ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

TEMA:

"RECONOCIMIENTO Y CLASIFICACIÓN DE LENGUAJE DE SEÑAS USANDO KINECT E INTELIGENCIA ARTIFICIAL"

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN MECATRÓNICA Y ROBÓTICA

AUTOR:

Ing. Tomás Guamán

ing.guaman_gamboa@hotmail.es

DIRECTOR:

Ing. Patricio Cruz, PhD

patricio.cruz@epn.edu.ec

Quito, Junio de 2019

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Ing. Tomás Santiago Guamán Gamboa CI: 1802792646, bajo mi supervisión.

_____

Ing. Patricio J. Cruz, PhD.
Director

DECLARACIÓN

Yo, Tomás Santiago Guamán Gamboa CI: 1802792646, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

Ing. Tomás Santiago Guamán Gamboa

CI 1802792646

DEDICATORIA

A Dios.

Por haberme permitido culminar con éxito mi formación académica y siempre llenarme de bendiciones en lo que sea que me proponga.

A mi padre Alonso.

Por ser mi modelo a seguir, una persona que en verdad admiro mucho, por sus concejos y sabiduría que día a día me han hecho mejor. Para mi es un desafío ser como usted, por su crecimiento espiritual, moral y sentimental, pero algún día espero crecer tanto como usted.

A mi madre Nubia.

Por siempre llenarme de amor y comprensión, por tener un hogar con mucha armonía, calidez, paz y siempre en orden. Esto me ha permitido dedicar mis esfuerzos en alcanzar mis objetivos.

A mi hermana Evelyn.

Por ser una persona dedicada, que disfruta en gran manera lo que hace, incansable y una gran amiga.

A mi hija Doménica.

Por ser luz en vida y alegrarme los días siempre con sus ocurrencias. Me demuestras con tu forma de ser lo mucho que te pareces a mí y me llena mucho verme reflejado en ti.

AGRADECIMIENTO

Primero a Dios que es que todo lo hace posible, después a mis familiares y amigos que siempre me han acompañado durante mi vida personal, profesional y académica. A mis docentes que con sus conocimiento y enseñanzas me han permitido nutrirme de sabiduría, principalmente a mi director de Tesis por apoyarme en este proyecto desde que era solo una idea hasta ahora que gracias a su ayuda la hemos podido materializar. A mis compañeros que con su ayuda y camaradería hemos podido cursar y culminar esta maestría en un buen ambiente con una competencia muy ardua pero la vez muy sana. A la Mg. María Alejandra que con sus conocimientos acerca de la lengua de señas contribuyó a este trabajo. Finalmente un abrazo a todas las personas que conforman la EPN, directores de carrera, departamentos administrativos, biblioteca, laboratoristas, etc. Ellos también han puesto su granito de arena para que esto sea posible.

ÍNDICE

Índice de Figuras

Índice de Tablas

RESUMEN

En el mundo existen alrededor de 466 millones de personas con discapacidad auditiva lo cual representa más del 5% de la población mundial [1]. Estas personas presentan serios inconvenientes para poder comunicarse con otras que no hablan la lengua de señas, lo que repercute en un aislamiento social e impide un desarrollo igualitario. Este proyecto se basa en la necesidad de que las personas, quienes no hablan este tipo de lenguaje, puedan entender las señas realizadas por alguien con discapacidad auditiva mediante un programa de ordenador que utiliza inteligencia artificial y un sensor 3D.

En esta investigación se utilizan las trayectorias de las manos obtenidas con la ayuda del sensor 3D, para entrenamiento, prueba y validación de un algoritmo de inteligencia artificial basado en redes neuronales responsable del reconocimiento y clasificación de 5 señas básicas determinadas por un experto en este campo. La red neuronal utilizada es un perceptrón multicapa entrenado mediante retro propagación.

El entrenamiento de la red neuronal se lo realizó con 6 personas y las pruebas con 9 personas adicionales. En la fase de simulación, el sistema tiene una efectividad en reconocimiento y clasificación del 99.6%; mientras que, en las pruebas de campo es del 98.7%. Además se desarrolló un HMI para la toma de datos y la presentación de resultados.

Palabras Clave: Lengua de señas, Redes Neuronales, Kinect, Reconocimiento, Clasificación.

ABSTRACT

In the world there is about 466 millions of people with deafness, it represents more than 5% of global population [1]. These people have several inconvenient to communicate with others who do not speak the sign language. This issue as social isolation and prevents equal growth. This project is based on the needs for people who do not speak this type of language, so they can understand the signs made by someone deaf through a software which uses artificial intelligence and a 3D sensor.

In this research, the hand trajectories obtained with the help of the 3D sensor are used for training, validation and test of an artificial intelligence algorithm based on neural network. The network is responsible for recognition and classification of 5 basic signs determined with the help of an expert in this field. The neural network is a multilayer perceptron trained by back propagation.

The training is done with 6 people and the test with 9 additional. In the simulation phase, the system has an effectiveness in recognition and classification of 99.6%; while, in the field test of 98.7%. In addition, an HMI was developed for data acquisition and the presentation of results.

Keywords: Sign Language, Neural Networks, Kinect, Recognition, Classification.

INTRODUCCIÓN

En el mundo existen alrededor de 466 millones de personas sordas o hipo acústicas, 18 millones de ellas se encuentran en Sudamérica. En el Ecuador, según el Consejo Nacional para la igualdad de Discapacidades (CONADIS) [2], hay 62177 personas con discapacidad auditiva.

Una persona con discapacidad auditiva tiene complicaciones en el desarrollo del lenguaje, desarrollo cognitivo, desarrollo personal y social. En el Ecuador, muy pocas personas hablan el lenguaje de señas y para una persona con discapacidad auditiva el no poder comunicarse dificulta sus actividades cotidianas como ir de compras, visitar al médico, realizar trámites en instituciones públicas o privadas, etc. Esto conlleva aislamiento de la sociedad a la persona.

Además, la constitución del Ecuador en el artículo 48 estipula que el Estado adoptará a favor de las personas con discapacidad: "La inclusión social, mediante planes y programas estatales y privados coordinados, que fomenten su participación política, social, cultural, educativa y económica". Sin embargo, en la actualidad no existe algún sistema que permita a las personas con discapacidad auditiva comunicarse con alguien que no hable el lenguaje de señas.

Una de las razones de la dificultad para implementar este sistema es que el lenguaje de señas no es universal. El lenguaje de señas ecuatoriano es el LSEC y esta normado bajo la Federación Nacional de Personas Sordas del Ecuador (FENASEC) [3]. Este modelo está basado en el lenguaje LSE de la real academia de la lengua española y aunque varios países Latinoamericanos han tomado como referencia el lenguaje LSE hay diferencias notables de país a país. Otra de las dificultades es que si bien es cierto existe un patrón para cada seña, es imposible que una persona siga el patrón de manera precisa. Dicho de otra manera, para una misma seña puede haber varias trayectorias de la mano y dependerá de factores como la fisionomía de la persona, el estado de ánimo, etc., lo que complica su reconocimiento y clasificación. Adicionalmente, una persona con discapacidad auditiva utiliza varias señas para formar una frase, estas señas son expresadas una a continuación de otra sin una pausa que facilite a un sistema reconocer en donde termina una seña y empieza otra.

Por estas razones, esta investigación busca ser un punto de partida para la implementación de un sistema capaz de reconocer y clasificar lenguaje de señas capturando los datos del movimiento de una persona a través de un Kinect. Estos datos serán analizados mediante un algoritmo de inteligencia artificial el cual es el responsable del reconocimiento y clasificación. Finalmente los resultados se mostraran a través de un HMI con el objetivo de sentar una base para que posteriores investigaciones permitan a las personas con discapacidad auditiva comunicarse con otras que no hablen el lenguaje de señas.

En el Capítulo 1 se determina la naturaleza, tipos y características de la lengua de señas a fin de encontrar las diferencias más relevantes que permitan su clasificación; además, debido a la gran cantidad de señas se delimita el número de señas a 5 basado en el criterio de un experto. Posteriormente se analizan los diferentes sensores 3D disponibles en el mercado y se escoge uno acorde a la aplicación. Finalmente se analizan los diferentes algoritmos de inteligencia artificial y se selecciona uno compatible con el sistema. En el Capítulo 2 se detallan los componentes del sistema tanto en software como en hardware además de cómo se realizó la toma de datos, el diseño del algoritmo de inteligencia artificial, el entrenamiento del algoritmo, el diseño del HMI y el funcionamiento del sistema. En el Capítulo 3 se presenta el funcionamiento del sistema, así como también los resultados obtenidos; además, se detalla cómo se realizó el Test. En el Capítulo 4 se muestran las conclusiones y recomendaciones.

Objetivo General

Reconocer y Clasificar Lenguaje de señas usando Kinect e Inteligencia Artificial.

Objetivos Específicos

1. Recopilar Información sobre el Lenguaje de Señas, Kinect e Inteligencia Artificial.
2. Obtener las coordenadas del exoesqueleto de la persona del Kinect.
3. Diseñar el algoritmo de inteligencia artificial.
4. Realizar una interfaz HMI.
5. Realizar Pruebas del Sistema.

Alcance

El sistema será capaz de reconocer y clasificar un pequeño grupo de señas representativas. El movimiento de la persona se obtendrá a través del Kinect y esta información será utilizada por una algoritmo de inteligencia artificial para su posterior reconocimiento y clasificación. El entrenamiento del algoritmo de inteligencia artificial se lo realizará con un pequeño grupo de personas y su funcionamiento será probado con al menos una persona. Los signos serán reconocidos y clasificados uno a uno siendo el HMI responsable de indicar cuando la persona puede expresar una seña y reconocer cuando terminó. Finalmente los resultados del reconocimiento y clasificación se podrá visualizar a través de un HMI.

1. MARCO TEÓRICO

1.1. Lenguaje de señas

El ser humano por su naturaleza social necesita comunicarse, expresar ideas y establecer relaciones con otras personas. A lo largo del tiempo la sociedad se ha ingeniado formas y medios para poder comunicarse, por ejemplo, comunidades primitivas utilizaban señales de humo, sonido de tambores, palomas mensajeras e incluso emulaban el sonido de los animales para satisfacer dicha necesidad. A medida que las organizaciones sociales se hicieron más complejas, las personas utilizaban señas, gestos, sonidos y pictogramas para poder expresarse. Esto ha venido evolucionando hasta convertirse en lo que hoy conocemos como Lenguaje.

Según La Real Academia de la Lengua Española el Lenguaje se define como "la facultad del ser humano de expresarse y comunicarse con los demás a través del sonido articulado o de otros sistemas de signos" [4]. El lenguaje se clasifica en: lenguaje escrito, lenguaje oral, lenguaje pictórico y lenguaje mímico o lenguaje de señas. [4].

Existen muchas denominaciones para referirse al lenguaje de señas, en algunos textos se puede encontrar expresiones como: lenguaje mímico, lenguaje gestual, lenguaje manual, lenguaje de signos manuales, etc. El lenguaje de señas se puede definir como un lenguaje no verbal en que se utilizan señas o gestos para enviar un mensaje.

1.1.1. Discapacidad Auditiva

Una persona sorda o hipoacúsica posee una pérdida de una función fisiológica del sistema auditivo que deriva en una discapacidad para oír. Este déficit puede ser leve, medio, severo o profundo dependiendo el lugar en donde se encuentre la lesión [5]. La Figura 1.1. muestra el grado de decibeles que una persona hipoacúsica puede oír.

Considerando el momento en el que aparece la lesión se puede clasificar la hipoacusia en cuatro: Si la lesión aparece en los tres primeros meses de vida se denomina hipoacusia congénita, cuando aparece antes del desarrollo de la lengua oral se denomina hipoacusia pre locutiva, si aparece durante el desarrollo de la lengua oral se conoce como hipoacusia peri locutiva y si aparece después del desarrollo de la lengua oral se denomina hipoacusia post locutiva [6]. Es importante citar este aspecto puesto que la quinésica oral[1] puede diferir en una persona con una hipoacusia prelocutiva que en otra con hipoacusia post locutiva puesto que la primera sufrió la lesión antes de aprender el lenguaje oral por ende no puede relacionar el lenguaje de señas con el lenguaje oral.

[1] Movimiento de los labios que simula la pronunciación de una palabra.

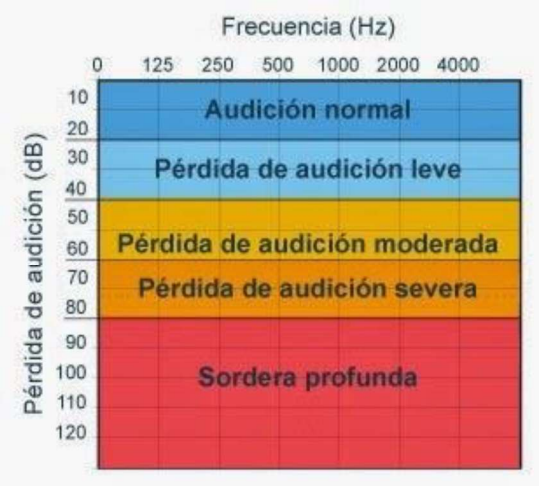


Figura 1.1. Audiograma con Niveles de Pérdida Auditiva.
(Fuente: [6])

1.1.2. Tipos de Lenguaje de Señas

Alrededor del mundo existen varios tipos de Lenguaje de Señas. Estos están sujetos al proceso natural del cambio lingüístico por lo que hay diferencias notables de país a país. Las principales familias de lengua de señas en el mundo están representadas en la Figura 1.2. y son: familia británica (rojo), familia hispano-francesa (verde), familia alemana (morado), familia sueco-finesa (azul), familia indo pakistaní (celeste), familia japonesa (ámbar), familia keniana (marrón) y familia árabe (caqui), las que se encuentran en color gris tienen su propia lengua de señas.

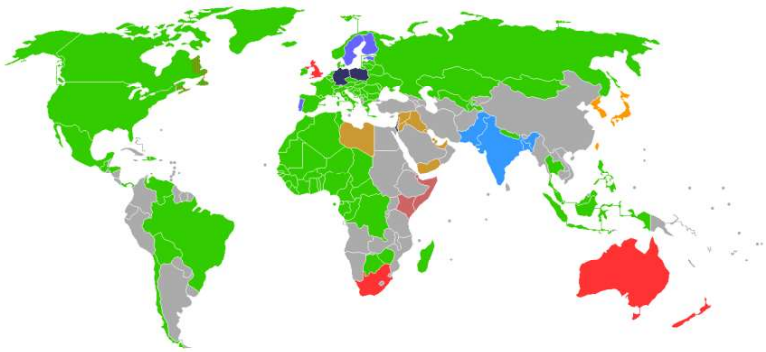


Figura 1.2. Tipos de Lengua de Señas en el Mundo.
(Fuente: [7])

Estas lenguas han ido evolucionando desde el siglo XVII y en la actualidad las diferencias son notables por lo que cada país a conformado su propio lenguaje de señas. Por ejemplo EEUU utiliza el lenguaje ASL(American Sing Language), México el LSM(Lengua de signos Mexicana), España el LSE (Lengua de señas Española), Brasil el LIBRAS(Língua Brasileira

dos Sinais), etc. En el caso de Ecuador la lengua de señas es la LSEC(Lengua de señas Ecuatoriana) [3].

La Figura 1.2. divide los tipos principales de lengua de señas en el mundo, las regiones representadas con un mismo color presentan similitudes; sin embargo, las regiones pintadas de color distinto manejan su propio lenguaje de señas.

La lengua de señas Ecuatoriana esta normada por la FENASEC (Federación Nacional de Personas Sordas del Ecuador), se compone de cinco mil palabras. Esta se puede descargar de forma gratuita en la página del CONADIS [8]. Para mejor compresión, las palabras están divididas en 11 categorías y cada una de ellas cuenta con una representación oral y en lengua de señas.

1.1.3. Elementos del Lenguaje de Señas

El lenguaje de señas está integrado por varios componentes. De acuerdo a [9], los componentes que integran el lenguaje de señas son: signos manuales, quinésica facial, quinésica oral, dactilología y quinésica somática. Todos estos elementos intervienen en la transmisión de un mensaje entre personas con discapacidad auditiva siendo el más importante los signos manuales.

Utilizando la quinésica facial se puede transmitir estados de ánimo lo cual es un complemento al mensaje que se desea transmitir, La Figura 1.3. muestra algunas representaciones quinésicas comunes.

Figura 1.3. Quinésica Facial.
(Fuente: [10])

Por otra parte la quinésica oral es el movimiento auxiliar de los labios de la persona con deficiencia auditiva cuando realiza signos manuales. Este movimiento no corresponde a la palabra en leguaje oral, a su vez la persona utiliza ciertos rasgos como: abertura de la mandíbula, configuración de las mejillas, posiciones de la lengua para enfatizar el mensaje similar a lo que ocurre con la quinésica facial.

La dactilología corresponde al alfabeto de lenguaje de señas, cada letra del alfabeto le corresponde un signo como se puede apreciar en la Figura 1.4. Utilizar la dactilología para comunicarse implica expresar las palabras letra por letra lo cual resulta poco funcional; sin embargo, es importante para denotar sustantivos, nombres propios, direcciones y expresar alguna palabra que no tenga un signo correspondiente. Además la dactilología es universal.

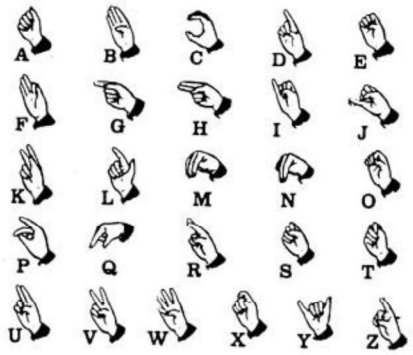


Figura 1.4. Alfabeto Dactilológico.
(Fuente: [11])

La quinésica somática hace referencia a gestos que las personas realizan de manera no premeditada, por ejemplo si una persona sufre una quemadura en el dedo hace un gesto el cual es una reacción instintiva; sin embargo, con este gesto la persona puede expresar dolor.

Signos Manuales

Cada palabra tiene una representación en lengua de señas, normalmente los textos enseñan los signos de cada palabra a través de ilustraciones las cuales intentan expresar como es el movimiento de las manos. De acuerdo a [12], estos movimientos y trayectorias se representan en las Figuras 1.6. y 1.7. Por otro lado los signos, tomando en cuenta sus relaciones semánticas, se pueden clasificar en los grupos de la Figura 1.5. [9].

