

Grado Universitario en Ingeniería en Tecnologías
Industriales
2018-2019

Trabajo Fin de Grado

“Sistema de detección de emociones para su aplicación en navegación semántica”

Jose Luis Cernuda Cueto

Tutor/es

Jonathan Crespo Herrero

8 Julio 2019



[Incluir en el caso del interés de su publicación en el archivo abierto]

Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Deri**

Agradecimientos

Deseo expresar ante todo mis agradecimientos a mi hermano Sergio, por su apoyo mientras realizaba el trabajo y su ayuda haciendo de sujeto para las pruebas del mismo; también por intentar molestarme mientras lo hacía, aunque sepa que nunca lo consigue. También a mis padres, a mi madre porque sin ella ni siquiera estaría aquí y a mi padre que siempre está preocupado por mi evolución académica. Mi agradecimiento asimismo a mi tutor quien me ha guiado y orientado en la realización del trabajo.

Resumen

Las emociones son una parte muy importante en el ser humano, afectan en gran medida a las personas y a menudo se toman decisiones guiados por estas. Es por ello que contienen una información muy valiosa acerca de las personas y su entorno. Debido al valor de esta información vamos a desarrollar un sistema automático para la detección de emociones y se analiza como expresamos las personas las emociones.

Para el desarrollo del sistema se va a hacer uso de las técnicas de visión artificial y aprendizaje automático para tratar de predecir las emociones mediante el análisis de la expresión facial. Además, ha sido implementado el sistema en ROS para su posible aplicación en robótica.

Una vez desarrollado el sistema se compara el rendimiento de distintos algoritmos de aprendizaje tales como Máquinas de Soporte Vectorial, Bosques Aleatorios, K-Vecinos y Perceptrón Multicapa y se analiza la forma en la que el sistema obtiene la información sobre la emoción; para ello es medida la influencia de las distintas zonas faciales con métricas típicamente usadas en el aprendizaje automático.

Finalmente se obtiene un sistema con una precisión del 84% de acierto empleando una Máquina de Soporte Vectorial entrenada para detectar 5 emociones: alegría, tristeza, ira, sorpresa y neutra. Se comprueba que la Máquina de Soporte Vectorial es un algoritmo que ofrece una mayor precisión para este problema por su planteamiento y por el conjunto de datos disponible para el entrenamiento. Además, se observa que ciertas zonas faciales como la boca o los ojos proporcionan mayor información que otras como pudiera ser la nariz.

Abstract

Emotions are one of the most important features of human psychology and behavior, they have information about ourselves as well as our surroundings that can be useful for certain tasks in robotics. That is the reason why we developed a system for the detection of emotions based on the facial expression analysis.

For the development of the system we made use of machine learning techniques that made it possible to build the system just with some examples of the same type of data we want to predict. We also implemented the system in ROS for a possible application in robotics related with semantic navigation.

For the development first the data was extracted and organized. Once the data is ready a feature selection is made as well as a parameter tuning for getting the most out of the learning process. After all this is done, learning takes place and the implementation in ROS is done once it has finished.

To determine the importance of each facial zone, a study was carried out by looking at the performance of the system when we do not use the information provided by that facial zone and when just that information is available for the learning algorithm. A comparison of different learning algorithms was also done by comparing the accuracy of four different algorithms: SVM, KNN, Random Forest and MLP, this study is not only sensitive to the type of problem but also the approach it is taken as well as the data available to us.

Finally, a system with a 84% precision was obtained with the use of an SVM which fulfils with the objective of this project, but also leaves lots of new possible improvements to be done in the future that can result in a better performance and new functionalities.

Índice

Capitulo 1: Introducción.....	1
Motivacion	1
Objetivos	2
Capitulo 2: Estado del arte.....	3
Tecnicas visión.....	3
Detección mediante la expresión facial	3
Detección mediante la gesticulación y posición corporal.....	4
Tecnicas audio	5
Enfoque multimodal.....	5
Teoria emocional.....	7
Conexión entre la expresión facial y las emociones	7
Número de emociones que existe	8
Capitulo 3: Arquitectura del sistema	9
Lenguaje de programación.....	10
ROS	11
Librerías.....	12
Machine Learning.....	13
Máquina de Soporte Vectorial	15
K-Vecinos.....	17
Random Forest	16
Perceptrón Multicapa	18
Capitulo 4: Desarrollo del sistema	19
Obtención de los datos	19
Organización de los datos	20
Extracción puntos faciales y selección de atributos.....	22

Entrenamiento del algoritmo	25
Uso del algoritmo	29
Integración en ROS.....	30
Capítulo 5: Pruebas con distintas partes faciales y algoritmos.....	36
Influencia de las zonas faciales	36
Comparativa algoritmos.....	46
Capítulo 6: Marco Regulatorio	50
Análisis de la legislación aplicable.....	50
Estándares técnicos.....	50
Capítulo 7: Entorno socio-económico	52
Presupuesto	52
Impacto socio-económico	52
Capítulo 8: Conclusiones y trabajos futuros.....	54
Conclusiones	54
Trabajos Futuros.....	55
Bibliografía.....	57

Índice de figuras

Figura 1: Detección de emociones usando sensores de flexión	5
Figura 2: Sistema de detección basada en el habla de una persona.....	6
Figura 3: Flujo de la información del sistema	10
Figura 4: Comunicación entre procesos en ROS.....	11
Figura 5: Sesgo y varianza.....	15
Figura 6: Ejemplo de clasificación con SVM.....	16
Figura 7: Ejemplo de clasificación con bosques aleatorios.....	17
Figura 8: Ejemplo de clasificación con knn	17
Figura 9: Esquema perceptron multi-capas.....	18
Figura 10: Ejemplos de datos de entrenamiento con las diferentes emociones.....	21
Figura 11: Organización de las imágenes por emoción.....	21
Figura 12: Extracción de las caras en las imágenes y reorganización en un nuevo directorio.....	22
Figura 13: Matriz de confusión con las coordenadas como atributos	23
Figura 14: Matriz de confusión con las distancias entre puntos.....	24
Figura 15: Entrenamiento del algoritmo.....	26
Figura 16: Entrenamiento del algoritmo.....	26
Figura 17: Entrenamiento del algoritmo.....	27
Figura 18: Matriz de confusión con el ajuste de parámetros.....	28
Figura 19: Código para hacer la predicción.....	29
Figura 20: Código para hacer la predicción.....	30
Figura 21: Inicialización del nodo y publicación de la información	31
Figura 22: Nodo receptor de la información	31
Figura 23: Archivo CMakeLists.txt.....	32
Figura 24: El sistema detectando la emoción de alegría.	33

Figura 25: El sistema detectando la emoción de ira.	33
Figura 26: El sistema detectando la emoción neutral.	34
Figura 27: El sistema detectando la emoción de sorpresa.	34
Figura 28: El sistema detectando la emoción de tristeza.....	35
Figura 29: Puntos de interés facial	37
Figura 30: Matriz de confusión sin contorno.....	40
Figura 31: Matriz de confusión solo contorno de la cara	40
Figura 32: Matriz de confusión sin cejas.....	41
Figura 33: Matriz de confusión solo cejas.....	41
Figura 34: Matriz de confusión sin boca	42
Figura 35: Matriz de confusión solo boca	42
Figura 36: Matriz de confusión sin nariz.....	43
Figura 37: Matriz de confusión solo nariz.....	43
Figura 38: Matriz de confusión sin ojos	44
Figura 39: Matriz de confusión solo ojos	44
Figura 40: SVM.....	47
Figura 41: Random Forest	47
Figura 42: Multi Layer Perceptron	48
Figura 43: KNeighbors	48

Índice de tablas

Tabla 1: Peso de cada clase en el aprendizaje	28
Tabla 2: Resultados del entrenamiento con diferentes zonas faciales	37
Tabla 3: Precisión con distintos algoritmos de aprendizaje	46

Capítulo 1: Introducción

Este capítulo tiene un carácter introductorio donde se exponen las motivaciones que han llevado a su realización, así como los objetivos planteados

1.1 Motivación

Las emociones que sienten las personas contienen una gran cantidad de información acerca de quien las siente, ya que afectan en gran medida el comportamiento y las decisiones; pero no solo contienen información acerca de quien las siente, también sobre su entorno, ya que el entorno tiene una influencia directa sobre estas.

La principal motivación de este trabajo es desarrollar un sistema para reconocer las emociones y, por tanto, acceder a la información que se puede extraer de estas y tratar de determinar el mejor método para conseguir esta información haciendo una comparación entre diferentes sistemas y un análisis de las características que contienen información para tratar de conocer que elementos y características deberemos tener en cuenta a la hora de desarrollar un sistema de este tipo.

Además, debido a la creciente importancia de la robótica, el conocimiento de las emociones resulta interesante ya que nos puede ayudar aportando información para el desarrollo de funcionalidades como la navegación semántica. Al contener información acerca del estado de una persona y su entorno, conocer las emociones ayuda a contextualizar la tarea que se le ha sido asignada al robot y también ayuda a conocer el lugar donde se encuentra ya que diferentes entornos producen diferentes emociones; con esta información, se facilita el cumplimiento de la tarea siempre que el robot se desplace en un entorno con personas.

1.2 Objetivos

El objetivo de este proyecto es el desarrollo de un sistema de detección de emociones basado en el reconocimiento de la expresión facial de los sujetos a los que se les quiere aplicar el sistema mediante la toma de imágenes de su rostro. El sistema desarrollado debe ser además integrado en ROS para su posible aplicación en robótica.

Además, se proponen una serie de objetivos secundarios en el desarrollo del sistema:

- Determinar las zonas faciales con mayor cantidad de información para la detección.
- Tratar de entender porque ciertas zonas de la cara aportan más información
- Comparar algoritmos de aprendizaje automático para determinar que algoritmo tiene una precisión mayor.
- Tratar de entender porque ciertos algoritmos dan una precisión mayor que otros en el problema y con los recursos empleados.

Capítulo 2: Estado del arte

Detectar la emoción de una persona basado en su expresión facial es una tarea realizada a diario por cualquier persona y que generalmente nos resulta fácil, tenemos una habilidad innata para esto. Pero a pesar de esta facilidad innata para las personas es un problema de una gran complejidad cuando se trata de hacerlo mediante computación.

A la hora de detectar la emoción de forma automática existen diversos canales a través de los cuales obtener la información para la clasificación. Se puede hacer el reconocimiento usando la información procedente de factores visuales; uno de ellos es la expresión facial. También mediante el canal visual se pueden analizar los gestos y la posición corporal que aportan información acerca del estado emocional de las personas. Otro canal por el que es posible obtener la información es el auditivo, analizando el habla, la expresión oral de una persona y las señales sonoras. Existen también técnicas basadas en la expresión escrita de una persona, mediante el análisis del vocabulario, la gramática y otros aspectos lingüísticos [10].

También existen enfoques multimodales; es decir, utilizan información de los diversos canales, analizan la expresión facial, los gestos y el habla para determinar, con la información conjunta, las emociones.

2.1 Técnicas basadas en la visión

Las principales técnicas que hacen uso de la visión para la detección de emociones son dos, el análisis de la expresión facial y el análisis de la gesticulación y la posición corporal.

2.1.1 Detección mediante la expresión facial.

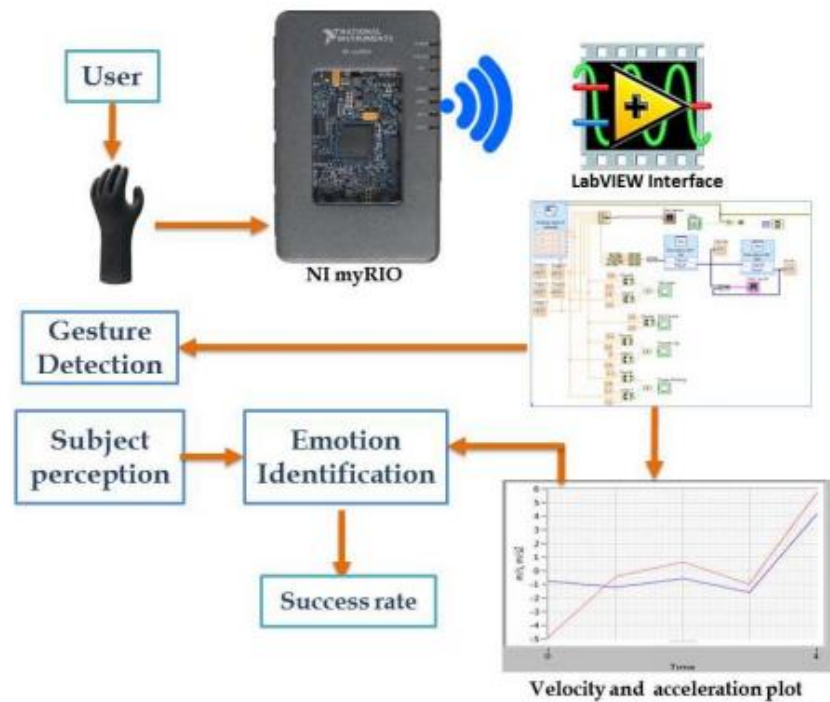
Para conseguir extraer información de la expresión facial se realizan técnicas de visión artificial; estas incluyen en la mayoría de los casos uso de modelos de detección facial para saber ubicar la cara que se quiere analizar en la imagen y algoritmos de aprendizaje automático [1-3].

Hay dos formas principales de abordar el problema de la detección de la expresión facial; en la primera se utilizan típicamente un tipo de redes neuronales profundas conocidas como redes neuronales convolucionales [1-2]. Estas redes neuronales están basadas en la estructura de la corteza visual del cerebro y consiguen excelentes resultados cuando se dispone de una gran cantidad de datos y se le introducen las imágenes sin procesar. Debido a su inspiración en las redes neuronales de la corteza visual, son idóneas para tareas de visión artificial. También se puede optar por tratar de obtener los puntos de interés facial y usarlos en algoritmos de aprendizaje más sencillos como máquinas de soporte vectorial. Además, se pueden considerar otras características como la activación muscular [3].

Entre las distintas organizaciones y empresas que emplean este tipo de solución destaca Emotient, que fue adquirida por Apple Inc en 2016. [4-5] Emotient es una empresa que se dedica a ayudar a detectar las emociones de los clientes de las empresas por las que eran contratados con fines analíticos del marketing.

2.1.2 Detección mediante la gesticulación y posición corporal

La detección de la gesticulación es un campo menos desarrollado que la detección de la expresión facial y se puede hacer por métodos de visión [6], aunque también hay otro tipo de sensores como sensores de flexión que pueden servir para este propósito como se muestra en [7], el esquema de este sistema se muestra en la Figura 1 que explica como con los sensores de la mano se extraen la velocidad y la aceleración de la mano que se usa para identificar el gesto y la emoción. Una forma posible es monitorizar la estructura esquelética y ver los movimientos corporales y analizar la relación entre la posición corporal y las emociones con algoritmos de aprendizaje automático. La detección de la postura corporal y los gestos se puede realizar con diversos sensores, como por ejemplo un Kinect. En [8] se logra una precisión del 81% utilizando un sistema basado en la postura corporal.



[Figura 1] Detección de emociones usando sensores de flexión. [7]

2.2 Técnicas basadas en el audio

En técnicas de audio se puede distinguir dos principales enfoques:

- Analizar las propiedades acústicas
- Analizar el lenguaje empleado

En el primer caso se encuentra la relación entre las propiedades acústicas de la voz y las emociones mediante el entrenamiento de un algoritmo de aprendizaje automático utilizando atributos como el tono, timbre y otras características acústicas de la voz. Se puede conseguir un 86% de precisión como se explica en [9].

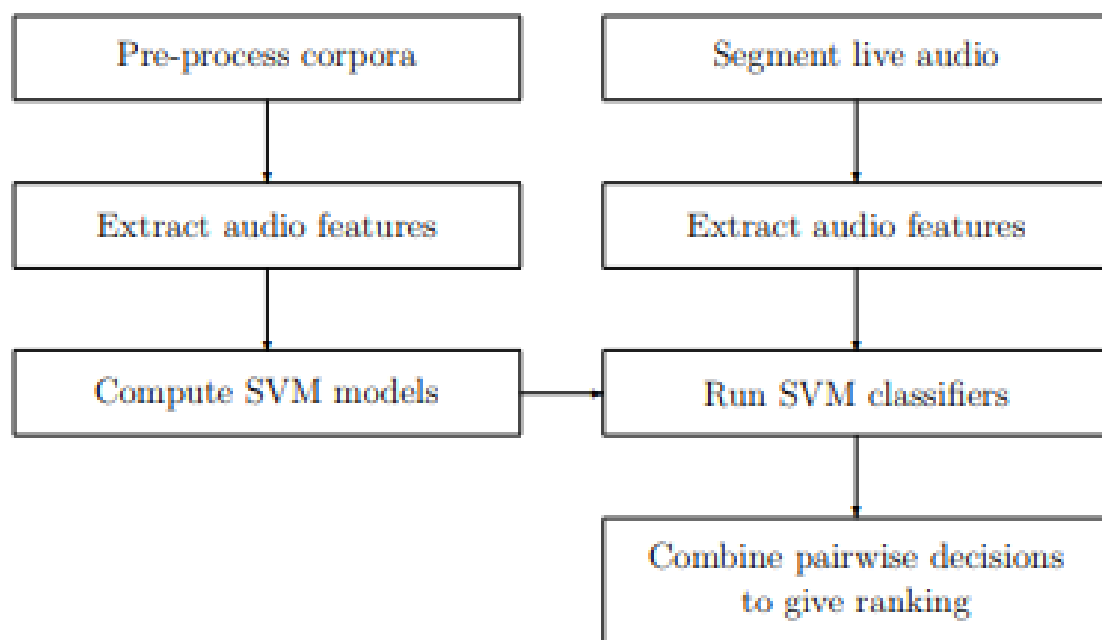
Al analizar el lenguaje típicamente se analiza el vocabulario empleado y las connotaciones que tienen las palabras empleadas, para ello no es explícitamente necesario el uso de algoritmos de aprendizaje automático, pero se debe tener acceso una base de datos que contenga la información acerca del léxico en un sistema como el de la figura 10[10].

2.3 Enfoque multimodal

También existen sistemas que usan una combinación de los anteriores para conseguir la máxima información disponible. Un ejemplo del uso de detección multimodal de emociones es Afectiva, se trata de una empresa que cuenta con un software de

detección de emociones que emplea tanto visión artificial como análisis del habla. Además, trabajan en aplicaciones para el uso de su software para la seguridad de los ocupantes de un vehículo como la monitorización de los ocupantes para detectar posibles riesgos y preocupaciones como se muestra en la figura 2[11].

También en la Universidad Carlos III de Madrid se han realizado interesantes investigaciones en la detección multimodal de emociones y el desarrollo de un sistema con excelentes resultados como en [12], que además fue probado experimentalmente en el robot Maggie de la UC3M.



[Figura 2] Sistema de detección basado en el habla de una persona, [10]

2.4 Teoría sobre emociones

Al tratar de hacer un sistema de detección de emociones se ha de conocer en primer lugar de donde extraer la información que puede indicar la emoción. También ha de definirse el número de emociones que hay. El sistema de detección que se realiza se basa en la extracción de información de la expresión facial y distingue entre 5 emociones, siendo estas ira, tristeza, sorpresa, alegría y neutro. La razón por la que se ha decidido esto se basa en los estudios y teorías científicas de las emociones de los que se trata a continuación.

2.4.1 Conexión entre la expresión facial y las emociones

El estudio de las emociones es realizado por la ciencia de la psicología, a lo largo de la historia han aparecido una gran cantidad de teorías sobre la psicología emocional que tratan de entender y explicar como se producen las emociones y que propósito biológico y social tienen [13-14]. Charles Darwin fue uno de los primeros en atreverse a buscar respuestas y escribió el libro [15] en el que trata de aplicar su teoría de la evolución a la teoría emocional y concluye que la forma en que los seres vivos expresan las emociones que sienten a través de la expresión facial y los cambios en el rostro tiene una importante función comunicativa dentro de una especie. Darwin por tanto entiende que la expresión facial expresa la emoción que se está sintiendo. Esta teoría es conocida como la teoría evolutiva de la emoción.

Más tarde, en 1884, los científicos William James y Carl Lange propusieron, de forma independiente cada uno, una teoría basada en la neurología para el entendimiento de como el sistema nervioso procesa los estímulos y nos hace reaccionar físicamente a estos expresando nuestras emociones, entre otras cosas, mediante la expresión facial [16]. Aunque no es una teoría completa, supuso un gran avance en la materia. A esta teoría se la conoce como la teoría de James-Lange en honor a estos dos psicólogos.

Otra importante teoría sobre las emociones es la conocida como teoría de Cannon-Bard, desarrollada por Walter Cannon después de que se basara en experimentos realizados por Philip Bard [17], completó y rebatió a la teoría de James-Lange en ciertos aspectos, mediante el uso de la neurología. Sugirió que una parte del encéfalo conocido como

tálamo es la responsable del envío de las señales nerviosas que nos hacen expresar las emociones.

Muchas de estas teorías que se han desarrollado nos dicen que la expresión facial es producto de la emoción que estamos sintiendo y por tanto podemos tratar de detectarla mediante una visualización de la cara y el análisis de su expresión.

2.4.2 Número de emociones que existe

Otro tema dentro del estudio de las emociones es determinar el número de emociones que existen y cuales son. Respecto a este tema no hay un consenso universal entre los psicólogos y el número exacto varía dependiendo de las fuentes a las que nos refiramos.

El psicólogo Paul Ekman, uno de los más destacados del siglo XX, publicó un notable artículo en la revista Science en 1983 [18 - 19]. tras el estudio de una tribu aislada en Papua Nueva Guinea. En el artículo se argumenta que hay 6 emociones básicas; estas emociones son ira, asco, miedo, tristeza, sorpresa y alegría. La teoría de las seis emociones básicas es una de las más aceptadas entre la comunidad científica.

