



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

SISTEMA DE CONTROL DE TELEVISIÓN MEDIANTE KINECT Y REDES NEURONALES

TESIS

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTAN:

**MARÍA NELLY ESPINOSA VÁZQUEZ
ANA SILVIA MARTÍNEZ HERNÁNDEZ**

ASESOR:

M. EN C. BEATRIZ ADRIANA JAIME FONSECA



CIUDAD DE MÉXICO, DICIEMBRE DE 2016

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL “ADOLFO LÓPEZ MATEOS”

TEMA DE TESIS

**QUE PARA OBTENER EL TÍTULO DE
POR LA OPCIÓN DE TITULACIÓN
DEBERÁ (N) DESARROLLAR**

**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA
TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL
C. MARIA NELLY ESPINOSA VAZQUEZ
C. ANA SILVIA MARTINEZ HERNANDEZ**

“SISTEMA DE CONTROL DE TELEVISIÓN MEDIANTE KINECT Y REDES NEURONALES”

DISEÑAR Y CONSTRUIR UN SISTEMA DE CONTROL DE TELEVISIÓN, MEDIANTE MOVIMIENTOS DETECTADOS POR KINECT Y RECONOCIDOS POR UNA RED NEURONAL

- ❖ INTRODUCCIÓN
- ❖ ANTECEDENTES
- ❖ TÉCNICAS Y HERRAMIENTAS
- ❖ DESARROLLO DEL PROTOTIPO
- ❖ RESULTADOS EXPERIMENTALES
- ❖ CONCLUSIONES

CIUDAD DE MÉXICO, A 20 DE DICIEMBRE DE 2016.

ASESOR

M. EN C. BEATRIZ ADRIANA
JAIME FONSECA

*Ramírez
Lorena*



ING. PATRICIA LORENA RAMÍREZ RANGEL
JEFA DEL DEPARTAMENTO DE INGENIERÍA EN
COMUNICACIONES Y ELECTRÓNICA

Con amor y cariño a mis padres, mi esposo y mis hijos.

Nelly Espinosa

A Silviano Martínez Hernández y Juana Hernández Estefes

Silvia Martínez

Agradecimientos

Con estas líneas quiero agradecer sinceramente a mi padre por hacer el sacrificio de vivir separados, por darme los recursos y las armas para poder concluir mis estudios.

A mi madre por darme las palabras de aliento que me hicieron falta levantarme después de caer.

Agradezco de manera muy especial a mi esposo por creer en mí cuando ni yo misma lo hice, por la constancia y el esfuerzo de estar en todo momento a mi lado.

A mis hijos les agradezco darme la fortaleza de no desistir, de lograr una meta.

María Nelly Espinosa Vázquez
Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco
Instituto Politécnico Nacional, México, 2016

A mis padres, Silviano Martínez y Juana Hernández, por todo el apoyo y amor incondicional que me han brindado. Agradezco que siempre se han preocupado por mi bienestar y educación. Porque gracias a su ejemplo de lucha constante y determinación he podido alcanzar mis sueños y metas.

A mi hermana, Esther Martínez, por ser mi cómplice y confidente. Gracias por hacerme reír con ese sentido del humor que te caracteriza y por ayudarme cuando lo necesito.

A mi novio, Jesús Arenas, por todo el amor y comprensión que me has dado. Por darme ánimos cuando lo necesito y creer en mí en todo momento.

A mis amigos y compañeros de la ESIME, Luciano López, Joshua Romero, Iris López, Mariana Hernández, Nelly Espinosa, Marcos Patiño y Daniel Cubos, por su confianza y por hacer de mi estancia en la ESIME una experiencia inolvidable.

A la M. en C. Beatriz Adriana Jaime Fonseca y el Dr. Jesús Jaime Moreno Escobar, por guiarnos en la realización de este proyecto. Gracias por las enseñanzas, consejos y apoyo brindado.

Y finalmente al Instituto Politécnico Nacional por brindarme las herramientas necesarias para mi formación profesional.

Ana Silvia Martínez Hernández
Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Zacatenco
Instituto Politécnico Nacional, México, 2016

Resumen

Actualmente el uso de control remoto para televisor se ha vuelto escencial, debido a que algunas televisiones modernas ya no tienen integrados los botones para controlar sus funciones básicas. Por lo tanto, si el control remoto se extravía o deja de funcionar, el usuario queda inhabilitado para emplear el televisor. Es por ello que en la última década se han realizado diversas investigaciones para encontrar una alternativa al uso del control remoto infrarrojo. Sin embargo, la mayoría de estos trabajos se enfocan en diseñar una interfaz para interactuar con televisores inteligentes.

En este trabajo se propone un sistema para controlar tres funciones básicas de un televisor mediante movimientos con los brazos. Las funciones que controla el sistema son: encender/apagar, subir/bajar volumen y cambiar canal. Los movimientos para controlar dichas funciones se capturan mediante el sensor Kinect y se reconocen mediante una Red Neuronal Artificial (RNA) de retropropagación. El resultado del reconocimiento de los movimientos se envía por medio de comunicación serial a un dispositivo Arduino UNO para que éste a su vez, envíe el comando necesario por medio de una señal infrarroja al televisor. Además, el sistema cuenta con una interfaz gráfica de usuario que se visualiza en la computadora y sirve para que el usuario active o desactive el sistema cuando lo necesite.

Para el entrenamiento de la RNA se construyó una base de datos que almacena información de cinco movimientos asignados para el control del televisor. La información de la base de datos contiene muestras de movimientos realizados por personas con distintas complejiones. Cada muestra contiene la posición de las articulaciones del usuario durante un movimiento. Estos datos se obtienen mediante Kinect debido a que cuenta con un sensor de profundidad que rastrea el esqueleto de una persona en movimiento.

El sistema de control gestual propuesto en este trabajo está probado en diferentes regiones del campo visual del sensor Kinect y tiene un porcentaje de efectividad del 91.55%. Este porcentaje incluye la correcta detección de los gestos definidos para el control del televisor y la capacidad para descartar los gestos que no están definidos para el sistema. Las pruebas fueron realizadas en un televisor LG modelo 32LF510B, pero el sistema puede adaptarse fácilmente a otro modelo de televisor que se controle mediante señales de infrarrojo.

Glosario

| | |
|---------------------|--|
| Acelerómetro | Dispositivo que realiza una medida de aceleración o vibración según la variación física. |
| Clase | Conjunto de entidades que comparten alguna característica que las diferencia del resto. |
| Gesto | Movimiento corporal (especialmente manos y rostro) realizado para expresar alguna idea o interactuar con el medio. |
| GUI | <i>Graphical User Interface.</i> Interfaz Gráfica de Usuario. Permite el uso de iconos u otros indicadores visuales para interactuar con la computadora, en vez de usar sólo texto en una línea de comandos. |
| IA | Inteligencia Artificial. Máquina o conjunto de programas computacionales que realizan tareas basándose en el aprendizaje humano y el comportamiento inteligente. |
| IDE | <i>Integrated Development Environment.</i> Ambiente de Desarrollo Integrado. Programa informático que brinda las herramientas al desarrollador o programador para diseñar software. |
| INEGI | Instituto Nacional de Estadística, Geografía e Informática. Organismo del gobierno mexicano a cargo del Sistema Nacional de Información Estadística y Geográfica del país. |
| Infrarrojo | Señal del espectro luminoso con una longitud de onda en el intervalo de 0.7 a 1000 micrómetros. |

GLOSARIO

| | |
|------------------|---|
| Kinect | Sensor creado por Microsoft para interactuar con la consola Xbox 360 por medio de gestos y voz. |
| MODUTIH | Módulo Sobre Disponibilidad y Uso de las Tecnologías de la Información en los Hogares. Es una encuesta cuyo objetivo es generar información estadística que permita conocer la disponibilidad y el uso de las tecnologías de información y comunicaciones en los hogares y por los individuos de seis y más años de edad en México. |
| Patrón | Conjunto de atributos que describen a una entidad. |
| RNA | Red Neuronal Artificial. Modelo que simula el comportamiento de una red neuronal biológica por medio un algoritmo matemático. |
| SDK | <i>Software Development Kit.</i> Kit de Desarrollo de Software. Conjunto de herramientas de desarrollo de software que le permite al programador o desarrollador de software crear aplicaciones para un sistema concreto. |
| Teletexto | Información en forma de texto que se envía junto con la señal de televisión. |

Contenido

| | |
|---|-------------|
| Lista de Figuras | xi |
| Lista de Tablas | xiii |
| 1 Introducción | 1 |
| 1.1 Planteamiento del problema | 1 |
| 1.2 Justificación | 1 |
| 1.3 Hipótesis | 3 |
| 1.4 Objetivos | 3 |
| 1.4.1 General | 3 |
| 1.4.2 Particulares | 3 |
| 1.5 Alcance del proyecto | 4 |
| 2 Antecedentes | 5 |
| 2.1 Control remoto para televisor | 5 |
| 2.1.1 Historia | 5 |
| 2.1.2 Componentes principales | 6 |
| 2.1.3 Funcionamiento | 6 |
| 2.2 Controles gestuales para televisor | 8 |
| 2.2.1 Primera televisión controlada por gestos | 8 |
| 2.2.2 Sistema de interacción de una <i>Smart TV</i> basado en reconocimiento de gestos con manos | 10 |
| 2.2.3 Reconocimiento del estado de un usuario para un sistema de control de televisión de bajo costo basado en gestos | 12 |
| 2.2.4 <i>Samsung Smart Interaction</i> | 13 |
| 2.2.5 Diseño de una interfaz de usuario con integración de gestos dinámicos y reconocimiento de números manuscritos | 15 |
| 2.2.6 Control remoto con reconocimiento de gestos basado en acelerómetro para interacción con la TV digital | 17 |
| 2.3 Aplicaciones de las redes neuronales de retropropagación | 18 |
| 3 Técnicas y Herramientas | 19 |
| 3.1 Introducción | 19 |
| 3.2 Kinect | 19 |
| 3.2.1 Generalidades y componentes | 19 |
| 3.2.2 Especificaciones | 20 |
| 3.2.3 Funcionamiento | 22 |
| 3.2.4 Kit de Desarrollo de Software | 22 |

CONTENIDO

| | | |
|----------|--|-----------|
| 3.2.5 | Flujo de datos | 24 |
| 3.3 | Redes Neuronales | 27 |
| 3.3.1 | Neuronas biológicas | 27 |
| 3.3.2 | Neuronas artificiales | 28 |
| 3.3.3 | Perceptrón Multicapa | 29 |
| 3.3.4 | Algoritmo de Retroproyagación | 31 |
| 3.3.5 | Pasos del proceso de aprendizaje | 32 |
| 3.4 | MatLab | 33 |
| 3.4.1 | Toolbox de adquisición de imágenes | 33 |
| 3.4.2 | Toolbox de redes neuronales | 34 |
| 3.4.3 | Entorno para el diseño de interfaces gráficas de usuario | 35 |
| 3.4.4 | Comunicación serial | 37 |
| 3.5 | Arduino | 37 |
| 3.6 | Protocolos de comunicación para el control remoto infrarrojo | 38 |
| 3.6.1 | Protocolo de comunicación por infrarrojo <i>SONY Serial InfraRed Control</i> | 39 |
| 3.6.2 | Protocolo de comunicación por infrarrojo Philips RC-5 | 41 |
| 4 | Desarrollo del prototipo | 45 |
| 4.1 | Introducción | 45 |
| 4.2 | Creación de una base de datos | 46 |
| 4.2.1 | Delimitación de los datos | 46 |
| 4.2.2 | Adquisición de datos | 49 |
| 4.2.3 | Selección de muestreos útiles | 51 |
| 4.2.4 | Extracción de las coordenadas <i>xy</i> | 53 |
| 4.2.5 | Bases de datos para entrenamiento y simulación | 54 |
| 4.3 | Creación de una red neuronal artificial | 55 |
| 4.3.1 | Configuración inicial | 55 |
| 4.3.2 | Fase de entrenamiento | 57 |
| 4.3.3 | Fase de simulación o recuperación | 59 |
| 4.4 | Diseño de sistema de control gestual para televisor | 61 |
| 4.4.1 | Diseño de una interfaz gráfica de usuario | 61 |
| 4.4.2 | Algoritmo del sistema de control gestual para televisor | 63 |
| 5 | Resultados Experimentales | 73 |
| 5.1 | Introducción | 73 |
| 5.2 | Definición de los variables y experimentos | 73 |
| 5.3 | Resultados | 74 |
| 6 | Conclusiones | 77 |
| 6.1 | Conclusiones Finales | 77 |
| 6.2 | Contribuciones | 77 |
| 6.3 | Trabajo Futuro | 78 |
| A | Código | 79 |
| A.1 | Creación de la base de datos | 79 |
| A.1.1 | Captura | 79 |
| A.1.2 | Revisión | 80 |
| A.1.3 | Extracción de coordenadas | 80 |
| A.2 | Interfaz | 81 |
| A.2.1 | Menú principal | 81 |
| A.2.2 | Instrucciones | 85 |
| A.3 | Arduino | 87 |

CONTENIDO

| | |
|---|------------|
| B Entrenamiento y simulación de redes neuronales | 91 |
| B.1 Resultados de la fase de entrenamiento | 91 |
| B.2 Resultados de la fase de simulación | 102 |
| C Análisis de costos | 111 |
| C.1 Material y Herramientas | 111 |
| C.2 Desarrollo | 112 |
| Referencias | 113 |

Listado de Figuras

| | | |
|------|--|----|
| 1.1 | Sistema de control gestual | 3 |
| 2.1 | Botones de un control remoto | 6 |
| 2.2 | Pulso generado por control de TV | 7 |
| 2.3 | Protocolo RC-5 | 7 |
| 2.4 | Menú de primer control gestual de TV | 9 |
| 2.5 | Método de correlación | 9 |
| 2.6 | Muestra de reconocimiento de gestos | 10 |
| 2.7 | Reconocimiento de gestos estáticos | 11 |
| 2.8 | Reconocimiento de gestos dinámicos | 12 |
| 2.9 | Reconocimiento de gesto "click" | 12 |
| 2.10 | Detección de manos | 13 |
| 2.11 | Control gestual de <i>Smart TV</i> | 14 |
| 2.12 | Gestos y sistema de coordenadas | 18 |
| 3.1 | Kinect para Xbox 360 | 20 |
| 3.2 | Componentes del Kinect | 21 |
| 3.3 | Patrón de puntos infrarrojos | 22 |
| 3.4 | Laptop Acer modelo Aspire E15 E5-522-86AU | 23 |
| 3.5 | Imagen de profundidad obtenida con el sensor Kinect | 25 |
| 3.6 | Articulaciones del esqueleto | 26 |
| 3.7 | Ejes del sistema de coordenadas de Kinect | 26 |
| 3.8 | Neurona biológica | 27 |
| 3.9 | Neurona artificial | 28 |
| 3.10 | Estructura de una red neuronal artificial | 29 |
| 3.11 | Entorno de desarrollo de GUI | 36 |
| 3.12 | Tarjeta Arduino UNO | 37 |
| 3.13 | Bits que conforman las diferentes versiones del protocolo SONY SIRC | 39 |
| 3.14 | Temporización de los bits que conforman las diferentes versiones del protocolo SONY SIRC | 40 |
| 3.15 | Temporizador de receptor del protocolo SONY SIRC | 40 |
| 3.16 | Estado lógico '0' en protocolo RC-5 | 42 |
| 3.17 | Estado lógico '1' en protocolo RC-5 | 42 |
| 3.18 | Tren de pulsos de protocolo RC-5 | 42 |
| 4.1 | Sistema de control de televisión mediante gestos | 45 |
| 4.2 | Gestos para controlar el televisor | 47 |
| 4.3 | Arquitectura de la red neuronal artificial | 48 |
| 4.4 | Algoritmo para adquisición de datos con Kinect | 50 |

LISTA DE FIGURAS

| | | |
|------|---|-----|
| 4.5 | Algoritmo para la revisión de muestras de la base de datos. | 51 |
| 4.6 | Fotogramas útiles para la construcción de la base de datos. | 52 |
| 4.7 | Construcción de base de datos de fotogramas útiles. | 52 |
| 4.8 | Archivo de la base de datos total. | 53 |
| 4.9 | Algoritmo exportar base de datos de MatLab a Excel. | 54 |
| 4.10 | Archivo de las clases de la base de datos. | 55 |
| 4.11 | <i>Neural Network/Data Manager</i> de la herramienta <i>nntool</i> | 56 |
| 4.12 | Ventana para importar bases de datos a <i>nntool</i> | 56 |
| 4.13 | Creación de una red neuronal artificial. | 57 |
| 4.14 | Arquitectura de la red neuronal artificial. | 57 |
| 4.15 | Ventana para entrenamiento de la red neuronal artificial. | 58 |
| 4.16 | Configuración de parámetros de entrenamiento. | 58 |
| 4.17 | Resultado del entrenamiento de la red neuronal artificial. | 59 |
| 4.18 | Ventana <i>Simulate</i> para la simulación de la red neuronal. | 60 |
| 4.19 | Interfaz gráfica del sistema de control gestual para televisor. | 61 |
| 4.20 | Vista del Kinect. | 62 |
| 4.21 | Diferentes casos de la sección "Función detectada" de la interfaz de usuario. | 63 |
| 4.22 | Instructivo de gestos para controlar TV. | 63 |
| 4.23 | Algoritmo del sistema de control gestual para televisor. | 64 |
| 4.24 | Mensaje de error cuando el Kinect no se encuentra conectado. | 65 |
| 4.25 | Circuito receptor de códigos de un control remoto. | 68 |
| 4.26 | Obtención de los comandos de un control remoto. | 69 |
| 4.27 | Circuito para el control de un televisor. | 70 |
| 5.1 | Posiciones del usuario consideradas para los experimentos. | 74 |
| 5.2 | Porcentajes de reconocimiento para siete gestos. | 75 |
| 5.3 | Porcentajes de reconocimiento en diferentes distancia al sensor. | 76 |
| 5.4 | Porcentajes de reconocimiento en diferentes posiciones del eje x. | 76 |
| B.1 | Entrenamiento 1. | 92 |
| B.2 | Entrenamiento 2. | 93 |
| B.3 | Entrenamiento 3. | 94 |
| B.4 | Entrenamiento 4. | 95 |
| B.5 | Entrenamiento 5. | 96 |
| B.6 | Entrenamiento 6. | 97 |
| B.7 | Entrenamiento 7. | 98 |
| B.8 | Entrenamiento 8. | 99 |
| B.9 | Entrenamiento 9. | 100 |
| B.10 | Entrenamiento 10. | 101 |

Lista de Tablas

| | | |
|-----|---|-----|
| 2.1 | Guía de gestos del <i>Smart Interaction</i> | 16 |
| 2.2 | Reglas de escritura de números con gestos | 17 |
| 3.1 | Especificaciones del Kinect Xbox 360 | 21 |
| 3.2 | Propiedades del sensor de profundidad configurables con MatLab. | 35 |
| 3.3 | Comandos predefinidos del protocolo SIRC. | 41 |
| 3.4 | Comandos predefinidos del protocolo RC-5. | 43 |
| 4.1 | Definición de clases | 49 |
| 5.1 | Número de gestos reconocidos en diferentes posiciones. | 75 |
| B.1 | Resultados de la simulación de la red neuronal. | 102 |

Capítulo 1

Introducción

1.1 Planteamiento del problema

Actualmente, el control remoto se ha convertido en un dispositivo esencial para hacer uso de los televisores. Algunos nuevos modelos de televisores ya no cuentan con botones integrados en el dispositivo en sí, es necesario el uso de un control remoto para acceder a sus funciones.

El control remoto infrarrojo de televisión ofrece al usuario la comodidad de controlar la TV a distancia, sin embargo presenta una serie de desventajas que pueden impedir una experiencia satisfactoria a la hora de ver televisión y que varían de acuerdo a las necesidades de cada televíidente. Entre las principales desventajas de los controles remotos por infrarrojo destacan:

- En ocasiones, el control remoto no se encuentra a la mano, ya sea porque el usuario olvidó donde lo dejó por última vez o porque alguien más lo colocó en un lugar que no está a la vista. De esta manera, el televisor queda inhabilitado para su uso.
- Cuando un control tiene la batería muy baja o cuando se acaba la batería y no se cuenta con repuesto es difícil tener acceso a las funciones del televisor.
- Para las personas que tienen problemas de la vista es complicado distinguir de forma adecuada todos los botones del control remoto, por consiguiente no se localizan de manera efectiva las funciones que se necesitan.

1.2 Justificación

Aunque el uso de la computadora y el Internet en la sociedad va en aumento, la televisión sigue siendo el principal medio de comunicación por el cual las personas se informan y entretienen (11). De acuerdo al MODUTIH-2014 realizado por el INEGI, en México el 38.3% de los hogares cuentan con una computadora y el

1. INTRODUCCIÓN

34.4% tienen conexión a Internet; por otro lado reporta que el 95% de los hogares mexicanos cuentan con televisor, y de esa proporción el 31% cuenta con uno de tipo digital, dicha cifra representa un aumento del 15.3% con respecto al año 2013.

El control remoto le permite al televidente controlar desde la comodidad de su asiento las funciones del televisor; sin embargo con el aumento de las tecnologías inteligentes se buscan nuevas formas de controlar el televisor. Es necesario que el control remoto evolucione a un tipo de control que le permita al usuario interactuar con el televisor de manera más natural, fluida e intuitiva.

Actualmente existen diversos sustitutos del control remoto, como lo son el control vía teléfono inteligente, reloj inteligente, comandos de voz, control mediante movimientos y gestos, etc. En el caso de los teléfonos inteligentes y relojes inteligentes tienen la desventaja de que el usuario debe sostenerlos o portarlos cada vez que requiera ver la televisión y al igual que el control remoto, pueden no estar al alcance o no tener batería. Para el control mediante comandos de voz, existen algunos factores como el ruido del ambiente que impiden su correcto funcionamiento. Los controles gestuales emplean para la detección de movimientos distintos dispositivos como lo son acelerómetros, Kinect, Leap Motion y webcams. Los acelerómetros presentan las mismas desventajas que los teléfonos y relojes inteligentes, en cambio los sensores como Kinect y Leap Motion se mantienen fijos junto al televisor.

La desventaja de los controles inteligentes actuales es que no están diseñados para todo tipo de televisores, principalmente funcionan para televisores inteligentes (*Smart TV*) y desafortunadamente sólo una minoría cuenta con ese tipo de dispositivos. Además son costosos, pues una *Smart TV* con control vía voz y gestos rebasa el precio de 39, 999 pesos mexicanos (2).

El sensor Kinect de Microsoft es un dispositivo que detecta movimientos y proporciona la posición en coordenadas de las articulaciones del esqueleto humano. Con la creciente comercialización del sensor Kinect, se han diseñado diversas aplicaciones que son controladas mediante el reconocimiento de movimientos corporales, su uso se extiende desde el control de videojuegos hasta una herramienta para supervisar terapias médicas. (4)

Con la implementación de un control de TV mediante Kinect, se pueden ejecutar funciones básicas de un televisor con movimientos corporales simples y fáciles de aprender. Las ventajas del uso del Kinect comparado con el control remoto son: el Kinect siempre está al alcance del usuario pues se mantiene fijo junto al televisor; se pueden ejecutar las funciones del televisor sin la necesidad de quitar la vista del mismo. Adicionalmente es favorable para las personas que por diversas razones, no se pueden desplazar dentro de la habitación para buscar un control remoto.

1.3 Hipótesis

Este trabajo propone la implementación de las funciones básicas de un televisor mediante gestos que se capturan con el sensor Kinect y son clasificados con una red neuronal artificial de retropropagación.

En la figura 1.1 se muestra el sistema general para controlar el televisor mediante gestos, la entrada es el usuario que realiza un movimiento capturado por el sensor Kinect. Los datos adquiridos por el Kinect se transfieren al paquete computacional MatLab, que mediante una red neuronal de retropropagación identifica el gesto realizado y la función que le corresponde del televisor. Finalmente un emisor de infrarrojo envía una señal al televisor para que realice la función correspondiente.

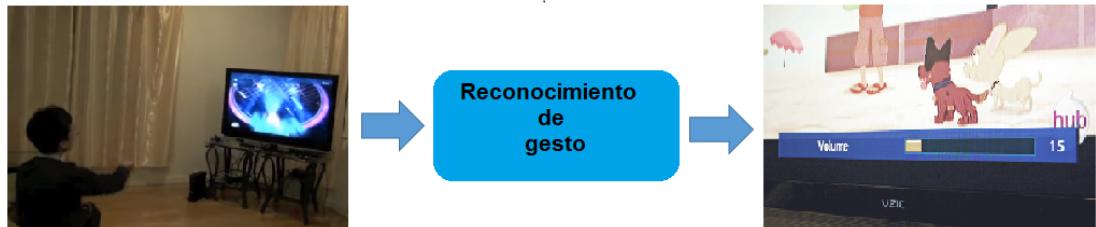


Figura 1.1: Sistema de control gestual.

1.4 Objetivos

1.4.1 General

Diseñar y construir un sistema de control de televisión, mediante movimientos detectados por Kinect y reconocidos por una Red Neuronal.

1.4.2 Particulares

- Examinar los trabajos relacionados con el sistema propuesto.
- Definir las herramientas empleadas en el reconocimiento de gestos.
- Clasificar una base de datos de gestos con una red neuronal artificial de retropropagación.
- Evaluar el funcionamiento del control gestual bajo diferentes condiciones.

1. INTRODUCCIÓN

1.5 Alcance del proyecto

Existen distintos dispositivos para el reconocimiento de gestos, que capturan el movimiento de una o varias partes del cuerpo. En base a ello cada uno entrega distintos tipos de datos para su procesamiento. El sensor Leap Motion es un dispositivo muy popular para desarrollar tecnología basada en reconocimiento de movimientos debido a su bajo costo, sin embargo sólo está limitada para la detección de movimientos con las manos. El sensor Kinect en cambio entrega información sobre el movimiento de todo el cuerpo, es por ello que se eligió para este proyecto.

El sistema propuesto en este proyecto controla tres funciones básicas que son comunes a todo tipo de televisor: subir/bajar volumen, subir/bajar canal y encender/apagar. Los movimientos empleados para ejecutar las funciones antes mencionadas son sencillos y fáciles de aprender para brindar al usuario mayor confort.

Los trabajos actuales emplean visión por computadora y redes neuronales artificiales para lograr la identificación de gestos. En este trabajo se emplea una red neuronal de *backpropagation* programada en el software MatLab. Aunque el IDE Visual Studio 2015 tiene más soporte para el desarrollo de aplicaciones con Kinect, MatLab tiene mejor soporte para redes neuronales artificiales.

Debido a la complejidad y costo de la comunicación vía Internet o Bluetooth con la televisión, en el proyecto actual se emplea comunicación por infrarrojo por su bajo costo y su compatibilidad con varios modelos de televisor. Para hacer la conexión del programa de reconocimiento de gestos con el televisor, se realiza una comunicación serial entre la computadora y un dispositivo Arduino Uno. Este último envía las señales de infrarrojo al televisor por medio de un LED infrarrojo. El sistema de control gestual propuesto en este trabajo controla un televisor LG modelo 32LF510B.

Capítulo 2

Antecedentes

2.1 Control remoto para televisor

2.1.1 Historia

El control remoto es un dispositivo electrónico portátil que sirve para operar las funciones de otro dispositivo, como la televisión, radio o reproductores de audio y video.

En 1950 Eugene Macdonald, trabajador de la Corporación de Electrónicos Zenith, desarrolló el primer control remoto para televisor llamado "Huesos Perezosos". Se conectaba a la TV por medio un cable y tenía botones que permitían rotar el sintonizador (para cambiar canal), apagar o encender la televisión.(24)

Eugene Polley de Electrónicos Zenith inventó en 1955 "*Flasmatic*", el primer control remoto para TV inalámbrico. Era como una linterna altamente direccional que activaba unas fotoceldas que se colocaban en las esquinas del televisor. Servía para encender o apagar el televisor y cambiar canales.

Robert Adler diseñó en 1956 un control remoto que usaba ondas ultrasónicas, no necesitaba baterías y se empleaba para subir canal, bajar canal, apagar o encender sonido y apagar o encender televisión. La desventaja de este control es que el receptor de la TV era muy caro.

El ultrasonido se siguió ocupando para inventar nuevos controles remotos y hasta 1980 se reemplazó por el infrarrojo. La *British Broadcasting Corporation (BBC)* encargó a varias compañías de televisores, diseñar un control remoto que integrara el teletexto y otras nuevas funciones de los televisores de la época. La compañía *International Telephone & Telegraph (ITT)*, fue la precursora del diseño de un sistema infrarrojo para transmitir comandos al televisor codificando la señal de acuerdo a un protocolo. (1)

2. ANTECEDENTES

2.1.2 Componentes principales

Las características físicas de los controles actuales, son muy similares entre sí, ya que para la mayoría de los controles se necesita el uso de baterías que pueden ser de tipo recargable o desecharable, usan infrarrojo para transmitir sus señales y tienen algunas funciones en común que se controlan por medio de botones, como se muestra en la Figura 2.1. En la actualidad estas funciones son muy variadas, y dependen de las características de cada televisor.



Figura 2.1: Funciones básicas de los botones de un control remoto (9).

2.1.3 Funcionamiento

El control remoto se comunica con el televisor de manera inalámbrica por medio de señales infrarrojas, que no son visibles. Estas señales tienen un código de comunicación entre receptor de la televisión y el control remoto que es el que envía las señales. Este código además de ejecutar las funciones del televisor, también se elabora de tal manera que el receptor pueda discriminar las interferencias. Estas interferencias provienen de diversas fuentes, tales como el sol, o hasta el cuerpo humano. Por lo tanto las señales infrarrojas que no se generaron en el control remoto deben eliminarse. Existe diferentes métodos para que el receptor discrimine estas señales, el más común es la modulación.

