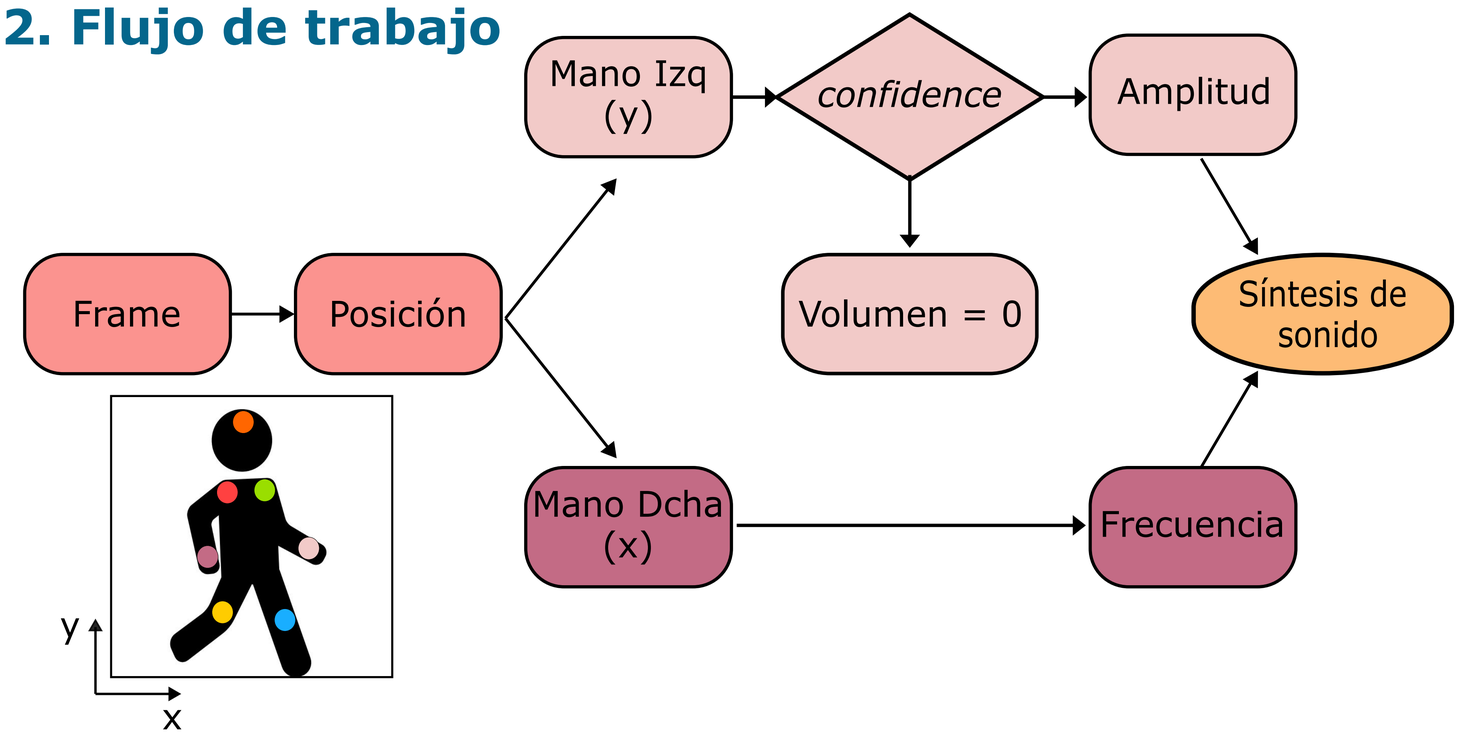
TheremIAn
Theremin con Inteligencia Artificial

Benas Sabrina[1,2], Emina Facundo[1,2] , Morales Julian[1,3]
[1]Universidad de Buenos Aires
[2]Laboratorio de Fisiología y algoritmos del Cerebro - FIL
[3]Laboratorio de óptica cuántica - DEILAP, CITEDEF

1. Introducción

El Theremin fue uno de los primeros instrumentos electrónicos creado por los años 1920 donde el intérprete no toca físicamente el instrumento; en cambio, controla el tono y el volumen mediante gestos en el aire. Está formado por en dos antenas: una vertical que controla la frecuencia (tono) y otra horizontal que controla el volumen. Al mover las manos cerca de estas antenas, se altera el campo electromagnético que las rodea afectando el sonido generado. En este trabajo se propuso realizar un Theremin con inteligencia artificial, tal que la computadora reconozca la posición de las extremidades de una persona en tiempo real mediante el uso de una GPU. Como en un Theremin clásico, la posición de las extremidades detectadas controla los parámetros que definen el sonido generado. Presentamos un stand interactivo para desarrollar nociones básicas de la generación de sonido.

2. Flujo de trabajo



DeepLabCut

DeepLabCut es un paquete de Python que sirve para estimar la pose de un animal y objetos con inteligencia artificial.

Para estimar la pose de una persona, usamos una red neuronal entrenada con más de 10000 imágenes etiquetadas de humanos (ImageNet).

En cada frame adquirido, se evalua la red neuronal y la computadora es capaz de reconocer en tiempo real las extremidades de una persona.

