

Domótica y Entornos Inteligentes



Práctica 2

Francisco Javier Pérez Martínez Francisco Joaquín Murcia Gómez

Óscar David Tremiño Guirao Marcos Cerdán Amat Álex Navarro Soria

25 de febrero de 2022

Índice

1. Descripción	3
2. Monitoreo de atención	3
2.1. Extracción de puntos	3
2.2. Cálculo del pitch roll yaw	4
2.3. Clasificación	5
2.4. Ejecución y pruebas	5
3. Control de asistencia	6
3.1. Data set	6
3.2. Entrenamiento	6
3.3. Ejecución y pruebas	6
4. Detección de mascarillas	7
4.1. Algoritmo	7
4.2. Ejecución y pruebas	7
5. Enlaces de interés	7

1. Descripción

Para el presente documento, a partir de las funcionalidades planteadas y desarrolladas en el documento de Teoría, se han implementado dos funcionalidades pertenecientes a los epígrafes de Seguridad y Educación del documento mencionado. Concretamente, las funcionalidades relacionadas con un monitoreo de atención de los alumnos en el aula (funcionalidad principal) y un sistema de asistencia y detección de mascarillas (funcionalidades secundarias) previo a la entrada de la clase.

2. Monitoreo de atención

Para la realización del algoritmo de monitoreo de atención, a modo de resumen, se han obtenido los seis puntos de la cara para así extraer los parámetros de orientación (pitch, roll y yaw) que obtenemos calculando la distancia 3D y 2D de los 6 puntos de la cara con respecto al foco de la cámara. Dependiendo del valor de dichos parámetros determinamos si el alumno esta atendiendo o no, para así dejarlo registrado en una gráfica a tiempo real. En los siguientes apartados, se explicará mas detalladamente todo el proceso seguido para llevar a cabo dicho algoritmo.

2.1. Extracción de puntos

Uno de los primeros pasos que se han realizado para poder realizar una correcta implementación de la funcionalidad, ha sido obtener los puntos de la cara que van a ser relevantes para la implementación. Lo primero que se ha realizado es la obtención de los puntos de la cara, los cuales van a ser obtenidos mediante el uso de un Face_mesh. Después de haber realizado la obtención todos los puntos de la cara, debemos filtrar los puntos que no consideramos relevantes, procesando únicamente los puntos que van a ser de importancia en la implementación. Estos puntos son:

- 2 puntos para los ojos. (Uno para el izquierdo y otro para el derecho)
- 1 punto para la nariz.
- 2 puntos para los extremos de la boca.
- 1 punto para la barbilla.

Cada punto de los utilizados tendrá un valor asociado dentro del Face_mesh, los cuales son los siguientes:

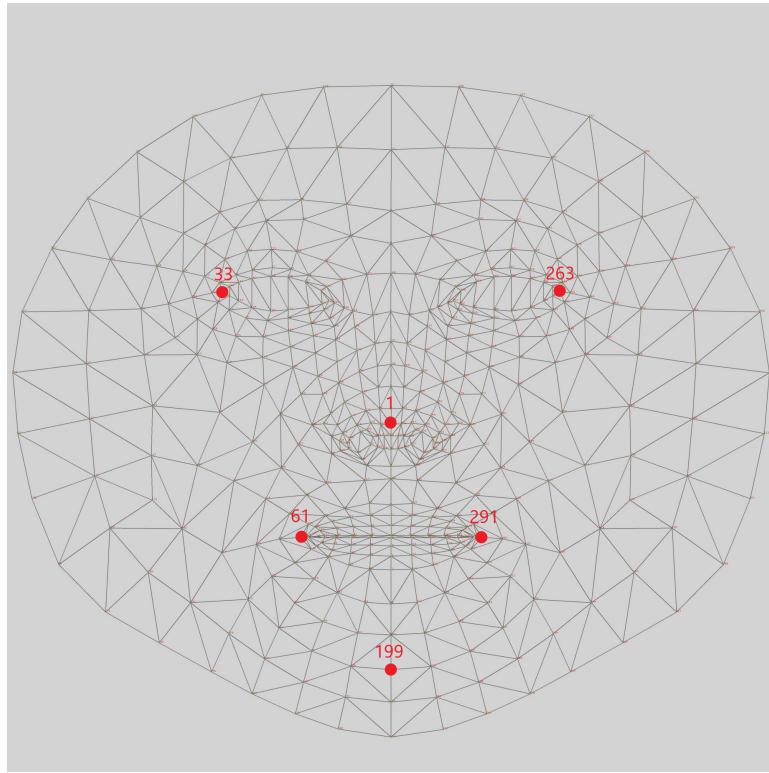


Figura 1: Valores de los puntos relevantes

2.2. Cálculo del pitch roll yaw

Para calcular el vector rotacional se ha usado Perspective-n-Point una fórmula para obtener dicho vector.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Figura 2: Formula Perspective-n-Point