Se modula a una frecuencia el haz infrarrojo, de tal manera que el receptor únicamente interprete una señal pulsante dentro del rango frecuencia mediante el cual trabaja. Y las señales restantes sean desechadas.

2.1 Control remoto para televisor

El rango de frecuencias de trabajo oscila, entre los 30 a 40KHz. El inconveniente con estas frecuencias, con respecto a la cuestión electrónica, radica en la elaboración del receptor de esta señal pulsante.

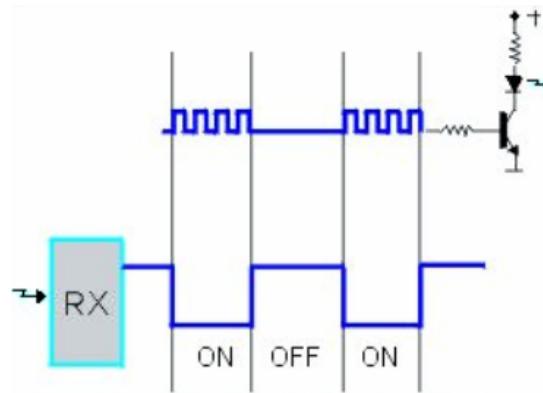


Figura 2.2: Generación de una señal pulsante del control remoto (9).

Al presionar un botón, este envía un tren de pulsos, que denota un estado lógico, este entra a la base de un transistor, para que éste controle un led infrarrojo que genera la señal pulsante Figura 2.2. El receptor trata de detectar la señal pulsante enviada por el infrarrojo y asigna el estado lógico que corresponda, para cada señal pulsante que reciba. Posteriormente se convierte en una secuencia de bits. Esta secuencia de bits se comunica dentro de un protocolo de comunicación infrarrojo, (como el RC-5 inventado por la compañía Phillips©). Este protocolo consiste en la transmisión de 14 bits en codificación Manchester a un nivel lógico alto y un 0 a un nivel lógico bajo) como se muestra en la Figura 2.3. Debido a que es una secuencia de bits, la transmisión bit a bit es de 1.776 ms por cada bit.

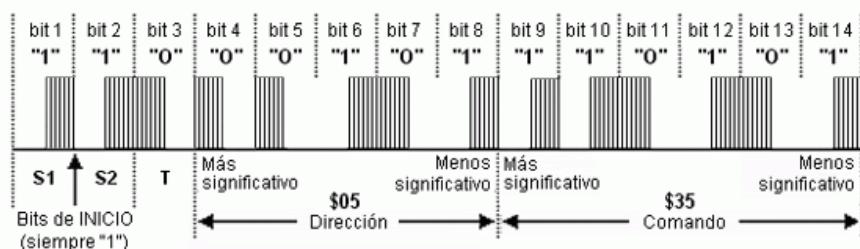


Figura 2.3: Protocolo de comunicación infrarrojo RC-5 (9).

Los primeros dos bits son para sincronizar y ajustar el receptor. El tercer bit es de control (*toggle*) y este cambia al presionar una tecla y permanece sin cambios cuando se mantiene presionado el botón del control. Los bits que van del cuatro al ocho son de direccionamiento, esto es de gran importancia ya que

2. ANTECEDENTES

variando estos bits se pueden controlar más dispositivos, o funciones en el mismo control. Los últimos seis bits de la secuencia son los que determinan cual es la función u orden que se le envía dispositivo, por ejemplo, subir o bajar volumen.

2.2 Controles gestuales para televisor

2.2.1 Primera televisión controlada por gestos

El reconocimiento de gestos tiene numerosas aplicaciones que proveen una interacción inteligente entre el usuario y la computadora. Se puede realizar mediante distintas tecnologías como pantallas táctiles, sensores de campo eléctrico, visión por computadora, etc. Actualmente el reconocimiento de gestos se aplica en: diseño de videojuegos, robótica, aplicaciones médicas, domótica, controles gestuales, entre otros.

Un control gestual consiste en un dispositivo o sistema que es capaz de manejar las funciones de otro dispositivo a distancia por medio de movimientos corporales. Desde los años 60 se han realizado investigaciones sobre controles basados en gestos para computadoras y otros aparatos electrónicos. (19)

El primer prototipo de control gestual para televisor se diseñó en 1994 (8). A partir de entonces se han realizado más investigaciones, prototipos y dispositivos comerciales relacionados con el tema, a continuación se presentan algunos de los más relevantes.

En 1994 William T. Freeman y Craig D. Weissman (8) presentaron el primer prototipo de control de televisión por medio de gestos con manos. Su objetivo era crear una forma de interacción con el televisor que no requiriera memorización de gestos o un entrenamiento exhaustivo para aprender a utilizar el control.

El prototipo de Freeman y Weissman funciona básicamente con un sólo gesto, que es la mano abierta del usuario. Cuando la mano abierta se detecta, se enciende el televisor y aparece en la parte posterior de la pantalla un menú de controles gráficos y el icono de una mano que se mueve siguiendo la posición de la mano abierta del usuario. El menú se muestra en la Figura 2.4, está constituido por un botón de apagado, un botón para silenciar, una barra deslizante que controla el nivel de volumen y otra que sirve para cambiar el canal. El usuario puede manipular el menú como si estuviera controlando el puntero de un ratón, a excepción de que no existe un botón para hacer clic, en lugar de ello si la mano del usuario se queda estática sobre un control por 200 milisegundos, el control cambia de color y se ejecuta un comando.

Para rastrear la mano del usuario se emplea un método de correlación normalizado, en el que se ocupa una variedad de diferentes representaciones de orientación de imágenes. El método de representación de la orientación tiene la ventaja de que no se ve afectado por las variaciones de luz. En la Figura 2.5 se muestra el método de reconocimiento de mano. Para evitar errores en la detección de la mano se remueven todos los objetos que permanecen fijos como los

2.2 Controles gestuales para televisor

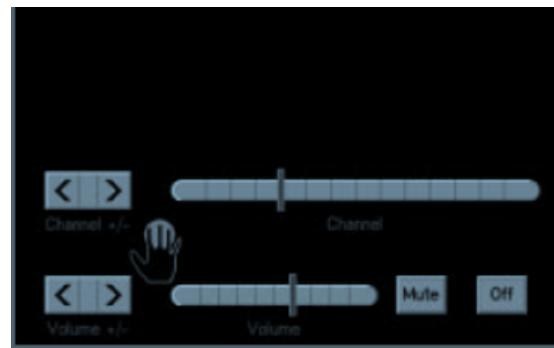


Figura 2.4: Menú de controles que se aparece en la parte superior de la pantalla de la TV (8).

muebles o el fondo de la imagen.

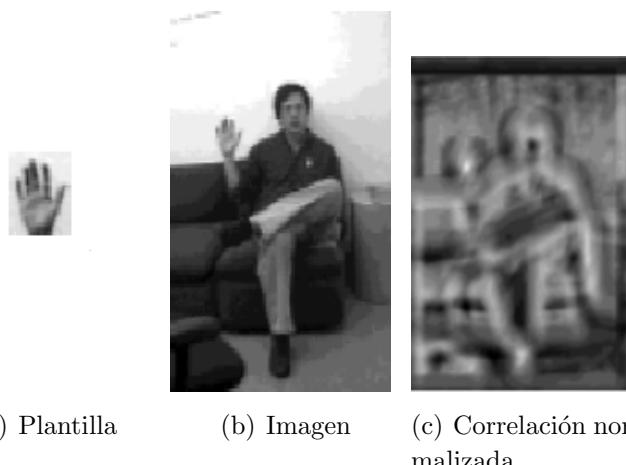


Figura 2.5: El método de reconocimiento de mano usado es la correlación normalizada. La correlación normalizada de la plantilla de la mano (a) con la imagen de entrada (b) se muestra en (c). La posición de máxima correlación está en la mano del usuario (8).

El sistema emplea una cámara de video *Flex-Cam* para adquirir imágenes en formato de televisión NTSC. Estas se digitalizan a 640x480 de resolución y se procesan en una computadora HP 735. El software empleado para realizar la detección de manos está programado en lenguaje C y C++. El menú de controles se visualiza en la pantalla de la computadora y en la televisión. La televisión es controlada por medio de un puerto serial conectado a un control remoto de televisor *All-In-One-12*. La interfaz para enviar comandos de la computadora al control es de *Home Control Concepts* y sólo puede transferirse un comando por segundo.

2. ANTECEDENTES

2.2.2 Sistema de interacción de una *Smart TV* basado en reconocimiento de gestos con manos

El sistema de interacción de *Smart TV* basado en movimiento de manos es un trabajo que implementa las funciones de un control remoto. Se propuso en 2013 por Cheng Yang, Xiaoyu Wu, Zhuojia Li y Qi Feng (7), de la Universidad de Comunicación de Beijing, China.

Como se observa en la Figura 2.6 el control se manipula con tres tipos de gestos: movimientos dinámicos para ejecutar las funciones de las teclas de navegación, la señal de "victoria" (V) para iniciar un teclado virtual que se muestra en la parte inferior de la pantalla del televisor y la simulación de un clic con las puntas de los dedos para detectar el evento de clic sobre el teclado virtual. El teclado virtual consta de los números del 0 al 9, la tecla DEL (suprimir) y la tecla OK (aceptar). Además de la funciones para controlar la TV, incluye un detector de privilegios que cede el control al usuario que hace el gesto de saludo.

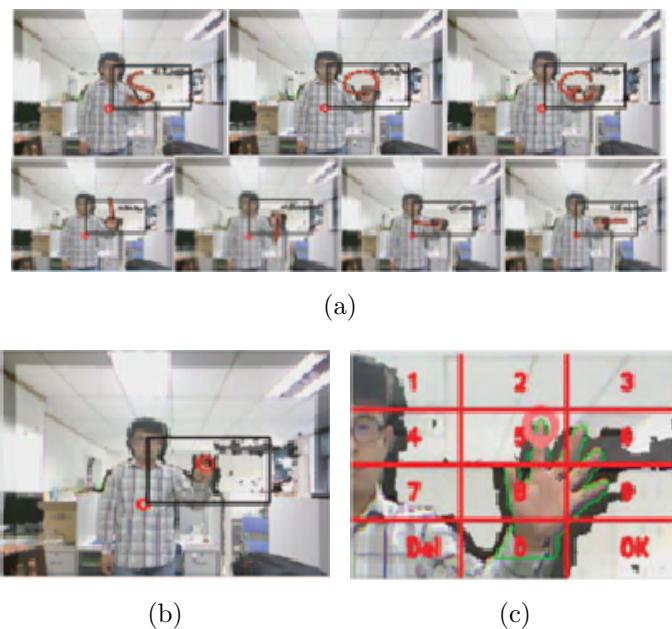


Figura 2.6: Muestras del reconocimiento de gestos: (a) siete gestos dinámicos; (b) reconocimiento de gestos estáticos; (c) entrada del teclado virtual. (7)

Se utiliza el sensor Kinect para obtener imágenes de color y de profundidad, después se segmenta el área de la mano para que con ello se puedan detectar las puntas de los dedos y se reconozcan los gestos con las manos.

Para los gestos estáticos, como la señal de "victoria", primero se detecta la posición de la mano, después se define una región de interés. Para la extracción de características de la mano se emplean Histogramas de Gradientes Orientados

2.2 Controles gestuales para televisor

(HOG), cuya ventaja es que muestran invariancia antes los cambios en la iluminación y rotación. Para entrenar el sistema de reconocimiento de gestos se utiliza el método de *Adaboost*, el cual es un método estadístico de detección de objetos que a partir de un conjunto de muestras determina cuales son las características que diferencian al objeto deseado de los demás objetos. En la Figura 2.7 se muestra un diagrama del método descrito.

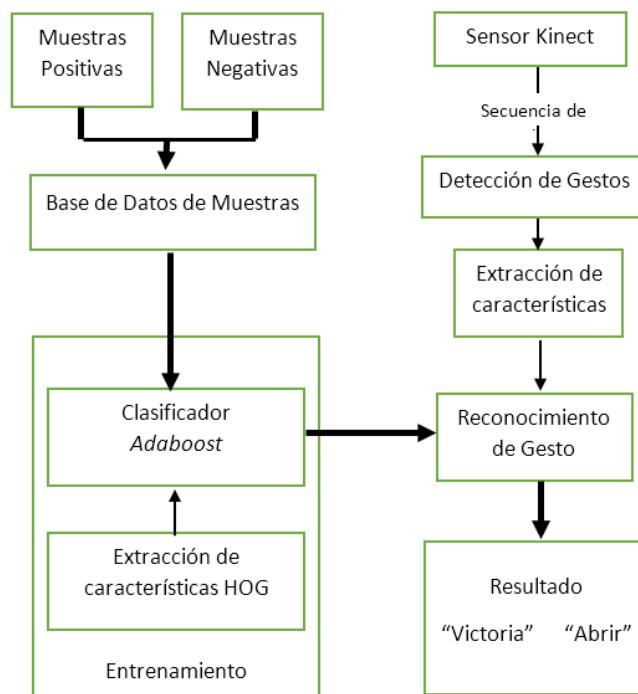


Figura 2.7: Proceso de reconocimiento de gestos estáticos. (7)

Para detectar gestos dinámicos se emplea el Modelo Oculto de Markov (*Hidden Markov Model - HMM*), que es un modelo estadístico que determina parámetros desconocidos u ocultos a partir de parámetros observables. Del video obtenido mediante Kinect se obtienen las coordenadas de la posición de la palma y a partir de ellas se extrae la trayectoria de la mano. Se crea la base de datos de trayectorias y para su reconocimiento se entrena un modelo con el HMM. Las trayectorias definidas son siete: arriba, abajo, izquierda, derecha, "S" (atrás), "O" (entrar) y "E" (regresar al programa anterior). (Figura 2.8)

El reconocimiento del click virtual se realiza utilizando la imagen de profundidad proporcionada por el Kinect. De dicha imagen se obtiene una imagen binaria, que básicamente consiste en separar al usuario del fondo, poniendo en color negro los pixeles del fondo y en color blanco los pixeles de la silueta del usuario. Después se toma el segmento de la mano para extraer el contorno y detectar las

2. ANTECEDENTES



Figura 2.8: Proceso de reconocimiento de gestos dinámicos. (7)

puntas de los dedos a partir un punto central. En la Figura 2.9 se muestra el proceso para el reconocimiento del click.

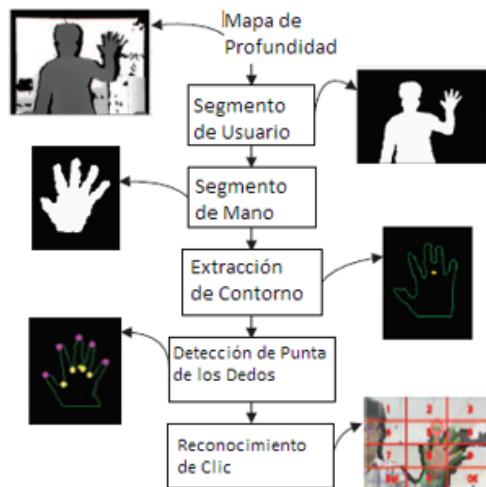


Figura 2.9: Proceso de reconocimiento del clic. (7)

2.2.3 Reconocimiento del estado de un usuario para un sistema de control de televisión de bajo costo basado en gestos

En 2014 Shiguo Lian, Wei Hu y Kai Wang (15) diseñaron uno de los pocos sistemas para controlar un televisor vía gestos que contempla el ahorro de energía. Esto se llevó a cabo con la detección del estado del usuario, para ello se consideraron cuatro posibles estados: Ausente, Otra Acción, Controlando TV y Viendo TV. Cuando se detecta que el usuario intenta controlar la TV, se activa el sistema

2.2 Controles gestuales para televisor

de reconocimiento de gestos, de lo contrario se mantiene apagado para reducir el gasto de energía y el gasto computacional.

El sistema propuesto se constituye de un arreglo de cuatro sensores de distancia ultrasónicos con el que se realiza la identificación del estado del usuario y de un sensor Kinect para el reconocimiento de gestos. Para la identificación del estado del usuario, el arreglo de sensores de distancia determina si el usuario se mueve calculando la distancia entre los brazos y el televisor. Para decidir si está viendo el televisor, se realizaba una detección facial mediante la técnica de extracción de características *Haar* y la técnica de clasificación *AdaBoost*.

Cuando se detecta el estado *Controlando*, se activa el sistema de reconocimiento de gestos y se muestra un mensaje en la pantalla del televisor indicando al usuario que puede comenzar a realizar gestos. Se utilizan tres tipos de algoritmos para el reconocimiento de gestos: reconocimiento de gestos estáticos, que consistía en detectar la palma y el puño que simulan un clic o doble clic; detección de gestos dinámicos, en los que se rastrea la trayectoria dibujada por la mano, la cual puede ser arriba, abajo, izquierda y derecha; reconocimiento de gestos basado en cámara de profundidad, en el que se obtiene la información de la cámara de profundidad del sensor Kinect. (Figura 2.10)

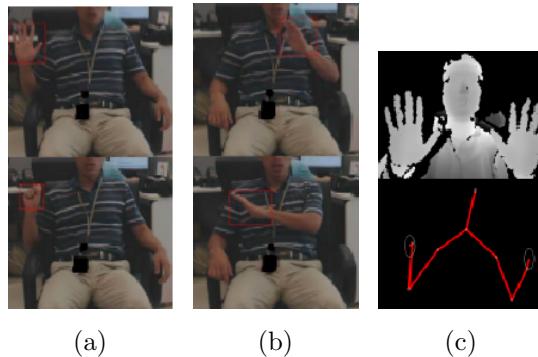


Figura 2.10: Resultados de la detección y rastreo de la mano, incluyendo la mano y el puño (a); rastreo de mano (b); detección de la mano basada en información de profundidad (c) (arriba se muestra la imagen de profundidad y abajo el esqueleto humano con las manos detectadas. (15)

2.2.4 Samsung Smart Interaction

A) Primera versión

En 2012 Samsung anunció que sus televisores inteligentes de la series 7000 y 8000 incluirían un nuevo tipo de tecnología llamada *Smart Interaction* (14), que consiste en un control vía gestos, reconocimiento facial y voz. Los televisores tienen una cámara y un micrófono integrados en la parte superior y externamente

2. ANTECEDENTES

cuentan con un emisor de infrarrojo. Este último se coloca frente al televisor y se comunica con él por medio de Bluetooth, el uso más común del emisor de IR es cambiar el volumen, cambiar canal y controlar otras opciones de los dispositivos externos conectados al televisor como pueden ser la caja de cable, el reproductor de DVD o el reproductor de Blue-ray.

El televisor se puede encender con el comando de voz "Hola tele encender". Una vez encendida la TV aparecen en pantalla las funciones que se pueden controlar por medio de la voz como son "subir volumen", "bajar volumen", "subir canal", "buscar" (para el navegador de Internet), "empezar" o "salir" (para manejar una aplicación), "cancelar" (para salir de un submenú). Para que desaparezca o aparezca de la pantalla el menú del control por voz se debe decir "Hola TV". El comando de voz que el usuario desea ejecutar aparece en pantalla para indicar que el reconocimiento se hizo adecuadamente. Se recomienda estar entre 1.5 metros a 5 metros del televisor para que el micrófono pueda captar adecuadamente la voz del usuario. Este control no sustituye del todo al control físico, pues si hay mucho ruido ambiental se debe usar el micrófono que viene integrado en el control remoto de la TV. Otra desventaja de este control es que no está disponible para todos los idiomas. (21)

Algunos comandos que son controlados por voz, también se pueden controlar por medio de gestos. Sólo basta con colocarse frente al televisor y mover la mano hasta que sea reconocida por el televisor, una vez que la mano es detectada, el cursor de la TV se puede controlar con el movimiento de la mano (Figura 2.11). Se puede cambiar el canal, manipular el volumen y ver fotografías deslizando la mano de izquierda a derecha y viceversa. Para seleccionar una aplicación o un menú (simular un clic) se debe abrir y cerrar el puño una sola vez. El control de gestos se puede calibrar para trabajar bajo condiciones de poca luminosidad ambiental.



Figura 2.11: Control gestual de una *Smart TV* serie 7000 de Samsung. (9)

La opción de reconocimiento facial sirve para que cinco usuarios creen su propio perfil con características personalizadas de acuerdo a las preferencias de

2.2 Controles gestuales para televisor

programas y aplicaciones de cada quien. Para entrar a cada perfil no es necesario ingresar un usuario y contraseña, el usuario sólo debe situarse frente al televisor para que se efectúe el reconocimiento facial.

B) Versión del 2013

En la versión de *Smart Interaction* del 2013 se agregaron nuevos gestos para controlar la TV. Además de poder controlar el cursor con el movimiento de la mano y seleccionar objetos, se puede dar zoom, rotar imágenes, dar "me gusta" en una red social, entre otras funciones controladas con nuevos comandos gestuales. Esta versión viene integrada en los televisores de las series 8000, 7500 y 9000, cuyo costo es superior a los 39,999 pesos mexicanos (2). Se pueden agregar las características de *Smart Interaction* a televisores de otras series mediante el dispositivo *Evolution Kit*, siempre y cuando sean *Smart TV*. El costo del *Evolution Kit* tiene un costo de 299 euros (16), aproximadamente 5,591 pesos mexicanos.

El conjunto de gestos de *Smart Interaction* se muestra en la Tabla 2.1 y está constituido de 9 comandos que se ejecutan con movimientos de una sola mano y 4 que se ejecutan con ambas manos.

2.2.5 Diseño de una interfaz de usuario con integración de gestos dinámicos y reconocimiento de números manuscritos

En 2014, Jia-Shing Sheu, Guo-Shin Huang y Ya-Ling Huang (23) presentaron una interfaz de usuario para el control de una TV digital. El sistema de control se constituye de dos partes principales: control del cursor de la TV mediante reconocimiento de gestos dinámicos y selección de canal por medio de números manuscritos virtuales.

Para lograr una Interfaz Natural de Usuario (*Natural User Interface - NUI*) se utiliza un sensor Kinect. La detección de gestos dinámicos se realiza mediante la extracción de las coordenadas del esqueleto del usuario, las cuales son proporcionadas por el sensor Kinect. Dicha detección se activa cuando el usuario levanta la mano derecha a la altura de su hombro y la desplaza horizontalmente de izquierda a derecha y se desactivaba cuando el usuario coloca sus manos debajo de la cintura. Cuando el sistema se activa, el usuario saluda con la mano izquierda para iniciar el software interactivo de la TV, después de eso puede controlar el cursor mediante gestos dinámicos con la mano derecha y si desea escribir un número vía gestos, debe alzar su mano izquierda para llamar a la interfaz de escritura virtual.

Para poder controlar el cursor y el trazo de números en pantalla, primero se adquiere la imagen de profundidad del sensor Kinect y a partir de ella obtiene la información del esqueleto (coordenadas de las articulaciones). Posteriormente se escala la posición de cada articulación dentro de un rectángulo de tamaño

2. ANTECEDENTES

Tabla 2.1: Guía de gestos del *Smart Interaction*

| Gesto | Funciones | Gesto | Funciones |
|-------|--|-------|---|
| | Iniciar el control gestual con una mano. | | Moverse al menú previo. |
| | Moverse a la siguiente página o imagen o adelantar 10 minutos de un video. | | Arrastrar imágenes, mapas o fotos. |
| | Moverse a la página o imagen previa o atrasar 10 minutos de un video. | | Iniciar el control gestual con dos manos. |
| .(22) | Controlar la posición del cursor. | | Hacer más grande una imagen (zoom in). |
| | Dar "me gusta" a un contenido de Facebook. | | Hacer más pequeña una imagen (zoom out). |
| | Seleccionar un contenido. | | Rotar imágenes 90 grados. |
| | Subir o bajar rápidamente un canal o el volumen (la diferencia con el gesto anterior es que la mano se mantiene cerrada por más tiempo). | | |

predeterminado y finalmente se mapea a las coordenadas del cursor de la TV.

La Tabla 2.2 muestra las reglas que debe seguir el televidente para trazar con movimientos los números virtuales, la trayectoria comienza en el punto y debe seguirse la dirección de la flecha. Para identificar adecuadamente los números virtuales trazados por el usuario se entrena a una red neuronal artificial de retropropagación (*Backpropagation Neural Network*) que es un modelo de entrenamiento supervisado para reconocer patrones con el mínimo error.

2.2 Controles gestuales para televisor

Tabla 2.2: Reglas de gestos dinámicos para escritura de números virtuales.(23)

| Dígito | Regla de escritura |
|--------|---|
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |

2.2.6 Control remoto con reconocimiento de gestos basado en acelerómetro para interacción con la TV digital

En 2014 José Ducloux, Pablo Petrashin, Wálter Lancioni y Luis Toledo (6) propusieron un control con reconocimiento de gestos mediante un acelerómetro para uso en el contexto de la televisión digital en Argentina.

El control se compone de 20 gestos diferentes que se muestran en la Figura 2.12. El usuario debe portar un acelerómetro y cuando desee interactuar con la TV debe mantener presionado un botón mientras realiza el gesto que corresponde a la función deseada, de esta manera el gesto puede ser reconocido.

Los gestos son representados por vectores que contienen las variaciones en los niveles de aceleración contra el tiempo en un espacio tridimensional. Esos datos se envian del acelerómetro a un microcontrolador PIC32MX250 por medio del bus I²C. En el microcontrolador se realiza el reconocimiento de gestos mediante una red neuronal artificial(RNA) de tipo perceptrón multicapa, que tiene una

2. ANTECEDENTES

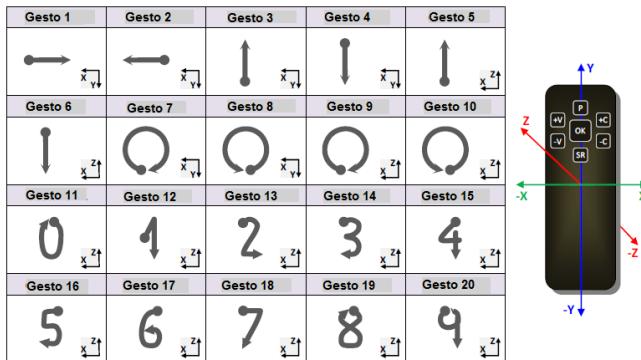


Figura 2.12: Gestos y sistema de coordenadas

tasa de reconocimiento del 98.65%. Posteriormente se emplea un led infrarrojo para enviar comandos a una televisión digital.

2.3 Aplicaciones de las redes neuronales de retropropagación

Además del reconocimiento de gestos las redes neuronales artificiales se emplean para desarrollar otras aplicaciones. El algoritmo de aprendizaje más común para redes neuronales es el de retropropagación que se considera un modelo de entrenamiento supervisado y adaptativo. La red de retropropagación se entrena hasta que se encuentre un mínimo error en la clasificación de patrones. Se puede aplicar para resolver varios tipos de problemas, pero no para todos funciona y no puede saberse de antemano si este modelo puede aplicarse para encontrar una solución. Algunos ejemplos de aplicaciones de las redes neuronales de retropropagación son:

- Detección de una enfermedad. Por medio de una RNA de retropropagación se mide la gravedad de una enfermedad. Por ejemplo se puede analizar una radiografía para medir la severidad de una osteoartritis.(20)
- Clasificación de imágenes. Resuelven problemas regulares de detección remota, que es la adquisición de información sobre un fenómeno a pequeña o grande escala como la observación de fenómenos meteorológicos. (13)
- Predicción de fenómenos naturales. Son capaces de detectar terremotos basándose en la información disponible sobre terremotos. (5)
- Detección de fallas en dispositivos eléctricos. Sirven de diagnóstico de fallas en transformadores evitando accidentes y pérdidas en la industria. (3)

Capítulo 3

Técnicas y Herramientas

3.1 Introducción

Para este proyecto se emplean las siguientes herramientas: Kinect, redes neuronales artificiales, MatLab, Arduino y protocolos de comunicación por infrarrojo. Para la captura de los movimientos del usuario se emplea el sensor Kinect, con ayuda del *toolbox* de adquisición de imágenes de MatLab se guarda y procesa la información capturada. Para reconocer los movimientos del usuario se emplea una red neuronal artificial programada en MatLab. Inmediatamente después, MatLab envía una señal de infrarrojo al televisor mediante una tarjeta Arduino UNO. A continuación se explica con detalle cada una de la herramientas y técnicas empleadas.

3.2 Kinect

3.2.1 Generalidades y componentes

Kinect para Xbox 360 es un dispositivo de detección de movimiento diseñado por Alex Kipman. Fue lanzado al mercado por la compañía Microsoft en noviembre del 2010 . Originalmente fue creado para controlar la consola de videojuegos Xbox 360 mediante la captación de movimientos corporales y reconocimiento de voz (Figura 3.1). Esto le permite al usuario interactuar con el videojuego sin la necesidad de emplear un control físico.

En junio de 2011 Microsoft desarrolló una versión del Kinect para PC, específicamente para el sistema operativo Windows. Lo anterior ha atraído la atención tanto de la industria como de la investigación, haciendo posible el desarrollo de aplicaciones para Kinect en diversas áreas, como el entretenimiento, la robótica, la medicina, la seguridad, la educación, entre otras.

Mediante una interfaz natural de usuario, Kinect le permite al usuario una

3. TÉCNICAS Y HERRAMIENTAS



Figura 3.1: Kinect para la consola de videojuegos Xbox 360. (9)

interacción más intuitiva con la computadora, la consola de Xbox y otros dispositivos. Esto quiere decir que el usuario aprende rápidamente a utilizar los dispositivos, no se requiere de un entrenamiento previo o de un aprendizaje tedioso, por el contrario, la interacción con la tecnología se da naturalmente. Es por ello, que en este trabajo se emplea el sensor Kinect para controlar un televisor.