Mapeo de posición *x* a frecuencia *f*

$$f = f_0 + \frac{x}{A}(f_{max} - f_0)$$

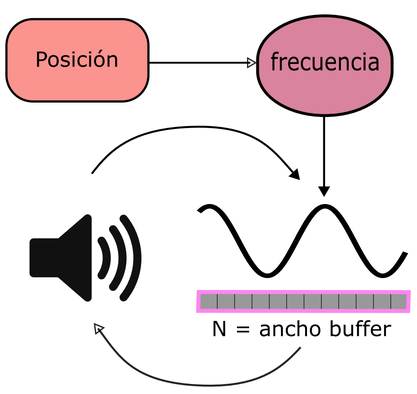
Definimos la frecuencia f en base a la posición x normalizandola dentro del la parámetro cámara A. Las variables f_{max} y f_0 definen el rango de frecuencias generadas.

3. Síntesis de sonido en tiempo real

La generación de sonidos que dependen de parámetros que varían en tiempo real se hace cada vez más difícil a medida que se complejiza la riqueza espectral de la onda (¡sobre todo si los cómputos se realizan en Python!)

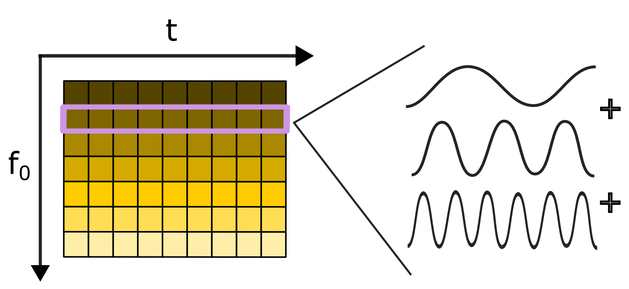
3a. Onda monocromática

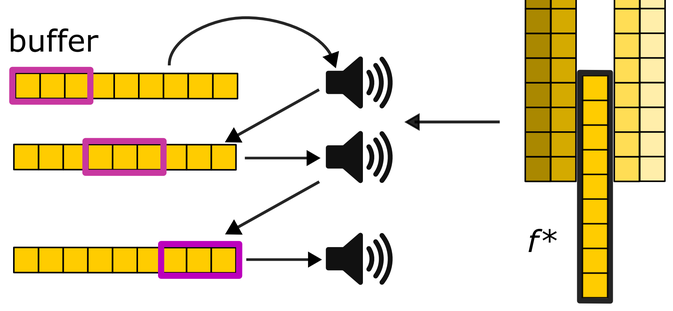
Hasta que la frecuencia cambie, recursivamente generamos N samples de la onda y reproducimos el sonido.



3b. Suma de armónicos

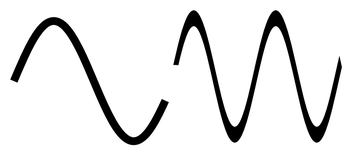
Dada una distribución espectral A_n, pre-generamos ondas de duración finita para una sucesión de frecuencias fundamentales.





Hasta que la frecuencia cambie, se reproduce la suma de armónicos previamente calculada.

4. Continua



$$\sin(2\pi f t_i) \rightarrow \sin(2\pi f' t_{i+1} + \phi)$$

$$\phi = 2\pi(f - f')t_i$$

Para poder generar un sonido que varie de forma continua en el tiempo es necesario ir actualizando la fase frente a cambios de frecuencia.

5. Discreta

Escala bien temperada

Dada una frecuencia mapeada por la posición *x*, se reproduce la frecuencia más cercana del conjunto de notas

$$f_u = 2^{\frac{u}{12}} f_0$$

Escala Pentatónica

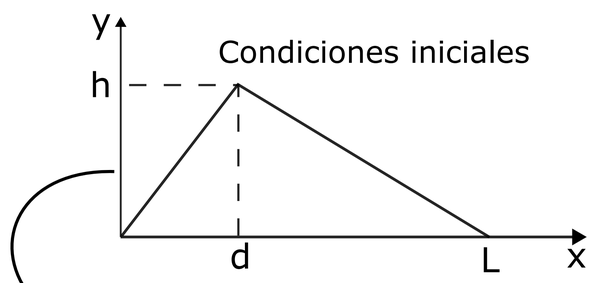
Es una sub-selección del conjunto anterior dado por los índices *{0,3,5,7,10}*

6. Suma de armónicos: sonido de guitarra

Ecuación de onda con rigidez (l) y amortiguamiento (γ)

$$\frac{\partial^2 y}{\partial x^2} - \frac{1}{v^2}\frac{\partial^2 y}{\partial t^2} - \gamma\frac{\partial y}{\partial t} - l\frac{\partial^4 y}{\partial x^4} = 0$$

$$y(x,t) = \sum_n^\infty \sin\left(\frac{n\pi x}{L}\right) A_n \cos(2\pi f_n t) e^{-n\gamma t}$$



$$f_n = n f_1 \left(1 + n^2 \frac{l^2\pi^2}{L^2}\right)^{1/2}$$

$$A_n = \frac{2hL^2}{n^2\pi^2 d(L-d)} \sin\left(\frac{n\pi d}{L}\right)$$

7. Perspectivas futuras

Mejorar la implementación del sonido con riqueza armónica en tiempo real, integrado al output de DeepLabCut.

Implementar librerias que generan sonido de forma más eficiente, como Pyo que compilan la generación de sonido en C, e integrarlo al output de DeepLabCut.