Los parámetros corresponden respectivamente a:

- El resultado sería el Vector de rotación (pitch roll yaw)
- La primera matriz correspondería a los parámetros de la cámara, como los puntos focales, la coordenada central y el parámetro de inclinación
- Las matrices restantes a las coordenadas 3D y 2D.

A partir de dicho vector, se obtienen los ángulos rotacionales de cada eje haciendo una descomposición RQ de la matriz del vector. Obteniendo así los dos ángulos que usaremos para clasificar, el cabeceo (pitch) y la guñada (yaw)

2.3. Clasificación

Para la clasificación de los valores resultantes, se ha decidido realizar la implementación de un sistema de reglas lógicas simple. Simplemente se comprueban que los valores resultantes del cabeceo y la guiñada, no sean superiores a 10° ni inferiores a -10° . Si no cumplen alguna de estas condiciones, se considerará como “No atiende”, teniendo asignado el valor numérico 0. Por el contrario, si cumple todas las condiciones implementadas, el resultado será “Atendiendo”, teniendo asignado como valor numérico 1.

2.4. Ejecución y pruebas

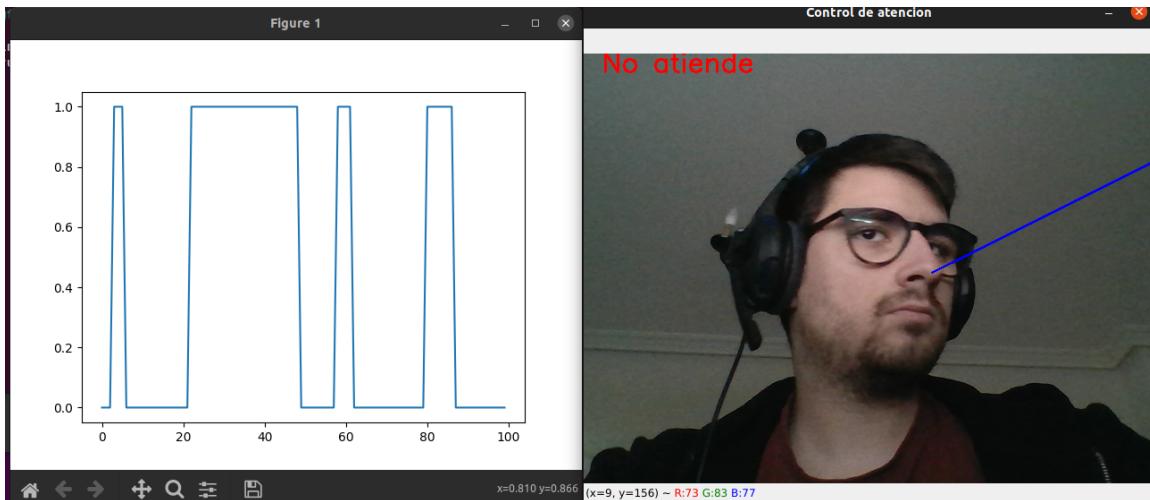


Figura 3: Ejemplo de persona no atendiendo

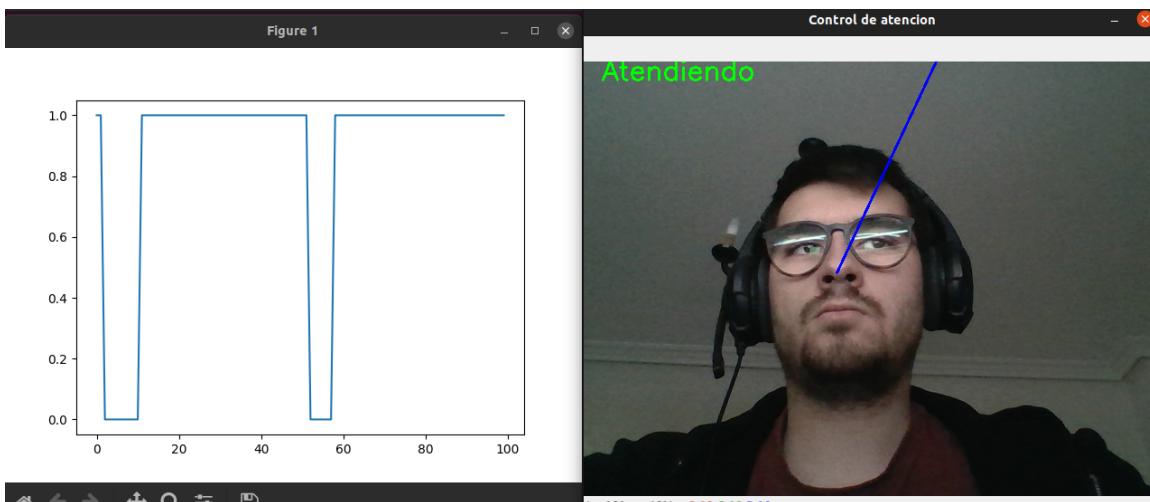


Figura 4: Ejemplo de persona atendiendo

Como podemos observar en la figura 3, al no estar atendiendo el alumno se transforma en un valor ”0” en la gráfica, mientras que en la figura 4 al estar atendiendo la gráfica dibuja el valor ”1”, de esta forma se tiene un seguimiento del tiempo atendido y no atendido de manera gráfica.

3. Control de asistencia

Para el control de asistencias se ha desarrollado un sistema de clasificación de rostros, el cual detecta el rostro de un alumno e indica su nombre.

3.1. Data set

Se ha realizado es un dataset de unas 300 imágenes por persona, las cuales contienen fotos de los rostros de las personas a reconocer. Estas imágenes se les ha aplicado una serie de rotaciones y de filtros de luz para maximizar la eficacia. Cada una de las imágenes posee una etiqueta asociada a la persona a la que pertenecen el rostro rostros.

3.2. Entrenamiento

Para la fase de entrenamiento se ha usado el método de vectores propios EigenFaces que emplea el método matemático PCA(Principal Component Analysis) para la clasificación de rostros. Hemos elegido EigenFaces ya que tarda menos tiempo a la hora de entrenar y la diferencia de aciertos no es mucha. Cuando entrenamos el modelo lo guardamos como XML.

3.3. Ejecución y pruebas

Para la hora de probar el modelo obtenemos el valor de confianza de cada etiqueta, la etiqueta con mejor valor de confianza es la etiqueta a la cual pertenece el rostro.



Figura 5: Ejemplo de un alumno reconocido

4. Detección de mascarillas

Para la siguiente funcionalidad hemos desarrollado una aplicación que reconoce si el alumno lleva la mascarilla puesta o no.

4.1. Algoritmo

Para ello, haremos uso de los módulos OpenCV, Numpy y MediaPipe con Python. Para comenzar, necesitaremos utilizar 2 conjuntos de dataset. El primer conjunto de fotos será de gente que lleva mascarilla y el segundo de gente que no lleva mascarilla.

El módulo de MediaPipe será el encargado de refactorizar las imágenes de todos los conjuntos para que tengan el mismo ancho y alto, así como cualquier filtro necesario para estabilizar los colores y sombras.

Una vez tengamos los dataset correctamente implementados, entrenaremos un modelo con LBPH (Local binary patterns histogram). Esta es una técnica de reconocimiento facial que vamos a usar para el algoritmo de reconocimiento de mascarillas.

Con todas estas herramientas aplicadas, ya tendremos un modelo entrenado que nos servirá para completar la funcionalidad final.

4.2. Ejecución y pruebas



Figura 6: Alumno sin mascarilla



Figura 7: Alumno con mascarilla

5. Enlaces de interés

Se ha creado un vídeo de la ejecución en directo de las funcionalidades de control de atención y del detector de mascarillas. https://youtu.be/rX7vHTxFu_0

También en el siguiente enlace se encuentra el repositorio de github con los códigos de cada funcionalidad: <https://github.com/fmurciag/CV-DEI-Funcionalidades>