Detección y búsqueda de Patrones de Sonido

Di Lorenzo, Maximiliano Rodrigo	38166442
González Romero, Matías	38325166
Meneghini, Daniel Maximiliano Ezequiel	38841427
Lell, Matías Nicolás	38819912
Rivero, Facundo	39486259

Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

<<maximiliano.rdl@gmail.com, gonzalezromeromatias@gmail.com,
mdanielmaximiliano@gmail.com,
matias.n.lell@gmail.com, faa.rivero@gmail.com >>

Resumen.

La base de esta investigación es asentar los fundamentos teóricos con respecto a la captura de la frase de una persona mediante un sensor de sonido, para en un futuro poder desarrollar un módulo que permita agregar a nuestra aplicación IOT ("LiMaS") la capacidad de detectar un patrón en la orden de la persona que hemos mencionado y al ocurrir este evento se envía una alerta al sistema arduino. el cual realizará la acción correspondiente a lo pedido por el usuario. El fin es que el sistema pueda ser manejado por medio de la voz sin la necesidad de utilizar el celular

Palabras clave: Sonido**,** Patrones, Voz, Reconocimiento, OpenMP

1. Introducción

El proyecto de nuestro grupo se basa en un sistema de gestión de luces de un hogar. Dicho sistema actualmente posee la característica de encenderse al detectar un aplauso, detectar presencia en el ambiente, o mediante la aplicación del celular. De la misma forma, se podrá apagar por los mismos medios (en el caso de la presencia, al dejar de detectarla). Además las luces se ajustan automáticamente dependiendo de la luz del entorno.
De todas formas, una limitación del mismo es que no posee la capacidad de encenderse frente a comandos de voz del usuario. Es por eso que buscamos incorporar al sistema la capacidad de recibir órdenes, como por ejemplo "encender", "apagar", "aumentar luz", mediante comandos de voz ejecutados por el usuario.

2. Desarrollo

2.1. Detección de la voz

Como punto de partida de la investigación consideramos necesario estudiar en primer lugar cómo el oído humano permite reconocer patrones de sonido, para luego poder entender cómo trabajan los distintas aplicaciones informáticas ya existentes.
Según Caldarelli y Campanella (2003) [1], el oído en el hombre tiene un rango muy amplio de operación que abarca de tres a cuatro órdenes de magnitud en el conjunto de frecuencias que son audibles (desde alrededor de 20 a 20 MHz). Esto indica que el oído funciona como un detector de amplio espectro que permite sensar el sonido proveniente de muy diversas fuentes. Se requiere por ende de un procesamiento avanzado de esta información para identificar su origen y sus cualidades.
A su vez, un aspecto importante es el referente a definir la capacidad de diferenciar tonos; ¿cuál es la diferencia mínima que puede ser apreciada?. A bajas frecuencias (por ejemplo 100 Hz) se requieren cambios hasta del tres

por ciento en un tono para detectar la diferencia; en cambio, para tonos de frecuencias mayores (por ejemplo 2000 Hz) variaciones de 0.5 por ciento bastan para distinguir la diferencia (Caldarelli y Campanella, 2003) [1].
La capacidad de notar diferencias tonales depende también de la duración de los tonos, y es más o menos independiente de la amplitud del sonido.
Por todo lo expuesto anteriormente se desprende que una de las características esenciales a definir en el proceso de captura de la señal de voz es la frecuencia de muestreo. Este factor es muy importante, pues es la limitante y posible causante de diferenciar entre una buena calidad de señal y los problemas que se pueden presentar si no se respetan las reglas que el procesamiento digital de señales enmarca.
Para poder llevar a cabo el procesamiento digital de una señal, según Albiñana (2014)[5], lo primero que se debe hacer es transformar la señal de sonido analógica es una señal digital, es decir el sonido debe ser representado con números binarios. Por lo tanto el sistema debe tener un conversor analógico a digital (ADC).
Se usará la modulación por impulsos codificados (PCM,Pulse Code Modulation) la cual se basa en el teorema de muestreo (Stallings, 2004)[3]. Dicho teorema dice que "Si una señal f(t) se muestrea a intervalos regulares de tiempo con una frecuencia mayor que el doble de la frecuencia más alta de la señal, las muestras así obtenidas contienen toda la información de la señal original. La función f(t) se puede reconstruir a partir de estas muestras mediante la utilización de un filtro paso baja." (Stallings, 2004, p. 158)[3].
De esta manera, la conversión tendrá tres etapas

