**UNIVERSIDAD TECNOLOGICA**

**NACIONAL**

**Facultad Regional Avellaneda**

Procesamiento Embebido de Señales

Ingeniería Electrónica

Control Gestual de Volumen y Brillo para Computadora con Python y Visión por Computadora

Alumnos

Cristian Bruten

brutencristian@gmail.com

Alejandro Conti

alejandroxconti@gmail.com

Docentes

Ing. Martín Hernán Enriquez

Ing. Axel Lucas Gomez

2024

**Índice**

|  |  |
| --- | --- |
|  |  |
|  |  |
| 1. Introducción | 3 |
| 2. Planteamiento del problema | 3 |
| 3. Objetivos | 3 |
| 4. Justificación | 3 |
| 5. Alcance del proyecto | 3 |
| 6. Etapas de desarrollo | 4 |
| 7. Herramientas de software y equipos utilizados | 4 |
| 8. Resultados esperados | 4 |
| 9. Conclusión | 6 |
| 10. Referencias | 6 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* **1. Introducción**

En el presente informe trataremos el proyecto final de la materia “Procesamiento Embebido de Señales”, el cual se trató de “Controlador de brillo y audio a través de procesamiento de imágenes y símbolos”, en el cual nos basamos en gestos propios de la mano para traducir dichos gestos en la configuración de dos parámetros importantes de una computadora.

* **2. Planteamiento del Problema**

Los métodos tradicionales de ajuste de volumen y brillo requieren contacto físico con algún periférico de la computadora, lo cual puede ser poco práctico en determinadas situaciones o en equipos con controles físicos limitados.

La implementación de un sistema de control gestual permitirá a los usuarios realizar estos ajustes de forma rápida y natural.

* **3. Objetivos**
* **Objetivo General**

El objetivo general fue desarrollar un sistema de control de volumen y brillo para una computadora utilizando gestos de manos detectados mediante visión por computadora.

* **Objetivos Específicos**
* Implementar un sistema de detección de manos y reconocimiento de dedos usando Python y librerías como OpenCV y Mediapipe.
* Asignar niveles de brillo y volumen en función de la cantidad de dedos levantados.
* Desarrollar una funcionalidad de cambio de modo (brillo o volumen) mediante el reconocimiento de mano derecha/izquierda y cantidad de dedos levantados para ajustar dichos niveles o a través de detección de "área mínima" de cada mano.
* Crear una interfaz visual simple que indique al usuario el modo actual y el nivel seleccionado.
* **4. Justificación**

Este proyecto ofrece una alternativa eficiente al control de volumen y brillo mediante métodos tradicionales. El uso de gestos como interfaz de entrada ofrece una experiencia interactiva, especialmente útil para usuarios que buscan simplificar su interacción con la computadora. Además de mejorar la accesibilidad, fomenta el uso de interfaces de usuario avanzadas y podría ser aplicada en otros sistemas de control en el futuro.

* **5. Alcance del Proyecto**

**Etapa 1:**

* **Detección de manos y reconocimiento de dedos**: Implementación de un modelo de detección que reconozca la cantidad de dedos levantados para asignar los niveles de volumen o brillo.
* **Control de volumen y/o brillo**: Ajuste de estos parámetros en cinco niveles según la cantidad de dedos detectados (1-5).
* **Alternancia entre modos (volumen o brillo)**: Implementación de uno de los dos métodos de distinción (mano izquierda/derecha o gesto de "área mínima").

**Etapa 2**:

* **Interfaz gráfica** (opcional): Visualización en tiempo real del nivel de ajuste y del modo activo en la computadora.
* **Control de brillo y volumen:** Ajuste de ambos parámetros en cinco niveles según la cantidad de dedos detectados (1-5), mostrando en texto la cantidad de dedos detectados.
* **Barras de nivel de ajuste:** Muestra el valor ajustado tanto para nivel de volumen como intensidad de brillo.
* **Dos Checkbox o casillas de verificación:** Una para seleccionar si se ve o no el video, y otra para optar por fondo negro en la cámara.
* **Modularidad del código**: Estructura del código en módulos separados para facilitar futuras mejoras y actualizaciones.
* **Sugerencias de mejoras en la Etapa 2**: Si la aplicación ya fue iniciada, el

botón de “Iniciar Detección” quedará deshabilitado, y quedará solamente el botón de “Detener Detección” y viceversa al quedar detenida la detección.

También para evitar cambios bruscos y que se realice mas estable el control, se agrega una demora en la detección

* **6. Etapas de Desarrollo**

**6.1 Investigación y Análisis**:

En primera instancia se estudió las posibilidades de cómo aprovechar correctamente la cámara de una notebook para captar imágenes. Nos encontramos ante la problemática que el entorno de desarrollo elegido no funcionaba, el cual era Google colab, debido a esto emigramos a jupyter notebooks.