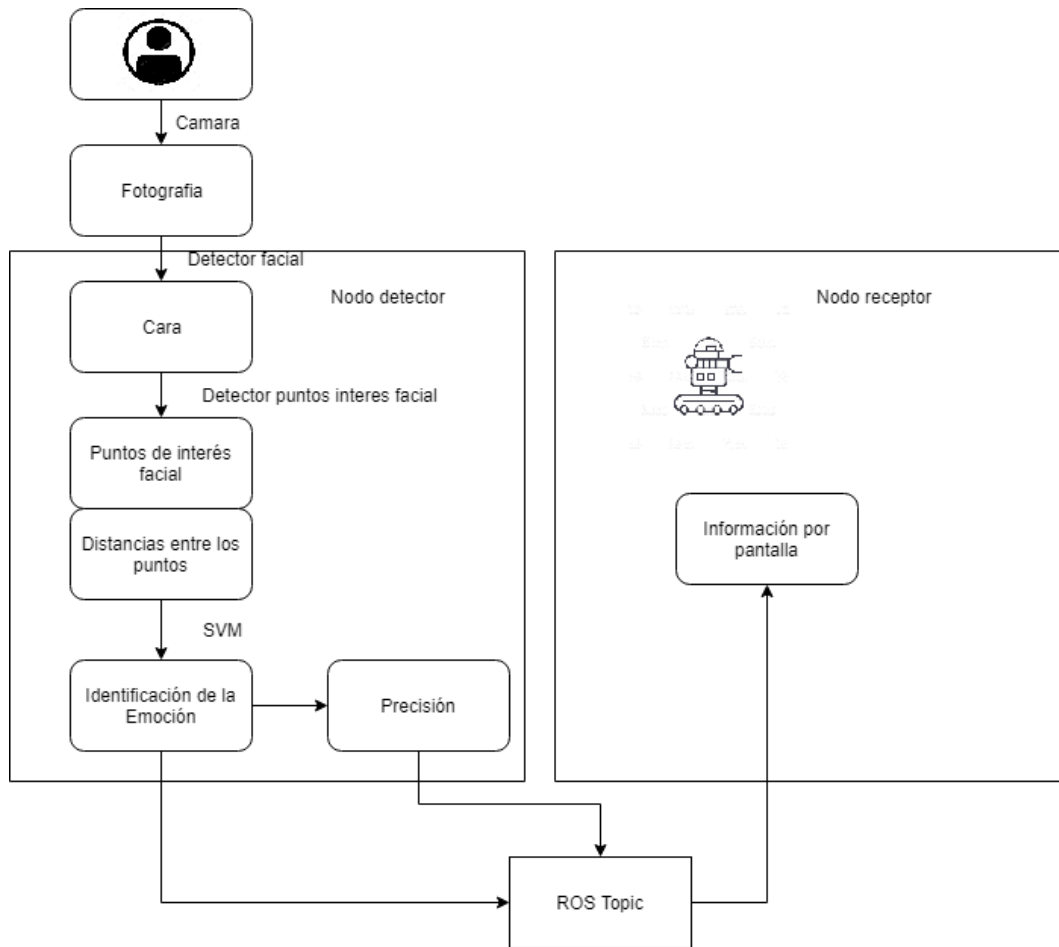
Recientemente, científicos de la universidad de Glasgow publicaron un estudio [20] en el que analizan las señales nerviosas que se envían cuando se tienen las emociones básicas definidas por Paul Ekman. Descubrieron que el miedo y la sorpresa, así como el asco y la ira tienen grandes similitudes, mientras que el resto tienen una señal notoriamente distinta por lo que argumentan que las emociones básicas son 4 y no 6.

Teniendo en cuenta los últimos estudios realizaremos el sistema de detección de emociones para las 4 emociones mencionadas anteriormente más una neutra que indica la ausencia de una de las emociones básicas.

Capítulo 3: Arquitectura del sistema

El sistema funciona de acorde a como se muestra en la figura 2; primero el sujeto es capturado por la cámara en una fotografía, esta fotografía es procesada por el detector facial extrayendo la cara de la fotografía, una vez hecho esto, se procesa la imagen de la cara con un detector de puntos de interés facial que nos devolverá estos puntos de interés con lo que podremos calcular todas las combinaciones de distancias entre estos puntos. Las distancias son la información que empleamos para entrenar el sistema y para reconocer las emociones, una vez que el sistema hace una predicción, este obtiene la emoción que corresponde a la predicción y una precisión; esta información es enviada a través de un topic de ROS a un nodo receptor que muestra por pantalla esta información con propósitos de prueba.

La cámara con la que se han realizado todas las pruebas graba en una resolución de 640x480, el uso de una resolución baja tiene una ventaja y una desventaja. La ventaja es que la potencia de procesamiento se reduce en gran medida ya que las operaciones con video requieren de una gran potencia y esto podría ser un gran impedimento; como contrapartida, la falta de resolución puede hacer más inexacto la detección de los puntos de interés facial y hacer que no se estabilicen en la imagen.



[Figura 3] Flujo de información en el sistema

3.1 Decisión del lenguaje de programación

Antes de comenzar a escribir código se ha de decidir primeramente por un lenguaje de programación, para este proyecto los lenguajes Python y C++ son muy posiblemente los más indicados debido a su compatibilidad con ROS y la cantidad de librerías disponibles. Los motivos para destacar estos dos lenguajes por encima de los otros radican en las herramientas que se usarán, para implementar código en ROS debemos usar Python o C++ por lo que ya de partida sería una complicación añadida el desarrollo en otro lenguaje. Además, librerías como OpenCV, dlib; dos de las más usadas y potentes en lo que a visión por ordenador se refiere; solo cuentan con interfaz en Python y C++.

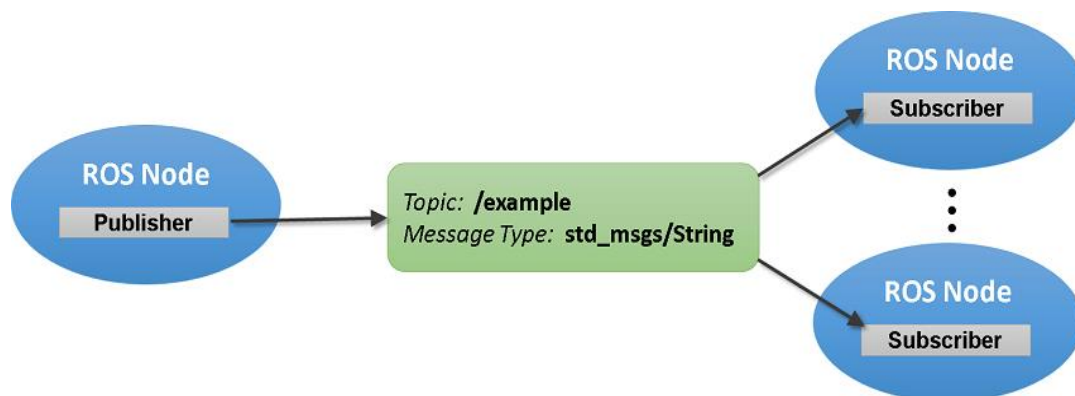
Finalmente, el lenguaje que se utilizará es Python, ya que ofrece ciertas ventajas frente a C++ para este tipo de problema.

3.2 ROS

El Sistema Operativo Robótico [21] (ROS por sus siglas en inglés) es un middleware para aplicaciones de robótica, desarrollado por Willow Garage y que se ha convertido prácticamente en un estándar de la industria. En nuestro caso usaremos la versión Kinetic Kame, orientada para Ubuntu 16.04, cuyo lanzamiento fue el 23 de Mayo de 2016. ROS nos permite el desarrollo de software orientado a aplicaciones robóticas.

El Sistema Operativo Robótico (ROS por sus siglas en inglés) es un middleware para aplicaciones de robótica, desarrollado por Willow Garage y que se ha convertido prácticamente en un estándar de la industria. En este caso usaremos la versión Kinetic Kame, orientada para Ubuntu 16.04. ROS permite el desarrollo de software orientado a aplicaciones robóticas.

Los principales conceptos que debemos de conocer de ROS son los nodos, los mensajes y los topics. Los nodos son procesos que realizan una determinada tarea, estos son programados. Los topics son canales de comunicación entre nodos sobre un contexto concreto y donde los nodos publican y extraen información. Los mensajes son los datos que son publicados en los topics y contienen la información que requiere un nodo o que un nodo entrega, la Figura 3 muestra visualmente la comunicación entre nodos.



[Figura 4] Comunicación entre procesos en ROS.

<https://es.mathworks.com/help/robotics/examples/get-started-with-ros.html>, 12/06/19

Teniendo en cuenta todo esto, se podría crear un nodo cuyo proceso consiste en detectar las emociones y lo publica en un topic. A este topic se conecta otro nodo cuyo proceso es informar por pantalla la emoción detectada.

3.3 Librerías

Todas las librerías que se usan en este proyecto son open-source por lo que es posible usarlas de forma libre y gratuita.

Las principales librerías de las que se harán uso son las siguientes:

-OpenCV [22]: La librería de visión por computador por excelencia, en este ámbito es la librería más ampliamente usada que otorga una gran variedad de posibilidades. Permite hacer todas las operaciones básicas de visión, abrir imágenes, obtener imágenes de la cámara, realizar transformaciones en las imágenes (ej. Pasar a blanco y negro una imagen), trabajar con matemáticas en la matriz de la imagen, incluso permite la detección de ciertos objetos en la imagen fácilmente gracias a modelos que tiene incluidos basados en el sistema de Viola-Jones [23] e incluso sería posible implementar algoritmos de aprendizaje automático únicamente con esta librería. Además, aunque no vayan a ser usados en este proyecto, OpenCV cuenta con varios módulos extra que, aunque no forman parte de la librería oficial debido a cierta falta de estabilidad, pueden añadir funciones interesantes (ej. Detección de puntos de referencia en la cara o Facial-Landmarks).

-dlib [24]: Una librería que ofrece funciones en distintos ámbitos, entre los que se encuentra el procesamiento digital de imágenes, la principal característica por la que será usado es que cuenta con un modelo pre-entrenado de detección de puntos característicos de la cara estable y que ofrece excelentes resultados.

-Scikit-learn [25]: Una de las librerías más populares de aprendizaje automático para el lenguaje Python que incluye los algoritmos más usados. Contiene algoritmos de clasificación, regresión; aprendizaje supervisado y no supervisado; etc. Debido a la naturaleza de nuestro problema usaremos algoritmos de clasificación con aprendizaje supervisado, ya que lo que se busca es precisamente clasificar la emoción correspondiente. Debido a que esta librería está construida sobre Numpy, SciPy y matplotlib; estas tres últimas son un requisito indispensable.

-joblib [26]: El propósito con el que se hace uso de joblib es para la persistencia del modelo entrenado y así no ser necesario entrenarlo cada vez que se desea usarlo. La razón por la que se va a incluir joblib en vez de otras es que es la librería que recomiendan los desarrolladores de scikit-learn para combinar con su librería ya que argumentan que es más eficiente en objetos que llevan arrays de Numpy internamente [27].

3.4 Machine learning

La inteligencia artificial es un área del conocimiento dentro del campo de la informática que trata de crear máquinas inteligentes, con capacidad para tomar decisiones con juicio y la simulación de los procesos inteligentes de los humanos. Dentro de esta área se encuentra una rama llamada aprendizaje automático (machine learning en inglés). El principal objetivo del aprendizaje automático es lograr que una máquina pueda resolver un problema sin que sea programada explícitamente para ello. Para ello se basa en la habilidad que poseemos los humanos del aprendizaje, mediante la experiencia se trata de adquirir conocimiento para ser capaz de resolver un problema. Aplicando esto a una máquina podemos lograr que aprenda de los datos que poseemos a resolver la tarea propuesta.

Dentro del aprendizaje automático podemos encontrar tres ramas principales:

- Aprendizaje supervisado.
- Aprendizaje no supervisado.
- Aprendizaje por refuerzo.

En este proyecto únicamente haremos uso del primer tipo, el aprendizaje supervisado. Usaremos los datos de entrada y los resultados deseados para entrenar a la máquina. Los datos de entrada son conocidos como atributos o características, una buena selección de los atributos es fundamental para el correcto aprendizaje de la máquina ya que si decidimos entrenar la máquina para que aprenda a resolver un problema en base a información irrelevante esta no conseguirá resolver el problema o no dará un rendimiento óptimo en su resolución. Los resultados deseados en este caso serán las emociones a detectar por la máquina.

Debido a la naturaleza del proyecto, deberemos usar el aprendizaje automático en un

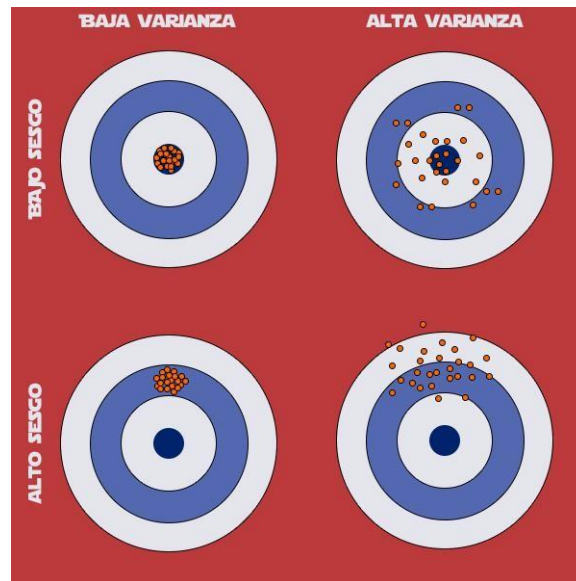
problema de clasificación. Al tratarse de las emociones, esto significa que la máquina deberá aprender a clasificar dentro de las emociones que existen a la emoción que siente la persona a la que se le aplica el sistema. Como la clasificación que se va a realizar es entre las 5 clases correspondientes a las emociones se trata de un problema de clasificación multiclase.

Cuando se intenta entrenar un algoritmo, hemos de tener distintas consideraciones en cuenta; el sistema puede sufrir un sobreajuste, esto quiere decir que el sistema aprende de los datos que le hemos introducido y aprende a clasificar estos datos de manera correcta, pero falla a la hora de generalizar en el problema. Así como el sobreajuste, la sobre generalización supone también un problema ya que no logramos capturar la estructura que siguen los datos.

También se ha de conocer los conceptos de sesgo y varianza para analizar el rendimiento que tienen los algoritmos.

Cuando un algoritmo no es capaz de capturar la fórmula de separación de dos clases esto es un indicador de un alto sesgo; si por ejemplo tratásemos de resolver un problema cuadrático con un algoritmo lineal. Si se tien un gran sesgo no conseguimos aprender las relaciones relevantes entre los atributos y los resultados.

La varianza por el contrario consiste en el aprendizaje de las fluctuaciones de los datos o el ruido, esto significa que nuestro algoritmo modelizará este ruido y hará clasificaciones más dispersas ya que el ruido no deseado formará parte del cálculo en la clasificación. Este efecto se produce cuando se sobre ajusta un algoritmo. La figura 5 muestra de forma visual y con un símil como afectan la varianza y el sesgo a la precisión.



[Figura 5] Sesgo y varianza. <https://koldopina.com/equilibrio-varianza-sesgo/>, 12/06/19

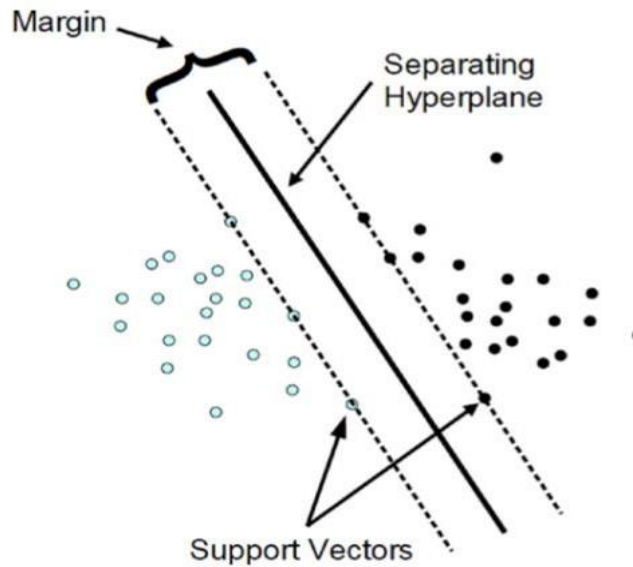
Cuando se trata del sesgo y la varianza siempre hay que hacer un balance entre estos dos, ya que si tratamos de compensar demasiado uno acabaremos sufriendo del otro.

Dentro del aprendizaje supervisado existen distintos algoritmos basados en distintos conceptos matemáticos, los que usaremos son los siguientes [28]:

- Máquina de Soporte Vectorial (SVM, por sus siglas en inglés)
- Bosques Aleatorios (Random Forest en inglés)
- K-Vecinos más próximos (KNN)
- Perceptrón Multicapa (MLP, por sus siglas en inglés)

3.4.1 Máquina de Soporte Vectorial

El principio detrás de la máquina de soporte vectorial es construir un hiperplano en un espacio multidimensional que separe las clases. Para ello la SVM intenta situar el hiperplano tal que haya la distancia máxima entre los puntos más cercanos de las clases (distancia conocida como margen) y clasifique correctamente el máximo número. El hiperplano puede tener distintas formas geométricas pudiendo seguir formas lineares o no lineares tal como se muestra en la Figura 6.

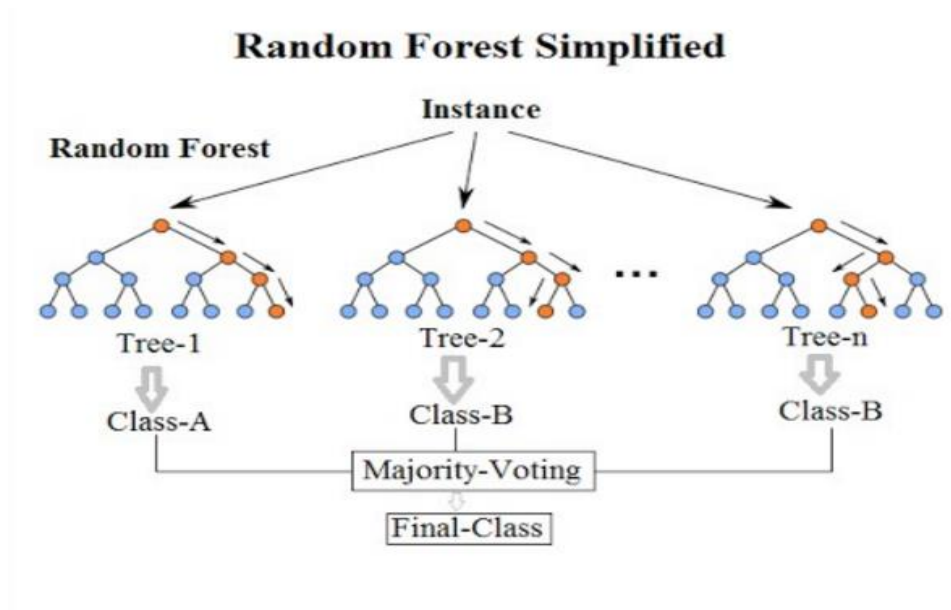


[Figura 6] Ejemplo de clasificación con SVM. <https://en.proft.me/2014/04/22/how-simulate-support-vector-machine-svm-r/>, 12/06/19

Las Maquinas de Soporte Vectorial no requieren una cantidad muy elevada de datos para funcionar correctamente, lo que las aporta ventaja sobre otros algoritmos en ciertas situaciones [29].

3.4.2 Random Forest

El algoritmo de los Bosques Aleatorios se basa en los árboles de decisión. Un árbol de decisión es un algoritmo de aprendizaje que toma las decisiones de clasificación de acorde a un esquema de decisión. Para ello genera un esquema de decisión basándose en los datos con los que se le entrena. En los bosques aleatorios se genera una cierta cantidad de árboles de decisión, se hace la predicción con cada uno de ellos y se hace un recuento de las predicciones. Una vez obtenido el recuento, la clase con mayor número de predicciones es la clase que se predecirá para el nuevo dato [30] como se muestra en la Figura 7.

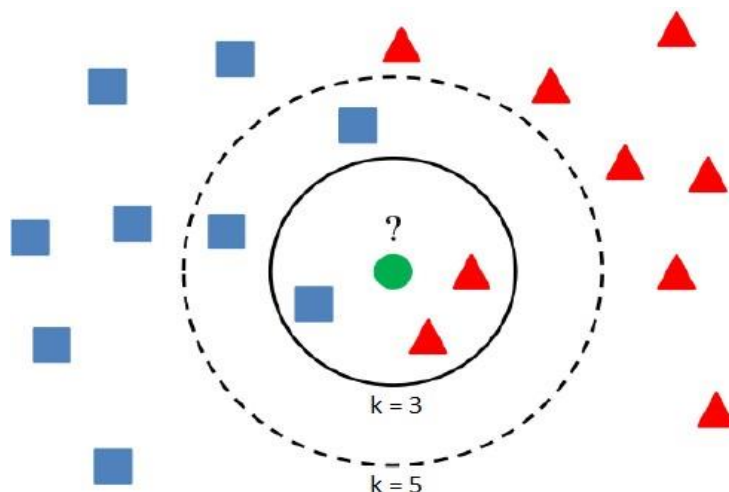


[Figura 7] Ejemplo de clasificación con bosques aleatorios.

<https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d>, 12/06/19

3.4.3 K-Vecinos

En este algoritmo la clasificación se realiza de acorde a las instancias de los datos más cercanos en el espacio multidimensional. Para clasificar una instancia de los datos nos fijamos en las clases de los k puntos más cercanos, la clase con el mayor número de ejemplos dentro de estas k instancias es la clase a la que se clasificará el nuevo valor, la figura 8 muestra de forma visual el funcionamiento de este algoritmo.



[Figura 8] Ejemplo de clasificación con knn.

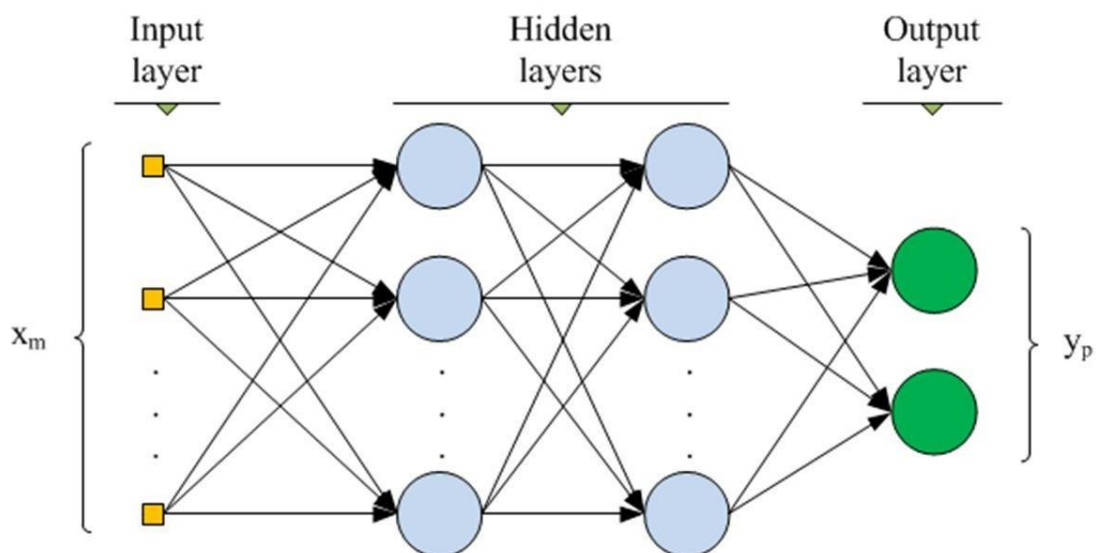
https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos

El valor de k es el parámetro más importante de este algoritmo y hay que tratar de

elegirlo correctamente ya que cambia de forma significativa la precisión [31].