- Muestreo
- Cuantización
- Codificación

Muestreo

El muestreo funciona midiendo la amplitud de la señal continua a intervalos de igual duración (Stallings, 2004)[3]. La distancia temporal o el intervalo de tiempo que hay entre dos muestras consecutivas se denomina período de muestreo, y se mide en segundos. Su inversa fm = 1/ Tm se denomina frecuencia de muestreo, y se mide en ciclos por segundo o Hz . Debido a esto, en el proceso de muestreo pasamos de una señal continua a un conjunto de muestras (es decir, puntos discretos en el tiempo). Es importante muestrear la señal lo suficientemente rápido como para capturar toda la información (Stallings, 2004)[3]. El teorema de muestreo, o teorema de Nyquist, demuestra que para representar adecuadamente una señal senoidal es necesario tener al menos dos muestras por cada ciclo de la sinusoide (Stallings, 2004)[3]. Por tanto, para representar adecuadamente un sonido, la frecuencia de muestreo fm tiene que ser mayor, como mínimo, del doble de la frecuencia más alta contenida en la señal:

$$fm \geq 2 \cdot fmaxima$$

Cuantización

La cuantización se realiza al limitar los posibles valores de amplitud de una señal, definiendo una serie discreta (no continua) de valores posibles (Stallings, 2004). [3]

Codificación

El proceso de codificación consiste en asignar un código binario o conjunto de bits a cada uno de los valores posibles de las muestras de la señal (Stallings, 2004) [3].

Una vez que hayamos logrado convertir las señales analógicas de sonido deberemos concentrarnos en ver que tipos de algoritmos se suelen utilizar para el reconocimiento de patrones de sonido o voz. Encontramos que uno de estos algoritmos para llevar a cabo este procedimiento es (Simancas y Meléndez-Pertuz, 2017; Albiñana, 2014) [2] [5]:

Recolección de muestras de voz: Se toman varias muestras de voz, tanto en ambientes silenciosos como con ruido.

Pre-procesamiento de Señales: Consiste en un filtro digital que procesa las señales por medio de la Transformada Rápida de Fourier, con el fin de eliminar el ruido externo de las señales obtenidas en la etapa de recolección.

Sistema de reconocimiento de patrones: Una vez depuradas las señales obtenidas se procede a analizar el patrón de frecuencia de las mismas para poder definir un ancho de banda de frecuencias que nos permita reconocer el sonido buscado con facilidad

2.2. Búsqueda de similitud de palabra

Luego de obtener la señal de la voz, el sistema deberá proceder a realizar una búsqueda de esa señal modulada, intentando encontrar una similitud entre los patrones guardados en la base de datos, la cual tendrá por cada orden del sistema diferentes señales diferenciadas por sexo, edad, acento, ruido ambiente y demás factores que pueden afectar a la identificación de la señal.
De acuerdo a lo presentado por Coffin (2002) hay 6 tipos de registros vocales: Soprano,Mezzosoprano y alto,para voces femeninas. Tenor, Barítono y bajo para voces masculinas. los cuales presentan aproximadamente 5 subcategorías en cada una de ellas.
Tomando solo las instrucciones que el sistema posee actualmente (encender, apagar, bajar intensidad, subir intensidad, titular, alarma), agregandosele futuras posibles interacciones al sistema. agregandosele el factor de ruido ambiental el cual puede variar completamente la señal (Stallings, 2004) se deberan tomar un número muy grande de muestras para poder tener una respuesta confiable por parte del sistema, siendo esta dada por:

$$r = \sum_{i=0}^{a} ((v*sv)*m)$$

donde **r** son la cantidad de registros de la base de datos. **a** la cantidad de órdenes que recibe el sistema. **v** los distintos registros, **sv** los subtipos de cada registro y **m** la cantidad de muestras tomadas con distintos ambientes. Siempre teniendo en cuenta que mientras mayor sea **más** certero será el sistema a la hora de interpretar la orden, disminuyendo así resultados erróneos

Debido a esto, la base de datos deberá ser grande para lograr un buen funcionamiento. Optamos por usar OpenMp para paralelizar la búsqueda adaptando lo realizado por Al-Dabbagh, Barnouti, Naser y Ali (2016) [4], debido a que de esta manera se paralelizan las búsquedas si se puede obtener un resultado en menos tiempo, permitiendo una rápida respuesta en tiempo real.