Luego de poder captar video la siguiente rama de investigación fue la detección de la mano humana con alta fidelidad, la primera opción estudiada fue el uso de “OpenCV”, una fuente de código abierto utilizada en procesamiento de señales e imágenes, sin embargo, los resultados obtenidos no fueron los mejores debido a que esta librería utiliza dos métodos para detección de manos:

-Detección por forma:

Esta forma de detección no fue exitosa debido a la imprecisión con la que adoptaba la forma de la mano, en varias pruebas realizadas jamás se lograron buenos resultados.

-Detección por tonalidad de piel: Este otro método funciono mejor que el anterior, pero posee un gran defecto, al ser dependiente de los colores detectados tiene un gran margen de error. En varias pruebas realizadas el método fallo puesto que los colores madera (como un mueble que se encontraba en el fonde de la imagen durante las pruebas) posee un espectro de color que se asemeja mucho al de la piel. Además de este gran defecto, cabe aclarar, que el cuello de una persona también la detectaba como una posible mano, puesto que cumple las condiciones de detección por color.

Ante estas problemáticas se intentó combinar ambos métodos, pero no se consiguieron resultados positivos.

Luego de continuar investigando se optó por los métodos existentes en “MediaPipe”, otra fuente de código abierto enfocada en la detección de imágenes, la cual ya posee métodos muy precisos de detección de manos, los cuales constan de trazar una matriz de puntos y líneas para detectar la forma natural de una mano humana con un alto valor de fidelidad.

El último tramo de la investigación consto en cómo controlar el brillo y el audio de la computadora con Python, lo cual se resolvió con las librerías “screen\_brightness\_control” para el control del brillo y “pycaw” para el control del audio.

Se agrego un bloque extra de investigación el cual consistió en buscar como implementar una pequeña interfaz para el proyecto, la cual culmino con el ensayo e implementación de la librería “tkinter” y “threading”, las cuales sirvieron para realizar la interfaz y darle una funcionalidad de hilos de ejecución a la aplicación.

**6.2 Desarrollo del Sistema de Detección**:

Como se mencionó anteriormente utilizamos “MediaPipe” para implementar esta sección del proyecto. Detallando más el proceso utilizamos el método “mp.solutions.hands” para el modelado de la mano y con openCV detectamos el video sobre el cual se analizó la presencia de una mano. Esta librería traza una matriz de 21 puntos con el objetivo de definir la mano humana. Cada uno de estos puntos posee coordenadas x y z en la imagen.

Con “hands.process()” logramos diferenciar entra la mano izquierda y derecha, sin embargo se estudió el caso de detectar la mano de frente o el dorso de la mano, porque es válido pensar que a un usuario se le puedo ocurrir utilizar la aplicación así. Con una serie de condicionales se solucionó esta problemática consultando por la posición de los Gráfico

Descripción generada automáticamentepuntos situados entre el pulgar y el meñique, para clarificar lo mencionado colocamos la siguiente imagen:

En la cual se comparó la coordenada del punto número 1 y el numero 17 para diferenciar entre frente y torso una vez detectada de que mano se trata.

Luego con comparaciones de coordenadas x o y (según corresponda) se contaron el número de dedos extendidos, el cual se traduce en la intensidad de brillo y volumen deseado.

**6.3 Implementación del Control de Volumen y Brillo**:

Con las librerías mencionadas previamente fue muy sencillo el control de estos parámetros. La librería “screen\_brightness\_control” posee un método directo para setear el brillo, el cual es “sbc.set\_brightness()”.

En el caso del audio se confecciono una pequeña función que trata del seteo del master de audio de la computadora, puesto que podríamos modificar alguna de las varias posibles sesiones de audio presentes, la función de llamo “set\_volume()”.

En ambos casos se tomó el contador de dedos correspondiente y se lo multiplico por 20, debido a que, al contar con un máximo de 5 dedos por mano el salto elegido fue de a 20%.

6.4 **Implementación del Cambio de Modo**:

Se estudio la comparativa entre ambos métodos mencionados:

-El primer método consistió en la simple separación de un control por mano. Con la mano izquierda se controló el brillo y con la mano derecha se controló el audio. Debido a una limitación que no se encontró el motivo (no pudimos conseguir que le sistema funcione con los dos controles en simultaneo), se limitó la aplicación a que detecte una mano a la vez.

-El segundo método, que hemos finalmente descartado, consistió en utilizar una sola mano para controlar ambos niveles de ajuste, y utilizando como alternancia entre ambos, la detección de “Área mínima” colocando la mano de perfil. Este método no resultó ser preciso en la detección ya que también dependía de la distancia entre la cámara y la mano. Se agregó como adicional en la detección de alternancia, que el pulgar este en línea con el meñique cuando se coloca la mano de perfil, y se agregó un delay de tiempo una vez que detecta estas condiciones, para evitar cambios bruscos en el cambio de controles. El código resulta más complejo en comparación a la detección de ambas manos.