3.4.4 Perceptrón Multicapa

El perceptrón multicapa es un tipo de red neuronal artificial, el termino multicapa se refiere a la formación de múltiples capas que lo componen y permiten resolver problemas no lineales. En un perceptrón multicapa siempre tenemos una capa de entrada, una o varias capas ocultas y una capa de salida, tal como se muestra en la figura 9. Debido a que el número de capas no es fijo uno de los factores que puede afectar al rendimiento de este algoritmo es la disposición de las capas ocultas; podemos variar el número de capas, cambiar su tamaño [32]. La forma en la que el perceptrón multicapa aprende es mediante el algoritmo de propagación hacia atrás.



[Figura 9] Esquema de Perceptrón Multicapa.

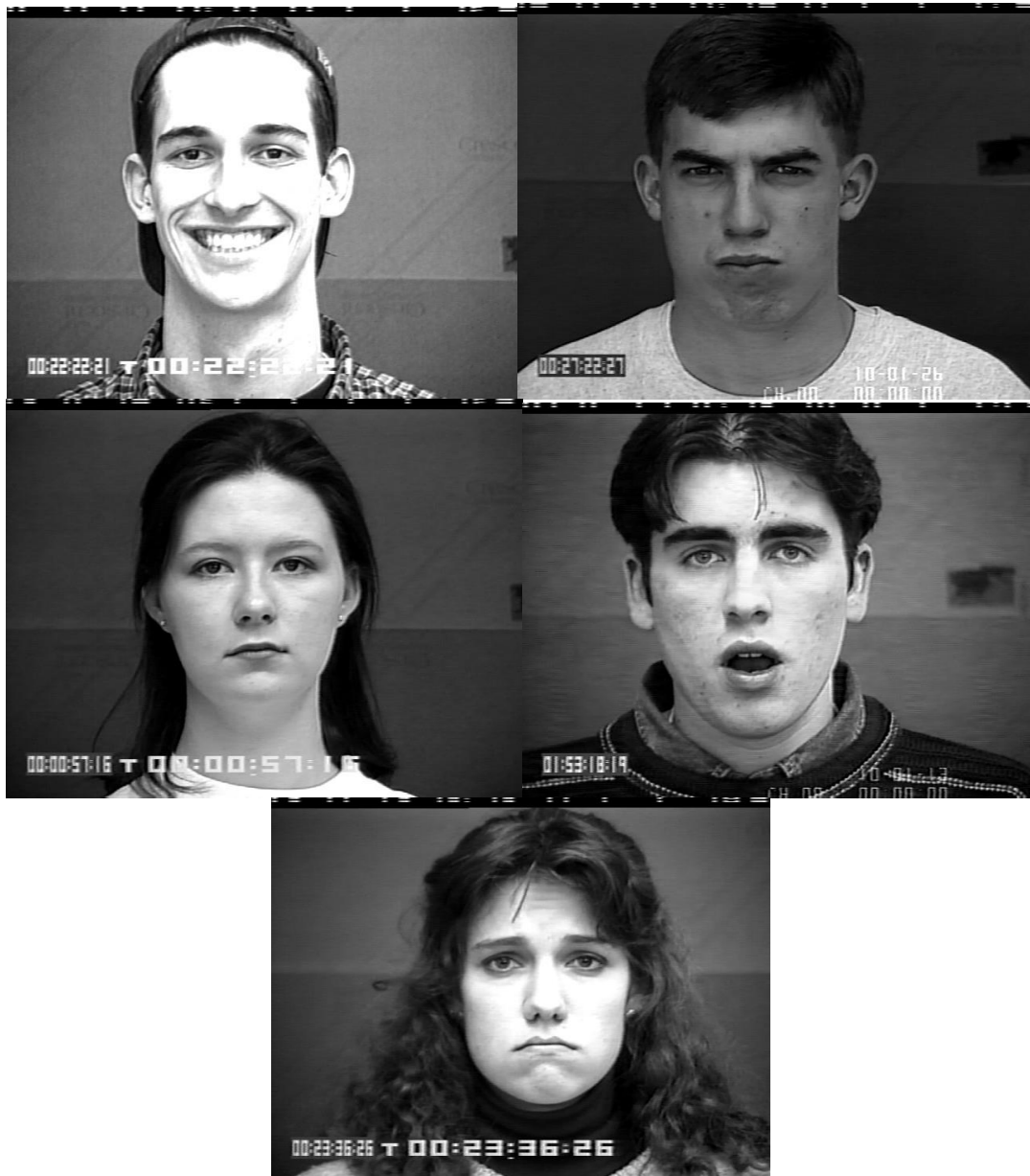
https://www.tutorialspoint.com/tensorflow/tensorflow_multi_layer_perceptron_learning.htm

Capítulo 4: Desarrollo del sistema

El desarrollo del sistema comprende la forma en la que ha sido desarrollado el sistema desde la organización de los datos, el entrenamiento del algoritmo, su uso para la detección y la integración en ROS

4.1 Obtención de los datos

El primer paso será obtener una cantidad suficiente de datos como para poder entrenar el sistema de detección de emociones hasta que tenga una precisión aceptable. En nuestro caso los datos se componen de imágenes en las que las personas expresan una emoción de las que queremos detectar. Hay una gran cantidad de datasets que pueden ser útiles, pero se hace uso del Extended Cohn-Kanade Dataset (CK+) [33-34]. Cuenta con 593 secuencias de imágenes de 123 sujetos en las que en cada secuencia la primera fotografía el rostro tiene una emoción neutra y en la última una de las emociones con su expresión facial más acentuada. Dentro del directorio que contiene todo el dataset se encuentran subdirectorios que indican el número del sujeto, dentro del directorio de cada sujeto encontramos un número que corresponde a una emoción concreta y dentro de cada directorio de la emoción se encuentran las secuencias de fotografías. Esta estructura del tipo: CK+ Dataset/Sujeto/Emoción/Foto es fácilmente manejable y facilita a la hora de organizar los datos. Las imágenes de los datos son como las que se muestran en la figura 9, en estas se expresan las emociones de alegría, ira, tristeza, sorpresa y la emoción neutra.



[Figura 10] Ejemplos de datos de entrenamiento con las diferentes emociones

4.2 Organización de los datos

A la hora de organizar los datos, lo primero que se ha de hacer es crear un directorio donde se almacenan las imágenes de todos los sujetos por emoción, los subdirectorios que contendrá esta carpeta son tantos como emociones haya y llevarán el nombre de la emoción de la que contendrán las imágenes. Una vez hecho esto hacemos uso de Python y sus módulos para organizar el dataset como hemos dicho. Primero se importa el módulo glob, que permite manejarse entre los directorios que hay en el sistema y trabajar con ellos fácilmente ya que puede devolver una lista con todos los sub-

directorios. También importamos desde el módulo `shutil`, la función `copyfile()` que nos permite copiar archivos de un directorio a otro. Una vez que tenemos todo lo necesario podemos empezar a crear el algoritmo para la organización. El algoritmo consiste en ir por todos los sujetos viendo qué emociones tienen fotografías, dentro de cada emoción se copia la primera imagen, correspondiente a la ausencia de emociones o emoción neutra, dentro de la nueva carpeta en el subdirectorio neutral. De la misma forma se copia la última imagen, que corresponde a la emoción de la forma más acentuada, dentro de la nueva carpeta en el subdirectorio correspondiente a la emoción de acuerdo al código que se muestra en la figura 10.

```

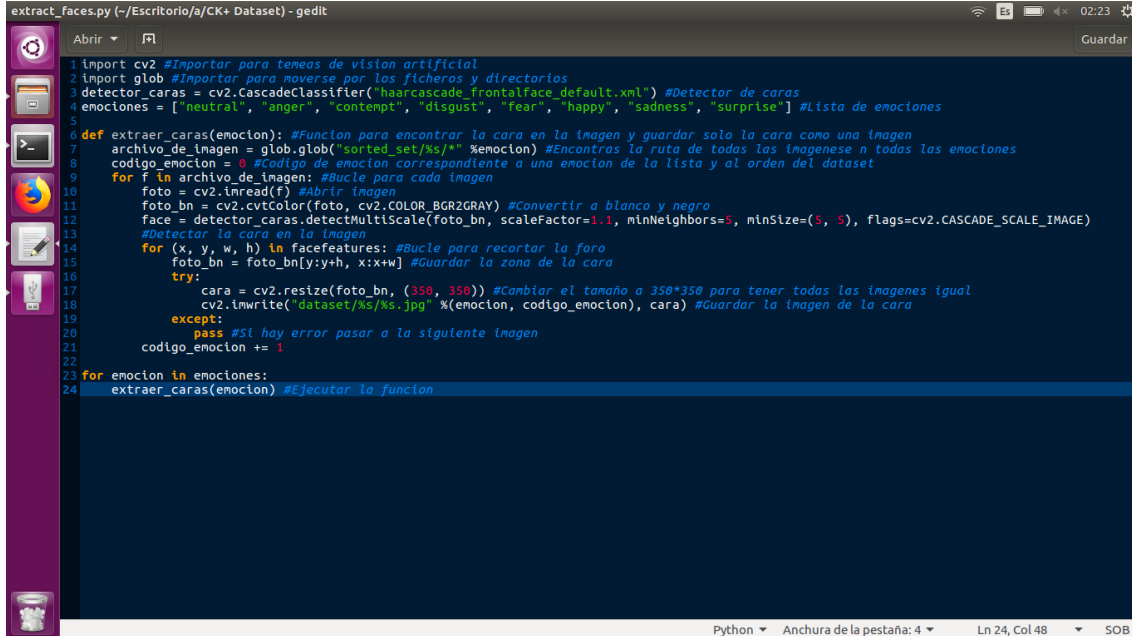
1 # coding=utf-8
2 import glob #Importar módulo glob para moverse por los directorios y archivos
3 from shutil import copyfile #Importar para copiar las imagenes en un directorio con las imagenes que queremos
4 emociones = ["neutral", "anger", "contempt", "disgust", "fear", "happy", "sadness", "surprise"] #Lista de emociones que contiene nuestro dataset
5 sujetos = glob.glob("source_emociones/*") #Obtener la lista de todos los sujetos que tiene el dataset
6 for x in sujetos: # Iniciar bucle para todos los sujetos
7     participante = "%s" %x[-4:] #Cojer las ultimas 3 caracteres que corresponden a la emociones
8     for emocion in glob.glob("%s/*" %x): #Bucle para cada emocion
9         for ficheros in glob.glob("%s/*" %emocion): #Obtener todas las imagenes de cada emocion
10             sesion = ficheros[20:-30] #Obtener el numero de imagen
11             file = open(ficheros, 'r') #Abrir la imagen
12             emociones = int(float(file.readline())) #Leer la emocion que tiene la imagen
13             imagen_emocion = sorted(glob.glob("source_images/%s/%s" %(participante, sesion)))[-1] #Obtener la ultima imagen que
14             #corresponde a la imagen en su maxima expresion
15             imagen_neutral = sorted(glob.glob("source_images/%s/%s" %(participante, sesion)))[0] #Obtener la primera imagen que
16             #corresponde a la imagen en la expresion neutra
17             dest_neut = "sorted_set/neutral/%s" %imagen_neutral[25:] #Definir el path que queremos para la emocion neutra
18             dest_emot = "sorted_set/%s/%s" %(emociones[emociones], imagen_emocion[25:]) #Definir el path que queremos para la emocion
19             copyfile(imagen_neutral, dest_neut) #Copiar la imagen neutra
20             copyfile(imagen_emocion, dest_emot) #Copiar la imagen de la emocion

```

[Figura 11] Organización de las imágenes por emoción.

Una vez que tenemos todas las imágenes ordenadas en directorios por emociones, vamos a extraer la porción de la imagen donde se encuentra la cara y guardaremos estas imágenes que contienen solo nuestra zona de interés en un nuevo directorio. Para ello haremos uso además de `glob`, de la librería `OpenCV`. El algoritmo para realizar esta tarea consiste en ir directorio por directorio dentro del directorio creado anteriormente abriendo/leyendo las imágenes y posteriormente aplicando el modelo pre-entrenado con el que cuenta `OpenCV` de detección de caras; las imágenes son pasadas a blanco y negro antes de aplicarse ya que esto mejora su rendimiento. Al aplicar el detector la cara es contenida en un rectángulo y su detección nos devuelve las coordenadas del vértice inferior izquierdo, así como la anchura y altura. Una vez que tenemos esto se puede

recortar la porción de la imagen que contiene el rostro, generar una nueva imagen de esta porción, redimensionarla para que todas las imágenes tengan el mismo tamaño y guardarla en nuestra nueva carpeta que contiene el dataset organizado tal como se muestra en el código de la figura 11.

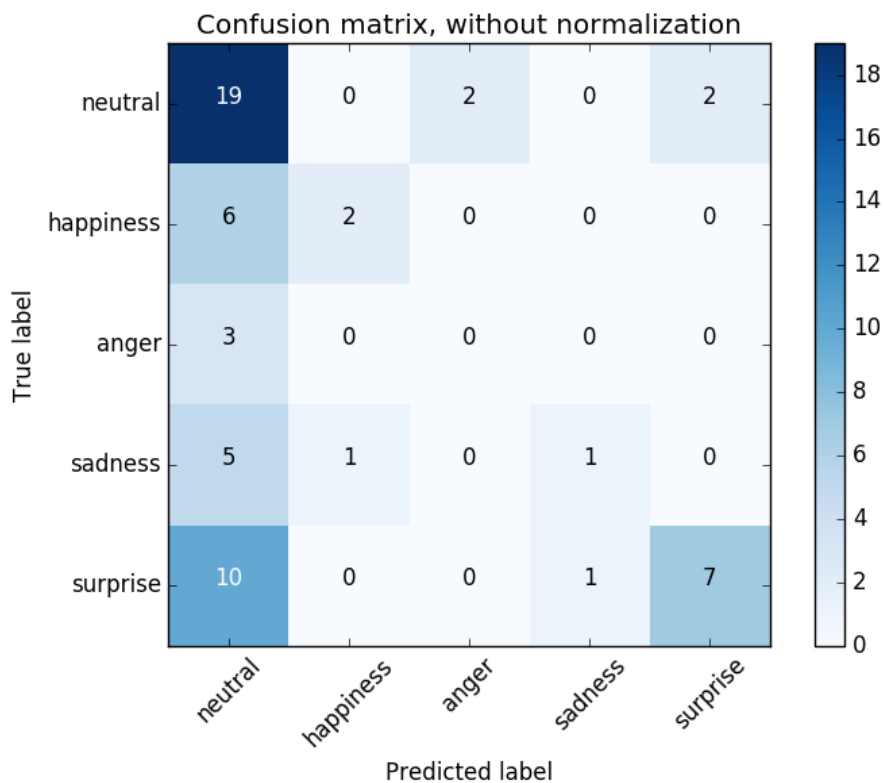
A screenshot of a code editor window titled 'extract_faces.py (-/Escritorio/a/CK+ Dataset) - gedit'. The editor shows a Python script for face extraction. The script imports cv2 and glob, initializes a Haar cascade classifier, and defines a list of emotions. It then defines a function 'extraer_caras' that iterates over a dataset, reads each image, converts it to grayscale, detects faces, crops the face, and saves it with a specific filename based on the emotion and a counter. The script also includes a loop to call this function for each emotion in the list. The status bar at the bottom indicates 'Python', 'Anchura de la pestaña: 4', 'Ln 24, Col 48', and 'SOB'.

[Figura 12] Extracción de las caras en las imágenes y reorganización en un nuevo directorio.

4.3 Extracción de los puntos de interés facial/Selección de atributos

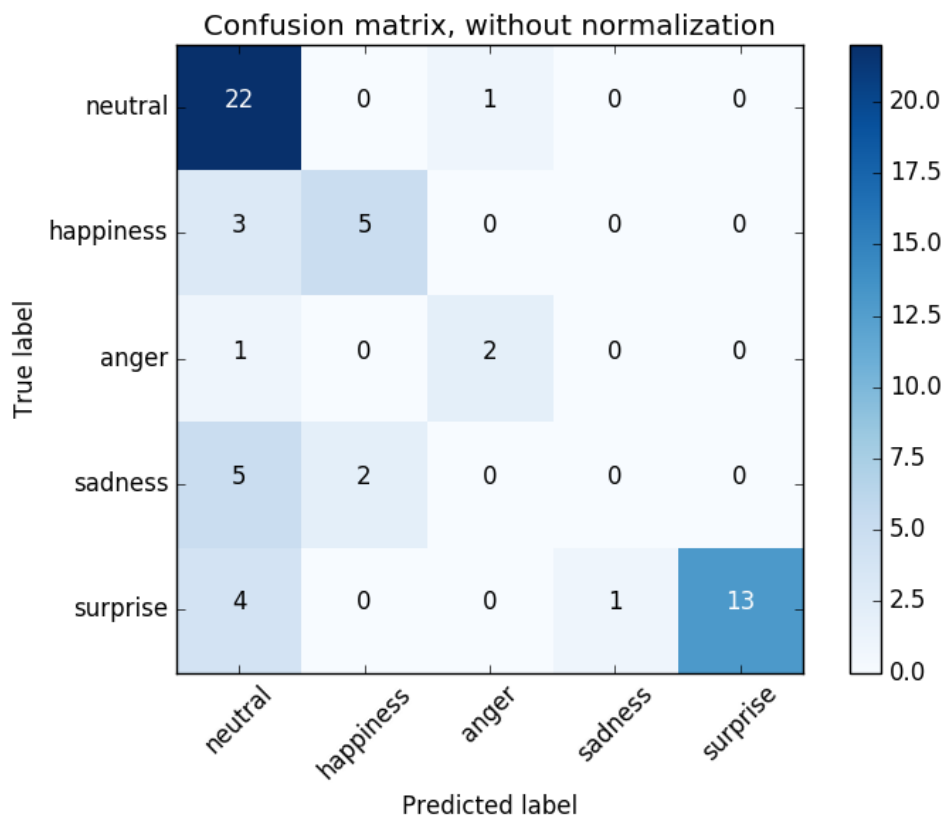
Una vez que se tiene el dataset organizado, hay que decidir que atributos se va a emplear para que nuestro algoritmo de inteligencia artificial aprenda, esta es una de las fases más críticas a la hora de enfrentarnos a un problema de aprendizaje automático, ya que una selección de atributos irrelevantes despista al algoritmo que se quiere entrenar y supone un problema muy crítico. Una de las posibilidades podría ser utilizar como atributos todos los píxeles que componen la imagen, pero esto no funciona muy bien con los algoritmos más básicos. Sin embargo, es una selección de atributos muy positiva cuando usamos algoritmos de Deep Learning, las redes neuronales convolucionales (CNNs) dan excelentes resultados cuando se usan de esta forma [35]. Estas redes son una variación del perceptrón multicapa donde las neuronas forman una red de forma similar a las de la parte asociada a la visión en el ser humano, por ello funcionan de forma excelente cuando se les da la información pura y sin pulir. Debido a que estas redes son bastante complejas y requieren de mucha potencia de procesamiento, trataremos de abordar el problema de forma distinta. En vez de utilizar la intensidad de

cada pixel como atributo, se localizarán los puntos de interés facial, ya que estos contienen la mayor parte de la información que nos es de utilidad y se entrena una máquina de soporte vectorial. La librería dlib tiene un detector de estos puntos que funciona de manera estable y da buenos resultados, además tiene un modelo pre-entrenado que es capaz de detectar hasta 68 de estos puntos en el rostro de una persona; pudiendo así tener información de los ojos, las cejas, la nariz, la boca y el contorno del rostro de una forma sencilla. Así una vez aplicando el detector de puntos se puede optar por introducir directamente las coordenadas de los puntos como atributos, el resultado es que usando una SVM con los parámetros predeterminados en scikit-learn da una precisión de 49% aproximadamente. Aunque es el primer modelo que entrenamos, la precisión esta lejos de ser la deseada, ya que la posibilidad de acertar con nuestro sistema es casi igual que la probabilidad de acertar en el juego aleatorio de cara o cruz. Además, analizando la matriz de confusión obtenida, en la figura 12 podemos ver que la predicción del algoritmo tiene una tendencia a predecir la expresión neutral, esto se debe a que es la clase que cuenta con más datos por lo que probablemente necesitemos además de intentar mejorar los atributos, realizar un ajuste de parámetros en nuestro algoritmo.



[Figura 13] Matriz de confusión con las coordenadas como atributos.

Una vez que se ha visto que posiblemente se pueda hacer una mejor selección de los atributos que se introducen al algoritmo de aprendizaje, se va a probar a usar como atributos las distancias euclídeas de todos los puntos de interés facial; es decir se probará a calcular la distancia entre todas las posibles combinaciones de pares de puntos e introduciremos eso en el algoritmo. El resultado es una muy notable mejora de la precisión, ahora nuestra maquina de soporte vectorial tiene una precisión del 71%, como se muestra en la matriz de confusión de la figura 13 que es una precisión más aceptable y podemos fiarnos ahora un poco más. Este resultado resalta la enorme importancia que tiene una buena selección de los atributos, aunque tengamos una enorme cantidad de datos a nuestra disposición, un buen algoritmo de aprendizaje y todos los recursos necesarios, es crítico y de vital importancia hacer una buena selección de atributos y esto es aplicable no solamente a este trabajo, sino a cualquier proyecto que haga uso del aprendizaje automático.



[Figura 14] Matriz de confusión con las distancias entre puntos.