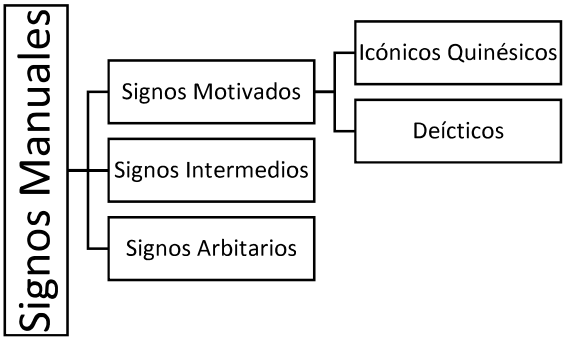


Figura 1.5. Clasificación de Signos Gestuales.
(Fuente: Propia)

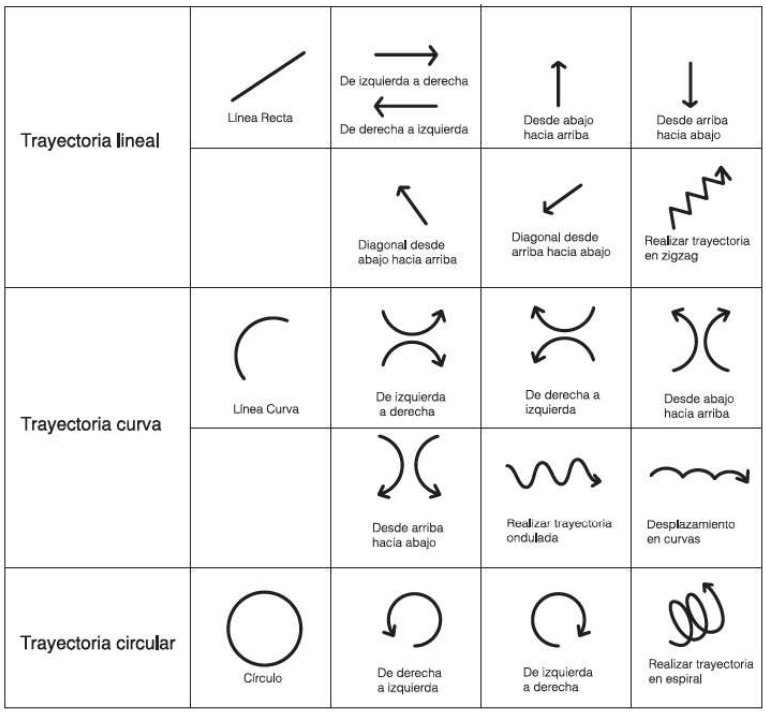


Figura 1.6. Representación de las Trayectorias y Dirección de Movimientos.
(Fuente: [12])

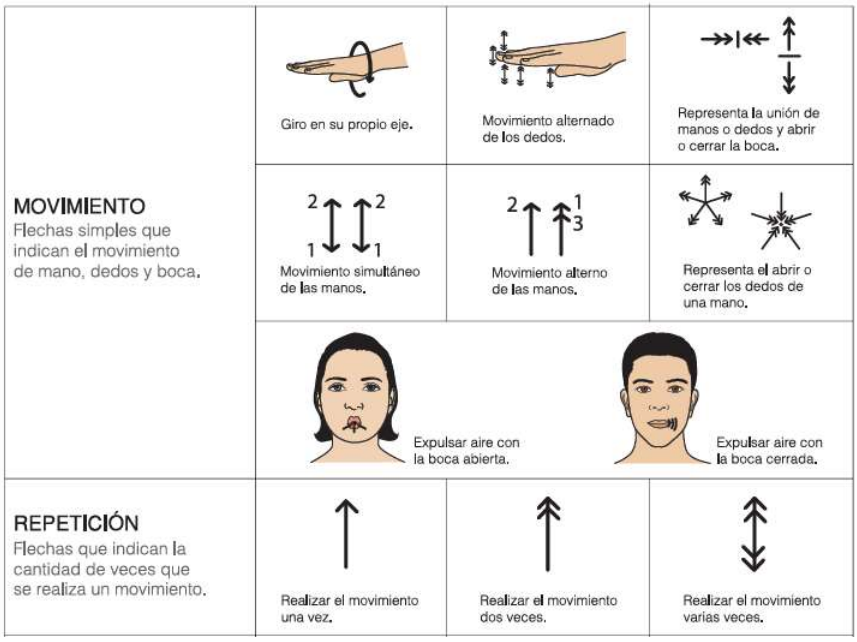


Figura 1.7. Movimiento de la mano dedos y boca.
(Fuente: [12])

Basado en la clasificación dada en la Figura 1.5., los signos motivados icónicos quinésicos en algún sentido se parecen a lo que representan. Por ejemplo para expresar médico, se colocan los dedos índice y pulgar de la mano derecha toman la muñeca izquierda simulando localizar el pulso como se muestra en la Figura 1.8(a)., por otro lado, la palabra verano se expresa con la mano derecha simulando quitarse el sudor cuando hace mucho calor, ver Figura 1.8(b).

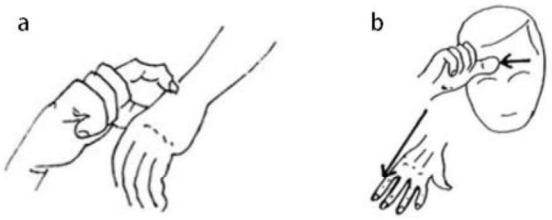


Figura 1.8. (a) Signo Médico,(b) Signo Verano.
(Fuente: [9])

Los signos motivados deícticos corresponden a señalar lo que se refiere y son utilizados para expresar pronombres personales, tiempo y espacio. Por ejemplo, la Figura 1.9. muestra como una persona con discapacidad auditiva señala a otras personas para establecer de quien se está hablando en la conversación.

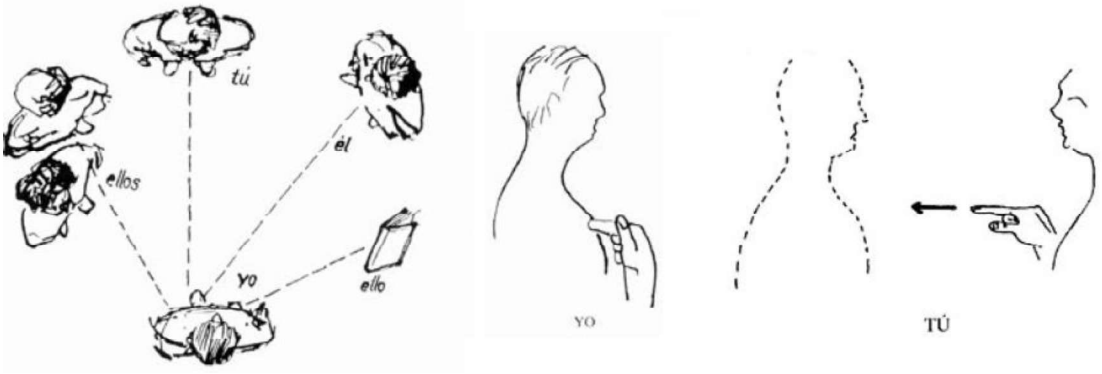


Figura 1.9. Pronombres Personales.
(Fuente: [9])

De la misma manera para referirse al tiempo se utilizan los movimientos de la Figura 1.10.

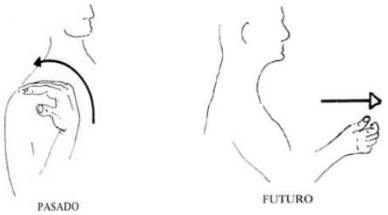


Figura 1.10. Signos de Pasado y Futuro.
(Fuente: [9])

Por otra parte se conoce como signos intermedios a aquellos que tienen origen dactilológico además de algunas expresiones muy breves. La Figura 1.11(a). muestra la palabra “Té”, mientras que la Figura 1.11(b). muestra la palabra “Sí”.

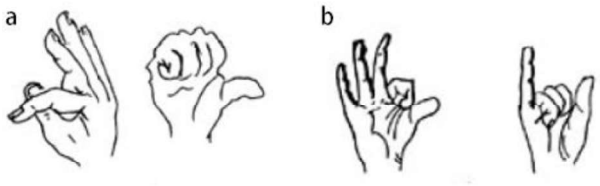


Figura 1.11. Signos Intermedios.
(Fuente: [9])

Finalmente los signos arbitrarios corresponden a representaciones con un componente no figurativo. El hecho más representativo de este tipo de signos son los nombres propios, principalmente los de personas ya que en la comunidad de personas con discapacidad auditiva cada persona tiene un gesto representativo el cual ha sido asignado de forma aleatoria tratando de resaltar alguna característica física de la persona.

1.1.4. Lengua de Señas Ecuatoriano LSEC

La logopedia es la ciencia que estudia el lenguaje de señas. Según la entrevista realizada a la experta, el aprendizaje del lenguaje de señas es un proceso por el cual todas las personas pasamos en los primeros años de vida [13]. Las personas utilizan señas y gestos para obtener lo que necesitan como alimento, atención, juguetes, etc. A medida que las personas crecen, alrededor de los dos años de edad, se puede hacer una diferencia entre personas con discapacidad auditiva las cuales necesitan una instrucción formal acerca del lenguaje de señas. Las primeras señas que se aprenden son “Toma”, “Dame”, ”Hola”, ”Chao”, “Buenos Días”, “Buenas Noches”, “¿Cómo estás?”, “¿Dónde vives?”. Tomando el consejo de [13] las señas que se implementaron en el sistema propuesto son: “Toma/Dame”, ”Hola”, “Buenos Días” “Buenas Noches” y “¿Cómo estás?”, las mismas que, además de ser las primeras que se enseñan, constituyen también en un grupo de palabras que expresan cordialidad y normalmente son utilizadas para iniciar una conversación. La Figura 1.12 muestra las señas escogidas para validar el sistema. Finalmente en la entrevista realizada a [13], se manifestó que una de las grandes desventajas es que un 90% de los niños con discapacidad auditiva provienen de padres hablantes lo que conlleva a un serio problema puesto que los adultos desconocen la lengua de señas. Otro de los aspectos que dificulta el aprendizaje es las variaciones de este lenguaje de región a región.

Buenos Días

La mano desde la frente se desplaza hacia abajo y después hacia arriba mientras se abre simultáneamente

Hola

La mano con los dedo juntos se mueve de forma horizontal de un lado al otro varias veces

Entregar/Tomar

Las manos se deslizan hacia adelante en curva y se detienen

Buenas Noches

La mano en la frente se desplaza hacia abajo, las manos abiertas desde la altura de los hombros se desplazan hacia el centro mientras simultáneamente se cierran.

¿Cómo estás?

La mano se separa desde el pecho y se cierra el pulgar afuera. Los labios están en forma de beso abierto.

Figura 1.12. Señas Básicas del Lenguaje de Señas.
(Fuente: [12])

De lo expuesto, es importante para el sistema a implementar el tener una noción acerca del lenguaje de señas para determinar las diferencias más relevantes entre seña y seña lo cual será utilizado como patrón por el algoritmo de inteligencia artificial para su posterior reconocimiento y clasificación. Además, en esta sección, se delimita las señas a las mostradas en la Figura 1.12. las cuales han sido escogidas en base a la experiencia de un experto.

A continuación, se analizan los sensores 3D que sirven para obtener los datos de las coordenadas espaciales de la persona, los cuales serán ingresados a un algoritmo de Inteligencia Artificial para su reconocimiento y clasificación

1.2. Sensores 3D

Los sensores 3D son dispositivos capaces de medir profundidad, distancia o rango de un objeto. Para poder medir la profundidad el dispositivo ilumina el objeto con un laser u otra fuente de luz con el fin de medir la luz retro dispersada [14]. Existen dos tipos de sensores 3D:

- Sensores de luz proyectada: Estos sensores combinan la proyección de un patrón de luz con una cámara 2D estándar midiendo la profundidad vía triangulación.
- Sensores de tiempo de vuelo (TOF por sus siglas en inglés): Estos sensores emiten una luz la cual regresa al sensor tras rebotar en un objeto, la profundidad se estima basada en el tiempo que tarda la luz desde que sale del emisor hasta que llega al receptor.

En el mercado existen varios sensores 3D. En [15] se realizó una comparación entre los siguientes: Microsoft Kinect, ASUS Xtion Pro Live, Fotonic E70P, IFM O3D200 y Nippon Signal FX6. Para todos los sensores el ruido, el error y la velocidad de puntos detectados se evaluaron bajo diferentes condiciones como son: la distancia del sensor al objeto, la iluminación ambiental y la superficie del objeto. De todas estas condiciones según [15], la que más influye es la distancia del sensor al objeto. Finalmente se concluye que el más robusto es Nippon Signal FX6, el cual es confiable para la mayoría de condiciones, principalmente la distancia. Además, los sensores de luz proyectada Microsoft Kinect y ASUS Xtion Pro Live tienen una gran precisión para distancias de hasta 3.5 metros pero pueden presentar inconvenientes en exteriores puesto que no proporcionan ninguna medida de luz solar sobre un objeto. Los sensores Fotonic E70P y IFM O3D200 presentaron un menor desempeño frente a los demás.

En esta aplicación la distancia entre la persona y el sensor no superan los 3.5 metros, puesto que, ambas se encuentran en un ambiente cerrado lo que elimina también un

posible inconveniente generado por la luz solar. Tomando en cuenta estas consideraciones los mejores sensores para esta aplicación serían Microsoft Kinect y ASUS Xtion Pro Live, de los cuales se utilizará el Kinect debido a que sus librerías son de fácil acceso, además de que ya se lo ha utilizado en otras aplicaciones similares dando resultados satisfactorios; sin embargo, Microsoft por razones desconocidas descontinuó la producción de Kinect desde noviembre del 2017 por lo que en un futuro se debe considerar el uso de otros sensores.

1.2.1. Kinect

Es un dispositivo creado por la compañía Microsoft en el año 2011 capaz de reconocer gestos, comandos de voz, objetos e imágenes. Gracias a los componentes que lo integran como son: un sensor de profundidad, una cámara RGB, un array de micrófonos y sensores infrarrojos (ver Figura 1.13.). Puede también reconocer el esqueleto humano y posicionarlo en un plano permitiendo a la persona interactuar con elementos virtuales.

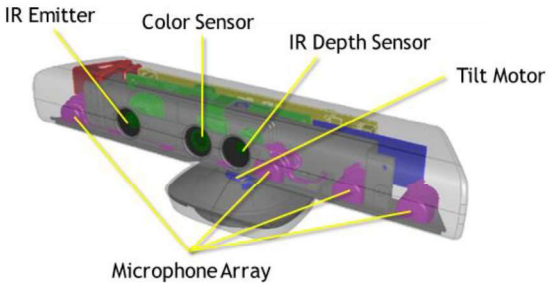


Figura 1.13. Kinect y sus Componentes.
(Fuente: [16])

Kinect fue originalmente diseñado para la consola de juegos X-box 360; sin embargo, sus desarrolladores han creado drivers de libre acceso los cuales permiten que otros usuarios creen aplicaciones a través de ellos. Estas librerías se denominan SDK (Software Development Kit) y existen varias versiones siendo la última la versión 2.0. Por otra parte en cuanto al hardware, existen dos versiones de Kinect la 1.0 y la 2.0. Para este sistema se utilizó la versión 1.0 de Kinect con la librería SDK 1.8.

Componentes y Especificaciones Técnicas del Kinect

Los principales componentes del Kinect son:

Video

- Cámara a color CMOS.
- Cámara infrarroja CMOS.
- Proyector infrarrojo -830nm, diodo laser de 60m W.

Audio

- Cuatro Micrófonos.

Inclinación

- Motor.
- Acelerómetro (3 Ejes).

Procesador y Memoria

- PrimeSense chip PS1080-A2.
- 64 MB DDR2 SDRAM.

Las especificaciones técnicas del Kinect se resumen en la Tabla 1.1.

Tabla 1.1. Especificaciones Técnicas de Kinect.

Kinect	Especificación
Cámara RGB	640x480 @ 30fps
Cámara Infrarroja	1280x1024 @ 30fps
Sensor Infrarrojo	Rango Operación 0.8m-3.5m
Campo de Visión	Horizontal 43º , Vertical 57º
Inclinación Vertical	± 27º
Formato de Audio	16-kHz, modulación mono pulso de 24 bits (PCM)
Características Audio de Entrada	4 micrófonos con conversor ADC de 24 bits incluyendo cancelación de eco y supresor de ruido
Características Acelerómetro	Acelerómetro 2G/4G/8G configurado para rango 2G

(Fuente: [16])

Los requisitos del sistema para que Kinect se ejecute sin ningún inconveniente son:

- Windows 8 o superior.
- Procesador de 64 bits.
- 4 Gb de memoria.
- Procesador físico dual core 3.1 GHz o superior.
- Controlador USB 3.0
- Software .NET framework 4.5
- Software SDK V2.0, V1.8

Para desarrollar una aplicación con Kinect es importante tomar en cuenta las especificaciones técnicas, requisitos del sistema, software necesario y recomendaciones de otros desarrolladores. Uno de los aspectos a considerar es el rango de visión del Kinect [17], lo más recomendable es que la persona se ubique en un rango entre 0.8 m y 3.5 m., para la visión horizontal, como se aprecia en la Figura 1.14. Verticalmente la Kinect posee motores lo cual hace posible un rango de 0.3m hasta 3.0 m como se puede apreciar en la Figura 1.15.

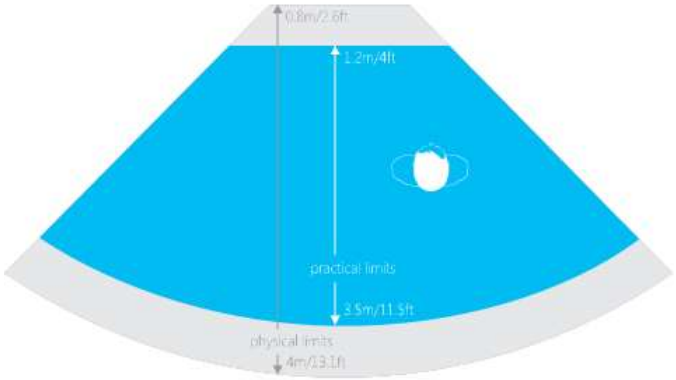


Figura 1.14. Rango de Visión Horizontal de Kinect.
(Fuente: [17])

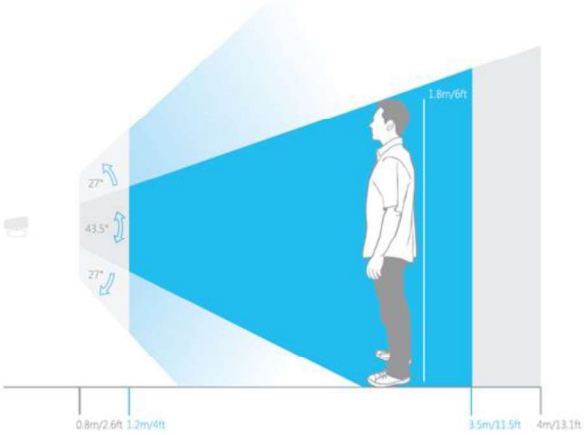


Figura 1.15. Rango de Visión Vertical de Kinect.
(Fuente: [17])

1.2.2. Skeletal Tracking

El Kinect tiene la capacidad de reconocer un máximo de 6 personas que se encuentren dentro del rango visible, de estas, puede calcular las posiciones del exoesqueleto de 2. Para representar el exoesqueleto por default la Kinect entrega las posiciones de 20 juntas referenciales [18], las mismas que se muestran en la Figura 1.16.

La representación de la Figura 1.16., corresponde a la primera versión del Kinect puesto que en su segunda versión es capaz de reconocer las manos y dedos lo que facilitaría el reconocimiento y clasificación de señas principalmente las de origen dactilográfico. Gracias a las librerías del SDK se puede tener acceso a las coordenadas espaciales x, y, z de los puntos referenciales.

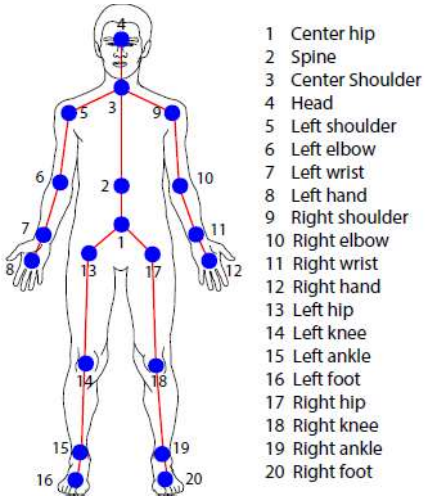


Figura 1.16. Representación Gráfica del Esqueleto.
(Fuente: [18])

Dado que el Kinect es un sensor 3D de tipo luz proyectada [14], este proyecta un patrón de luz sobre una escena y observa la deformación del patrón en la superficie del objeto. En la izquierda de la Figura 1.17. se muestra un patrón de líneas, a la derecha un patrón de puntos. Este patrón puede ser generado por un proyector LCD o por un barrido laser, una cámara ligeramente desplazada respecto al proyector captura la deformación de las líneas y calcula la distancia de cada punto utilizando una técnica de triangulación [19].

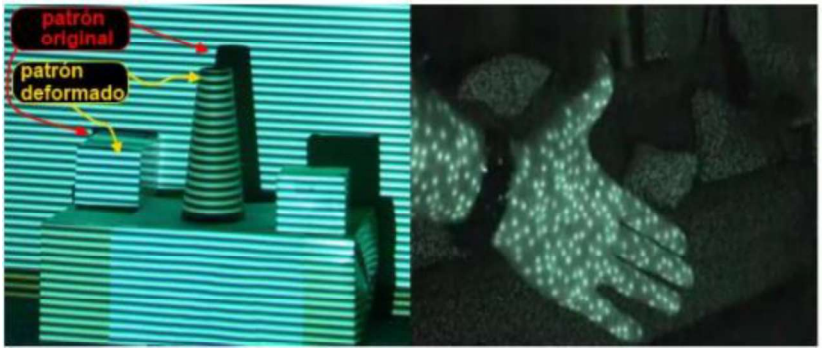


Figura 1.17. Patrón de luz proyectada por el sensor Kinect.
(Fuente: [19])

La Figura 1.18. muestra cómo se emplea la técnica de triangulación para el patrón de líneas. En una superficie plana se espera que la línea capturada por la cámara sea recta, una pequeña deformación en la misma puede ser convertida en una coordenada. Para esto se identifica cada línea con un método de reconocimiento de patrones [20].

