud de un directorio de prueba que no participa de la etapa de entrenamiento. Los resultados se muestran en el Cuadro 6.2.

Variación base de datos

#	tamaño BD	Tiempo	Exactitud modelo	Exactitud test
1	584 CA; 522 CN	344 seg.	91 %	100 %
2	300 CA; 300 CN	258 seg.	94 %	81 %
3	200 CA; 200 CN	198 seg.	91 %	81 %
4	100 CA; 100 CN	83 seg.	85 %	81 %
5	50 CA; 50 CN	9 seg.	81 %	70 %

Cuadro 6.2: Rendimiento con variación de base de datos SVM LINEAL.

6.2.2. Kernel

Para determinar el mejor clasificador SVM, se necesita determinar el kernel que devuelva la exactitud más alta en virtud de las 2 clases existentes, además de evaluar la precisión del modelo, tiempo empleado utilizado para entrenar y variables requeridas según el kernel usado. Los siguientes kernel son estándar, y en base a sus predicciones se determina cúal usar y qué variable se configura. Cada prueba es iterada en 10 ocasiones, a fin de determinar un promedio más ajustado del modelo en la etapa de entrenamiento. El tamaño de la base de datos del directorio **CN: Cejas_Normal** es de 522 imágenes y 584 para el directorio **CA: Cejas_Arriba**. El tiempo del entrenamiento del modelo es medido en segundos. Además, se determina la exactitud promedio de las imágenes de un directorio de pruebas que no participa en la etapa de entrenamiento. Este directorio permite corroborar o acercarse a los resultados entregados por el modelo.

El Cuadro 6.3 muestra el comportamiento del clasificador utilizando un kernel **LINEAL**. Posee la variable configurable coste = 1, que es determinada por defecto por Scikit-learn.

#	**Tiempo**	**Coste**	**Exactitud Modelo**	**Exactitud test**
1	492 seg.	1	91,9 %	100 %

Cuadro 6.3: Rendimiento clasificador SVM utilizando kernel *LINEAL*.

El Cuadro 6.4 muestra el comportamiento del clasificador utilizando un kernel **RBF**. Posee la variable configurable *gamma* = AUTO. El valor AUTO es asignado automáticamente por Scikit-learn, busca y determina la cantidad de clases y asigna el mejor valor para gamma.

#	**gamma**	**Tiempo**	**Exactitud Modelo**	**Exactitud test**
1	AUTO	349 seg.	58 %	80 %

Cuadro 6.4: Rendimiento clasificador SVM utilizando kernel *RBF*.

El Cuadro 6.5 muestra el comportamiento del clasificador utilizando kernel **POLY**. Posee la variable configurable *degree*, que es asignada por defecto por Scikit-learn con valor 3, pero dado que sólo existen 2 clases, toma el valor de *degree* = 2.

#	degree	Tiempo	Exactitud Modelo	Exactitud test
1	2	435 seg.	87,9%	50%

Cuadro 6.5: Rendimiento clasificador SVM utilizando kernel *POLY*.

6.2.3. Coste SVM Kernel LINEAL

Las pruebas anteriores determinan que el kernel con mejor rendimiento es el LINEAL. Las evaluaciones de rendimiento son desarrolladas con variación de coste y 10 iteraciones en la etapa de entrenamiento. El tamaño de la base de datos del directorio **CN** (Cejas_Normal) es de 522 imágenes y 584 imágenes para el directorio **CA** (Cejas_Arriba), el tiempo de entrenamiento es medido en segundos. Se evalúa la exactitud del modelo y la precisión del reconocimiento utilizando un directorio con imágenes de prueba que no participó en la etapa de entrenamiento. Se utiliza para corroborar los resultados entregados por el modelo en su predicción por medio de validación cruzada. Los resultados se muestran en el Cuadro 6.6.

#	Coste	Tiempo	Exactitud Modelo	Exactitud test
1	1	472 seg.	91,9%	100%
2	5	506 seg.	92,3%	77,7%
3	10	489 seg.	90,9%	100%
4	15	465 seg.	90,9%	88,8%
5	20	478 seg.	92,2%	100%
6	40	443 seg.	90,9%	90,9%
7	100	479 seg.	91%	90,9%
8	200	469 seg.	91%	90,9%

Cuadro 6.6: Rendimiento clasificador SVM con kernel LINEAL con variación de coste.

6.3. FME_API

Se realizan pruebas a las peticiones y se determinan los tiempos de respuesta de modo local y remoto. Por medio de solicitudes con la función `time` de cURL se obtienen los tiempos de ejecución en tiempo real y por parte del sistema.

6.3.1. Reconocimiento Facial

El modelo de reconocimiento facial es entrenado con 1,106 imágenes que corresponden a 43 directorios. El Cuadro 6.7 muestra los tiempos de ejecución de la petición de entrenamiento del modelo. El Cuadro 6.8 muestra cuando el reconocimiento facial es exitoso. Cuando el reconocimiento no es claro, o la imagen consultada es difusa, ver el Cuadro 6.9. El Cuadro 6.10 muestra la petición de la imagen que se quiere detectar y no se encuentra en la base de datos.

Petición
`$ time curl http://0.0.0.0:5000/api/v1.0/facial/train_model`

`{`
`  "modelo_facial_entrenado": true`
`}`
`real 2m18.732s`
`user 0m0.004s`
`sys  0m0.010s`

Cuadro 6.7: Prueba entrenamiento modelo de reconocimiento facial.