En este proyecto se emplea Kinect para Xbox 360, ya que es compatible tanto para la consola de videojuegos como para la computadora. Kinect para Xbox 360 es una barra horizontal de sensores montada sobre una base motorizada. Sus dimensiones son de 7.5 cm de alto, 28 cm de ancho y 6 cm profundidad. Los componentes principales del Kinect se muestran en la Figura 3.2 y son los siguientes:

- Sensor RGB: Captura imágenes a color.
- Emisor de infrarrojo y sensor infrarrojo de profundidad: hacen posible el seguimiento del usuario en tres dimensiones y la captura de imágenes de profundidad.
- Arreglo de micrófonos: Se usa un conjunto de micrófonos en el borde frontal inferior del sensor para reconocimiento de voz.
- Un acelerómetro de tres ejes configurado para un rango 2G, donde G es la aceleración debida a la gravedad. Se emplea para determinar la orientación del Kinect.
- Inclinación motorizada: mediante una impulsión mecánica que se encuentra en la base del Kinect el sensor es capaz de inclinarse verticalmente.

3.2.2 Especificaciones

Cuando se utilice el sensor Kinect hay que asegurarse de colocarlo en una superficie estable, donde no pueda sufrir alguna caída. Además se recomienda:

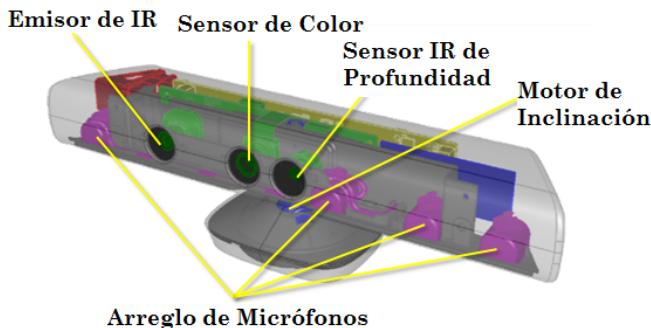


Figura 3.2: Componentes principales de Kinect Xbox 360 (9).

- Evitar colocar el sensor enfrente de un altavoz o una superficie que vibre.
- Mantenerlo alejado de la luz directa del sol.
- Alejar el sensor de las fuentes de calor.
- Usar el sensor dentro de su temperatura especificada: 41 a 95 grados centígrados. Si se sobrecalienta, debe apagarse y esperar a que se estabilice su temperatura.
- No ajustar el grado de inclinación vertical de sensor manualmente. Los motores del sensor realizan esa función.

Las especificaciones generales del Kinect se muestran en la Tabla 3.1

Tabla 3.1: Especificaciones del Kinect Xbox 360 (17).

| Característica | Especificaciones |
|--------------------------------|---|
| Campo de visión horizontal | 57° |
| Campo de visión vertical | 43° |
| Rango de inclinación vertical | ±27° |
| Rango de profundidad | 1.2m a 3.5m |
| Resolución de imágenes a color | VGA 640x480 |
| Imágenes por segundo | 30 cuadros por segundo (FPS). |
| Acelerómetro | Acelerómetro configurado para un rango de 2G. |
| Sistema de seguimiento | Rastrea hasta 6 personas, incluyendo 2 jugadores activos. Rastrea 20 articulaciones por persona. |
| Sistema de audio | Arreglo de cuatro micrófonos, cada canal procesa 16-bits con una frecuencia de muestreo de 16kHz. Posee un sistema de cancelación de eco y de supresión de ruido. |

3. TÉCNICAS Y HERRAMIENTAS

3.2.3 Funcionamiento

Para obtener imágenes de profundidad y el seguimiento tridimensional del usuario, el sensor Kinect proyecta una patrón infrarrojo determinado al ambiente y mide el patrón infrarrojo que regresa (usando el sensor infrarrojo de profundidad) (Figura 3.3). El patrón recibido es comparado con el patrón proyectado y las diferencias son usadas para extraer datos de profundidad. Lo anterior hace que el sensor Kinect no dependa completamente de la iluminación ambiental.



Figura 3.3: Patrón de puntos proyectados por el emisor infrarrojo. (9)

Kinect contiene internamente una variedad de chips de procesamiento y controladores, el más relevante es el procesador de imágenes *PrimeSense* que procesa las salidas de los sensores RGB y de profundidad para que a cada pixel se le asigne una información de color y de profundidad y esta sea transmitida por la interfaz USB.

3.2.4 Kit de Desarrollo de Software

El Kit de Desarrollo de Software (SDK) de Kinect para Windows es una herramienta esencial para los desarrolladores de aplicaciones. Permite diseñar aplicaciones que soporten reconocimiento de voz y gestos usando la tecnología de los sensores en computadoras con Windows 7, 8 y 10. El software se encuentra disponible en la página oficial de Microsoft (18).

A) Características del Kit de Desarrollo de Software

Algunas de características que incluye el SDK son:

- Drivers para utilizar los sensores de Kinect en una computadora con Windows 7, Windows 8 y Windows 10.
- Interfaces de programación de aplicaciones (APIs) e interfaces de dispositivos.

- Captura de imágenes a color, imágenes de profundidad, datos del esqueleto y entada de audio.
- Documentación y programas de muestra.

B) Requerimientos del sistema

Sistema operativos soportados y arquitecturas:

- Windows 7 (de 32 o de 64 bits)
- Windows 8 (de 32 o de 64 bits)
- Windows 8.1 (de 32 o de 64 bits)
- Windows 10 (de 32 o de 64 bits)
- Windows Embedded Standard 7



Figura 3.4: Laptop Acer modelo Aspire E15 E5-522-86AU (9).

Requerimientos mínimos de hardware:

- Procesador de 32 bits(x86) o 64 bits(x64).
- Computadora con procesador de 2 núcleos a velocidad de 2.66GHz o superior.
- Bus USB 2.0 dedicado al Kinect.
- 2GB de RAM o más.
- Tarjeta gráfica que soporte *DirectX 9.0c*.

3. TÉCNICAS Y HERRAMIENTAS

- Cable USB adaptador para conectar la Kinect a la PC.

Requerimientos de software:

- Visual Studio 2010 o MatLab 2013a como mínimo.
- .NET Framework 4.5

La computadora empleada en este proyecto es una laptop Acer modelo Aspire E15 E5-522-86AU(Figura 3.4) y tiene las siguientes características:

- Procesador AMD de cuatro núcleos A8-7410 a 2.8 GHz.
- Tarjeta gráfica AMD *Radeon R5 Graphics*.
- Memoria RAM de 8.00 GB.
- 2 puertos USB 2.0.
- Sistema operativo Windows 10 *Home Single Language*.
- Sistema operativo de 64 bits, procesador x64.
- MatLab 2014b con el soporte para Kinect instalado.

3.2.5 Flujo de datos

El sensor Kinect se configura para que devuelva uno o varios flujos de datos, los cuales se enumeran a continuación:

1. Imágenes a color (proporcionados por el sensor de color).
2. Datos de profundidad (proporcionados por el sensor de profundidad).
3. Datos de las articulaciones del usuario(proporcionados por el sensor de profundidad).
4. Audio (proporcionado por el arreglo de micrófonos).

Para este trabajo se emplearán únicamente algunos atributos de los datos del esqueleto, pues sólo es necesario conocer las coordenadas de algunas articulaciones del esqueleto del usuario.

A) Datos de profundidad

El sensor de profundidad retorna los datos de segmentación de una persona usando el sensor de profundidad de Kinect. El mapa de profundidad es la distancia en milímetros desde el plano del sensor Kinect. En la Figura 3.5 se muestra un ejemplo de imagen de profundidad capturada por el Kinect. El flujo de datos de profundidad soporta los siguientes formatos:

- Profundidad 640x480: Resolución de 640x480, 30 imágenes por segundo.
- Profundidad 320x240: Resolución de 320x240, 30 imágenes por segundo.
- Profundidad 80x60: Resolución de 80x60, 30 imágenes por segundo.



Figura 3.5: Imagen de profundidad obtenida con el sensor Kinect (9).

B) Datos de las articulaciones del usuario

El flujo de datos del esqueleto es retornado por el sensor de profundidad del Kinect. Los cuadros proporcionados contienen datos del campo de visión del Kinect y del tiempo. Contiene la posición del esqueleto al Kinect y la posición en coordenadas 3-D de 20 articulaciones en metros.

Para el seguimiento del esqueleto sólo dos personas pueden ser rastreadas a la vez, aunque se puede obtener la información de segmentación de seis personas a la vez. Esto significa que el Kinect provee el rastreo completo de dos esqueletos(rastreo activo) y una posición parcial de rastreo de cuatro personas más (rastreo pasivo). Los rangos de rastreo son el rango por default de 50cm a 400cm y el rango cercano de 40cm a 300cm.

El sensor Kinect detecta y proporciona información de 20 articulaciones del esqueleto que se muestran en la Figura 3.6. Dicha información es referente a las imágenes de profundidad obtenidas y a las coordenadas 3D de las articulaciones del usuario.

El sistema de coordenadas del esqueleto está conformado por coordenadas x , y y z de cada articulación y se expresan en metros. La ubicación del Kinect es el punto de partida, pues representa el punto de origen con un eje z positivo, que se extiende hacia donde apuntan los sensores. A la izquierda del punto mencionado anteriormente se encuentra la dirección positiva del eje x y la derecha del mismo

3. TÉCNICAS Y HERRAMIENTAS



Figura 3.6: Articulaciones del esqueleto detectadas por Kinect. (9)

se encuentra la dirección negativa. El eje *y* positivo se extiende hacia arriba del punto de origen y el negativo hacia abajo del eje de dicho punto. (Figura 3.7)

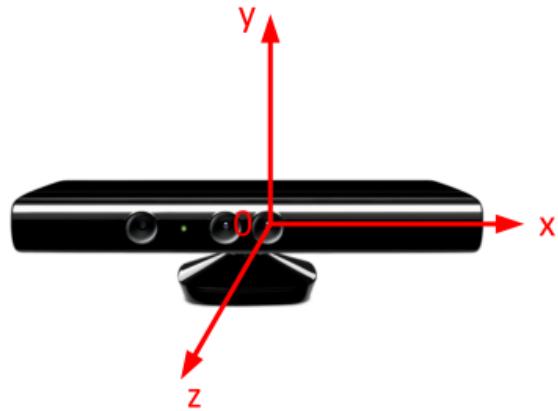


Figura 3.7: Ejes del sistema de coordenadas de Kinect. (9)

3.3 Redes Neuronales

La etapa de reconocimiento de los gestos capturados por el sensor Kinect se realiza mediante redes neuronales artificiales, que se explicarán en esta sección.

3.3.1 Neuronas biológicas

Las redes neuronales artificiales tienen como fin imitar de la manera más fiel el comportamiento de las redes neuronales biológicas, lo que implica el diseño de sistemas con capacidades de procesamiento paralelo.

Las neuronas biológicas se componen de un cuerpo celular, que posee ramificaciones llamadas dendritas y una ramificación más larga llamada axón. El axón se encarga de transportar la señal de salida de la neurona hacia otras neuronas. Las entradas llegan a través de las dendritas, que provienen de los axones de otras neuronas u otros receptores. Se denomina sinapsis a las conexiones entre el axón de una neurona y la dendrita de otra. Por medio de la sinapsis, una neurona recibe señales de muchas otras (hasta varias decenas de miles) y envía señales a otras más. En la Figura 3.8 se muestra una neurona biológica.

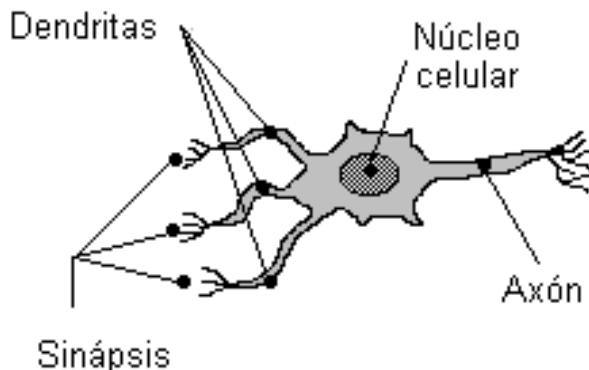


Figura 3.8: Neurona biológica. (9)

Las neuronas son dispositivos considerados de "todo o nada", pues o se dispara su nivel de actividad o permanecen inactivas. Una neurona sólo se activa si la influencia colectiva de todas sus entradas supera un nivel mínimo de potencial eléctrico. Esta activación se traduce en un impulso electroquímico que se propaga a lo largo del axón hasta otras neuronas, activándolas o inhibiéndolas.

Una de las características más importantes de las conexiones sinápticas es que poseen plasticidad, es decir, el grado de influencia de la sinapsis cambia con el tiempo, e incluso se crean nuevas sinapsis. Aunque no se conocen los mecanismos exactos, se sabe que las capacidades de aprendizaje y memorización que poseen los seres vivos se basa en la plasticidad del cerebro.

3. TÉCNICAS Y HERRAMIENTAS

3.3.2 Neuronas artificiales

La neurona artificial es un elemento que posee un estado interno llamado nivel de activación y recibe señales que le permiten cambiar de estado (12). Se denomina S al conjunto de estados que puede tomar la neurona y puede tomar dos o más valores. La función que permite el cambio de estado se denomina función de transición de estado o función de activación. El nivel de activación depende de sus entradas que pueden provenir del exterior o de las neuronas con las que mantiene una conexión. Para calcular el estado de activación se calcula primero la entrada total de la célula, E_i . Este valor se obtiene sumando todas las entradas ponderadas por ciertos valores.

En la Figura 3.9 se muestra como las entradas definidas por un vector \bar{X} corresponden a las señales de sinapsis de una neurona biológica. Cada señal de entrada se multiplica por un peso asociado $\omega_1, \omega_2, \dots, \omega_n$, y posteriormente se suman todos esos productos, dando como resultado una salida E:

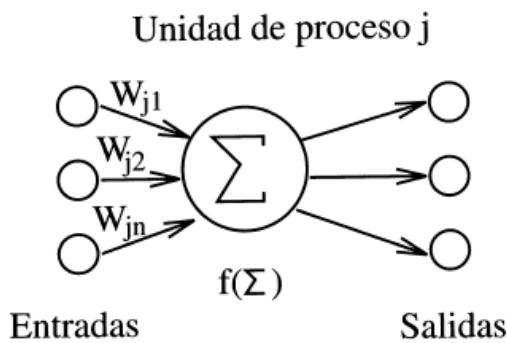


Figura 3.9: Neurona artificial.(12)

$$E = x_1\omega_1 + x_2\omega_2 + \dots + x_n\omega_n \quad (3.1)$$

Las señales E son procesadas por una función de transferencia llamada función de activación F que produce una salida S .

La interconexión de neuronas artificiales forma una red neuronal (Figura 3.10). Las redes neuronales tienen dos fases:

1. Fase de aprendizaje: El aprendizaje en una RNA consiste en la determinación de los valores precisos de los pesos para todas sus conexiones, que la capacite para la resolución eficiente de un problema. El proceso general de aprendizaje consiste en ir introduciendo paulatinamente todos los ejemplos del conjunto de aprendizaje, y modificar los pesos de las conexiones siguiendo un determinado esquema de aprendizaje. Una vez introducidos todos los ejemplos se comprueba si se ha cumplido cierto criterio de convergencia; de no ser así se repite el proceso y todos los ejemplos del conjunto vuelven a ser introducidos.

2. Fase de recuperación: Se presentan a la entrada patrones que no conoce para probar su capacidad de aprendizaje.

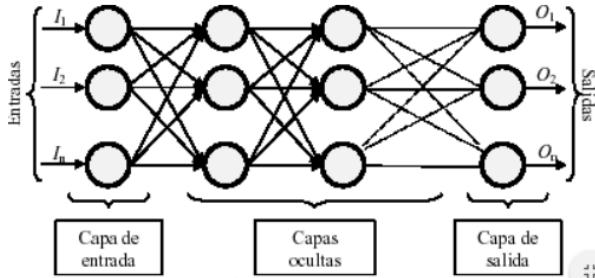


Figura 3.10: Estructura de una red neuronal artificial. (9)

3.3.3 Perceptrón Multicapa

El perceptrón multicapa es una clase de red neuronal artificial que propaga los errores medidos en la salida de la red hacia las capas ocultas. Esta RNA es ampliamente usada para resolver problemas reales debido a que a partir de un conjunto de ejemplos aproxima relaciones no lineales y filtra ruido en los datos.

El perceptrón multicapa tiene capas de diferentes niveles y en cada capa hay un conjunto de neuronas. Existen tres tipos de capas: la capa de entrada, las capas ocultas y la capa de salida. No hay reglas concretas para determinar el número de neuronas o número de capas de una red para resolver un problema concreto. En general tres capas son suficientes (entrada, oculta y salida).

Las neuronas de la capa de entrada se encargan de recibir los patrones que ingresan a la red y propagarlos a todas las neuronas de la siguiente capa. La capa de salida entrega el resultado de la clasificación de patrón de entrada. Las neuronas de las capas ocultas llevan a cabo un procesamiento no lineal de los patrones ingresados.

A las conexiones entre neuronas se les vincula un número real llamado peso de la conexión y a cada neurona se le asocia un número llamado umbral. Las neuronas de cada red están conectadas con todas las neuronas de la capa siguiente.

Para encontrar la relación entre las variables de entrada y los valores deseados de salida, el perceptrón multicapa propaga hacia adelante los valores de entrada. Todas las neuronas procesan los valores de entrada y producen una activación que se propaga hacia las neuronas de la siguiente capa.

Suponiendo que se tiene una perceptrón multicapa con C capas, donde $C-2$ son ocultas y n_c son las neuronas de la capa c , para $c=1,2,\dots,C$. La matriz de pesos que asocia las conexiones de la capa c a la capa $c+1$ para $c=1,2,\dots,C-1$ es

3. TÉCNICAS Y HERRAMIENTAS

$W^c = (w_{ij}^c)$. Donde (w_{ij}^c) es el peso de la conexión de la neurona i de la capa c a la neurona j de la capa $c+1$. El vector de umbrales de las neuronas de la capa c para $c=2,\dots,C$ es $U^c = (u_i^c)$. Además, las activaciones de la neurona i de la capa c se representan como a_i^c y se calculan como se muestra a continuación.

- Activación de las neuronas de la capa de entrada. Las neuronas de la capa de entrada sólo se encargan de transmitir hacia la red los patrones de entrada. Así que:

$$a_i^1 = x_i \quad (3.2)$$

para $i=1,2,\dots,n_1$

- Activación de las neuronas de las capas ocultas. Por medio de la aplicación de la función de activación f , las neuronas ocultas de la red procesan la información recibida:

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} + u_i^c\right) \quad (3.3)$$

para $i=1,2,\dots,n_c$ y $c=2,3,\dots,C-1$

donde a_i^{c-1} son las activaciones de las neuronas de la capa $c-1$

- Activación de las neuronas de la capa de salida.

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} + u_i^C\right) \quad (3.4)$$

donde $Y = (y_1, y_2, \dots, y_{n_C})$ es el vector salida de la red.

Las funciones de activación que más se emplean son la función sigmoidal y la función tangente hiperbólica. La función sigmoidal proporciona salidas dentro del intervalo $[0,1]$ y la tangente hiperbólica en el intervalo $[-1,1]$. Sus correspondientes expresiones se presentan a continuación:

- Función sigmoidal:

$$f_1(x) = \frac{1}{1 + e^{-x}} \quad (3.5)$$

- Función tangente hiperbólica:

$$f_2(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (3.6)$$

En este proyecto se emplea la función tangente hiperbólica debido a que el rango del posibles activaciones es mayor.

3.3.4 Algoritmo de Retropropagación

Un algoritmo de aprendizaje es aquel que se emplea para adaptar y modificar todos los parámetros de la red (pesos y umbrales). Para el perceptrón multicapa se emplea un algoritmo de aprendizaje supervisado, es decir que sus parámetros se modifican en base a las salidas deseadas o supervisor.

Para obtener el error que presenta el perceptrón multicapa se comparan las salidas de la red con las salidas deseadas mediante la siguiente función de error:

$$E = \frac{1}{N} \sum_{n=1}^N e(n) \quad (3.7)$$

donde N es el número de patrones de entrada y $e(n)$ es el error cometido por en patrón n , que está dado por la siguiente expresión:

$$e(n) = \frac{1}{2} \sum_{i=1}^{n_c} (s_i(n) - y_i(n))^2 \quad (3.8)$$

donde y_i son los elementos del vector de salidas de la red y s_i son los elementos del vector de salidas deseadas para el correspondiente patrón n .

Para alcanzar un aprendizaje óptimo se debe encontrar un mínimo de la función de error, el procedimiento más utilizado para llegar a ese objetivo está basado en métodos de gradiente estocástico. En estos métodos se minimizan sucesivamente los errores para cada patrón $e(n)$, en lugar de minimizar el error total E . En el método de descenso de gradiente estocástico, cada parámetro w de la red se modifica para cada patrón de entrada n de acuerdo con la siguiente ley de aprendizaje:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w} \quad (3.9)$$

donde $e(n)$ es el error para el patrón n y α es la razón o tasa de aprendizaje.

El método de gradiente se aplica de forma eficiente debido a que las neuronas están agrupadas en capas de diferentes niveles, esto da como resultado el algoritmo de retropropagación o regla delta generalizada. Lleva ese nombre debido a que el error cometido en la salida de la red es propagado hacia atrás, así el error se transforma en uno para cada neurona oculta de la red.

3. TÉCNICAS Y HERRAMIENTAS

3.3.5 Pasos del proceso de aprendizaje

1. Inicializar los pesos y umbrales de la red con valores pequeños(alrededor de cero) aleatorios.
2. Presentar un patrón de entrada y especificar la salida deseada que debe generar la red.
3. Se calcula la salida actual de la red $Y(n)$ propagando hacia la salida los valores del paso anterior con las Ecuaciones 3.2, 3.3, 3.4.
4. Calcular el error cuadrático para el patrón ingresado empleando la Ecuación 3.8.
5. Actualizar los pesos y umbrales de la red, aplicando la regla delta generalizada. Para ello llevan a cabo los siguientes pasos:

- (a) Calcular los valores ∂ para las neuronas de la capa de salida con la Ecuación 3.10.

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha \partial_i^C(n) a_j^C - 1(n) \quad (3.10)$$

para $j=1,2,\dots,n_{C-1}$ $i=1,2,\dots,n_C$

- (b) Calcular ∂ para el resto de las neuronas utilizando la Ecuación 3.11, se empieza desde la última capa oculta y se propagan hacia atrás los valores hasta la capa de entrada.

$$\partial_j^{c+1}(n) = f'(\sum_{k=1}^{n_c} w_{kj}^c a_k^c + u_j^c) \sum_i = 1^{n_{c+1}} \partial_i^{c+2}(n) w_{ji}^c \quad (3.11)$$

- (c) Modificar pesos y umbrales de red. Emplear las Ecuaciones 3.12 y 3.13 para los pesos y umbrales de la capa de salida y aplicar 3.14 y 3.15 para resto de los pesos y umbrales.

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha \partial_i^C(n) a_j^{C-1}(n) \quad (3.12)$$

para $j=1,2,\dots,n_{C-1}$ $i=1,2,\dots,n_C$

$$u_i^C(n) = u_i^C(n-1) + \alpha \partial_i^C(n) \quad (3.13)$$

para $j=1,2,\dots,n_{C-1}$ $i=1,2,\dots,n_C$

$$w_{kj}^c(n) = w_{kj}^c(n-1) + \alpha \partial_j^{c+1}(n) a_k^c(n) \quad (3.14)$$

para $k=1,2,\dots,n_c$ y $c=1,2,\dots,C-2$

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \alpha \partial_j^{c+1}(n) \quad (3.15)$$

para $k=1,2,\dots,n_{c+1}$ y $c=1,2,\dots,C-2$

6. Repetir los pasos 2,3,4 y 5 para todos los patrones de entrada, con lo que se tiene un ciclo de aprendizaje.
7. Evaluar el error total de la red mediante la Ecuación 3.7.
8. Repetir los pasos del 2 al 7 hasta obtener el mínimo error de entrenamiento.

3.4 MatLab

Para la adquisición de imágenes del sensor Kinect, el reconocimiento de gestos del usuario y la comunicación con la tarjeta Arduino se emplea MatLab. Existen otros paquetes computacionales para realizar los procesos antes mencionados, tal es el caso de Visual Studio; sin embargo en este proyecto se optó por usar MatLab, debido a la facilidad que proporciona para implementar las funciones antes mencionadas.

MatLab(Matrix Laboratory) es un paquete computacional que combina cálculo numérico, gráficos, visualización y un lenguaje de programación de alto nivel. En MatLab se pueden realizar operaciones aritméticas reales y complejas con matrices y escalares, resolver sistemas de ecuaciones no lineales, integrar funciones y sistemas de ecuaciones diferenciales y algebraicas.

Se eligió esta herramienta porque además de las operaciones matemáticas antes mencionadas posee *toolboxes* que son paqueterías que contienen algoritmos, funciones y aplicaciones que facilitan el manejo de algún área en específico. Los *toolboxes* empleados para este trabajo son *Image Acquisition Toolbox*, *Neural Network Toolbox* y *GUIDE*. El primero permite la adquisición de datos del Kinect, el segundo se emplea para la implementación de la red neuronal de retropropagación y el último para diseñar una interfaz gráfica de usuario. Además se emplean algunos comandos para establecer la comunicación serial con el dispositivo Arduino.

3.4.1 Toolbox de adquisición de imágenes

*Image Acquisition Toolbox*TM provee funciones y bloques que permiten conectar cámaras industriales y científicas a MatLab y Simulink. Incluye una aplicación que mediante una interfaz gráfica permite detectar y configurar las propiedades de hardware. A continuación se mencionan las funciones del toolbox empleadas para este proyecto:

3. TÉCNICAS Y HERRAMIENTAS

- ***imaqhwinfo***: Muestra la información acerca del hardware disponible para la adquisición de imágenes.
- ***imaqreset***: Desconecta y elimina todos los objetos de adquisición de imágenes.
- ***videoinput***: Crear el objeto de adquisición de imágenes del hardware indicado.
- ***start*** : Obtiene el uso exclusivo del dispositivo de adquisición de imágenes.
- ***stop***: Detiene el objeto del video de entrada.
- ***getselectedsource***: Retorna el objeto de video de origen seleccionado actualmente.
- ***load***: Carga el objeto de adquisición de imágenes al *workspace* de MatLab.
- ***save***: Guarda el objeto de adquisición de imágenes en un archivo *.mat*.
- ***trigger***: Inicia el registro de los datos de entrada.
- ***triggerconfig***: Configura las propiedades de disparo del objeto del video de entrada.

Este *toolbox* soporta Kinect y se configuran varias propiedades para el sensor de color y el de profundidad. En la Tabla 3.2 se muestran las propiedades para el seguimiento del esqueleto, específicas del sensor de profundidad, que es el que se emplea en este proyecto.

3.4.2 Toolbox de redes neuronales

Neural Network ToolboxTM provee algoritmos, funciones y aplicaciones para crear, entrenar, visualizar y simular redes neuronales. Se puede realizar clasificación, regresión, clustering, reducción de dimensionalidad y modelado de sistemas dinámicos y de control.

El *toolbox* incluye redes neuronales convolucionales y algoritmos de aprendizaje para clasificación de imágenes y tareas de aprendizaje de características.

MatLab cuenta con una interfaz gráfica para implementar una red neuronal que se abre con el comando *nntool*. Al ejecutar dicho comando se abre la ventana *Network/Data Manager*, la cual permite importar, crear, usar y exportar redes neuronales y datos.

Tabla 3.2: Propiedades del sensor de profundidad configuables con MatLab.

| Propiedad del sensor de profundidad | Descripción |
|-------------------------------------|--|
| <i>Acelerometer</i> | Retorna un vector 3D de datos de aceleración. La información se actualiza mientras el dispositivo está funcionando. |
| <i>BodyPosture</i> | Indica si los esqueletos rastreados están parados o sentados. En modo Standing (parado) detecta 20 articulaciones y en modo Seated (sentado) detecta 10 articulaciones. |
| <i>CameraElevationAngle</i> | Controla el ángulo de sensor y toma valores de -27 a 27 grados. |
| <i>IREmitter</i> | Controla si está encendido o no el emisor de infrarrojo. |
| <i>FrameRate</i> | Cuadros por segundo para la adquisición. |
| <i>TrackingMode</i> | Indica el estado de rastreo. Puede tener tres valores: <i>Skeleton</i> (rastrea todas las articulaciones de los esqueletos), <i>Position</i> (rastrea sólo la posición de la cadera) y <i>Off</i> (deshabilita el rastreo de la posición de los esqueletos). |

3.4.3 Entorno para el diseño de interfaces gráficas de usuario

GUIDE es un entorno de programación visual en MatLab para diseñar interfaces gráficas de usuario (Figura 3.11).

Para acceder a esta herramienta se escribe la siguiente instrucción en la ventana de comandos:

```
>>guide
```

Esta herramienta contiene varios controles para el diseño de la interfaz gráfica de usuario (GUI) y son los siguientes:

- **Check Box:** Indica el estado de una opción o atributo.
- **Editable Text:** Caja para editar texto.
- **Pop-up Menu:** Provee una lista de opciones.

3. TÉCNICAS Y HERRAMIENTAS

- **List Box:** Muestra una lista deslizable.
- **Push Button:** Botón que invoca un evento al presionarlo.
- **Radio Button:** Botón para seleccionar una opción.
- **Toggle Button:** Botón con dos estados, "on" u "off".
- **Slider:** Usado para representar un rango de valores.
- **Static Text:** Etiqueta para mostrar un texto.
- **Panel Button:** Agrupa varios controles.
- **Button Group:** Agrupa varios botones.

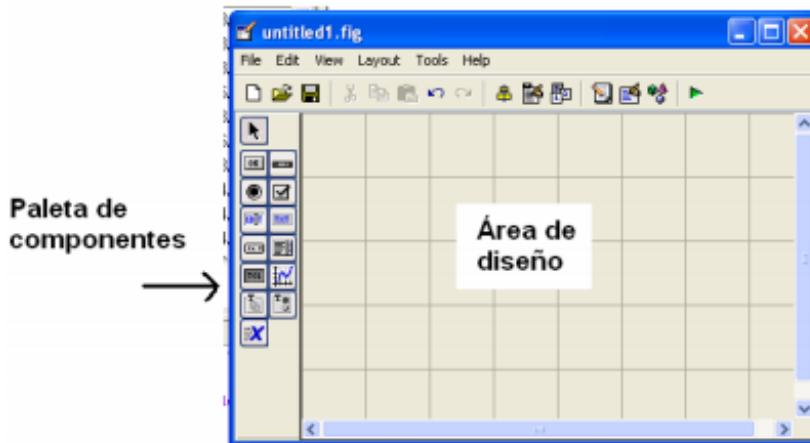


Figura 3.11: Entorno de desarrollo de GUI. (9)

Cada componente tiene propiedades que se pueden modificar de acuerdo a la necesidades del diseñador, para ello se da clic derecho sobre el componente y se elige la opción *Property Inspector*.

Para programar un componente de la GUI se da clic derecho sobre el control deseado y se elige la opción *View Callbacks*. Esta opción abre un archivo en MatLab asociado al GUI y posiciona al usuario en la parte del programa que corresponde a la subrutina que se ejecutará cuando se realice alguna acción sobre el control que se está editando.

3.4.4 Comunicación serial

Para que MatLab establezca comunicación con la tarjeta Arduino que se conecta al un puerto USB de la computadora se emplean las siguientes funciones:

- **serial:** Sirve para crear un objeto de puerto serial especificando el puerto dentro de la función. Ejemplo: `s = serial ('COM5')`.
- **fopen:** Conecta el objeto de puerto serial al dispositivo. Ejemplo: `fopen(s)`.
- **fprintf:** Función para transmitir un dato al dispositivo conectado. Ejemplo: `fprintf(s, '1')`.
- **fscanf:** Función para leer un dato del dispositivo conectado. Ejemplo: `dato = fscanf(s)`.
- **fclose:** Función para desconectar el objeto de puerto serial con el dispositivo. Ejemplo: `fclose(s)`.

3.5 Arduino

Para este proyecto se emplea una tarjeta Arduino como intermediario entre el procesamiento realizado en la computadora y el televisor. A continuación se presentan las características generales de la tarjeta Arduino que se emplea para el prototipo.

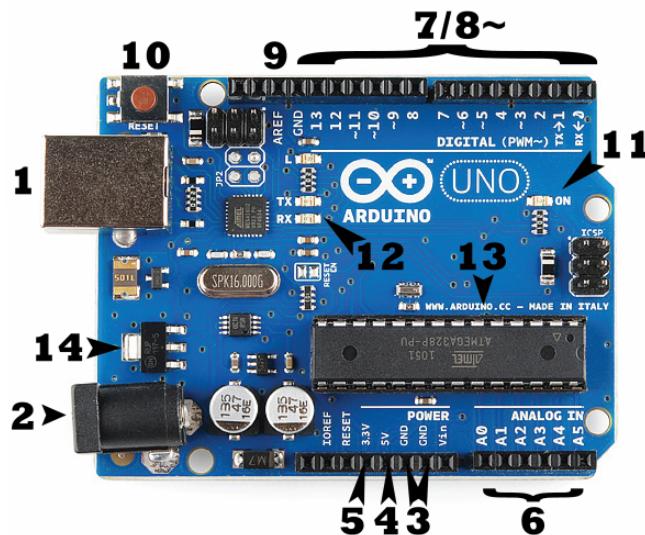


Figura 3.12: Tarjeta Arduino UNO.(9)

3. TÉCNICAS Y HERRAMIENTAS

Arduino es una plataforma de código abierto usada para construir proyectos de electrónica. Consiste en una tarjeta programable que contiene un microcontrolador de la marca ATMEL y un IDE usado para escribir código y cargarlo a la tarjeta.

Existen diferentes modelos de tarjetas Arduino, pero el más popular es el Arduino Uno. En la Figura 3.12 se muestra una placa Arduino Uno y se señalan cada uno de sus componentes, los cuales se describen a continuación:

- **Puerto USB (1):** Sirve para energizar la tarjeta mediante un cable USB.
- **Jack (2):** Sirve para energizar la tarjeta mediante un cable con conector Jack.
- **GND (3):** Pines para conectar la tierra de los circuitos.
- **5V (4) y 3.3V (5):** Pines que suministran voltaje de alimentación, 5V y 3.3V.
- **Entradas analógicas (6):** Estos pines pueden leer la señal de un sensor analógico y convertirlo en un valor digital.
- **Pines digitales (7):** Estos pines pueden ser usados como entradas o salidas digitales.
- **PWM (8):** Estos pines además de funcionar como entradas y salidas digitales convencionales, también pueden ser usados para la Modulación de Ancho de Pulso (PWM).
- **AREF (9):** Pin de referencia analógica, sirve como una referencia externa de voltaje.
- **Botón de *reset* (10):** Reinicia cualquier código cargado al Arduino.
- **LED indicador de alimentación (11):** Se enciende cuando la placa está conectada a la fuente de poder.
- **LEDs TX y RX (12):** Son LEDs indicadores de la comunicación serial se activan dependiendo de si se transmite (TX) o recibe una señal (RX).
- **Circuito Integrado (13):** Chip donde se carga el código escrito en el IDE, usualmente es el modelo ATMEGA328P-PU.

3.6 Protocolos de comunicación para el control remoto infrarrojo

La mayoría de los controles remoto o mandos a distancia para televisor, se comunican con este último por medio de señales infrarrojas, para realizar ciertas

3.6 Protocolos de comunicación para el control remoto infrarrojo

funciones. Motivo por el cual los fabricantes crearon sus propios protocolos para los aparatos que fabrican. Actualmente las compañías usan alguno de los protocolos creados previamente por otras compañías con ligeras modificaciones. Es por ello que en este trabajo, se emplea una tarjeta Arduino que envía comandos al televisor que dependen del los protocolos de comunicación infrarroja. A continuación se presentan los protocolos de comunicación de los que parten los fabricantes de televisores.

3.6.1 Protocolo de comunicación por infrarrojo **SONY Serial InfraRed Control**

El protocolo SIRC (*Serial InfraRed Control*) es un protocolo de control serial creado por la compañía SONY, es uno de los más usados por otras compañías.

Así pues, este protocolo tiene tres diferentes versiones (12, 15, y 20 bits), que comparten características en común, y es que las tres versiones cuentan con bits de comando y bits de dirección, a continuación se mostrarán las características específicas de cada uno. (Figura 3.13)

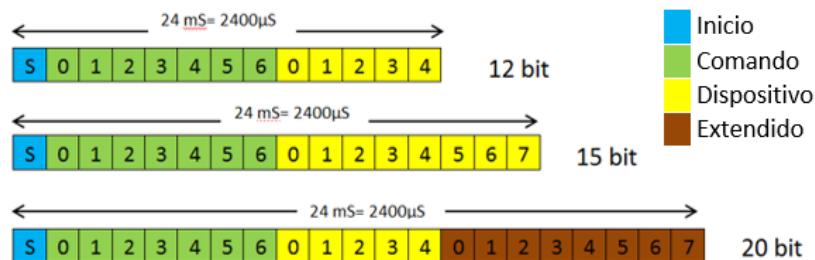


Figura 3.13: Bits que conforman las diferentes versiones del protocolo SONY SIRC.

El protocolo SONY SIRC 12, está conformado por un bit de inicio (Start), siete bits de comando y cinco bits de selección de dispositivo lo que nos da un total de 13 bits.

El protocolo SONY SIRC 15, además de contar con un bit de inicio y siete de comando, tiene otros ocho bits para selección del dispositivo, sumando un total de 15 bits para el tren de pulsos de este protocolo.

El protocolo SONY SIRC 20, tiene cuatro bits para selección de dispositivo y ocho para funciones extendidas, adicionalmente al bit de inicio y los siete de comando, conformando así 20 bits para el tren de pulsos.

Independientemente de la versión que se emplee de este protocolo se usa una modulación por ancho de pulso con una frecuencia de 40kHz para la portadora.

3. TÉCNICAS Y HERRAMIENTAS

La temporización de cada bit o pulso están determinados por múltiplos de un periodo T. Este periodo es de $600\mu\text{s}$. Un estado lógico '1' es representado por dos periodos (2T) en estado alto, es decir $1200\mu\text{s}$ y al terminar este bit se da un espacio de $600\mu\text{s}$ (T) en estado bajo, lo que da un tiempo de transmisión de 3T para el '1' lógico en cualquiera de las posiciones del tren de pulsos.

Un estado lógico '0' solamente está determinado por un periodo de tiempo (T) en estado alto, al terminar este bit se da un espacio de $600\mu\text{s}$ (T), igual que en el caso del estado lógico '1'. Por lo tanto se requiere de un tiempo de $1200\mu\text{s}$ para transmitirlo (Figura 3.14). El tren de pulsos o trama se repite en lapsos de $45\text{m}\text{s}$, por lo menos tres veces.

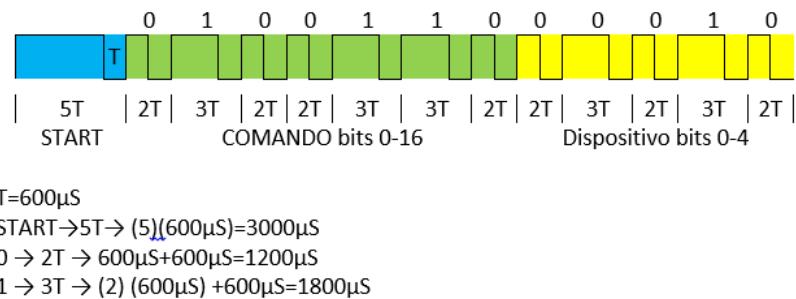


Figura 3.14: Temporización de los bits que conforman las diferentes versiones del protocolo SONY SIRC.

El receptor tiene que esperar un flanco de bajada del pulso inicio. Cuando el receptor recibe este impulso inicia un temporizador de $2500\mu\text{s}$ (este tiempo es menor que 5T pulso start), con lo que garantiza que si existiera otro flanco de bajada quiere decir que no es el inicio de la trama. Pero si no recibe otro flanco de bajada, comienza a recibir los datos en fase. (Figura 3.15)

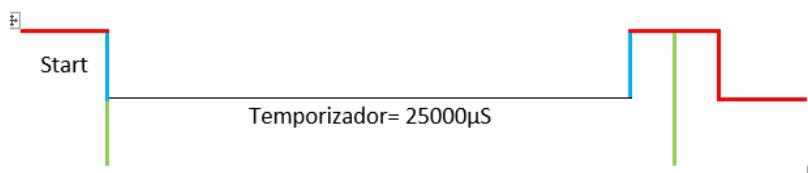


Figura 3.15: Temporización del receptor del protocolo SONY SIRC.

Cabe destacar que la forma de envío de la trama comienza desde el bit menos significativo (LSB), por lo que es necesario invertir el número decimal que tenemos que transmitir.

3.6 Protocolos de comunicación para el control remoto infrarrojo

En la Tabla 3.3 se muestran los comandos predefinidos para el protocolo SIRC.

Tabla 3.3: Comandos predefinidos del protocolo SIRC.

| Comando | Botón presionado |
|----------------|-------------------------|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |
| 8 | 9 |
| 9 | 0 |
| 16 | Canal+ |
| 17 | Canal- |
| 18 | Volumen+ |
| 19 | Volumen- |
| 20 | Mute |
| 21 | Encender/Apagar |
| 22 | Reinicio |
| 23 | Modo de audio |
| 24 | Contraste+ |
| 25 | Contraste- |
| 26 | Color+ |
| 27 | Color- |
| 30 | Brillo+ |
| 31 | Brillo- |
| 39 | Balance derecha |
| 40 | Balance izquierda |
| 47 | Suspendido |

3.6.2 Protocolo de comunicación por infrarrojo Philips RC-5

El protocolo Philips RC-5 es comúnmente empleado por muchos fabricantes. Es un protocolo, creado por la compañía Philips. Este protocolo se basa en una

modulación Manchester de doble fase sobre una portadora de 36 kHz. Los bits ya sea en estado lógico '1' o '0' tienen la misma longitud que es de 1.778mS, o 64 pulsos.

3. TÉCNICAS Y HERRAMIENTAS

En esta codificación el '0' lógico está representado por un tren de pulsos (de la portadora de 36kHz), hasta alcanzar la mitad del bit ($889\mu s$), la siguiente mitad en estado bajo. (Figura 3.16)

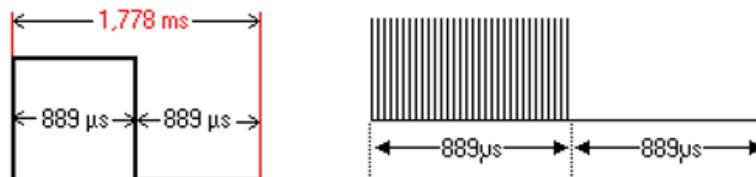


Figura 3.16: Estado lógico '0' en protocolo RC-5

Por el contrario, el '1' lógico está representado por la primera mitad en estado bajo ($889\mu s$), y la segunda mitad por un tren de pulsos en estado alto. (Figura 3.17)

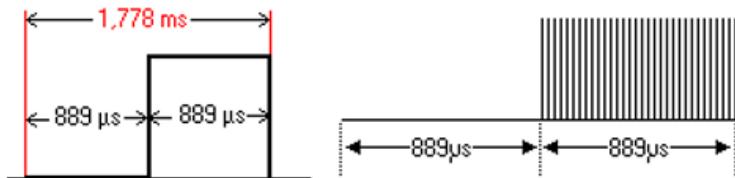


Figura 3.17: Estado lógico '1' en protocolo RC-5

El protocolo RC-5, se encuentra integrado dos bit de inicio (*Start*), , uno de conmutación denominado T, cinco bits de dirección y seis bits de comando, por lo que este protocolo está conformado por catorce bits.



Figura 3.18: Tren de pulsos de protocolo RC-5.

Los dos primeros bits (inicio) siempre deberán permanecer en estado lógico '1'. El tercer bit que es el de conmutación, dependiendo del estado lógico en que

3.6 Protocolos de comunicación para el control remoto infrarrojo

se encuentre, se invierte cada vez que se libera un botón en el control remoto y se le presiona de nuevo. Para que el receptor pueda distinguir cuando un botón está presionado de manera constante, ya que este bit no cambiará su estado, pero si por el contrario, está presionado y liberado de manera repetitiva un mismo botón, este bit cambiará de un estado a otro con cada repetición. (Figura 3.18)

En la Tabla 3.4 se muestran los comandos predefinidos para el protocolo SIRC.

Tabla 3.4: Comandos predefinidos del protocolo RC-5.

| Comando | TV Comando |
|---------|------------|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 12 | Standby |
| 13 | Mute |
| 16 | Volumen+ |
| 17 | Volumen- |
| 18 | Brillo+ |
| 19 | Brillo- |
| 32 | Canal+ |
| 33 | Canal- |

Capítulo 4

Desarrollo del prototipo

4.1 Introducción

En este capítulo se presenta la metodología para desarrollar el sistema de control de televisión mediante gestos. La Figura 4.1 muestra el funcionamiento general del sistema. En la entrada se captura un movimiento mediante Kinect, el proceso es el reconocimiento del gesto realizado mediante una red neuronal artificial de retropropagación y la salida es la ejecución de la función del televisor que le corresponde a dicho gesto, además de la visualización del resultado del reconocimiento en una interfaz gráfica de usuario (GUI).

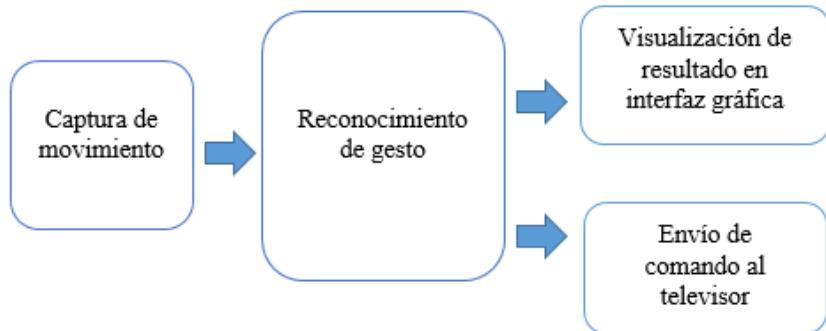


Figura 4.1: Sistema de control de televisión mediante gestos.

Para implementar el algoritmo de reconocimiento de gestos se llevan a cabo tres etapas principales:

1. Creación de una base de datos: En esta etapa se eligen los cinco gestos que controlarán las funciones básicas del televisor. De cada gesto se cap-

4. DESARROLLO DEL PROTOTIPO

turan 500 muestras mediante el sensor Kinect, dando como resultado 2500 patrones en total.

2. Entrenamiento de una RNA: Consiste en introducir a la red el 75% de los patrones de la base creada en la etapa 1, posteriormente mediante un algoritmo de aprendizaje se modifican los parámetros de la red hasta que resuelva efectivamente el problema de la clasificación de movimientos.
3. Simulación de una RNA: Se introduce a la red el 25% restante de las muestras de la base de datos para poner a prueba la capacidad de reconocimiento de gestos.

Cuando se logra el entrenamiento de la red neuronal con la efectividad deseada se integra al sistema de control gestual para que reconozca gestos en tiempo real. Los resultados del reconocimiento se envían a un dispositivo Arduino para que este envíe el comando correspondiente a la televisión por medio de un led infrarrojo. Además, el resultado obtenido de la red neuronal se visualiza en una interfaz gráfica de usuario.

4.2 Creación de una base de datos

4.2.1 Delimitación de los datos

Para el entrenamiento y prueba de la red neuronal es necesaria una base de datos que contenga muestras de los movimientos que controlan el televisor. En la actualidad no existe una base de datos, en la que se encuentren contenidas características numéricas (posiciones en xy) de movimientos corporales. De tal manera que se requiere crear una base de datos que caracterice los movimientos requeridos para este sistema.

En primer lugar se requiere de la selección de los gestos que controlarán el televisor. El sistema de control gestual propuesto en este trabajo, contempla el manejo de tres funciones del televisor, que son encender y apagar el televisor, controlar el nivel de volumen, así como cambiar de canal. Pero en realidad son cinco comandos del televisor, los que se encargan de controlar estas funciones, así que para este proyecto, son cinco movimientos diferentes los que se requieren para poder acceder a cada una de estas funciones. Por lo que se eligieron los siguientes movimientos para controlar las funciones del televisor:

- Aplaudir: Encender o apagar el televisor será por medio de un aplauso o simplemente juntar las manos, aproximadamente a la altura del abdomen (Figura 4.2(a)).
- Subir mano derecha: Se realiza para aumentar la intensidad del volumen (Figura 4.2(b)).
- Subir mano izquierda: Se realiza en caso de querer disminuir el volumen del sonido (Figura 4.2(c)).

4.2 Creación de una base de datos

- Desplazar mano derecha de izquierda a derecha horizontalmente: Este movimiento controla el cambio de canal de manera creciente (Figura 4.2(d)).
- Desplazar mano izquierda de derecha a izquierda horizontalmente: Al realizar este movimiento se realiza un cambio de canal de forma descendente (Figura 4.2(e)).

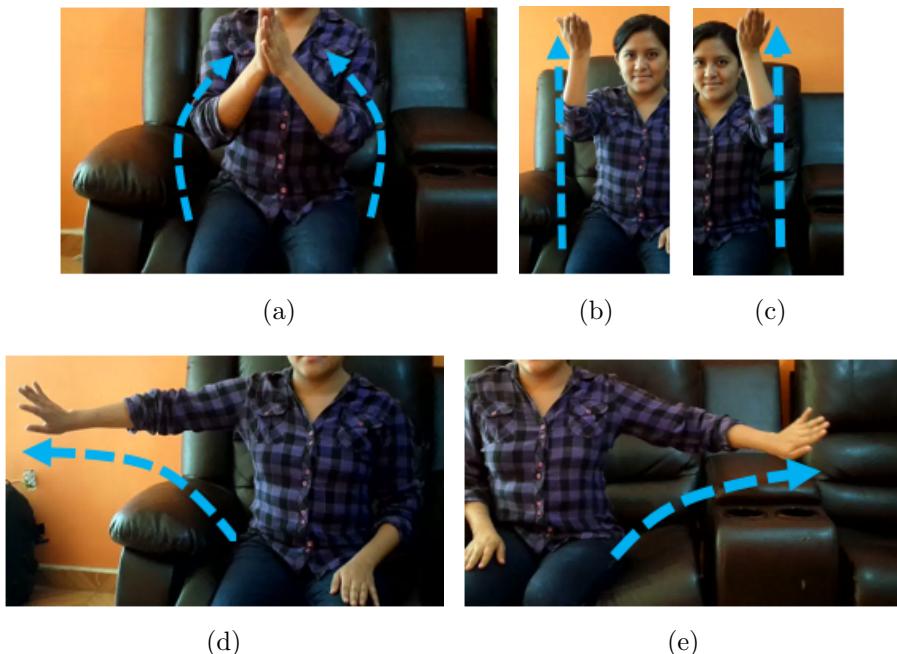


Figura 4.2: Gestos para controlar el sistema: encender/Apagar(a); subir volumen(b); bajar volumen(c); subir canal(d); bajar canal(e).

Los movimientos del usuario son capturados mediante el sensor Kinect. De cada movimiento el sensor obtiene una serie de imágenes de profundidad y coordenadas en xyz de las articulaciones del usuario. Para que los gestos sean clasificados por medio de una red neuronal se extraen las posiciones xy de las articulaciones en los diferentes fotogramas que componen al movimiento. Para este proyecto se toman en cuenta sólo ocho articulaciones que son las siguientes:

1. Hombro derecho
2. Hombro izquierdo
3. Codo izquierdo
4. Codo derecho

4. DESARROLLO DEL PROTOTIPO

5. Muñeca derecha
6. Muñeca izquierda
7. Mano derecha
8. Mano izquierda

Las posiciones de las ocho articulaciones en los diferentes fotogramas que componen el movimiento representan las entradas de la red neuronal, como se ilustra en la Figura 4.3

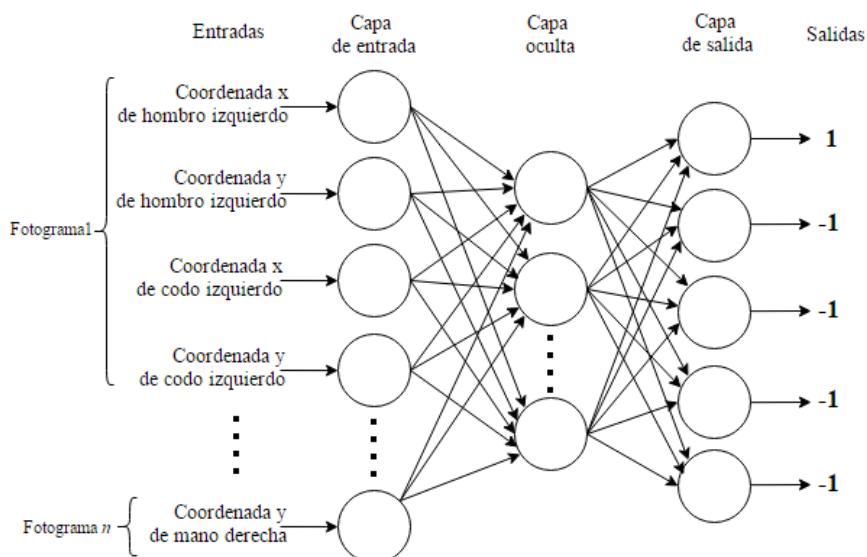


Figura 4.3: Arquitectura de la red neuronal artificial.

La salida de la red neuronal representa la clasificación del patrón ingresado, para este proyecto se tienen cinco neuronas a la salida, que representan los cinco gestos. Las neuronas toman valores en el intervalo [1,-1] (Figura 4.3, para este proyecto cuando una neurona toma un valor mayor o igual a 0.75 se considera 1 y cuando toma un valor menor o igual a -0.75 se considera -1. De acuerdo a estas consideraciones se definen cinco clases (o salidas deseadas de la red) correspondientes a los cinco gestos (Tabla 4.1).

4.2 Creación de una base de datos

Tabla 4.1: Definición de clases

| Gesto | Función del televisor | Clase |
|---|------------------------------|---------------------|
| Aplaudir | Encender o apagar televisión | [1, -1, -1, -1, -1] |
| Subir mano derecha | Subir Volumen | [-1, 1, -1, -1, -1] |
| Subir mano izquierda | Bajar Volumen | [-1, -1, 1, -1, -1] |
| Desplazar mano derecha de izquierda a derecha horizontalmente | Subir canal | [-1, -1, -1, 1, -1] |
| Desplazar mano izquierda de derecha a izquierda horizontalmente | Bajar canal | [-1, -1, -1, -1, 1] |

4.2.2 Adquisición de datos

Para la creación de la base de datos se capturan movimientos de diez personas (voluntarias) mediante el sensor Kinect. El grupo de voluntarios está conformado por niños, adolescentes y adultos de ambos géneros. El rango de estaturas de las personas es de 1m a 1.80m. Cada una de las persona debió realizar cincuenta veces un mismo movimiento, es decir, se obtienen quinientas muestras para cada gesto.

Para la obtención de los datos se requiere el diseño de un programa en MatLab que realice la captura de imágenes de profundidad, así como la adquisición de las coordenadas *xy* de los movimientos realizados por los voluntarios. El algoritmo de captura se muestra en la Figura 4.4 y los pasos se describen a continuación:

1. Configurar Kinect: En virtud de que el sensor Kinect cuenta con emisor y sensor de infrarrojo, se necesita configurar el sensor Kinect para obtener únicamente la imagen de profundidad así como cada uno de los cambios que se producen en la posición de las articulaciones. Para lo cual se crea una entrada de video del sensor de profundidad.
2. Postura: Definir la postura en la que se encontrara el usuario durante el muestreo. En este proyecto se toma en cuenta la postura en modo "Sentado".
3. Muestreo: Definir un periodo durante el cual se realizara el muestreo, que incluye el tiempo que necesita Kinect para detectar a la persona y el tiempo que requiere cada movimiento requiere. En este caso se capturan 25 fotografías para cada muestra.
4. Datos: Por cada movimiento realizado frente al sensor Kinect, se realiza un muestreo, para que obtenga los datos de las coordenadas *xy* de las articulaciones que se movieron.

4. DESARROLLO DEL PROTOTIPO

5. Guardar datos: Guardar los fotogramas da cada muestreo en un archivo diferente, para posteriormente extraer los datos.

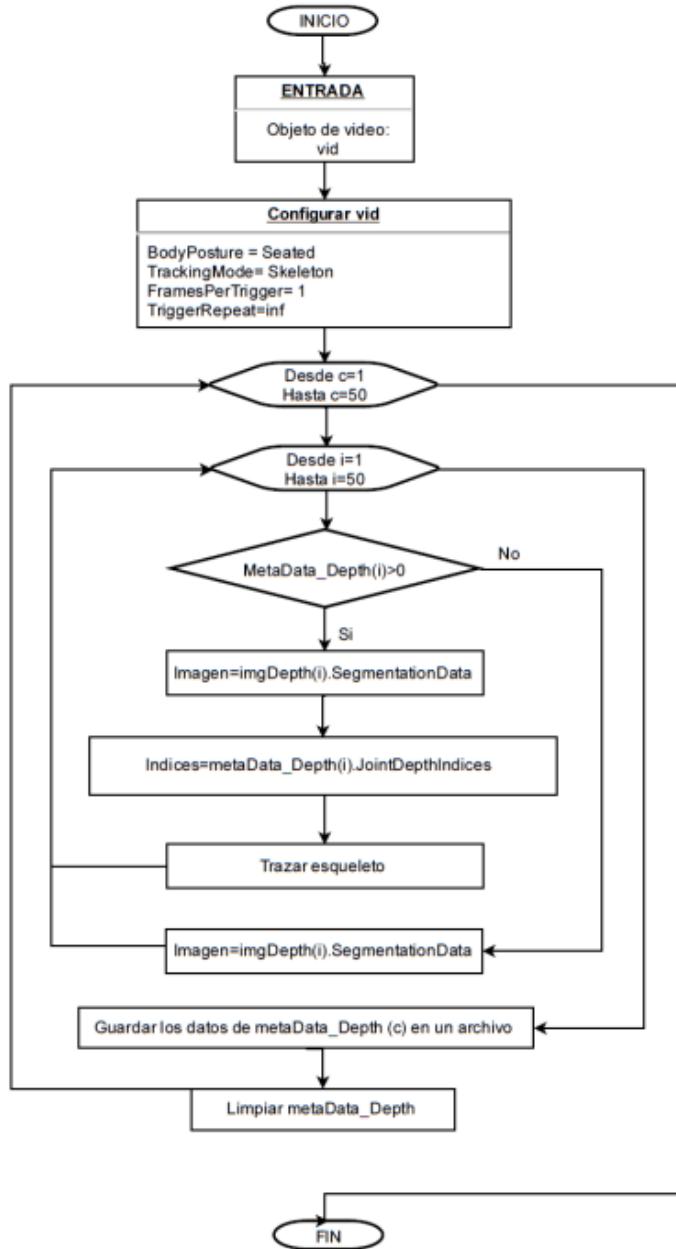


Figura 4.4: Algoritmo para adquisición de datos con Kinect.

4.2.3 Selección de muestras útiles

Después de obtener las capturas de los movimientos realizados por cada uno de los voluntarios, cada una de las muestras debe pasar por un proceso de selección. Esta selección es necesaria para descartar muestras en las que los movimientos no fueron detectados por el sensor Kinect o se ejecutaron de manera incorrecta. Para las muestras correctas, se seleccionan los fotogramas útiles, es decir, aquellos en los que se realizó efectivamente el movimiento.

Para la selección, se realiza una inspección visual de los 25 fotogramas de cada muestra, mediante un programa que fue diseñado para visualizar en pantalla el movimiento. La Figura 4.5 muestra el algoritmo implementado en MatLab para la revisión de las tramas. Cabe mencionar que el programa que se desarrolla siguiendo este algoritmo, se ejecuta para cada uno de los archivos que contienen, los datos y fotogramas de cada muestra.

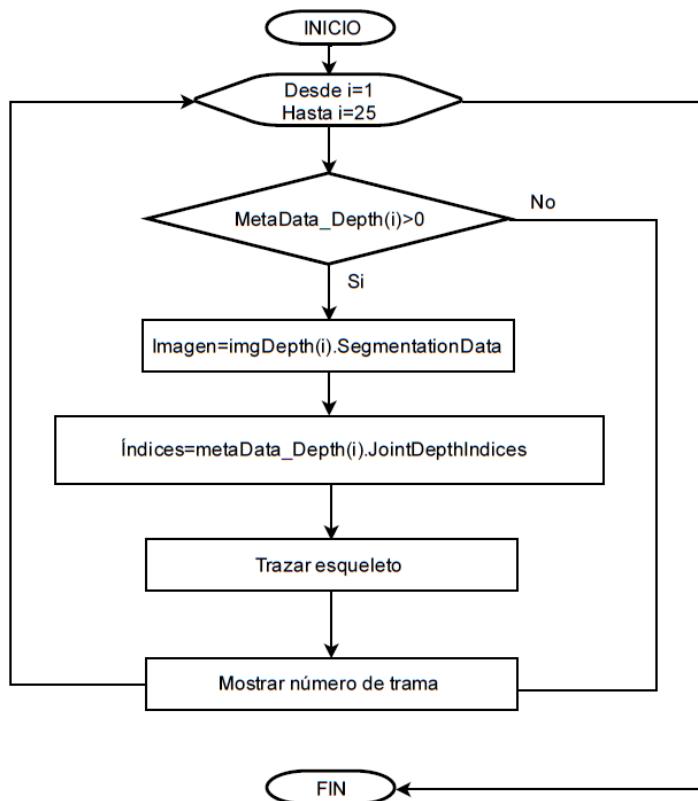


Figura 4.5: Algoritmo para la revisión de muestras de la base de datos.

4. DESARROLLO DEL PROTOTIPO

Por otro lado, cada movimiento requiere de un máximo de cinco tramas para su ejecución, en consecuencia de las veinticinco tramas de cada archivo sólo se toman datos de cinco fotogramas. En la Figura 4.6 Se muestran un ejemplo de los fotogramas útiles de una muestra.

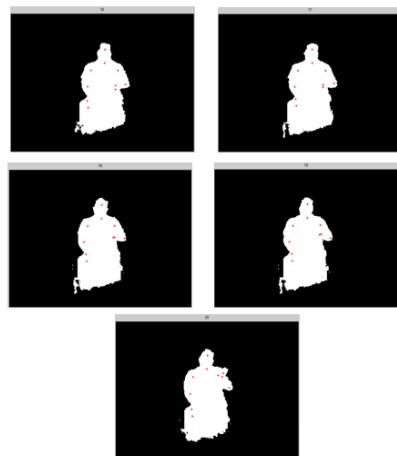


Figura 4.6: Fotogramas útiles para la construcción de la base de datos.

A la par de la visualización se realiza una base de datos auxiliar en Excel la cual contiene, en la primera columna el nombre del archivo de la muestra, la segunda columna contiene el número del fotograma donde comienza el movimiento, y la tercera columna contiene el número del fotograma donde termina el movimiento (Figura 4.7).

The screenshot shows a Microsoft Excel spreadsheet. The ribbon at the top has tabs for Archivo, Inicio, Insertar, Diseño de página, Fórmula, and others. The 'Inicio' tab is selected. The font group shows Calibri 11pt, bold, italic, underline, and alignment options. The formula bar shows 'B1'. The table below has columns labeled A, B, and C. Row 1 is the header with values 1, 501, and 16 respectively. Rows 2 through 7 contain data: (2, 502, 20), (3, 503, 17), (4, 504, 11), (5, 505, 11), (6, 506, 10), and (7, 507, empty). The table has a light gray background and thin black borders between cells.

| | A | B | C |
|---|-----|----|----|
| 1 | 501 | 16 | 20 |
| 2 | 502 | | |
| 3 | 503 | 17 | 21 |
| 4 | 504 | 7 | 11 |
| 5 | 505 | 7 | 11 |
| 6 | 506 | 6 | 10 |
| 7 | 507 | | |

Figura 4.7: Construcción de base de datos de fotogramas útiles.

4.2.4 Extracción de las coordenadas xy .

La base de datos auxiliar se usa como apoyo para la extracción de las coordenadas xy de las articulaciones de los hombros, codos, muñecas y manos de cada fotograma. Con estos valores numéricos se genera una base de datos requerida para el entrenamiento y simulación de la red neuronal. Se crea un archivo en Excel como el de la Figura 4.8 que está estructurado de la siguiente manera:

- 2500 columnas que corresponden a los 2500 patrones capturados.
- Cada columna contiene ochenta filas que representan las coordenadas en metros de las articulaciones xy de los cinco fotogramas o tramas durante los que se realiza un gesto.
- Las filas nones son las coordenadas x de cada articulación.
- Las filas pares son las coordenadas y de cada articulación.

| A | B | C |
|----|--------------------|--------------|
| 1 | | subv1 |
| 2 | Hombro izquierdo x | 0.2553 |
| 3 | Hombro izquierdo y | 0.1838 |
| 4 | Codo izquierdo x | 0.2554 |
| 5 | Codo izquierdo y | 0.2103 |
| 6 | Muñeca izquierda x | -0.0469 |
| 7 | Muñeca izquierda y | 0.0458 |
| 8 | Mano izquierda x | -0.0772 |
| 9 | Mano izquierda y | -0.1532 |
| 10 | Hombro derecho x | -0.0715 |
| 11 | Hombro derecho y | -0.3123 |
| 12 | Codo derecho x | -0.0696 |
| 13 | Codo derecho y | -0.3653 |
| 14 | Muñeca derecha x | 0.2086 |
| 15 | Muñeca derecha y | 0.0455 |
| 16 | Mano derecha x | 0.2753 |
| 17 | Mano derecha y | -0.0306 |
| 18 | Hombro izquierdo x | 0.2553 |

Figura 4.8: Archivo de la base de datos total.

La extracción de las coordenadas xy se realiza mediante comandos de MatLab para leer y escribir en Excel, empleando el algoritmo de la Figura 4.9

4. DESARROLLO DEL PROTOTIPO

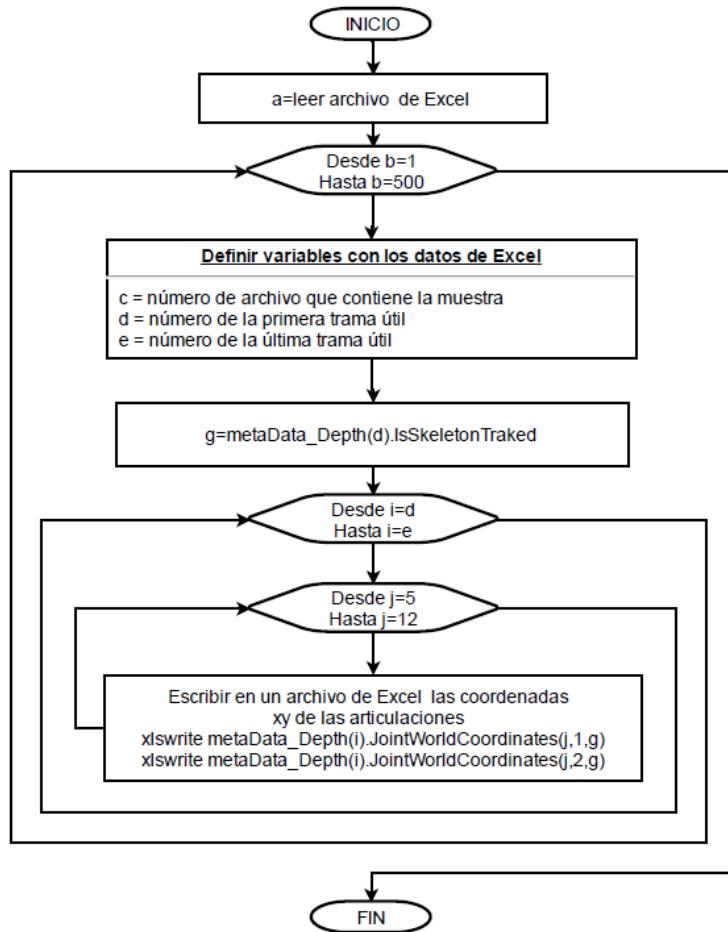


Figura 4.9: Algoritmo exportar base de datos de MatLab a Excel.

4.2.5 Bases de datos para entrenamiento y simulación

La base de datos total se divide en dos, una para entrenamiento y otra para simulación de la red neuronal artificial. Para la base de datos de entrenamiento se elige el 75% de las muestras y se pasa a una hoja de cálculo. La hoja de cálculo queda de 1875 columnas por 80 filas.

En otra hoja de cálculo se escriben las clases a las que pertenecen los patrones de entrenamiento; es decir una tabla que contenga las salidas deseadas para cada patrón que se ingrese a la red neuronal, la tabla generada es de 1875 columnas por 5 filas (Figura 4.10).

Con el 25% de las muestras restantes se integra otra base de datos que servirá para la fase de simulación de la red neuronal. La fase de simulación también

4.3 Creación de una red neuronal artificial

| | A | B | C | D | E | F | |
|---|----|----|----|----|----|----|----|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 3 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 4 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 5 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

Figura 4.10: Archivo de las clases de la base de datos.

requiere que se le especifique las salidas que se espera obtener para cada patrón que ingresa, por lo que de igual manera se realizó una base de datos en la que contenga las salidas esperadas para cada patrón de entrada.

4.3 Creación de una red neuronal artificial

4.3.1 Configuración inicial

La red neuronal empleada para la clasificación de los patrones, es una red neuronal de Perceptrón Multicapa entrenada con un algoritmo de retropropagación.

El diseño de la red neuronal basa la cantidad de neuronas en la capa de entrada en el número de atributos de los patrones de entrada y la de salida en el número de clases; quedando así una red neuronal de 80 neuronas de entrada que corresponden a los 80 atributos de los patrones de la base de datos creada y 5 neuronas de salida que corresponden a las salidas deseadas para cada patrón.

Con el apoyo de la herramienta nntool de MatLab, se configura la red neuronal hasta alcanzar el objetivo de clasificar la mayoría de los patrones de la base de datos. A continuación se describe el procedimiento que se sigue para alcanzar dicho objetivo.

En el programa MatLab se cargan las bases de datos con los patrones que funcionan para entrenamiento y las que son para la simulación. Para ello se utiliza la instrucción de lectura de un archivo de Excel en MatLab, que es `xlsread`, la cual se usó de la siguiente manera:

```
entradas_1= xlsread ('1_75primeras.xlsx','A1:BTC80');
salidas_1= xlsread('1_75p_clases.xlsx','A1:BTC5');
entradas_2= xlsread('1_25ultimas.xlsx','A1:XA80');
salidas_2= xlsread('1_25u_clases.xlsx','A1:XA5');
```

En el workspace de MatLab se escribe la instrucción `nntool`.

Con esta instrucción se despliega la ventana del toolbox de MatLab destinado para redes neuronales (Figura 4.11).

4. DESARROLLO DEL PROTOTIPO

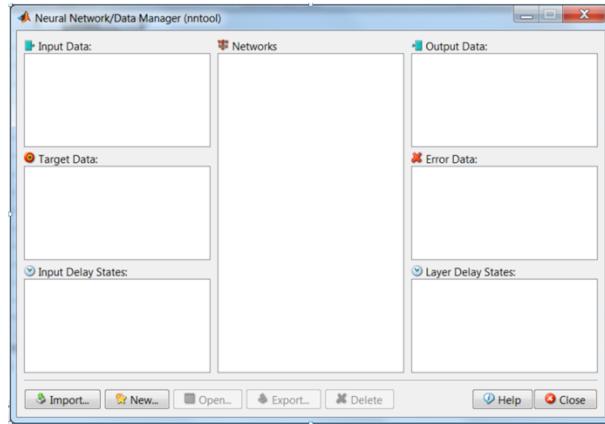


Figura 4.11: Neural Network/Data Manager de la herramienta *nntool*.

A continuación se importan del *Workspace* de MatLab las bases de datos pulsando el botón *Import* del Neural Network/Data Manager. Este botón abre la ventana *Import to Network/Data Manager* (Figura 4.12).

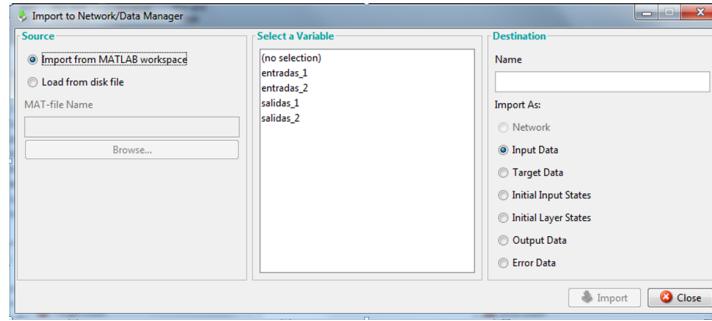


Figura 4.12: Ventana para importar bases de datos a *nntool*.

Las entradas son seleccionadas como *Input Data* e importadas y las salidas son seleccionadas como *Target Data*. Se da clic nuevamente en el botón *Import*. Luego de realizar esta importación se cierra la ventana *Import to Network/Data Manager* para continuar con la creación de la red neuronal.

Para la creación de la red se configuran diferentes aspectos (Figura 4.13):

- Selección de entradas y salidas deseadas.
- La capa oculta es de 100 neuronas para que la red neuronal alcance una linealidad en la clasificación.
- Se selecciona la función tangente hiperbólica como función de transferencia, debido a que las salidas están dadas en valores de 1's y -1's.

4.3 Creación de una red neuronal artificial

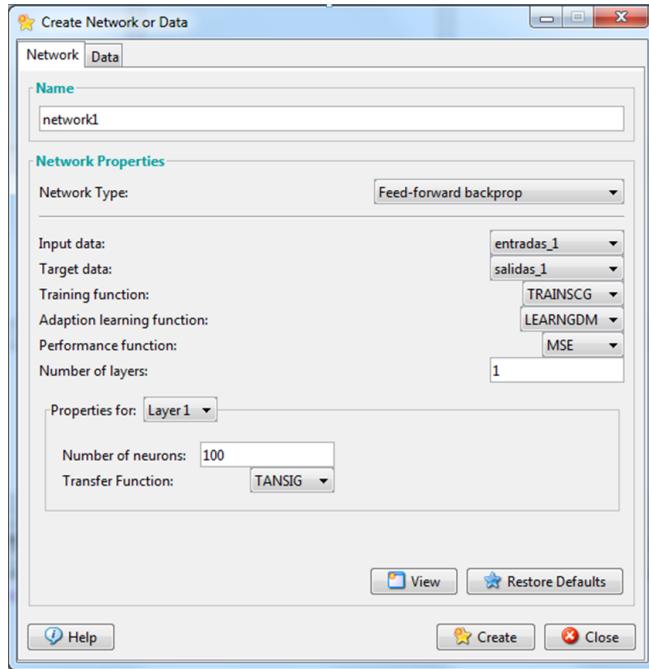


Figura 4.13: Creación de una red neuronal artificial.

La arquitectura de la red queda constituida como el la Figura 4.14.

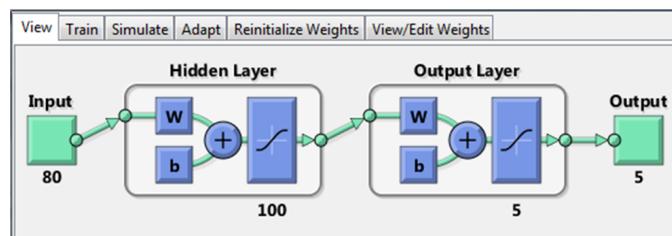


Figura 4.14: Arquitectura de la red neuronal artificial.

4.3.2 Fase de entrenamiento

El entrenamiento se realiza al presionar la pestaña *Train*, para cambiar los parámetros necesarios para el entrenamiento (Figura 4.15).

La pestaña *Train* contiene dos subdivisiones las cuales deben ser configuradas (Figura 4.15). En la subdivisión *Train Info* se realiza lo siguiente:

4. DESARROLLO DEL PROTOTIPO

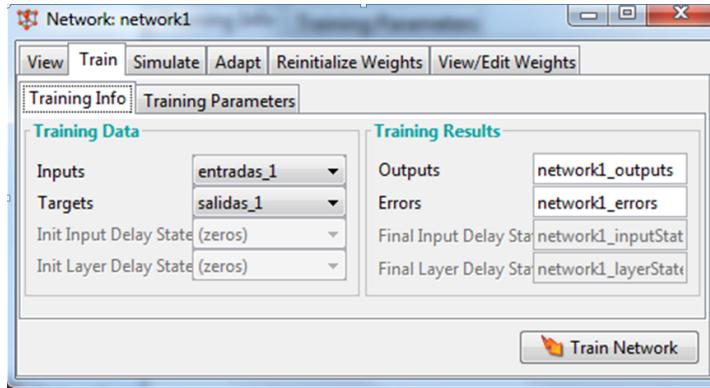


Figura 4.15: Ventana para entrenamiento de la red neuronal artificial.

- Para *Inputs* se elige la base que contiene el 75% de las muestras.
- Para *Targets* la base de datos de clasificación que le corresponde.

La siguiente subdivisión se usa para cambiar los parámetros con los que va a entrenar la red neuronal (Figura 4.16). Los parámetros épocas y *max-fail* se configuran con un valor de 100000, este último valor es el número de comprobaciones acertadas que tiene la red. Además de estos parámetros se configura el error a la salida de la red para que sea el mínimo y en consecuencia la red detenga el entrenamiento al alcanzar este valor.

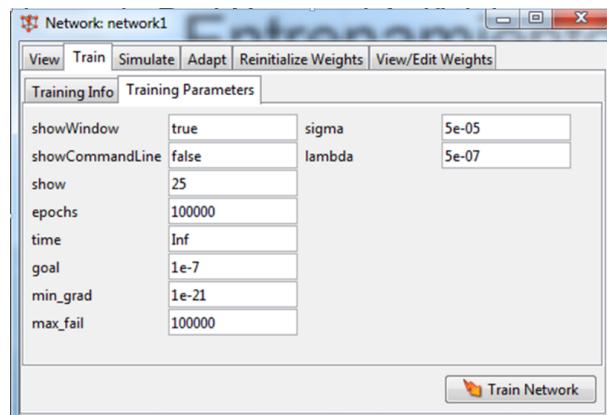


Figura 4.16: Configuración de parámetros de entrenamiento.

Luego de configurar estos parámetros se presiona el botón *Train Network* para dar inicio al entrenamiento de la red. La red neuronal comienza el entrenamiento y se detiene hasta alcanzar alguno de los valores configurados con anterioridad, el

4.3 Creación de una red neuronal artificial

parámetro por medio del cual nos interesa que la red neuronal detenga el entrenamiento es el denominado como *Performance* (Figura 4.17). Para llegar al resultado deseado se crean y entranan varias redes neuronales, variando los parámetros de entrenamiento y el número de neuronas en la capa oculta (Ver Apéndice B).

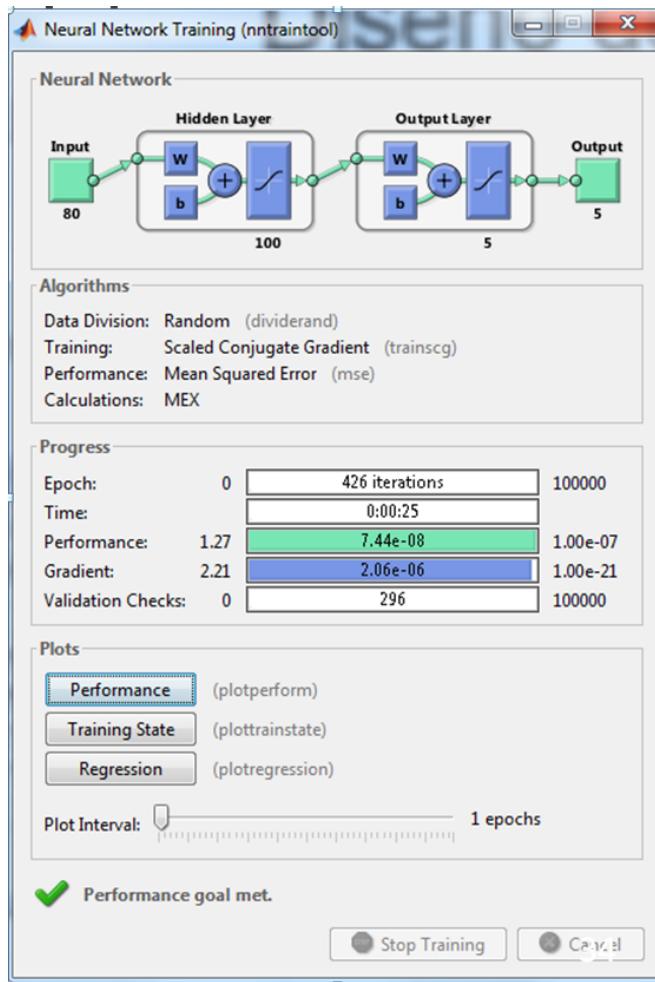


Figura 4.17: Resultado del entrenamiento de la red neuronal artificial.

4.3.3 Fase de simulación o recuperación

Cuando el entrenamiento de la red neuronal alcanza un error a la salida igual o menor al deseado, se procede a la fase de simulación. Esta simulación se hace dando clic en la pestaña *Simulate*, en la cual se seleccionan las entradas y salidas deseadas. Posteriormente se presiona el botón *Simulate Network* (Figura 4.18).

4. DESARROLLO DEL PROTOTIPO

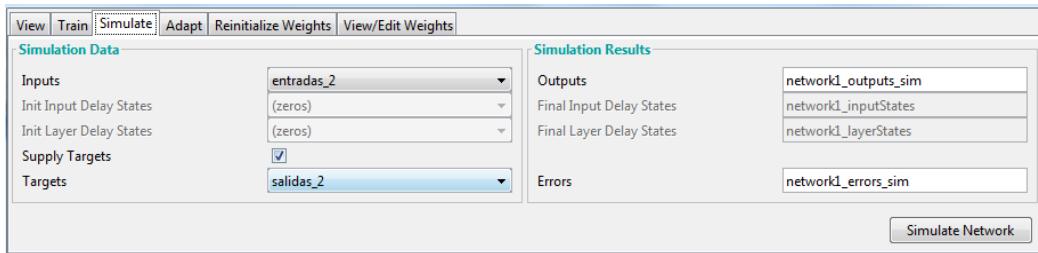


Figura 4.18: Ventana *Simulate* para la simulación de la red neuronal.

A continuación, se exportan los resultados del entrenamiento y la red neuronal entrenada al *Wokspace* de MatLab, para trabajar con ellos posteriormente (Ver Apéndice B).

4.4 Diseño de sistema de control gestual para televisor

4.4.1 Diseño de una interfaz gráfica de usuario



Figura 4.19: Interfaz gráfica del sistema de control gestual para televisor.

Una vez entrenada la red neuronal, se integra a un sistema que sea capaz de reconocer gestos en tiempo real y de ejecutar las correspondientes funciones del televisor. Para que el usuario tenga un fácil manejo del sistema se diseña una interfaz gráfica, que permita la activación o desactivación del mismo (Figura 4.19). La implementación de la interfaz se realiza con la herramienta GUIDE de MatLab.

La interfaz contiene los siguientes elementos:

- Vista del Kinect: En caso de que el sistema se encuentre activado, proporciona una visualización de las imágenes de profundidad que el Kinect captura en tiempo real. Además contiene una etiqueta para indicar el estado de captura del sensor. Existen tres estados:
 1. "Sistema apagado": Mensaje que se muestra cuando el sensor Kinect no se encuentra adquiriendo datos (Figura 4.20(a)).
 2. "No se detecta movimiento": Se despliega este mensaje cuando el sensor Kinect se encuentra activo pero no detecta datos de esqueleto, ya sea porque el usuario no se encuentra frente al Kinect o porque

4. DESARROLLO DEL PROTOTIPO

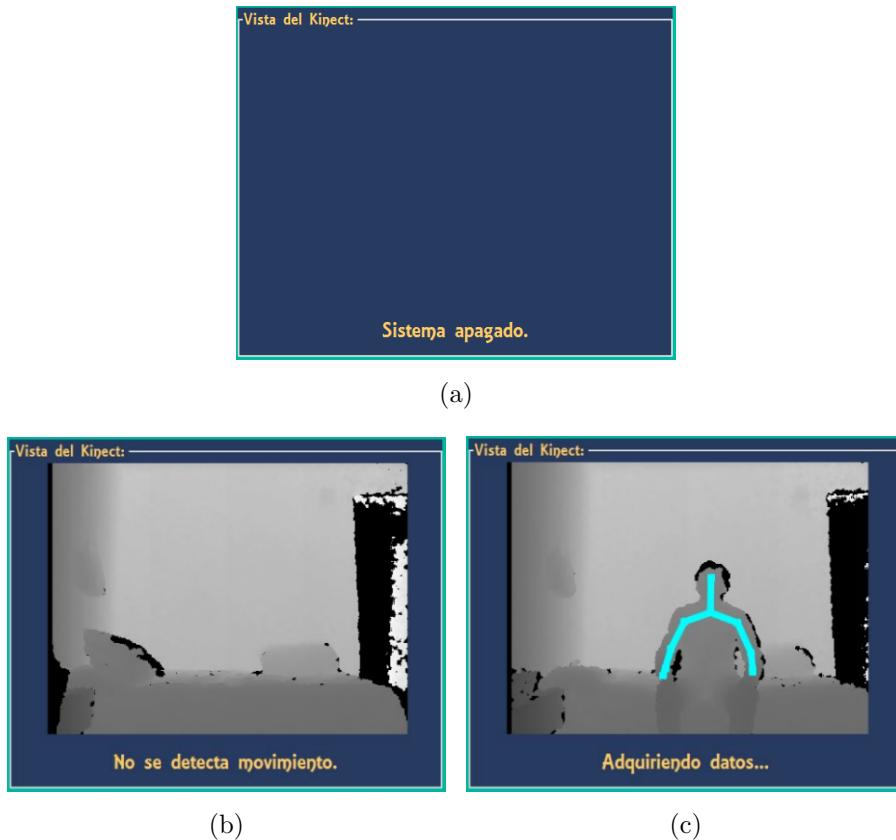


Figura 4.20: Vista del Kinect cuando el sistema está apagado (a); vista del Kinect cuando no se detecta movimiento (b); vista del Kinect cuando inicia el reconocimiento de gestos (c).

se encuentra frente al Kinect sin realizar movimiento alguno (Figura 4.20(a)).

3. ”Adquiriendo datos”: Esta etiqueta se muestra cuando Kinect detecta las articulaciones de algún usuario, los que indican que el reconocimiento de gestos se encuentra activo. Cuando se presenta este caso, se traza el esqueleto del usuario sobre la imagen de profundidad. (Figura 4.20(a)).
- Función detectada: Contiene seis etiquetas que simulan a los LEDs que sirven de indicadores de las funciones reconocidas. Los diferentes casos se presentan en la Figura 4.21.
 - Iniciar sistema: Botón que inicia la captura y reconocimiento de gestos.
 - Detener sistema: Botón que detiene la captura y el reconocimiento de gestos.
 - Instrucciones: Botón que abre una nueva ventana donde se explica la manera correcta de realizar los gestos para controlar las funciones básicas del televisor (Figura 4.22).

4.4 Diseño de sistema de control gestual para televisor

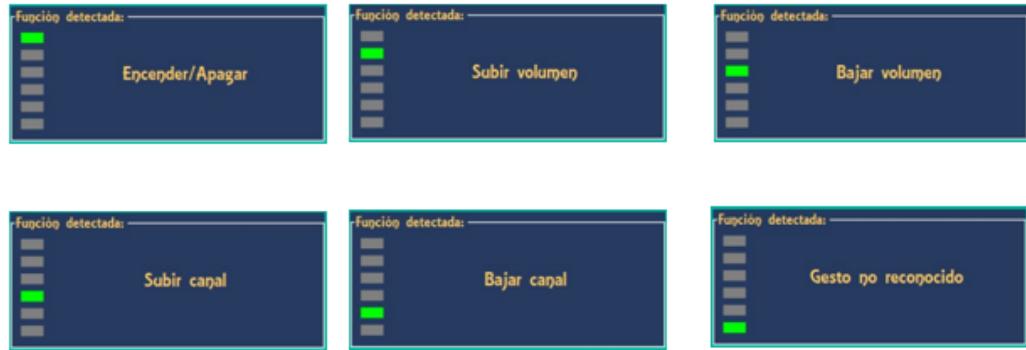


Figura 4.21: Diferentes casos de la sección "Función detectada" de la interfaz de usuario.

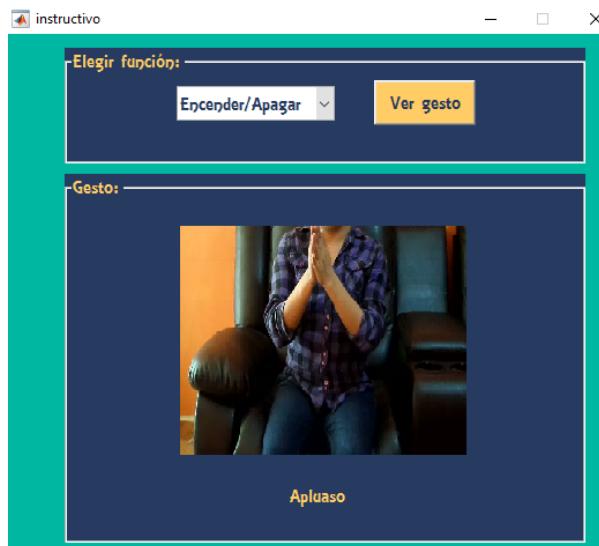


Figura 4.22: Instructivo de gestos para controlar TV.

4.4.2 Algoritmo del sistema de control gestual para televisor

Cuando el usuario inicia el sistema (mediante el botón "Iniciar sistema" de la GUI), se manda a llamar una función en MatLab que realiza la captura y reconocimiento de gestos para el control de un televisor. El funcionamiento del algoritmo se muestra en el diagrama de la Figura 4.23.

4. DESARROLLO DEL PROTOTIPO

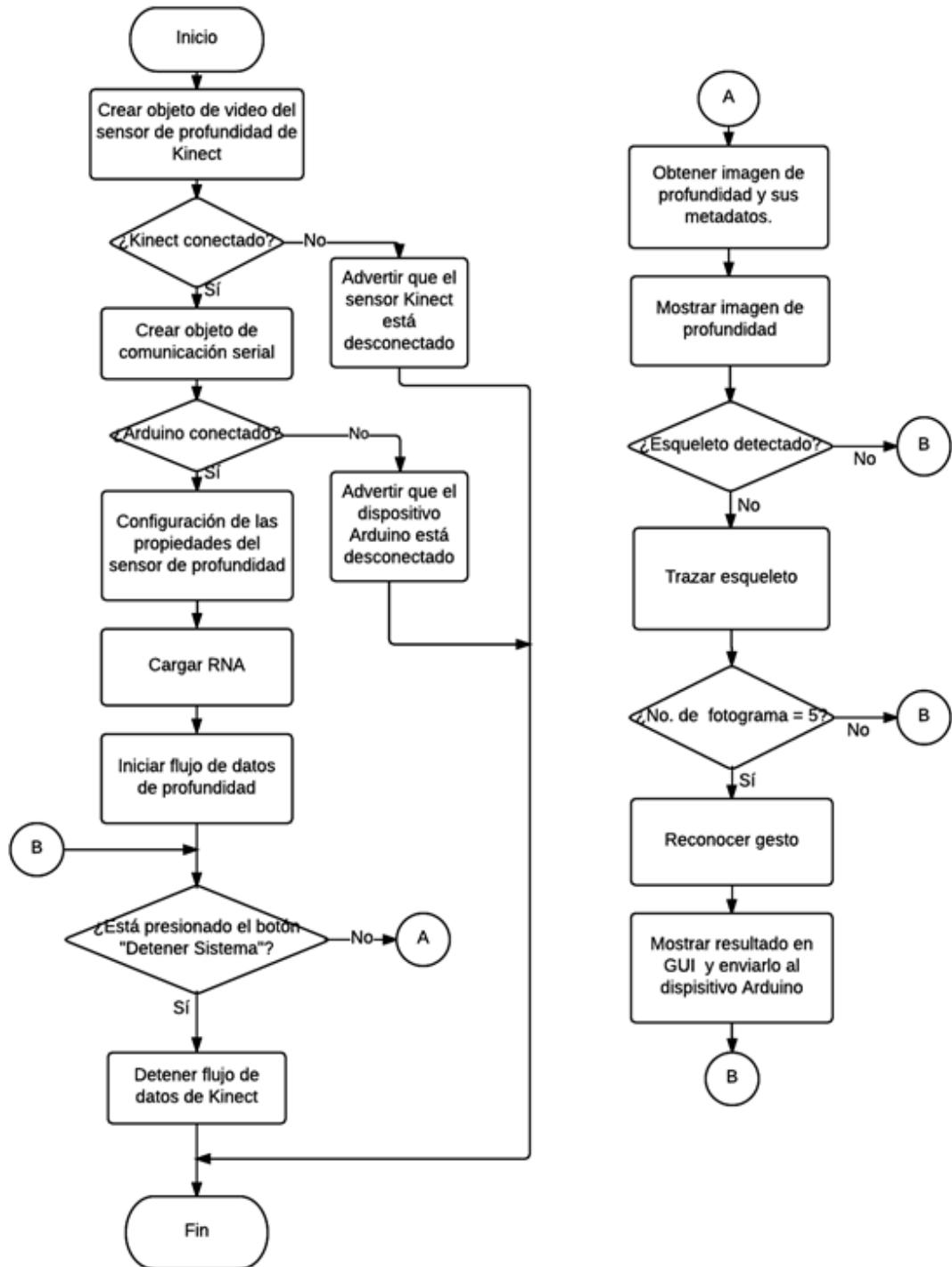


Figura 4.23: Algoritmo del sistema de control gestual para televisor.

4.4 Diseño de sistema de control gestual para televisor

A continuación se explican los pasos más relevantes del algoritmo del sistema.

A) Configuración inicial

Para el correcto funcionamiento del sistema, se deben configurar los dispositivos de entrada (Kinect) y salida (Arduino) antes de comenzar la detección y reconocimiento de gestos. En primer lugar se debe crear un objeto de entrada de video de Kinect. En este caso se elige una entrada de video del sensor de profundidad para obtener los datos de las articulaciones del usuario.

```
vid= videoinput('kinect',2);
```

Es importante verificar que el sensor Kinect se encuentre conectado a la computadora. En caso de que no se detecte un sensor se envía un mensaje al usuario para que realice la conexión del dispositivo (Figura 4.24)

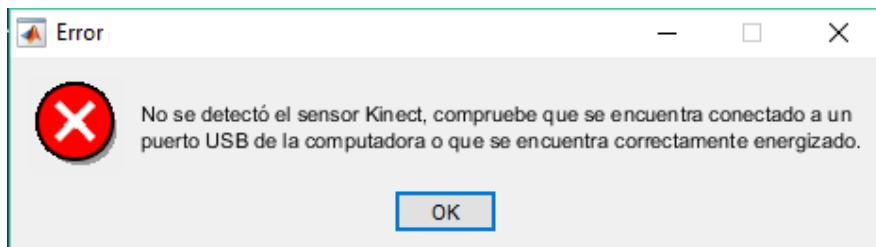


Figura 4.24: Mensaje de error cuando el Kinect no se encuentra conectado.

Se deben configurar las propiedades del sensor de profundidad para obtener la adquisición de los datos del esqueleto del usuario. En este caso la postura del cuerpo (**BodyPosture**) se configura en modo sentado (**Seated**) y para el modo de seguimiento (**TrackingMode**) se elige la opción de rastreo de las articulaciones del esqueleto (**Skeleton**).

Una vez finalizada la configuración del sensor Kinect, se procede a configurar el objeto de entrada del dispositivo Arduino que envía las señales de infrarrojo al televisor para su control.

```
obj1 = serial('COM5');
fopen(obj1);
```

De la misma forma que se realiza con el sensor Kinect, se debe verificar que el dispositivo se encuentre conectado y enviar una notificación al usuario en caso de que el dispositivo esté desconectado.

4. DESARROLLO DEL PROTOTIPO

B) Adquisición de datos de profundidad

Después de haber configurado los dispositivos de entrada y salida y de inicializar las variables necesarias, comienza la captura de datos con el sensor Kinect para ello se inicia el flujo de datos de profundidad:

```
start(vid);
```

Mientras que el usuario no presione el botón "Detener sistema" de la GUI, el sensor Kinect inicia una continua adquisición de imágenes (tramas) y sus correspondientes metadatos. Los datos se obtienen en un arreglo, a su vez los metadatos se van guardando en una estructura para que posteriormente sean usados en el reconocimiento de gestos:

```
[imgDepth, ~ , metaData_Depth(i)] = getdata(vid);
```

Las imágenes obtenidas se emplean para mostrar al usuario la vista del Kinect. En cada captura se sobrescriben las imágenes, no se guardan pues no son necesarias para su procesamiento, sólo los metadatos derivados de ellas.

Si el sensor detecta a un usuario en movimiento (`skeletonTracked`), se guardan las coordenadas de las articulaciones (`JointWorldCoordinates`). Por cada cinco tramas continuas en las que se hayan obtenido los datos del esqueleto se genera un vector columna llamado muestra. Este vector tiene la información de las coordenadas de las 8 articulaciones consideradas para el reconocimiento de gestos. Dicha información se ordena en el vector de la misma manera que se ordenó en los patrones de la base de datos, para que posteriormente sea usado como vector de entrada de la RNA.

C) Reconocimiento de gestos

Cada vez que la variable muestra se actualiza, es decir, cada vez que se guarda en ella información sobre cinco nuevas tramas capturadas por el Kinect, se procede al reconocimiento de gestos mediante la RNA. Para ello, el vector muestra se le presenta a la RNA como patrón de entrada de la siguiente forma:

```
y=network1(muestra);
```

En la variable *y* se obtiene un vector que corresponde a la salida de la RNA. Cada uno de los elementos del vector se compara con los elementos de las clases establecidas previamente, que son:

```
Encender/Apagar -> [1 -1 -1 -1 -1]
Subir volumen -> [-1 1 -1 -1 -1]
Bajar volumen -> [-1 -1 1 -1 -1]
Subir canal -> [-1 -1 -1 1 -1]
Bajar canal -> [-1 -1 -1 -1 1]
```

Los elementos del vector *y* no siempre resultan en valores de 1 y -1, por lo que debe aplicarse un criterio de tolerancia para que dichos elementos se consideren

4.4 Diseño de sistema de control gestual para televisor

dentro de la clasificación establecida. De acuerdo a la teoría de la función de activación tangencial hiperbólica, se aplica el siguiente criterio de tolerancia:

- Si elemento de $y > 0.75$ se considera como un 1.
- Si elemento de $y <-0.75$ se considera un -1.

Por ejemplo, cuando se compara con la clase Encender/Apagar, se realiza lo siguiente:

```
if y(1)>0.75 && y(2)<-0.75 && y(3)<-0.75 && y(4)<-0.75  
&& y(5)<-0.75  
    set(handles.funcion,'String','Encender/Apagar');  
    set(handles.led1,'BackgroundColor',[0 1 0]);  
    fprintf(obj1, '1');
```

De acuerdo a la línea de código anterior, si se cumple la condición se enciende el led y cambia el texto de la etiqueta de la interfaz gráfica según corresponda. Además se envía al dispositivo Arduino un carácter que sirve de auxiliar para activar el envío de una señal infrarroja correspondiente a la función detectada.

Si y no coincide con ninguna de las clases de las funciones del televisor, se envía un mensaje en la interfaz para indicar que ningún gesto ha sido reconocido.

D) Envío de comandos al televisor

La última etapa del sistema consiste en enviar al televisor señales de infrarrojo para controlar sus funciones. Para ello se emplea un dispositivo Arduino, al que se le instala la biblioteca **IRremote** creada por Ken Shirriff, la cual permite recibir y transmitir códigos de Control Remoto Infrarrojo.

En primer lugar, deben obtenerse los códigos necesarios para controlar el televisor. En este trabajo se decodifican las señales de infrarrojo del control LG AKB74475433. A continuación se presentan los pasos para copiar los códigos de dicho control remoto:

- Paso 1. Se carga el programa **IRrecvDump** incluido en la biblioteca **IRremote** al dispositivo Arduino.
- Paso 2. Se arma un circuito que reciba las señales de infrarrojo del control remoto (Figura 4.25). En este caso se emplea el módulo receptor de infrarrojos VS1838B.
- Paso 3. Se abre el Monitor Serie del IDE de Arduino para que cuando se reciban las señales del control remoto se visualicen en pantalla los códigos requeridos.
- Paso 4. Se apunta el control remoto en dirección del receptor de infrarrojos.

4. DESARROLLO DEL PROTOTIPO

- Paso 5. Se presionan las teclas del control en el siguiente orden:
 1. Encender/Apagar
 2. Subir volumen
 3. Bajar volumen
 4. Subir canal
 5. Bajar canal
- Paso 6. En el monitor serie se obtiene el protocolo bajo el que trabaja el control remoto y los comandos en formato hexadecimal de 32 bits que corresponden a cada una de las funciones. (Figura 4.26)

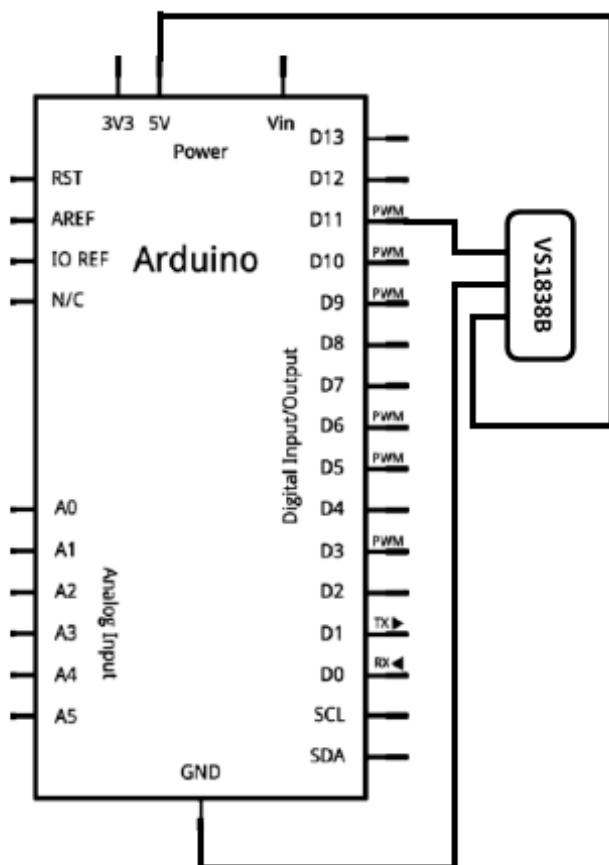


Figura 4.25: Circuito receptor de códigos de un control remoto.

4.4 Diseño de sistema de control gestual para televisor

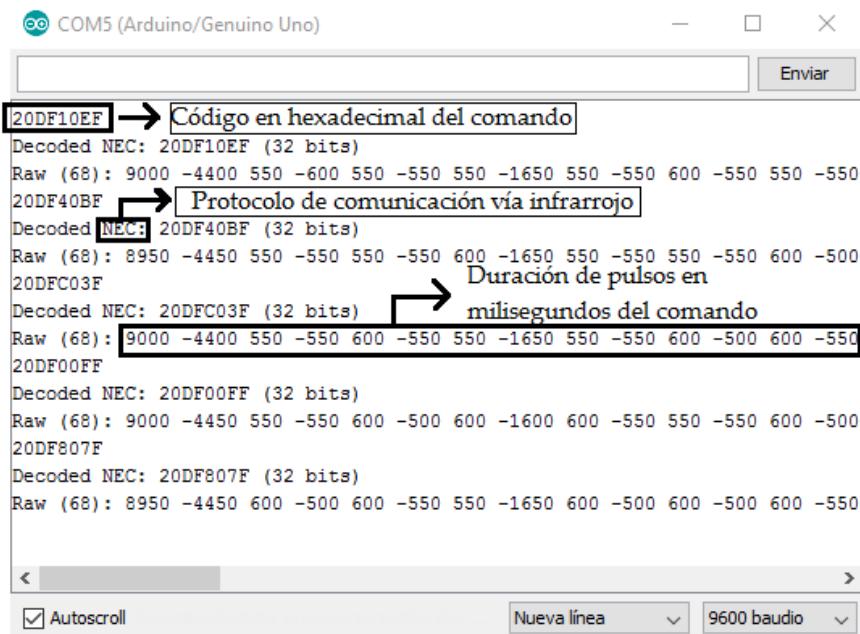


Figura 4.26: Obtención de los comandos de un control remoto.

Una vez obtenidos los códigos del control, se programa el dispositivo Arduino para que mediante un led infrarrojo, envíe señales al televisor. Adicionalmente se agregan seis LEDs, que corresponden a los LEDs de la GUI y sirven como indicadores de la función detectada. El algoritmo para controlar el televisor con Arduino se presenta a continuación:

- Paso 1. Se declara un objeto de la clase `IRsend` que forma parte de la biblioteca `IRremote`.

```
IRsend irsend;
```

- Paso 2. Se inicializan las variables y se realizan las configuraciones iniciales del dispositivo, es decir, se configuran las salidas y la velocidad de transmisión en la comunicación serial.

```
Serial.begin(9600); //configurar velocidad de transmisión
```

- Paso 3. Se lee constantemente lo que llega al receptor de la comunicación serial para determinar si se envía algún comando al televisor.

```
mensaje = Serial.read(); //lectura del buffer del receptor
```

4. DESARROLLO DEL PROTOTIPO

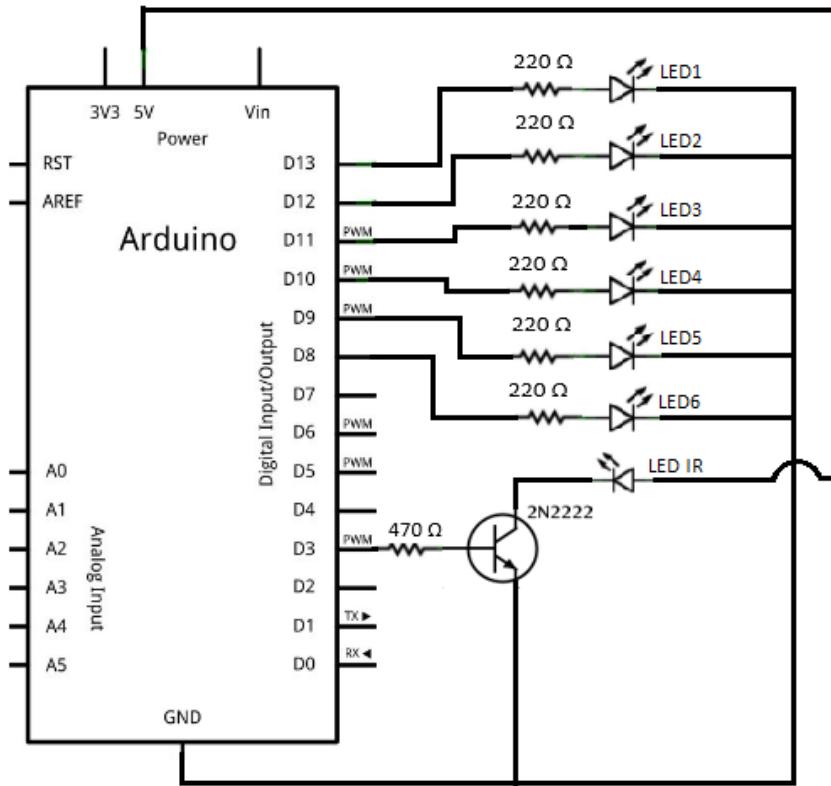


Figura 4.27: Circuito para el control de un televisor.

- Paso 4. En una estructura de tipo switch, se busca alguna coincidencia de mensaje con diferentes casos.

```

switch(mensaje)
{
case '1':
    /* Código para encender led1 y apagar o encender
    TV. Se envía un comando al televisor mediante
    un led infrarrojo. La señal enviada está dada
    por un código en hexadecimal de 32 bits. */

    irsend.sendNEC(0x20DF10EF,32);

    /* Se enciende el primer led y se apaga el resto
    para indicar que se ha detectado la función
    encender/apagar. */

    digitalWrite(led1,HIGH);
    digitalWrite(led2,LOW);
}

```

4.4 Diseño de sistema de control gestual para televisor

```
        :  
        digitalWrite(led6,LOW);  
        break;  
  
    case '2': //Código para encender led2 y subir volumen  
    case '3': //Código para encender led3 y bajar volumen  
    case '4': //Código para encender led4 y subir canal  
    case '5': //Código para encender led5 y bajar canal  
    case 'e': /*Código para encender led6  
    No se reconoce gesto, por lo que no se  
    envía un comando al televisor */  
}
```

El programa se carga al dispositivo Arduino y se arma el circuito de la Figura 4.27. Posteriormente el circuito se coloca apuntando hacia el televisor para que éste reciba correctamente los comandos.

Capítulo 5

Resultados Experimentales

5.1 Introducción

Una vez integrados todos los elementos del sistema gestual para televisor, el prototipo se sometió a una serie de pruebas para evaluar su funcionalidad bajo diferentes condiciones.

Para evaluar la efectividad del prototipo se realizaron diversos experimentos en los que se puso a prueba el reconocimiento de gestos considerando dos variables principales:

- La distancia del usuario al Kinect.
- La posición del usuario en el campo de visión horizontal del sensor (eje *x*).

5.2 Definición de los variables y experimentos

Se realizaron pruebas del reconocimiento de siete gestos, de los cuales cinco son los que están definidos para el control del televisor y dos que son gestos ajenos al sistema. Estos últimos se eligieron para probar la capacidad de la red de descartar movimientos que no corresponden a una función del televisor. Los dos movimientos elegidos para este fin son: estirar ambos brazos hacia los lados y estirar ambos brazos hacia arriba. Los siete gestos fueron realizados por 5 voluntarios en diversos experimentos.

Se realizaron pruebas variando la posición del usuario con respecto al Kinect (Figura 5.1). Se tomaron en cuenta cuatro distancias al Kinect 1.5m, 2m, 2.5m y 3m; además para cada una de esas distancias se consideraron tres posiciones con respecto al campo visual horizontal del sensor: centro, eje *x* positivo y eje *x* negativo. Para las distancias del eje *x* no se establecieron medidas exactas, sólo se cuidó que el voluntario se colocara sobre tres diferentes puntos del eje *x*.

5. RESULTADOS EXPERIMENTALES

Para cada posición (punto) de la Figura 5.1. se realizaron 10 repeticiones de cada gesto.

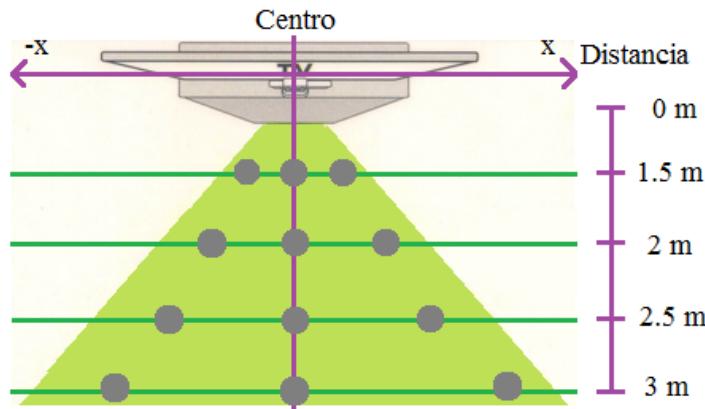


Figura 5.1: Posiciones del usuario consideradas para los experimentos.

5.3 Resultados

En la Tabla 5.1 se muestra el número de gestos reconocidos correctamente por el sistema para diferentes posiciones del usuario. Los gestos se representan de la siguiente manera:

- Encender/Apagar = G1
- Subir volumen = G2
- Bajar volumen = G3
- Subir canal = G4
- Bajar canal = G5
- Estirar brazos hacia los lados = G6
- Estirar los brazos hacia arriba = G7

Las posiciones con respecto al eje x se representan de la siguiente manera:

- I = Posición en el eje x negativo.
- D = Posición en el eje x positivo.
- C = centro del eje x.

5.3 Resultados

Tabla 5.1: Número de gestos reconocidos en diferentes posiciones.

| Distancia en metros | Posición en eje x | G1 | G2 | G3 | G4 | G5 | G6 | G7 |
|---------------------------|----------------------|-----|-----|-----|-----|-----|-----|-----|
| 1.5 | C | 8 | 8 | 10 | 10 | 10 | 10 | 10 |
| | I | 10 | 9 | 10 | 10 | 10 | 10 | 9 |
| | D | 9 | 9 | 10 | 10 | 9 | 10 | 9 |
| 2 | C | 10 | 10 | 10 | 9 | 10 | 10 | 10 |
| | I | 8 | 9 | 9 | 10 | 9 | 9 | 9 |
| | D | 10 | 9 | 10 | 9 | 10 | 9 | 9 |
| 2.5 | C | 9 | 8 | 8 | 10 | 10 | 10 | 9 |
| | I | 9 | 7 | 9 | 10 | 9 | 9 | 10 |
| | D | 10 | 8 | 8 | 9 | 10 | 9 | 8 |
| 3 | C | 8 | 7 | 10 | 9 | 9 | 10 | 9 |
| | I | 10 | 8 | 8 | 8 | 8 | 9 | 8 |
| | D | 9 | 8 | 9 | 10 | 9 | 10 | 10 |
| Total: | | 110 | 100 | 109 | 113 | 113 | 114 | 110 |

En total se realizaron 840 gestos de los cuales se reconocieron efectivamente el 91.55%. De los cinco gestos asignados para el control del televisor, los que tuvieron un mayor porcentaje de efectividad fueron "Subir canal" (G4) y "Bajar Canal" (G5) con un con 94.17%, por otro lado el que presentó menor porcentaje de efectividad fue "Subir Volumen" (G2) con un 83.33 %. Los gestos que no corresponden al sistema de control fueron descartados en un porcentaje mayor al 90% (Figura 5.2).

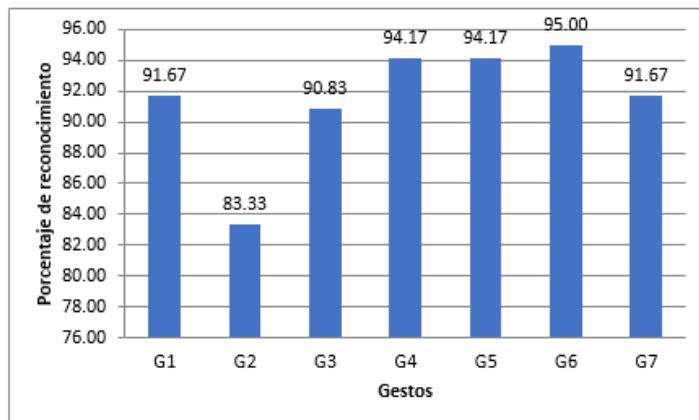


Figura 5.2: Porcentajes de reconocimiento para siete gestos.

5. RESULTADOS EXPERIMENTALES

Los gestos realizados a 1.5m y 2m del Kinect fueron reconocidos en un 93.81%, mientras que el sistema presentó menor efectividad cuando el usuario se colocó a 3m del sensor, con un 88.57% de gestos reconocidos (Figura 5.3).

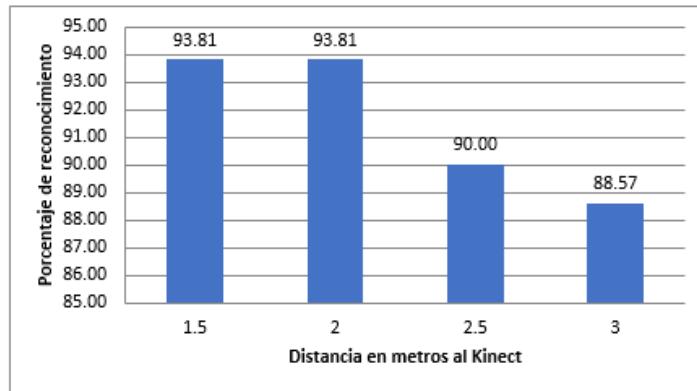


Figura 5.3: Porcentajes de reconocimiento en diferentes distancias al sensor.

En cuanto a las posiciones en el eje x , los gestos realizados en el centro del eje presentaron un mayor porcentaje de reconocimiento con un 93.21% (Figura 5.4).

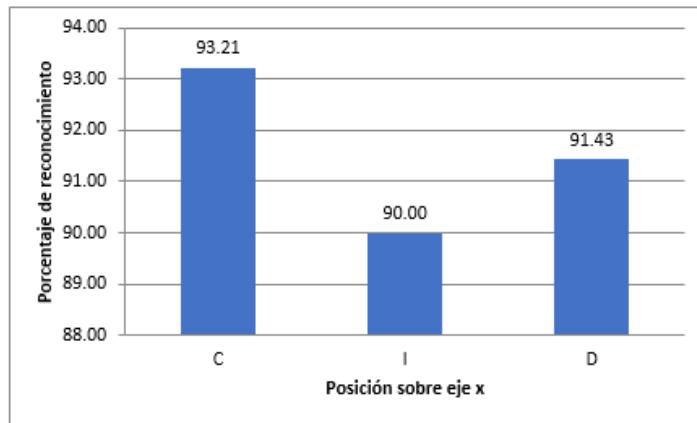


Figura 5.4: Porcentajes de reconocimiento en diferentes posiciones del eje x.

Además de los experimentos anteriores se realizó un estudio de costos del proyecto, el cual se puede verificar en el Anexo C.

Capítulo 6

Conclusiones

6.1 Conclusiones Finales

En base al desarrollo del sistema de control gestual para televisor propuesto en este proyecto se concluye lo siguiente:

- Se examinaron los trabajos relacionados con el sistema propuesto.
- Se definieron y explicaron las técnicas y herramientas empleadas en el desarrollo del prototipo como lo son Kinect, redes neuronales y Arduino.
- Se creó una base de datos de cinco gestos para el entrenamiento y prueba de una red neuronal. Posteriormente la red neuronal entrenada, se integró a un sistema para controlar un televisor mediante reconocimiento de gestos.
- Se evaluó el funcionamiento del sistema de control gestual en diferentes puntos del campo visual del sensor Kinect.

En base a las pruebas, se determinó que el sistema de control de televisión mediante Kinect y redes neuronales tiene un 91.55% de efectividad en el reconocimiento de gestos. Este porcentaje de reconocimiento contempla tanto el correcto reconocimiento de los cinco gestos asignados para controlar las funciones básicas del televisor como la capacidad de descartar gestos que no corresponden al control del sistema. Además, para alcanzar el mayor porcentaje de efectividad, el usuario debe sentarse frente al Kinect a una distancia de entre 1.5m a 2m.

6.2 Contribuciones

Algunas contribuciones del desarrollo del proyecto son:

- Una base de datos de cinco movimientos conformada por las posiciones del usuario en coordenadas xy .

6. CONCLUSIONES

- Ofrece un sistema de control gestual que se adapta a una televisión sin la necesidad de alterarla.

6.3 Trabajo Futuro

El sistema de control gestual para televisor se puede optimizar de la siguiente manera:

- Depurar y ampliar la base de datos para mejorar el reconocimiento de los gestos.
- Ampliar el número de funciones de televisor que se puedan controlar mediante el sistema.
- Reducir el gasto computacional del sistema.
- Modificar el sistema para que sea capaz de controlar cualquier modelo de televisor.

Apéndice A

Código

A.1 Creación de la base de datos

A.1.1 Captura

```
vid= videoinput('kinect',2);
src = getselectedsource(vid);
src.BodyPosture = 'Seated';
src.TrackingMode = 'Skeleton';
vid.FramesPerTrigger = 1;
triggerconfig(vid,'manual');

for c = 1:50
a= 0;
start(vid);
for i = 1:25
    trigger(vid)
    [imgDepth, ts_depth, metaData_Depth(i)] = getdata(vid);
    imshow(imgDepth,[0 4096]);
    title(['\fontsize{24}',num2str(metaData_Depth(i).FrameNumber),
' , captura ',num2str(c)]);

    if sum(metaData_Depth(i).IsSkeletonTracked)>0
        a = a + 1;
        skeletonJoints = metaData_Depth(i).JointDepthIndices(:, :, ...
metaData_Depth(i).IsSkeletonTracked);
        hold on;
        if(a==1)
            plot(skeletonJoints(:,1),skeletonJoints(:,2),'*', 'Color', 'g');
        else
            plot(skeletonJoints(:,1),skeletonJoints(:,2),'*', 'Color', 'r');
```

A. CÓDIGO

```
    end
    hold off;
    end
end
stop(vid);
save(strcat('C:\Users\Sil\Documents\MATLAB\base_de_datos\...
enc-ap\enc',num2str(c),'.mat'),'metaData_Depth');
clear metaData_Depth;
close;
pause(2);
end
```

A.1.2 Revisión

```
load('C:\Users\Sil\Documents\MATLAB\base_de_datos\enc-ap\enc1')
utilpath = fullfile(matlabroot, 'toolbox', 'imaq', 'imaqdemos', ...
'html', 'KinectForWindows');
addpath(utilpath);

for i=1:25
    if sum(metaData_Depth(i).IsSkeletonTracked)>0
        trackedSkeletons = find(metaData_Depth(i).IsSkeletonTracked);
        jointIndices = metaData_Depth(i).JointDepthIndices(:, :, ...
            metaData_Depth(i).IsSkeletonTracked);
        image = metaData_Depth(i).SegmentationData;
        nSkeleton = length(trackedSkeletons);
        util_skeletonViewer(jointIndices, image, nSkeleton);
        title(num2str(metaData_Depth(i).FrameNumber));
    else
        imshow(metaData_Depth(i).SegmentationData)
        title(num2str(metaData_Depth(i).FrameNumber));
        pause(0.15);
    end
end
close;
```

A.1.3 Extracción de coordenadas

```
a = xlsread('C:\Users\Sil\Documents\MATLAB\base_de_datos\
            base_aux.xlsx','enc','A1:C500');
for b=1:500
    c=a(b,1);
    d=a(b,2);
    e=a(b,3);
load(strcat('C:\Users\Sil\Documents\MATLAB\base_de_datos\
```

```

enc-ap\enc',num2str(c),'.mat'));
g=find(metaData_Depth(d).IsSkeletonTracked,1);
for i=d:e
    for j=5:12
        muestra(2*(j-4)-1+16*(i),b)= ...
        metaData_Depth(i).JointWorldCoordinates(j,1,g);
        muestra(2*(j-4)+16*(i),b)= ...
        metaData_Depth(i).JointWorldCoordinates(j,2,g);
    end
end
clear metaData_Depth;
end
xlswrite('C:\Users\Sil\Documents\MATLAB\base_de_datos\enc',...
    muestra,'Hoja1');

```

A.2 Interfaz

A.2.1 Menú principal

```

function varargout = controlTV(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name',          mfilename, ...
                       'gui_Singleton',   gui_Singleton, ...
                       'gui_OpeningFcn', @controlTV_OpeningFcn, ...
                       'gui_OutputFcn',  @controlTV_OutputFcn, ...
                       'gui_LayoutFcn',  [], ...
                       'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] =
            gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end

    function controlTV_OpeningFcn(hObject, eventdata, handles, varargin)
        handles.output = hObject;
        guidata(hObject, handles);

    function varargout = controlTV_OutputFcn(hObject, eventdata, handles)
        varargout{1} = handles.output;

```

A. CÓDIGO

```
function inicio_Callback(hObject, eventdata, handles)
handles.detener_ahora = 0;
guidata(hObject,handles);
try
vid= videoinput('kinect',2);
catch ME
    if(strcmp(ME.identifier,'imaq:videoinput:noDevices'))
        msgbox('No se detectó el sensor Kinect, compruebe que se
        encuentra conectado a un puerto USB de la computadora
        o que se encuentra correctamente energizado.',...
        'Error','error');
    end
    return
end

obj1 = instrfind('Type', 'serial', 'Port', 'COM5', 'Tag', '');
if isempty(obj1)
    obj1 = serial('COM5');
else
    fclose(obj1);
    obj1 = obj1(1);
end
try
fopen(obj1);
catch ME
    if(strcmp(ME.identifier,'MATLAB:serial:fopen:opfailed'))
        msgbox('No se detectó el dispositivo Arduino. Verifique que
        se encuentra conectado a un puerto USB de la computadora.',...
        'Error','error');
    end
    return
end

src = getselectedsource(vid);
src.BodyPosture = 'Seated';
src.TrackingMode = 'Skeleton';
vid.FramesPerTrigger = 1;
vid.TriggerRepeat = inf;
triggerconfig(vid,'manual');
load('net1.mat')
frames=1;
ON=false;
muestra=zeros(80,1);

start(vid);
```

```

while ~(handles.detener_ahora)
    trigger(vid)
    [imgDepth, ~ , metaData_Depth(frames)] = getdata(vid);
    imshow(imgDepth,[0 4096]);
    skeletonTracked=find(metaData_Depth(frames).IsSkeletonTracked,1);
    if isempty(skeletonTracked)
        frames=1;
        set(handles.mensaje,'String','No se detecta movimiento.');
    else
        hold on;
        skeletonJoints = metaData_Depth(frames).JointDepthIndices(:, :, ...
            metaData_Depth(frames).IsSkeletonTracked);
        for joints=5:12
            muestra(2*(joints-4)-1+16*(frames-1),1)= ...
                metaData_Depth(frames).JointWorldCoordinates(joints, ...
                    1,skeletonTracked);
            muestra(2*(joints-4)+16*(frames-1),1)= ...
                metaData_Depth(frames).JointWorldCoordinates(joints, ...
                    2,skeletonTracked);
        end
        set(handles.mensaje,'String','Adquiriendo datos...');

        line([skeletonJoints(3,1) skeletonJoints(4,1)],...
            [skeletonJoints(3,2) skeletonJoints(4,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(3,1) skeletonJoints(5,1)],...
            [skeletonJoints(3,2) skeletonJoints(5,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(3,1) skeletonJoints(9,1)],...
            [skeletonJoints(3,2) skeletonJoints(9,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(5,1) skeletonJoints(6,1)],...
            [skeletonJoints(5,2) skeletonJoints(6,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(6,1) skeletonJoints(7,1)],...
            [skeletonJoints(6,2) skeletonJoints(7,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(7,1) skeletonJoints(8,1)],...
            [skeletonJoints(7,2) skeletonJoints(8,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(9,1) skeletonJoints(10,1)],...
            [skeletonJoints(9,2) skeletonJoints(10,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');
        line([skeletonJoints(10,1) skeletonJoints(11,1)],...
            [skeletonJoints(10,2) skeletonJoints(11,2)],'LineWidth',...
                6,'Marker','+','Color', 'c');

```

A. CÓDIGO

```
line([skeletonJoints(11,1) skeletonJoints(12,1)],...
[skeletonJoints(11,2) skeletonJoints(12,2)],'LineWidth',...
6,'Marker', '+','Color', 'c');
frames=frames+1;
if frames>5
y=network1(muestra);
set(handles.led1,'BackgroundColor',[0.502 0.502 0.502]);
set(handles.led2,'BackgroundColor',[0.502 0.502 0.502]);
set(handles.led3,'BackgroundColor',[0.502 0.502 0.502]);
set(handles.led4,'BackgroundColor',[0.502 0.502 0.502]);
set(handles.led5,'BackgroundColor',[0.502 0.502 0.502]);
set(handles.led6,'BackgroundColor',[0.502 0.502 0.502]);
if y(1)>0.75 && y(2)<-0.75 && y(3)<-0.75 ...
&& y(4)<-0.75 && y(5)<-0.75
ON=~ON;
set(handles.funcion,'String','Encender/Apagar');
set(handles.led1,'BackgroundColor',[0 1 0]);
fprintf(obj1, '1');
elseif y(1)<-0.75 && y(2)>0.75 && y(3)<-0.75...
&& y(4)<-0.75 && y(5)<-0.75
set(handles.funcion,'String','Subir volumen');
set(handles.led2,'BackgroundColor',[0 1 0]);
fprintf(obj1, '3');
elseif y(1)<-0.75 && y(2)<-0.75 && y(3)>0.75...
&& y(4)<-0.75 && y(5)<-0.75
set(handles.funcion,'String','Bajar volumen');
set(handles.led3,'BackgroundColor',[0 1 0]);
fprintf(obj1, '4');
elseif y(1)<-0.75 && y(2)<-0.75 && y(3)<-0.75...
&& y(4)>0.75 && y(5)<-0.75
set(handles.funcion,'String','Subir canal');
set(handles.led4,'BackgroundColor',[0 1 0]);
fprintf(obj1, '5');
elseif y(1)<-0.75 && y(2)<-0.75 && y(3)<-0.75...
&& y(4)<-0.75 && y(5)>0.75
set(handles.funcion,'String','Bajar canal');");
set(handles.led5,'BackgroundColor',[0 1 0]);
fprintf(obj1, '6');
else
set(handles.funcion,'String','Gesto no reconocido');
set(handles.led6,'BackgroundColor',[0 1 0]);
fprintf(obj1, 'e');
end
frames=1;
pause(1);
```

```

        end
    hold off;
end
handles = guidata(hObject); %Get the newest GUI data
clear metaData_Depth;
end
stop(vid);
cla(handles.axes1);
cla(handles.uibuttongroup1);
set(handles.mensaje,'String','Sistema apagado.');
fclose(obj1);

function detener_Callback(hObject, eventdata, handles)
handles.detener_ahora = 1;
guidata(hObject, handles);

function figure1_CloseRequestFcn(hObject, eventdata, handles)
handles.detener_ahora = 1;
guidata(hObject, handles);
imaqreset;
delete(hObject);

function instrucciones_Callback(hObject, eventdata, handles)
instructivo

```

A.2.2 Instrucciones

```

function varargout = instructivo(varargin
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',   gui_Singleton, ...
                   'gui_OpeningFcn', @instructivo_OpeningFcn, ...
                   'gui_OutputFcn',  @instructivo_OutputFcn, ...
                   'gui_LayoutFcn', [], ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

handles.output = hObject;

```

A. CÓDIGO

```
axes(handles.axes2);
guidata(hObject, handles);

function varargout = instructivo_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function ver_Callback(hObject, eventdata, handles)
load('onoff.mat');
load('sv.mat');
load('bv.mat');
load('sc.mat');
load('bc.mat');
gesto=get(handles.menu,'Value');
switch gesto
    case 1
        set(handles.descripcion,'String','Aplauso.');
        for i=1:5
            imshow(imread(onoff(i).name))
            pause(0.05);
        end
    case 2
        set(handles.descripcion,'String','Subir mano derecha.');
        for i=1:5
            imshow(imrotate(imread(sv(i).name),90))
            pause(0.05);
        end
    case 3
        set(handles.descripcion,'String','Subir mano izquierda.');
        for i=1:5
            imshow(imrotate(imread(bv(i).name),-90))
            pause(0.05);
        end
    case 4
        set(handles.descripcion,'String','Desplazar mano derecha...
            horizontalmente de izquierda a derecha.');
        for i=1:5
            imshow(imread(sc(i).name))
            pause(0.05);
        end
    case 5
        set(handles.descripcion,'String','Desplazar mano izquierda...
            horizontalmente de derecha a izquierda.');
        for i=1:5
            imshow(imread(bc(i).name))
            pause(0.05);
```

```
    end
end

function menu_Callback(hObject, eventdata, handles)
    cla(handles.axes2);
    set(handles.descripcion,'String','Descripción del gesto');
```

A.3 Arduino

```
#include <IRremote.h>

IRsend irsend;

int led1 = 13;
int led2 = 12;
int led3 = 11;
int led4 = 10;
int led5 = 9;
int led6 = 8;
char mensaje;

void setup()
{
    Serial.begin(9600);
    pinMode(led1,OUTPUT);
    pinMode(led2,OUTPUT);
    pinMode(led3,OUTPUT);
    pinMode(led4,OUTPUT);
    pinMode(led5,OUTPUT);
    pinMode(led6,OUTPUT);
}

void loop()
{
    mensaje=Serial.read();
    switch(mensaje)
    {
        case '1': //Apagar leds
            irsend.sendNEC(0x20DF10EF,32);
            delay(100);
            digitalWrite(led1,LOW);
            digitalWrite(led2,LOW);
            digitalWrite(led3,LOW);
            digitalWrite(led4,LOW);
            digitalWrite(led5,LOW);
```

A. CÓDIGO

```
    digitalWrite(led6,LOW);
    break;
case '2': //Encender led2 - Subir volumen
    irsend.sendNEC(0x20DF40BF,32);
    delay(100);
    digitalWrite(led1,LOW);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
    digitalWrite(led5,LOW);
    digitalWrite(led6,LOW);
    break;
case '3': //Encender led3 - Bajar volumen
    irsend.sendNEC(0x20DFC03F,32);
    delay(100);
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,HIGH);
    digitalWrite(led4,LOW);
    digitalWrite(led5,LOW);
    digitalWrite(led6,LOW);
    break;
case '4': //Encender led4 - Subir canal
    irsend.sendNEC(0x20DF00FF,32);
    delay(100);
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
    digitalWrite(led4,HIGH);
    digitalWrite(led5,LOW);
    digitalWrite(led6,LOW);
    break;
case '6': //Encender led5 - Bajar canal
    irsend.sendNEC(0x20DF807F,32);
    delay(100);
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
    digitalWrite(led5,HIGH);
    digitalWrite(led6,LOW);
    break;
case 'e': //Encender led6 - Standby
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
```

A.3 Arduino

```
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
    digitalWrite(led5,LOW);
    digitalWrite(led6,HIGH);
    break;
}
}
```

Apéndice B

Entrenamiento y simulación de redes neuronales

B.1 Resultados de la fase de entrenamiento

A continuación se presentan los resultados de entrenamiento de varias redes neuronales artificiales mediante la herramienta *nntool*. El entrenamiento de estas redes presentó un error de aprendizaje no favorable para el sistema.

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

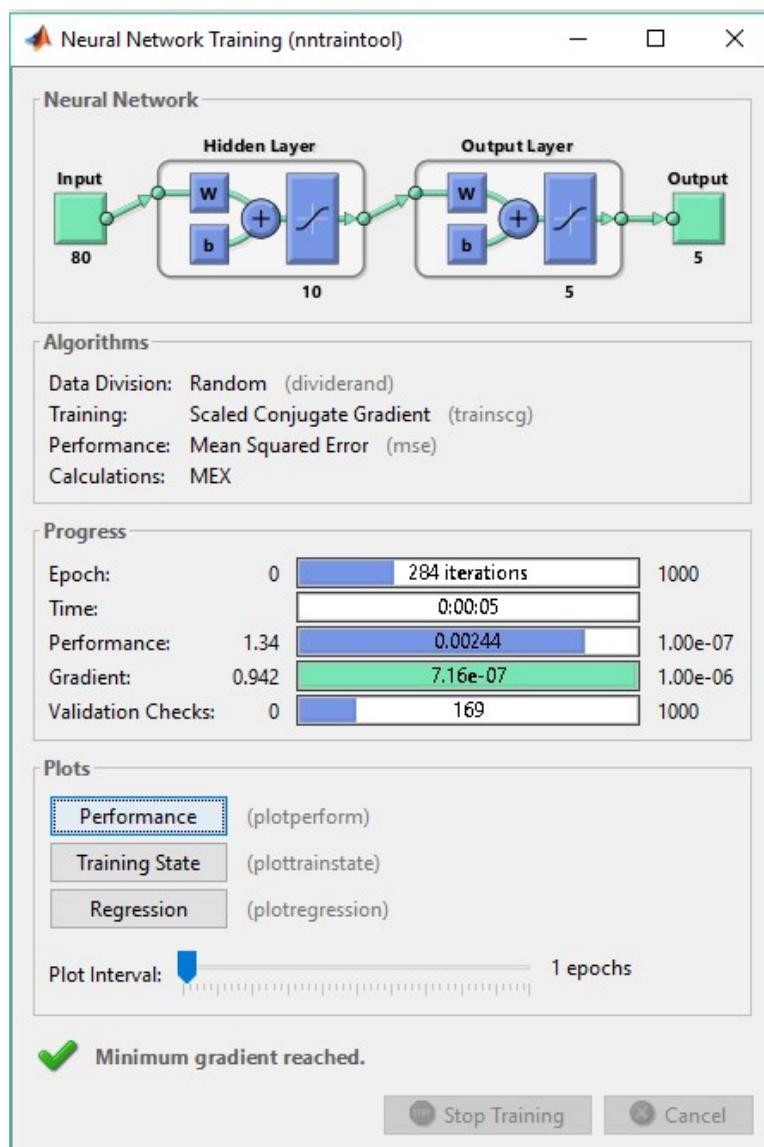


Figura B.1: Entrenamiento 1.

B.1 Resultados de la fase de entrenamiento

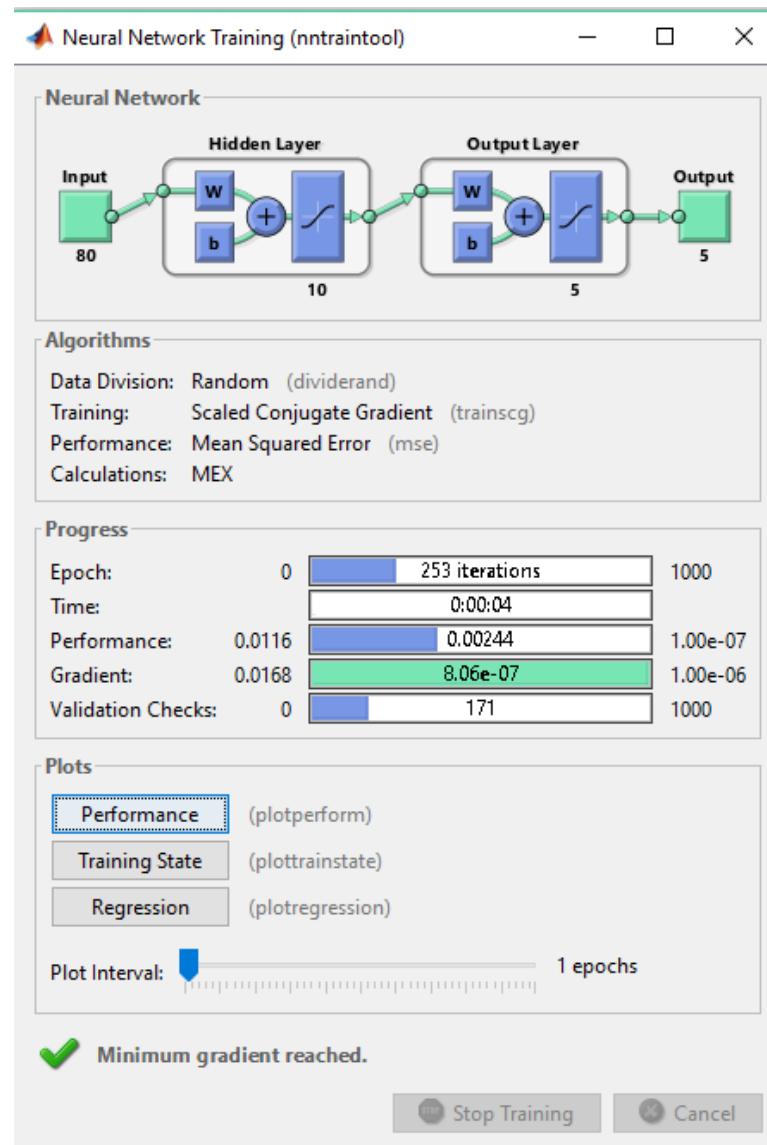


Figura B.2: Entrenamiento 2.

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

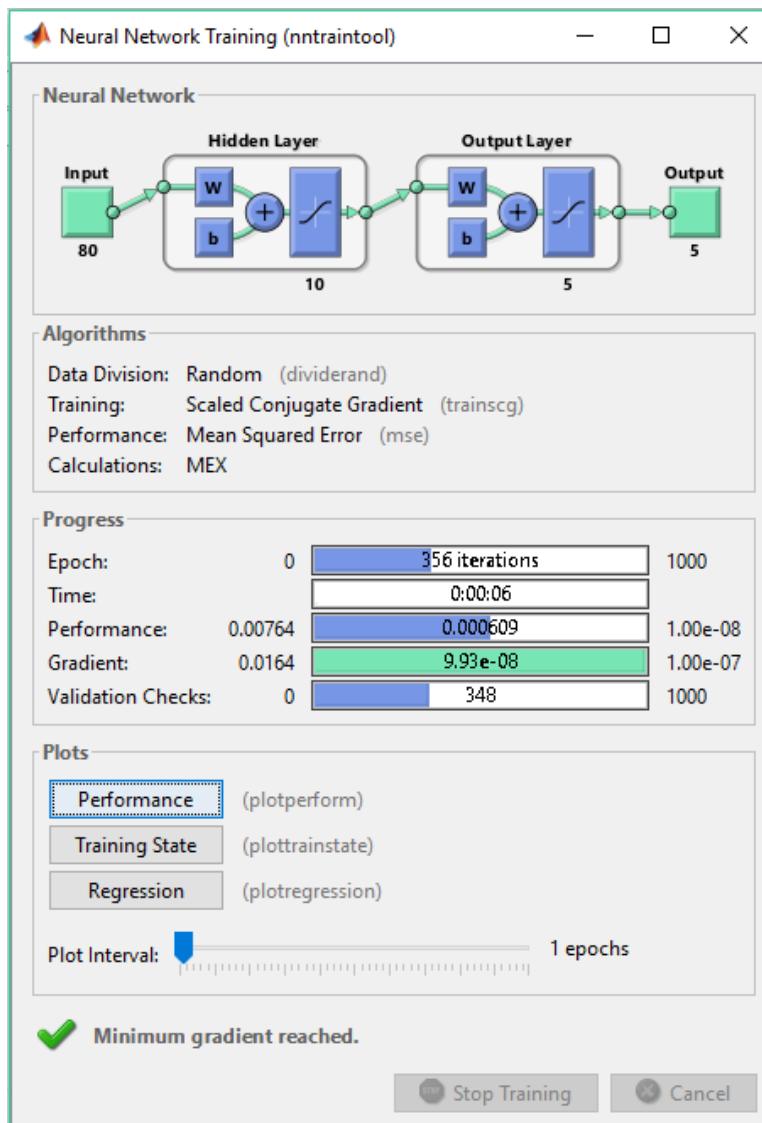


Figura B.3: Entrenamiento 3.

B.1 Resultados de la fase de entrenamiento

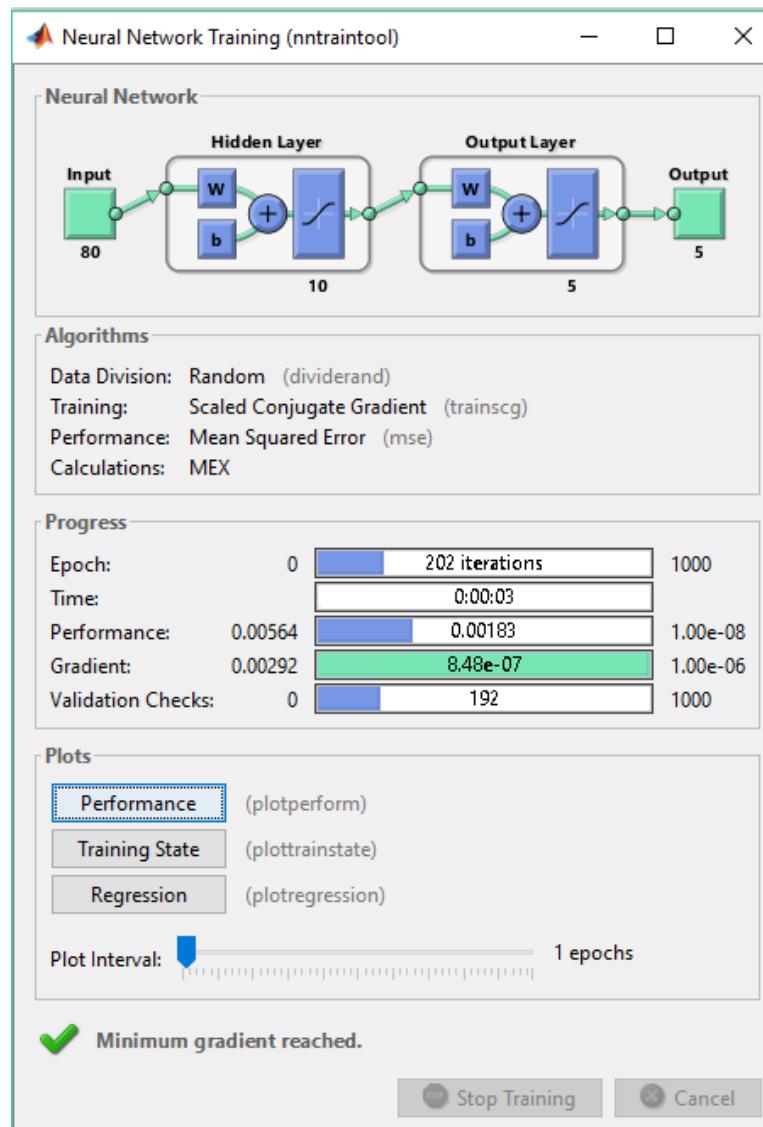


Figura B.4: Entrenamiento 4.

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

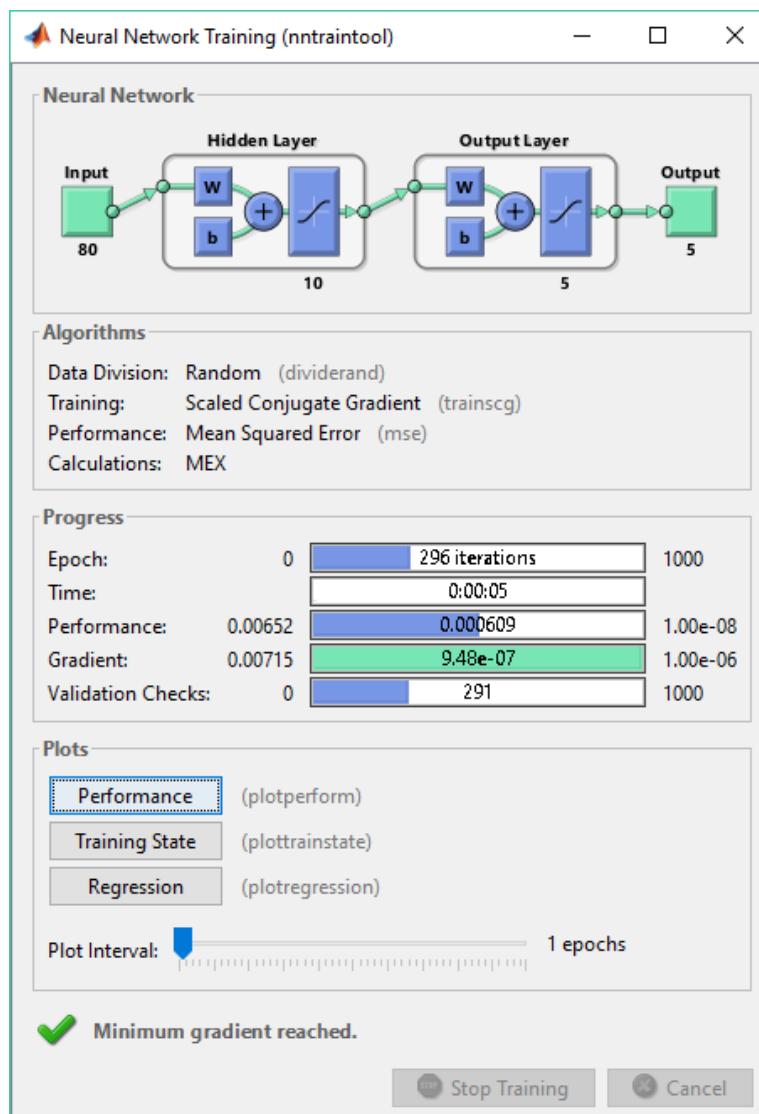


Figura B.5: Entrenamiento 5.