3. Explicación del algoritmo

Consideramos con el cálculo de la amplitud recibida un período de muestreo mayor a 2 para enviar el mensaje.
Programa: Detección y búsqueda de patrón de voz
Entorno: Amplitud es un número entero.

```
Algoritmo de detección de voz:
        recibe  “la amplitud de la señal continua”
        leer Amplitud
        calcula “cálculo del periodo recibido”
        calcular  Muestreo = Cálculo(Amplitud)
        SI Muestreo >2 ENTONCES
                EnviaAlerta(“Se detectó voz”)
        FINSI
    Finprograma
```

Para la parte de busqueda de la señal se debera comparar con el resto de las señales, como se explico anteriormente, estas son cadenas de bits. por lo tanto por cada registro existente en la base de datos se debera hacer un XOR y se enviara la cadena que haya obtenido el menor numero de cambios, ergo, la mas cercana a la recibida:

```
Algoritmo de búsqueda de similitud en secuencial:
        recibe “señal de voz”
        leer señal
        for each (registro en database)
        { resultado = XOR (registro, señal)
            if( resultado < resultado_minimo)
                resultado_minimo = resultado
        }
        EnviarOrden (resultado_minimo)
Finprograma
```

Debido a la gran cantidad de registros a analizar es un problema si se quiere llevar a cabo un sistema en tiempo real de rápida respuesta, para esto se paralelizan las búsquedas, por ejemplo, lo haremos con 6 threads, uno para cada categoría de tipo de voz, de esta manera todos tendrán la misma cantidad de comparaciones, para esto necesitaremos un procesador con 6 cores debido a las limitaciones de la directiva for propia de OpenMP

```
Algoritmo de búsqueda de similitud en paralelo:

        recibe “señal de voz”
        señal <- leer_señal()
#pragma omp parallel
 {
         #pragma omp for reduction(min:resultado_minimo)
        for each (registro en registro_database)
        {
                señalMasCercana = min (señalMasCercana, XOR(registro, señal)
        }

}
        Enviar orden (señalMasCercana)
finprograma
```

De esta manera al finalizar todos los threads se obtendrá el resultado mas cercano al solicitado por el usuario, obteniendo el resultado en menor tiempo y pudiendo el sistema responder en tiempo real de manera mas rapida de forma similar al trabajo realizado por Al-Dabbagh, Barnouti, Naser y Ali (2016) [4] en su trabajo sobre búsqueda de cadenas utilizando OpenMP

4. Pruebas que pueden realizarse

Para utilizar esta nueva característica del sistema deberemos tomar diferentes muestras de cada orden que el sistema posee, variando tanto en sexo, edad y tono de voz de los sujetos, como en distancia del micrófono, ruido ambiental, etc... para así generar una base de datos mayor, lo que permitirá realizar búsquedas de similitud más certeras.

5. Conclusiones

El trabajo realizado se llevó adelante como una investigación cuyo objetivo fue saber cómo capturar y reconocer las órdenes de los usuarios al sistema de gestión de luces.
Se encontró información acerca de cómo se convierte la voz a un dato digital y cómo hacer para encontrar una coincidencia con ese dato en un búsqueda rápida en tiempo real mediante el uso de paralelismo.
Con la información desarrollada en este trabajo planeamos implementar a futuro esta nueva característica en el sistema LiMaS.

6. Referencias

- Caldarelli, David D. and Ruth S. Campanella. Ear. World Book Online Americas Edition. 26 May 2003.
- Simancas, José & Meléndez-Pertuz, Farid. (2017). Algoritmo de reconocimiento de comandos voz basado en técnicas no-lineales. Revista Espacios. 38. 4.
- STALLINGS, WILLIAM (2004). Comunicaciones y Redes de Computadoras. Madrid: PEARSON EDUCACIÓN, S. A.
- Al-Dabbagh, Sinan Sameer Mahmood & Hazim, Nawaf & Naser, Mustafa Abdul Sahib & Ali, Zaid. (2016). Parallel Quick Search Algorithm for the Exact String Matching Problem Using OpenMP. Journal of Computer and Communications. 04. 1-11. 10.4236/jcc.2016.413001.
- David Reig Albiñana (2014). Implementación de algoritmos para la extracción de patrones característicos en Sistemas de Reconocimiento De Voz en Matlab. Gandia, Valencia: Universidad Politécnica de Valencia
- Coffin, Berton (2002). Historical Vocal Pedagogy Classics. Scarecrow Press, Inc.