Petición
`$ time curl http://0.0.0.0:5000/api/v1.0/facial/recognition`
POST: t11.png

```
{
  "confianza": 200.86037574254266,
  "indice_prediccion": 0,
  "nombre": "Gonzalo_Mardones"
}
real 0m0.108s
user 0m0.003s
sys  0m0.008s
```

Cuadro 6.8: Reconocimiento facial de cara conocida, ver imagen en Anexo E.1.a.

Petición
`$ time curl http://0.0.0.0:5000/api/v1.0/facial/recognition`
POST: t10.png

```
{
  "confianza": 478.1477314603432,
  "motivo": "Imagen difusa",
  "resultado": "Desconocido"
}
real 0m0.030s
user 0m0.003s
sys  0m0.004s
```

Cuadro 6.9: Reconocimiento facial no es preciso, ver imagen en Anexo E.1.b.

Petición
`$ time curl http://0.0.0.0:5000/api/v1.0/facial/recognition`
POST: t12.jpg

```
{
  "confianza": 551.7286116684757,
  "motivo": "No se encuentra en base de datos",
  "resultado": "Desconocido"
}
real 0m0.026s
user 0m0.003s
sys  0m0.005s
```

Cuadro 6.10: Reconocimiento facial, cara no encontrada en BD, ver imagen en Anexo E.1.d.

6.4. Reconocimiento de Microexpresiones Oculares

El modelo de reconocimiento de microexpresiones oculares muestra la respuesta de la petición para una cara con cejas arriba, ver Cuadro 6.11. Cuando la cara posee las cejas en estado normal el modelo responde con la salida mostrada en el Cuadro 6.12.

Ante peticiones erróneas o que no han sido mencionadas anteriormente, la API responde con un mensaje de error 404, ver Cuadro 6.13.

Petición
`$ time curl http://0.0.0.0:5000/api/v1.0/microexpression`
POST: t13.jpg

`{`
`  "Accion": "Cejas_Arriba"`
`}`
`real 0m0.201s`
`user 0m0.004s`
`sys  0m0.003s`

Cuadro 6.11: Reconocimiento ocular, cejas arriba, ver imagen en Anexo E.1.c.

Petición
`$ time curl http://0.0.0.0:5000/api/v1.0/microexpression`
POST: t12.jpg

`{`
`  "Accion": "Cejas_Normal"`
`}`
`real 0m0.077s`
`user 0m0.003s`
`sys 0m0.005s`

Cuadro 6.12: Reconocimiento cejas en estado normal, ver imagen en Anexo E.1.d.

6.4.1. Resumen Peticiones FME_API

El Cuadro 6.14 muestra el método que se evalúa, formato de tipo GET o POST y los tiempos de respuesta local y remoto, medidos en minutos y milisegundos.

Petición

```
$ time curl http://0.0.0.0:5000/api/v1.0/micro/t11.png
```

```
{
  "error": "la URL solicitada no se encontro en el servidor.
  Si ingreso la url manualmente, verifique su ortografia
  y vuelva a intentarlo."
}
real 0m0.009s
user 0m0.001s
sys  0m0.006s
```

Cuadro 6.13: Error 404, dirección URL invalida.

#	Método	Tipo	Tiempo local	Tiempo remoto
1	`/train_model`	GET	1:28 min.	1:39 min.
2	`/recognition` [1]	POST	52.4 ms.	232.2 ms.
3	`/recognition` [2]	POST	48.4 ms.	132.4 ms
4	`/recognition` [3]	POST	108 ms.	586.2 ms
5	`/microexpression` [4]	POST	2975.2 ms.	5902 ms.
6	`/microexpression` [5]	POST	102 ms.	137.4 ms.

Cuadro 6.14: Tiempos reales de ejecuciones locales y remotas.

De acuerdo al Cuadro 6.14 se aprecian las diferencias de tiempo local y remoto para la misma entrada. La diferencia de tiempo más significativa se aprecia en el entrenamiento del modelo de reconocimiento facial, siendo el tiempo remoto más costoso. En estas pruebas se usaron imágenes de 92x112 píxeles hasta imágenes de 2444x1718 píxeles.

[1] Reconocimiento facial - No reconoce.
[2] Reconocimiento facial - Indeciso.
[3] Reconocimiento facial - Reconoce.
[4] Reconocimiento microexpresiones - Cejas Arriba.
[5] Reconocimiento microexpresiones - Cejas Normal.

6.5. Resultados

En base al trabajo desarrollados y experimentos planteados, es posible obtener diversos resultados que a continuación se comentan:

Los algoritmos tradicionales de reconocimiento facial, si bien proveen de propiedades y métodos claros, proveen estudios de iluminación, rotaciones de la cara y expresión de emociones generales que pueden entregar resultados cada vez más precisos. Pero, no existe un listado de las operaciones que deben ser llevadas a cabo para obtener un nivel de precisión más alto, sólo la prueba y error, sumado a la calidad de las imágenes son relevantes a la hora de desarrollar modelos de reconocimiento facial más precisos.

Por otro lado, para lograr un nivel de confianza aceptable existen personas que requieren menores cantidades de imágenes, en cambio otras personas requieren mayor cantidad de registros. La documentación de OpenCV no detalla estas particularidades a la hora de implementar los modelos de clasificación. Es por ello que es necesario recurrir a la literatura en el área para comprender mejor las variables necesarias para el entrenamiento.