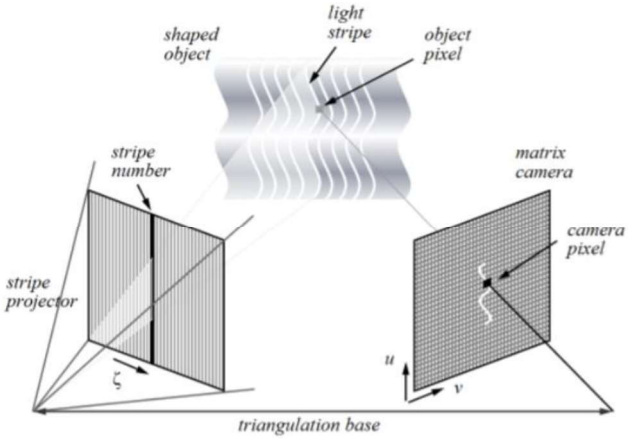


Figura 1.18. Principio de Triangulación Aplicado a patrones de Líneas.
(Fuente: [20])

El método de triangulación está basado en la semejanza de triángulos y determina la profundidad a la que se encuentra el objeto Z_p^M. Analizando los triángulos UVW y FVG de la Figura 1.19., en donde la relación entre profundidad y distancia focal quedaría descrita por la Ecuación 1.1, despejando la profundidad se obtendría la Ecuación 1.2. [21].

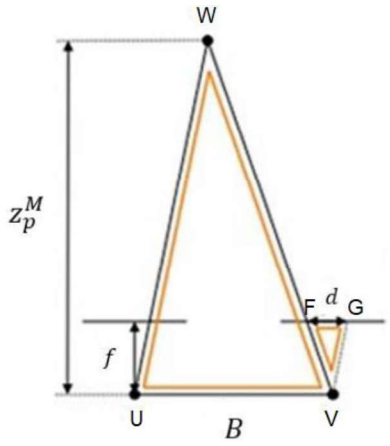


Figura 1.19. Triangulación para cálculo de profundidad.
(Fuente: [21])

$$\frac{B}{Z_p^M} = \frac{f}{d} \qquad \text{Ec. 1.1}$$

$$Z_p^M = \frac{f.B}{d} \qquad \text{Ec. 1.2}$$

En donde B es la distancia entre el emisor infra rojo y la cámara infra roja, f la distancia focal y d la disparidad que representa la diferencia relativa de un mismo pixel tomado por diferentes cámaras, en este caso la cámara IR y RGB.

Los datos obtenidos por el Kinect de sus cámaras RGB, IR y de profundidad se muestran en la Figura 1.20.

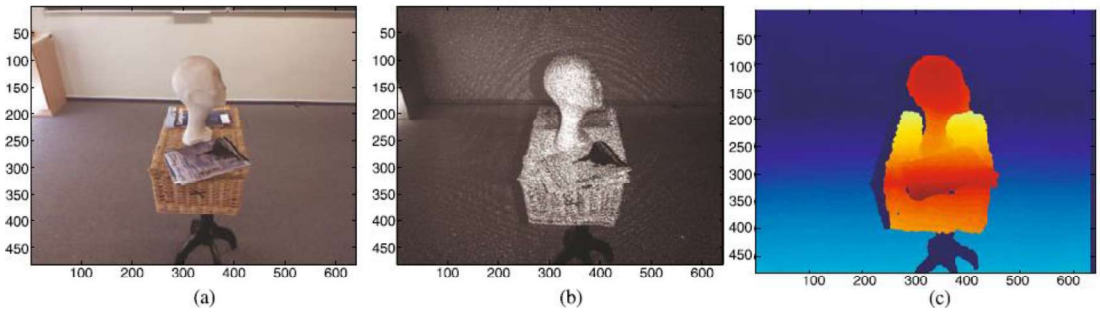


Figura 1.20. Imágenes Obtenidas por el Kinect (a) Imagen RGB, (b) Imagen Infrarroja, y (c) Imagen del proyector laser.
(Fuente: [22])

Una vez obtenida las imágenes y los datos sobre la profundidad de los objetos dentro de la imagen el siguiente paso es reconocer si en la imagen existe una persona para esto el equipo de investigación de Microsoft, en conjunto con los desarrolladores de Xbox en [23] explican de manera detallada como el Kinect es capaz de reconocer personas. A continuación se presenta un resumen corto de este procedimiento.

Las posiciones 3D de las articulaciones del cuerpo son obtenidas desde una imagen de profundidad mediante un algoritmo de reconocimiento de objetos que detecta las partes del cuerpo y las mapea independientemente de su postura o vestimenta, como se muestra en la Figura 1.21. Una vez detectadas y etiquetadas las partes del cuerpo las coordenadas del exoesqueleto se obtienen utilizando una clasificación de pixeles. El Kinect es capaz de reconocer el cuerpo humano a través de sus partes. Una imagen de profundidad es segmentada y etiquetada a una parte del cuerpo como se muestra en la Figura 1.22.

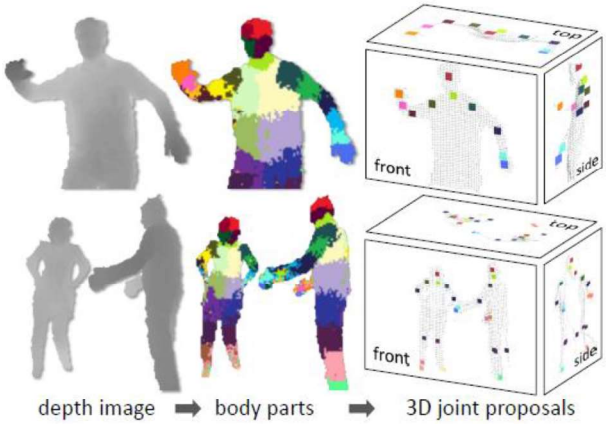


Figura 1.21. Imagen de Profundidad, Mapeo de partes del cuerpo y Juntas 3D.
(Fuente: [23])

Figura 1.22. Etiquetado y Segmentación de partes del cuerpo de una imagen de profundidad. (Fuente: [23])

Al tener identificadas y etiquetadas las partes del cuerpo humano se realiza un cálculo de su centro de masa, para determinar las coordenadas de los puntos de interés. Estos datos son almacenados en la memoria del Kinect, los cuales pueden ser extraídos mediante una serie de comandos.

La librería del SDK 1.8 posee varios comandos que permiten obtener los datos generados por el sensor. La Tabla 1.2. muestra todas las propiedades que se pueden obtener del Kinect.

Tabla 1.2. Comandos y Propiedades del Kinect.

Propiedades	Descripción
AudioSource	Obtiene el audio desde el sensor
ColorStream	Obtiene la imagen desde el sensor
CoordinateMapper	Obtiene el coordinador de coordenadas desde el sensor
DepthFilter	Establece un filtro para cada imagen de profundidad
DepthStream	Obtiene la imagen de profundidad del sensor
DeviceConnectionId	Obtiene el Id de la instancia de dispositivo USB
ElevationAngle	Cambia el ángulo de elevación de la cámara.
ForceInfraredEmiterOff	Enciende o apaga el emisor infrarrojo
IsRunning	Verifica si el Kinect esta enviando datos
MaxElevationAngle	Obtiene el ángulo máximo de elevación de la cámara en grados
MinElevationAngle	Obtiene el ángulo mínimo de elevación en grados.
SkeletonStream	Obtiene los datos del esqueleto
Status	Muestra el estado del sensor
UniqueKinectId	Obtiene el código único de los Kinects conectados

(Fuente: Propia)

El análisis y procesamiento de los datos obtenidos por el Kinect es a través de un algoritmo de Inteligencia Artificial, dicho análisis tiene como objetivo identificar qué serie de datos corresponde a una determinada seña, lo que permitirá su posterior reconocimiento y clasificación. En la sección siguiente se analizan las diversas técnicas de Inteligencia Artificial disponibles; además, se concluye cual es la más conveniente para este sistema.

1.3. Inteligencia Artificial

Inteligencia Artificial IA es un término originario del siglo XX que está asociado a técnicas y métodos computacionales con el propósito de emular el raciocinio humano para la toma de decisiones frente a diversos escenarios. Existen muchas definiciones para IA, por ejemplo, es “El análisis y el diseño de sistemas autónomos capaces de exhibir un comportamiento inteligente” [24]. A pesar que hasta la actualidad no se ha podido cumplir en su totalidad este objetivo existen técnicas que permiten la solución a problemas complejos, generalmente difíciles para otros métodos informáticos. El avance computacional ha permitido que la IA pueda estar presente de forma embebida en varios equipos de uso cotidiano como lavadoras, secadoras, hornos de microondas, procesadores de texto, etc. Además la IA ha sido utilizada para la calendarización de operaciones en naves espaciales, vehículos no tripulados, robots industriales, transporte, mantenimiento de telecomunicaciones, entretenimiento, entre otras. La Figura 1.23. muestra algunas de las aplicaciones en donde se ha utilizado la IA.

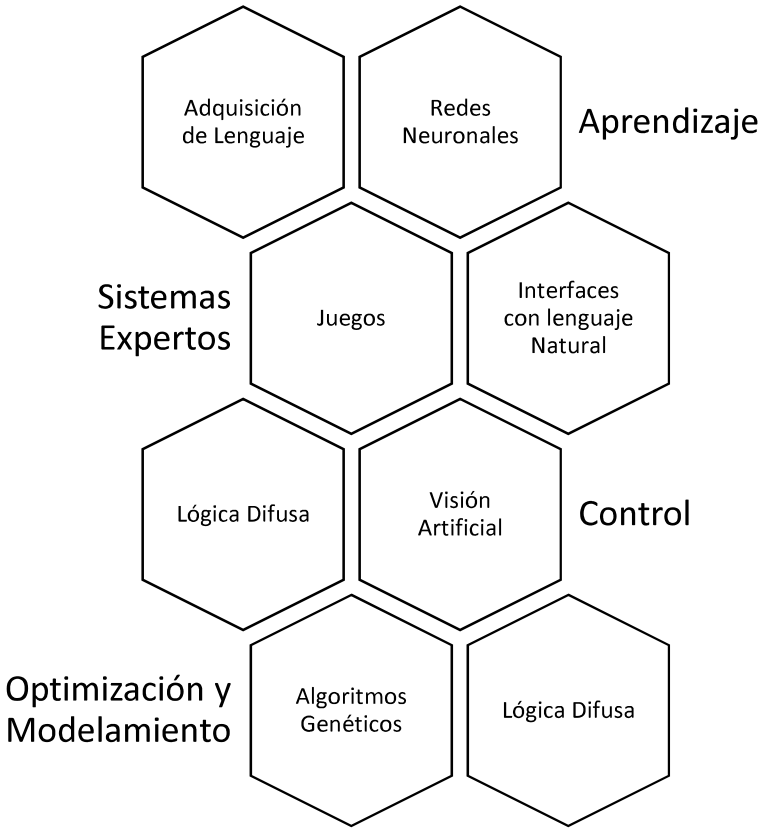


Figura 1.23. Aplicaciones de la IA.
(Fuente: Propia)

Para que un sistema sea catalogado como inteligente debe ser capaz de percibir, razonar, aprender, adaptarse, tomar decisiones y actuar racionalmente para satisfacer sus metas, en un determinado entorno [25]. Un sistema se puede considerar inteligente validando la prueba de Turing [26] en la que el sistema debe actuar, razonar y pensar como un humano.

El ser humano capta información del entorno para generar conocimiento a este procedimiento se lo conoce como aprendizaje. Es posible llegar al conocimiento a través de la compresión de patrones asociados con el lenguaje, hechos, conceptos, principios, procedimientos, reglas, ideas, abstracciones, lugares, costumbres y asociaciones. Para [25] el aprendizaje es un cambio en la conducta relativamente permanente resultado de la experiencia personal, pero que se refuerza con la interacción social.

Para algunos autores [27] y [28] , la inteligencia artificial tiene dos enfoques de clasificación. El primero utiliza representaciones simbólicas basadas en un número finito de reglas para la manipulación de símbolos como: redes semánticas, lógica de predicados, etc. Otro tipo de enfoque es el sub simbólico que utiliza representaciones numéricas, de donde se derivan sistemas con capacidad de aprendizaje como: las redes neuronales y los algoritmos genéticos [27].

En [28] se hace énfasis en algunas técnicas de inteligencia artificial como: lógica difusa, redes neuronales artificiales y algoritmos genéticos. La inteligencia artificial comprende un amplio campo de estudio, por motivos prácticos y siguiendo el criterio de [27] y [28] en este trabajo se analizarán las principales técnicas de inteligencia artificial que son:

- Lógica Difusa
- Redes Neuronales Artificiales
- Algoritmos Genéticos

A continuación se hace una breve presentación de cada una de estas ramas.

1.3.1. Lógica Difusa

La lógica convencional vista desde un enfoque computacional es una cadena de unos y ceros. Uno para proposiciones verdaderas y cero para proposiciones falsas; sin embargo; el ser humano tiene la capacidad de responder frente a circunstancias en las que se presenta cierta incertidumbre o juicio de valor, por ejemplo si se habla de la temperatura de un horno esta puede tomar varias variables lingüísticas, como “caliente”, “normal”, “frío”. Estas variables lingüísticas están regidas por reglas que determinan la salida del sistema, las reglas a su vez son determinadas por expertos. La lógica difusa para [28] es un conjunto de principios matemáticos basados en grados de membresía o pertenencia, cuya función

es modelar información. Se puede decir entonces que la lógica difusa es una extensión de la lógica convencional en la que existen valores intermedios entre verdadero y falso (1 y 0) para casos de incertidumbre en donde existe una verdad parcial.

Para comprender de mejor manera la lógica difusa es importante definir los siguientes términos:

- Variable Lingüística: Es una variable cuyos valores son palabras y permiten describir el estado de un objeto como: caliente, alto, medio, bajo, rápido, lento, fuerte, delgado, largo, cálido, fresco, etc.
- Fusificación: Proceso mediante el cual se asigna variables lingüísticas tanto a las salidas como a las entradas del problema.
- Conjunto Difuso: Es un conjunto en la cual los elementos lo integran de forma parcial o con cierto grado de pertenencia. Por ejemplo, la Figura 1.24. muestra los conjuntos difusos correspondientes a la velocidad angular de un motor, como se puede apreciar la velocidad actual pertenece en parte a los conjuntos difusos negativo bajo y cero. Las operaciones que pueden existir entre conjuntos difusos son unión, intersección, diferencia negación o complemento.

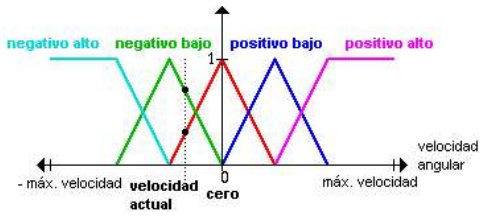


Figura 1.24. Conjuntos Difusos.
(Fuente: Propia)

- Reglas Difusas: Permiten crear relaciones entre las variables lingüísticas de entrada y las de salida. Tomando en cuenta el ejemplo anterior supongamos que se quiere controlar la posición de un péndulo invertido una de las reglas difusas sería, si el ángulo (entrada) es positivo bajo entonces la velocidad angular es negativa baja (salida).
- Función de Membresía: Es la función mediante la cual se relacionan los conjuntos difusos y puede ser rectangular, trapezoidal, triangular, tipo S, tipo Z y gaussiana como se muestra la Figura 1.25.
- Inferencia Difusa: Es el proceso de obtener un valor de salida para un valor de entrada utilizando la teoría de conjuntos difusos. Los tipos de inferencia más utilizados son el modelo de Mamdami y el de TSK (Takagi, Sugeno y Kang). [29]

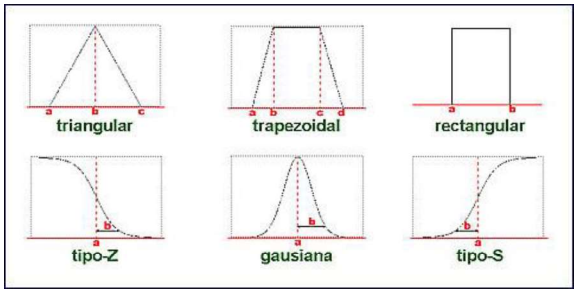


Figura 1.25. Funciones de Membresía.
(Fuente: [30])

La lógica difusa tiene aplicaciones en el campo de control, reconocimiento e identificación, transporte y logística, etc. En control su utilidad radica en que no necesita el modelo matemático de la planta para poder realizar un controlador, a su vez se basa en la experiencia del operador, su implementación es sencilla y tiene una alta adaptabilidad.

Sistemas Difusos

Un sistema difuso se compone de un conjunto de reglas difusas que transforman la entrada a salida, estas reglas tienen la forma IF THEN, por ejemplo: IFX es A THEN Y es B; de donde X es la entrada, Y es la salida, A y B son conjuntos difusos.

Para crear un sistema difuso se debe seguir el siguiente procedimiento:

1. Asignar variables lingüísticas a las entradas y a las salidas del sistema, este proceso es la Fusificación.
2. Determinar los valores de los conjuntos difusos correspondientes a las variables lingüísticas de entrada y de salida.
3. Escoger que función de membresía a utilizar.
4. Relacionar los conjuntos de entrada y salida a través de reglas difusas.
5. Determinar un método de inferencia difusa.
6. Calcular la salida.

Una de las características de los sistemas difusos es que todas las reglas se ejecutan en paralelo y contribuyen a generar la respuesta según su grado de pertenencia. Los conjuntos difusos resultantes de todas las reglas son acumulados en un solo conjunto difuso de salida el cual después de un método de razonamiento difuso obtiene el valor numérico de salida. Los métodos de razonamiento difuso para [25] pueden ser de implicación, agregación y defusificación. El más utilizado el método de defusificación por centroide en la que se obtiene el valor de salida segmentando las funciones de membresía en dos secciones según su grado de pertenencia, posteriormente se suma las secciones

inferiores de los conjuntos difusos y se calcula el centroide aplicando el principio de superposición

1.3.2. Algoritmos Genéticos

Los algoritmos genéticos (AG) son métodos adaptativos utilizados en problemas de búsqueda y optimización de parámetros [31]. Estos métodos son basados en la selección natural, lo que permite a los seres vivos ir evolucionando hasta tener características que los hacen óptimos para habitar en su medio. [32]

Normalmente la solución de un problema consiste en dar valores a una serie de parámetros, dichos parámetros para un algoritmo genético son conocidos como genes y son codificados a través de una serie de valores conocida como cromosomas. Cuando el cromosoma es representado mediante cadenas de códigos binarios recibe el nombre de genotipo, este contiene la información necesaria para la construcción de la solución real al problema o fenotipo, como se muestra en la Figura 1.26. [31]

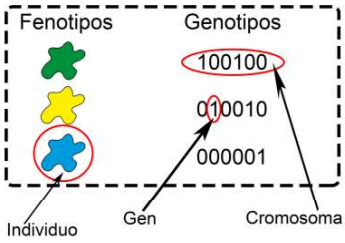


Figura 1.26. Codificación de problemas utilizando AG
(Fuente: [33])

Los cromosomas evolucionan a través de iteraciones conocidas como generaciones. Las nuevas generaciones son creadas utilizando operadores genéticos repetidamente. Según [34] estos operadores genéticos son: selección, cruce y mutación . Un algoritmo básico de AG consiste de los siguientes pasos: [28]

1. Selección/muestreo de padres
2. Cruce
3. Mutación
4. Selección de supervivientes/Evaluación.

Selección

El algoritmo de esta operación otorga un mayor número de oportunidades de reproducción a los individuos más aptos, incrementando la probabilidad de tener buenos individuos en la siguiente generación. Existen varios algoritmos de selección y pueden ser probabilísticos

o determinísticos siendo el más utilizado el método probabilístico de selección por ruleta. [31]

Cruce

Una vez determinados los individuos estos se recombinan con la finalidad de producir una descendencia que se insertara en la siguiente generación. Estos pueden recombinarse de tres formas: cruce de un punto, cruce de dos puntos y cruce uniforme. [31], [35].

Mutación

La mutación de un individuo consiste en alterar alguno de sus genes de forma aleatoria, con la finalidad de explorar dominios del problema no visitados aún. Normalmente la probabilidad de que la mutación mejore el resultado es menor al uno por ciento puesto que los individuos tienden a un ajuste menor después de ser mutados.

Evaluación

El apropiado funcionamiento del algoritmo genético requiere de un método que determine que individuo presenta una buena solución al problema y cual no, por lo tanto se debe emplear una medida que controle el número de operaciones genéticas que se van a llevar a cabo. Para [31], este control se lleva a cabo a través de una medida de ajuste (Fitness). Existen 4 tipos de ajuste: Fitness Puro, Fitness Estandarizado, Fitness Ajustado y Fitness Normalizado [36].

Una vez evaluada la nueva generación obtenida del cruce, se procede a seleccionar nuevos padres para la generación siguiente y este procedimiento se repite hasta que los individuos de la generación de como resultado un ajuste adecuado al problema.

1.3.3. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (ANN por sus siglas en inglés) son algoritmos computacionales que buscan emular ciertas características de los humanos, como la capacidad de memorizar, resolver problemas y asociar hechos [37]. Las ANN están formadas de varias unidades de procesamiento llamadas neuronas que al trabajar de manera simultánea buscan simular al funcionamiento del cerebro [38]. Su aplicabilidad comprende ramas como la medicina, industria, diagnóstico médico, robótica, control, reconocimiento, clasificación de datos, compresión, codificación de la información, etc.