4.4 Entrenamiento del algoritmo

Después de haber decidido como se entrenará a nuestro sistema de aprendizaje, toca implementar el código que realice esta tarea. Una vez más, primero se importan todos los módulos y librerías de los que se van a hacer uso, estos incluyen OpenCV, dlib, dentro de sklearn, se importa el modelo de maquina de soporte vectorial y joblib para la persistencia del modelo. Instanciamos un detector de caras, esta vez de la librería de dlib para usarlo junto a su detector de puntos de interés facial; también se instancia el detector de puntos de interés de la librería dlib. Cuando se use el detector facial trataremos de mejorar su rendimiento haciendo ciertas transformaciones en la imagen; se pasa la imagen blanco y negro y se mejora el contraste en la imagen con el uso ecualizador de histograma adaptable de contraste limitado (CLAHE). A continuación, se instancia de la librería scikit-learn el algoritmo deseado, en este caso una SVM. El esquema que va a seguir el código del entrenamiento es el siguiente, tal como se muestra en las figuras 14-16:

- Se dividen de forma aleatoria los datos en set de aprendizaje y set de prueba para evaluar el algoritmo, en nuestro caso un 80% son de aprendizaje y un 20% de prueba.

- Se extraen los puntos de interés facial y con esto se calculan las distancias entre puntos, que serán nuestros atributos y se guardan en un array.

- Se entrena el algoritmo usando el método fit() pasándole los atributos que hemos decidido y la clase a la que corresponden los atributos de cada instancia de los datos.

- Se calcula el rendimiento y precisión que tiene el algoritmo haciendo uso del set de pruebas y el método score().

- Se guarda el modelo en un archivo .joblib haciendo uso de esta librería.

```

train_tfg.py (-/Escritorio) - gedit
Abrir Guardar
1 # coding=utf-8
2 import cv2 #Importar OpenCv para tareas de visión
3 from sklearn.svm import SVC #Importar el modelo de máquina de soporte vectorial de scikit-learn
4 from joblib import dump, load #Importar joblib para la persistencia del modelo
5 import glob #Importar glob para manejarse entre los datos y los directorios
6 import random #Importar aleatorio para dividir el dataset
7 import math #Para operaciones matemáticas
8 import numpy as np
9 import dlib #dlib para obtener la cara y los puntos de interés facial
10
11
12 emociones = ["neutral", "happiness", "anger", "sadness", "surprise"] #Lista de las emociones que existen de acuerdo al estudio mas la neutral
13 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8)) #Mejora el contraste para la detección
14 modelo_cara = dlib.get_frontal_face_detector() #Nos permitira detectar las caras en las imagenes
15 modelo_puntos = dlib.shape_predictor('/home/jose/Escritorio/shape_predictor_68_face_landmarks.dat') #Nos permitira detectar los puntos
    faciales
16 algoritmo = SVC(gamma='auto') #Algoritmo que aprenderá a detectar
17 data = {} #Aqui se guardaran los atributos con los que se entrenará el algoritmo para cada imagen
18 ventana = dlib.image_window() #Ventana para visualizar las imagenes
19
20
21 def get_archivos_imagenes(emocion): #Dividir entre aprendizaje y testeo
22     archivos_imagenes = glob.glob("dataset/%s/*" %emocion)
23     random.shuffle(archivos_imagenes) #Aleatoriamente mezclar las imagenes
24     imagenes_entrenamiento = archivos_imagenes[:int(len(archivos_imagenes)*0.8)]
25     imagenes_testeo = archivos_imagenes[int(len(archivos_imagenes)*0.8):] #Se divide en una proporción de 80-20
26     return imagenes_entrenamiento, imagenes_testeo #Devolver las imagenes con la función
27
28
29 def get_puntos_interes_facial(foto): #Obtener puntos de interés facial
30     detecciones = modelo_cara(foto, 1)
31     for k,d in enumerate(detecciones):
32         shape = modelo_puntos(foto, d)
33         lista_de_x = []
34         lista_de_y = [] #Aqui se guardan las coordenadas de los puntos de interes facial
35         for i in range(1,68): #Con los numeros de aqui elegimos que partes de la cara queremos entrenar (1-68 significa que coje todos los
            puntos)
36             lista_de_x.append(float(shape.part(i).x))
37             lista_de_y.append(float(shape.part(i).y))
38         distancia_puntos = []

```

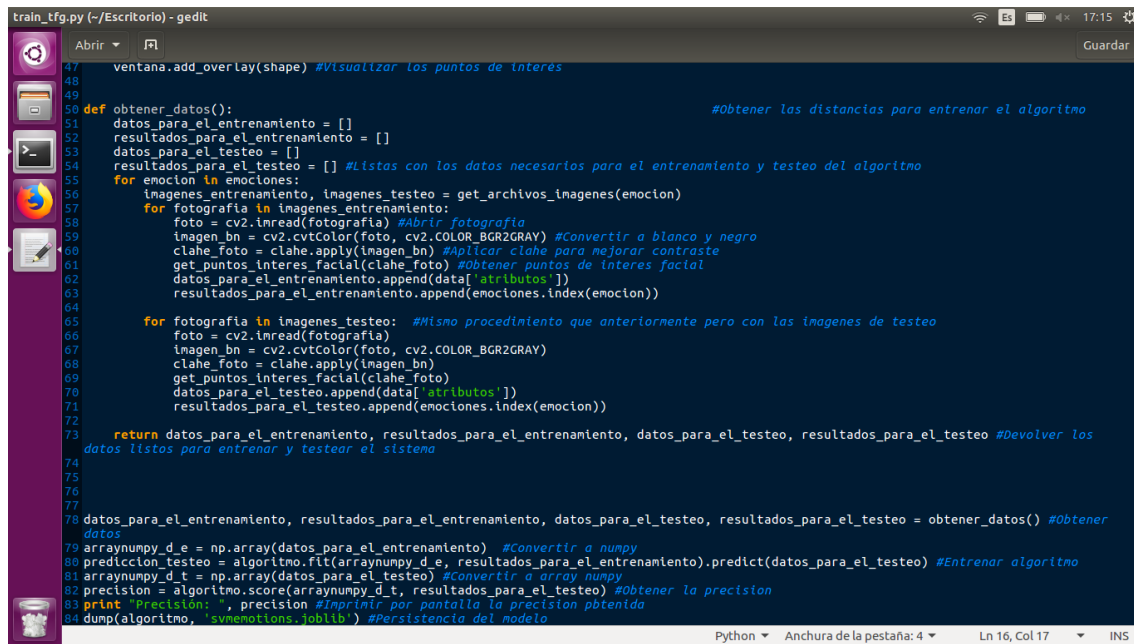
[Figura 15] Entrenamiento del algoritmo

```

train_tfg.py (-/Escritorio) - gedit
Abrir Guardar
36         lista_de_x.append(float(shape.part(i).x))
37         lista_de_y.append(float(shape.part(i).y))
38         distancia_puntos = []
39         for i in range(len(lista_de_x)):
40             for o in range(len(lista_de_x)):
41                 distance = math.sqrt((lista_de_x[i]-lista_de_x[o])**2+(lista_de_y[i]-lista_de_y[o])**2) #Calcular la distancia que hay entre
                    todas las combinaciones de puntos
42                 distancia_puntos.append(distance)
43         data['atributos'] = distancia_puntos
44         ventana.clear_overlay()
45         ventana.set_image(foto) #Visualizar la imagen
46         ventana.add_overlay(detecciones) #Visualizar la detección de la cara
47         ventana.add_overlay(shape) #Visualizar los puntos de interés
48
49
50 def obtener_datos(): #Obtener las distancias para entrenar el algoritmo
51     datos_para_el_entrenamiento = []
52     resultados_para_el_entrenamiento = []
53     datos_para_el_testeo = []
54     resultados_para_el_testeo = [] #Listas con los datos necesarios para el entrenamiento y testeo del algoritmo
55     for emocion in emociones:
56         imagenes_entrenamiento, imagenes_testeo = get_archivos_imagenes(emocion)
57         for fotografia in imagenes_entrenamiento:
58             foto = cv2.imread(fotografia) #Abrir fotografia
59             imagen_bn = cv2.cvtColor(foto, cv2.COLOR_BGR2GRAY) #Convertir a blanco y negro
60             clahe_foto = clahe.apply(imagen_bn) #Aplicar clahe para mejorar contraste
61             get_puntos_interes_facial(clahe_foto) #Obtener puntos de interes facial
62             datos_para_el_entrenamiento.append(data['atributos'])
63             resultados_para_el_entrenamiento.append(emociones.index(emocion))
64
65         for fotografia in imagenes_testeo: #Mismo procedimiento que anteriormente pero con las imagenes de testeo
66             foto = cv2.imread(fotografia)
67             imagen_bn = cv2.cvtColor(foto, cv2.COLOR_BGR2GRAY)
68             clahe_foto = clahe.apply(imagen_bn)
69             get_puntos_interes_facial(clahe_foto)
70             datos_para_el_testeo.append(data['atributos'])
71             resultados_para_el_testeo.append(emociones.index(emocion))
72
73     return datos_para_el_entrenamiento, resultados_para_el_entrenamiento, datos_para_el_testeo, resultados_para_el_testeo #Devolver los

```

[Figura 16] Entrenamiento del algoritmo



```
train_tfg.py (-/Escritorio) - gedit
47 ventana.add_overlay(shape) #Visualizar los puntos de interes
48
49
50 def obtener_datos():
51     datos_para_el_entrenamiento = []
52     resultados_para_el_entrenamiento = []
53     datos_para_el_testeo = []
54     resultados_para_el_testeo = [] #Listas con los datos necesarios para el entrenamiento y testeo del algoritmo
55     for emocion in emociones:
56         imagenes_entrenamiento, imagenes_testeo = get_archivos_imagenes(emocion)
57         for fotografia in imagenes_entrenamiento:
58             foto = cv2.imread(fotografia) #Abrir fotografia
59             imagen_bn = cv2.cvtColor(foto, cv2.COLOR_BGR2GRAY) #Convertir a blanco y negro
60             clahe_foto = clahe.apply(imagen_bn) #Aplicar clahe para mejorar contraste
61             get_puntos_interes_facial(clahe_foto) #Obtener puntos de interes facial
62             datos_para_el_entrenamiento.append(data['atributos'])
63             resultados_para_el_entrenamiento.append(emociones.index(emocion))
64
65         for fotografia in imagenes_testeo: #Mismo procedimiento que anteriormente pero con las imagenes de testeo
66             foto = cv2.imread(fotografia)
67             imagen_bn = cv2.cvtColor(foto, cv2.COLOR_BGR2GRAY)
68             clahe_foto = clahe.apply(imagen_bn)
69             get_puntos_interes_facial(clahe_foto)
70             datos_para_el_testeo.append(data['atributos'])
71             resultados_para_el_testeo.append(emociones.index(emocion))
72
73     return datos_para_el_entrenamiento, resultados_para_el_entrenamiento, datos_para_el_testeo, resultados_para_el_testeo #Devolver los
74     datos listos para entrenar y testear el sistema
75
76
77
78 datos_para_el_entrenamiento, resultados_para_el_entrenamiento, datos_para_el_testeo, resultados_para_el_testeo = obtener_datos() #Obtener
79 datos
80 arraynumpy_d_e = np.array(datos_para_el_entrenamiento) #Convertir a numpy
81 prediccion_testeo = algoritmo.fit(arraynumpy_d_e, resultados_para_el_entrenamiento).predict(datos_para_el_testeo) #Entrenar algoritmo
82 arraynumpy_d_t = np.array(datos_para_el_testeo) #Convertir a array numpy
83 precision = algoritmo.score(arraynumpy_d_t, resultados_para_el_testeo) #Obtener la precision
84 print "precision: ", precision #Imprimir por pantalla la precision obtenida
85 dump(algoritmo, 'svmotions.joblib') #Persistencia del modelo
```

[Figura 17] Entrenamiento del algoritmo

Como se ha visto antes la SVM da un rendimiento del 71% con los parámetros por defecto, pero se puede mejorar sustancialmente esta precisión haciendo un ajuste de parámetros; además viendo la tendencia que tiene nuestro algoritmo de clasificar como neutro podemos intentar compensar esto.

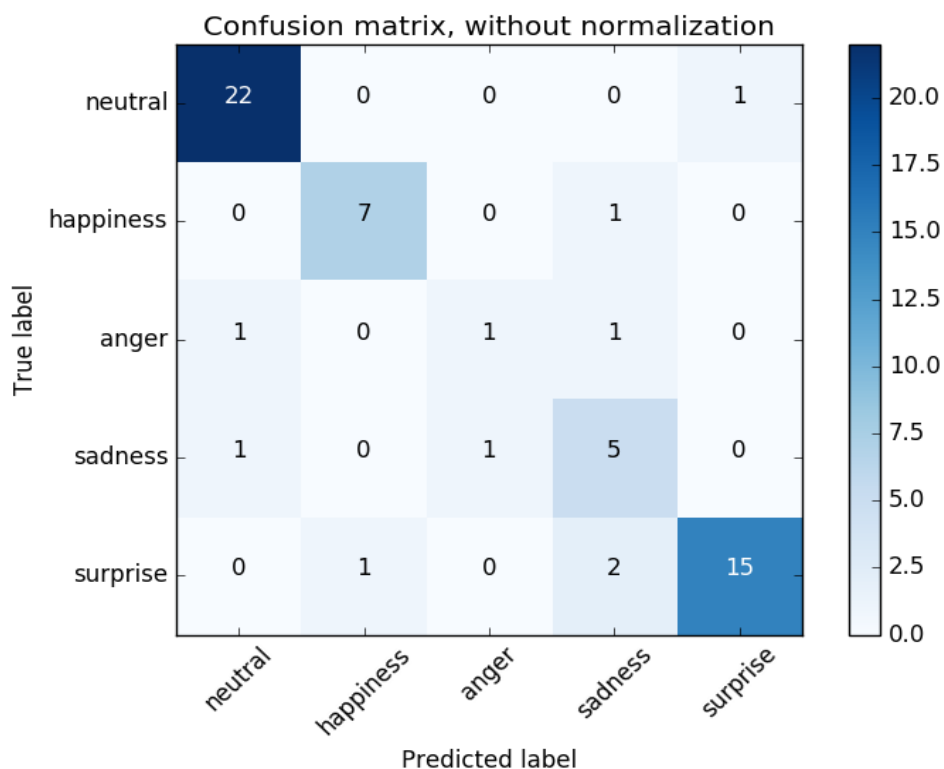
Para tratar de mejorar el rendimiento de nuestro algoritmo se debe entender más en profundidad cómo funcionan los parámetros de una máquina de soporte vectorial. Los principales parámetros son:

- C: cuando se trata de ajustar el hiperplano, hay que hacer un balance entre el numero de aciertos que la maquina hará y la distancia mínima a la que separa las instancias de distintas clases en él entrenamiento. El parámetro C controla este balance, cuando se aumenta el hiperplano conseguirá más aciertos, aunque el margen se vuelva menor y viceversa. Si la C es demasiado pequeña se obtendrá un gran error en el entrenamiento; por el contrario, si la C es demasiado grande, se fallará en generalizar con nuestro algoritmo.

- kernel: especifica qué tipo de hiperplano se usa para hacer la separación de los datos, en este caso se va a usar siempre un hiperplano linear, aunque los existen basados en otras disposiciones matemáticas como polinomios.

-gamma: Solo tiene uso en SVM con un kernel no linear Trata de ajustar la importancia de los puntos con los que entrenamos la máquina y su influencia, cuando tenemos un valor alto de gamma tendremos mayor sesgo y una varianza menor y viceversa.

Además de todo esto se puede dar distintos costes de error dependiendo de la clase que se este entrenando y así tratar de compensar el desequilibrio en el numero de instancias en las clases.



[Figura 18] Matriz de confusión con el ajuste de parámetros.

Configurando los parámetros de la SVM con una $C=20$, kernel linear y dando los pesos que se ven en la Tabla 1, logramos una precisión de casi el 85% y una matriz de confusión como en la figura 17.

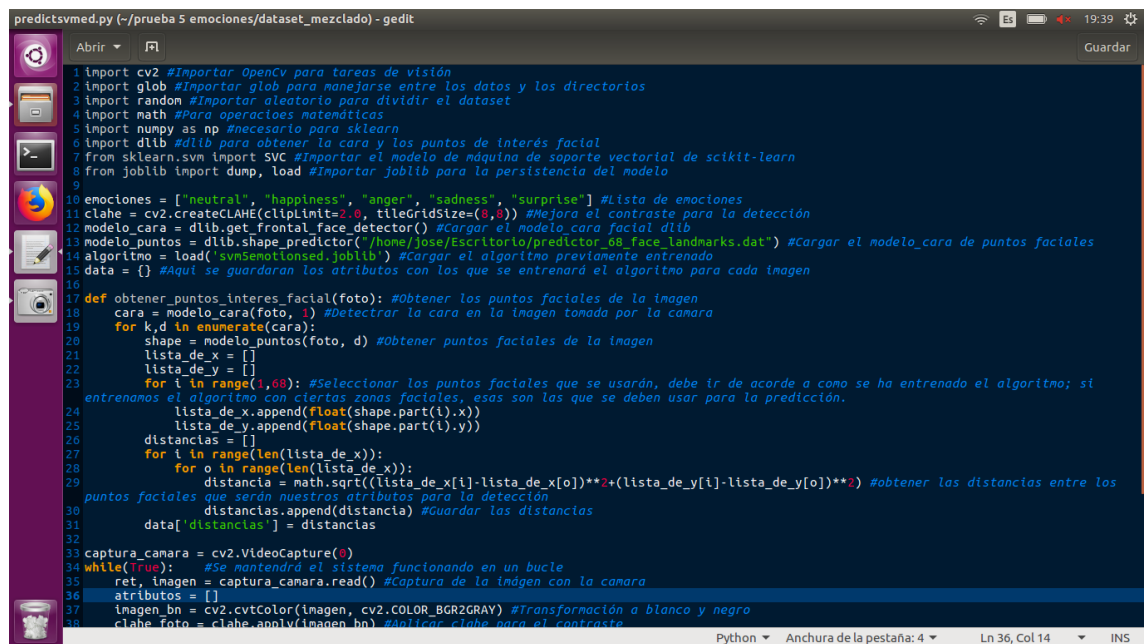
Neutral	Felicidad	Ira	Tristeza	Sorpresa
2	0.25	4	3	2

[Tabla 1] Pesos de cada clase en el aprendizaje

4.5 Uso del algoritmo para clasificar las emociones

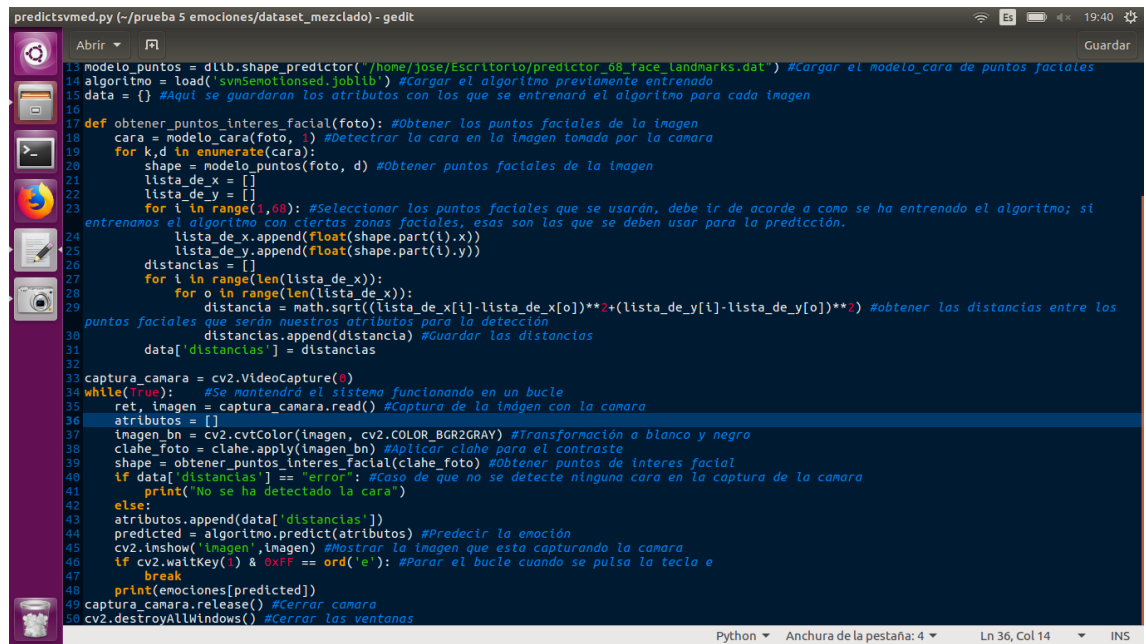
Lo siguiente que se debe hacer es darle uso al modelo entrenado y que sea capaz de detectar las emociones de quien es capturado por las imágenes de la cámara. Para ello se va a hacer un código muy parecido al anterior, usado en el entrenamiento, pero en el orden inverso. El código de detección de las emociones funciona de la siguiente forma, como se muestra en las figuras 18-19:

- Primero se importan las librerías necesarias.
- Después se carga el modelo guardado en el archivo correspondiente y se instancia un clasificador con ese modelo.
- Se capturan en tiempo real las imágenes que está viendo la cámara.
- Se aplica el detector facial.
- Se aplica el detector de puntos de interés facial.
- Se calcula las distancias entre puntos y las almacenamos en un array.
- Se aplica el clasificador a los datos que tenemos guardados en el Array y nos muestra la emoción que tiene la persona a la que se le aplica el sistema, en cada frame.



```
predictsvmed.py (~/prueba 5 emociones/dataset_mezclado) - gedit
Abrir Guardar
1 import cv2 #Importar OpenCV para tareas de visión
2 import glob #Importar glob para manejarse entre los datos y los directorios
3 import random #Importar aleatorio para dividir el dataset
4 import math #Para operaciones matemáticas
5 import numpy as np #necesario para sklearn
6 import dlib #dlib para obtener la cara y los puntos de interés facial
7 from sklearn.svm import SVC #Importar el modelo de máquina de soporte vectorial de scikit-learn
8 from joblib import dump, load #Importar joblib para la persistencia del modelo
9
10 emociones = ["neutral", "happiness", "anger", "sadness", "surprise"] #Lista de emociones
11 clahe = cv2.createCLAHE(cliclplimit=2.0, tileGridSize=(8,8)) #Mejora el contraste para la detección
12 modelo_cara = dlib.get_frontal_face_detector() #Cargar el modelo_cara facial dlib
13 modelo_puntos = dlib.shape_predictor('/home/jose/Escritorio/predictor_68_face_landmarks.dat') #Cargar el modelo_cara de puntos faciales
14 algoritmo = load('svmEmotionsed.joblib') #Cargar el algoritmo previamente entrenado
15 data = {} #Aquí se guardaran los atributos con los que se entrenará el algoritmo para cada imagen
16
17 def obtener_puntos_interes_facial(foto): #Obtener los puntos faciales de la imagen
18     cara = modelo_cara(foto, 1) #Detectar la cara en la imagen tomada por la cámara
19     for k,d in enumerate(cara):
20         shape = modelo_puntos(foto, d) #Obtener puntos faciales de la imagen
21         lista_de_x = []
22         lista_de_y = []
23         for i in range(1,68): #Seleccionar los puntos faciales que se usarán, debe ir de acorde a como se ha entrenado el algoritmo; si
24             #entrenamos el algoritmo con ciertas zonas faciales, esas son las que se deben usar para la predicción.
25             lista_de_x.append(float(shape.part(i).x))
26             lista_de_y.append(float(shape.part(i).y))
27         distancias = []
28         for i in range(len(lista_de_x)):
29             for o in range(len(lista_de_x)):
30                 distancia = math.sqrt((lista_de_x[i]-lista_de_x[o])**2+(lista_de_y[i]-lista_de_y[o])**2) #obtener las distancias entre los
31                 #puntos faciales que serán nuestros atributos para la detección
32                 distancias.append(distancia) #Guardar las distancias
33             data['distancias'] = distancias
34
35 captura_camara = cv2.VideoCapture(0)
36 while(True): #Se mantendrá el sistema funcionando en un bucle
37     ret, imagen = captura_camara.read() #Captura de la imagen con la cámara
38     atributos = []
39     imagen_bn = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY) #Transformación a blanca y negro
40     clahe_foto = clahe.apply(imagen_bn) #Aplicar clahe para el contraste
41     #Aquí se aplicaría el algoritmo de predicción de emociones
```

[Figura 19] Código para hacer la predicción.