Situación parecida sucede con los clasificadores de Support Vector Machine, donde la prueba y error del clasificador determina la mejor precisión del modelo. Existen situaciones donde es posible inferir cúal podría ser el mejor candidato para utilizar, pero no hay seguridad de qué kernel utilizar a menos que se realicen las pruebas. Las imágenes de las caras deben ser muy claras y fáciles de distinguir. Ejemplo de ello son rostros donde la persona de forma deliberada ubica las las cejas en estado normal y cejas hacia arriba, lo cual facilita la separación entre las clases. El modelo no esta entrenado para determinar cejas en un punto intermedio, en cuyo caso intentará responder con la clase que no corresponde.

7. Conclusión

En este capítulo se presentan las conclusiones basadas en los resultados obtenidos durante el desarrollo de la API de reconocimiento facial y microexpresiones oculares utilizando los algoritmos de Fisherface y Support Vector Machine. Por último se da luces acerca de los trabajos a futuro que de este desarrollo se puede desprender.

7.1. Del Estudio Teórico

El proceso de investigación consistió en una revisión literaria sobre las expresiones faciales, procesamiento digital de imágenes, algoritmos de reconocimiento facial, como se comportan y comparten elementos con los algoritmos de clasificación. Se presentan definiciones y características que permiten entender mejor la problemática y el diseño de solución de la API.

7.2. Del Reconocimiento Facial

El área de visión por computadora se encuentra en constante crecimiento. Los modelos de reconocimiento facial son cada vez más precisos ante condiciones adversas, sean estos: variaciones de iluminación, rotación bruscas de cara, cambios físicos producto de la edad, entre otros. Es por ello que el estudio de nuevos algoritmos de reconocimiento facial es cada vez más solicitado por empresas que desean mejorar sus procesos comerciales.

El desarrollo de la API inicia con la etapa de reconocimiento facial, donde las fotografías son capturadas. Posteriormente, son procesadas y clasificadas. Por medio de los resultados obtenidos en la etapa de prueba es posible concluir que el modelo de reconocimiento facial entrega rangos de confianza acotados en los siguientes segmentos: si la confianza es mayor o igual que 0 y menor a 500, el modelo identifica a la personas y devuelve el nombre del directorio. Si la confianza se encuentra entre 500 y 600 el modelo

no predice con precisión, y puede reconocer o no a la persona, ya que la imágen evaluada no presenta mayores variaciones de iluminación o pose. Si la confianza es mayor a 500 indica que la cara es desconocida o no posee registro en la base de datos.

7.3. Del Reconocimiento de Microexpresiones Oculares

Los algoritmos de Machine Learning están viviendo su auge en los últimos años y con ello los clasificadores SVM han mostrado un gran desempeño, simplicidad, efectividad y mejor uso de recursos frente a otros métodos de aprendizajes tradicionales. Dado lo anterior, se utiliza el algoritmo de Support Vector Machine con kernel lineal. Este kernel entrega la mejor precisión frente los otros. Las pruebas determinan que la variable de coste presenta una importancia trascendental en el entrenamiento del modelo, ya que una mala elección de esta variable implica que el modelo no retorne la mejor precisión. El estudio determina que el costo= 20 retorna el modelo de predicción con una exactitud del 92,2 % para la detección de cejas arriba y cejas en estado normal.

7.4. Trabajos Futuros

En base al trabajo realizado, es posible plantear nuevas tareas que permitan profundizar el problema y solucionar otros que surgen en base a este desarrollo, tales como las que a continuación se comentan:

- Evaluar el uso de otros algoritmos de reconocimiento facial y de clasificación de microexpresiones a fin de determinar posibles mejoras en los tiempos y precisión de los modelos.
- Incluir nuevas microexpresiones, sean estos: boca, guiño de ojos, sentimientos, entre otros.
- Estudiar el comportamiento del modelo de reconocimiento facial discriminando entre personas gemelas y perfiles laterales.
- Estandarizar y proveer de una documentación que permita determinar los ambientes y variantes necesarios para obtener un buen porcentaje de exactitud en el reconocimiento facial.

Bibliografía

[1] General Information, The Making of Python. http://www.artima.com/intv/pythonP.html. Visto el 19 de Octubre del 2017.

[2] M.Á.V. Alonso, A.L.A. Díaz, A. Aguado, and M.Á.V. Alonso. *Personas con Discapacidad: Perspectivas Psicopedagógicas y Rehabilitadoras*. Manuales (Siglo XXI de España Editores): Psicología. Siglo XXI de España, 2005.

[3] Gustavo Betancourt. Las máquinas de soporte vectorial (svms). *Scientia et Technica*, XI(27):67–68, aug 2005. ISNN 0122-1701.

[4] Joshua Bloch. *How to Design a Good API and Why it Matters*. Google, 2006.

[5] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):35, 1998.

[6] José Luis Alba Castro. Curso de Doctorado - Comparación ANN versus SVM. http://web.archive.org/web/20140801145654/http://www.gts.tsc.uvigo.es:80/~jalba/dOctubreorado/SVM.pdf. Visto el 10 de Septiembre del 2017.