Modelo de una neurona

Tomando como referencia el modelo propuesto por [39], una neurona recibe un conjunto de entradas $\{x_1, x_2, \dots, x_D\}$ y devuelve una única salida y, como se visualiza en la Figura 1.27. Esta neurona a su vez está relacionada con otras en mayor o menor intensidad

(sinapsis neuronal). La intensidad de la sinapsis entre neuronas se representa con un peso w_i, de manera que cada entrada tiene asociada un peso w_i.

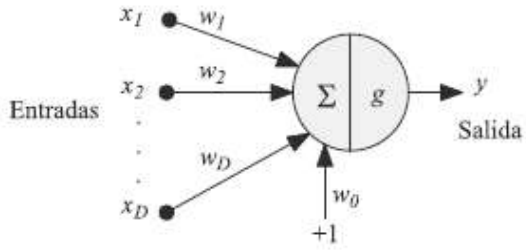


Figura 1.27. Modelo de una Neurona.
(Fuente: [24])

La activación a de la neurona equivale a la suma ponderada de sus entradas utilizando la Ecuación 1.3. En donde w_0 corresponde al umbral que se utiliza para compensar el valor medio de las entradas con el valor medio de las salidas deseadas.

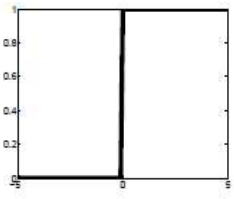
$$a = \sum_{i=1}^{D} w_i x_i + w_0 \qquad \text{Ec. 1.3}$$

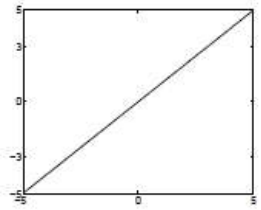
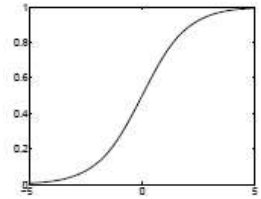
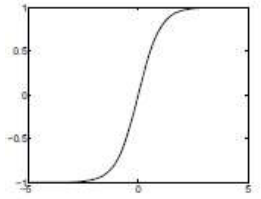
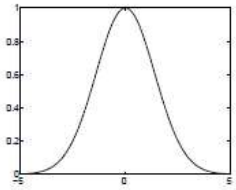
Partiendo del valor de a se obtiene la salida de la neurona y ,a través de una función de activación $g(a)$, como se muestra en la Ecuación 1.4.

$$y = g(a) = g\left(\sum_{i=1}^{D} w_i x_i + w_0\right) = g\left(\sum_{i=0}^{D} w_i x_i\right) \qquad \text{Ec. 1.4}$$

La función de activación puede ser: escalón, lineal, sigmoidea, tangente hiperbólica y gaussiana, como se muestra en la Tabla 1.3.

Tabla 1.3. Funciones de Activación.

Función	Modelo	Gráfico
Escalón	$g(a) = \begin{cases} 0 & para \quad a < 0 \\ 1 & para \quad a \geq 0 \end{cases}$	

Lineal	$g(a) = a$	
Sigmoidea	$g(a) = \frac{1}{1 + e^{-a}}$	
Tangente Hiperbólica	$g(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$	
Gaussiana	$g(a) = \exp\left(\frac{(a-\mu)^2}{2\sigma^2}\right)$	

(Fuente: Propia)

Clasificación de las Redes Neuronales Artificiales

Las redes neuronales artificiales poseen varias configuraciones; sin embargo, pueden clasificarse según el problema que se desea resolver. Una ANN puede utilizarse para: predicción, clasificación, asociación, conceptualización, filtrado y optimización [38]. La Tabla 1.4. muestra las ANN tomando en cuenta su aplicación.

Tabla 1.4. Clasificación de Redes Neuronales según su aplicación.

Nombre	Descripción	Propuesto por	Tipo
Adaline y Madaline	Técnicas de Adaptación para el reconocimiento de Patrones	Bernard Widrow	Predicción
Adaptive Resonance Theory (ART)	Reconocimiento de patrones y modelo del sistema Neuronal.	Carpenter, Grossberg	Conceptuali-zación

	Concepto de resonancia adaptiva		
Back Propagation	Solución a las limitaciones de su red predecesora el perceptrón	Rumelhart, Parker	Clasificación
Bi-Directional Associative Memory (BAM)	Inspirada en la red ART	Bart Kosko	Asociación
The Boltzmann Machine	Similar a la red Hopfield	Ackley, Hinton y Sejnowski	Asociación
Brain State in a Box	Red asociativa lineal	James Anderson	Asociación
Cascade Correlation	Adición de nuevas capas ocultas en cascada	Fahhman y Lebiere	Asociación
Counter Propagation	Clasificación adaptativa de patrones	Hecht, Nielsen	Clasificación
Digital Neural Network Architecture (DNNA)	Implementación hardware de la función sigmoid	Neural Semiconductor	Predicción
Directesd Random Search (DRS)	Técnica de valores Random Enel mecanismo de ajuste de pesos	Maytas, Solis	Clasificación
Hamming	Clasificador de vectores binarios utilizando la distancia Hamming	Lippman	Asociación
Hopfield	Concepto de la red en términos de energía	Hopfield	Optimización
Learning Vector Quantization (LVQ)	Red clasificadora	Kohonen	Clasificación
Perceptron Networks	Primer modelo de sistema neuronal artificial	Rosenblatt	Predicción

Recirculation Networks	Alternativa de la red Back Propagation	Hinton, MacCLelland	Filtrado
Self Organizing Maps (SOM)	Aprendizaje sin supervisión	Kohonen	Clasificación

(Fuente: [38])

Analizando la Tabla 1.4. las Redes Neuronales que se podrían utilizar para el reconocimiento y clasificación de lenguaje de señas son: Back Propagation, Counter Propagation y Learning Vector Quantization (LVQ). Para comprender el funcionamiento de dichas redes neuronales es importante empezar analizando el perceptrón simple.

Perceptrón Simple

El perceptrón corresponde al primer modelo de ANN, planteado por Rosenblatt en 1962 y consiste en una capa de neuronas j con funciones $\emptyset_j$ que transforman los datos de entrada y una capa de salida, como se muestra en la Figura 1.28. La función de activación para este caso es la escalón. La importancia de su estudio radica en que todos los modelos posteriores se basan en su funcionamiento.

Para explicar su funcionamiento se toma como ejemplo simular las funciones lógicas AND y OR. Los valores de los pesos w_1 y w_2 en este caso van a ser iguales, únicamente variando el umbral w_0 se puede lograr que la neurona se comporte como compuerta OR o AND, como se muestra en la Tabla 1.5. Tomando en cuenta que la función de activación es cero para $a < 0$ y uno para $a \geq 0$, el umbral necesario para la compuerta OR sería -0.5 mientras que para la compuerta AND sería -1.5.

Tabla 1.5. Red Neuronal Perceptrón Simple, $w_0 = -1.5$ para compuerta AND y $w_0 = -0.5$ para compuerta OR.

x_1	x_2	w_1	w_2	$\sum_{i=1}^{2} w_i x_i$	$\sum_{i=1}^{2} w_i x_i + w_0$	AND	$\sum_{i=1}^{2} w_i x_i + w_0$	OR
1	1	1	1	2	0.5	1	1.5	1
1	0	1	1	1	-0.5	0	0.5	1
0	1	1	1	1	-0.5	0	0.5	1
0	0	1	1	0	-1.5	0	-0.5	0

(Fuente: Propia)

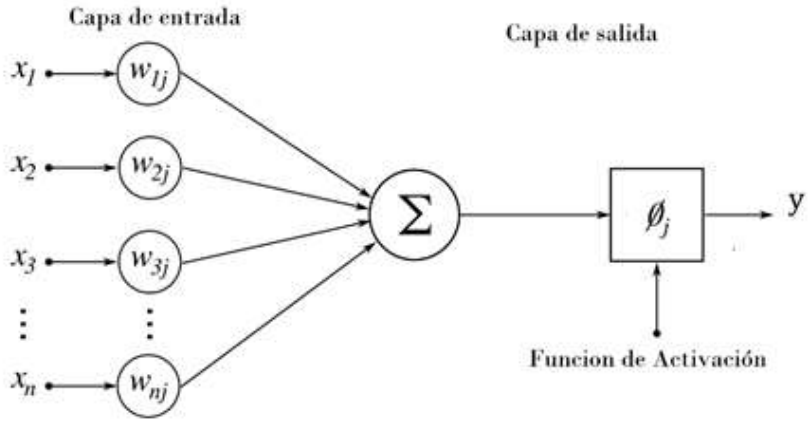


Figura 1.28. Estructura de un Perceptrón Simple.
(Fuente: [40])

El perceptrón simple es capaz de clasificar elementos linealmente separables; sin embargo, muy pocos problemas son de esta naturaleza puesto que la mayoría son no lineales e incluso inseparables como se muestra en la Figura 1.29.

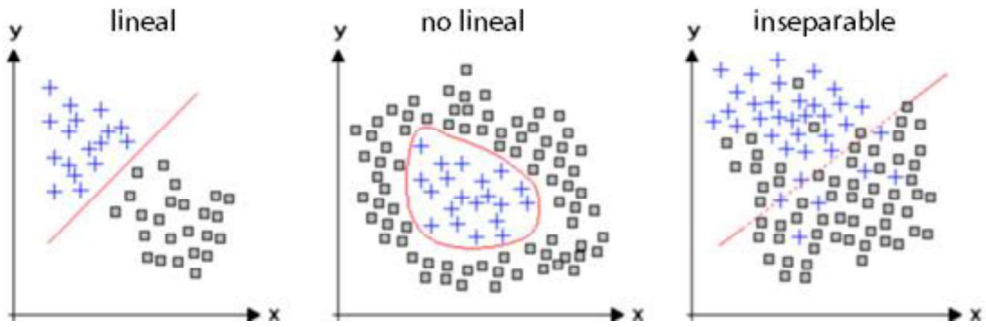


Figura 1.29. Separabilidad en Clasificación.
(Fuente [41])

Perceptrón Multicapa

El perceptrón multicapa MLP (Multi Layer Perceptron), incluye una o varias capas intermedias de neuronas conocidas también como capas ocultas (ver Figura 1.30.). Dichas capas permiten separar los datos utilizando varias líneas o polígonos de manera que la salida pueda ser una clasificación correcta. Esta característica hace que el modelo MLP sea conocido como un aproximador de funciones universal. [24] [42]

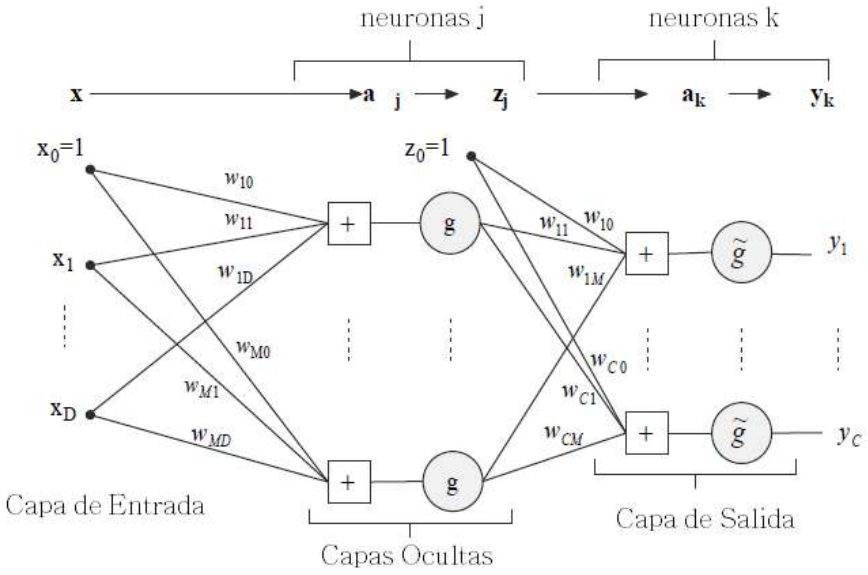


Figura 1.30. Estructura de una Red Neuronal MLP.
(Fuente: [24])

Si se considera una MLP con una única capa oculta, con D entradas, M número de neuronas en su capa oculta y C número de salidas. El nivel de activación a_j de la neurona j de la capa oculta es una combinación lineal de las D entradas x_i que recibe sobre las que, tras aplicar una función de activación g se obtiene la salida z_j. La salida de la neurona j en la capa oculta se expresa con la Ecuación 1.5 y las funciones de activación g se muestran en la Tabla 1.3.

$$z_j = g(a_j) = g\left(\sum_{i=0}^{D} w_{ji} x_i\right);\ \ k = 1,2,3 \ldots, M. \qquad \text{Ec. 1.5}$$

En donde w_{ji} es el peso asociado a la neurona j y a la entrada x_i. De la misma forma la salida de la red será la suma ponderada de las salidas de las neuronas de la capa oculta, sobre la que se le aplica una función de activación g. La salida de una neurona k se representa con la Ecuación 1.6.

$$y_k = \tilde{g}(a_k) = \tilde{g}\left(\sum_{j=0}^{M} w_{kj} z_j\right) = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}\, g\left(\sum_{i=0}^{D} w_{ji} x_i\right)\right);\quad k = 1,2,3 \ldots, C. \qquad \text{Ec. 1.6}$$

Redes Back Propagation (BP)

Back propagation es un algoritmo de entrenamiento supervisado para redes multicapa que busca determinar los pesos de las conexiones de manera que la salida de la red coincida con la deseada a través de una regla de ajuste del error. El ajuste de los pesos comienza

en la capa de salida y se va propagando hacia las capas anteriores hasta llegar a la capa de entrada de ahí el nombre de retro propagación. [43]

Este algoritmo de aprendizaje se basa en el descenso del gradiente para ajustar los pesos de la red. Inicialmente en su primera iteración se establecen unos pesos aleatorios de valor a la red así como también aplica un patrón a la entrada de la red para posteriormente obtener su salida aplicando la Ecuación 1.6. En las siguientes iteraciones se recalculan los pesos siguiendo la regla de ajuste del error que compara la salida obtenida con la deseada. Una de las funciones de error más utilizada es la de la Ecuación 1.7.

$$E^n = \frac{1}{2}\sum_{k=0}^{C}(e_k^n)^2 \qquad \text{Ec. 1.7}$$

En donde el factor $^1/_2$ se introduce para facilitar cálculos posteriores, el error e_k^n es la diferencia entre la salida deseada t_k^n y la salida obtenida y_k^n como se muestra en la Ecuación 1.8.

$$e_k^n = y_k^n - t_k^n \qquad \text{Ec. 1.8}$$

La Figura 1.31. muestra el descenso del gradiente para una red con dos pesos, w_1 y w_2, las flechas indican el sentido en el que el gradiente va descendiendo. Para este caso el punto 1 es un mínimo global mientras que los puntos 2 son mínimos locales.

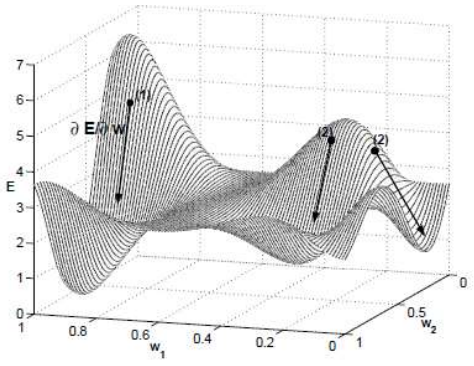


Figura 1.31. Descenso de gradiente en una red con dos pesos.
(Fuente: [24])

En una iteración n se añade un incremento Δw_{kj} al peso sináptico, dicho incremento es proporcional al gradiente $\partial E/\partial w_{kj}$ el cual determina la dirección en la que el valor del peso w_{kj} conduce a un mínimo valor de E. Aplicando la regla de la cadena este gradiente se representa en la Ecuación 1.9 y se obtiene derivando las Ecuaciones 1.6,1.7 y 1.8.

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = -e_k\widetilde{g'}(a_k)z_j \qquad \text{Ec. 1.9}$$

La Ecuación 1.10 corresponde a la definición de gradiente local δ de una neurona k.

$$\delta_k = \frac{\partial E}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial a_k} = e_k \widetilde{g}'(a_k) \qquad \text{Ec. 1.10}$$

El incremento Δw_{kj} que se debe aplicar al peso w_{kj} es:

$$\Delta w_{kj} = \eta \frac{\partial E}{\partial w_{kj}} = \eta \delta_k z_j \qquad \text{Ec. 1.11}$$

En donde η es el factor de aprendizaje. De la Ecuación 1.11, se puede concluir que para obtener el incremento Δw_{kj} se debe multiplicar el gradiente local δ_k para la unidad de salida de la neurona por el valor de z de la unidad de entrada de la neurona.

Redes Counterpropagation (CP)

Las redes contra propagación es un algoritmo de aprendizaje que combina las técnicas de aprendizaje supervisado y no supervisado. Su estructura proviene de una combinación del mapa auto organizativo de Kohonen [44] y la estructura de salida en estrella de Grossberg [45].

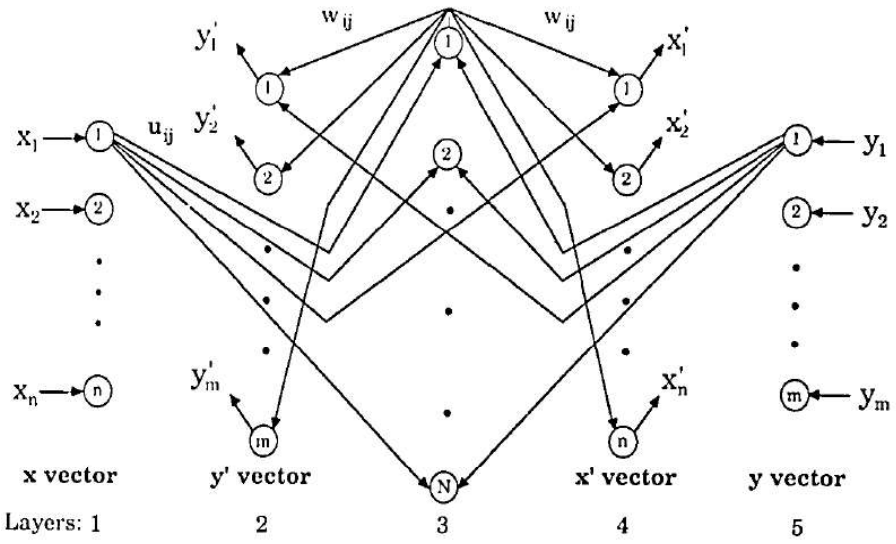


Figura 1.32. Estructura de una red Contra Propagación.
(Fuente: [46])

Durante el aprendizaje, el vector de entradas x_n y el vector de salidas y_m se presentan a las capas de entrada (1) e interpolación (5), respectivamente. Estos vectores se propagan a través de la red en sentidos opuestos para transformarse en vectores de salida x'_n y y'_m, como se muestra en la Figura 1.32.

Las neuronas de la capas 2 y 4 reciben n entradas de cada elemento de procesamiento de la capa 3 (las señales de salida del elemento de procesamiento de la capa 3 están etiquetadas como z_i). Cuando un par de vectores x_n y y_n se presentan en la red, las neuronas de la capa 3 compiten entre sí siguiendo la Ecuación 1.12.

$$z_i = \begin{cases} 1 \; if \; I_i > I_j \\ 0 \; otherwise \end{cases} \quad I_i = \sum_{j=1}^{n} u_{ij} x_j + \sum_{j=1}^{m} v_{ij} y_j \qquad \text{Ec. 1.12}$$

La neurona con las correlaciones de vectores de mayor peso promedio (es decir, el valor más alto de I_i) tiene su señal de salida z_i establecida en uno. Todas las otras salidas z_i se ponen a cero. Aplicando la ley de cambio de peso [44], se obtienen los valores de los pesos u o v para dicha neurona, ver Ecuación 1.13.

$$\dot{u_i} = \propto (x - u_i) z_i \;\; \dot{v_i} = \beta (y - v_i) z_i \qquad \text{Ec. 1.13}$$

En donde $\propto$ y β representa la fracción de la distancia u_i, v_i y sus objetivos u_j, v_j, como se aprecia en la Figura 1.33.

