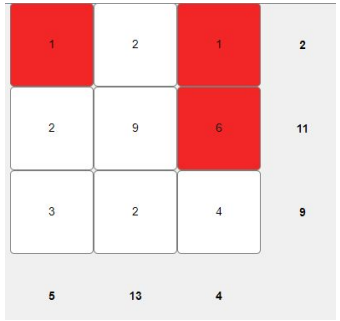
Sumplete



Agustín Harispe Lucarelli
Mikolaj Witt
Alejandro García Ramos

2. Herramientas y tecnologías utilizadas

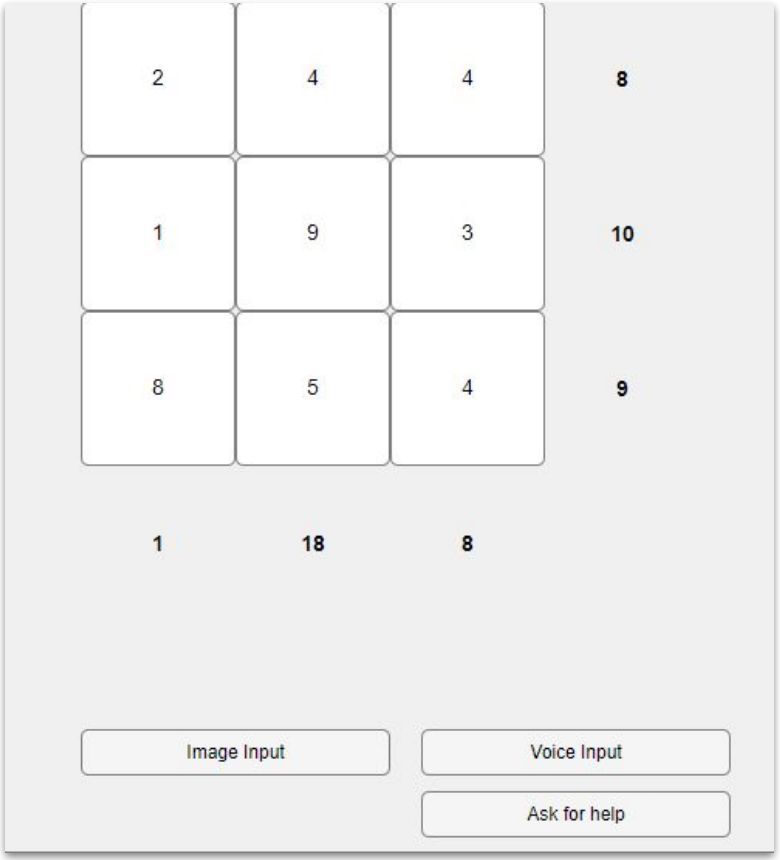
- **MATLAB:**
  El juego fue desarrollado utilizando el entorno de programación de MATLAB, que es ideal para tareas de procesamiento numérico y creación de interfaces gráficas.
- **Audio y Reconocimiento de Voz:**
  Se implementó un sistema para capturar entradas de voz para el procesamiento de audio y un modelo de reconocimiento de voz usando Modelos Ocultos de Markov.
- **Interfaz Gráfica (UI):**
  Se diseñó una interfaz visual utilizando las funciones de `uifigure`, `uibutton`, y `uilabel` para interactuar con el jugador.

Elección de tablero

El tablero del juego se puede generar en diferentes tamaños (3x3, ..., 9x9) con el botón correspondiente. El tablero es generado aleatoriamente y se muestra en la interfaz gráfica o se puede leer uno desde la webcam.

El juego

- **Interacción con la cuadrícula:**
  - El jugador puede hacer clic en las celdas para marcar o desmarcar una casilla. Al hacer una marca, se verifica que la acción sea válida y se actualiza el tablero.
- **Entrada por Webcam:**
  - Una de las características innovadoras es la opción de ingresar el tablero mediante una imagen capturada por una webcam. El tablero es detectado automáticamente y el jugador puede aceptar o rehacer la lectura si es necesario.
- **Entrada por Voz:**
  - Utilizando un modelo HMM y la captura de audio, el jugador puede indicar la fila y la columna de la casilla que desea marcar. El sistema identifica los números pronunciados y actualiza la interfaz del juego.
- **Ayuda:**
  - El jugador puede pedir ayuda, lo que hace que el sistema sugiera una casilla incorrecta que debe ser marcada, para guiar al jugador.