[7] Steven Clarke. Measuring API Usability. http://www.drdobbs.com/windows/measuring-api-usability/184405654. Visto el 13 de Noviembre del 2017.

[8] David Cournapeau. Scikit-learn, Machine Learning in Python. http://scikit-learn.org/stable/index.html. Visto el 10 de Junio del 2017.

[9] Dirección General de Formación Profesional. *Educación Inclusiva - Discapacidad Visual*. Secretaria de Estado de Educación Profesional, Instituto de Tecnologías Educativas, 2015.

[10] S. Debbie and Joshua Correll. Chicago Face DataBase. http://faculty.chicagobooth.edu/bernd.wittenbrink/cfd/index.html. Visto el 23 de Agosto del 2017.

[11] P.A Devijver and J Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, Londres, 1982.

[12] Paul Ekman and Wallace Friesen. *Facial Action Coding System (FACS): A Technique for the Measurement of Facial Movement*. Psychologists Press, 1978.

[13] Python Software Foundation. Do I have to like "Monty Python's Flyung Circus". `https://docs.python.org/3/faq/general.html#why-was-python-created-in-the-first-place`. Visto el 19 de Octubre del 2017.

[14] Louis L. Frenzel. *Sistemas Electrónicos de Comunicaciones (Tercera reimpresión)*. México, D. F: Alfaomega. pp. 21 a 23, 2003.

[15] Hermenegildo García. *Cámara Fotográfica*. Sección de Informática Gráfica - Departamento de Sistemas Informáticos y Computación (DSIC), Universidad Politécnica de Valencia, 2017.

[16] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirami. *An Introduction to Statistical Learning with Applications in R*. Springer New York Heidelberg Dordrecht London, 2013.

[17] Renato Jijena. *La protección penal de la intimidad y el delito informático)*. Santiago: Jurídica de Chile, 1992.

[18] Carmen Lasa Gomez and Alonso Alvarez García. *MÉTODOS ÁGILES: SCRUM, KANBAN, LEAN*. Anaya Multimedia, 2017.

[19] Dlib C++ Library. Overview, Machine Learning Guide. `http://dlib.net/intro.html`. Visto el 30 de Septiembre del 2017.

[20] James D. Murray and William van Ryper. *Encyclopedia of Graphics File Formats, Seceond Edition*. Sebastopol, Calif.: O'Reilly, 1996.

[21] W3C Working Group Note. Relationship to the World Wide Web and REST Architectures. `https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest`. Visto el 15 de Octubre del 2017.

[22] Guillermo Ottado. *Reconocimiento de caras: Eigenfaces y Fisherfaces*. Universidad ORT - Uruguay, 2010.

[23] Armin Ronacher. Flask - web development, on drop at a time. `http://flask.pocoo.org`. Visto el 30 de Octubre del 2017.

[24] Daniel Stenberg. command line tool and library for transferring data with URLs. `https://curl.haxx.se`. Visto el 5 de Octubre del 2017.

[25] Mariano Tapiador and J SigÃ$\frac{1}{4}$enza. *Tecnologías Biométricas Aplicadas a la Seguridad*. Editorial RAMA„ 1982.

[26] OpenCV Team. About OpenCV. `https://opencv.org/about.html`. Visto el 30 de Septiembre del 2017.

[27] OpenCV Dev Team. FaceRecognizer - createFisherFaceRecognizer. `https://docs.opencv.org/3.0-beta/modules/face/doc/facerec/facerec_api.html`. Visto el 20 de Septiembre del 2017.

Glosario

API : Interfaz de Programación de Aplicaciones, da a los programas accesos a los recursos de hardware y servicios habilitados.

FME_API : Facial Microexpressions API, es una API de reconocimiento facial y de microexpresiones oculares.

BSD : Berkeley Software Distribution, es una licencia que permite el uso de software libremente para propósitos comerciales y académicos.

DLF : Discriminante Lineal de Fisher, permite reducir de dimensiones y busca maximizar la varianza de las muestras entre clases.

EMM : Espectro Electromagnético, es el conjunto de longitudes de onda de todas las radiaciones electromagnéticas.

JSON : JavaScript Object Notation, basada en JavaScript, es un formato de texto para el intercambio de datos.

PCA : Análisis de Componentes Principales, es una técnica utilizada para reducir la dimensionalidad de un conjunto de datos.

PGM : Portable Graymap, es un formato de imágenes en escala de grises, utiliza 8 bits.

Python : Lenguaje de programación desarrollado por *Guido van Rossum* en 1991.

SVMs : Support Vector Machine, es un método basado en Machine Learning que es utilizado en problemas de clasificación y regresión.

ANEXOS

A. Documentos Oficiales

En este capítulo se presentan los documentos generados en la obtención de requisitos. Se describen cada uno de los requerimientos expresados por parte de la empresa *EXE Ingeniería y Software*. Además, se muestran el documento presentado a los participantes a fin que voluntariamente sean parte de la investigación.

Cada requisito incluye una descripción breve, fuente de obtención, prioridad asignada, estabilidad (nivel de fallos reducidos), fecha de actualización del requisito, estado (cumple, no cumple) y tipo (funcional, no funcional).

A.1. Formato Consentimiento Informado

Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Consentimiento Informado

Título del Proyecto: API de reconocimiento facial que utilice micro-expresiones oculares por medio de redes neuronales artificiales.