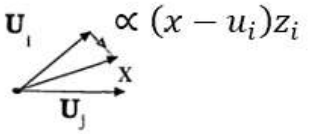


Figura 1.33 Ley de Cambio de peso de Kohonen
Fuente: [44]

Finalmente los pesos w_{ij} se ajustan hasta que las salidas x'_n y y'_m se aproximan a los vectores x_m y y_m utilizando la Ecuación 1.14.

$$\dot{y_i} = \sum_{j=1}^{n} w_{ij} \, z_j \qquad \text{Ec. 1.14}$$

Redes Learning Vector Quantization (LVQ)

Estos tipos de Redes es una evolución de los mapas de rasgos auto organizados (SOFM por sus siglas en inglés). Esta red se compone por una capa de entrada seguida de una capa simple de neuronas de Kohonen sin relaciones de vecindad como se muestra en la Figura 1.34. [47].

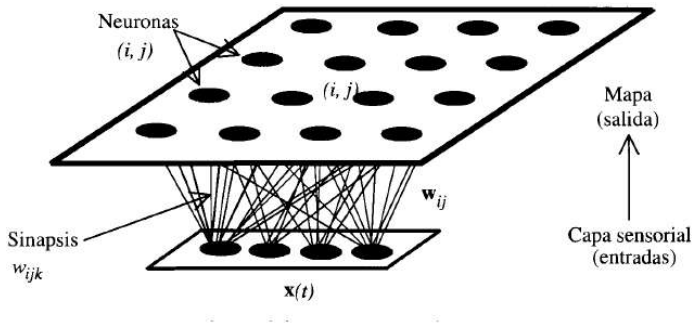


Figura 1.34. Red Neuronal LVQ.
(Fuente: [47])

Supongamos que cada una de las neuronas de entrada n representa una clase, y almacena un vector de referencia w_j que representa el prototipo de una clase. Cuando se presenta a la red un patrón de entrada x, compuesto por m entradas, cada neurona calcula la distancia que separa su vector de referencia del de entrada utilizando la Ecuación 1.15.

$$d^2(w_j, x) = \sum_{k=1}^{m} (w_{ik} - x_k)^2 \qquad \text{Ec. 1.15}$$

El entrenamiento se basa en premiar a aquellas neuronas que clasifican correctamente un determinado patrón, actualizando sus pesos con la regla convencional y castigar a las que realizan una clasificación errónea siguiendo la regla de Kohonen. Si se dispone de p patrones de aprendizaje $x^u\ (u = 1, \dots, p)$ y cada patrón pertenece a una clase C_{xu} conocida, habiendo en total n clases; de la misma manera, cada neurona de pesos w_j es representada por una clase C_{wc}. Si w_c es el vector de referencia más próxima al patrón de entrada x^u presentado, se denomina C_{wc} a la clase definida por dicho vector de referencia por lo tanto la neurona c se actualiza de la siguiente manera:

- Si $C_{wc} = C_{xu}$ entonces la clasificación es correcta por lo tanto el vector de referencia se modificará con la Ecuación 1.16.

$$W_c(t+1) = W_c(t) + \alpha_t[x^u - W_c(t)] \qquad \text{Ec. 1.16}$$

- Si $C_{wc} = C_{xu}$ significa que la clasificación es incorrecta, por lo que la modificación se realizará en sentido contrario como indica la Ecuación 1.17.

$$W_c(t+1) = W_c(t) - \alpha_t[x^u - W_c(t)] \qquad \text{Ec. 1.17}$$

Este proceso se repetirá iterativamente para todos los casos de entrenamiento, para establecer vectores de referencia iniciales w_j pueden emplearse vectores de ejemplo x^u

Comparación entre Arquitecturas para Reconocimiento y Clasificación

El uso del algoritmo de retro propagación o contra propagación depende de la aplicación específica. Por ejemplo [48] determinó que la red contra propagación tiene una mejor capacidad de aprendizaje para modelar el proceso de soldadura TIG; sin embargo, la red de retro propagación tiene una mejor capacidad de generalización para diversas configuraciones del proceso de soldadura TIG. Por otra parte [49] realizó una clasificación filogenética de secuencias de ARN ribosómicas determinando que la red de retro propagación es más pequeña ya que no necesita tantas neuronas en la capa oculta como la de contra propagación que al menos necesita una neurona por cada capa de salida, a pesar de esto, la red contra propagación se entrena a la mitad de tiempo que la retro propagación, tomando en cuenta que ambos algoritmos tuvieron una precisión del 100% la red de retro propagación tuvo una mayor velocidad de respuesta (0.12s vs 0.26s).

Otro estudio en donde se comparan varios tipos de redes neuronales utilizados en predicción de fallas de líneas de transmisión de energía [50], determina que las redes CP y LVQ tienen un aprendizaje más rápido, similar tasa de error y buena robustez; sin embargo, la red BP es más compacta y se espera que funcione mucho más rápido en tiempo real.

La red CP tiene un comportamiento similar a LVQ, puesto que las dos utilizan una capa de Kohonen, por lo que para fines prácticos se comparará únicamente entre BP y LVQ. La Tabla 1.6 muestra una comparación entre una red BP y dos redes LVQ, los datos mostrados han sido obtenidos mediante simulación en Matlab y se encuentran disponibles en el Anexo 1. Las configuraciones de las redes probadas se detallan a continuación:

La red BP_30 tiene la siguiente configuración:

- Entrada: 180 neuronas.
- Primera capa oculta: 30 neuronas.
- Segunda capa oculta: 5 neuronas.
- Capa de salida: 5 neuronas.

La red LVQ_30 tiene la siguiente configuración:

- Entrada: 180 neuronas.
- Primera capa oculta: 30 neuronas.
- Segunda capa oculta: 5 neuronas.
- Capa de salida: 5 neuronas.

La red LVQ_60 tiene la siguiente configuración:

- Entrada: 180 neuronas.
- Primera capa oculta: 60 neuronas.
- Segunda capa oculta: 5 neuronas.
- Capa de salida: 5 neuronas.

Tabla 1.6. Comparación entre BP_30, LVQ_30 y LVQ_60.

Característica	BP_30	LVQ_30	LVQ_60
Tiempo de entrenamiento para el mejor desempeño	1 minuto, 45 segundos	55 segundos	12 segundos
Número de iteraciones	15	115	9
Error mínimo cuadrático	2.16e-09	0.002963	0.00741
Cantidad de Neuronas	220	220	250

(Fuente: Propia)

Analizando la Tabla 1.6, se puede notar que las 3 redes permiten reconocer y clasificar la lengua de señas puesto que el error mínimo cuadrático es mínimo; sin embargo la que menor error presenta es la red BP_30 con un error de 2.16e-9. La única desventaja que presenta la red BP es el tiempo de entrenamiento necesario que supera a la red LVQ_30 con 50 segundos. La red LVQ_60 a pesar de que mejora considerablemente el tiempo de entrenamiento frente a las otras dos, aumenta la cantidad de neuronas necesarias y aumenta su error.

Luego de considerar los tres tipos de redes neuronales se determina que el perceptrón multicapa entrenado mediante retro propagación es el que menos error cuadrático presenta, además el cálculo de salida es más sencillo comparado con la red LVQ por lo que se espera que funcione más rápido en tiempo real. Finalmente el tiempo de entrenamiento necesario para esta red no es un limitante para su implementación.

1.3.4. Clasificación y Reconocimiento Utilizando IA

A continuación se analizarán las diferentes técnicas de inteligencia artificial y su aplicación en reconocimiento y clasificación de patrones. Además se seleccionará el más apropiado para esta aplicación específica.

Lógica Difusa

En aplicaciones de reconocimiento es importante establecer un criterio de clasificación, como se mencionó anteriormente no todas las variables son numéricas pues existe también variables lingüísticas cuyo grado de pertenencia no necesariamente se satisface con la lógica convencional. La lógica difusa permite calcular o hallar la relación entre estas variables lingüísticas y su grado de pertenencia con los conjuntos difusos planteados como patrones de clasificación. Para explicar de mejor manera, [51] plantea el siguiente ejemplo: Supongamos que se requiere clasificar personas, de las cuales se puede obtener la siguiente información: peso, altura, genero, religión, educación, apariencia. Esta información se convierte en variables numéricas y lingüísticas. Si se desea clasificar según el tamaño (grandes, medianas y pequeñas) únicamente se necesitaría el peso y la altura, por lo cual resulta bastante sencillo; sin embargo, supongamos ahora que se requiere clasificar a las personas como bueno o malos vecinos, en este caso el número de variables o características a utilizarse no está claro por lo que se convierte en un problema de clasificación.

Normalmente en los problemas de clasificación el número y tipo de características así como el criterio de clasificación varían a medida que se siguen manipulando los datos. Para [51] los métodos de clasificación más utilizados son dos: la clasificación utilizando relaciones equivalentes [52] y el segundo método es conocido como FCM (Fuzzy C-means) planteado por [53] . A pesar de que es posible clasificar utilizando lógica difusa no hay un trabajo previo en clasificación de lenguaje de señas; en consecuencia, resulta complicado establecer unas reglas difusas que puedan aplicarse de manera general, puesto que si se siguen los métodos mencionados se obtendrán una serie de reglas válidas únicamente para las señas ingresadas y si se desearía ingresar una seña adicional se deberían recalibrar todas las reglas anteriores lo cual resulta difícil a medida que las señas que se vayan a clasificar aumenten. Por esta razón la Lógica Difusa no permite el reconocimiento y clasificación del lenguaje de señas de una manera óptima.

Algoritmos Genéticos

Como se indicó anteriormente los AG tienen un campo de aplicación en tareas de búsqueda y optimización de parámetros; sin embargo, pueden trabajar en conjunto con otras técnicas de inteligencia artificial para lograr reconocimiento y clasificación. Por ejemplo [54] utilizó un algoritmo genético para determinar el número óptimo de sentencias if-then en un sistema de clasificación difuso, las reglas difusas y sus funciones de membresía son obtenidas por el AG de una serie de datos y no de la experiencia de un experto, logrando

reducir hasta en un 98% las reglas difusas utilizadas. Por otra parte [55], utilizó un AG para entrenar una red multicapa de retro propagación, a pesar de que el algoritmo de retro propagación establece los pesos de manera óptima y en un tiempo corto, el algoritmo genético tuvo una mejor convergencia ya que es capaz de trabajar con funciones de activación discontinuas; sin embargo, uno de los problemas encontrados fue que el algoritmo genético debe ser modificado para secuencias de datos de entrenamiento continuamente cambiantes. Cambiar el algoritmo genético cada vez que se requiera ingresar una nueva seña resulta una tarea muy compleja y dificultaría su implementación. Por esta razón los algoritmos genéticos no son la mejor opción para el reconocimiento y clasificación de lengua de señas.

Redes Neuronales Artificiales

A diferencia de la lógica difusa las redes neuronales artificiales permitirían de mejor manera anexar más señas al algoritmo de reconocimiento y clasificación puesto que únicamente se necesitaría añadir más neuronas, lo cual se ajusta en mejor medida al problema en cuestión. Si bien es cierto en este trabajo se va a clasificar un número pequeño de señas representativas lo óptimo sería que el sistema pueda ingresar más señas de manera sencilla y con un bajo coste computacional. Además hay cierta incertidumbre sobre las reglas difusas que deberían ingresar al sistema, mientras que usando redes neuronales el ajuste de pesos se realiza minimizando el error como se mostró anteriormente.

Finalmente los algoritmos genéticos tienen una aplicación enfocada a búsqueda y optimización de parámetros, más no a reconocimiento y clasificación. Por estas razones la mejor opción para reconocer y clasificar la lengua de señas son las redes neuronales. En este sistema se implementará una red neuronal multicapa entrenada mediante retro propagación, debido a que para esta aplicación específica demuestra un mejor desempeño sobre otras.

2. METODOLOGÍA

En esta sección se detallan los aspectos considerados durante el diseño del sistema. Se desarrolló un HMI en Visual Studio que permite tomar y exportar los datos del exoesqueleto en tiempo real, dichos datos son utilizados para diseñar el algoritmo de Inteligencia Artificial, el cual basado en el anterior capítulo corresponde a una red neuronal multicapa entrenada mediante retro propagación. El diseño de la red neuronal y su entrenamiento fueron desarrollados en Matlab, posteriormente los pesos y umbrales obtenidos como resultado del entrenamiento se exportaron al sistema para las pruebas pertinentes. A continuación se detallan tanto el hardware como el software utilizados en el sistema.

2.1. Hardware

El hardware utilizado en este sistema consta de lo siguiente:

- Un trípode que sostiene el Kinect en una posición adecuada. Tomando en cuenta la recomendación del fabricante se realizaron pruebas en diferentes posiciones siguiendo las Figuras 1.14 y 1.15. En donde se determinó que la mejor posición es ubicando el sensor a 1.15 metros de altura respecto al suelo, mientras que, la persona se debe ubicar a 1.8 metros del sensor, con esto se asegura una correcta identificación del exoesqueleto.
- El sensor Kinect, cuyas especificaciones técnicas fueron mencionadas en la Sección 1.2.1 de este documento.
- Una pantalla o proyector (opcional).
- Una computadora marca ASUS con características resumidas en la Tabla 2.1.

Tabla 2.1. Características de la Computadora utilizada para el sistema.

Sistema Operativo	Windows 10 Pro
Memoria RAM	8,00 GB
Tipo de Sistema	Sistema Operativo de 64 bits, procesador x64
Procesador	Intel(R) Core(TM) i-7 @ 2.40 GHz
Memoria en disco Duro0	1 TB
Tarjeta gráfica	Nvidia geforce 940M

(Fuente: Propia)

Estos elementos se pueden visualizar en la Figura 2.1. La pantalla o el proyector es para que los usuarios puedan visualizar la aplicación, en especial la señal de cuando empezar a realizar una seña y el resultado de la clasificación.

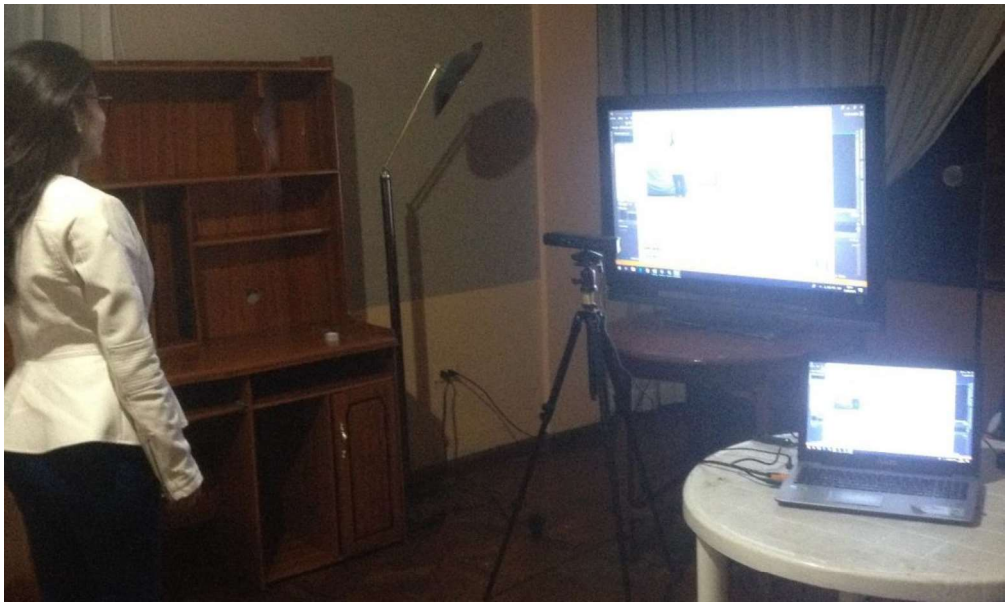


Figura 2.1. Elementos del sistema.
(Fuente: Propia)

2.2. Software

Los programas utilizados para crear el sistema son los siguientes: Visual Studio 2015, Matlab 2017, y el SDK para Kinect versión 1.8.

Como se mencionó anteriormente la interfaz con el usuario (HMI) fue desarrollada en Visual Studio mientras que el diseño de la red neuronal se lo realizo en Matlab. El SDK permite a Visual Studio acceder a los datos del Kinect mediante las librerías mencionadas en la Tabla 1.2.

2.2.1. Adquisición de Datos

Un punto importante a consideración es el análisis de la naturaleza de los datos con la finalidad de escoger las variables más apropiadas que permitan a la red neuronal diferenciar entre seña y seña. En el capítulo anterior se determinó que para representar las señas se utilizan los movimientos de las extremidades superiores, principalmente las manos y codos, por lo que el resto de puntos no se tomarán en cuenta.

Analizando el diccionario de lengua de señas se concluye que la duración promedio de una seña es de 3 segundos siendo como máximo 5 segundos y como mínimo 1.5 segundos . El Kinect es capaz de generar un máximo de 30 fps (frames per second); es decir, se generarán 30 esqueletos por segundo. Por ejemplo, para un movimiento circular de la mano en sentido horario se obtuvieron los datos mostrados en la Tabla 2.2.

Tabla 2.2. Datos de la mano izquierda @30Fps.

No.	Posición X	Posición Y	Posición Z
1	-0,218	0,458	1,256
2	-0,252	0,460	1,270
3	-0,286	0,455	1,288
4	-0,309	0,438	1,309
5	-0,331	0,409	1,335
6	-0,358	0,386	1,359
7	-0,342	0,336	1,396
8	-0,345	0,308	1,418
9	-0,359	0,280	1,434
10	-0,333	0,239	1,458
11	-0,363	0,204	1,468
12	-0,384	0,165	1,479
13	-0,322	0,126	1,490
14	-0,350	0,079	1,493
15	-0,322	0,042	1,494

(Fuente: Propia)

Tomando en cuenta que los 15 datos de la Tabla 2.2 fueron generados en 0.5 segundos, se puede notar que la variación máxima entre movimientos es de alrededor de 0.04 metros, lo cual es muy pequeño. Por esta razón el muestreo se lo realizó a 6 datos por segundo; además, con esto se consigue optimizar recursos computacionales. Una seña dura como máximo 5 segundos de manera que se tomaran un total de 30 datos para cada coordenada que se desee analizar.

Los datos exportados corresponden a las coordenadas xyz, de los puntos de interés: mano derecha y mano izquierda. Como se estableció previamente, una duración máxima de 5 segundos a una tasa de muestreo de 6 datos/segundo, da como resultado que el número de datos exportados será de 90 por cada punto de interés, lo que suma un total de 180 datos. Estos datos serán utilizados para entrenar a la red neuronal como se explica posteriormente.

2.2.2. Preprocesamiento de Datos

El preprocesamiento de los datos facilita la tarea de diseño de la red neuronal. Acorde a [56] el procesamiento de datos presenta las siguientes ventajas: ayuda a disminuir el tiempo de entrenamiento de la red neuronal. Existen varias técnicas de preprocesamiento de los datos como son: Data Mining, Data Warehousing, entre otras. En el nprtool de Matlab se realiza un preprocesamiento de los datos con la finalidad de limitar su amplitud, además también, mueve la ubicación de ciertos puntos para que el problema se asemeje a uno

separable de forma no lineal como se mostró en la Figura 1.28. Estas aproximaciones se las realiza sumando un valor a cada entrada (offsets) para posteriormente multiplicar el resultado por una ganancia Estos datos se generan en la fase de entrenamiento.

2.2.3. Diseño de la Red Neuronal

El algoritmo de AI que se implemento es una red neuronal perceptrón multicapa entrenada mediante retro propagación. La arquitectura de la red neuronal es la mostrada en la Figura 2.2. La red neuronal propuesta consta de 180 entradas, la primera capa oculta tiene 30 neuronas, la segunda capa oculta posee 5 neuronas y finalmente existen 5 salidas, una para cada seña que se busca reconocer y clasificar.

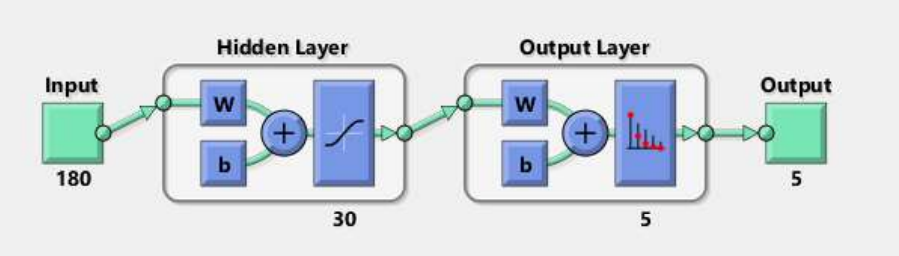


Figura 2.2. Arquitectura de la Red Neuronal.
(Fuente: Propia)

Los 180 datos de entrada están distribuidos de la siguiente manera: 90 correspondientes a las coordenadas xyz, de la mano derecha; mientras que, los otros 90 restantes son los correspondientes a las coordenadas xyz, de la mano izquierda. Las salidas que se establecieron para el sistema se muestran en la Tabla 2.3. De esta manera se activará una única neurona de salida para cada seña.