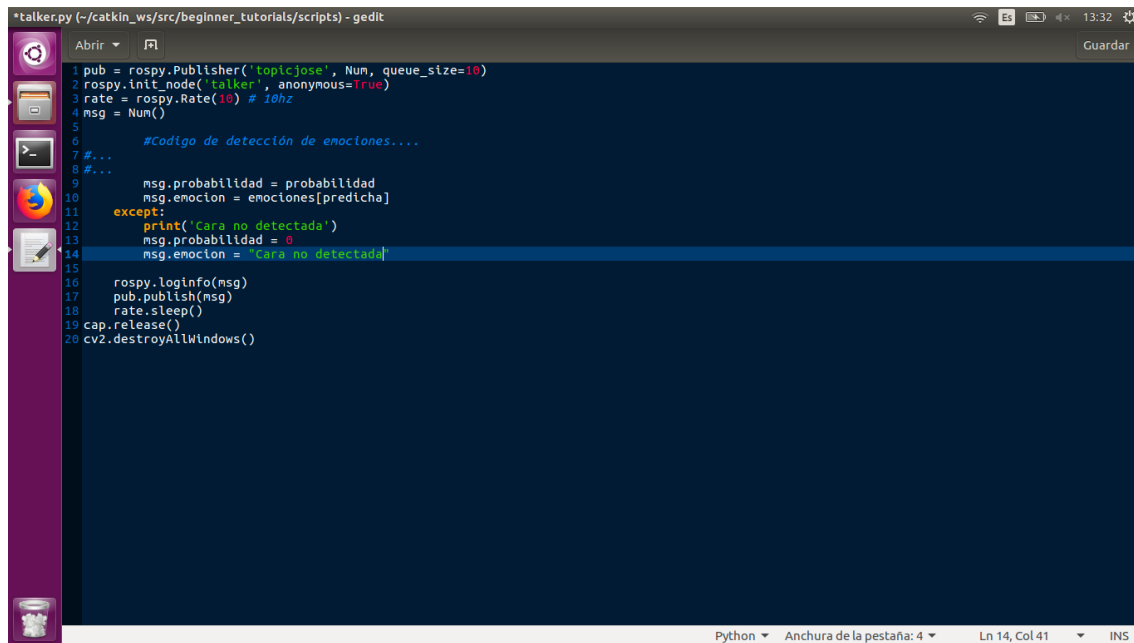
```
13 modelo_puntos = dlib.shape_predictor('/home/jose/escritorio/predictor_68_face_landmarks.dat') #Cargar el modelo cara de puntos faciales
14 algoritmo = load('svm5emotionsed.joblib') #Cargar el algoritmo previamente entrenado
15 data = {} #Aqui se guardaran los atributos con los que se entrenará el algoritmo para cada imagen
16
17 def obtener_puntos_interes_facial(foto): #Obtener los puntos faciales de la imagen
18     cara = modelo_cara(foto, 1) #Detectar la cara en la imagen tomada por la camara
19     for k,d in enumerate(cara):
20         shape = modelo_puntos(foto, d) #Obtener puntos faciales de la imagen
21         lista_de_x = []
22         lista_de_y = []
23         for i in range(1,68): #Seleccionar los puntos faciales que se usarán, debe ir de acorde a como se ha entrenado el algoritmo; si
24             entrenamos el algoritmo con ciertas zonas faciales, esas son las que se deben usar para la predicción.
25             lista_de_x.append(float(shape.part(i).x))
26             lista_de_y.append(float(shape.part(i).y))
27         distancias = []
28         for i in range(len(lista_de_x)):
29             for o in range(len(lista_de_x)):
30                 distancia = math.sqrt((lista_de_x[i]-lista_de_x[o])**2+(lista_de_y[i]-lista_de_y[o])**2) #obtener las distancias entre los
31                 puntos faciales que serán nuestros atributos para la detección
32                 distancias.append(distancia) #guardar las distancias
33             data['distancias'] = distancias
34
35 captura_camara = cv2.VideoCapture(0)
36 while(True): #Se mantendrá el sistema funcionando en un bucle
37     ret, imagen = captura_camara.read() #Captura de la imagen con la camara
38     atributos = []
39     imagen_bn = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY) #Transformación a blanco y negro
40     clahe_foto = clahe.apply(imagen_bn) #Aplicar clahe para el contraste
41     shape = obtener_puntos_interes_facial(clahe_foto) #Obtener puntos de interes facial
42     if data['distancias'] == "error": #caso de que no se detecte ninguna cara en la captura de la camara
43         print("No se ha detectado la cara")
44     else:
45         atributos.append(data['distancias'])
46         predicted = algoritmo.predict(atributos) #Predecir la emoción
47         cv2.imshow('imagen',imagen) #Mostrar la imagen que esta capturando la camara
48         if cv2.waitKey(1) & 0xFF == ord('e'): #Parar el bucle cuando se pulsa la tecla e
49             break
50     print(emociones[predicted])
51     captura_camara.release() #Cerrar camara
52 cv2.destroyAllWindows() #Cerrar las ventanas
```

[Figura 20] Código para hacer la predicción.

4.6 Integración en ROS

Por último, se debe integrar el sistema en ROS para poder ser aplicado en robótica. Para ellos se va a crear un paquete de ROS, desde la consola de comandos vamos al directorio src dentro del espacio de trabajo catkin (~/.catkin_ws/src) y se usa el comando catkin_create_pkg seguido del nombre y luego de las dependencias de roscpp, rospy y std_msgs. Esto debería crear una carpeta con un archivo xml y otro nombrado CMakeLists.txt cuyo contenido debe estar condicionado por la información de dependencias introducidas anteriormente. Después se crea un subdirectorio donde guardaremos los scripts de Python que se han escrito antes con unas modificaciones. Además, se debe modificar el archivo CMake para que las dependencias queden correctamente definidas y crear un mensaje para la comunicación entre los nodos. Para definir el mensaje se crea un archivo .msg que contiene los tipos de variables que serán pasados entre nodos, en este caso se va a pasar el nombre de la emoción detectada y su probabilidad, en variables de tipo string y float64 respectivamente.

Ahora se modifica el script de detección de emociones para que sea un nodo que envía información del tipo que acabamos de crear en un ROS topic al que se conectara posteriormente otro nodo de acuerdo al código de las figuras 20-21.

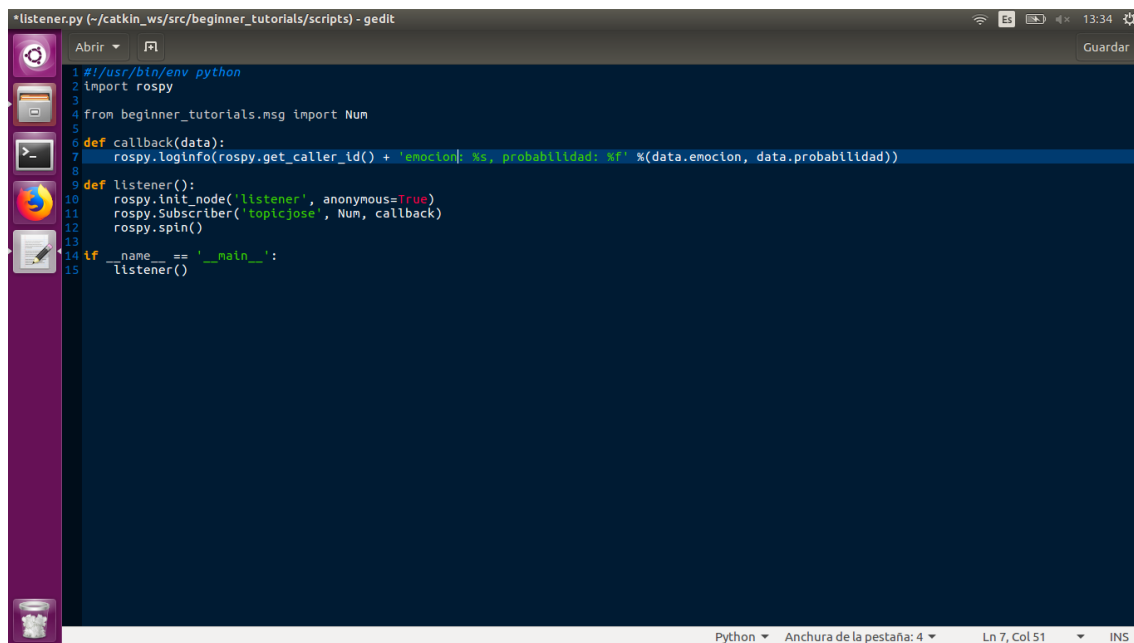


```
*talker.py (-/catkin_ws/src/beginner_tutorials/scripts) - gedit
Abrir Guardar
1 pub = rospy.Publisher('topicjose', Num, queue_size=10)
2 rospy.init_node('talker', anonymous=True)
3 rate = rospy.Rate(10) # 10hz
4 msg = Num()
5
6 #Codigo de detección de emociones...
7 #...
8 #...
9 msg.probabilidad = probabilidad
10 msg.emocion = emociones[predicha]
11 except:
12     print('Cara no detectada')
13     msg.probabilidad = 0
14     msg.emocion = 'Cara no detectada'
15
16 rospy.loginfo(msg)
17 pub.publish(msg)
18 rate.sleep()
19 cap.release()
20 cv2.destroyAllWindows()

Python Anchura de la pestaña: 4 Ln 14, Col 41 INS
```

[Figura 21] Inicialización del nodo y publicación de la información.

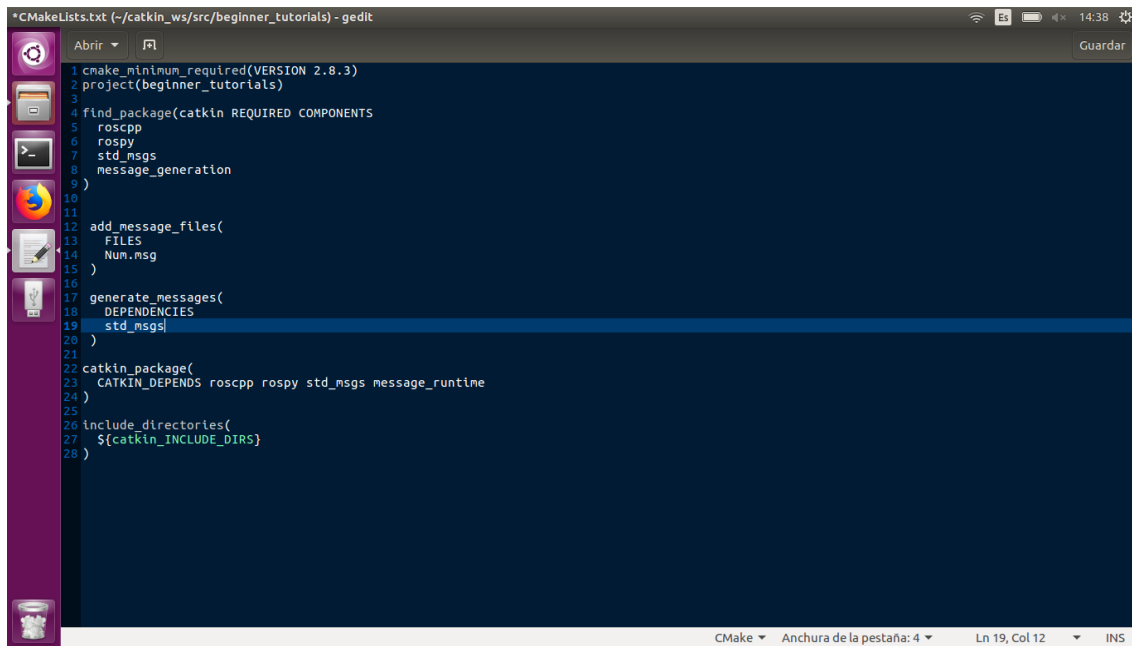
También se crea otro script que será el receptor de la información transmitida por el nodo detector de emociones. Este se conectará al topic donde el nodo detector de emociones publica la información e informará por pantalla de la emoción detectada y la confianza con la que el sistema la está detectando como en el código de la figura 22.



```
*listener.py (-/catkin_ws/src/beginner_tutorials/scripts) - gedit
Abrir Guardar
1 #!/usr/bin/env python
2 import rospy
3
4 from beginner_tutorials.msg import Num
5
6 def callback(data):
7     rospy.loginfo(rospy.get_caller_id() + 'emocion: %s, probabilidad: %f' % (data.emocion, data.probabilidad))
8
9 def listener():
10     rospy.init_node('listener', anonymous=True)
11     rospy.Subscriber('topicjose', Num, callback)
12     rospy.spin()
13
14 if __name__ == '__main__':
15     listener()

Python Anchura de la pestaña: 4 Ln 7, Col 51 INS
```

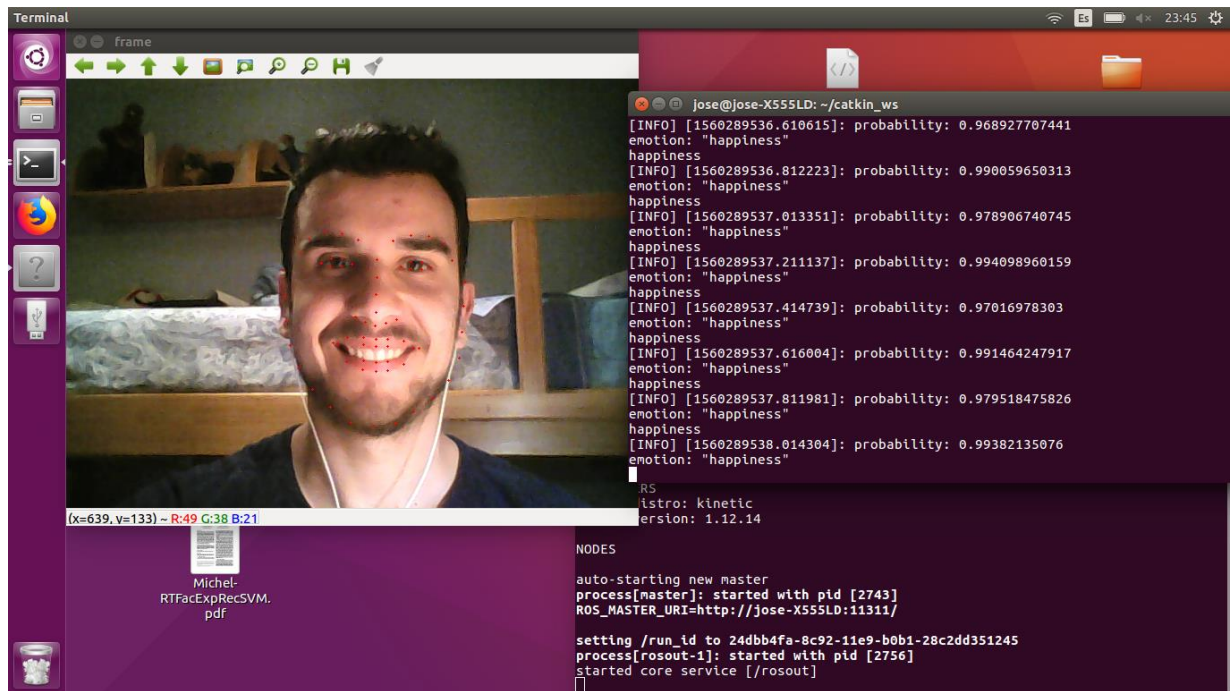
[Figura 22] Nodo receptor de la información.



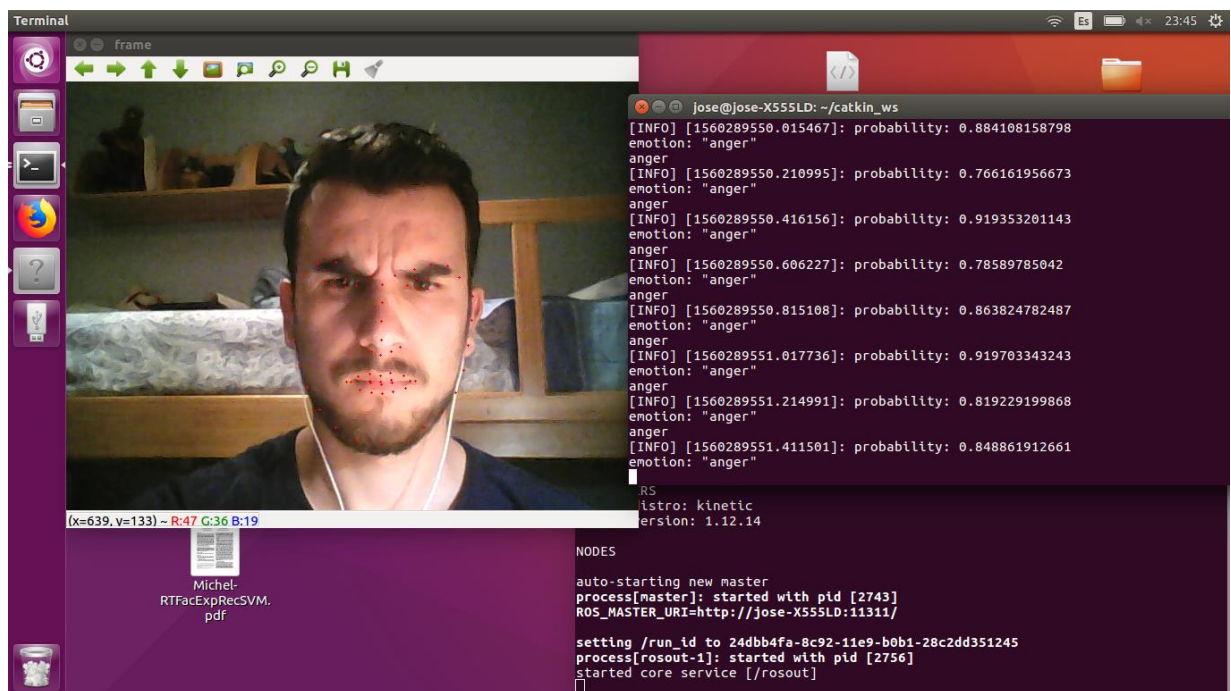
```
*CMakeLists.txt (~/.catkin_ws/src/beginner_tutorials) - gedit
Abrir Guardar
1 cmake_minimum_required(VERSION 2.8.3)
2 project(beginner_tutorials)
3
4 find_package(catkin REQUIRED COMPONENTS
5   roscpp
6   rospy
7   std_msgs
8   message_generation
9 )
10
11
12 add_message_files(
13   FILES
14   Num.msg
15 )
16
17 generate_messages(
18   DEPENDENCIES
19   std_msgs
20 )
21
22 catkin_package(
23   CATKIN_DEPENDS roscpp rospy std_msgs message_runtime
24 )
25
26 include_directories(
27   ${catkin_INCLUDE_DIRS}
28 )
```

[Figura 23] Archivo CMakeLists.txt

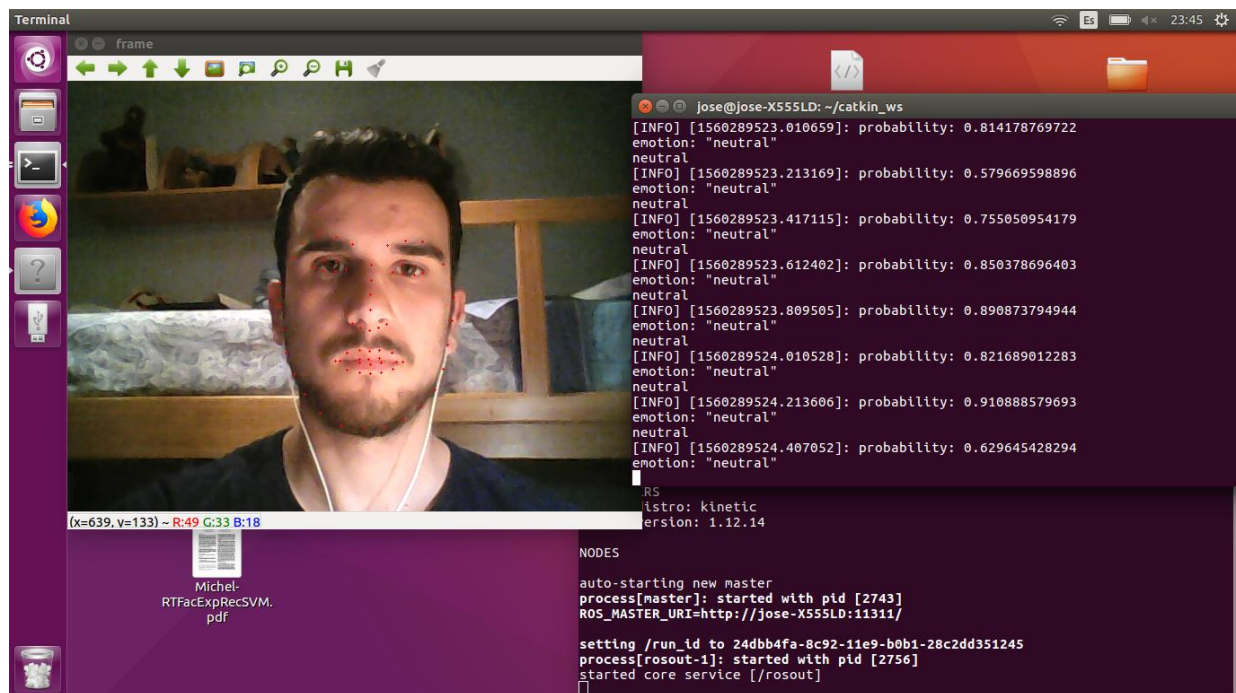
Una vez que se termina de modificar los scripts y el archivo CMake mostrado en la figura 23 se deben hacer los archivos .py ejecutables con el uso del comando `chmod`. Lo siguiente será irnos al `catkin_workspace` donde se usa el comando `catkin_make` para compilar y crear el paquete. Con esto hemos acabado de crear nuestro paquete de ROS donde se integra el sistema de detección de emociones que entrenamos anteriormente. Se ha de tener siempre en cuenta donde se encuentra el archivo generado por `joblib` que contiene el algoritmo entrenado. A continuación, se muestra el sistema detectando las distintas emociones en las figuras 24-28:



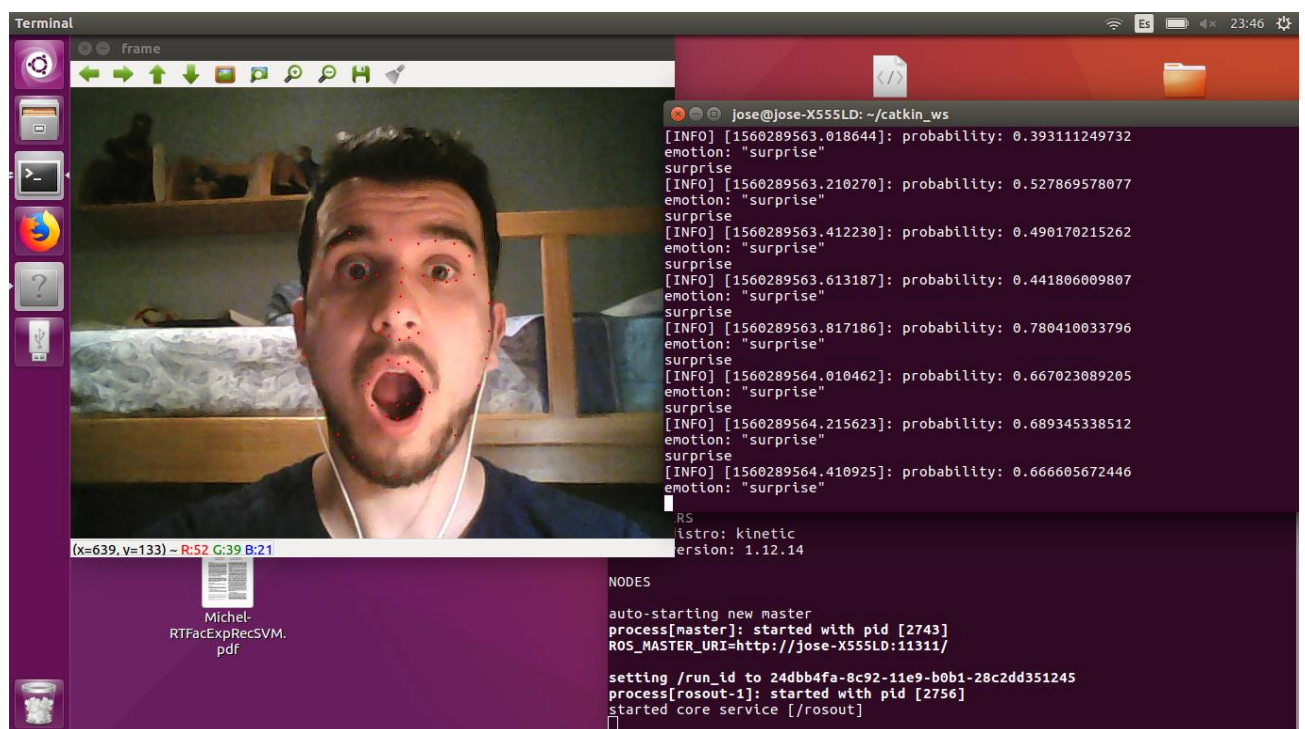
[Figura 24] El sistema detectando la emoción de alegría.



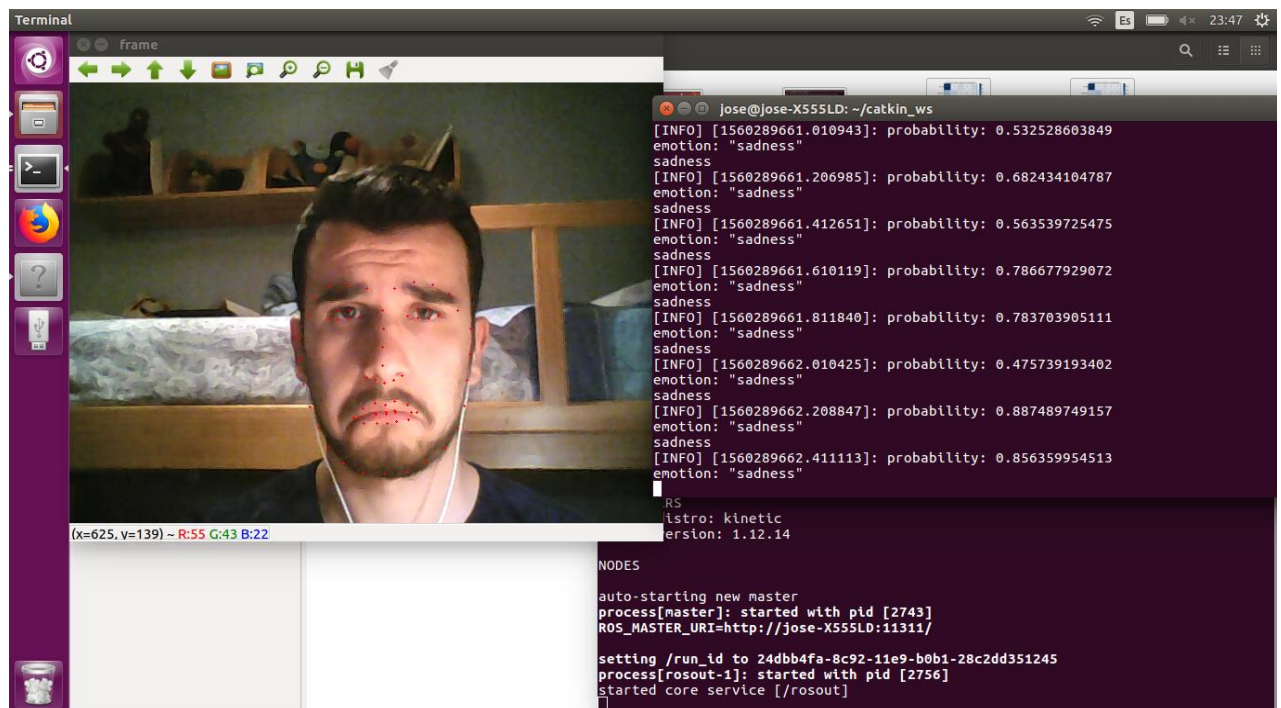
[Figura 25] El sistema detectando la emoción de ira.



[Figura 26] El sistema detectando la emoción neutral.



[Figura 27] El sistema detectando la emoción de sorpresa.



[Figura 28] El sistema detectando la emoción de tristeza.

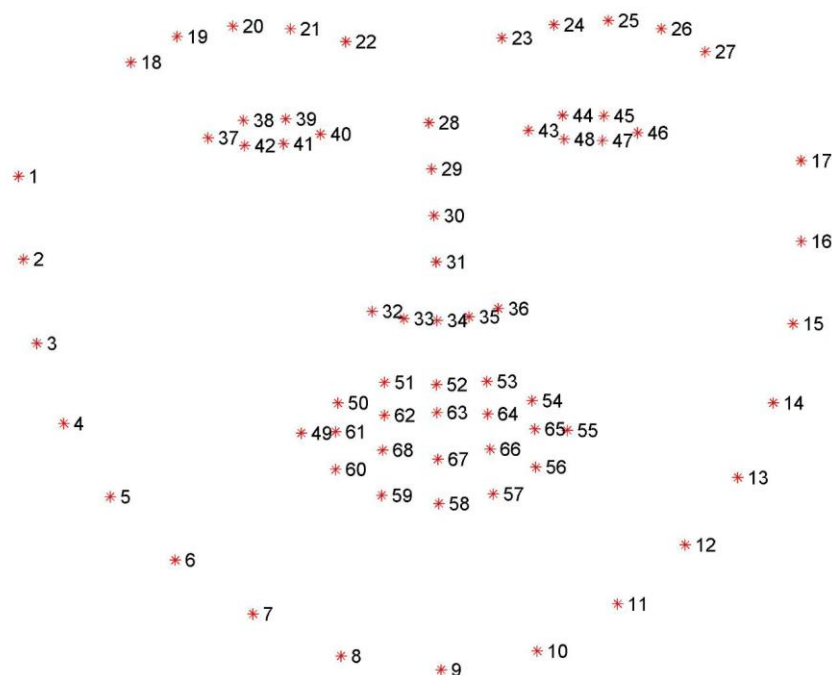
Capítulo 5: Pruebas con distintas partes faciales y algoritmos

Una vez ya desarrollado el sistema en su totalidad, se va a realizar una comparativa entre el mismo sistema de detección primero usando distintas partes de la cara para ver cuanta información aporta cada una y luego aplicando distintos algoritmos de aprendizaje con el fin de ver cual es más apropiado para nuestro objetivo.

5.1 Distintas partes faciales / influencia de las partes de la cara

A continuación, se va a analizar la influencia de cada una de las partes de la cara y cuanta información aporta cada una. Para ello se entrena la máquina de soporte vectorial, con los parámetros que previamente fueron usados y con los que se obtuvo un 84% utilizando la información de todos los puntos de la cara; pero esta vez se hacen dos pruebas por cada parte de la cara de la que queremos ver la influencia, una con todos los puntos faciales de la cara menos los de esta parte y otra únicamente con la parte facial de la que queremos estudiar su influencia. Posteriormente se analiza la precisión y las matrices de confusión obtenidas. El detector de puntos faciales devuelve 68 puntos distribuidos de la siguiente forma, mostrado en el esquema de la figura 29:

- 1 al 17, pertenecientes al contorno de la cara.
- 18 al 27, pertenecientes a la ceja.
- 28 al 36, pertenecientes a la nariz.
- 37 al 48, pertenecientes a los ojos.
- 49 al 68, pertenecientes a la boca.



[Figura 29] Puntos de interés facial

<https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>,

12/06/19

Los resultados que obtenemos se muestran en la tabla 2.

	Solo con la parte	Sin la parte
Contorno	57%	81%
Cejas	48%	80%
Boca	73%	64%
Nariz	40%	83%
Ojos	64%	78%

[Tabla 2] Resultados del entrenamiento con distintas zonas faciales

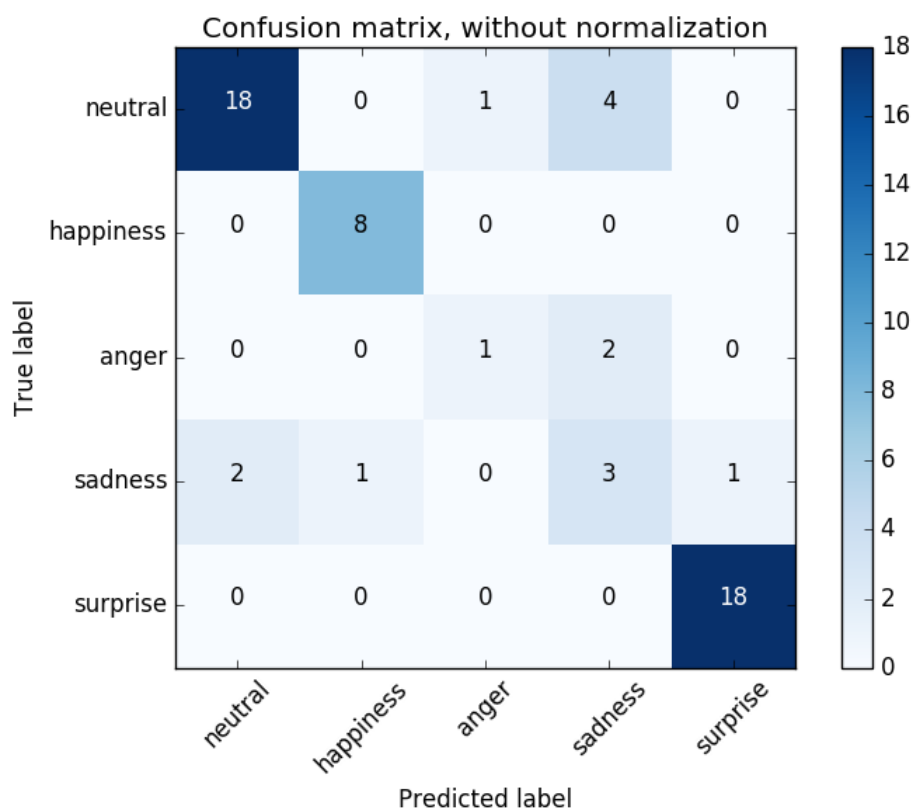
Observando las matrices de confusión se puede entender mejor como afectan las partes de la cara a la detección de cada una de las emociones. En la Figura 30 se observa que sin el contorno de la cara apenas se pierde precisión y el único efecto negativo es que se tiende a predecir la emoción de tristeza, mientras que en la Figura 31 se observa cómo solamente las emociones de alegría y de sorpresa se consiguen detectar sin una confusión excesiva. En la figura 32 se puede observar que sí no consideramos la información aportada por las cejas el sistema es incapaz de detectar la emoción de ira;

mientras que solo con las cejas se observa en la figura 33 que no se puede detectar de forma fiable ninguna de las emociones. Al analizar la boca, se observa en la figura 34 que si no usamos la información que nos proporciona perdemos precisión en todas las emociones y el sistema deja de ser fiable por completo, en cambio, si solo se atiende a esta zona para la extracción de información se observa en la figura 35 que se logra un rendimiento suficientemente fiable y existe la capacidad de detectar tres de las cinco emociones de forma fiable, la neutra, alegría y la sorpresa. Al analizar la importancia de la nariz vemos en la figura 36 que el sistema apenas pierde precisión sin esta emoción y en la figura 37 se observa que únicamente con la nariz no se puede detectar ninguna emoción y el sistema no ha aprendido prácticamente nada con esta información. Finalmente, en la figura 38 vemos que si no se cuenta con la información proporcionada por los ojos el sistema pierde un 6% de precisión y se pierde fiabilidad al detectar todas las emociones y al únicamente usar los ojos como información se observa en la figura 38 que somos capaces de detectar las emociones de sorpresa y alegría.

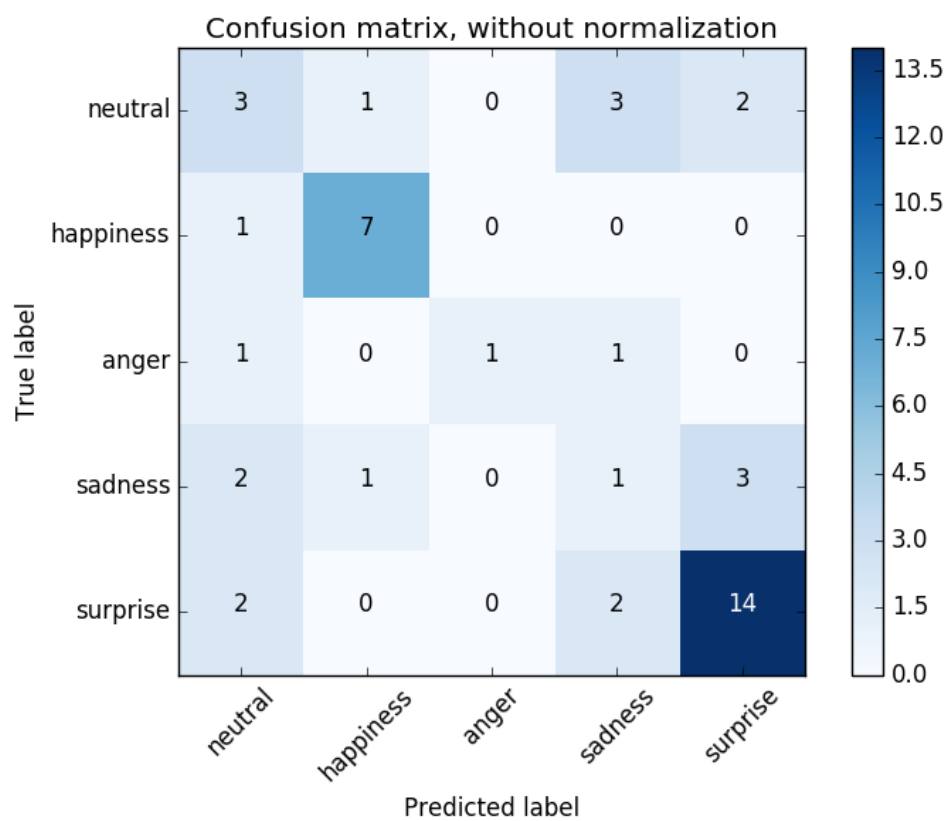
Se puede observar a primera vista que las emociones más fáciles de predecir son la de alegría y sorpresa, ya que son emociones muy marcadas y diferenciadas de las otras. En el caso de sorpresa es tanto los ojos como la boca serían suficientes para predecir esta emoción. Contrariamente, las emociones de tristeza e ira resultan más difíciles de predecir.

La facilidad con la que una emoción será detectada depende directamente de la forma que adopten las diferentes partes de la cara y la distancia que haya entre estas; esto es debido al tipo de atributos que hemos elegido, las distancias entre los puntos de interés facial. Como ejemplo, la expresión facial asociada a la emoción de sorpresa tiene la característica de los ojos abiertos, las cejas subidas y la boca también abierta; debido a que ninguna otra tiene unas características similares a esta y a lo pronunciadas que aparecen en esta emoción, la emoción de sorpresa resulta fácilmente distinguible para el algoritmo. Por el contrario, la emoción de ira tiene como característica en su expresión facial las cejas juntas y hacia abajo; al ser menos pronunciada y menos distinguible, dificulta la tarea de su distinción. Debido a esto, ciertas partes de la cara aportarán información útil únicamente de ciertas emociones mientras que con otras no ayudarán, o lo harán en menor medida, a su clasificación.

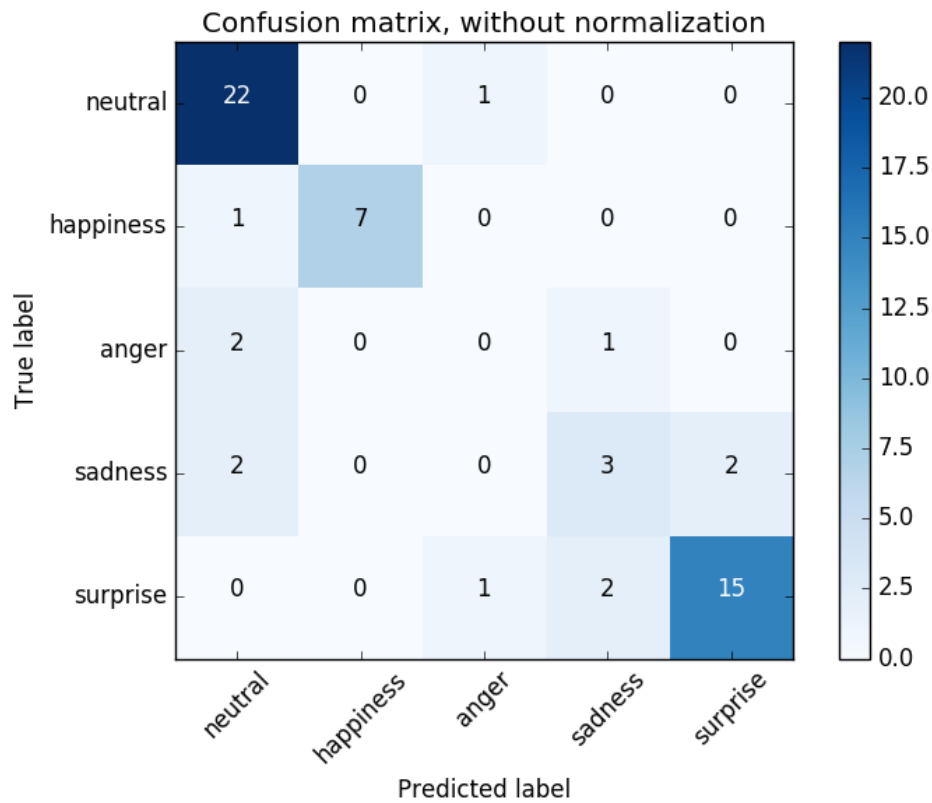
Las distintas emociones tienen expresiones faciales asociadas que, debido a los impulsos nerviosos generados, se pueden reflejar en más de una parte. Por tanto, al no incluir la información de una cierta zona facial puede tener mayor o menor impacto dependiendo de lo crítica que sea para las emociones; si una emoción se expresa únicamente en una de las partes de la cara, al retirar esa información el algoritmo no tendrá información suficiente como para determinar de la emoción de la que se trata y se confundirá o dará una previsión correcta pero no basada en el aprendizaje realizado, sino por aleatoriedad.



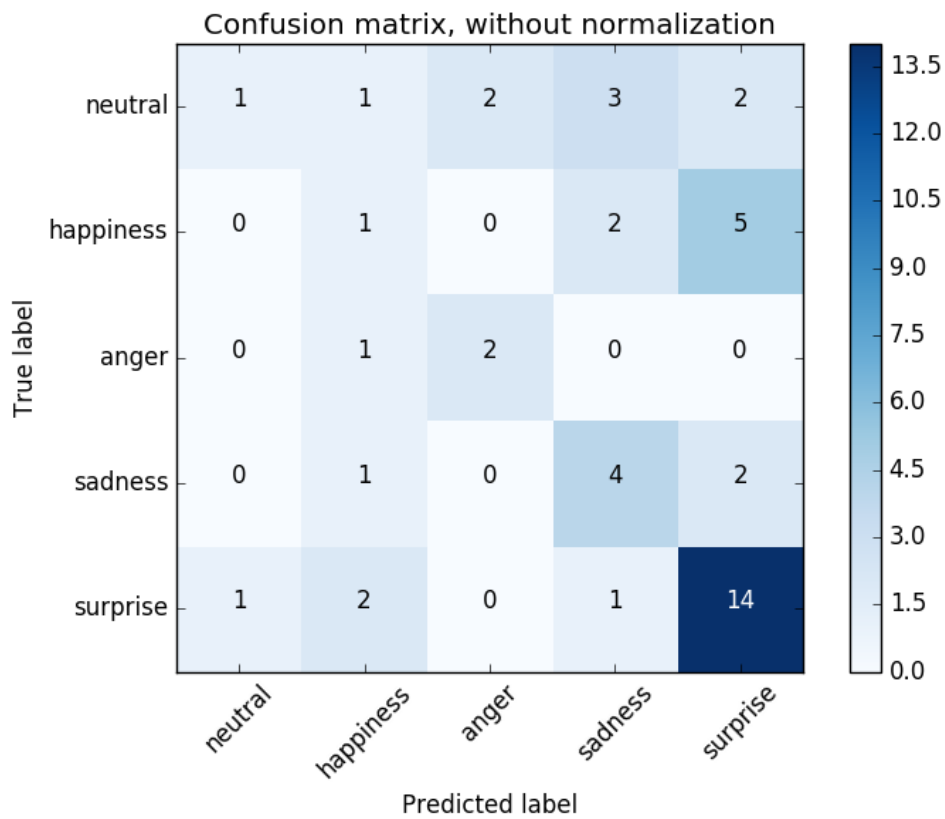
[Figura 30] Matriz de confusión sin contorno



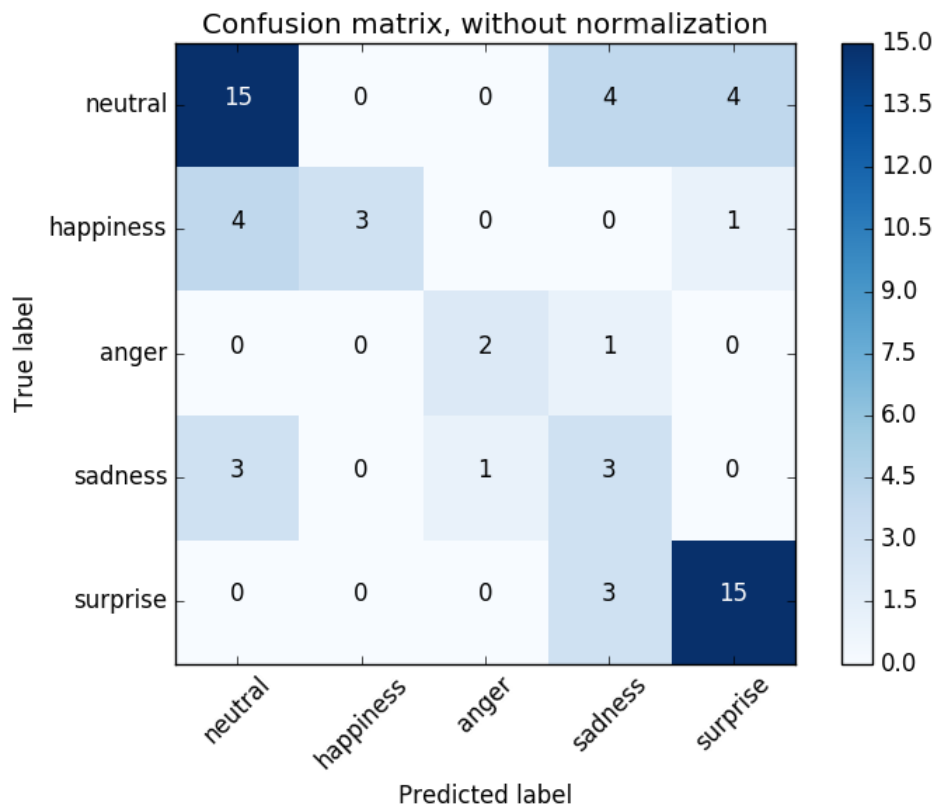
[Figura 31] Matriz de confusión solo contorno de la cara



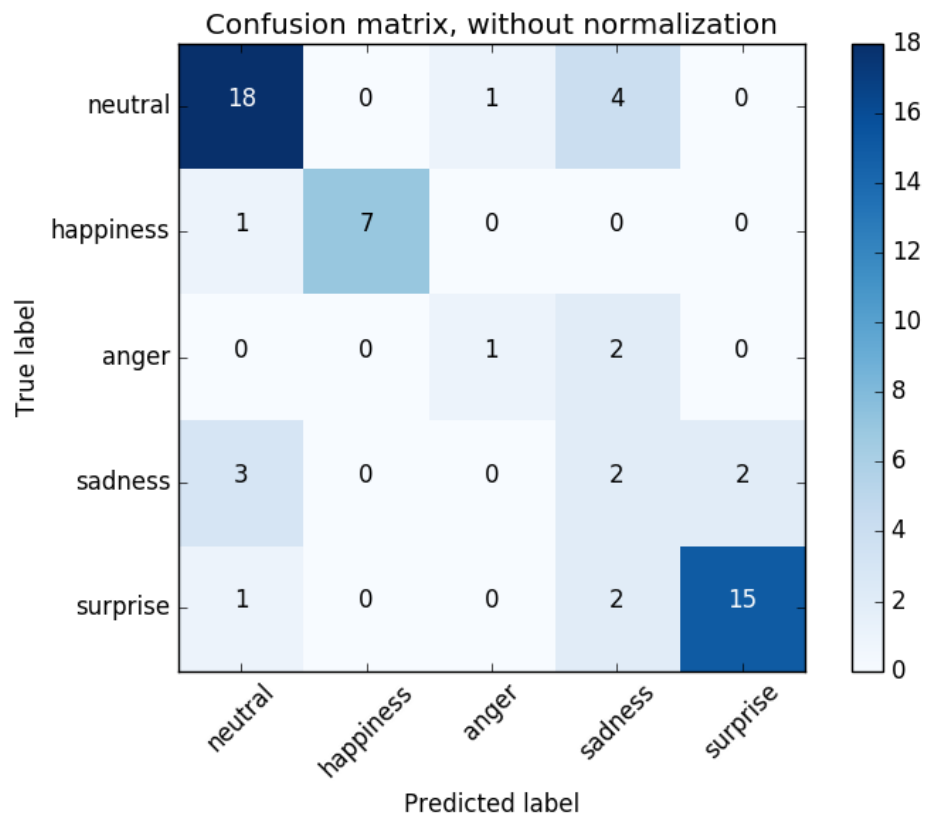
[Figura 32] Matriz de confusión sin cejas



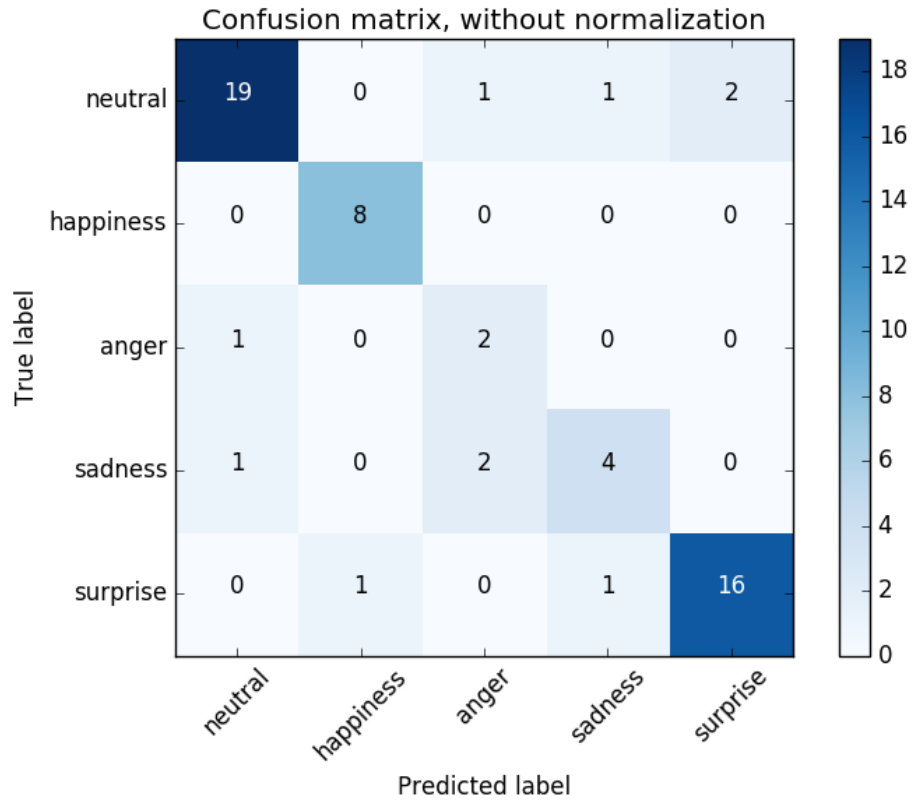
[Figura 33] Matriz de confusión solo cejas



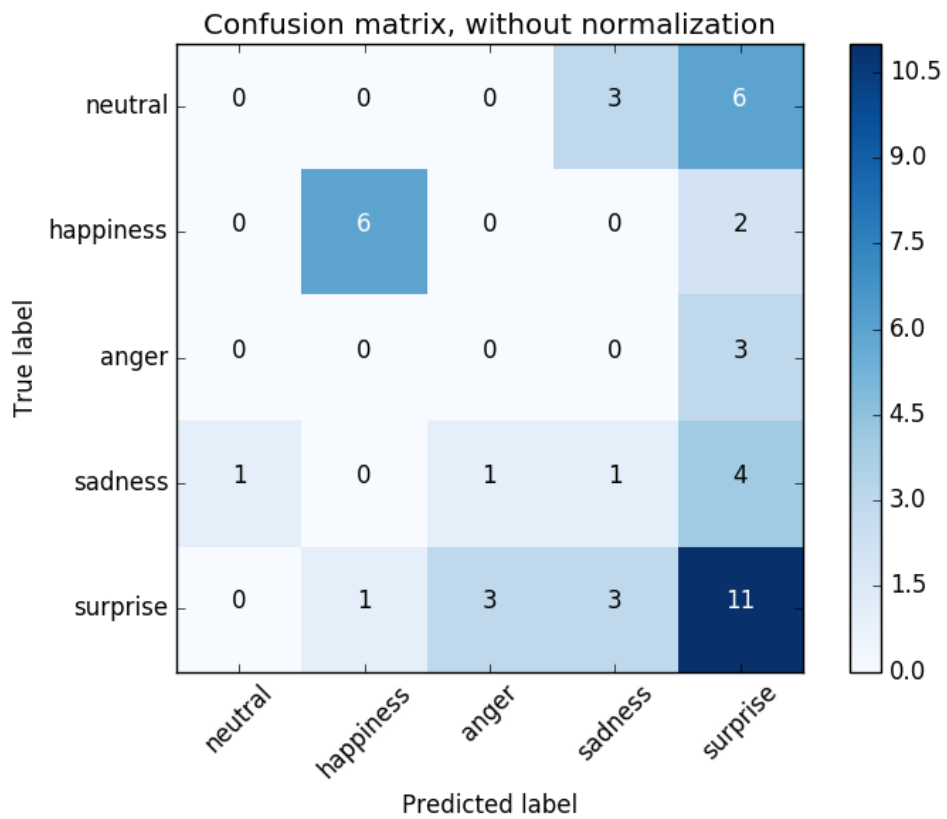
[Figura 34] Matriz de confusión sin boca



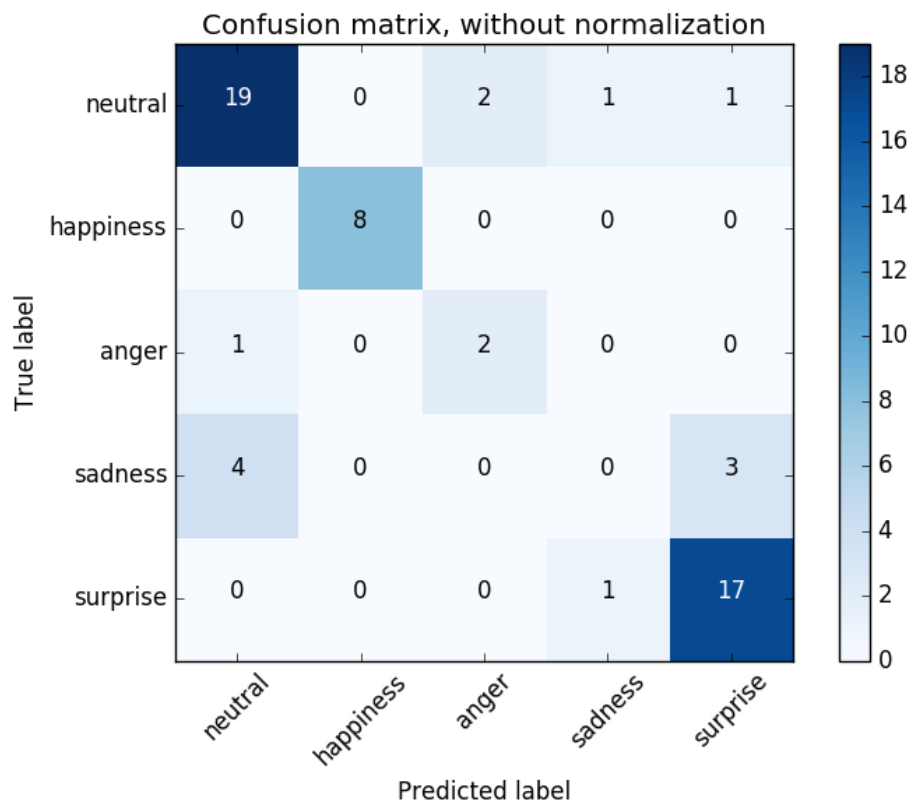
[Figura 35] Matriz de confusión solo boca



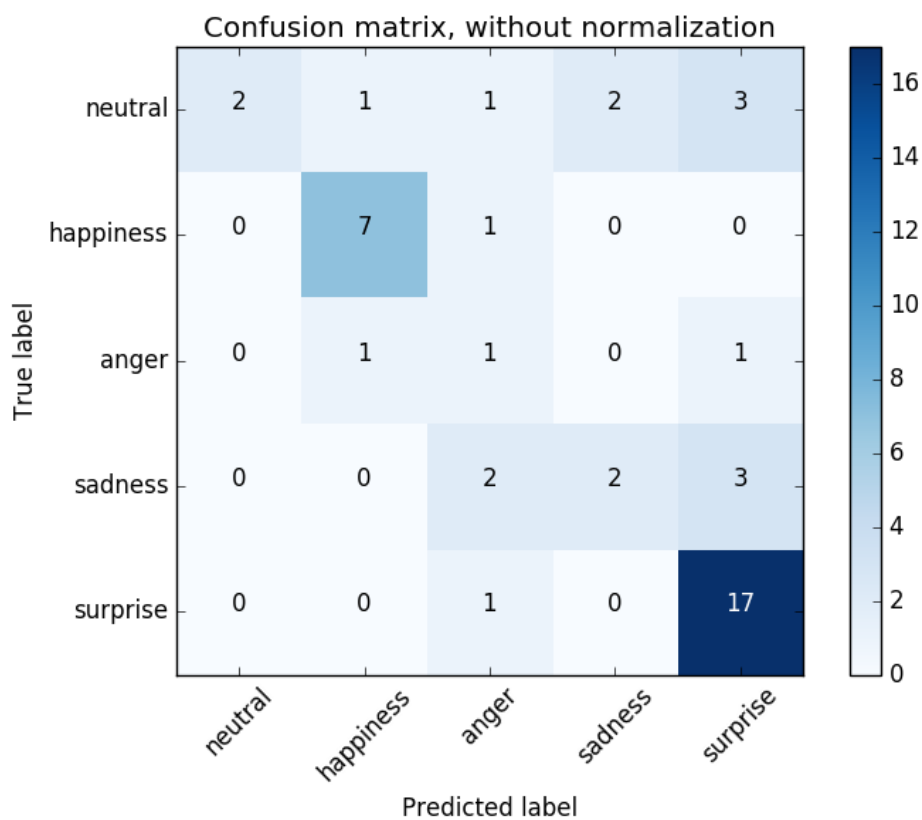
[Figura 36] Matriz de confusión sin nariz



[Figura 37] Matriz de confusión solo nariz



[Figura 38] Matriz de confusión sin ojos.



[Figura 39] Matriz de confusión solo ojos

Con los datos que obtenidos podemos se observa que, a rasgos generales, las partes de la cara ordenadas de mayor a menor importancia son:

1. Boca
2. Ojos
3. Cejas y contorno de la cara (misma importancia aproximadamente)
4. Nariz

Cuando se observa la matriz de confusión del contorno de la cara vemos que aporta información principalmente para la emoción de sorpresa; esto puede deberse a que esta emoción conlleva abrir la boca y por tanto la zona del contorno de la barbilla cambia y se diferencia. Esta zona también aporta información a la emoción de alegría ya que la parte de los mofletes se ensancha al sonreír y esto le hace más diferenciable.

Analizando las cejas se observa que aporta información a la emoción de sorpresa, ya que tendemos a subir las cejas al expresar facialmente esta emoción y por tanto se diferencia en la parte de las cejas. También aporta información a la emoción de ira, ya que al fruncir el ceño se diferencia de las otras.

La boca se podría decir que aporta información a cada una de las emociones, las emociones de alegría, sorpresa y la neutral son prácticamente identificables solo con la boca. También aporta información a las otras tres, pero con menor importancia ya que son más similares y cuando usamos únicamente la boca a veces las confunde. Cuando no incluimos la boca, la precisión baja un 20%.

La nariz es la parte que menos información aporta, tan solo aporta un poco de información con respecto a la emoción de alegría, ya que se ensancha un poco la nariz al sonreír, la precisión apenas baja al prescindir de la información de esta parte.

Los ojos aportan información principalmente a las emociones de sorpresa y alegría. En la primera, los ojos abiertos son una de las principales características, por lo que no es de extrañar que los ojos jueguen un papel importante a la hora de detectar sorpresa en la expresión facial; mientras que, en la felicidad, al sonreír subimos los pómulos y cambia la forma de los ojos.

5.2 Distintos algoritmos de aprendizaje automático

Una vez analizada la influencia de las distintas partes de la cara a la hora de clasificar las emociones, se hace una comparativa entre distintos algoritmos de aprendizaje automático para ver cual de ellos consigue un mejor rendimiento para esta tarea. Los algoritmos que se van a comparar son la Maquina de Soporte Vectorial (SVM, por sus siglas en inglés), los Bosques Aleatorios o Random Forest, el Perceptrón Multicapa y el método de los k vecinos más próximos.

Para ello van a ser entrenados utilizando exactamente el mismo procedimiento que anteriormente y vamos a obtener su precisión en el set de pruebas y la matriz de confusión correspondiente con el fin de comprobar el rendimiento y las causas de que este sea bajo o alto. Los algoritmos son comparados sin hacer un ajuste de parámetros, estos serán entrenados con los parámetros predeterminados de scikit learn.

Parámetros predeterminados en scikit learn en los algoritmos:

-SVM: $C=1$, $\text{kernel}='rbf'$, $\text{gamma} = 'auto'$ ($=1/n^\circ$ atributos = 4624)

-Random Forest: $n_estimators = 10$, $\text{max_depth} = \text{None}$, $\text{min_samples_split} = 2$, $\text{min_samples_leaf} = 1$.

-MLP: $\text{hidden_layers} = 100$, $\text{solver} = 'adam'$, $\alpha = 0,0001$.

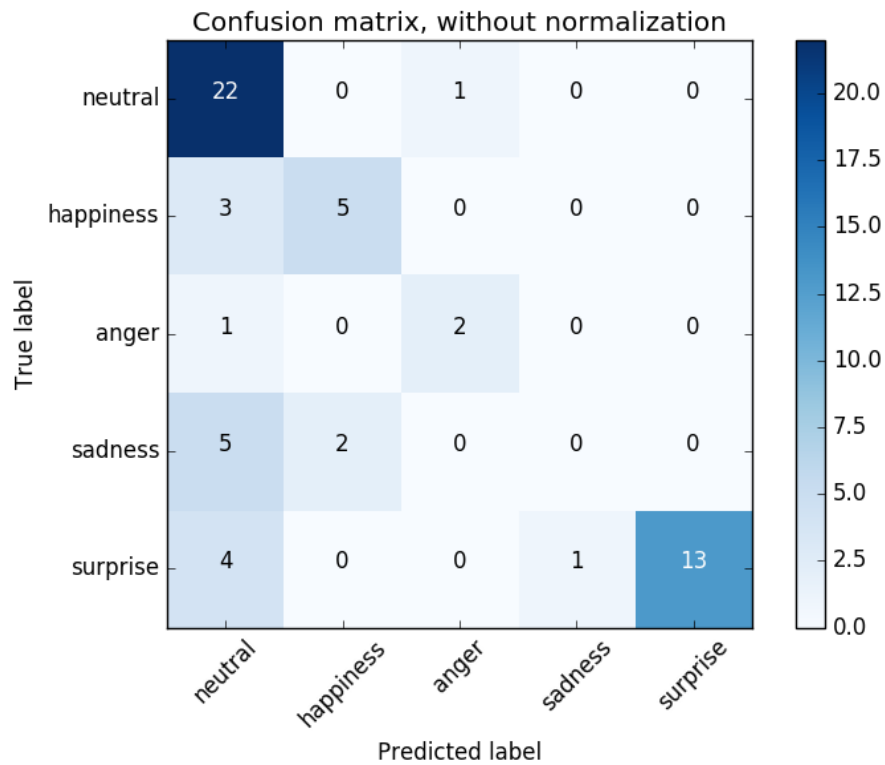
-KNN: $n_neighbors = 5$, $\text{weights} = 'uniform'$.

Una vez se entrenan los algoritmos y son probados en el set de testeo se obtienen las siguientes precisiones que se muestran en la tabla 3.

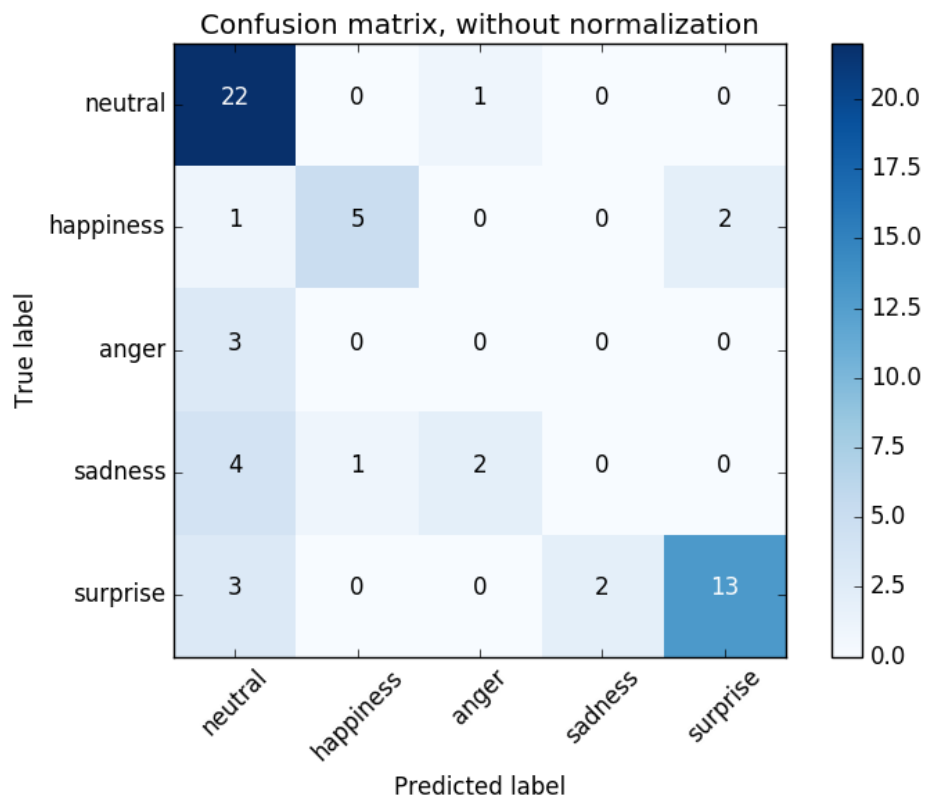
Algoritmo	Precisión
SVM	71%
Random Forest	68%
ML-Perceptron	48%
K-Vecinos	69%

[Tabla 3] Precisión con los distintos algoritmos de aprendizaje

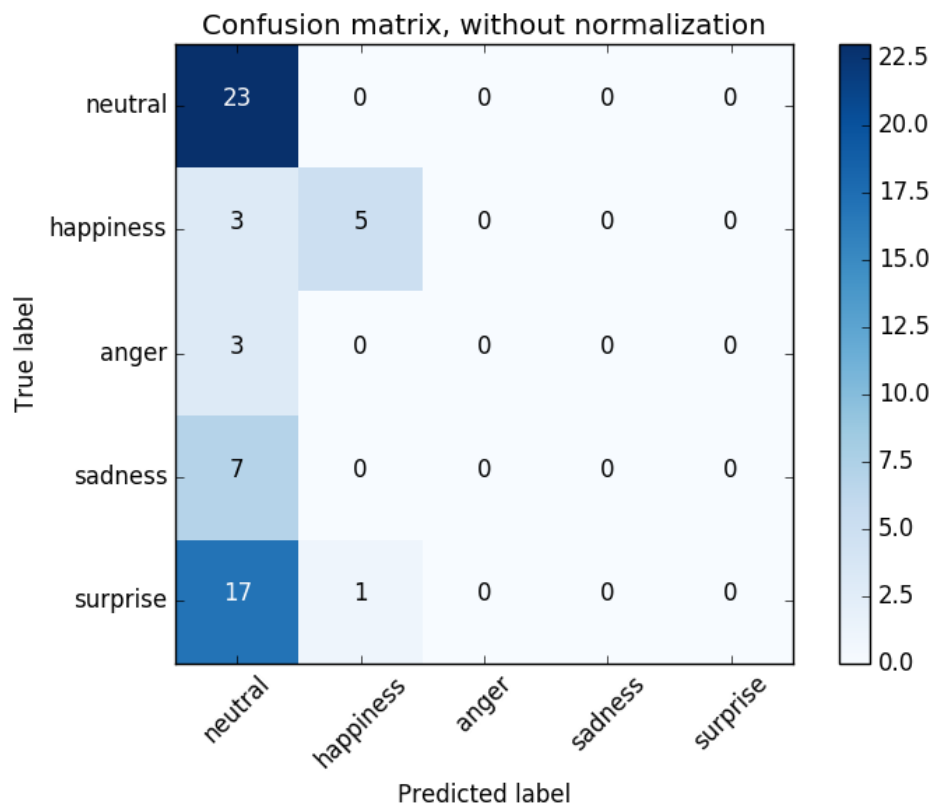
Y las siguientes matrices de confusión mostradas en las figuras 40-43:



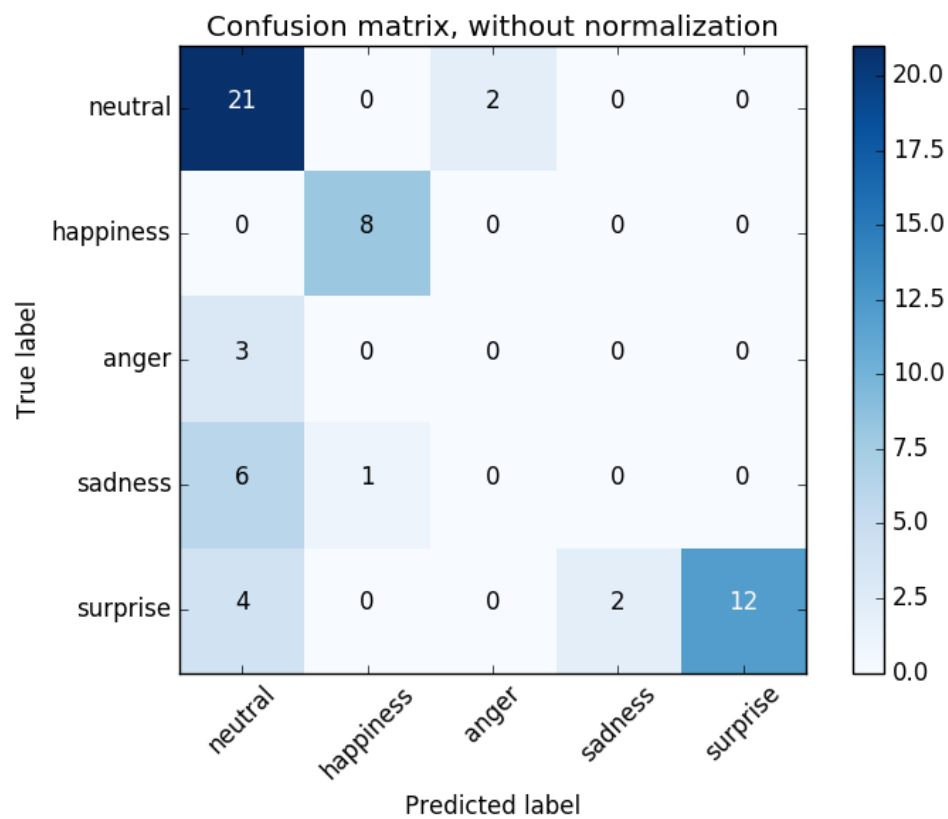
[Figura 40] SVM



[Figura 41] Random Forest



[Figura 42] Multi Layer Perceptron



[Figura 43] KNeighbors

El algoritmo que da la mayor precisión es la Máquina de Soporte Vectorial con un 71% y el que menos consigue es el perceptrón multicapa con un 48%.

Para entender los resultados obtenidos se debe que entender como es el dataset que ha sido usado para el entrenamiento ya que afectará de forma notable. Nuestro dataset es relativamente pequeño (593 instancias), con un grande número de atributos y por lo tanto los algoritmos trabajaran en hiperespacios de muchas dimensiones (4624 dimensiones) y es un dataset que no está equilibrado ya que el número de imágenes pertenecientes a la emoción neutral es mayor que el resto. Si nuestro dataset fuera otro, los algoritmos se comportarían de forma distinta.

La SVM es un algoritmo que funciona bien en espacios de muchas dimensiones y es de los más efectivos cuando además de haber muchas dimensiones hay pocas instancias de datos. El perceptrón multicapa tiene un aprendizaje lento, es decir, por cada instancia aprende menos que otro tipo de algoritmos, esto hace que en un dataset pequeño su precisión sea baja. Esto explicaría el porque estos algoritmos son el que mejor y el que peor rendimiento consiguen respectivamente. Si se lograra conseguir un dataset con un numero mucho mayor de instancias, tal que el perceptrón tuviera suficientes como para llegar al punto en el que se estabiliza la precisión, podría convertirse en el algoritmo con mejor precisión y rendimiento de todos.

El algoritmo de los K-vecinos más próximos se beneficia del hecho de que un dataset desbalanceado no le afecta tanto como a la SVM o al Perceptrón, ya que no tiene que aprender una función discriminativa, solo le afectan en el caso de que las distancias de los puntos en el hiperespacio de la clase con mayor número de instancias se sitúen cerca de otros, por eso consigue una precisión razonablemente alta.

El algoritmo de Random Forest también obtiene una buena precisión debido a que es un algoritmo que funciona bastante bien cuando se utiliza con datos de una alta dimensionalidad.

Capítulo 6: Marco Regulatorio

6.1 Análisis de la legislación aplicable

Este proyecto ha sido desarrollado casi en su totalidad con tecnologías de uso libre, por lo que se aplican pocas restricciones o regulaciones.

Respecto al software usado las licencias bajo las que son publicadas las librerías son:

- ROS, Scikit learn, OpenCV y joblib son publicados bajo licencia BSD.
- joblib es publicada bajo la Boost Software License

La licencia BSD nos permite el uso del código con cualquier propósito, incluido el comercial y para el desarrollo de software no libre. Es una de los tipos de licencia más permisivos que nos impone unas restricciones mínimas.

La licencia Boost Software License es un tipo de licencia para software libre que impone unas restricciones mínimas como en el caso de la licencia BSD.

Respecto al dataset usado se nos imponen una serie de condiciones, estas son el uso para usos no comerciales, la no distribución a terceros, la no reproducción de las imágenes de ciertos sujetos tanto de forma electrónica como impresa y la citación en caso de uso para el desarrollo de cualquier artículo o paper.

Teniendo todo esto en cuenta, las regulaciones a las que se ciñe el proyecto no son muy estrictas y puede ser empleado con fines de investigación, pero no comerciales debido a la restricción del dataset. Si se deseara tratar de comercializar el sistema se debería hacer uso de un dataset que no nos restrinja su uso para esta aplicación, o bien crear uno propio.

6.2 Estándares Técnicos

La convención PEP 8 es un estándar en la escritura de código Python, su principal objetivo es la legibilidad del código. Comprende la librería estándar de Python y atiende a aspectos como la indentación, el uso de espacios o tabulador, el número máximo de caracteres por línea, etc....

No se ha seguido el estándar ya que el código no es el principal objetivo del proyecto, sino el estudio del uso del aprendizaje automático para la detección de emociones y los parámetros que afectan al rendimiento.

Capítulo 7: Entorno socio económico

7.1 Presupuesto

Al tratarse de un proyecto desarrollado mediante software el presupuesto necesario es muy limitado. De los recursos mencionados con anterioridad ninguno a excepción de la cámara requiere inquirir en ningún coste. La cámara con la que se ha desarrollado es una webcam integrada en el ordenador portátil con el que se ha desarrollado el proyecto. El ordenador es un ASUS f555l i7 cuyo coste son 650€ en Diciembre de 2015. Si se deseara se podría adquirir una cámara con mayor resolución para tratar de mejorar el rendimiento.

7.2 Impacto socio-económico

Debido a que el trabajo consiste en la implementación y estudio de un sistema de detección de emociones cuya precisión está por debajo del estado del arte el impacto socio-económico no sería muy grande del sistema desarrollado en concreto. No obstante, los sistemas de detección de emociones implementados en la industria si conllevan un gran impacto a nivel económico, social y ético.

Para entender el impacto que tienen este tipo de sistemas tenemos que entender que estamos en la era de los datos y el big-data, cuya importancia ha crecido exponencialmente en los últimos años. Los datos ahora son los activos más valiosos de las empresas y tienen un valor monetario muy elevado, las empresas adquieren datos de otras empresas para usarlos.

A nivel económico ver que tendría un impacto en la forma de operar de las entidades financieras, ya que la digitalización ha hecho que los bancos utilicen también técnicas de aprendizaje automático para determinar la concesión de préstamos; estos se basan en datos personales para decidir si se concede o no. Conocer las emociones que sentimos serían una información muy relevante para estos pues podrían llegar a determinar el carácter y comportamiento del solicitante del préstamo y tratar de determinar quienes no lo van a devolver y por tanto no son buenos clientes.

A nivel social la detección de emociones facilita el desarrollo de robots sociales para la interacción con personas, los cuales hacen una importante labor social al interactuar con gente solitaria, asocial y poco integrada.

A nivel ético se producen muchos dilemas, nos podríamos plantear si es ético el uso de las emociones de una persona como datos y más aún en el caso de posibles usos comerciales. También nos podemos plantear si los sistemas de detección de emociones pueden llegar a ser usados en espionaje o malintencionadamente. En el caso del uso por entidades financieras también surgen dilemas, ya que la intención podría ser para hacer una discriminación y no conceder préstamos a determinadas personas en base a su estado emocional, pero también podría ser usado con la intención de la protección financiera del solicitante del préstamo que podría no ser capaz de devolver el préstamo e incurriría en graves costes y consecuencias negativas.

En general se puede decir que el impacto que tenga la aplicación de un sistema como este depende del uso y la intencionalidad. Su impacto social y económico podría ser positivo si se desarrolla adecuadamente y se emplea en aplicaciones que añadan valor.

Capítulo 8: Conclusiones y Trabajos futuros

Tras el desarrollo de este trabajo y el estudio de los resultados obtenidos en los experimentos, se exponen una serie de conclusiones que han sido consideradas como más relevantes, así como se plantean líneas de mejora y/o evolución del trabajo

8.1 Conclusiones

Las emociones que sienten las personas afectan a su comportamiento y son en muchos casos son producto de los estímulos que recibe de su alrededor. Es por ello que las emociones son una fuente de información muy útil a la hora de intentar averiguar el comportamiento de una persona o la situación y el lugar en el que se encuentra.

El reconocimiento de emociones es una tarea que las personas realizan sin esfuerzo y es de su ayuda en multitud de situaciones. Debido a que constituye una fuente de información sobre las personas de gran interés se han realizado grandes avances en el campo de la detección de emociones; muchos de ellos en la visión artificial ya que es una de las maneras más directas de detectar las emociones. Los sistemas de detección automático de emociones tienen una gran cantidad de aplicaciones en campos como la medicina, el marketing, el entretenimiento, el mundo jurídico, la robótica, etc...

Estos sistemas están basados casi en su totalidad en el uso del aprendizaje automático, una de las ramas de la inteligencia artificial. Es posible entrenar un algoritmo que sea capaz de hacer este reconocimiento de emociones con el uso de bases de datos que contengan imágenes de gente expresando facialmente estas emociones, pues siempre se expresan de la misma forma y se puede, por tanto, entrenar el algoritmo para su uso generalizado.

Pero se ha de tener en cuenta ciertos factores al abordar el problema de la detección de emociones por medio de la expresión facial; como en todo problema de aprendizaje automático hay algoritmos que tienen un rendimiento mejor que otros y este no es solo dependiente del tipo de problema, sino también de su planteamiento y de los recursos que se tengan a disposición. Mientras que unos funcionan mejor para cuando nuestro

dataset no es muy grande, otros acaban superando en precisión a medida que aumentamos el tamaño del dataset; ciertos algoritmos, como las redes neuronales convolucionales consiguen una precisión muy elevada cuando se les da los datos en bruto y una gran cantidad de ellos; mientras que otros funcionan mejor cuando hacemos una preparación de los datos y extraemos ciertas características como los puntos de interés facial. También tenemos que tener en consideración que hay partes de la cara que nos proveen de mayor información que otras ya que adoptan posiciones y formas más distintivas y pronunciadas que otras.

Podemos concluir diciendo que la detección de emociones resulta de gran interés cuando se quiere comprender el comportamiento y los alrededores de las personas, es por esto que una detección de las mismas puede ser de gran utilidad en innumerables aplicaciones ya que permiten adecuar las tecnologías a las personas y sus entornos. Las formas de lograr esto son muy variadas y es por ello que se debe tratar de comprender el problema y la información que hay disponible para conseguir un sistema que sea fiable que funcione de la forma más óptima posible.

8.2 Trabajos futuros

A partir de estas conclusiones se plantean posibles futuras investigaciones y ampliaciones al proyecto.

Se podría plantear la mejora en la precisión de las detecciones mediante el uso de técnicas más sofisticadas de aprendizaje, uso de otros conjuntos de datos que permitan a ciertos algoritmos rendir de una forma mayor, aplicación de filtros a la imagen para mejorar la detección de puntos faciales y tener predicciones en video más estables o añadir un canal de información más, creando un sistema multimodal de detección de emociones.

También se plantea el reconocimiento de las caras y detección de la persona de la que se trata, así como otros datos relevantes como podría ser el sexo o la edad de la persona en cuestión para general perfiles de las diferentes personas que identifica el sistema. Esto es una información que además de tener un gran interés por si misma, podría ayudar a la detección de emociones. Se podría intentar entrenar el algoritmo en base a la persona a la que ha reconocido y su perfil para que sea una detección personalizada.

El sistema se podría integrar además en un sistema mayor orientado a la navegación semántica ya que la información que obtenemos acerca del entorno del robot cuando este trabaja con personas puede ayudar al robot a comprender el lugar y entorno donde se encuentra ya que el lugar puede afectar las emociones y estas pueden ser un indicativo de que lugar se trata; además puede ayudar a contextualizar la narrativa semántica de las ordenes que pueda recibir el robot.

Bibliografía

- [1] Li, S., & Deng, W. (2018). Deep Facial Expression Recognition: A Survey. *CoRR*, *abs/1804.08348*.
- [2] Li, Jiaying & Zhang, Dexiang & Zhang, Jingjing & Zhang, Jun & Li, Teng & Xia, Yi & Yan, Qing & Xun, Lina. (2017). Facial Expression Recognition with Faster R-CNN. *Procedia Computer Science*. 107. 135-140. 10.1016/j.procs.2017.03.069.
- [3] Automatic Facial Expression Recognition Using Combined Geometric Features
Garima Sharma. Latika Singh. Sumanlata Gautam.
- [4] <https://imotions.com/emotient/>, Fecha de última consulta: 11/06/19
- [5] <https://hipertextual.com/2016/01/apple-emotient>, Fecha de última consulta: 11/06/19
- [6] Noroozi, F., Corneanu, C.A., Kaminska, D., Sapinski, T., Escalera, S., & Anbarjafari, G.(2019). Survey on Emotional Body Gesture Recognition. *CoRR*, *abs/1801.07481*
- [7] Ali, S.K., Yunus, R.B., Arif, A.K., Ayaz, Y., Sial, S.B., Asif, R., Naseer, N., & Khan, M.J. (2018). Hand Gestures Based Emotion Identification Using Flex Sensors
- [8] BERNHARDT, Daniel; ROBINSON, Peter. Detecting emotions from everyday body movements. *Presencia PhD Sym.*, Barcelona, 2007.
- [9] Shaw, A., Vardhan, R.K., & Saxena, S. (2016). Emotion Recognition and Classification in Speech using Artificial Neural Networks.
- [10] PFISTER, Tomas. Emotion detection from speech. *Gonville & Caius College*, 2010.

- [11] <http://go.affectiva.com/auto>, Fecha de última consulta: 11/06/19
- [12] Alonso-Martín,F., Malfaz,M., Sequeira,J., Gorostiza,J.F., Salichs,M.A. (2013).A Multimodal Emotion Detection System during Human–Robot Interaction.
- [13] <https://www.lifeder.com/psicologia-emocional/>, Fecha de última consulta: 11/06/19
- [14] <https://www.verywellmind.com/theories-of-emotion-2795717>, Fecha de última consulta: 11/06/19
- [15] Darwin, C. (1872). The expression of the emotions in man and animals. London, England: John Murray.
- [16] Cannon, W. B. (1927). The James-Lange theory of emotions: a critical examination and an alternative theory. *The American Journal of Psychology*, 39, 106-124.
- [17] Dror, O. E. (2014). The cannon–bard thalamic theory of emotions: A brief genealogy and reappraisal. *Emotion Review*, 6(1), 13-20.
- [18] P. Ekman and W. V. Friesen. Facial Action Coding System. Consulting Psychologists Press Inc., 577 College Avenue, Palo Alto, California 94306, 1978.
- [19] Ekman, P., Levenson, R. W., & Friesen, W. V. (1983). Autonomic Nervous System Activity Distinguishes Among Emotions. *Science*, 221(4616), 1208-1210.
- [20] Rachel E. Jack, Oliver G.B. Garrod, Philippe G. Schyns. Dynamic Facial Expressions of Emotion Transmit an Evolving Hierarchy of Signals over Time. *Current Biology* (2014). DOI: 10.1016/j.cub.2013.11.064.
- [21] <https://www.ros.org/>, Fecha de ultima consulta: 11/06/19
- [22] <https://opencv.org/>, Fecha de última consulta: 11/06/19

- [23] Viola, P. & Jones, M.J. International Journal of Computer Vision (2004) 57: 137. <https://doi.org/10.1023/B:VISI.0000013087.49260.fb>
- [24] <http://dlib.net/>, Fecha de última consulta: 11/06/19
- [25] <https://scikit-learn.org/stable/>, Fecha de última consulta 11/06/19
- [26] <https://joblib.readthedocs.io/en/latest/>, Fecha de última consulta: 11/06/19
- [27] https://scikit-learn.org/stable/modules/model_persistence.html, Fecha de última consulta: 11/06/19
- [28] N. Cristianinio and J. Shawe-Taylor, An introduction to support Vector Machines: and other kernel-based learning methods. Cambridge University Press, 2000
- [29] Michel, Philipp & El Kaliouby, Rana. (2003). Real time facial expression recognition in video using support vector machines. Proceedings of the International Conference on Multimodal Interfaces. 10.1145/958432.958479.
- [30] Dapogny, Arnaud & Bailly, Kevin & Dubuisson, Séverine. (2015). Pairwise Conditional Random Forests for Facial Expression Recognition. 3783-3791. 10.1109/ICCV.2015.431
- [31] Panchal, Goutami & N Pushpalatha, K. (2017). A Local Binary Pattern Based Facial Expression Recognition using K- Nearest Neighbor (KNN) Search. International Journal of Engineering Research and. V6. 10.17577/IJERTV6IS050284.
- [32] Abidin, Zaenal & Harjoko, Agus. (2012). A Neural Network based Facial Expression Recognition using Fisherface. International Journal of Computer Applications. 59. 30-34. 10.5120/9531-3956

[33] Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), Grenoble, France, 46-53.

[34] Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101.

[35] Gilligan, T. (2016). Emotion AI, Real-Time Emotion Detection using CNN.