Autor: Gonzalo Mardones Baeza

El propósito de este documento es entregarle toda la información necesaria para que usted pueda decidir libremente si desea participar de la investigación, que se le ha explicado verbalmente, y que a continuación se describe de forma resumida:

La investigación busca tomar fotografías del rostro de la persona a fin de obtener sus micro-expresiones oculares. Estas micro-expresiones tienen como objeto la creación de un modelo de aprendizaje artificial que determine que la persona "es ... quien dice ser".

Al respecto expongo que:

- He sido informado sobre el estudio a desarrollar.
- Estoy en pleno conocimiento que la información obtenida en el estudio en que participaré, será absolutamente confidencial, y que no darán a conocer mis datos personales en ninguna publicación derivadas de la investigación.
- Estoy en conocimiento que la decisión de participar en esta investigación, es totalmente voluntaria. Que queda totalmente a mi arbitrio, una vez iniciada la investigación, seguir o no participando en dicho proyecto, sin que esto acarree ninguna consecuencia en mi contra.

Yo, Sr/a: .....
Cédula de identidad Nº .....
Correo electrónico .....
He leído el documento de consentimiento informado que me ha sido entregado, he comprendido las declaraciones contenidas en él y he podido resolver todas las dudas y preguntas que he planteado al respecto.

Consiento mi participación en la captura de fotografías a la cual fui convocado/a y que los datos que se deriven de mi participación sean utilizados para cumplir con los objetivos especificados en el documento, para lo cual firme libre y voluntariamente.

En ..... a ..... de ..... de 2017.

Firma: .....

A.2. Formato Documento de Requisitos

Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Documento de Requisitos

Título del Proyecto: API de reconocimiento de microexpresiones faciales utilizando algoritmos de Fisherface y Support Vector Machine

Autor: Gonzalo Andrés Mardones Baeza

El propósito de este documento es la ratificación de los requerimientos generados en Marzo de 2017. Obedeciendo al proyecto de **corto plazo**: Construir una API de reconocimiento de faciales utilizando algoritmos de Fisherface y reconocimiento de microexpresiones oculares por medio de clasificador Support Vector Machine.

Requisitos de usuario

RU001	
Descripción :	La API debe identificar a la persona dentro de la imagen.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Funcional.

RU002	
Descripción :	El sistema debe capturar las microexpresiones oculares de las fotográficas.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Funcional.

Requisitos de usuario

RU003	
Descripción :	El sistema debe integrar las capturas fotográficas a un clasificador de FisherFace para la identificación de la persona.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Necesario.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU004	
Descripción :	El sistema debe crear de base de datos con imágenes de personas que contengan distintas expresiones faciales para clasificar por medio de SVM.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU006	
Descripción :	La API debe poder entrenar el modelo de reconocimiento facial.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Necesario.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Funcional.

RU007	
Descripción :	El sistema debe reconocer a personas existentes en la base de datos.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Funcional.

Requisitos de usuario

RU008	
Descripción :	El sistema debe responder a la API diseñada.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Usuario.

RU009	
Descripción :	La API debe estar debidamente documentado a fin de que sea mantenible en el tiempo.
Fuente :	Empresa.
Prioridad :	Necesaria.
Estabilidad :	Alta.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Funcional.

RU010	
Descripción :	La API debe identificar microexpresiones oculares, sean estás cejas arriba o en estado normal.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	14 de Marzo de 2017.
Estado :	No Cumple.
Tipo :	Funcional.

Yo, Sr/a: ..... , cédula de identidad Nº ..... He leído el documento de validación de requisitos que me ha sido entregado, he comprendido las declaraciones contenidas en él y he podido resolver todas las dudas y preguntas que he planteado al respecto.

En ..... a ..... de ..... de 2017.

Firma: .....

A.3. Documento de Requisitos Validado

UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Documento de Requisitos

Título del Proyecto: API de reconocimiento de microexpresiones faciales utilizando algoritmos de Fisherface y Support Vector Machine

Autor: Gonzalo Andrés Mardones Baeza

El propósito de este documento es la ratificación de los requerimientos generados en Marzo de 2017. Obedeciendo al proyecto de **corto plazo**: Construir una API de reconocimiento de faciales utilizando algoritmos de Fisherface y reconocimiento de microexpresiones oculares por medio de clasificador Support Vector Machine.

Requisitos de usuario

RU001	
Descripción :	La API debe identificar a la persona dentro de la imagen.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU002	
Descripción :	El sistema debe capturar las microexpresiones oculares de las fotográficas.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

Requisitos de usuario

RU003	
Descripción :	El sistema debe integrar las capturas fotográficas a un clasificador de FisherFace para la identificación de la persona.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Necesario.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU004	
Descripción :	El sistema debe crear de base de datos con imágenes de personas que contengan distintas expresiones faciales para clasificar por medio de SVM.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU006	
Descripción :	La API debe poder entrenar el modelo de reconocimiento facial.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Necesario.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU007	
Descripción :	El sistema debe reconocer a personas existentes en la base de datos.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

Requisitos de usuario

RU008	
Descripción :	El sistema debe responder a la API diseñada.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Usuario.