B.1 Resultados de la fase de entrenamiento

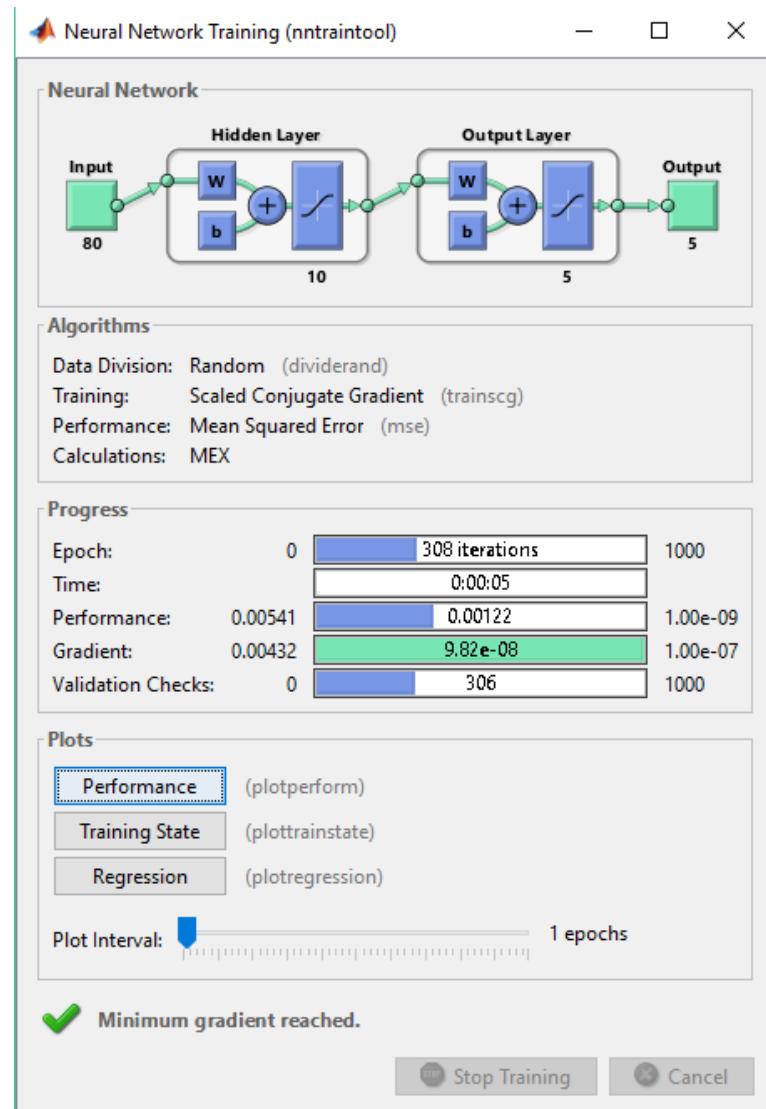


Figura B.6: Entrenamiento 6.

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

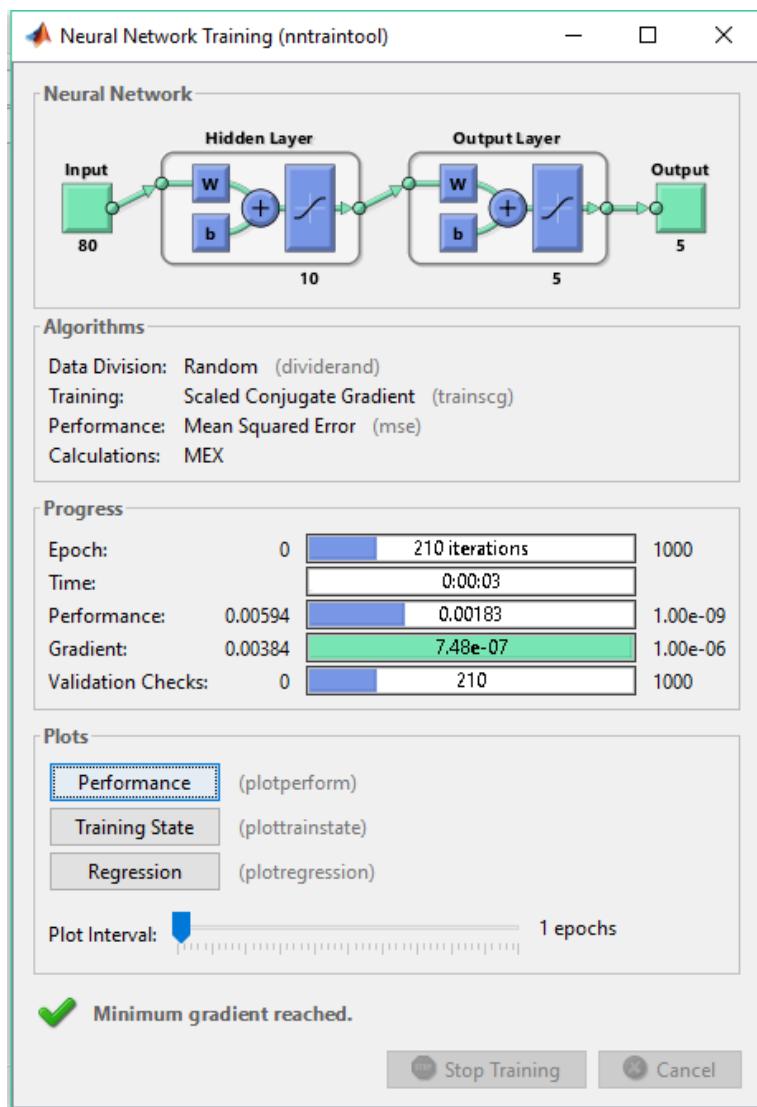


Figura B.7: Entrenamiento 7.

B.1 Resultados de la fase de entrenamiento

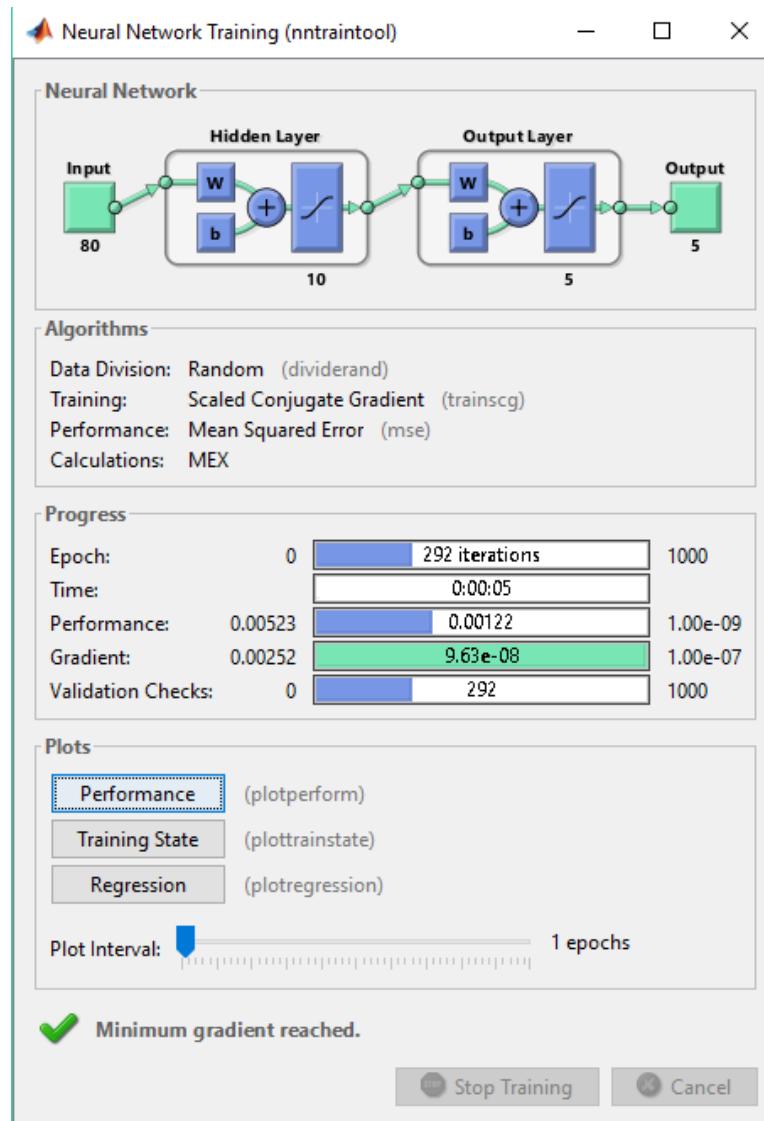


Figura B.8: Entrenamiento 8.

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

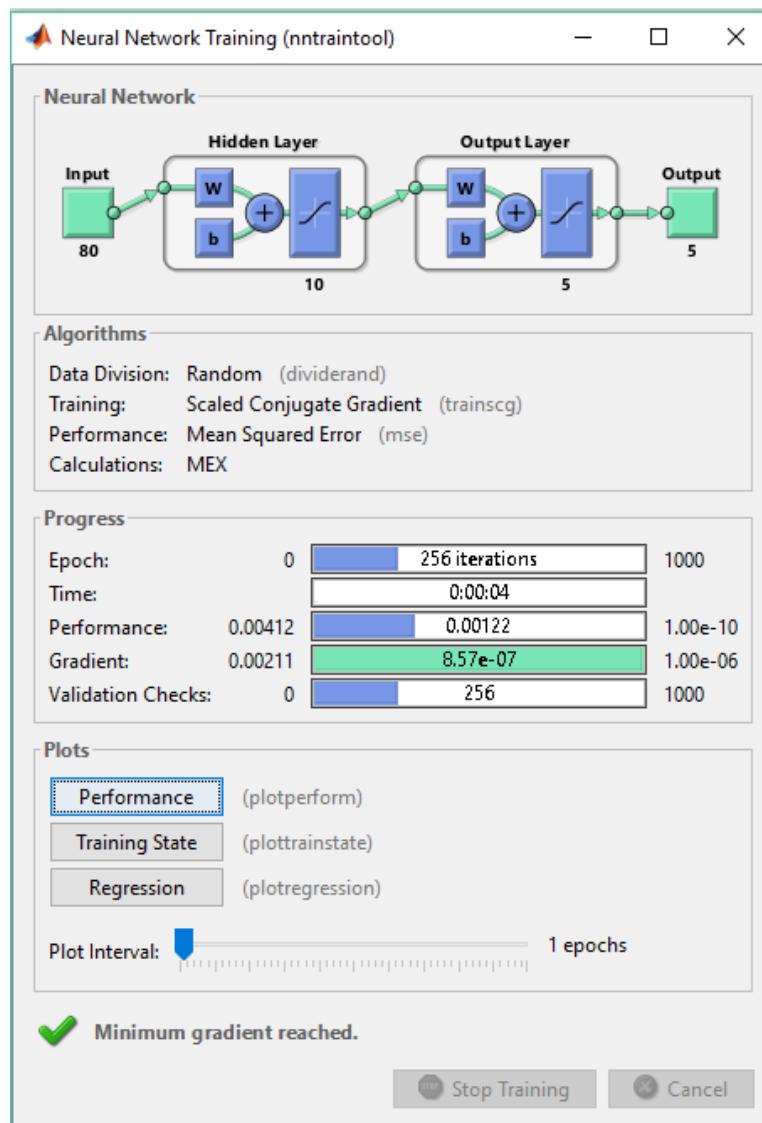


Figura B.9: Entrenamiento 9.

B.1 Resultados de la fase de entrenamiento

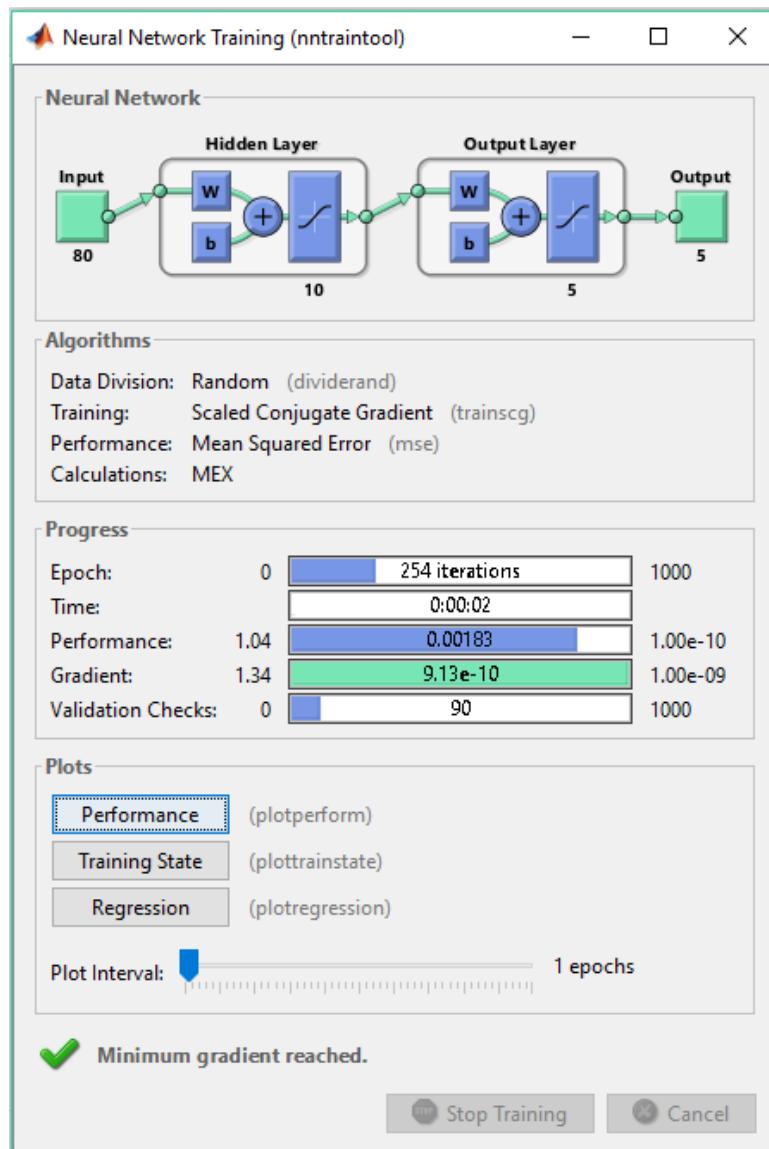


Figura B.10: Entrenamiento 10.

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

B.2 Resultados de la fase de simulación

A continuación se presentan los resultados de la simulación de la red neuronal que tuvo el resultado más favorable durante la fase de entrenamiento y que se emplea para el reconocimiento de gestos del sistema propuesto en este trabajo. La prueba se realizó con 625 patrones diferentes a los del entrenamiento.

Tabla B.1: Resultados de la simulación de la red neuronal.

| | Enc/Ap |
|------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| y(1) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9997 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -0.9974 | -0.9798 | -0.9974 | -0.9791 | -0.9942 | -0.9496 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | |
| | Enc/Ap |
| y(1) | -0.0686 | 1.0000 | 0.0419 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9999 | 1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -1.0000 | -0.9999 | -0.9994 | -0.9994 | -0.9612 | -0.9522 | -0.8893 | -0.9640 | -0.4177 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | |
| | Enc/Ap |
| y(1) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -1.0000 | -1.0000 | -0.9301 | -0.9881 | -0.9995 | -1.0000 | -0.9904 | -0.8287 | -0.7022 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | |
| | Enc/Ap |
| y(1) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -0.9982 | -0.9960 | -0.9372 | -0.9999 | -0.8620 | -0.9999 | -0.9769 | -0.9165 | -0.8308 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |

B.2 Resultados de la fase de simulación

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

B.2 Resultados de la fase de simulación

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

| | Baj.Vol. |
|------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| y(1) | -1.0000 | -1.0000 | -0.9999 | -1.0000 | -1.0000 | -0.9999 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -0.9998 | -0.9999 | -1.0000 | -0.8900 | -0.9603 | -0.9999 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9928 | -1.0000 | 0.9986 | 1.0000 | 1.0000 | 1.0000 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9947 | 1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -1.0000 | -1.0000 | -0.9999 | -1.0000 | -1.0000 | -0.9939 | -1.0000 | -1.0000 | -0.9943 | -1.0000 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9971 | -0.9997 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | 0.9996 | 1.0000 | 1.0000 | 1.0000 | 0.9999 | 0.9252 | 1.0000 | 1.0000 | -1.0000 | 1.0000 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | 0.9993 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | 0.7449 | -1.0000 | |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -0.9998 | -0.9999 | -0.7769 | -1.0000 | -1.0000 | -1.0000 | -0.9998 | -0.9979 | -0.9992 | -0.9999 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -0.4534 | -0.9946 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | 1.0000 | 1.0000 | 0.9961 | -1.0000 | -1.0000 | 0.9997 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -0.9823 | 1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9995 | -1.0000 | -1.0000 | -1.0000 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | -0.9998 | 0.4504 | 1.0000 | -0.2158 | 1.0000 | -0.9300 | 1.0000 | 1.0000 | 1.0000 | 0.9091 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | 0.6646 | 0.8477 | -1.0000 | -0.9978 | -1.0000 | 0.9162 | -1.0000 | -1.0000 | -1.0000 | -0.8390 | |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | -0.9978 | 0.9999 | -0.9965 | -0.8595 | -1.0000 | -0.7221 | 1.0000 | -0.5645 | 1.0000 | 0.9266 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | 0.9515 | -1.0000 | 0.9973 | -0.9779 | 0.9995 | -0.6152 | -1.0000 | 0.9626 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | -0.9036 | 0.9987 | 1.0000 | 1.0000 | 1.0000 | 0.4379 | 0.9999 | 1.0000 | -0.9948 | 0.9985 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | 0.3632 | -0.9999 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.8170 | -1.0000 | |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | 1.0000 | 0.9323 | 0.9558 | 1.0000 | 1.0000 | 0.9985 | 1.0000 | 0.7176 | 1.0000 | 1.0000 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9660 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Baj.Vol. |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(3) | 1.0000 | 1.0000 | 0.9994 | 1.0000 | 0.9999 | 0.9406 | 0.9973 | 1.0000 | 1.0000 | 1.0000 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9996 | -1.0000 | -1.0000 | |

B.2 Resultados de la fase de simulación

| | Baj.Vol. |
|------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | 1.0000 | 1.0000 | 1.0000 | 0.9987 | 0.8682 | 0.9998 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | -0.9996 | -1.0000 | -0.9988 | -0.8721 | -0.9624 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | | | |
| | Baj.Vol. | Baj.Vol. | Baj.Vol. | Baj.Vol. | Baj.Vol. | Sub.Canal | Sub.Canal | Sub.Canal | Sub.Canal | Sub.Canal | Sub.Canal |
| y(1) | -0.9999 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | -1.0000 | -1.0000 | -1.0000 | 0.9172 | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | | | |
| | Sub.Canal |
| y(1) | -1.0000 | -1.0000 | -0.9999 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9999 | |
| y(2) | -1.0000 | -1.0000 | 0.9248 | -1.0000 | -0.9998 | -0.9999 | -0.9990 | -0.9993 | -0.9999 | -1.0000 | |
| y(3) | -0.9862 | -0.9999 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(4) | 1.0000 | 1.0000 | -0.9997 | 1.0000 | 0.9999 | 0.9999 | 1.0000 | 0.9994 | 0.9999 | 1.0000 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Sub.Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -0.8345 | -0.9922 | -0.9998 | -0.9995 | -0.9999 | -0.9999 | -0.9976 | -0.9999 | 0.9816 | -0.9955 | |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(4) | 0.8980 | 0.9955 | 0.9999 | 0.9996 | 0.9999 | 0.9999 | 0.9997 | 1.0000 | -0.8961 | 0.9996 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Sub.Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | 1.0000 | -0.9988 | -0.9964 | -0.9999 | -0.9999 | -0.9998 | -0.9998 | -0.9878 | -0.9998 | -0.9999 | |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(4) | -1.0000 | 1.0000 | 0.9944 | 1.0000 | 0.9999 | 0.9997 | 0.9999 | 0.9907 | 1.0000 | 0.9999 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Sub.Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | 1.0000 | 0.7008 | -0.9915 | 1.0000 | 1.0000 | -0.9999 | -0.9999 | -0.9999 | -1.0000 | -0.9999 | -1.0000 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(4) | -1.0000 | -0.9722 | 0.9961 | -1.0000 | -1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Sub.Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | 1.0000 | -0.9999 | -1.0000 | -1.0000 | -0.9999 | -0.9999 | -0.9999 | -0.9999 | -1.0000 | -0.9891 | -0.9998 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(4) | -1.0000 | 0.9999 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9987 | 1.0000 |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| | | | | | | | | | | | |
| | Sub.Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9997 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -0.9990 | -0.9997 | 1.0000 | -0.9998 | -0.9998 | -1.0000 | -0.9163 | -0.9998 | -0.9997 | -1.0000 | |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |
| y(4) | 0.9999 | 0.9999 | -1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9844 | 1.0000 | 1.0000 | 1.0000 | |
| y(5) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | |

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

B.2 Resultados de la fase de simulación

| | Baj. Canal |
|------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.4890 | 0.9966 | -1.0000 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | -0.9979 | 1.0000 |
| | | | | | | | | | | |
| | Baj. Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.8998 | -1.0000 | -1.0000 | -1.0000 | -0.9998 | -1.0000 |
| y(2) | -0.9998 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | 1.0000 | 0.7076 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | 0.9994 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | -1.0000 | -1.0000 |
| | | | | | | | | | | |
| | Baj. Canal |
| y(1) | 0.8621 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | 0.9999 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -0.9992 | -0.9955 | -1.0000 | -1.0000 | -1.0000 | -0.9994 | -1.0000 |
| y(3) | -1.0000 | -0.9999 | -1.0000 | -0.9995 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | 1.0000 | 1.0000 | 1.0000 | 0.2509 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9521 | 1.0000 |
| | | | | | | | | | | |
| | Baj. Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -0.9999 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -1.0000 | 1.0000 | 1.0000 | 1.0000 | -1.0000 | -1.0000 | -0.2270 | -1.0000 | -1.0000 | -1.0000 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | 1.0000 | -1.0000 | -1.0000 | -1.0000 | 1.0000 | 1.0000 | 0.6019 | 1.0000 | 1.0000 | 1.0000 |
| | | | | | | | | | | |
| | Baj. Canal |
| y(1) | -1.0000 | -1.0000 | -0.9974 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | 1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | 1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -0.9996 | -0.9993 | -1.0000 | -0.9996 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | -1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9997 | 1.0000 |
| | | | | | | | | | | |
| | Baj. Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -0.9996 | -0.9995 | -0.9994 | -0.9997 | -0.9994 | -0.9997 | -0.9999 | -0.9993 | -0.9994 | -0.9997 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(4) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9997 | 1.0000 | 0.9989 | 0.9999 | 1.0000 |
| y(5) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| | | | | | | | | | | |
| | Baj. Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -0.9998 | -0.9997 | -0.9996 | -0.9995 | -0.9990 | -0.9991 | -0.9996 | -0.9992 | -0.9990 | -0.9993 |
| y(3) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(4) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9997 | 1.0000 | 0.9989 | 0.9999 | 1.0000 |
| y(5) | 1.0000 | -0.9988 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |

B. ENTRENAMIENTO Y SIMULACIÓN DE REDES NEURONALES

| | Baj. Canal |
|------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -0.9997 | -0.9996 | -0.9981 | -0.9998 | -0.9996 | -0.9997 | -0.9996 | -0.9998 | -0.9996 | -0.9997 | -0.9997 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| | | | | | | | | | | | |
| | Baj. Canal |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(3) | -1.0000 | -0.9999 | -0.9998 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | 1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 |
| y(5) | 1.0000 | 1.0000 | 0.9998 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | -1.0000 | 1.0000 | 1.0000 | 1.0000 |
| | | | | | | | | | | | |
| | Baj. Canal | | | | | | |
| y(1) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | | | | | | |
| y(2) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | | | | | | |
| y(3) | 0.4155 | -0.6703 | -1.0000 | -1.0000 | -1.0000 | | | | | | |
| y(4) | -1.0000 | -1.0000 | -1.0000 | -1.0000 | -1.0000 | | | | | | |
| y(5) | 0.0797 | 0.9562 | 1.0000 | 1.0000 | 1.0000 | | | | | | |

Apéndice C

Análisis de costos

C.1 Material y Herramientas

- Computadora Acer modelo Aspire E15 E5-522-86AU: \$8,000
- Sensor Kinect: \$1,800
- Licencia de MatLab (versión estándar): \$39,633
- Tarjeta Arduino Uno: \$250
- 6 Resistencia de 220 ohms: \$6
- 1 Resistencia de 470 ohms: \$1
- Transistor 2N2222: \$6
- Led Infrarrojo: \$7
- Receptor de Infrarrojo VS1838B: \$10
- 8 Cables Dupont Macho-Hembra: \$12
- Placa perforada: \$10
- Soldadura: \$5
- Cautín tipo lápiz: \$100

Se puede prescindir de la licencia de MatLab si la RNA se programa en un lenguaje abierto.

Costo parcial sin Licencia de MatLab: \$10,207

Costo parcial con Licencia de MatLab: \$49,840

C. ANÁLISIS DE COSTOS

C.2 Desarrollo

El proyecto se realizó en un periodo de seis meses, en el cual intervinieron dos programadores junior. El salario de un programador junior es de \$8,000 mensuales (10).

Salario total: \$96,000.

Costo total del proyecto sin licencia de MatLab: \$106,207

Costo total del proyecto con licencia de MatLab: \$145,840

Referencias

- [1] JAMES BENNETT AND NIKI STRANGE. *Television as digital media*. 2011. 5
- [2] BEST BUY. Precio de televisión samsung un55js9000fxzx. <http://www.bestbuy.com.mx/productos/experiencias/samsung/productos-samsung/samsung-pantalla-de-55-led-3840p-smart-3d-curva-tv-ultra-hd-negro.html>, 2015. 2, 15
- [3] PO-HUNG CHEN AND HUNG-CHENG CHEN. Application of back-propagation neural network to power transformer insulation diagnosis. In DERONG LIU, SHUMIN FEI, ZENGGUANG HOU, HUAGUANG ZHANG, AND CHANGYIN SUN, editors, *Advances in Neural Networks ISNN 2007*, **4493** of *Lecture Notes in Computer Science*, pages 26–34. Springer Berlin Heidelberg, 2007. 18
- [4] RODOLFO CRUZ, HÉCTOR FERNÁNDEZ, AND ADRIÁN OLVERA. *Plataforma interactiva de Kinect aplicada al tratamiento de niños autistas*. Master’s thesis, Instituto Politécnico Nacional, 2013. 2
- [5] HENG DAI AND COLIN MACBETH. Application of back-propagation neural networks to identification of seismic arrival types. *Physics of the Earth and Planetary Interiors*, **101**:177–188, 1997. 18
- [6] J. DUCLOUX, P. PETRASHIN, W. LANCIONI, AND L. TOLEDO. Remote control with accelerometer-based hand gesture recognition for interaction in digital tv. In *Micro-Nanoelectronics, Technology and Applications (EAMTA), 2014 Argentine Conference on*, pages 29–34, July 2014. 17
- [7] QI FENG, CHENG YANG, XIAOYU WU, AND ZHUOJIA LI. A smart tv interaction system based on hand gesture recognition by using rgb-d sensor. 2013. 10, 11, 12
- [8] WILLIAM T. FREEMAN AND CRAIG D. WEISSMAN. Television control by hand gestures. *Proc. of Int. Workshop on Automatic Face and Gesture Recognition.*, pages 179–183, 1995. 8, 9
- [9] GOOGLE. Imágenes tomadas de google imágenes. <https://images.google.com/>. 6, 7, 14, 20, 21, 22, 23, 25, 26, 27, 29, 36, 37
- [10] INDEED. <http://www.indeed.com.mx/?r=us>, Salarios 2016. 112

REFERENCIAS

- [11] INEGI. Módulo sobre disponibilidad y uso de las tecnologías de la información en hogares. http://www.sct.gob.mx/uploads/media/NOTA_T%C3%89CNICA_MODU-TIH_2014_DEL_COMUNICADO_379.pdf, 2014. 1
- [12] PEDRO ISASI AND INÉS GALVÁN. *Redes Neuronales Artificiales. Un enfoque práctico.* 2004. 28
- [13] JIEFENG JIANG, JING ZHANG, GEGE YANG, DAPENG ZHANG, AND LIAN-JUN ZHANG. Application of back propagation neural network in the classification of high resolution remote sensing image: Take remote sensing image of beijing for instance. *Geoinformatics*, 2010. 18
- [14] DAVID KATZMAIER. Samsung smart interaction: Hands-on with voice and gesture control. <http://www.cnet.com/news/samsung-smart-interaction-hands-on-with-voice-and-gesture-control/>, apr 2012. 13
- [15] SHIGUO LIAN, WEI HU, AND KAI WANG. Automatic user state recognition for hand gesture based low-cost television control system. *IEEE Consumer Electronics*, **60**:107–115, 2014. 12, 13
- [16] CARLOS MARTÍNEZ. Samsung lanza oficialmente el evolution kit por 299 euros. <http://es.engadget.com/2013/06/11/samsung-evolution-kit-precio-instalacion/>, 2013. 15
- [17] MICROSOFT. Kinect for windows sensor components and specifications. <https://msdn.microsoft.com/en-us/library/jj131033.aspx>. 21
- [18] MICROSOFT. Descarga el sdk v1.8 de kinect para windows. <http://www.microsoft.com/en-us/download/details.aspx?id=40278>, 2015. 22
- [19] BRAD MYERS. A brief history of human computer interaction technology. *ACM Interactions*, **5**(2):44–54, 1998. 8
- [20] DIAN PRATIWI, SANTIKA DIAZ, AND BENS PARDAMEAN. An application of backpropagation artificial neural network method for measuring the severity of osteoarthritis. *International Journal of Engineering & Technology*, **11**(3), jun 2011. 18
- [21] SAMSUNG. Smart interaction: el mando a distancia eres tú. <http://www.samsung.com/es/article/smart-interaction-you-are-the-remote-control>, 2012. 14
- [22] SAMSUNG. All of the smart tv gestures. http://www.samsung.com/ph/smarttv/common/guide_book_3p_si/waving.html, 2013. 16

REFERENCIAS

- [23] JIA-SHING SHEU, GUO-SHING HUANG, AND YA-LING HUANG. Design an interactive user interface with integration of dynamic gesture and handwritten numeral recognition. *Computer, Consumer and Control*, pages 1295–1298, 2014. 15, 17
- [24] ZENITH. Five decades of channel surfing: History of the tv remote control. https://web.archive.org/web/20080116212531/http://www.zenith.com/sub_about/about_remote.html, 2003. 5