Técnicas utilizadas

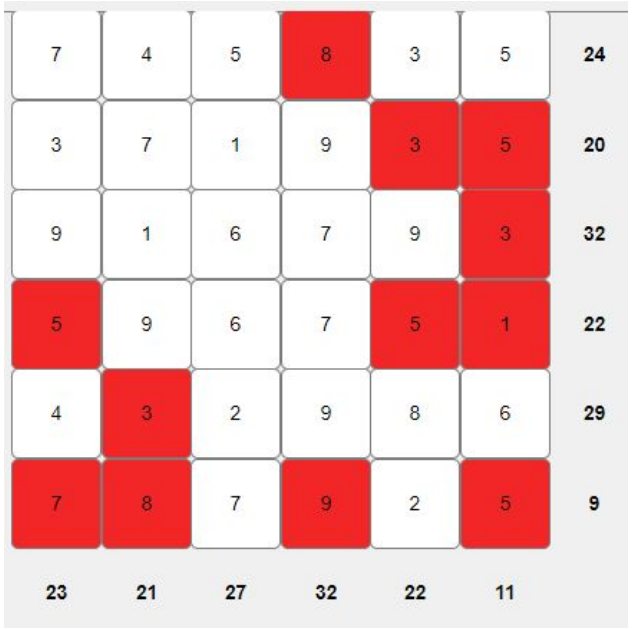
Generación de tablero aleatorio

Creación del tablero: Se genera un tablero de tamaño n×nn \times nn×n con números aleatorios entre 1 y 9.

Selección de celdas para borrar: Se selecciona aleatoriamente un tercio de las celdas del tablero y se ponen en 0.

Cálculo de las sumas: Se calculan las sumas de las filas y columnas del tablero modificado.

Construcción del tablero final: El tablero final tiene las sumas en la última fila y columna, formando un tablero de tamaño (n+1)×(n+1)(n+1) \times (n+1)(n+1)×(n+1).



Al iniciar el juego

1.Solución del Juego y Configuración Inicial

Calcula la solución del tablero con un algoritmo genético `solveGenetic`.

Inicializa variables como el tamaño del tablero (`game_data.N`), la matriz del juego (`game_data.board`), y las celdas tachadas (`game_data.crossed`).

2.Creación del Panel del Juego

Crea un panel (`game_data.game_panel`) donde se dibuja el tablero.

Calcula el tamaño de cada celda para ajustar el tablero en la ventana.

3.Creación de Botones del Tablero

Crea un botón para cada celda que muestra el número correspondiente y cambia de estado (tachado/no tachado) al hacer clic.

5. Sumas de Filas y Columnas

Añade etiquetas a la derecha de cada fila y debajo de cada columna con las sumas objetivo.

6. Botones de Entrada de Datos

Se añaden botones para:

Image Input: Usar la cámara para reconocer un tablero físico.

Voice Input: Introducir valores mediante voz.

Ask for Help: Proporcionar pistas al jugador.

7. Función `cellClicked(row, col)`

Cambia el estado de una celda al hacer clic.

Actualiza el tablero visualmente y verifica si el jugador ha ganado.

Solucion Genética y botón ayuda

Entrada:

- La función `solveGenetic(matrix)` recibe una matriz `matrix`, donde las últimas fila y columna contienen las sumas objetivo de las filas y columnas, respectivamente. La parte restante de la matriz representa el tablero de juego.

Configuración del algoritmo genético:

- Se define un algoritmo genético que utiliza una población de 3000 individuos representados como cadenas de bits. Cada individuo es una posible solución al problema.
- El algoritmo se ejecuta durante un máximo de 200 generaciones y puede detenerse antes si no hay mejoras significativas en 150 generaciones consecutivas.

Función de fitness:

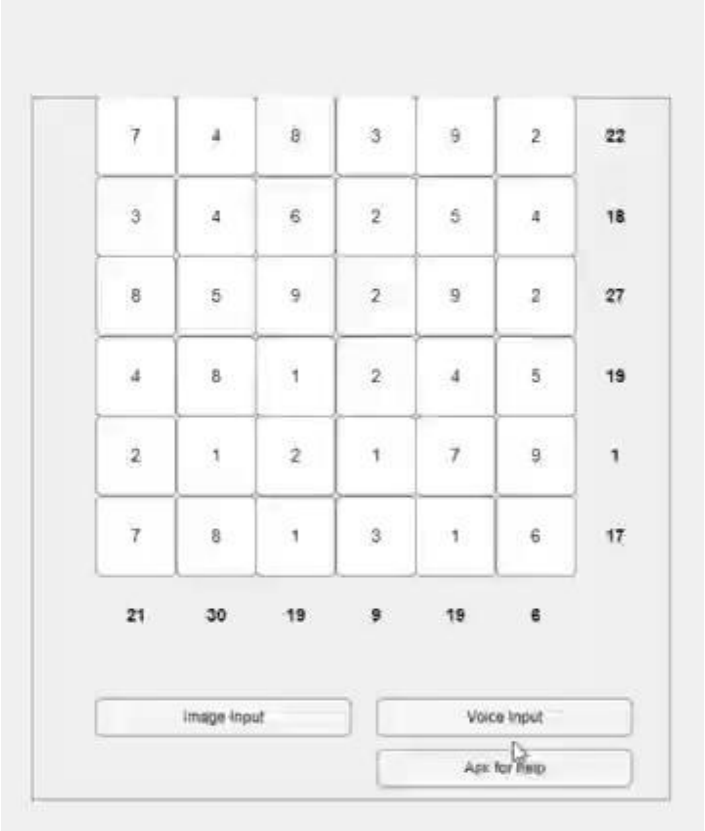
- Se evalúa qué tan buena es una solución basada en las sumas de filas y columnas
- Cuanto más coincidan las sumas de las filas y las columnas de la solución con las sumas objetivo, mejor es la solución, lo que da una puntuación de fitness más alta.

Búsqueda de la solución:

- El algoritmo genético busca minimizar la diferencia entre las sumas actuales y las sumas objetivo a través de selección, cruce y mutación de las soluciones en la población.

Resultado:

- El algoritmo devuelve la mejor solución encontrada en forma de una matriz del tamaño adecuado (N x N) y verifica si cumple con las condiciones de victoria, es decir, si las sumas de filas y columnas coinciden exactamente con las sumas objetivo.

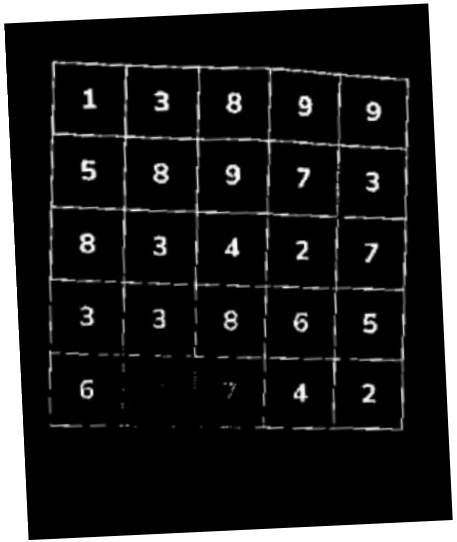


Reconocimiento de tablero

Captura de la imagen: Se accede a la cámara web disponible y se captura una imagen en escala de grises.

Preprocesamiento:

- Se convierte la imagen a formato binario invertido (para hacer el tablero positivo).
- Se aplica una dilatación utilizando un elemento estructurante de 5x5.
- Se eliminan áreas pequeñas en la imagen.
- Se extraen todas las áreas conexas con regionprops().



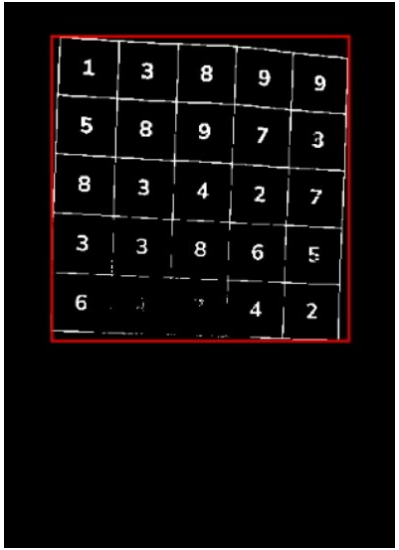
Detección de la Cuadrícula

Con la imagen binarizada y preprocesada, se procede a detectar la cuadrícula del tablero utilizando la función **encuentra_cuadricula()**. Esta función devuelve las coordenadas de un cuadro delimitador (BoundingBox) del prop que marca el área que contiene el tablero de la siguiente manera:

- **Se verifica que el boundingbox sea un cuadrado.**
- **Se verifica que sea más grande que el 5% de la imagen.**
- **Se verifica que contenga otros objetos dentro de ella.**

Si el cuadro delimitador es válido, el código ajusta los márgenes del rectángulo

para incluir los números exteriores y asegurar que la imagen esté correctamente centrada.



Corrección de la imagen

Corrección de la Inclinación de la Imagen
Si la imagen no está perfectamente alineada, se utiliza la Transformada de Hough para detectar las líneas presentes en la imagen. A partir de las líneas detectadas, se calcula el ángulo de inclinación y se rota la imagen para corregirla.

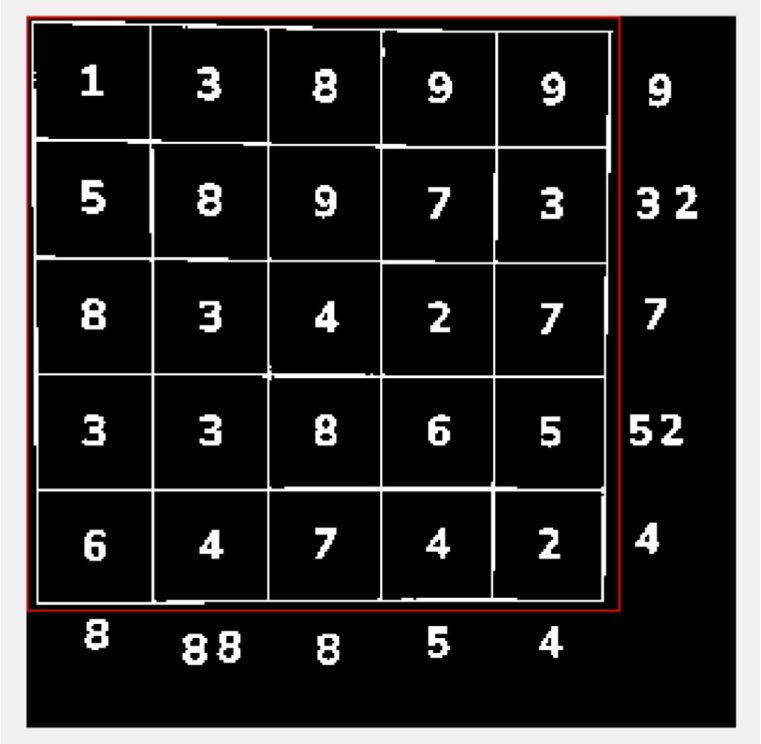
Recorte de la Imagen

Una vez rotada se vuelve a buscar la cuadrícula

(esta vez al ser la imagen más centrada la binarización es mucho mejor)

Luego se vuelve a recortar la imagen añadiendo un margen que depende del número de props(posibles números) que haya dentro de la cuadrícula.

(de ⅓ a 1/9 de la imagen)



Detección de los números

La imagen recortada se binariza nuevamente y se limpia de ruido . Se identifican las posiciones de los números en el tablero, ordenándolos por sus coordenadas X (para saber que número va antes y después en los de dos cifras)

La función **detectarNumeros** utiliza correlación para reconocer los números y solo devuelve aquellos que coinciden por encima de un umbral.

Verificación del Tamaño del Tablero
El código verifica que el número de números detectados dentro de la cuadrícula forme un cuadrado perfecto . Si no, el proceso se reinicia (falta algún número).

Asignación de los Números al Tablero

1. Cada número detectado se coloca en su posición correspondiente en la matriz del tablero.
2. Si un número corresponde a una posición ocupada, se añade (para números de dos cifras).
3. La posición se calcula teniendo en cuenta la posición del número y la dimensión de la cuadrícula.
4. Si todos los números se asignan correctamente, se emite un sonido de confirmación y el proceso termina.

Reconocimiento de número individual

• Inicializa la cámara web y establece su resolución maxima.

• Captura una imagen en escala de grises.

• Realiza el preprocesamiento de la imagen (binarización, eliminación de pequeñas áreas, dilatación).

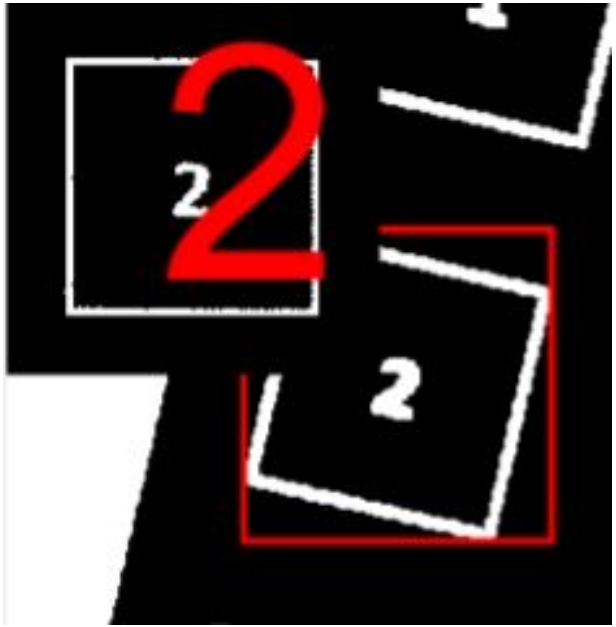
• Detecta la cuadrícula que contiene el número (se elige teniendo en cuenta el tamaño de la bbox y el área de esta)