**6.5 Estabilidad de gestos y cambios graduales**

Para evitar ajustes erráticos, se implementan dos mecanismos:

* + **Historial de dedos**: Almacena las últimas muestras detectadas para verificar estabilidad.

def is\_stable(history):

return len(set(history)) == 1

Esto asegura que un cambio en el brillo o volumen solo ocurra si la cantidad de dedos levantados es consistente en las últimas 5 detecciones.

Garantiza que los cambios en el volumen o el brillo sean **consistentes y estables**, en lugar de realizar ajustes con base en detecciones individuales, que podrían ser erróneas o fluctuantes debido a imprecisiones en la detección de gestos.

La detección de gestos puede ser inexacta o variar de un fotograma a otro debido a:

* Movimientos bruscos de las manos.
* Problemas con la iluminación.
* Errores en la detección de puntos clave (landmarks) de MediaPipe.

Si el programa ajustara el volumen o el brillo cada vez que detecta un número diferente de dedos levantados, estos cambios pueden ser erráticos. Por eso, se verifica si la cantidad de dedos detectada ha sido **consistente** durante los últimos fotogramas antes de realizar un ajuste.

6.6 **Retardos temporales**: Se controla el tiempo mínimo entre cambios consecutivos para evitar ajustes demasiado rápidos.

DELAY\_THRESHOLD = 0.3 # 300 ms de retardo mínimo

Se optó por valor de delay que se pueda visualizar, pero en la práctica, se puede llevar a valores más bajos, cumpliendo con la función de cambios mas graduales.

**6.7 Pruebas y Validación**:

Se realizaron pruebas a lo largo de todo el proyecto, por lo que está probada la aplicación en su totalidad. La interfaz realizada cuenta con un botón de inicio de detección, un botón de finalización de la detección, una tilde que define si el usuario quiere ver o no el video, dos barras que indican el nivel ajustado para el volumen y el brillo, y una tilde donde se puede seleccionar si se desea visualizar los valores presentados y los textos con un fondo negro en la pantalla de la cámara.

* **7. Herramienta de software y equipos utilizados:**
* **Python 3**: Lenguaje principal para el desarrollo del software.
* **Librerías de Python**: OpenCV, Mediapipe, pycaw, screen-brightness-control, Tkinter.
* **Computadora con cámara**: Para ejecutar y probar el sistema de control de volumen y brillo mediante gestos.
* **8. Resultados**
  1. **Resultados Esperados**
* Implementación de un sistema que ajuste el volumen y el brillo de la computadora de manera intuitiva, según los gestos detectados.
* Eficiencia y precisión en la detección de gestos de manos y ajuste de los niveles.
* Prototipo funcional listo para pruebas y mejorable con extensiones para otros controles.

**8.2.** **Resultados obtenidos**

* Sistema capaz de detectar con gran precisión el número de dedos extendidos y, dependiendo de la mano detectada, setear el nivel de brillo/volumen correspondiente al número de dedos extendidos multiplicado por un 20%.
* Sistema con interfaz para mayor entendimiento del usuario y con la posibilidad de visualizar o no el video, o presentar el video con fondo negro, visualizando solo la información relevante.
* Comparación entre dos formas diferentes de realizar el cambio entre controles, obteniendo una gran diversidad de posibilidades en lo que funcionamiento se refiere.
* **9. Conclusion**

La principal conclusión que nos llevamos de este proyecto es la experiencia de haber conseguido realizar un sistema de seteo de parámetros con detección y procesamiento de una imagen. También es válido aclarar que nos vimos forzados a utilizar mediapipe debido a su clara ventaja frente a openCV, lo cual fue un gran hallazgo por las grandes posibilidades que esta fuente de código abierto dispone. Dentro del uso de esta librería cabe destacar que nos fue mucho más fácil tratar con la mano completa que intentar diferencias por área que ocupa la mano.

* **10. Referencias**
* OpenCV:

<https://vovkos.github.io/doxyrest-showcase/opencv/sphinx_rtd_theme/index.html>

* MediaPipe:

<https://ai.google.dev/edge/mediapipe/solutions/guide?hl=es-419>

* Sreen\_brightness\_control:

<https://pypi.org/project/screen-brightness-control/>

* Pycaw:

<https://pypi.org/project/pycaw/>

* Tkinter:

<https://docs.python.org/3/library/tk.html>

* <https://www.toolify.ai/es/ai-news-es/control-de-volumen-avanzado-mediante-gestos-opencv-python-646685#google_vignette>
* https://github.com/Deepika2909/Volume-control-using-hand-gesture-using-python-and-openCv/blob/main/Code.py