RU009	
Descripción :	La API debe estar debidamente documentado a fin de que sea mantenible en el tiempo.
Fuente :	Empresa.
Prioridad :	Necesaria.
Estabilidad :	Alta.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

RU010	
Descripción :	La API debe identificar microexpresiones oculares, sean estás cejas arriba o en estado normal.
Fuente :	Empresa.
Prioridad :	Alta.
Estabilidad :	Intransable.
Fecha Actualización :	20 de Octubre de 2017.
Estado :	Cumple.
Tipo :	Funcional.

Yo, Sr/a: Luis Gonzalez Martinez , cédula de identidad N° 16.004.129-3 **He leído el documento de validación de requisitos que me ha sido entregado, he comprendido las declaraciones contenidas en él y he podido resolver todas las dudas y preguntas que he planteado al respecto.**

En Taka a 17 de 11 **de 2017.**

Firma:

A.4. Documentación API

Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Documentación FME_API

API de reconocimiento de microexpresiones faciales utilizando algoritmos de Fisherface y Support Vector Machine.

Autor: Gonzalo Andrés Mardones Baeza.

- Palabra clave: **FME_API** (Facial Microexpressions API).

La documentación de FME_API ofrece una descripción detallada de los métodos disponibles, con ejemplos de las respuestas obtenidas, así como información acerca del uso.

1. FME_API Endpoint's

Todas las URL's listadas en la documentación son relativas a `http://IP:PUERTO/api/v1.0/`

Algunas consideraciones para tener en cuenta:

- Todas las respuestas de FME_API son retornadas en formato JSON.
- Las peticiones de la FME_API deben ser realizadas como peticiones GET, POST según sea el caso.
- Puede considerar todas las respuestas que no sean devueltas con un código HTTP 404 como un recurso no encontrado. Se utiliza cuando el servidor web no encuentra la página o recurso solicitado.

Para iniciar el servicio de API y visualizar de mejor manera el comportamiento, se utiliza de modo local la dirección IP de *localhost* (dirección que apunta al computador que se esta utilizando), para conexiones remotas es necesario ingresar la IP y PUERTO de conexión del computador servidor. Es necesario iniciar una consola o terminal e ingresar los siguientes parámetros:

```
[user@US-pc FME_API]$ export FLASK_APP = fme_api.py
[user@US-pc FME_API]$ python -m flask run --host = 0.0.0.0

 * Serving Flask app "fme_api"
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

2. Funcionalidades FME_API

FME_API dispone de cuatro funcionalidades: entrenamiento de modelo facial, reconocer cara, reconocer microexpresiones oculares y mensaje de error. A continuación se describe sus características:

2.1. Entrenamiento modelo de reconocimiento facial

Método encargado de retornar el modelo de reconocimiento facial entrenado.

Petición
`http://IP:PORT/api/v1.0/facial/train_model`
Formato tipo: GET

`{`
`  "modelo_facial_entrenado": true`
`}`

`modelo_facial_entrenado`: Indica si el modelo se entreno, retornando `true` si se entreno o `false`, en caso contrario.

2.2. Reconocimiento Facial

Requisito previo: El modelo de reconocimiento facial debe estar entrenado. El método se encarga de retornar información de la cara identificada.

A continuación se presentan por medio de ejemplos los niveles de confianza establecidos para el reconocimiento facial.

Confianza <500	**500 ≥ Confianza < 600**	**Confianza ≥ 600**

`confianza: 200.8603,`
`indice_prediccion: 0,`
`nombre: Gonzalo_Mardones`
	`confianza: 578.1477,`
	`motivo: Imagen difusa,`
	`resultado: Desconocido`
		`confianza: 651.7286,`
		`motivo: No se encuentra en base de datos,`
		`resultado: Desconocido`

Reconoce	Reconoce, pero no estable	No reconoce

Petición
`http://IP:PORT/api/v1.0/facial/recognition`
Formato tipo: POST / key: (filename): imagen.jpg/png

`{`
`  "confianza": float,`
`  "indice_prediccion": int,`
`  "nombre": string`
`}`

`confianza`: Porcentaje de similitud con la persona que más se parezca. `indice_prediccion`: Indica al directorio a la que pertenece la persona. `nombre`: Nombre de la persona identificada.

2.3. Reconocimiento de Microexpresiones Oculares

Requisitos previos: El modelo de reconocimiento de microexpresiones oculares debe estar entrenado (operación que se realiza de forma manual y debe ser comunicado a quien tenga acceso a FME_API, Sección 5) previo a la etapa de reconocimiento. El método se encarga de retornar si la cara posee las cejas arriba o en estado normal.

Petición
`http://IP:PORT/api/v1.0/microexpression`
Formato tipo: POST / key: (filename): imagen.jpg/png

`{`
`  "Accion": "Cejas_Arriba"`
`}`

`Accion`: Indica si la ceja se encuentra arriba o en estado normal.

2.4. Mensaje de Error

Método encargado de retornar mensaje de error debido a que la URL escrita es errónea o inválida.

Petición
`http://IP:PORT/api/v1.0/micro/t11.png`

`{`
`  "error": "la URL solicitada no se encontro en el servidor.`
`  Si ingreso la url manualmente, verifique su ortografia`
`  y vuelva a intentarlo."`
`}`

`error`: De tipo 404, recurso no encontrado.

3. Base de Datos de Reconocimiento Facial

NOTA: Las imágenes enviadas por POST quedan almacenadas en la base de datos del propietario de FME_API.

Si quiere incluir un nuevo registro de una persona en FME_API, debe comunicar al propietario para que valide el registro y poder entrenar el modelo de reconocimiento facial.

4. Requisitos de Hardware y Software

Es necesario contar con un servidor o computador-servidor que posea como mínimo los siguientes requisitos:

- **Sistema Operativo**: Linux - Fedora v. 25.
- **Nombre Procesador**: Intel Core i5.
- **Velocidad Procesador**: $2,5$ GHz.
- **Cantidad Procesadores**: 1.
- **Cantidad total de núcleos**: 2.
- **Memoria**: 8 GB 1600 MHz DDR3.

- **Python**: v. `2.7.13`.
  - PIP: v. `9.0.1`.
  - Numpy v. `1.13.3`.
  - Sttckit-learn: v. `0.19.1`.
  - Flask: v. `0.12.2`.
- **OpenCV**: v. `2.4.13`.
- **Dlib**: v. `19.7`.

No se considera el espacio en disco para almacenar base de datos, modelo de reconocimiento facial y de microexpresiones oculares.

5. Contactos

Los programas de entrenamiento del modelo de microexpresiones oculares, normalización de imágenes, extracción de características faciales, duda, requerimiento e información complementaria. Favor contactase al correo electrónico `gonzalo-a@hotmail.com`

B. Gráficos Elección de Algoritmos de Reconocimiento Facial

Este capítulo presenta las figuras que apoyan la elección del algoritmo de reconocimiento facial tradicionales: *Eigenfaces, Eigenfaces s/3* y *Fisherfaces*.

B.1. Comparación de Iluminación

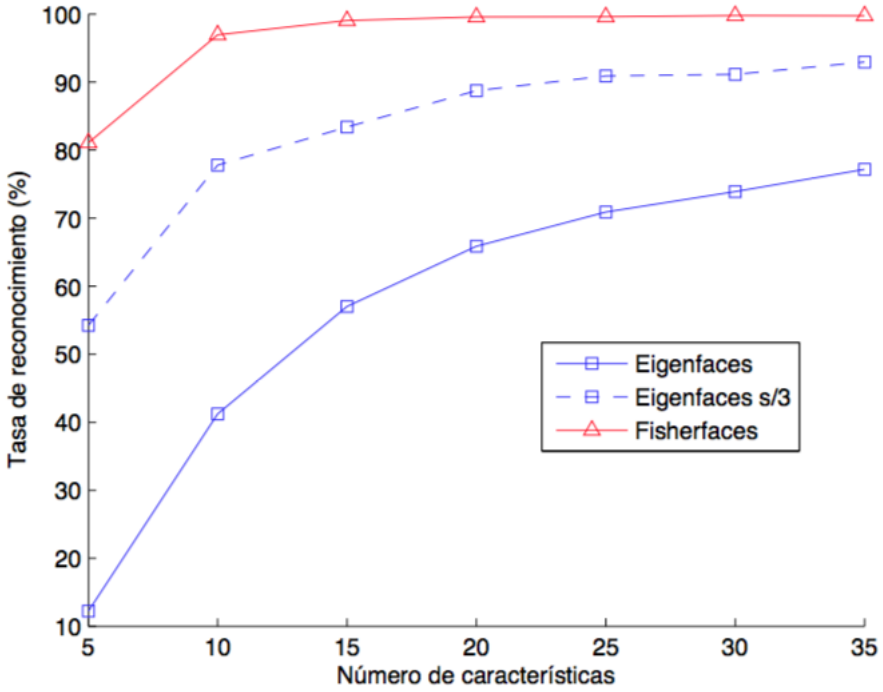


Figura B.1: Comparación del desempeño de los algoritmos en función del número de características, utilizando variaciones de iluminación.

B.2. Comparación de Pose

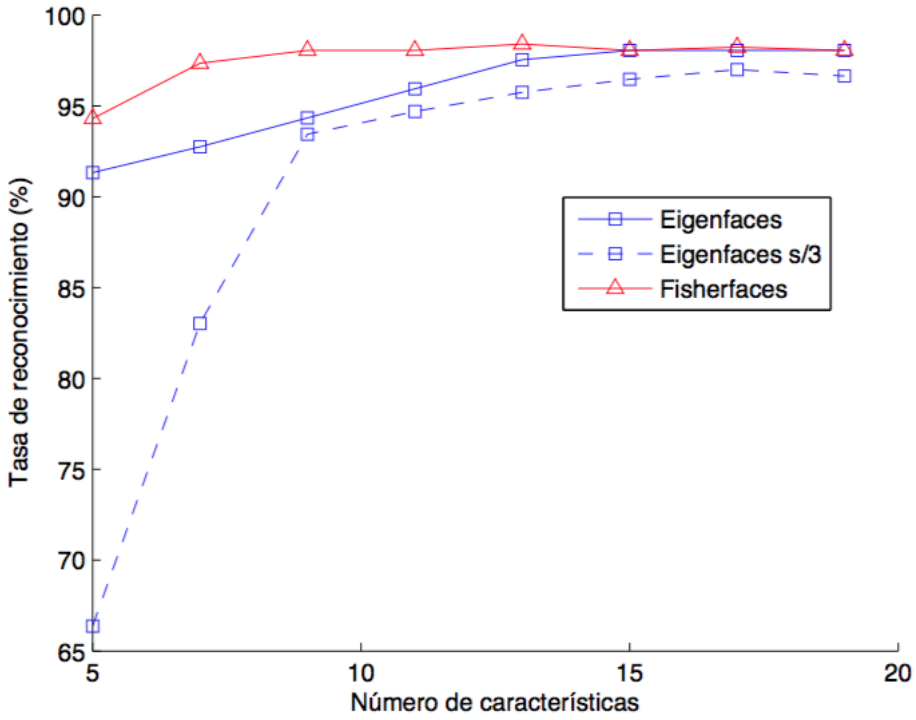


Figura B.2: Comparación del desempeño de los algoritmos en función del número de características, utilizando imagágenes de caras en distintas poses.

C. Dependencias de Instalación

A continuación se listan las dependencias necesarias para el correcto funcionamiento de la API, dentro de las que se enuncian dependencias de *Python*, para el tratamiento de imágenes, operaciones de clasificación, reconocimiento facial, operaciones matemáticas, entre otros.

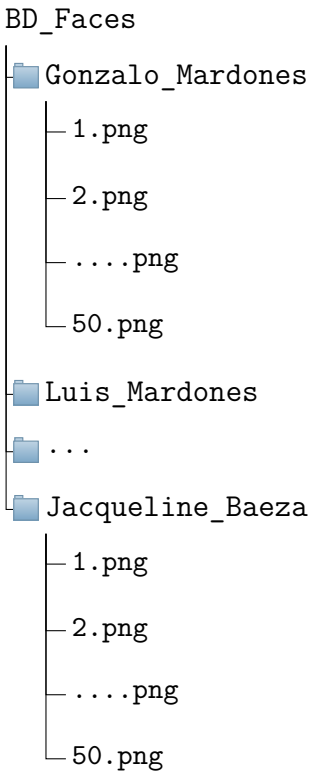
1. `$ sudo yum update`
2. `$ sudo yum upgrade python-setuptools`
3. `$ sudo yum install python-pip python-wheel`
4. `$ sudo install numpy opencv*`
5. `$ sudo yum install cmake python-devel numpy gcc gcc-c++`
6. `$ sudo yum install gtk2-devel libdc1394-devel libv4l-devel`
7. `$ sudo yum install gstreamer-plugins-base-devel`
8. `$ sudo yum install boost*`
9. `$ sudo yum install redhat-rpm-config`
10. `$ sudo yum install python-matplotlib`
11. `$ sudo yum install tkinter`
12. `$ sudo yum install python-imaging`
13. `$ sudo python setup.py install`

14. `$ sudo pip install -upgrade pip`
15. `$ sudo pip install scipy`
16. `$ sudo pip install dlib`
17. `$ sudo pip install pyglet`
18. `$ sudo pip install -U scikit-learn`
19. `$ sudo pip install -U scikit-image`
20. `$ sudo pip install h5py`
21. `$ sudo pip install -no-deps git+git://github.com/Theano/Theano.git`
22. `$ sudo pip install -upgrade -I setuptools`
23. `git clone https://github.com/Theano/Theano`
24. `cd Theano`
25. `$ sudo pip install Image`
26. `$ sudo pip install imutils`
27. `$ sudo pip install Flask`
28. `$ sudo pip install Flask-Images`

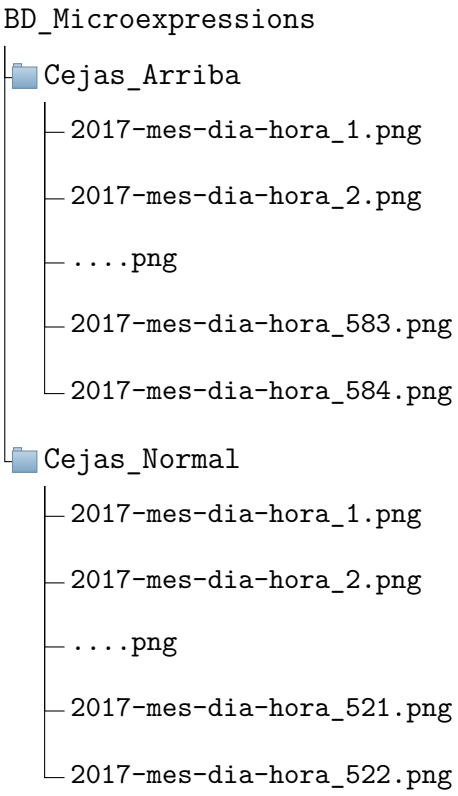
D. Base de Datos

A continuación se muestran las bases de datos utilizadas en el desarrollo de la API. La base de datos de reconocimiento facial, lista las imágenes de una persona en particular. La base de datos de microexpresiones oculares lista imágenes de los directorios Cejas_Arriba y Cejas_Normal.

D.1. Reconocimiento Facial



D.2. Microexpresiones Oculares



E. FME API

A continuación se muestran las imágenes utilizadas en las pruebas desarrolladas en la API. La imagen `t10.png` presenta una cara difusa, la imagen `t11.png` muestra el rostro de una persona identificada en la base de datos, la imagen `t12.jpg` pertenece a una cara con cejas en estado normal. Por último, la imagen `t13.jpg` muestra a una persona que no se encuentra en la base de datos y posee sus cejas arriba.

E.1. Fotografías Usadas en Pruebas

(a) `t10.png`.

(b) `t11.png`.

(c) `t12.jpg`.

(d) `t13.jpg`.