ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

APLICACIÓN DE FACTORIZACIÓN DE TRAYECTORIA ESPACIAL EN RECONOCIMIENTO AUTOMATICO DE SEÑALES PARA USO EN SISTEMA DE CONTROL DE DISPOSITIVOS DOMÓTICOS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

DANIEL ALEJANDRO TACO GALLARDO
daniel.taco@epn.edu.ec

DIRECTOR: Msc. Henry Patricio Paz Arias
henry.paz@epn.edu.ec

Quito, 27 de octubre del 2021

DECLARACIÓN

Yo, Daniel Alejandro Taco Gallardo, 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 correspondientes de este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Daniel Alejandro Taco Gallardo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Daniel Alejandro Taco Gallardo, bajo mi supervisión.

HENRY PATRICIO PAZ ARIAS

Firmado digitalmente por HENRY PATRICIO PAZ ARIAS
Fecha: 2021.11.04 14:43:09 -05'00'

DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradezco a Dios y a mi madre, Silvia Taco, por su infinito amor y por permitirme tener esta experiencia llamada vida.

Quisiera agradecer a mi familia, Taco Gallardo, por ser mi apoyo, respaldo, ejemplo de lucha, amor y perseverancia. Igualmente, a Violeta, mis amigos y compañeros por haberme brindado a su lado, una de las mejores épocas de mi vida.

También quisiera agradecer a mi tutor de Proyecto de Titulación, MSc. Henry Paz, por ser más que un docente, un amigo. Asimismo, ser mi mentor en lo que verdaderamente me apasiona, la ciencia y la Inteligencia Artificial en acción para el mundo.

Sin más, Quisiera agradecer a la Escuela Politécnica Nacional por permitirme crecer en sus aulas y ser mi alma mater en mi historia. Finalmente, un agradecimiento especial a CEDIA por su amabilidad en la cooperación del desarrollo del presente proyecto.

DEDICATORIA

Al mundo y a todas las
personas que creyeron en mí,
como agente de cambio y
esperanza.

A Byron, y mis abuelos Olimpia,
Cesar y Herminia.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

RESUMEN

El presente proyecto muestra la aplicación de una novedosa técnica de Visión Artificial (VA), conocida como Factorización de la Trayectoria Espacial (TSF), la cual reconstruye la estructura 3D de objetos en video, a partir de una secuencia de imágenes (fotogramas) en 2D, obteniendo así, información significativa del objeto imposible de adquirir y representar en dos dimensiones. Además, de la idealización y ejecución de un sistema de control de dispositivos domóticos como caso de estudio, el cual traduce un conjunto definido de señas gestuales utilizadas en el American Sign Language (ASL), en instrucciones que manipulan dispositivos terminales inteligentes por medio de Amazon Alexa Smart Home Skill.

En el sistema propuesto las señas o señales son reconocidas automáticamente mediante la aplicación de técnicas de preprocesamiento de VA, asimismo con la recuperación de la forma 3D de las manos por medio de TSF. Posteriormente, se reduce su representación tridimensional, por medio de la técnica de reducción de dimensionalidad en aprendizajes no supervisados (tSNE), en una correcta entrada para la aplicación de un modelo de clasificación conocido como Red Neuronal Recurrente (RNN), con el fin de clasificarlas dentro de un conjunto instrucciones predefinidas. El modelo de clasificación escogido es el resultado de una comparación entre el modelo de máquinas de soporte vectorial (SVM) y RNN, siendo la ultima la de mejor exactitud.

La parte final del documento contiene los resultados de la implementación del sistema de control de dispositivos domóticos, por demás se presentan conclusiones y recomendaciones sobre la aplicación de TSF en el reconocimiento automático de señales.

Palabras clave: Visión artificial, Factorización de la trayectoria especial, Deep Learning, Redes neuronales recurrentes, Máquinas de soporte vectorial, Dataset, Entrenamiento, Validación, Modelo Cliente-Servidor, Smart Home, Amazon Alexa, Amazon Alexa Smart Home Skill.

ABSTRACT

This project presents the application of a novel Machine Vision (VA) technique, known as Spatial Trajectory Factorization (TSF), which reconstructs the 3D structure of objects in video, from a sequence of images (frames) in 2D, thus obtaining significant information of the object impossible to acquire and represent in two dimensions. In addition, the idealization and execution of a domotic device control system as a case study, which translates a defined set of gestural signs used in American Sign Language (ASL), into instructions that manipulate intelligent terminal devices through Amazon Alexa Smart Home Skill.

In the proposed system the signs or signals are automatically recognized by applying VA preprocessing techniques, likewise with 3D hand shape retrieval by means of TSF. Subsequently, their three-dimensional representation is reduced, by means of the technique of dimensionality reduction in unsupervised learning (tSNE), into a correct input for the application of a classification model known as Recurrent Neural Network (RNN), in order to classify them within a predefined set of instructions. The chosen classification model is the result of a comparison between the support vector machine model (SVM) and RNN, the latter being the most accurate.

The final part of the paper contains the results of the implementation of the control system for home automation devices, as well as conclusions and recommendations on the application of TSF in automatic signal recognition.

Keywords: Artificial Vision, Trajectory Space Factorization, Deep Learning, Recurrent Neural Networks, Support Vector Machines, Dataset, Training, Validation, Client-Server Model, Smart Home, Amazon Alexa, Amazon Alexa Smart Home Skill.

1. INTRODUCCIÓN

1.1. Estructura del Documento

El presente trabajo de titulación está compuesto en 6 secciones:

- **Contenido previo:** Conjunto de hojas obligatorias que cumplen con el esquema general del documento.
- **Introducción:** Sección del documento donde se describe información, objetivos, alcances de la aplicación realizada. Además, esta sección se incluye el Marco Teórico.
- **Metodología:** Sección del documento donde se especifica el desarrollo y evaluación de la aplicación. Además, se describen las metodologías de desarrollo empleadas.
- **Resultados y Discusión**: Sección del documento donde se detalla al producto final, los resultados de su evaluación y la respuesta obtenida por la academia y el público. Además, contiene información sobre la arquitectura y base de datos de la aplicación.
- **Conclusiones y Recomendaciones:** Sección del documento donde se describen las conclusiones obtenidas y las recomendaciones para trabajos futuros.
- **Referencias Bibliográficas**: Lista de referencias (formato IEEE) hacia documentos, páginas web, artículos que han sido tomados en cuenta para la realización del presente documento.

1.2. Descripción del Problema

El mundo actual se transforma hacia una era en la cual la tecnología está presente en la mayoría de las actividades del ser humano, y a la vez, se ha convertido en una ciencia más al servicio del hombre. Al comenzar el tercer milenio, la humanidad está creando una red global de transmisión instantánea de información, de ideas y de juicios de valor en la ciencia, el comercio, la educación, el entretenimiento, la política, el arte, la religión, y en todos los demás campos[1]. En específico, las ciencias de la computación han contribuido a la ampliación del saber humano con su aportación en diferentes campos, por ende, la interrelación humano-máquina debe ser más fácil y accesible con el fin que este recurso sea aprovechado completamente en el desarrollo de la civilización.

En interacción Humano Computador (IHC) se entiende interacción como un diálogo entre la computadora y el humano[2]. Una mejor interacción entre el humano y computador se

logra cuando se aprovecha el lenguaje natural utilizado por el hombre; la comunicación juega un rol importante en la correlación de estos “sujetos”. Existen tres tipos de vías comunicativas usadas por el hombre, el canal escrito, oral o gestual [3]; cada vía utiliza un código de patrones que los representan y hacen uso para su entendimiento. La vía comunicativa gestual ha sido desatendida ya que no es un canal frecuente de comunicación; 466 millones de personas en todo el mundo son sordas [4], es decir el 5% de la población mundial; el que no sea un campo abiertamente explorado deja muchas posibilidades a su estudio. El futuro de la IHC se centra en el análisis, comprensión y desarrollo de modelos que coadyuven en la búsqueda de una comunicación más sencilla y práctica.

La visión computacional (VC) puede contribuir al reconocimiento automático de señales, ya que su función principal es reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes [5]. Actualmente la VC amplió el alcance de lo que se puede lograr con imágenes concatenadas de un mismo objeto, al recolectar, procesar, y analizar datos que en conjunto permiten reconocerlo. Cada señal de manos capturada en una serie de imágenes son la representación visual de un concepto, por lo una vez reconocido se puede transformar el significado de la forma, movimiento y ubicación espacial de las manos hacia una representación semántica, sin embargo, es realmente difícil llegar a reconocer una señal de manos con la información que puede ofrecer una imagen en dos dimensiones, por lo que actualmente se utilizan sensores que ayuden en su reconocimiento. La factorización de la trayectoria espacial (FTE) busca determinar los movimientos significativos de una persona, directamente a partir de los datos de la imagen [6].

Lo que se cuestiona en este proyecto integrador es:

- ¿Es posible aplicar la factorización de la trayectoria espacial en el reconocimiento automático de señales manuales y estos puedan ser interpretados como instrucciones ejecutadas por un sistema de control de dispositivos domóticos mediante el uso de habilidades del asistente virtual Amazon Alexa?

Con esta pregunta se obtienen las siguientes interrogantes:

- ¿La técnica de factorización de la trayectoria espacial puede usarse para el procesamiento de imágenes que tienen como fuente un video en tiempo real utilizando una cámara web ordinaria?

- ¿Qué condiciones deben existir para que un sujeto en cámara pueda gestuar señales efectuadas por las manos y estas puedan ser reconocidas?
- ¿Puede una señal reconocida representar una orden específica a ser ejecutada? ¿Es posible utilizar habilidades del asistente virtual Amazon Alexa en la creación de un sistema de control de dispositivos domóticos?

Este proyecto de titulación busca dar respuesta a estas interrogantes mediante la investigación respecto a visión artificial y procesamiento de imágenes, específicamente con la aplicación de la técnica de factorización de la trayectoria espacial, y posteriormente trabajar en el desarrollo de un sistema de control que permita realizar las siguientes operaciones:

- Reconocer automáticamente señales manuales específicas.
- Mapear señales manuales especificas en un set de instrucciones.
- Enviar la instrucción al asistente virtual Alexa por medio de la creación de una habilidad de Amazon Alexa.
- Crear una habilidad de Amazon Alexa que me permita controlar un dispositivo domótico; el caso de estudio escogido es el control de interruptores de luz inteligentes.

1.3. Objetivos

1.3.1. Objetivo General

Aplicar la factorización de la trayectoria espacial para el reconocimiento automático de señales para su uso en el desarrollo de un sistema de control de dispositivos domóticos mediante el uso del asistente virtual Amazon Alexa.

1.3.2. Objetivos Específicos

- Investigar respecto a visión artificial y procesamiento de imágenes, específicamente con la aplicación de la técnica de factorización de la trayectoria espacial.
- Evaluar las condiciones que deben existir para que un sujeto en cámara pueda gesticular señales y estas puedan ser reconocidas.
- Establecer el prototipo del sistema de control que interprete una señal como una instrucción a ser enviada y ejecutada por el asistente virtual Amazon Alexa mediante el uso de habilidades.

- Definir el caso de estudio para la prueba del prototipo en el control de dispositivos domóticos, específicamente interruptores de luz inteligentes.

1.4. Marco Teórico

Sección del documento utilizada para establecer el contexto técnico sobre el cual se desarrolló este trabajo y la arquitectura de la aplicación. Además, contiene información sobre el marco legal que rige el desarrollo y funcionamiento de aplicaciones tipo Marketplace dentro del territorio ecuatoriano. Finalmente, se exponen las herramientas, frameworks y librerías de desarrollo empleadas durante el desarrollo del software.

1.4.1. Conceptos Lingüísticos

1.4.1.1. Lengua

La lengua es un conjunto de elementos significativos y de señales asociados arbitrariamente, el cual tiene sus reglas de combinación con cuya realización en cadena pueden expresarse pensamientos, emociones y deseos [7]. Lengua difiere de la palabra *lenguaje,* al expresar este último una connotación netamente de comunicación entre dos individuos, a diferencia de representar todo un idioma con estructura y vocabularios propios.

1.4.1.2. Lengua de Señas Estadounidenses

La lengua de señas estadounidense (American Sign Language ASL) es un lengua completa y natural que tiene las mismas propiedades lingüísticas que las lenguas habladas, con una gramática que difiere del inglés. El ASL se expresa mediante movimientos de las manos y la cara [8] , la Figura 1 muestra un ejemplo de las diferentes formas de movimiento utilizadas en ASL. Cabe mencionar que, la lengua de señas es especifica de una región o país, por lo que no tiene una característica universal.

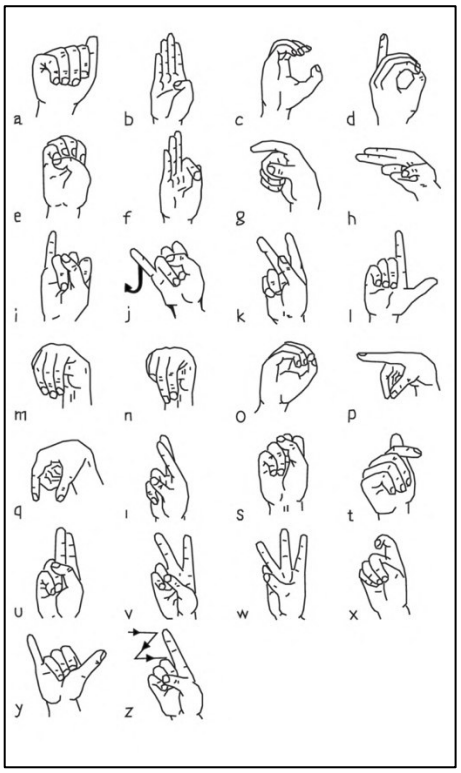


Figura 1. *Abecedario de la lengua de señas estadounidense (ASL)*

ASL se basa en la idea de que la visión es la herramienta más útil que tiene una persona sorda o muda para comunicarse y recibir información[9]. En el desarrollo del presente proyecto ASL es utilizado como el canal de comunicación entre el computador y el humano.

1.4.2. Conceptos Técnicos

En esta sección del documento se detalla las definiciones teóricas y técnicas involucradas en el desarrollo de este proyecto.

1.4.2.1. Visión artificial

Dentro de las ciencias de la computación, específicamente en la Inteligencia Artificial (IA), la visión artificial es un campo en auge el cual se centra en proveer al computador todas las capacidades visuales que tiene el ser humano al momento de percibir imágenes del mundo real. Varios autores consideran a la visión artificial como una materia multidisciplinaria que utiliza técnicas procedentes de la óptica, la electrónica, la ingeniería mecánica e informática, en la cual, mediante la utilización de las técnicas adecuadas, permite la obtención, procesamiento y análisis de cualquier tipo de información especial obtenida a través de imágenes digitales [10] [11].

Dotar al computador la capacidad de ver es una tarea compleja, ya que al vivir en un mundo tridimensional (3D) y, cuando las computadoras intentan analizar objetos en el espacio 3D, los sensores visuales disponibles (por ejemplo, cámaras web) suelen ofrecer imágenes bidimensionales (2D), al ser esta una proyección con un número inferior de dimensiones supone una enorme pérdida de información. Un sistema de VA trata de solucionar este inconveniente recuperando información útil sobre una escena a partir de sus proyecciones bidimensionales. Hoy en día varios investigadores de VA han desarrollado, en paralelo, técnicas matemáticas, modelos basados en física y probabilística para recuperar la forma tridimensional y la apariencia de los objetos en las imágenes, disponiendo de técnicas fiables para calcular con precisión un modelo tridimensional parcial de un entorno a partir de miles de fotografías parcialmente superpuestas. Sin embargo, modelar el mundo visual en toda su rica complejidad es mucho más difícil que, por ejemplo, modelar el tracto vocal que produce los sonidos hablados [12][13][14].

Para entender el tipo de información que se manipula en VA, se debe tener noción de lo que engloba la representación de una imagen, la Figura 2 muestra a grandes rasgos una posible división de la organización de los datos en cuatro niveles [12], en donde cada nivel es representado por un ovalo sombreado.

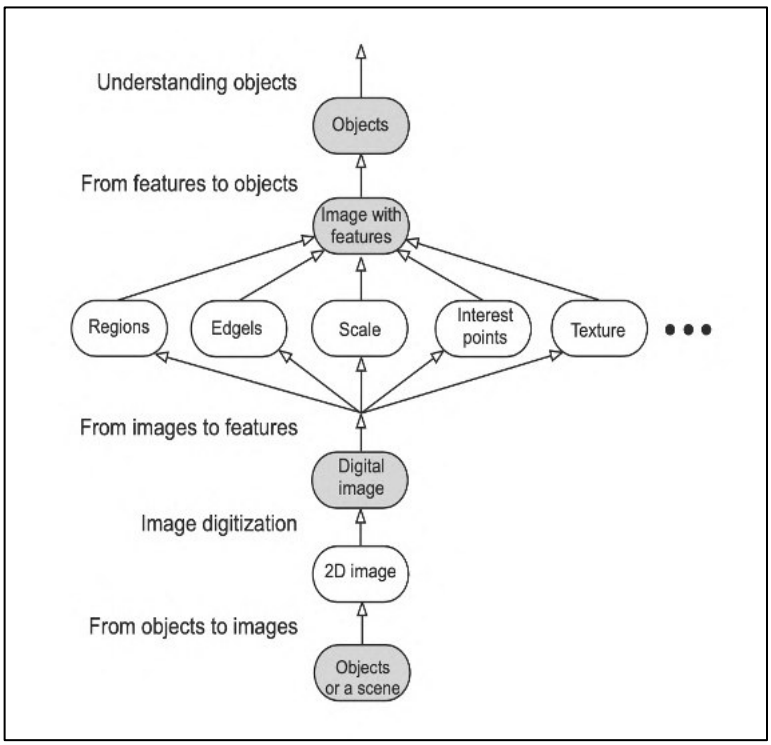


Figura 2. Cuatro posibles niveles de representación de imágenes indicados para problemas de análisis de imágenes en los que hay que detectar y clasificar objetos.

1.4.2.1.1. Geometría de la imagen

Como anteriormente se mencionó la VA es comprendida como la abstracción de múltiples bloques en una escena, por lo que cada imagen o conjunto de bloques tienen propiedades geométricas y espaciales que se mencionan a continuación:

1. La geometría de la formación de la imagen, que determina en qué lugar del plano de la imagen se situará la proyección de un punto de la escena.
2. La física de la luz, que determina el brillo de un punto en el plano de la imagen en función de la iluminación de la escena y las propiedades de la superficie.

Dentro de las propiedades físicas de una imagen, el modelo básico para la proyección de los puntos de una imagen en un plano cartesiano se muestra en la Figura 4. En este modelo, el centro de proyección del sistema de imagen coincide con el origen del sistema de coordenadas tridimensional. El sistema de coordenadas para los puntos de una escena es el espacio tridimensional abarcado por los vectores unitarios x, y, z que forman los ejes del sistema de coordenadas. Un punto de una escena tiene coordenadas (x, y, z). La coordenada x es la posición horizontal del punto en el espacio visto desde la cámara, la coordenada y es la posición vertical del punto en el espacio visto desde la cámara, y la coordenada z es la distancia desde la cámara al punto en el espacio a lo largo de una línea paralela al eje z . La línea de visión de un punto en la escena es la línea que pasa por el punto de interés y el centro de proyección. La línea dibujada en la Figura 4 es una línea de visión [13].

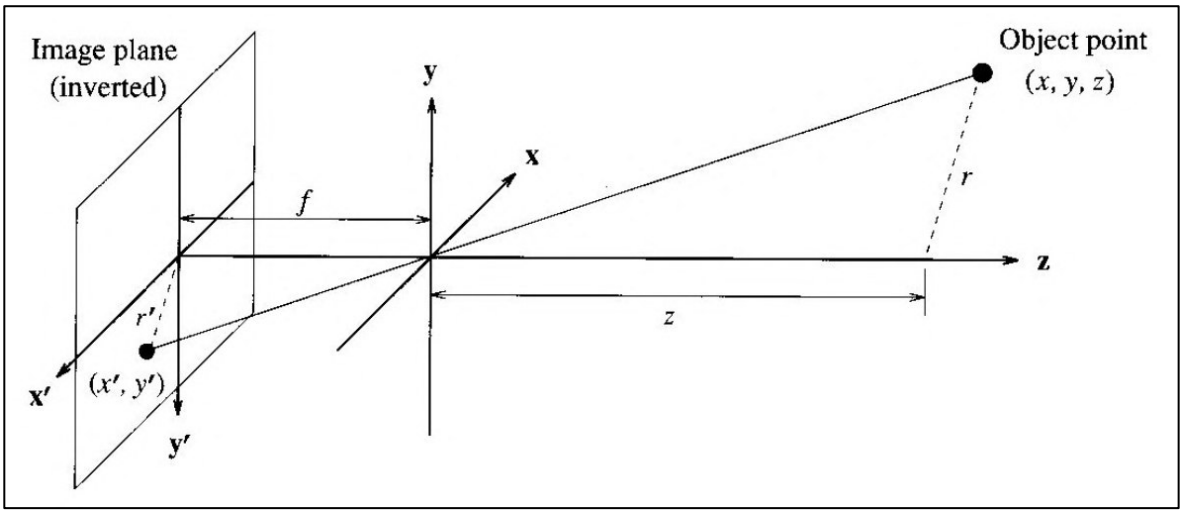


Figura 3. El punto en el plano de la imagen que corresponde a un punto concreto de la escena se encuentra siguiendo la línea que pasa por el punto de la escena y el centro de proyección.

El plano de la imagen es paralelo a los ejes x e y del sistema de coordenadas a una distancia f del centro de proyección, como se muestra en la Figura 4. El plano de la imagen en una cámara real está a una distancia f detrás del centro de proyección y la imagen proyectada está invertida. Es habitual evitar esta inversión suponiendo que el plano de la imagen está delante del centro de proyección, como se muestra en la Figura 5. El plano de la imagen está atravesado por los vectores x' e y' para formar un sistema de coordenadas bidimensional para especificar la posición de los puntos en el plano de la imagen. La posición de un punto en el plano de la imagen se especifica mediante las dos coordenadas x' e y'. El punto $(0,0)$ en el plano de la imagen es el origen del plano de la imagen. La posición en el plano de la imagen de un punto en la escena se encuentra mediante la intersección de la línea de visión con el plano de la imagen de acuerdo con el esquema de proyección[13].

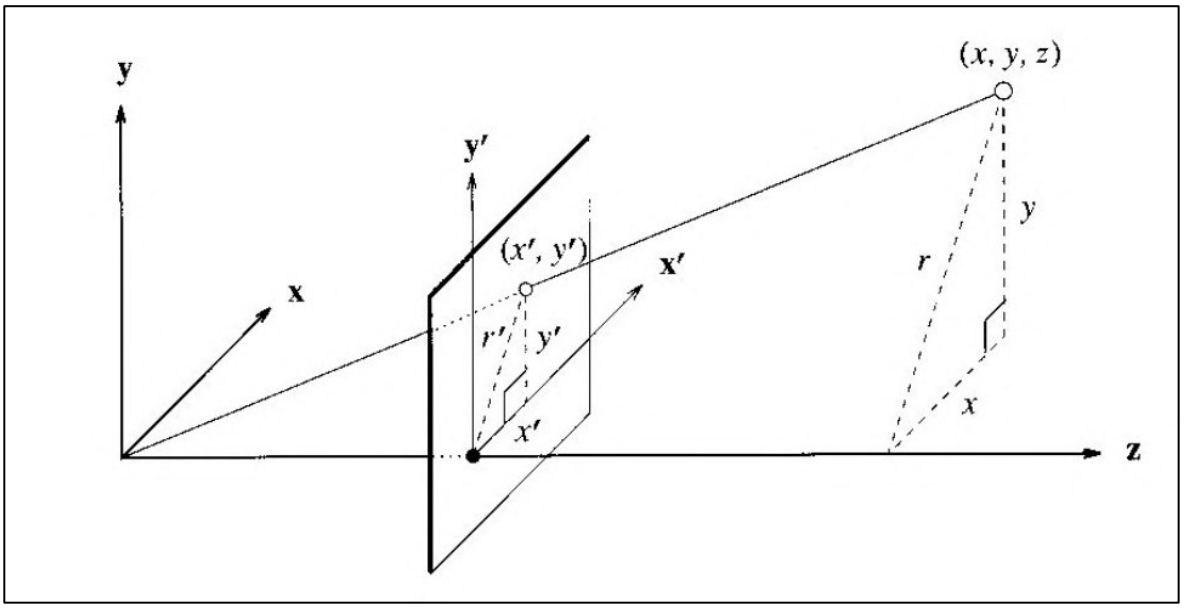


Figura 4. *Una ilustración que muestra la línea de visión que se utiliza para calcular el punto proyectado* (x', y') *desde el punto objeto* (x, y, z).

Proyección de la perspectiva

La posición (x', y') en el plano de la imagen de un punto en la posición (x, y, z) de la escena se encuentra calculando las coordenadas (x', y') de la intersección de la línea de visión que pasa por el punto de la escena (x, y, z) con el plano de la imagen, como se muestra en la Figura 5. La distancia del punto (x, y, z) al eje z es enunciada en la Ecuación 1, y la distancia del punto proyectado (x', y') al origen del plano de la imagen es $r = \sqrt{x^2 + y^2}$, y la distancia del punto proyectado (x', y') al origen del plano de la imagen es $r' = \sqrt{x'^2 + y'^2}$. El eje z, la línea de visión al punto (x', y') en el plano de la imagen, y el segmento de línea de longitud r' desde el punto (x', y') al eje z (perpendicular al eje z) forman otro triángulo. Los dos triángulos son similares, por lo que los cocientes de los lados correspondientes de los triángulos deben ser iguales[13].

$$\frac{f}{z} = \frac{r'}{r}$$

Ecuación 1. *Distancia del punto* (x, y, z) *al eje* z.

El triángulo formado a partir de las coordenadas (x, y) y la distancia perpendicular r y el triángulo formado a partir de las coordenadas del plano imagen (x', y') y la distancia perpendicular r' son también triángulos semejantes[13].

$$\frac{x'}{x} = \frac{y'}{y} = \frac{r'}{r}$$

Ecuación 2. *Igualdades entre la proporción de las coordenadas tridimensionales* (x, y, z) *y su proyección bidimensional* (x', y').

Combinando las ecuaciones Ecuación 1 y Ecuación 2 se obtienen las ecuaciones de la proyección en perspectiva[13].

$$\frac{x'}{x} = \frac{f}{z}$$

Ecuación 3*. Modelo de proyección en perspectiva en la coordenada* x..

$$\frac{y'}{y} = \frac{f}{z}$$

Ecuación 4*. Modelo de proyección en perspectiva en la coordenada* y..

La posición de un punto (x, y, z) en el plano de la imagen viene dada por las siguientes ecuaciones[13].

$$x' = \frac{f}{z}x$$

Ecuación 5*. Posición de un punto en el plano bidimensional de la imagen en la coordenada* x..

$$\mathrm{y}' = \frac{\mathrm{f}}{\mathrm{z}}\mathrm{y}$$

Ecuación 6*. Posición de un punto en el plano bidimensional de la imagen en la coordenada* y..

Sistemas de coordenadas

Con fines demostrativos se asume que el centro de proyección coincide con el origen del espacio tridimensional y que los ejes de la cámara están alineados con el sistema de coordenadas utilizado para especificar la ubicación de un punto en la escena. En la práctica, una cámara puede desplazarse y girar con respecto al sistema de coordenadas tridimensional utilizado para especificar las coordenadas de los puntos de la escena. Las coordenadas (x_a, y_a, z_a) en el sistema de coordenadas ideal deben transformarse en las coordenadas (x_c, y_c, z_c) del punto en el sistema de coordenadas de la cámara antes de proyectar los puntos en el plano de la imagen[13].

Muestreo y Cuantificación

Al ser una imagen un conjunto de señales captadas desde un sensor de forma continua, matemáticamente se expresa como una función continua, sin embargo, no puede representarse exactamente en un computador digital. La interfaz entre el sistema óptico que proyecta una escena en el plano de la imagen y un computador deben muestrear la imagen en un número fino de puntos y representar cada muestra dentro de un conjunto finito de palabras en el computador, como se muestra en la Figura 6. Podemos definir la cuantificación como un proceso en el que una señal se transforma de forma que sus valores se redondean a unos valores determinados (discretos) (niveles de cuantificación), mientras que el muestreo es el proceso de mapear el dominio de una señal desde el espacio $\mathbb{R}$ (números reales) al espacio $\mathbb{N}$ (números naturales)[13][15].

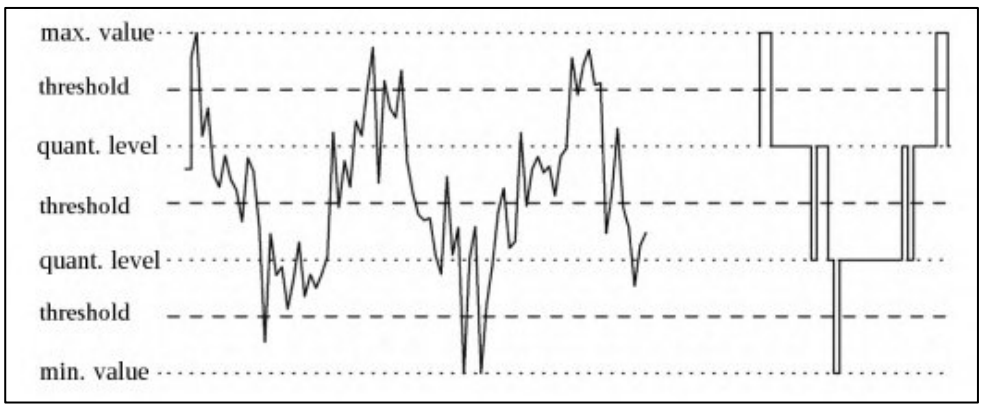


***Figura 5**. Señal continua después y antes de cuantificación.*

Cada muestra de la imagen se denomina píxel. Cada píxel se representa en el ordenador como un pequeño número entero. Con frecuencia, el píxel se representa como un entero de 8 bits sin signo en el rango [0,255], donde el 0 corresponde al negro, el 255 al blanco y los tonos de gris se distribuyen en los valores intermedios[16].

Muchas cámaras adquieren una imagen analógica, que luego se muestrea y cuantifica para convertirla en una imagen digital. La frecuencia de muestreo determina cuántos píxeles tendrá la imagen digital, por lo que representa la resolución de la imagen, y la cuantificación determina cuántos niveles de intensidad se utilizarán para representar el valor de la intensidad de grises en cada punto de muestreo. Como se muestra en las Figuras 6 y 7, una imagen tiene un aspecto muy diferente con distintas frecuencias de muestreo y niveles de cuantificación. En la mayoría de las aplicaciones de VA, las tasas de muestreo y cuantificación están predeterminadas debido a la limitada elección de cámaras y hardware de adquisición de imágenes disponibles[13][17].

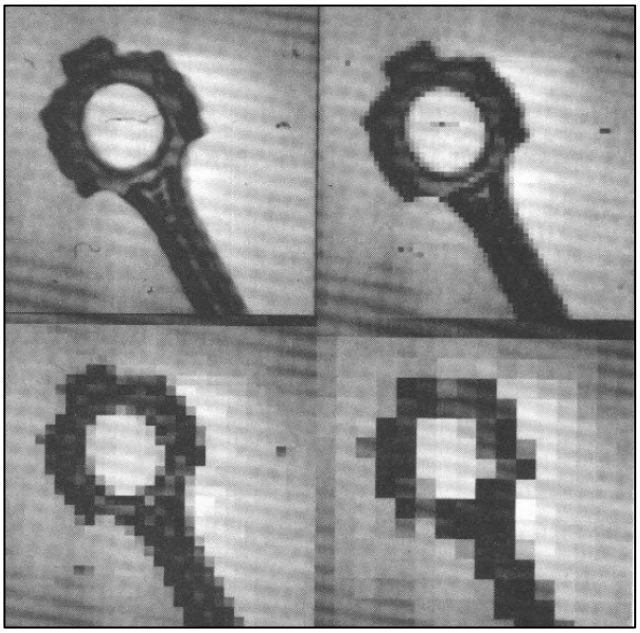


Figura 6. Una imagen mostrada con diferentes resoluciones espaciales. Arriba a la izquierda: Imagen original muestreada a 256 x 256 y 128 niveles de gris. Arriba a la derecha: 64 x 64. Abajo a la Izquierda: 32 x 32. Abajo a la derecha: 16 x 16.

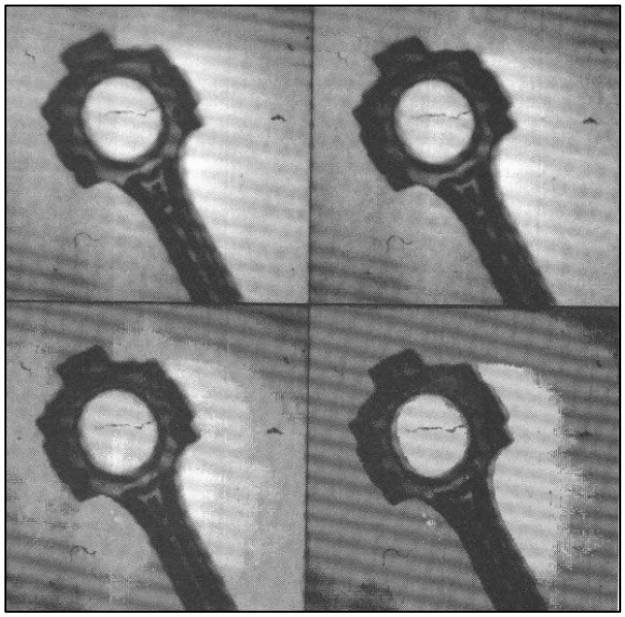


Figura 7. Una imagen mostrada con diferentes resoluciones de nivel de gris. Arriba a la izquierda: Imagen muestreada a 32 niveles de gris. Arriba a la derecha: 64 niveles de gris Abajo a la izquierda: 8 niveles de gris, Abajo a la derecha: 4 niveles de gris.

Definiciones de la Imagen

Es importante entender la relación entre la geometría de la formación de la imagen, descrita en las secciones anteriores, y la representación de las imágenes en el computador. Debe existir una conexión entre la notación matemática utilizada para desarrollar algoritmos de VA y la notación algorítmica utilizada en los programas. Una imagen es una matriz bidimensional de píxeles, en donde cada píxel es una muestra de la intensidad de la imagen cuantificada en un valor entero. Los índices de fila y columna $[i, j]$ del píxel son valores enteros que especifican la fila y la columna en la matriz de valores de píxeles. El índice i apunta hacia abajo y el j hacia la derecha; esta notación de índices se corresponde estrechamente con la sintaxis de matrices utilizada comúnmente en la representación computacional. Las coordenadas (x, y) son números reales almacenados como números de punto flotante en el ordenador[13][18][19].

Las coordenadas del plano de la imagen (x, y) pueden calcularse a partir de las coordenadas de los píxeles $[i, j]$ de una matriz de $(n * m)$ píxeles mediante las fórmulas[13]:

$$x = j - \frac{m-1}{2}$$

Ecuación 7. *Valor de la coordenada x a partir del índice j de las coordenadas de los píxeles $[i, j]$.*

$$y = -\left(i - \frac{n-1}{2}\right)$$

Ecuación 8. *Valor de la coordenada y a partir del índice i de las coordenadas de los píxeles $[i, j]$.*

En un sistema de imágenes, cada píxel ocupa un área finita en el plano de la imagen. La matriz de píxeles en un programa informático corresponde a la cuadrícula de ubicaciones del plano de la imagen en la que se obtuvieron las muestras, como se ilustra en la Figura 9 [13].

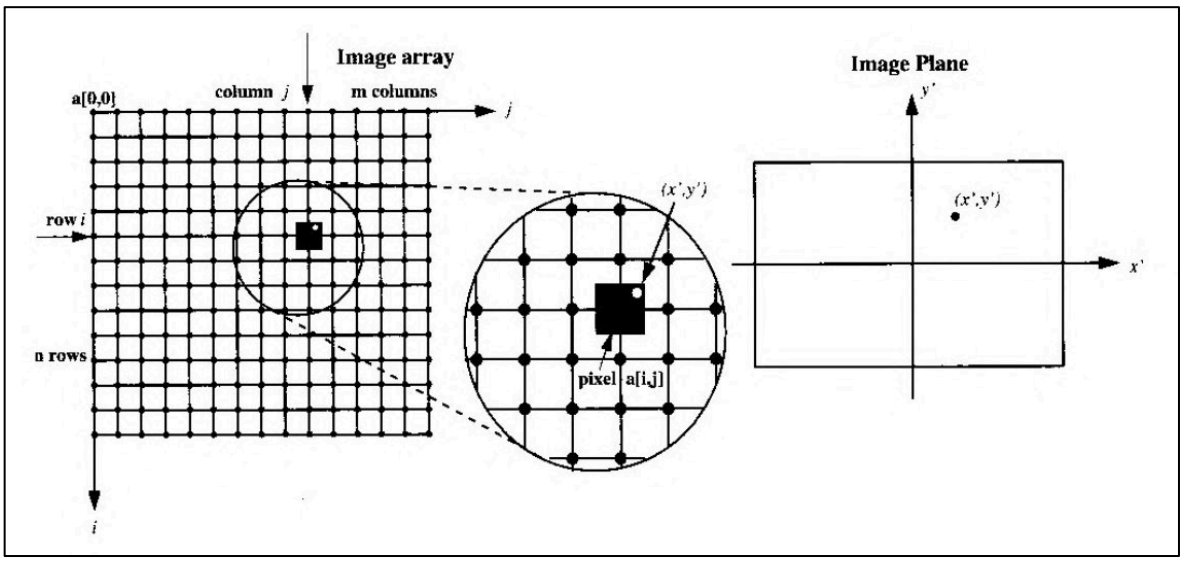


Figura 8*. Relación entre las coordenadas del plano de la imagen y los índices de la matriz de la imagen.*

Un píxel es tanto un valor de escala de grises, que es una muestra cuantificada de la intensidad continua de la imagen, como la ubicación de la imagen, especificada como los índices de fila y columna en la matriz de imágenes, la cual se obtiene mediante el muestreo de la intensidad de la imagen en puntos de una cuadrícula rectangular[20][19].

1.4.2.1.2. Etapas de la visión artificial

En esta sección se detallan las etapas del proceso de tratamiento de imágenes bidimensionales, generalmente consta de cuatro etapas, en donde se tiene una entrada y

una salida que hace alusión a los resultados. El proceso, ilustrado en la Figura 9, muestra los cuatro pasos: preprocesamiento, segmentación, extracción de características e interpretación. No todas las aplicaciones de VA requieren los cuatro pasos, algunas aplicaciones no necesitan el preprocesamiento y otras no necesitan la segmentación. Cualquiera de estos pasos puede requerir uno o más algoritmos[21].

Figura 9. *Proceso de procesamiento de imágenes de visión artificial.*

- **Procesamiento de Imágenes**

Haber obtenido una imagen con una perfecta abstracción del mundo real es una suposición de la que no se puede tener en la mayoría de los casos, de tal forma que puede acarrear análisis erróneos sobre una entrada de imagen obtenida. En consecuencia, un preprocesamiento resulta de gran utilidad al lograr incrementar en gran medida la efectividad de los análisis posteriores [22]. Entre los procesos más comunes se encuentra la reducción de ruido ilustrado en la Figura 10, el cual elimina en lo posible el ruido introducido por agentes externos o debido a problemas en la calidad de la captura de la imagen[23].

Figura 10. *Reducción de ruido en imagen.*

Además de reducir el ruido, se puede también realizar operaciones que transformen la imagen, en una imagen con nuevas características que podrían ser útiles posteriormente. Por ejemplo, existen casos en los que se prefiere analizar imágenes binarias (blanco y negro sin escalas de grises) con valores para cada píxel de 0 o 1, esta transformación es utilizada debido a su ligereza de procesamiento [24], [25]. Otro inconveniente que se puede resolver en esta etapa es el problema de la naturaleza multi-escala del mundo real, es decir, uno podría tener la tarea de identificar el tipo de objeto observado y si no se puede presuponer una escala específica, entonces se suma la dificultad de que la escala con la que se trabaja para identificar un automóvil es muy distinta a la que se utilizaría para identificar una célula en el torrente sanguíneo. Para poder resolver este problema varios investigadores de VA han desarrollado un sistema de teorías llamada Scale-space (Espacio de Escala)[26], el cual permite organizar, comparar y analizar objetos con estructuras de diversos tamaños posibles, mediante una serie de cálculos matemáticos de discretización gaussiana[27].

- **Segmentación**

La visión del ser humano tiene la capacidad de distinguir partes de una imagen que pueden representan el fondo, un rostro, parte de una silla, entre otros objetos etc. En la etapa previa se busca obtener un conjunto de puntos de referencia con ciertas características similares, que ahora pueden ayudar a identificar zonas de la imagen que en conjunto tengan algún significado dado (véase Figura 11).

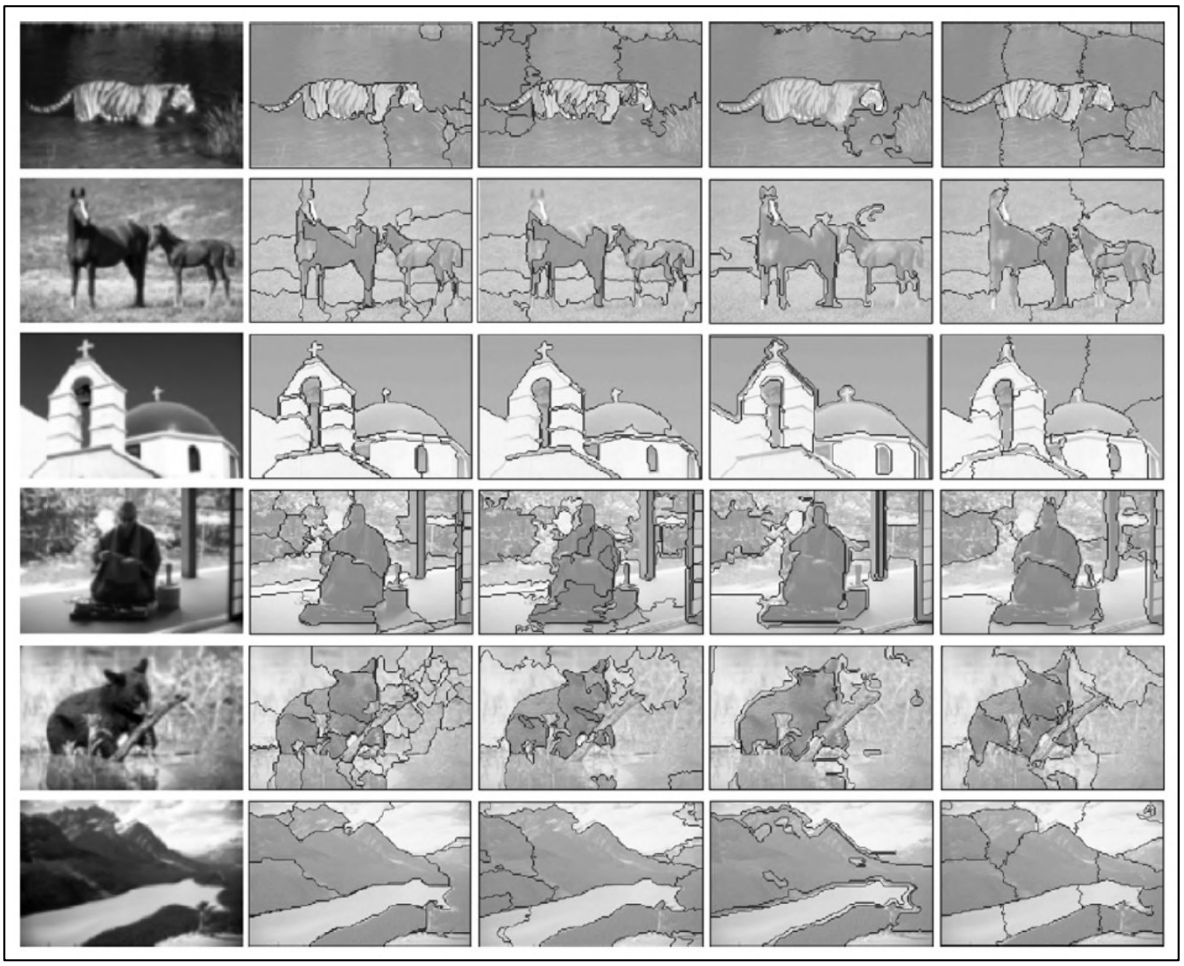


Figura 11. *Segmentación realizada por distintos tipos de algoritmos.*

La segmentación consiste en transformar una imagen compuesta de pixeles en un conjunto de regiones o súper pixeles semánticamente coherentes, o dicho de otra forma, encontrar los grupos de pixeles que están relacionados entre sí por compartir un significado común, como el de un objeto y etiquetarlo mediante algún modelo de abstracción [24][33][36]. Este proceso es fundamental para una gran variedad de aplicaciones como video vigilancia, reconocimiento de objetos, seguimiento de objetos y recuperación de imagen basada en contenido [28].

De la misma manera se puede utilizar la llamada sobre segmentación que forma los súper pixeles de acuerdo con la relación de color con sus píxeles vecinos y logra una detección de bordes especial, dejando regiones pequeñas que no necesariamente tengan un significado sólido. Luego de dicho paso se emplean diversos algoritmos para encontrar la relación entre las diferentes regiones ofreciendo un mayor significado[33] [36].

- **Detección de características**

Como segunda etapa y una vez que la imagen esta lista para los siguientes procesos de interpretación, lo primero que se realiza es detectar y generar abstracciones de todo tipo de información relevante que se pueda recuperar. Se debe examinar la imagen en busca de puntos de interés o en inglés (landmarks), como el caso de analizar el rostro de una persona distinguiendo patrones relacionados con los ojos, nariz y boca como se muestra en la Figura 12 [30].

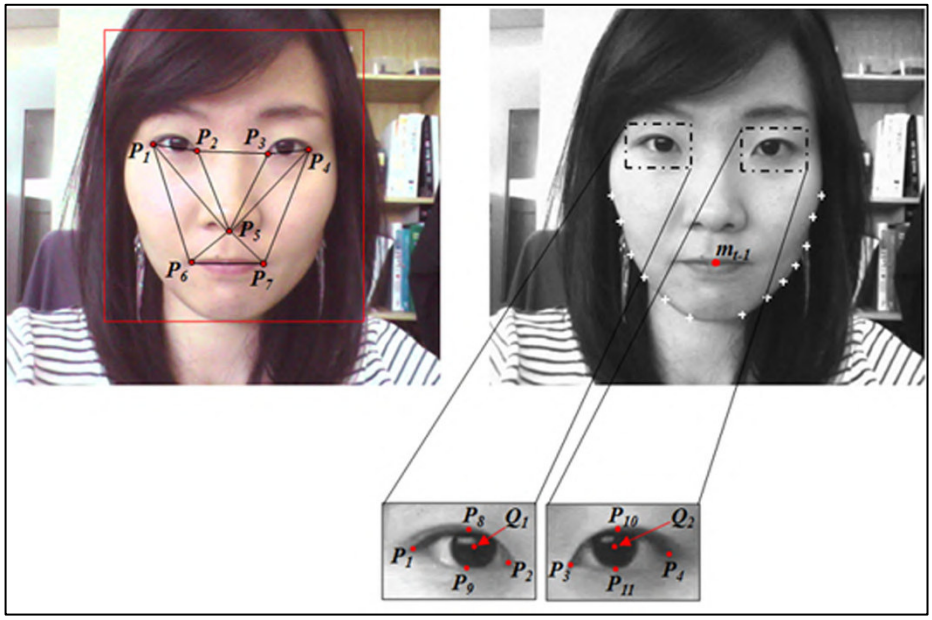


***Figura 12**. Detección de ojos en imagen.*

Se tiene la idea de que este tipo de procesos son rápidos e intuitivos para el ser humano, sin embargo, computacionalmente se puede llegar a tener varias complicaciones. Un ejemplo puntual es si quisiésemos detectar un vehículo, no bastaría con hacer una comparación píxel a píxel con otra imagen de un vehículo adquirido posteriormente ya que con tan solo cambiar un poco la iluminación o el ángulo de la imagen del auto ya no se lograría relacionar las dos imágenes. Por lo tanto, se necesita de procedimientos más ingeniosos y complejos, y no automáticos como se esperaría. Otro tipo de características a señalar son los bordes, los contornos y límites entre los objetos, zonas diferenciadas por brillo y oscuridad, esquinas y todo tipo de propiedades que sean de interés [29]. Las superposiciones entre características se pueden solucionar con cálculos matemáticos que determinan los límites y el grado de oclusión presentados [31]. Otras características

más complejas pueden estar relacionadas con las texturas y hasta el movimiento del objeto [23][32].

- **Procesamiento de alto nivel**

En la cuarta etapa del proceso, el grupo de datos con el que se opera debería ser más reducido debido a las etapas de análisis anteriores. Los datos usualmente se presentan como un conjunto de puntos o regiones que presumiblemente contienen algún interés en concreto como un objeto. El siguiente paso es realizar procesos de interpretación de más alto nivel que en gran medida dependen de la implementación que se desea [30]. Algunas de las tareas típicas en la visión artificial e utilizadas en el desarrollo de este proyecto son explicadas a continuación.

- **Reconstrucción 3D**

Del inglés, Structure from motion (SFM), es una técnica que utiliza un grupo o secuencia de imágenes, provenientes de una filmación, con el objetivo de con construir un modelo tridimensional del objeto en escena que se está captando. Mientras se tenga un mayor número de muestras (imágenes) será directamente proporcional al éxito en la obtención de un mejor modelo de representación del objeto. La presente técnica de procesamiento es la utilizada para la reconstrucción del modelo 3D de la mano, usada en el reconocimiento de señas (véase Figura 13).

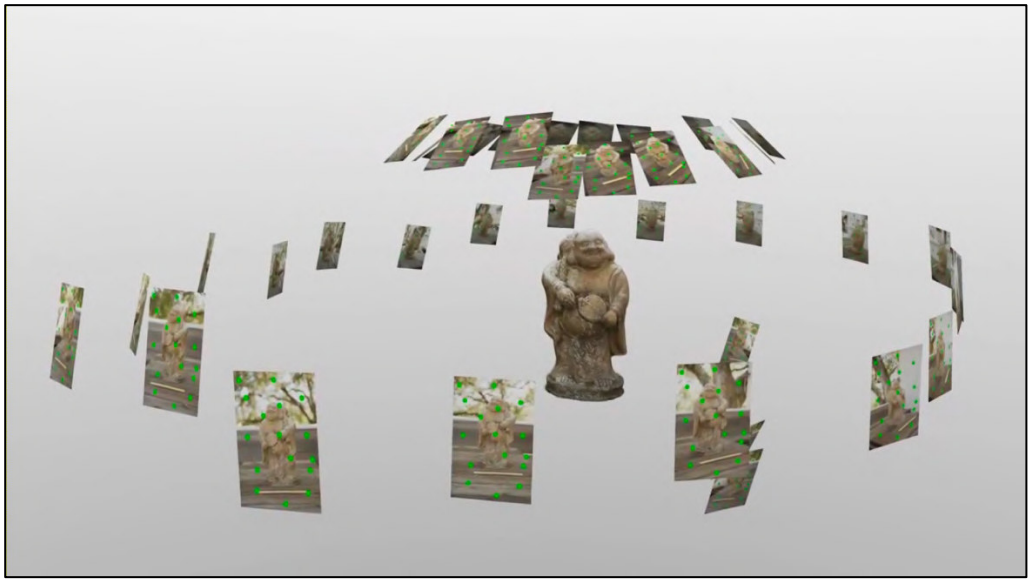


Figura 13. *Ejemplo de Structure from motion (SFM).*

- **Reconocimiento de objetos**

Dentro de VA, uno de los casos más comunes es el determinar si la imagen contiene algún objeto, característica o actividad especifica que se desea reconocer con algoritmos computacionales, tanto para dominios específicos como el reconocimiento de caras, como para dominios de objetos más generales.

La mayoría de estos enfoques requieren objetos segmentados y etiquetados para el entrenamiento, o al menos que el objeto de entrenamiento sea la parte dominante de las imágenes de entrenamiento. Ninguno de estos algoritmos puede ser entrenado en imágenes no etiquetadas que contengan grandes cantidades de desorden o múltiples objetos[33] (véase Figura 14).

Actualmente la detección de rostro es de uso muy frecuente como por ejemplo en las fotografías de las redes sociales o en cámaras fotográficas. Otra área muy estudiada es la interpretación de la imagen para describir cualidades más características, comprendiendo que lo observado se trata de un objeto, animal, persona o paisaje. También se puede llevar el reconocimiento al ámbito de la identificación haciendo una búsqueda en una base de datos para encontrar una correspondencia un registro especifico, como seria la identificación facial y dactilar[34][35][32][36].

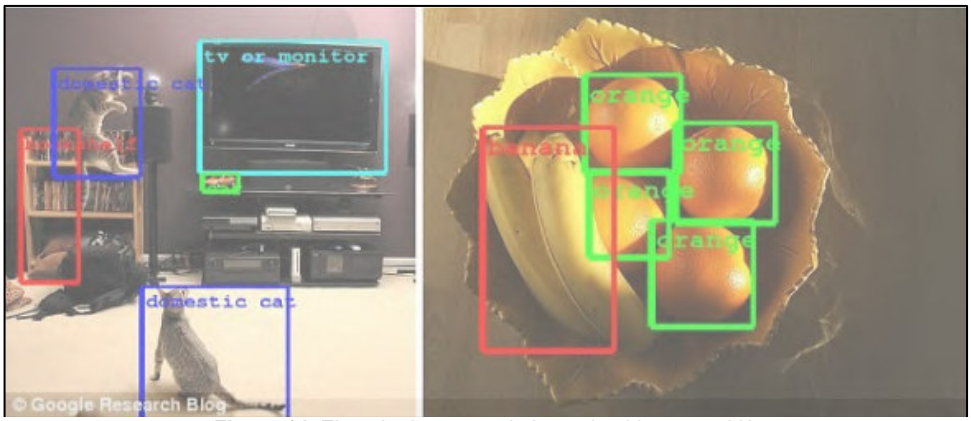


Figura 14. *Ejemplo de reconocimiento de objetos con VA*

1.4.2.1.3. Limitaciones

La pérdida de información en 3D en 2D es una manifestación que se produce en dispositivos típicos de captación de imágenes, como una cámara o un ojo. Sus propiedades geométricas se han aproximado durante siglos mediante un modelo estenopeico o del inglés “pinhole" (una caja con un pequeño agujero, una "cámara oscura" en latín). Este modelo físico se corresponde con un modelo matemático de proyección en perspectiva; la Figura 15 resume este principio. La transformación proyectiva asigna los puntos a lo largo de los rayos, pero no conserva los ángulos ni la colinealidad[12].

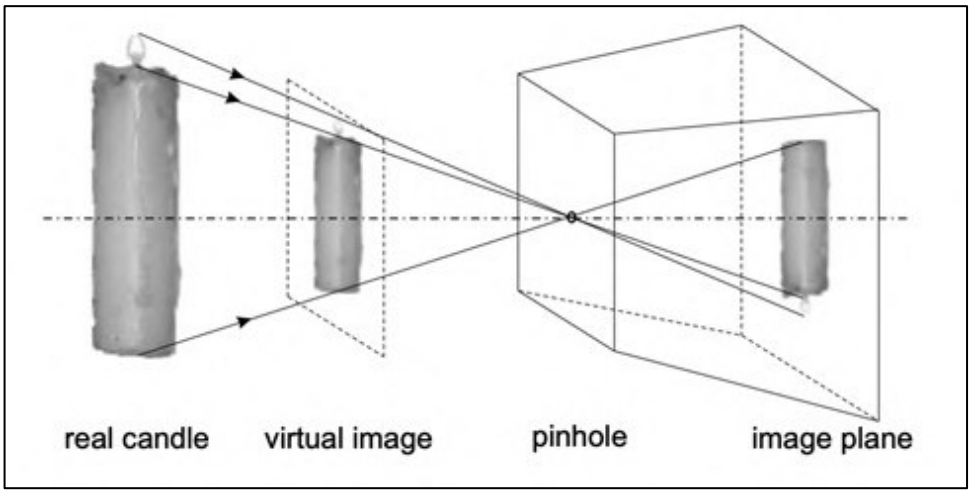


Figura 15. *El modelo estenopeico de la geometría de la imagen no distingue el tamaño de los objetos.*

El principal problema del modelo estenopeico y de la vista única disponible es que la transformación proyectiva ve un objeto pequeño cerca de la cámara de la misma manera que un objeto grande alejado de la cámara. En este caso, un ser humano necesita una "vara de medir" para adivinar el tamaño real del objeto que el ordenador no tiene. De manera habitual, los algoritmos de análisis de imágenes analizan una determinada ubicación (por ejemplo, un píxel de la imagen) y su vecindad local; el ordenador ve la imagen a través de una ventana de llave; esto dificulta mucho la comprensión de un contexto más global. A menudo es muy difícil interpretar una imagen si sólo se ve localmente o si sólo se dispone de unas pocas claves locales. La Figura 16 lo ilustra gráficamente[12].

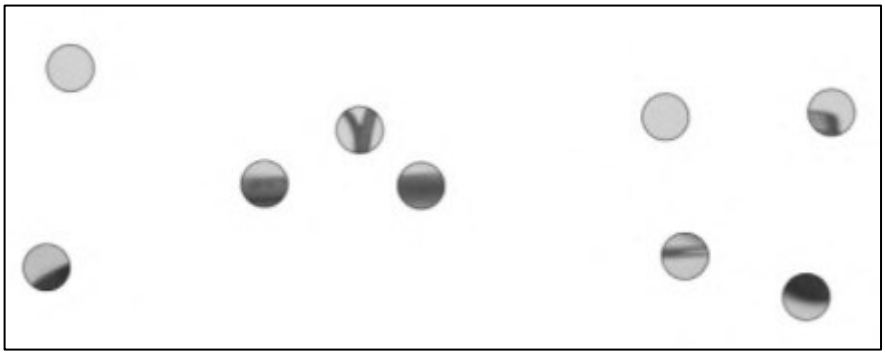


Figura 16. *Ilustración del mundo vista a través de varios ojos de buey que sólo proporcionan un contexto local. Es muy difícil adivinar qué objeto se representa.*

1.4.2.1.4. Aplicaciones

Dentro de la industria, la mayoría de los fabricantes se preocupan por la integridad óptica de sus productos; los clientes suelen equiparar la calidad de la apariencia con la calidad funcional. Por lo tanto, para garantizar el éxito de la comercialización a largo plazo de un artículo, es muy conveniente que su aspecto se compruebe visualmente antes de su envasado y envío. Asimismo, es deseable que el proceso de inspección esté automatizado y se realice sin intervención humana.

Los sistemas de VA pueden ayudar a un médico a recuperar información mejorando la calidad de las imágenes, además de realizar fácilmente mediciones cuantitativas de regiones de interés (véase Figura 17). Estos sistemas se están desarrollando para todos los modos de imagen útiles en diferentes aspectos de la atención sanitaria, por otro lado, se usan aplicaciones similares para la inspección de productos industriales, agrícolas y otros.

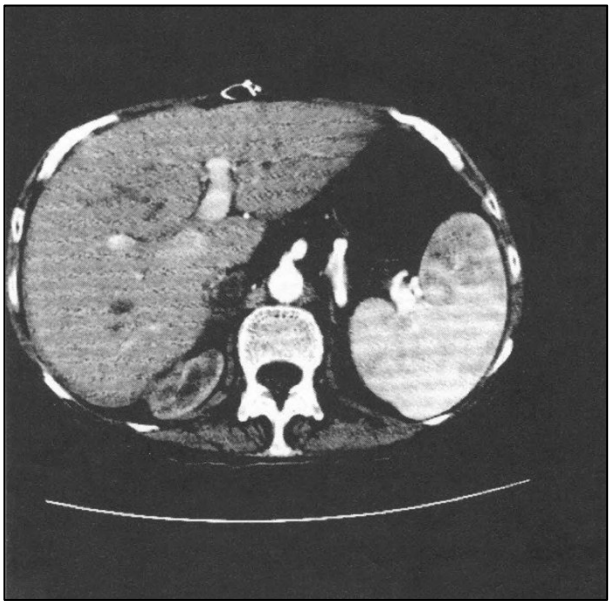


***Figura 17**: TC (tomografía computarizada de rayos X transversal) con contraste de un hígado humano mostrada en la configuración de la ventana de tejidos blandos.*

La visión por ordenador se utiliza hoy en día en una amplia variedad de aplicaciones del mundo real[14], entre las que se incluyen:

(i) Reconocimiento óptico de caracteres (OCR): lectura de códigos postales manuscritos en cartas (Figura 18a) y reconocimiento automático de matrículas (ANPR).

(ii) Inspección de máquinas: inspección rápida de piezas para garantizar la calidad mediante visión estereoscópica con iluminación especializada para medir las tolerancias de las alas de los aviones o de las piezas de la carrocería de los automóviles (Figura 18b).

(iii) Tiendas mayoristas: reconocimiento de objetos para cajas registradoras (Figura 18c).

(iv) Construcción de modelos 3D (fotogrametría): construcción totalmente automática de modelos 3D a partir de fotografías aéreas utilizadas.

(v) Imágenes médicas: registro de imágenes preoperatorias e intraoperatorias (Figura 18d).

(vi) Seguridad en automotores: detección de obstáculos inesperados, como peatones en la calle (Figura 18e).

(vii) Match Moving: fusión de imágenes generadas por ordenador (CGI) con secuencias de acción real mediante el seguimiento de puntos característicos en el vídeo de origen para estimar el movimiento de la cámara en 3D y la forma del entorno.

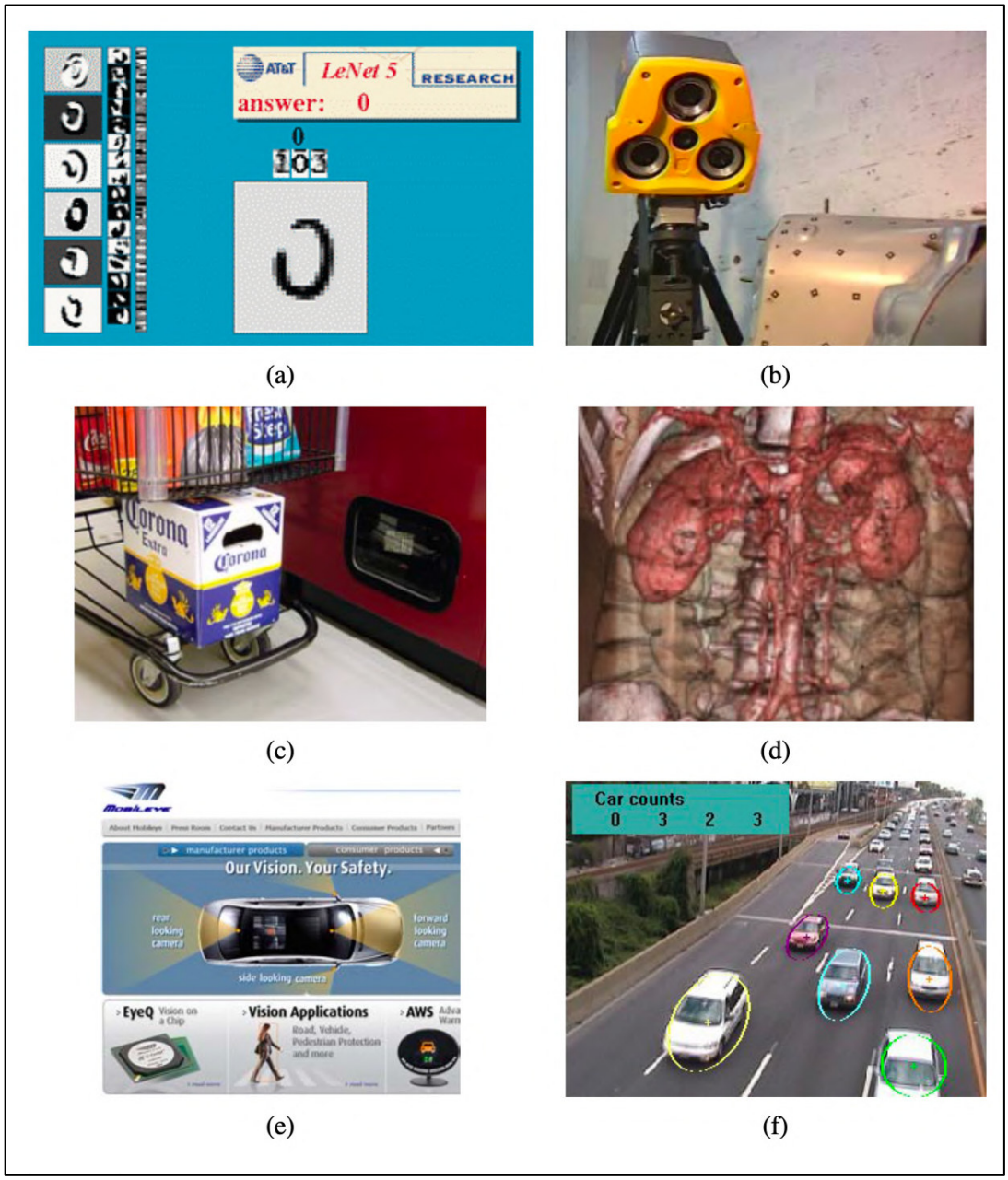


Figura 18. *Algunas aplicaciones industriales de la visión por ordenador.*

(viii) Captura de movimiento (mocap): uso de marcadores retro reflectantes vistos desde múltiples cámaras u otras técnicas basadas en la visión para capturar actores para la animación por ordenador.

(ix) Vigilancia: control de acceso por detección de rostros, análisis del tráfico en las carreteras (Figura 18f).

(x) Reconocimiento de huellas dactilares: para la autentificación automática de accesos, así como para aplicaciones forenses.

1.4.2.2. Machine Learning

ML se define como un subcampo de las ciencias de la computación en el cual permite a computadoras tener la habilidad de aprender sin la necesidad de tener explícitamente en su programación el código fuente de su aprendizaje. Un punto clave en ML es el concepto de *aprendizaje automático* o self-learning que se refiere a la aplicación de modelos estadísticos para detectar patrones e improvisar el rendimiento basado en los datos e información empírica, todo ello sin necesidad de instrucciones de programación; sin embargo, no significa que las computadoras tomen decisiones sin un código base. ML es la práctica de programar computadoras para que aprendan de los datos, o experiencia pasada [37][38][39]. La Figura 19 indica el proceso de ML que tiene como entrada datos y comandos como instrucciones.

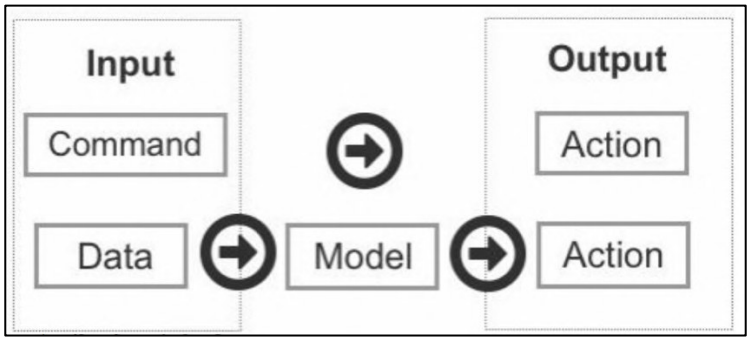


Figura 19. *Proceso básico de un sistema de Machine Learning.*

El modelo puede ser predictivo para hacer pronósticos en el futuro, o descriptivo para obtener conocimientos a partir de los datos, o ambos. El aprendizaje automático utiliza la teoría de la estadística en la construcción de modelos matemáticos, ya que la tarea principal es hacer inferencias a partir de una muestra[40].

1.4.2.2.1. Aprendizaje Supervisado

El aprendizaje supervisado se concentra en aprender patrones a partir de conectar las relaciones entre variables y salidas conocidas. Este tipo de aprendizaje trabaja alimentando la data con varias características o variables (representadas con "X") y el valor correcto de la salida (representada por "Y"), el hecho que las salidas y el valor de las variables son conocidas califican este conjunto de datos como “etiquetado”. Posteriormente, un algoritmo descifra patrones existentes en la data y crea un modelo que puede reproducir las mismas reglas subyacentes con nueva data[38][37]. En la Figura 20 se muestra un conjunto de datos con la solución deseada, denominada “etiqueta” o label.

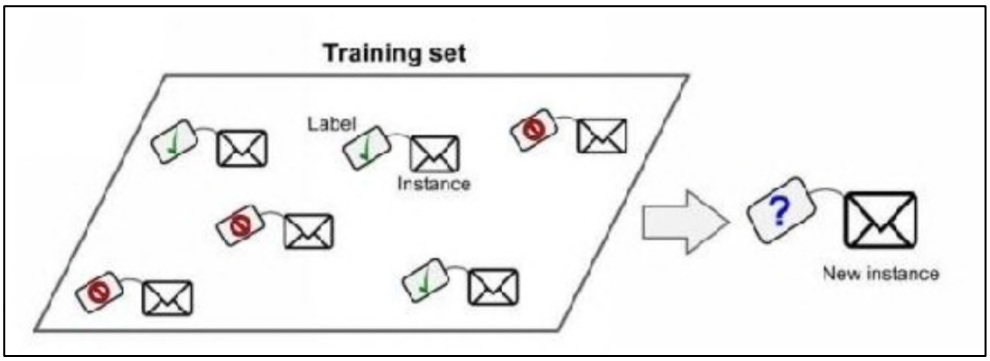


Figura 20. *Ejemplo de etiquetado en un conjunto de datos.*

Por ejemplo, si se desea predecir los valores del mercado para la compra de automóviles usados, un algoritmo de aprendizaje supervisado puede formular predicciones analizando la relación entre los atributos del automóvil (año de fabricación, marca, kilometraje, etc.) con el precio de venta de otros automóviles vendidos según datos históricos. Dado que el algoritmo supervisado conoce el precio final de otros autos vendidos, puede determinar retrospectivamente la relación entre características de un auto y su valor. La Figura 21 ilustra el ejemplo gráficamente.

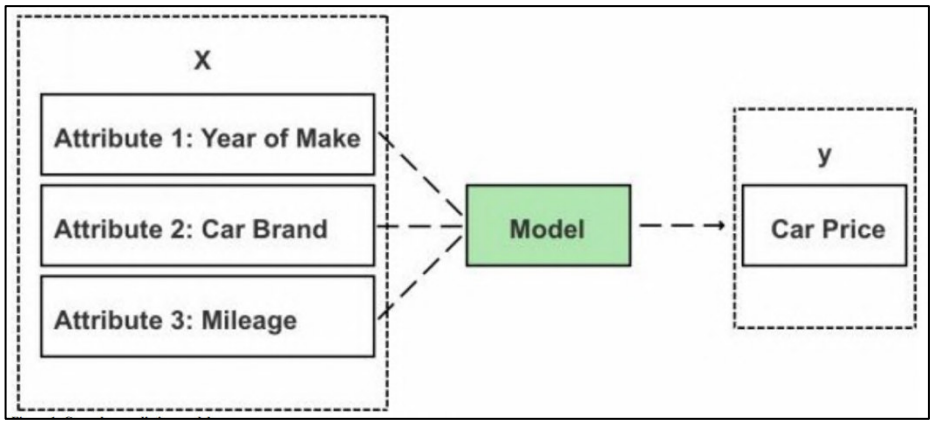


Figura 21. *Representación del sistema de predicción de precios de autos usados basado en aprendizaje supervisado.*

Una vez que el computador descifra las reglas y patrones de los datos, crea lo que se conoce como modelo: una ecuación algorítmica con el fin de producir un resultado con nuevos datos, basándose en las reglas derivadas de los datos de entrenamiento. Una vez que se prepara el modelo, se puede aplicar a nuevos datos y probar su precisión. Después de la modelo ha pasado las etapas de datos de prueba y entrenamiento, está listo para ser aplicado y utilizado en el mundo real.

A continuación, se lista algoritmos de aprendizaje supervisado:

- Análisis de regresión
- Árboles de decisión,
- k-nearest neightbors
- Redes neuronales y
- Support vector machine

1.4.2.2.2. Aprendizaje no Supervisado

Aprendizaje no supervisado se refiere a la capacidad de aprender y organizar la información sin proporcionar una señal de error para evaluar la solución potencial. A diferencia del aprendizaje supervisado, en este tipo de aprendizaje no todas las variables y patrones de datos son clasificados. En su lugar, la computadora debe agrupar la información no clasificada en función de similitudes, patrones y diferencias sin ningún tipo de entrenamiento previo de los datos, es decir el computador trata de aprender sin un instructor[39][41].

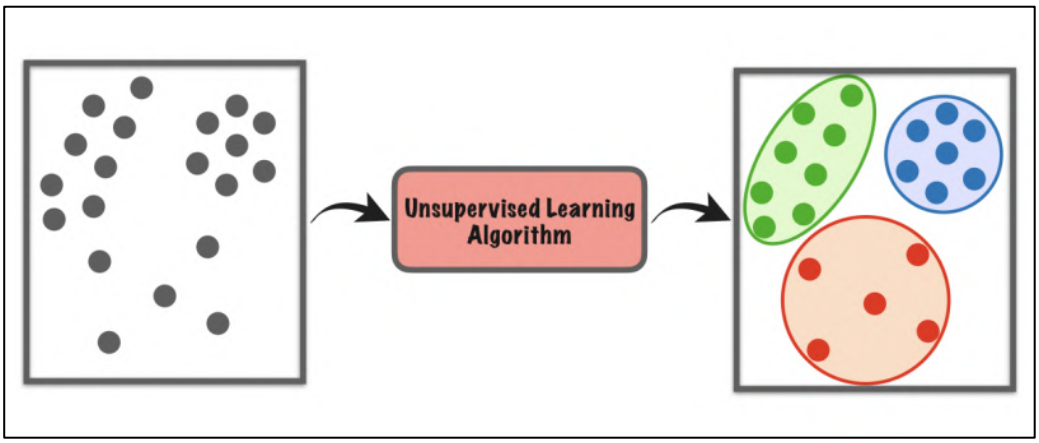


Figura 22. *Aplicación de algoritmo de aprendizaje no supervisado.*

En todos estos casos, se desea aprender la estructura inherente de los datos sin utilizar etiquetas proporcionadas explícitamente. El aprendizaje no supervisado utiliza algoritmos de ML para analizar y agrupar conjuntos de datos no etiquetados. Dado que no se proporcionan etiquetas, no hay una forma específica de comparar el rendimiento del modelo en la mayoría de los métodos de aprendizaje no supervisado. A pesar de ello, la falta de dirección del algoritmo de aprendizaje puede ser a veces ventajosa, ya que permite buscar patrones que no han sido considerados previamente[35][42][43].

Los algoritmos de aprendizaje no supervisado se utilizan para tres tareas principales: agrupación, asociación y reducción de la dimensionalidad. A continuación, se lista algoritmos de aprendizaje no supervisado[38][42]:

- **Clustering:** Permite agrupar datos no etiquetados en función de sus similitudes o diferencias, dentro de este grupo destacan los siguientes algoritmos, la Figura 22 ilustra este tipo de algoritmo:
    - k-means
    - Análisis cluster jerárquico (HCA)
    - Algoritmo esperanza-maximización

- **Aprendizaje de reglas de asociación:** Utiliza diferentes reglas en base a medidas de interés para encontrar relaciones entre variables en un conjunto de datos determinado, dentro de este grupo destacan los siguientes algoritmos:
    - Eclat
    - A priori

- **Visualización y reducción de dimensionalidad:** Se utiliza cuando el número de características (o dimensiones) de un conjunto de datos es demasiado elevado, reduciendo el número de entradas de datos a un tamaño manejable al tiempo que preserva la integridad de los datos, dentro de este grupo destacan los siguientes algoritmos:
    - Análisis de componentes principales PCA
    - Kernel PCA
    - Locally-Linear Embedding
    - t-distributed Stochastic Neighbor Embedding

1.4.2.2.3. Deep Learning

Para entender el contexto de Deep Learning (DL) o aprendizaje profundo, es necesario entender como se relacionan entre sí, IA, ML y DL. La Figura 23 muestra como se relacionan entre sí. DL es un subcampo específico de ML, en el cual se utiliza una nueva forma de aprender representaciones abstraídas del mundo real a partir de datos que hace énfasis en el aprendizaje de capas sucesivas de representaciones cada vez más significativas. Una representación es una forma diferente de ver los datos, de expresarlos o codificarlos. Por ejemplo, una imagen en color puede codificarse en el formato RGB (rojo-

verde-azul) o en el formato HSV (tono-saturación-valor): son dos representaciones diferentes de los mismos datos. En definitiva, el aprendizaje profundo es un marco matemático para aprender representaciones a partir de datos[44].

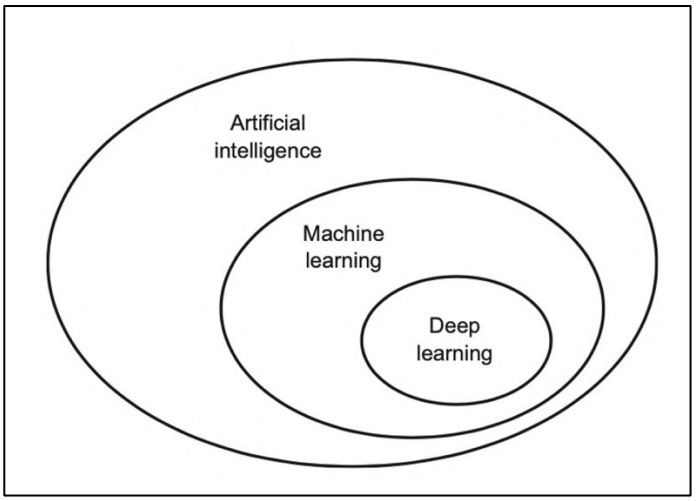


Figura 23. *Inteligencia artificial, machine learning y deep Learning.*

Lo profundo en DL no hace referencia a ningún tipo de comprensión más profunda lograda por el enfoque; más bien, representa la idea de capas sucesivas de representaciones. El número de capas que contribuyen a un modelo de datos se denomina profundidad del modelo. Hoy en día DL a menudo implica decenas o incluso cientos de capas sucesivas de representaciones, y todas ellas se aprenden automáticamente a partir de la exposición a los datos de entrenamiento. En DL, estas representaciones en capas se aprenden (casi siempre) a través de modelos llamados redes neuronales revisadas en la sección δ1.4.2.3, estructurados en capas verticales apiladas unas sobre otras[44][45][46].

- **Representaciones en DL**

Con el objetivo de exponer como son las representaciones aprendidas por un algoritmo de DL, se explica como una red de varias capas de profundidad (véase la Figura 24) transforma una imagen de un dígito hasta reconocer de qué dígito se trata[44].

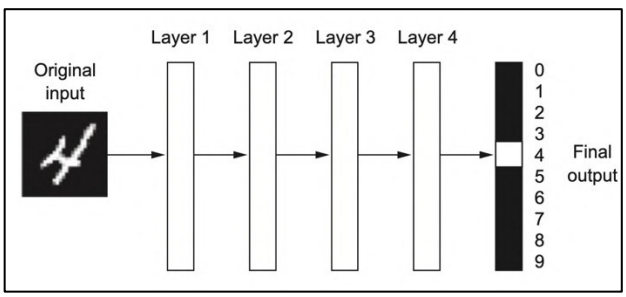


Figura 24. *Red Neuronal profunda para clasificación de dígitos.*

Como se puede ver en la Figura 25, la red transforma la imagen del dígito en representaciones que son cada vez más diferentes de la imagen original y cada vez más informativas sobre el resultado final. