Detección de Parpadeo para Comunicación a Través de Código Morse

---

Inteligencia Artificial

Profesor: Julio Godoy
Alumnos:
Cristian Contreras Moreno
Diego Henríquez Valenzuela
Catalina Pezo Vergara
11.06.2021

Resumen

El siguiente documento detalla la implementación de un sistema de detección de parpadeo para comunicación utilizando código morse. Se explican los motivos de esta implementación y algunos conocimientos previos. Luego se exponen las librerías utilizadas y se explica el código del sistema implementado. Finalmente se expresan las limitaciones que tiene este sistema y las ideas que nos gustaría desarrollar en el futuro.

Problema

El ser humano emplea principalmente su capacidad vocal para entablar una comunicación verbal. Sin embargo, con la pérdida de esta facultad surge la necesidad de comunicarse a través de diversos movimientos de sus articulaciones, los cuales permiten una comunicación mediante lenguaje de señas o mediante la asistencia tecnológica con la cual pueden digitar sus señales.

Sin embargo, en el ámbito clínico se han descubierto una gran cantidad de enfermedades, tales como el síndrome de enclaustramiento, la esclerosis lateral amiotrófica, entre otras enfermedades, y accidentes que concluyen en distintos tipos de parálisis corporales o desorden del habla, las cuales discapacitan al paciente de lograr una comunicación.

Nuestra investigación y posterior trabajo se centra en presentar una solución para el caso de los pacientes que presentan una parálisis completa, los cuales solo pueden pestañear para lograr enviar señales de comunicación, y pueden ver para recibir señales.

Código Morse

El código morse es un sistema de representación de letras y números mediante señales emitidas de forma intermitente, y consiste en un alfabeto base para establecer un canal de comunicación.

Las reglas de este código son las siguientes:

- Existen tres símbolos en el alfabeto Morse: Punto, Raya y Separador.
- La duración del punto es la mínima posible.
- Una raya tiene una duración de aproximadamente tres veces la del punto.
- Entre cada par de símbolos de una misma letra existe una ausencia de señal con duración aproximada a la de un punto.
- Entre las letras de una misma palabra, la ausencia es de aproximadamente tres puntos.
- Para la separación de palabras transmitidas el tiempo es de aproximadamente tres veces el de la raya.

Estos símbolos pueden ser representados mediante una función motora del ser humano: el pestañeo o el abrir y cerrar los ojos durante determinados periodos de tiempo.

Implementación

Para el desarrollo de la implementación de este proyecto se decidió emplear el lenguaje de programación Python 3, el cual ofrece una amplia gama de librerías y herramientas para el desarrollo de programas relacionados con la inteligencia artificial.
En el caso nuestro proyecto, necesitamos adentrarnos en la disciplina científica Computer Vision (Vision Artificial), con el objetivo de adquirir, procesar, analizar y comprender imágenes del mundo real para producir información numérica o simbólica para que puedan ser tratados por un ordenador.

OpenCV

OpenCV es una librería de código abierto de visión artificial, análisis de imágenes y aprendizaje automático. Dispone de una gran cantidad de algoritmos que permiten identificar rostros, reconocer objetos, clasificarlos, detectar movimientos de manos.
En nuestra implementación, OpenCV es utilizado para capturar la transmisión de imágenes desde la cámara web, para poder procesarlas en el programa. También empleamos esta librería para mostrar en una ventana los fotogramas de la cámara web en tiempo real, junto con trazos y texto que permiten comunicar al usuario final información relevante. Por último, se emplea para comprimir los fotogramas convirtiéndolas a escala de grises, el cual es un espectro de colores más reducido.

Dlib

Dlib es un moderno kit de herramientas de C ++ que contiene principalmente algoritmos de machine learning. Es una librería destinada a resolver problemas del mundo real. Se utiliza en la industria y en el mundo académico en una amplia gama de dominios, incluyendo la robótica, dispositivos integrados, teléfonos móviles y grandes entornos de computación de alto rendimiento. Además, posee una versión compatible con el lenguaje de programación utilizado en este proyecto, Python.
En nuestra implementación, Dlib es utilizado para detectar la cara frontal de cada imagen, lo cual es logrado mediante una función incluida en la librería: get_frontal_face_detector. Detrás de esta simple función se encuentra un detector de caras mediante HOG (histograma de gradientes orientados) + Linear SVM(máquinas de vectores de soporte lineales).

A partir de cada cara detectada en los fotogramas (previamente comprimidos a escala de grises) se aplica un predictor de puntos faciales.

Shape Predictor 68 Face Landmarks

Para la detección de los facial landmarks se emplea un modelo creado por Davis King [1].

Este modelo fue entrenado con un reconocido dataset de anotaciones de puntos faciales [2]. El dataset adjunta a cada imagen de una cara los 68 puntos faciales correspondientes, ingresados manualmente por los colaboradores.

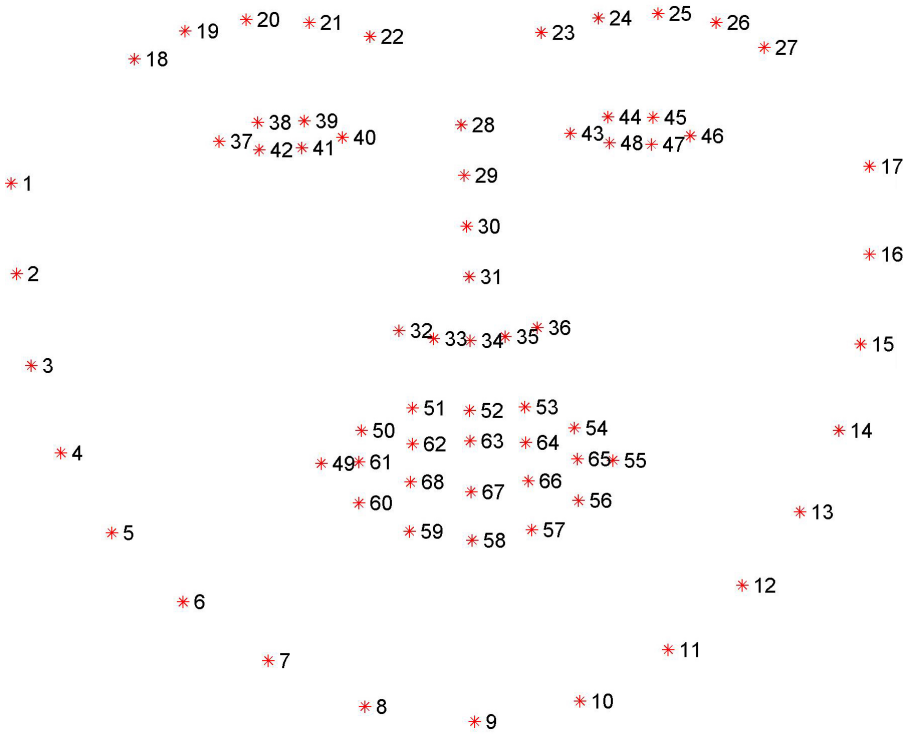

Figura 1: *68 puntos faciales*

Por lo tanto, el modelo de igual manera está entrenado para reconocer estos 68 puntos de referencia faciales. También se tiene en cuenta que este modelo está diseñado para su uso con el detector facial HOG+LSVM de dlib (el que estamos utilizando). Es decir, espera que los cuadros delimitadores del detector facial estén alineados de cierta manera, como lo hace el detector facial HOG de dlib. No funcionará tan bien cuando se usa con un detector facial que produce cuadros alineados de manera diferente, como el detector de caras basado en CNN(redes neuronales convolucionales).

Reconocimiento de la Apertura y Cerrado de Ojos en una Imagen

Se detectan los puntos de referencia del ojo en una imagen.

Dlib reconoce 6 puntos por cada ojo, con lo cual se puede formular una ecuación que indica la relación de aspecto entre la altura y el largo de cada ojo (EAR):

$$EAR = \frac{\|p2-p6\| + \|p3-p5\|}{2\,\|p1-p4\|}$$

Para saber cuando el ojo está abierto o cerrado se compara con el EAR threshold. El valor óptimo de este último puede variar según la fisionomía del usuario.

Código

El código consiste principalmente en 3 funciones y el ciclo while donde se hace el análisis *frame* por *frame* de lo capturado por pantalla.

Funciones:

- **get_midpoint:** obtiene el punto medio entre dos puntos.
    - Parámetros: dos pares.
    - Retorna: par.
- **get_EAR:** obtiene la relación de aspecto del ojo (EAR) de un ojo.
    - Parámetros: puntos del ojo, landmark.
    - Retorna: número decimal.
- **morse2char:** traduce una secuencia de código morse a una letra.
    - Parámetro: cadena de caracteres.
    - Retorna: caracter, o -1 en caso de que no se haya encontrado la traducción.

El ciclo while es la parte principal del código, que itera hasta presionar la tecla “ESC”. Acá analiza por cada ciclo el frame obtenido por cámara, reconociendo el ojo y calculando los EARs respectivos, con tal de verificar si la persona pestañeó o no. A través de distintas variables booleanas y medidores de tiempo se determina la duración del pestañeo o apertura del ojo, lo que se traduce a morse.

Modo de Empleo

Para hacer funcionar este programa, como se mencionó anteriormente, se deben tener instaladas las librerías Matplotlib, OpenCV y Dlib. Además, el uso del programa considera los archivos “recordar-codigo-morse.png” y “shape_predictor_68_face_landmarks.dat”, que contienen una traducción al código morse y los 68 puntos faciales que se describieron en una sección anterior, respectivamente.

La ejecución del código se efectúa ingresando en la terminal:

```
python main.py
```

Seguido de esto se abrirán 3 ventanas creadas con OpenCV:

1. Cámara, donde se marca la identificación de los ojos con líneas verdes.
2. Cuadro de texto, donde irá apareciendo la traducción a lenguaje natural de lo escrito en morse.
3. Imagen referencial con el código morse.

Además, en la terminal se irá escribiendo los "." y "-" reconocidos por los pestañeos.

La manera en que el programa funciona y traduce los pestañeos reconocidos a morse se basa en la toma de dos tiempos clave: la duración de mantener el ojo abierto (t_open) y la duración de mantener el ojo cerrado (t_close). Con esto, obtenemos las traducciones:

- "." = 0<t_close<0.5
- "-" = 1.5<t_blink<3
- Separador de letras: 3<t_open<5
- Separador de palabras (" ") = t_open>5

Cabe destacar, que para comenzar a escribir, se debe ingresar a "Modo Escritura", esto se logra manteniendo los ojos cerrados por al menos 5 segundos (t_close>=5). Para salir de "Modo Escritura" se debe volver a mantener cerrados por 5 segundos.

Esto se hace para indicar que el usuario realmente quiere comunicar algo a través de sus pestañeos en lugar de sólo pestañear por costumbre.

Para salir del programa, se debe presionar la tecla “ESC”, para lo cual un asistente externo al paciente con tetraplejia debe asistir.

Limitaciones

El presente programa se tiene considerado para ser usado para una persona con tetraplejía, por lo que todo aspecto comunicativo (traducción a morse y espacios) está considerada realizarse sólo con la vista. Sin embargo, aspectos como el inicio y fin de programa deben ser asistidos por parte de un tercero.

Además, como está destinado a un único paciente, el programa debe ser usado por solo un usuario que aparezca en la pantalla de la cámara, pues sólo se considerarán los pestañeos de dicha cara única.

Se debe considerar que el programa y la traducción de pestañeos a morse requiere práctica en ser usado, pues se debe adquirir la costumbre de mantener los párpados cerrados y abiertos, respectivamente y cuando corresponda, por un cierto tiempo determinado.

Trabajo Futuro

En el futuro continuaremos desarrollando este sistema, haciéndolo más amigable hacia el usuario. Queremos hacer una interfaz en la que se pueda cambiar el EAR threshold, de forma que el sistema se acomode mejor al usuario particular.

Además, consideramos que escribir utilizando pestañeos con el sistema tal y como está ahora puede ser algo difícil. Para esto queremos implementar un sonido que marque el ritmo de las unidades de tiempo, de forma que sea más fácil para el usuario escribir.

Finalmente, queremos permitir al usuario poder cambiar la velocidad de escritura, cambiando las unidades de tiempo. Así un usuario nuevo podría aprender a utilizar el sistema con mayor facilidad con una velocidad menor, mientras que un usuario experimentado podría comunicarse más rápido sin estar limitado por la velocidad del sistema.

Referencias

1. Davis King. (s.f). Dlib-models. Github https://github.com/davisking/dlib-models
2. C. Sagonas, S. Zafeiriou, et al. (s.f). Facial point annotations. Ibug. https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/