Tabla 2.3. Salidas de la Red Neuronal.

Seña	Salidas				
Buenos días	1	0	0	0	0
Buenas noches	0	1	0	0	0
Hola	0	0	1	0	0
Entregar/Tomar	0	0	0	1	0
¿Cómo estás?	0	0	0	0	1

(Fuente: Propia)

La función de activación es una variación de la sigmoide presentada en la Tabla 1.3. dada por la Ecuación 2.1.

$$g(a) = \frac{1 - e^{-2a}}{1 + e^{2a}} \qquad \text{Ec. 2.1}$$

El diseño y entrenamiento de la red neuronal se lo realizó utilizando el toolbox de Matlab "nprtool", el cual es específico para reconocimiento y clasificación de patrones.

Adicionalmente se han probado también otras configuraciones que han presentado un menor desempeño a la propuesta como se representa en la Tabla 2.4. En un inicio se tomó una red neuronal con 30 entradas correspondientes a los datos x de la mano derecha y una sola neurona de salida con 5 dígitos que fue incapaz de clasificar las señas. Posteriormente se utilizó una estructura con 90 entradas correspondientes a los datos xyz, y una salida de 5 neuronas según la tabla 2.3, esta red neuronal era capaz de reconocer y clasificar las señas; sin embargo, tenía un gran margen de error por lo que se decidió aumentar las neuronas de las capaz intermedias. La tercera red es similar a la anterior únicamente se aumentó el número de neuronas en su capa oculta lo que mejoró el desempeño de la red, el problema que presento es que al tomar los datos de la mano derecha, tuvo problemas para discriminar señas parecidas. Por esta razón se aumentó el número de entradas a 180 para formar la red de la estructura de la Figura 2.2, con esto se solucionaron todos los problemas de las anteriores estructuras.

Tabla 2.4. Otras Configuraciones probadas.

Arquitectura				**Resultado**
Input	Hidden layer	Output layer	Output	
30	10	10	1	Una neurona a la salida [1 1 1 1 1] presentó muchos problemas y no logró discriminar las señas.
90	20	5	5	Presento una mejora significativa sin embargo en simulación tenía una precisión del 80 %
90	30	5	5	Mejoró el tiempo de entrenamiento y logró una precisión del 98% en simulación; sin embargo, en la práctica al tomar solo los datos de la mano derecha confundía señas similares.

(Fuente: Propia)

No existe una teoría para determinar el número más óptimo de neuronas en sus capas ocultas; sin embargo, prácticamente se concluye que existe un punto en donde la red converge y añadir un número extra de neuronas en la capa oculta no mejora

significativamente la solución. Uno de los criterios aceptables para determinar si la arquitectura de la red es adecuada consiste en ver cómo responde al entrenamiento.

2.2.4. Entrenamiento de la Red Neuronal

El entrenamiento de la red neuronal se lo realizó con un total de 6 individuos. Cada persona realizó 9 veces cada seña frente al sensor, lo cual dio un total de 300 datos.

Los datos de entrada se conforman de la siguiente manera: el 70% se lo utilizó para el entrenamiento, el 15% para validación y el 15% restante para pruebas; es decir, se entrenó con 188 datos, se validó con 41 datos y se testeo con 41 datos (Ver Figura 2.3.). Estos datos se encuentran disponibles en el respaldo digital de este documento dentro de la carpeta “Traindata_1”.

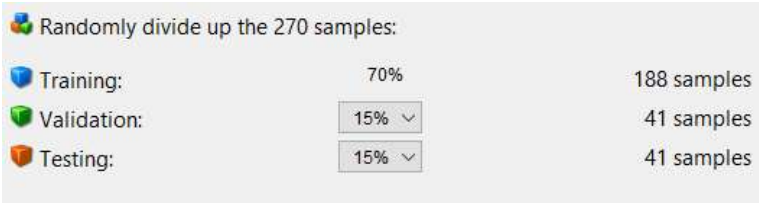


Figura 2.3. Número de Datos de Entrenamiento, Validación y Test.
(Fuente: Propia)

El entrenamiento se lo realizó utilizando el algoritmo de retro propagación. El mejor desempeño para la primera persona se logró luego de 26 iteraciones en donde el gradiente Δw_{kj} disminuyo hasta 0.00018095 como se muestra en la Figura 2.4. Es importante mencionar que el preprocesamiento de los datos logró disminuir el tiempo de entrenamiento de 1 minuto 45 segundos a 2 segundos.

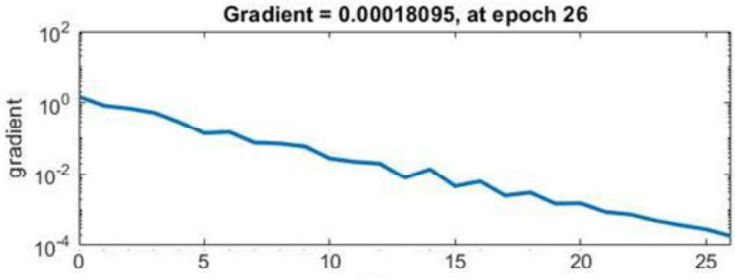


Figura 2.4. Descenso del Gradiente del Error)
(Fuente: Propia)

El mejor desempeño de la red neuronal se obtuvo en la iteración número 20, en donde el error cuadrático medio entre los datos de validación (verde) y los datos de prueba (rojos). se reduce a 0.0053684, como se visualiza en la Figura 2.5.

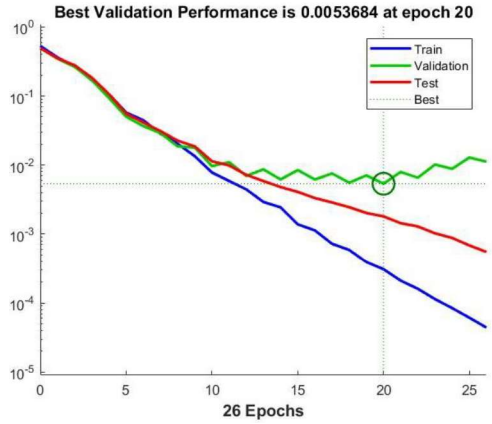


Figura 2.5. Diagrama de error cuadrático medio.
(Fuente: Propia)

Finalmente el desempeño de la red neuronal en las fases de entrenamiento, validación y prueba se representan a través de la matrices de confusión de la Figura 2.6. En donde los datos en espacios rojos representan la veces en que la red neuronal tuvo errores; mientras que, los espacios en verde representan las veces que la red neuronal clasificó correctamente.

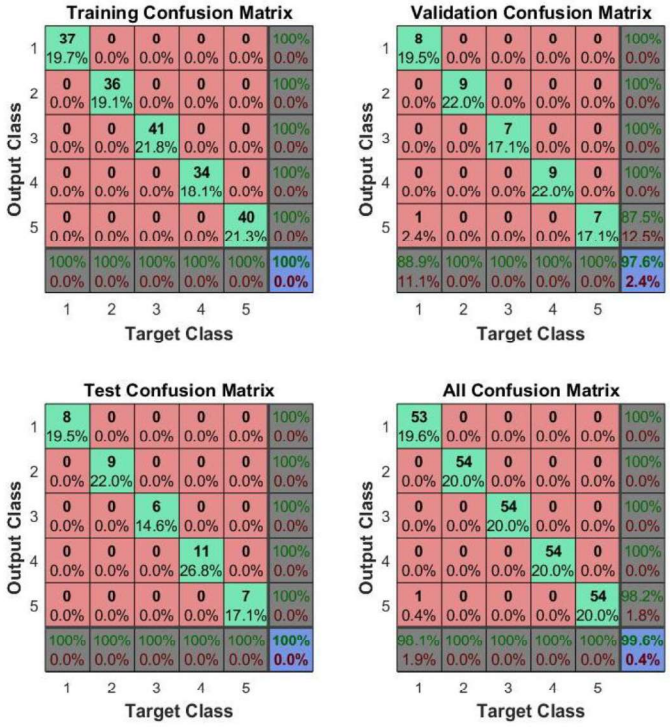


Figura 2.6. Matrices de Confusión.
(Fuente: Propia)

Analizando los diagramas de la Figura 2.6, se concluye que la red neuronal tiene una efectividad en reconocimiento y clasificación de la lengua de señas del 99.6 %, valor correspondiente a los 6 individuos para los que fue entrenado. El 0.4% de efectividad se perdió en la fase de validación en donde confundió un “Buenos días” con la seña “¿Cómo estás?”, esto corresponde a 1 error en 270 pruebas por lo que se valida tanto la arquitectura como el desempeño de la red.

Como resultado del entrenamiento el sistema genera una función de Matlab con extensión .m en donde se encuentran los pesos y umbrales de cada capa así como también la matriz de offsets y ganancias correspondientes al preprocesamiento. Estos datos se exportan desde Matlab a un archivo extensión .txt con el nombre de “Pesos_Umbrales2” que deberá ser copiado y pegado en la carpeta principal del HMI desarrollado en Visual. Una vez simulada la red Neuronal es necesario verificar su funcionamiento en pruebas de campo para lo cual a continuación se explicara el diseño del HMI y el funcionamiento del sistema.

2.2.5. Diseño del HMI

El HMI desarrollado se lo puede apreciar en la Figura 2.7. Las partes que conforman el mismo y su función se especifican en la Tabla 2.5. El HMI se desarrolló en Visual Studio que es una plataforma que soporta múltiples lenguajes de programación como: C++, C#, F#, Phyton, Ruby, Java y Php. De todas estos lenguajes C# es un lenguaje de programación orientado a objetos que es compatible con las librerías y controladores del Kinect. Por esta razón el HMI se desarrolló utilizando este lenguaje.

Figura 2.7. HMI del Sistema.
(Fuente: Propia)

Tabla 2.5. Partes del HMI.

No.	Nombre	Descripción
1	Barra de Título	Muestra el Título de la aplicación
2	Imagen RGB	Muestra la imagen obtenida por el sensor RGB
3	Barra de Selección de Modo	Permite seleccionar entre los modos Train y Test
4	Botones Start Stop	Inicia o termina la lectura de datos
5	Botón ANN	Permite cargar los pesos y umbrales al sistema
6	Botón Ayuda	Despliega la ayuda del sistema
7	Indicador On y Off	Indican cuando se debe iniciar la seña
8	Imagen Esqueleto	Muestra la representación 2D del Esqueleto de la persona
9	Barra de Datos	Muestra las coordenadas de las Manos en tiempo real
10	Barra de Estado	Ayuda a la persona a ubicarse dentro de la región de visión del Kinect
11	Barra de Resultados	Se compone por "Salida ANN" : la cual muestra la salida numérica de la red neuronal y por "Resultado" en donde se muestra la palabra resultante.

(Fuente: Propia)

Adicionalmente Adicional existen dos indicadores el primero denominado "Cantidad de datos" el cual muestra cuantos datos correspondientes al esqueleto se han tomado. La caja de texto llamada "Nombre de la Seña" sirve para especificar la seña que se está realizando en el modo Train. Otra ventana que aparece en el HMI es la ayuda al usuario, la misma que se desplegará al presionar el botón Ayuda. Esta ventana contiene una breve información explicando el funcionamiento del sistema, como se muestra en la Figura 2.8.

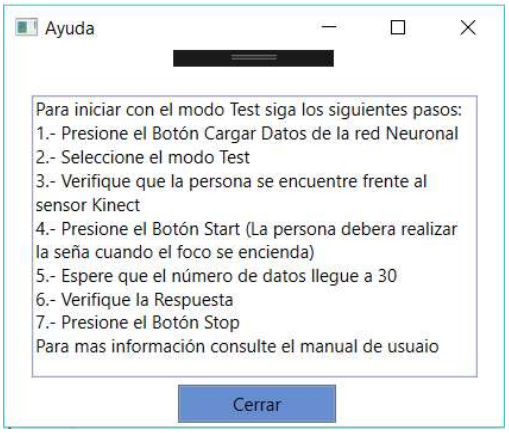


Figura 2.8. Menú de Ayuda.
(Fuente: Propia)

2.2.6. Funcionamiento

El sistema tiene dos modos de uso el modo Train y el modo Test. El modo Train se utiliza para capturar los datos del esqueleto, dichos datos son tomados para entrenar la red neuronal en el software Matlab. Como resultado del entrenamiento los umbrales y pesos

son exportados al sistema en un archivo .txt, para que posteriormente el usuario pueda realizar el reconocimiento y clasificación utilizando el modo Test.

Al abrir la aplicación el sistema automáticamente habilita la Imagen RGB extraída del Kinect y empieza a buscar si existe una persona. En caso de que una persona se encuentre frente al sensor, se mostrará la imagen del esqueleto, además, en la barra de datos se visualizarán las coordenadas correspondientes a sus manos. En este procedimiento pueden haber dos errores comunes. El primero ocurre cuando el Kinect no se encuentre conectado a la computadora en cuyo caso se mostrará un mensaje de error; mientras que, si no hay una persona en el rango visible del Kinect en la barra de Datos se mostrará: “No hay datos del esqueleto”. Este procedimiento se muestra en el diagrama de flujo de la Figura 2.9.

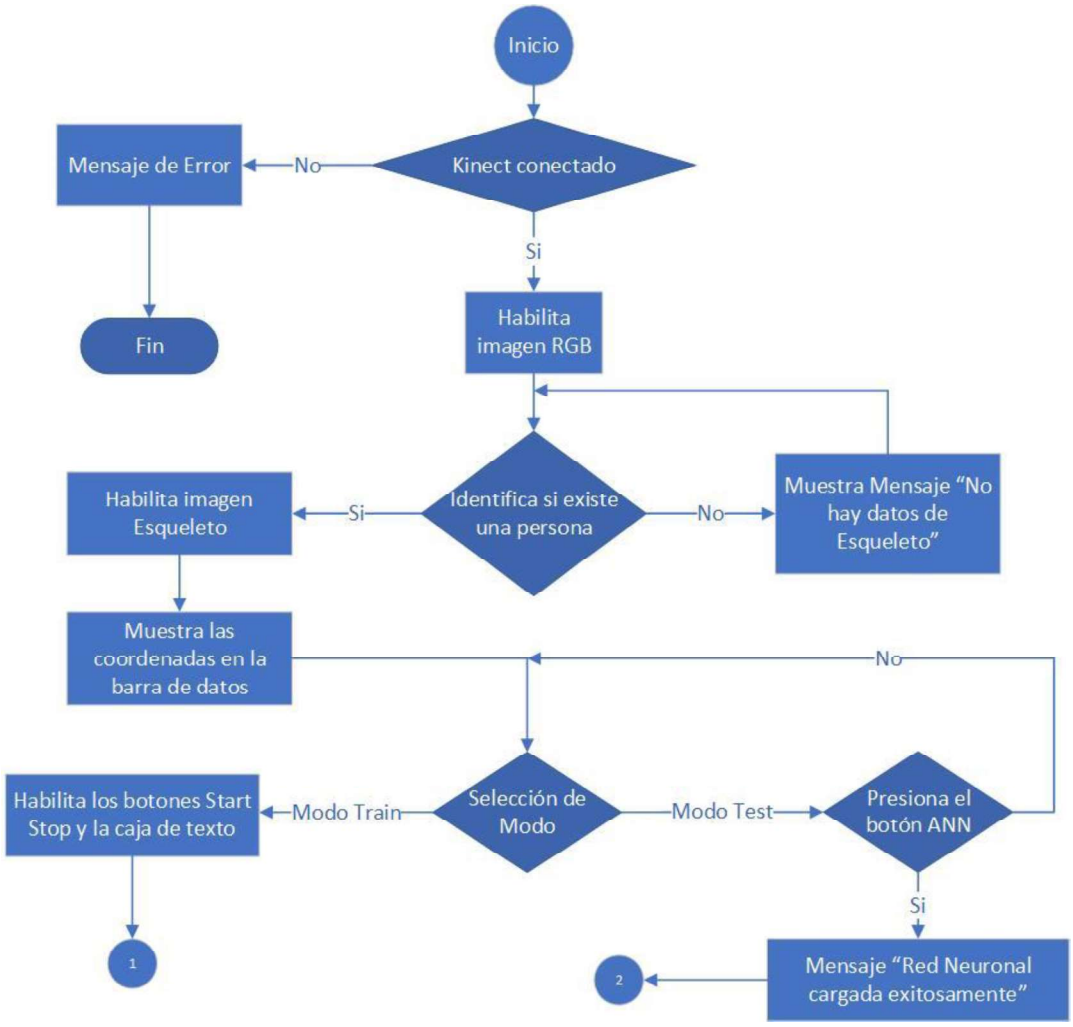


Figura 2.9. Diagrama de Flujo Inicial.
(Fuente: Propia)

En un inicio los botones “Start”, “Stop”, así como la pestaña de “Modo Test”, y la caja de texto se encontraran deshabilitadas. El modo test se habilitará únicamente si se ingresa los datos de la ANN presionando el botón ANN, este proceso desplegará el siguiente mensaje: “La red neuronal ha sido correctamente cargada”. Los botones “Start” y “Stop” se habilitarán después de escoger un modo de uso. La caja de texto se habilitará únicamente en el modo Train. Para poner en marcha el sistema se necesitan dos personas: la primera maneja el HMI y la segunda realiza la seña frente al sensor. Estas personas tomarán el nombre de Operario y Usuario, respectivamente.

Modo Train

En este modo se toman los datos provenientes de las coordenadas de los puntos de interés y se los exporta a un archivo formato txt. Una vez seleccionado el modo Train de la barra de modos, se habilitarán los botones “Start” y “Stop”, además de la caja de texto. Para tomar los datos en este modo de uso se deben realizar los siguientes pasos:

1. El operario debe escribir en la caja de texto el nombre de la seña.
2. El operario presiona el botón “Start”, para que el indicador (parte número 7 de la Tabla 2.5) se encienda y se apague; además, el sistema empezará a tomar los datos del esqueleto.
3. Cuando el indicador se encienda el usuario deberá realizar la seña.
4. Una vez que la cantidad de datos llegue a 30 el Operario deberá presionar la tecla “Stop”.

Al presionar la tecla stop el programa exporta la serie de datos leídos por el sistema y lo guarda dentro de la carpeta del programa, en la dirección “bin/Debug” con el nombre de “Datos_1” el numero irá aumentando conforme más datos se vayan exportando. El archivo generado tiene como encabezado lo que el usuario haya colocado en la caja de texto. Para exportar más datos se deberá repetir el procedimiento mencionado anteriormente. Si se cambia de seña se deberá cambiar la referencia de la caja de texto. Es recomendable que el usuario se encuentre dentro del rango de visión del Kinect, para esto el HMI presenta un mensaje en la barra de estado diciendo “Correctamente Ubicado”, si no está bien ubicado se presentara un mensaje indicando que se mueva más a la derecha, izquierda, abajo, según sea el caso. . Este procedimiento se representa de mejor manera en la Figura 2.10.

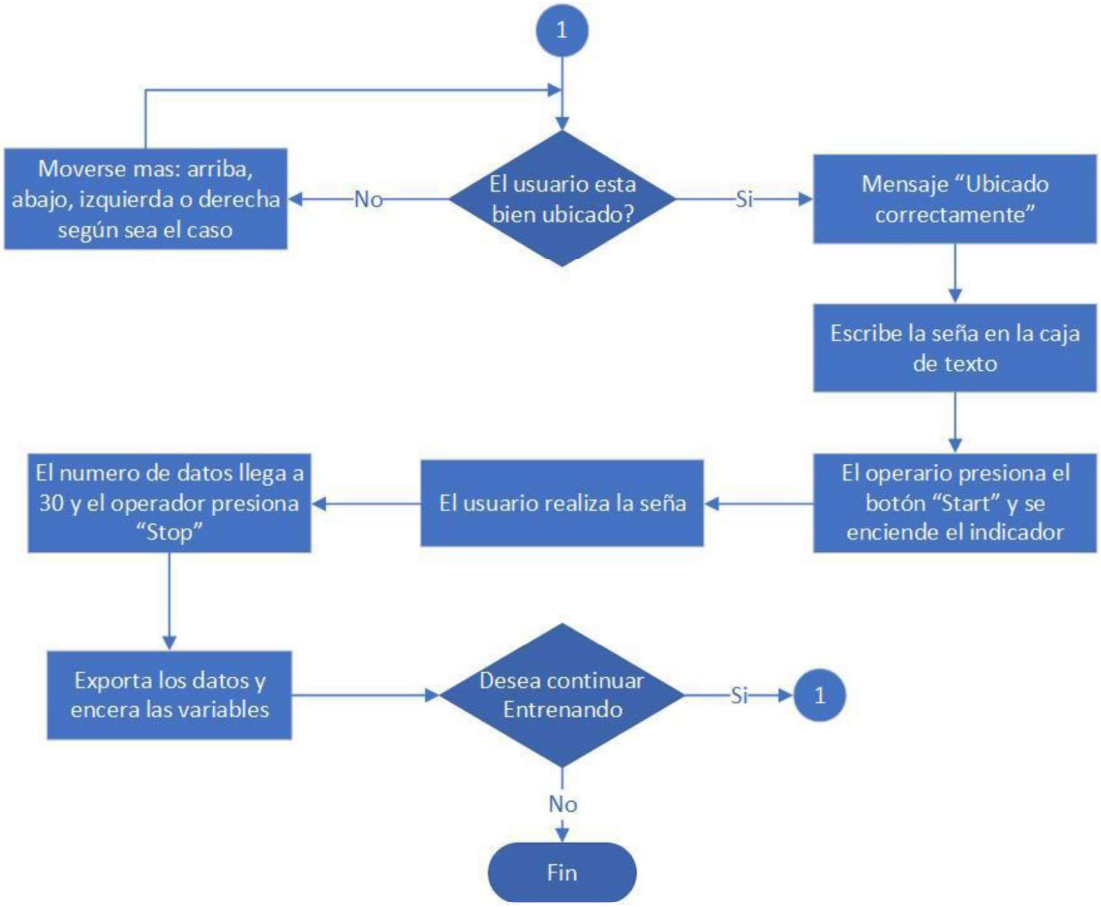


Figura 2.10. Diagrama de Flujo modo Train.
(Fuente: Propia)

Modo Test

En este modo se probará el funcionamiento del sistema, para que se habilite este modo es necesario que se carguen los datos de la red neuronal mediante el botón ANN. Una vez cargado los datos de la red neuronal se selecciona el modo Test en la barra de modos. Al seleccionar este modo se habilitará los botones "Start" y "Stop", para poner a prueba el sistema se deben seguir los siguientes pasos.