• Si se encuentra la cuadrícula, recorta la imagen y corrige su orientación con la transformada de Hough.

• Recorta nuevamente la imagen para ajustarla y eliminar márgenes.

• Detecta el número en la imagen procesada.

• Si se detecta un número, lo devuelve y emite un sonido.



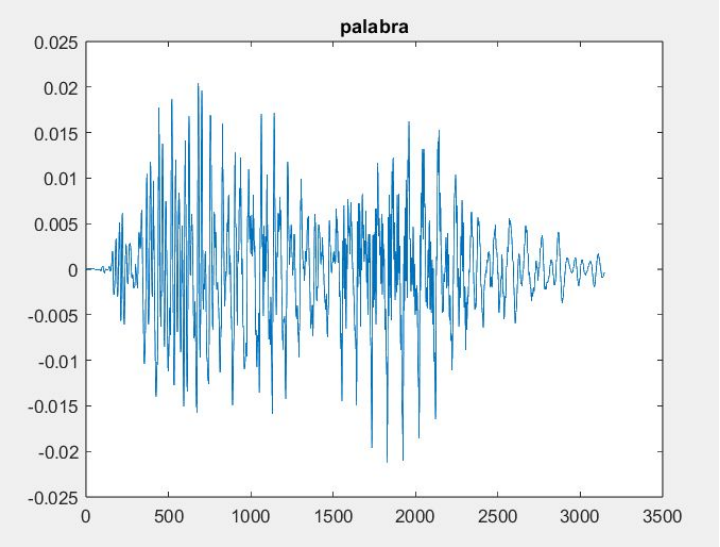
Dataset de audio y procesado

Conjunto de datos:

- Para reconocer los dígitos se ha generado inicialmente un dataset de cada número (unas 50 muestras por número)

Preprocesado:

- Se limpia y filtra cada muestra, eliminando inicio y fin vacíos, aplicando preénfasis y re-muestreando a una menor frecuencia (8 KHz)
- Se cuantifica cada muestra quedando en varias tramas de 40 características: 1 Energía + 13 MelCepstrum + 13 Delta + 13 Delta-Delta.
- Cada palabra se guardada en conjunto con los de su misma clase para la creación de los codebooks y por separado para el posterior uso en el entrenamiento.



Entrenamiento HMM

Hiper-Parámetros

- Se generaron distintas variantes de los modelos dependiendo del número de estados a transicionar (N = [3, 4, 5, 6]) y de la definición de nuestro codebook (número de centroides, K = [64 128 256];

0.8907	0.1093	0
0	0.8720	0.1280
0	0	1.0000

El modelo:

- Con cada muestra se estima una matriz de transiciones y otra de emisiones (A y B) donde la media de los mismos es el modelo para ese dígito.
- El modelo total consta del conjunto de todos los modelos generados junto con sus codebooks correspondientes; para cada dígito, cada N y cada K.

Reconocimiento de voz

Funciones

La implementación del reconocimiento para el juego se encapsuló en dos funciones, la que recoge la entrada, y la reconoce el número

Entrada de audio

- La función de entrada pide la muestra al usuario y avisa del proceso por pantalla.

Estimación

- Teniendo todos los modelos cargados, la función de estimación realiza *hmmdecode* con la secuencia y a partir del audio recogido calcula la moda del máximo de cada modelo generando una estimación fiable

```
Hable en 1 segundo...
Grabando...
Fin grabación
```

Más muestras

Al inicio la cantidad de muestras eran limitadas y la validación de los modelos se hacía manual desde las funciones de entrada de audio de Matlab simulando el uso de la aplicación.

Aprovechamos este proceso para recoger esas mismas muestras, asignarlas a una clase y reentrenar los modelos posteriormente. Con el uso, se obtuvieron más y mejores resultados

Gracias por su atención