1. El operario debe cargar la red neuronal con la tecla ANN
2. El operario debe seleccionar el modo Test en la barra de modo
3. El operario presiona la tecla "Start". Con esto se prende y apaga el indicador mencionado anteriormente.
4. El usuario realiza la seña.

5. Después que el contador llegue a 30 el sistema muestra la salida numérica de la ANN, así como también el resultado.
6. El operario presiona la tecla “Stop”, para reiniciar el sistema.

Cuando la aplicación se cierra todos los datos se borran; de manera que, si se requiere ver algún resultado o verificar algún error los datos se encuentran disponibles en la carpeta “bin/Debug” del sistema, estos datos se guardan al presionar el botón “Stop”. El archivo generado guarda los datos del esqueleto recopilados así como también el resultado del reconocimiento y clasificación en un archivo extensión .txt, con el nombre de “Prueba_1”. El número también ira aumentando a medida que se aumenten las pruebas. Este procedimiento se puede ver en la Figura 2.11.

Figura 2.11. Diagrama de flujo modo Test.
(Fuente: Propia)

Cálculo de Salidas

La salida se calcula a partir de la Ecuación 1.4 y se muestran en la barra de resultados. Estos resultados se dividen en dos: en la parte superior se muestra la salida matemática de la red Neuronal; mientras que, en la parte inferior se muestra la palabra de la seña a la que corresponde. En la fase de diseño de la red neuronal se establecieron las salidas según la Tabla 2.3. La salida esperada para cualquier seña es de 1; sin embargo, se tomarán como válidas salidas mayores a 0.5. Por ejemplo, si el sistema da como resultado los siguientes valores 0.70, 0.30, 0, 0, 0, significa que hay un cierto grado de incertidumbre entre “Buenos días” (0.70) y “Buenas noches” (0.30). En este caso la respuesta del sistema será “Buenos días” por tener un valor mayor a 0.5.

Es importante tomar en cuenta cierto grado de incertidumbre ya que la idea es probar el sistema con personas que no fueron parte del entrenamiento. Como se explicó en el primer capítulo todas las personas son diferentes; además que, realizan la seña de manera parecida, no igual.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se especifica como se entrenó, probó y validó el sistema en las pruebas de campo. Además, se considera la instalación y desempeño del hardware así como también el manejo del software. Posteriormente se muestra el desempeño del sistema entrenado y probado con la ayuda de 12 personas; finalmente, se muestran los resultados de una encuesta realizada a los usuarios del sistema.

3.1. Hardware

Los elementos físicos de los que más depende el sistema son el Kinect y la computadora. El Kinect no tiene problemas para identificar el exoesqueleto siempre y cuando la persona se encuentre dentro de los rangos de distancia recomendados. En ocasiones, cuando algún punto de los mostrados en la Figura 1.16. no se encuentra visible, el SKD calcula su posible posición basado en la posición del resto de puntos. Esto repercute en que puede haber algo de ruido en el censado del esqueleto; sin embargo, no es muy considerable y no afecta a los resultados.

En la prueba de campo se identificó que la mejor posición para la persona es a 1.8 metros del sensor puesto que a esta distancia el Kinect puede leer la totalidad de puntos del esqueleto. Si bien es cierto en este sistema solo se utilizan los puntos correspondientes a la mano izquierda y mano derecha, si el Kinect lee todos los puntos presenta un menor ruido en el sensado. Si la ubicación de la persona es la apropiada el sistema presenta el siguiente mensaje en la barra de estado “Ubicado correctamente”, caso contrario se presentarán mensajes como muévase más a la derecha, izquierda, según corresponda.

Las pruebas de campo se realizaron de dos maneras: en la primera como pantalla se utilizó una pantalla led de 50 pulgadas conectada a la computadora en donde el usuario podía interactuar con la aplicación; además, también se lo hizo utilizando únicamente la computadora para que se visualicen los datos. En ambos casos el sistema funciono apropiadamente; sin embargo, el utilizar una pantalla led o un proyector resulta más cómodo para el usuario.

Finalmente en cuanto al hardware se refiere, la computadora no presenta problemas para ejecutar el sistema, puesto que no existe interrupciones al ejecutar todas las órdenes y calcular la salida del algoritmo de inteligencia artificial.

3.2. Software

La principal ventaja del software es que se ha desarrollado en una plataforma que puede ser fácilmente sujeta a cambios ya que el código fuente se programó desde cero. El tiempo de respuesta para realizar los cálculos es imperceptible puesto que los realiza de manera inmediata. Una de sus desventajas es que no guarda datos cuando la aplicación se cierra, por esto al finalizar una prueba (modo Test) se genera un archivo .txt que guarda la información de las coordenadas y resultado.

Adicionalmente se ha configurado el sistema para que cualquier persona con la ayuda del manual puede utilizar el modo Test; sin embargo, para añadir más señas deberá tener un conocimiento previo referente a redes neuronales y uso de Matlab.

3.3. Pruebas de Campo

Para esta sección se han tomado 12 personas las cuales mediante una encuesta han validado ciertos aspectos del sistema como: el funcionamiento, tiempo de respuesta, facilidad de uso, si las instrucciones han sido claras, el menú de ayuda ha sido útil, si la interfaz ha sido amigable para el usuario, entre otros.

Las 12 personas fueron distribuidas de la siguiente manera: 6 fueron parte del entrenamiento; mientras que, el sistema se probó con otras 6 personas. Es importante resaltar que del grupo de personas de entrenamiento y test solo una tenía una instrucción formal en lengua de señas. Con el resto de personas se realizó un taller en el cual aprendieron a realizar las señas especificadas en la Tabla 1.12. Una de las principales causas de no contar con gente que utilice la lengua de señas LSEC como lengua madre fue la dificultad que representa comunicarse con este tipo de personas.

A continuación se detalla cómo se realizaron las pruebas de campo; además, como se tomaron los datos y calibraron los equipos.

3.3.1. Adquisición de Datos

La adquisición de datos se lo realizó con 6 personas, a cada una se le pidió que realice las señas 9 veces utilizando el modo Train como se aprecia en la Figura 3.1. La posición de la persona en este caso es la adecuada como se puede ver en la barra de estado.

Siguiendo el procedimiento para modo Train, mencionado en la Sección 2.2.6, se obtuvieron 45 archivos por cada persona, lo que para las 6 personas suma un total de 270

archivos. Estos datos fueron cargados a Matlab con la ayuda del programa "Cargardatos2manos.m" disponible en los archivos adjuntos a este trabajo.

Figura 3.1. Adquisición de Datos.
(Fuente: Propia)

Para cada persona se generaron dos variables InputP1 y OutputP1. La variable de entrada de la red neuronal sería una matriz de 270x180 y se forma a partir de las entradas de cada una de las personas como se muestra en la Ecuación 3.1

$$Input = [InputP1, InputP2, InputP3, InputP4, InputP5, InputP6] \quad \text{Ec. 3.1}$$

Mientras que la salida de la red neuronal sería una matriz de 5x270, como se muestra en la Ecuación 3.2.

$$Output = [OutputP1, OutputP2, OutputP3, OutputP4, OutputP5, OutputP6] \quad \text{Ec. 3.2}$$

3.3.2. Manejo de Datos

El entrenamiento de la red se lo realizó mediante el "nprtool" de Matlab. Para acceder a este toolbox únicamente se digita "nprtool" en la ventana de comandos, esto desplegará la ventana mostrada en la Figura 3.2 en donde se deberán seleccionar las variables definidas mediante las Ecuaciones 3.1 y 3.2.

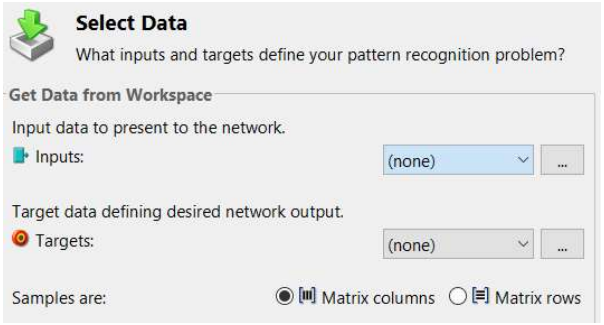


Figura 3.2. Ingreso de datos en nprtool.
(Fuente: Propia)

Como resultado del entrenamiento se generan un total de 5946 datos distribuidos como se muestra en la Tabla 3.1. Estos datos serán cargados al sistema presionando el botón ANN de la Figura 2.7.

Tabla 3.1. Pesos y Umbrales.

Variable	Símbolo	Dimensión
Offsets	Offset	1x180
Ganancia	Gain	1x180
Mínimo	ymin	1x1
Matriz de pesos capa 1	IW1_1	30X180
Matriz de pesos capa 2	LW2_1	5x30
Umbral 1	B1	1x30
Umbral 2	B2	1x5

(Fuente: Propia)

Las variables en las que se exportarán estos datos dentro de Visual Studio se especifican en la Figura 3.3. Si se desea cambiar la estructura de la red neuronal se deberán verificar las dimensiones de las variables así como también la distribución del archivo Pesos_Umbrales2.txt. Finalmente se realizaron pruebas como se especifica a continuación.

```
double[,] W11 = new double[30, 180];//Matriz de pesos [capa 1]
double[,] W12 = new double[5, 30];//Matriz de pesos [capa 2]
double[] Offset = new double[180];//Ofsets
double[] gain = new double[180];//Ganancias
double ymin;//ajuste
double[] U1 = new double[30];//Matriz de Umbrales [capa 1]
double[] U2 = new double[5];//Umbral [capa 2]
```

Figura 3.3. Variables Ingresadas en Visual Studio.
(Fuente: Propia)

3.3.3. Desempeño del Sistema

Para evaluar el desempeño del sistema a cada persona se le pidió que realice varias señas (entre 15 y 20). La Figura 3.4. muestra a la primera persona realizando las señas frente al sensor, todas las pruebas han sido grabadas en video y se encuentras disponibles como anexos.

La Figura 3.5 muestra los resultados obtenidos para la primera persona que realizó un total de 15 señas. En azul se representan las veces que el sistema respondió de manera correcta; mientras que, en tomate las veces que el sistema tuvo errores. En este caso para la primera persona no se registró errores, hay que considerar que esta persona fue parte del grupo de entrenamiento, por lo que siguiendo los datos obtenidos por la simulación se esperaba una precisión del 99.6%; sin embargo, en este caso el sistema obtuvo una precisión del 100%. Con esto se puede decir que el sistema es capaz de reconocer y clasificar la lengua de señas con personas para las que ha sido entrenada.

Figura 3.4. Primera persona probando el sistema en modo Test.
(Fuente: Propia)

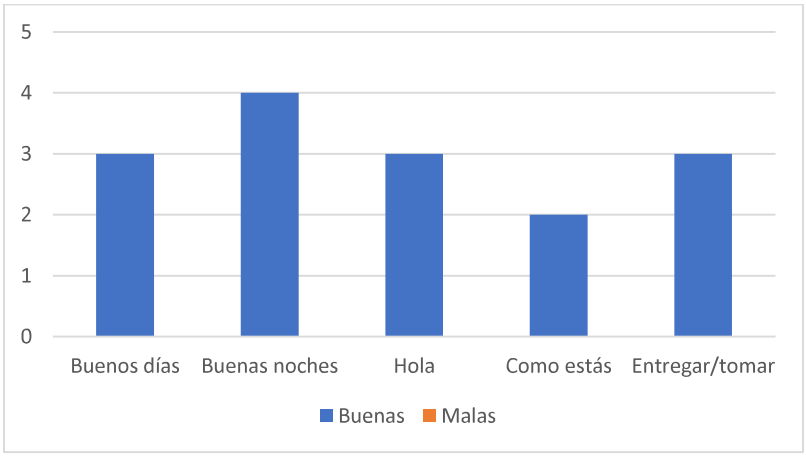


Figura 3.5. Resultados Persona 1.
(Fuente: Propia)

En total se realizaron 227 pruebas con 12 personas distintas, 3 de ellas fueron parte del entrenamiento; mientras que otras 9 no. La Figura 3.6. y 3.7. muestran el número de señas que realizó cada persona. Como se puede notar en las Figuras 3.6. y 3.7., el número de señas que realiza cada persona es distinto, esto es debido a que, las personas que tuvieron mayor afinidad con las señas realizaron pocas; mientras que, las personas que tuvieron cierta dificultad con alguna seña se les pidio que la realicen nuevamente.

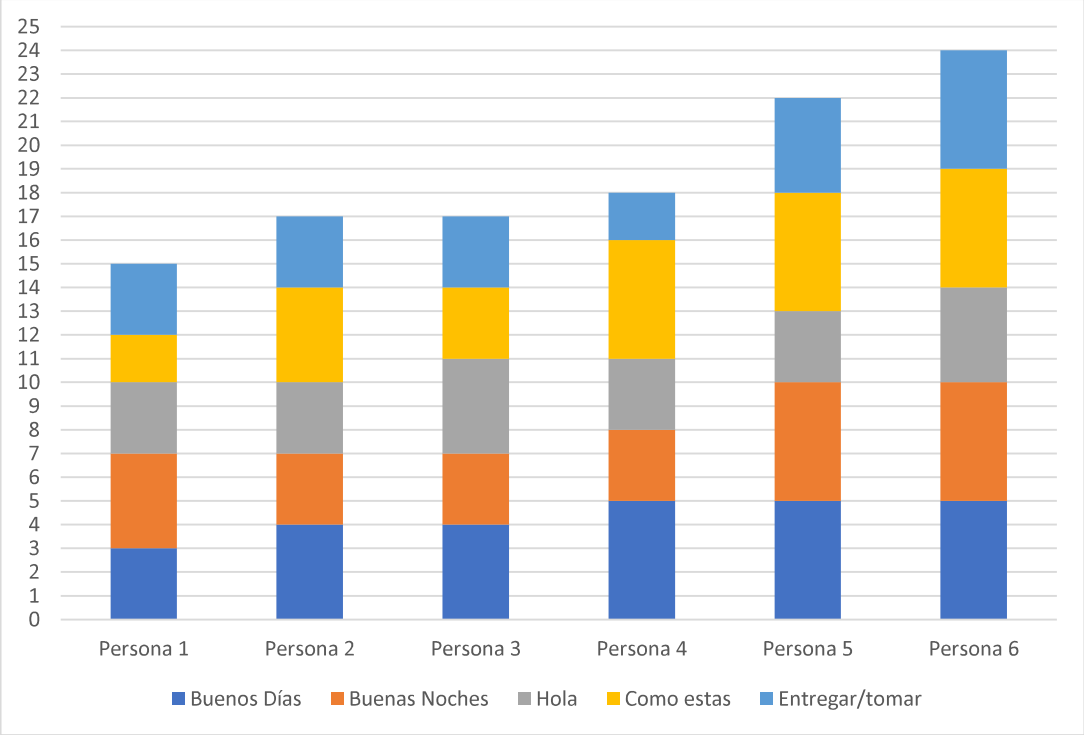


Figura 3.6. Distribución de señas según personas (1-6).
(Fuente: Propia)

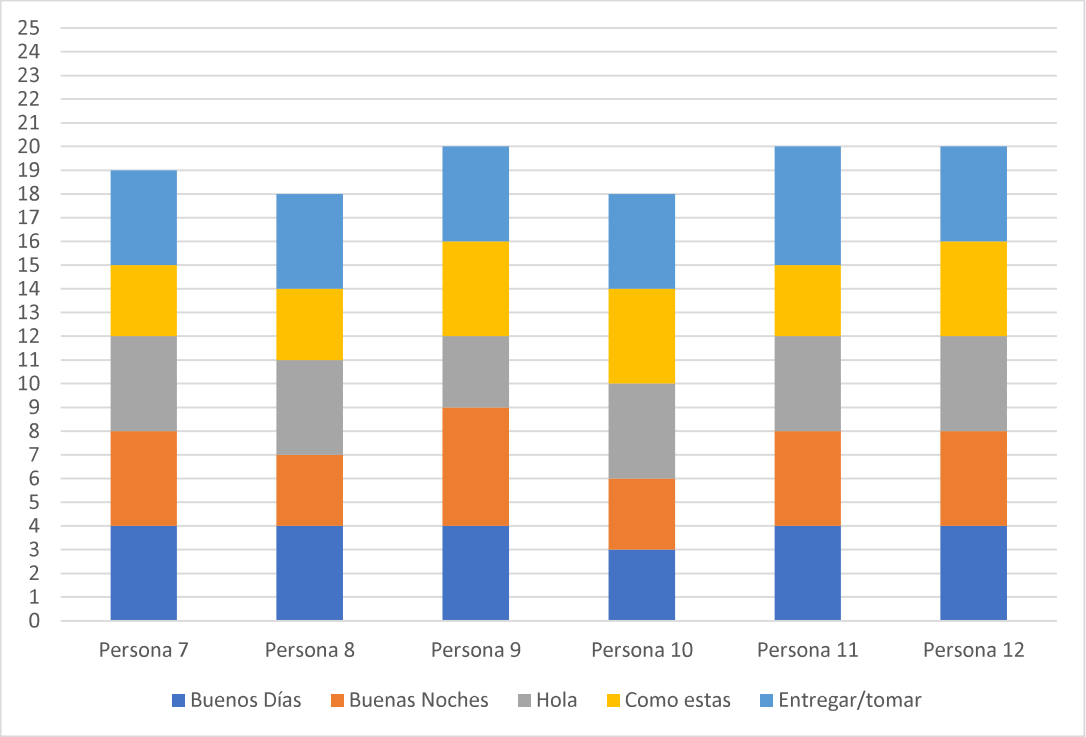


Figura 3.7. Distribución de señas según personas (7-12).
(Fuente: Propia)

La Figura 3.8. muestra el total de señas utilizadas para la validación del sistema. Del total de las pruebas hubieron 3 errores. El primer error ocurrió con la persona número 4, cuando el sistema calculo como salida la seña "¿Cómo estás?" en lugar de "¿Buenos días?". El segundo error se originó con la misma persona cuando el sistema no dio ninguna respuesta cuando el usuario efectuó la seña "Buenos días". Con la persona número 10 el sistema dio como salida la seña "¿Cómo estás?" en lugar de "¿Buenos días?".

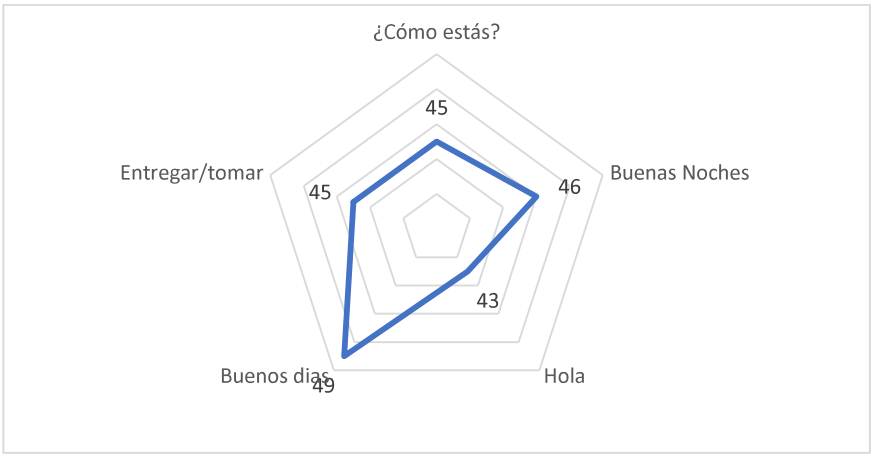


Figura 3.8. Número total de señas utilizadas para el test.
(Fuente: Propia)

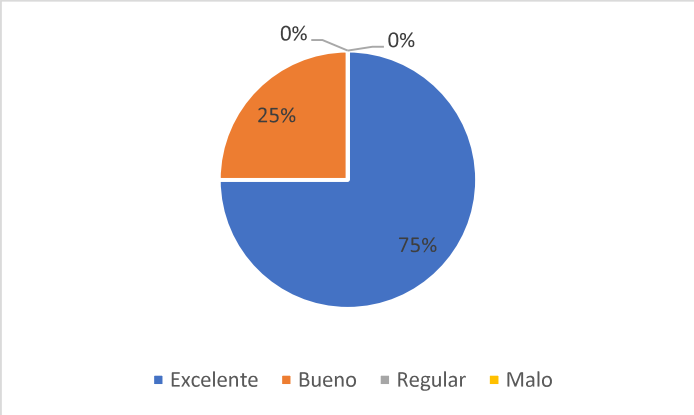
De las 227 pruebas 3 veces el sistema no dio una respuesta adecuada, por lo que el error en reconocimiento y clasificación es de 1.32%, como se muestra en la Figura 3.9. Este error es generado por dos motivos: el error humano al realizar la seña de una forma inadecuada y el segundo corresponde al error en simulación de la red (0.4%). Finalmente se puede notar que el individuo número 4 presenta dos errores, lo cual basado en los resultados de las demás personas es poco usual. Esto significa que la persona no realiza adecuadamente la seña o el sistema no está familiarizado en la forma de realizar la seña de esta persona, en cualquiera de los dos casos se puede optar por reentrenar al sistema con los datos de esta persona para asegurar un mayor desempeño.

Figura 3.9. Diagrama de Aciertos vs Errores del sistema.
(Fuente: Propia)

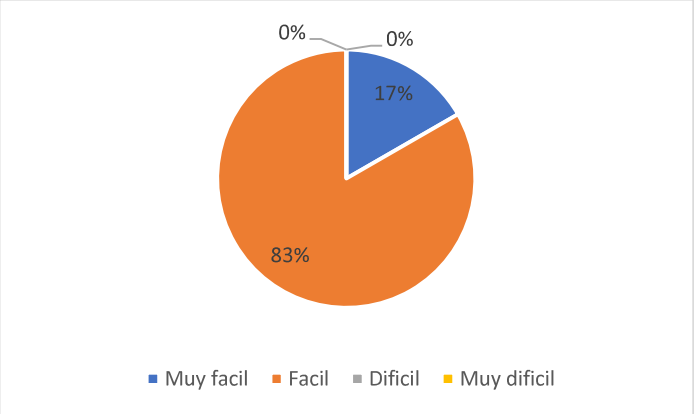
3.3.4. Validación del HMI del sistema

A las 12 personas que fueron parte fueron parte del entrenamiento y test se les realizó una encuesta con las preguntas que se muestran a continuación. En la parte inferior de cada pregunta se muestra las respuestas tabuladas.

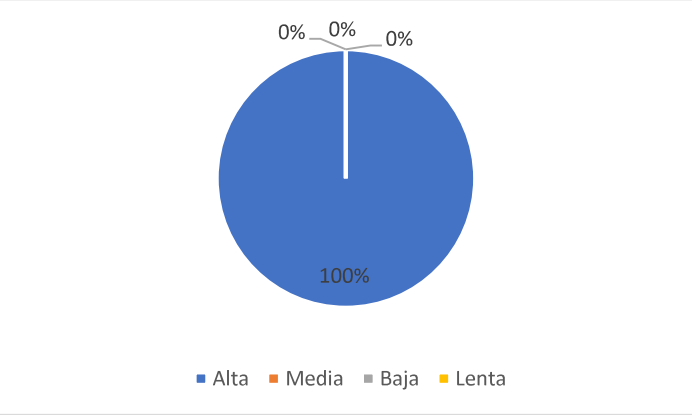
1. Según su criterio en el test ¿Qué tan amigable es la interfaz HMI utilizada para el reconocimiento de gestos?.



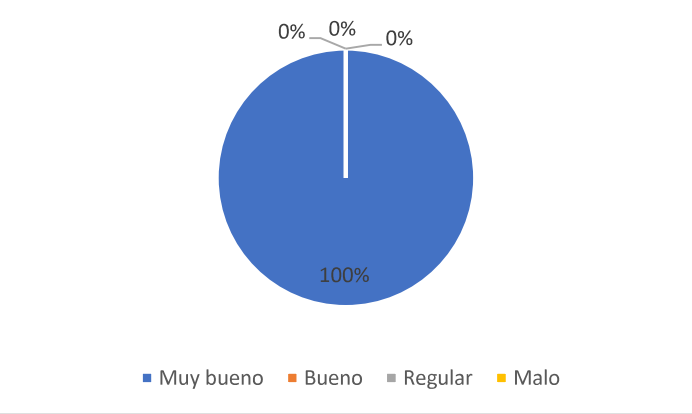
2. Según su criterio ¿Qué tan fácil es utilizar el sistema?.



3. Según su criterio ¿Qué tan rápida es la respuesta del sistema?.



4. Según su criterio ¿Cómo califica al sistema?.



Las preguntas realizadas intentan medir la satisfacción del usuario frente al HMI. Tomando en cuenta la mayoría de usuarios se puede decir que el sistema resulta fácil de usar, el tiempo de respuesta es alto. Finalmente en términos generales todos califican al sistema como muy bueno.

Adicional a las preguntas se adicionó un campo para comentarios en donde se dieron ciertas recomendaciones referentes al HMI. Acogiendo las sugerencias de los usuarios se modificado el HMI para mejorar la interfaz.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

Luego de analizar la lengua de señas se concluye que para realizar una seña principalmente se utiliza el movimiento de las manos, dichos movimientos pueden ser representados mediante trayectorias, siendo esta justamente la diferencia más notoria entre seña y seña.

En pruebas de campo se logró detectar con éxito con un 100% de efectividad las coordenadas del esqueleto principalmente de las manos, en diferentes entornos y condiciones. Por esta razón el Kinect y el SDK V1.8 tienen un buen desempeño para esta aplicación puesto que es robusto frente a interferencias del medio ambiente como luz y objetos.

El algoritmo de inteligencia artificial que mejor se adapta al reconocimiento y clasificación de patrones son las redes neuronales artificiales, puesto que los algoritmos genéticos tienen una mayor aplicación en optimización de parámetros y la lógica difusa presenta cierta incertidumbre en el momento de seleccionar reglas difusas.

La mejor arquitectura de red neuronal fue la mostrada en la Figura 2.2 con: 180 neuronas en la capa de entrada, 30 neuronas en la primera capa oculta, 5 neuronas en la segunda capa oculta y 5 neuronas en la capa de salida. Está arquitectura alcanzó una precisión en la fase de entrenamiento del 99.6%.

El algoritmo de retro propagación tuvo un buen desempeño en esta aplicación; de manera que, al entrenarlo con 270 datos, converge de forma rápida en 26 iteraciones logrando un error cuadrático medio de 0.0053684 (ver Figura 2.5) y en un tiempo no superior a los 10 segundos.

Tomando en cuenta el criterio de los usuarios se concluye que el HMI tiene una interfaz amigable, su velocidad de respuesta es adecuada y en términos generales el sistema cumple de manera eficiente con los objetivos planteados.

En pruebas de campo se determinó que el sistema es capaz de reconocer y clasificar la lengua de señas con un 98.7% de efectividad para personas que no fueron parte del entrenamiento.

Mediante el sistema implementado basado en Inteligencia Artifiicial y Kinect se logró reconocer y clasificar 5 señas representativas. El HMI facilita el entrenamiento y prueba del sistema; además, permite visualizar los resultados. Por lo cual se concluye que se cumplen con los objetivos de la tesis.

4.2. Recomendaciones

Se recomienda verificar que la posición de las personas sea la adecuada antes de utilizar la aplicación para que el Kinect no tenga inconvenientes al leer los datos del esqueleto del usuario.

Es importante poner en la caja de texto alguna referencia de la persona o de la prueba en el modo Test ya que este mensaje aparecerá en el encabezado del archivo .txt generado y servirá para poder identificar el archivo posteriormente.

Para complementar el estudio con un número mayor de señas, se recomienda que la persona tenga un conocimiento previo de: manejo de redes neuronales en Matlab, así como de programación en C#.

Este sistema permite determinar la seña realizada por una persona con discapacidad auditiva; sin embargo, para cerrar el lazo de comunicación se recomienda implementar un avatar que realice señas en base a texto escrito. Con esto el individuo que no habla la lengua de señas podría escribir un mensaje y el avatar realizar las señas para la otra persona.

El Kinect a pesar de demostrar un excelente rendimiento para esta aplicación, se debe tener en cuenta que Microsoft ha descontinuado su producción, por lo que se recomienda el uso de otros sensores 3D, principalmente el ASUS Xtion Pro Live que según [15] funciona bien en distancias cortas.

En el proceso de recopilación de información se detectó que existen varios problemas con niños especiales los cuales son incapaces de entender emociones como: feliz, triste, enojado, etc. Se recomienda utilizar como base este trabajo para desarrollar un sistema de clasificación de emociones basado en reconocimiento facial, este sistema ayudaría a dichos niños a reconocer y expresar emociones contribuyendo a su inclusión en la sociedad.

5. REFERENCIAS

[1] Organización Mundial de la Salud, «Sordera y pérdida de la Audición,» 2018 03 15. [En línea]. Available: https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss. [Último acceso: 2019 02 22].

[2] CONADIS, «Personas con Discapacidad registradas,» 02 05 2018. [En línea]. Available: https://www.consejodiscapacidades.gob.ec/wp-content/uploads/downloads/2018/03/index.html. [Último acceso: 26 06 2018].

[3] CONADIS, FENASEC, *Manual práctico para intérpretes en la Lengua de Señas Ecuatoriana,* Quito, 2014.

[4] R. H. Blake, Taxonomía de conceptos de la comunicación, México: Nuevomar, 2000.

[5] Helix, «Libro Blanco Sobre Discapacidad Auditiva,» 10 Julio 2017. [En línea]. Available: http://www.helixcv.com/wp-content/uploads/2017/07/LIBRO-BLANCO-SOBRE-DISCAPACIDAD-AUDITIVA-%C3%81mbitos-de-actuaci%C3%B3n.pdf. [Último acceso: 18 05 2018].

[6] Audiología didáctica para estudiantes, «Hipoacusia: concepto y etiologías,» 28 07 2017. [En línea]. Available: http://audiologiaacademica.blogspot.com/2014/09/hipoacusia-concepto-y-etiologias.html. [Último acceso: 28 05 2018].

[7] Wikipedia, «Lengua de Señas,» 23 05 2018. [En línea]. Available: https://upload.wikimedia.org/wikipedia/commons/4/4e/Main_Sign_Language_Families.png. [Último acceso: 30 05 2018].

[8] FENASEC, «Diccionario Gabriel Román,» [En línea]. Available: http://fenasec.ec/diccionario-lsec.html. [Último acceso: 2018 11 19].

[9] M. A. Rodríguez, Lenguaje de Signos, Alicante: Miguel de Cervantes, 2003.

[10] «Lideres Transformndo e Innovando,» 2014 septiembre 13. [En línea]. Available: https://litinn.com/2014/09/13/seis-gestos-que-toda-persona-de-exito-hace/. [Último acceso: 1 05 2018].

[11] J. Vilches, «La dactilología, ¿qué, cómo, cuándo?,» 2005. [En línea]. Available: http://www.uco.es/~fe1vivim/alfabeto_dactilologico.pdf. [Último acceso: 01 05 2018].

[12] CONADIS, «Diccionario Gabriel Román - Guía uso de Ilustraciones,» 2017. [En línea]. Available: http://plataformaconadis.gob.ec/diccionario/wp-content/uploads/2014/07/Guia-Uso-ilustraciones.pdf. [Último acceso: 18 05 2018].

[13] M. A. Medina, Interviewee, *Lengua de Señas: Caracteristicas, Señas Básicas y Aprendizaje.* [Entrevista]. 26 06 2018.

[14] R. Horaud, «INRIA Inventors for the Digital World,» 2016 05 26. [En línea]. Available: http://perception.inrialpes.fr/~Horaud/Courses/pdf/Horaud_3DS_1.pdf. [Último acceso: 28 06 2018].

[15] D. D. A. Z. Gerald Rauscher, «A Comparison of 3D Sensors for Wheeled Mobile Robots,» *Advances in Intelligent Systems and Computing,* vol. 302, nº 13, 2016.

[16] Microsoft, «Kinect for Windows Sensor Components and Specifications,» Developer Network, [En línea]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx. [Último acceso: 2018 05 24].

[17] D. L. Dihl, «The Kinect for Windows Sensor,» 2015. [En línea]. Available: http://docplayer.net/32007496-The-kinect-for-windows-sensor-dr-leandro-dihl.html. [Último acceso: 24 05 2018].

[18] P. P. K. G. C. M. Saeid Motiian, «Automated extraction and validation of children´s gait parameters with the Kinect,» *BioMedical Engineering ,* vol. 1, p. 37, 2015.

[19] J. O. M. M. Juan Salvatore, «Detección de Objetos Utilizando el sensor Kinect,» de *Excellence in Engineering To Enhance a Country´s Productivity*, Guayaquil, 2014.

[20] F. Córdova, Detección de Robo/Abandono de objetos en Interiores Utilizando Cámaras de Profundidad, Madrid, 2012.

[21] D. Navas, Dieseño e Implementación de un prototipo de detección y localización de obstáculos a través de reconstrucción tridimensional mediante el uso de una cámara estereoscópica, Quito, 2018.

[22] J. G. H. G. X. R. K. K. Andrea Fossati, Consumer Depth Cameras for Computer Vision, London: Springer, 2013.

[23] A. F. M. C. T. S. M. F. R. M. A. K. A. B. Jaime Shoton, «Real Time Human Pose Recognition in Parts from Single Depth Images,» *Computer Vision and Pattern Recognition (CVPR), IEEE,* p. 8, 2011.

[24] R. M. M. José Palma Méndez, Inteligencia Artificial, Madrid: Mc Graw Hill, 2008.

[25] H. Banda, Inteligencia Artificial Principios y Aplicaciones, Quito, 2014.

[26] A. M. Turing, «Computing Machine and Inteligence,» *Mind,* vol. LIX, pp. 433-460, 1950.

[27] J. Ponce, Inteligencia Artificial, Latin, 2014.

[28] P. P. Cruz, Inteligencia Artificial con aplicaciones a la ingeniería, Mexico: Alfaomega, 2010.

[29] C. González, «Escuela Superior de Informática,» 2011. [En línea]. Available: http://www.esi.uclm.es/www/cglez/downloads/docencia/2011_Softcomputing/LogicaDifusa.pdf. [Último acceso: 15 01 2018].

[30] Centro Nacional de Microelectrónica de Sevilla , «Fuzzy Logic Desing Tools,» 2012 12 12. [En línea]. Available: http://www2.imse-cnm.csic.es/Xfuzzy/FLEB/main/main22.htm. [Último acceso: 1 6 2018].

[31] D. R. J. R. J. D. A. P. Marcos Gestal, Inttroducción a los Algoritmos genéticos y la Programación Genética, Coruña: Consorcio Galego, 2010.

[32] A. Kuri, «Algoritmos Genéticos,» 05 2000. [En línea]. Available: http://cursos.itam.mx/akuri/PUBLICA.CNS/2000/AGS.PDF. [Último acceso: 17 09 2018].

[33] Castrovation , «Algoritmo Genético,» 2016 01 2016. [En línea]. Available: http://castrovation.blogspot.com/2016/01/algoritmo-genetico.html. [Último acceso: 2018 09 17].

[34] D. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Boston : Addison-Wesley Longman Publishing Co, Inc., 1989.

[35] C. Coello, «Introducción a los Algoritmos Genéticos,» *Tecnologías de Información y Estrategias de NEgocios ,* vol. 3, nº 17, pp. 5-11, 1995.

[36] J. Koza, Genetic Programing "On the Programing of Computers by Means of Natural Selection", Cambridge: The MIT Press., 1992.

[37] D. J. Matich, Redes Neuronales Conceptos Básicos y Aplicaciones, Rosario, 2001.

[38] X. B. Olabe, «Redes Neuronales Artificales y sus Aplicaciones,» 1998. [En línea]. Available: https://ocw.ehu.eus/file.php/102/redes_neuro/contenidos/pdf/libro-del-curso.pdf. [Último acceso: 2018 08 21].

[39] M. W. y. P. W., «A logical calculus of the ideas immanet in nervous activity,» *Bulletin of Mathematical Biophysics,* vol. 5, pp. 115-133, 1943.

[40] S. Richter, «Redes Neuronales Perceptron Multicapa,» webeletrónica, [En línea]. Available: http://serverpruebas.com.ar/news21/nota09.htm. [Último acceso: 2018 08 29].

[41] J. Kalita, «Simple Neural Nets for Pattern Classification,» University of Colorado Springs, [En línea]. Available: http://www.cs.uccs.edu/~jkalita/work/cs587/2014/03SimpleNets.pdf. [Último acceso: 2018 08 29].

[42] D. Kriesel, Neural Networks, Germany, 2005.

[43] M. B. Carlos Ruiz, Redes Neuronales: Conceptos Básicos y Aplicaciones, Argentina, 2001.

[44] Kohonen, Self Organizing Feature Maps, New York: Springer-Verlag, 1995.

[45] S. Grossberg, Studies of Mind and Brain, Boston, 1992.

[46] R. Hecht, «Counterpropagation Networks,» *Aplied Optics,* vol. 26, nº 23, pp. 4979-4984, 1997.

[47] M. d. B. Alfredo Sanz, Redes Neuronales y Sistemas Difusos, Zaragoza: Afaomega, 2001.

[48] Y. T. S Juang, «A comparison between the bakc propagation and counter propagation networks in the modeling of the TIG welding process,» *Science Direct ,* vol. 75, pp. 54-62, 1998.

[49] S. S. Cathy Wu, «Back Propagation and Counter propagation neural networks for phylogenetic classification of ribosomal RNA sequenses,» *Nucleic Acids Researchs,* vol. 22, nº 20, pp. 4291-4299, 1994.

[50] Q. X. A. J. Y. Song, «Comparison studies of five neural network based fault classifier for complex transmission lines,» *Electric Power Systems Research,* vol. 43, nº 2, pp. 125-132, 1997.

[51] T. J. Ross, Fuzzy Logic with Engineering Aplications, Alburquerque: Wiley, 2010.

[52] L. A. Zadeh, «Similarity Relations and Fuzzy Ordering,» *Information Sciences,* vol. 3, nº 2, pp. 177-200, 1981.

[53] J. Bezdek, «Patter Recognition with Fuzzy Objective Function Algorithms,» *Advanced Applications in Pattern Recognition. Springer,* pp. 43-93, 1991.

[54] K. N. N. Y. H. T. Hisao Ishibuchi, «Selecting Fuzzy If-Then Rules for Clasification Problems Using Genetic Algoritms,» *IEEE Transactions on Fuzzy Systems,* vol. 3, nº 3, pp. 260-270, 1995.

[55] L. D. David Montana, «Training Feed Forward Neural Networks Using Genetic Algorithms,» *Proceedings of the 11th International joint conference on Artificial Inteligence,* vol. 1, pp. 762-767, 1989.

[56] G. Colmenares, «Redes Neuronales,» 30 11 2004. [En línea]. Available: http://webdelprofesor.ula.ve/economia/gcolmen/programa/redes_neuronales/capitulo5_preprocesamiento_de_datos.pdf. [Último acceso: 18 12 2018].

[57] D. M. E. Charniak, Introduction to artificial intelligence, Massachusets: Addison Whesley , 1985.

6. ANEXOS (DISPONIBLES EN DIGITAL)

1. Entrenamiento red BP y LVQ
2. Encuesta realizada a los usuarios del sistema.
3. Video de las pruebas realizadas a las 12 personas
4. Video de la entrevista realizada a la experta en lengua de señas.
5. Datos de entrenamiento.
6. Umbrales y ófsets.
7. Archivos de Matlab:
    - Lectura de datos: “cargardatos2manos.m”
    - Función de la red Neuronal: “MyNeuralNetworkFunction.m”
    - Exportar datos: “ExportarDatos2manos.m”
8. Software “SLreconv1”.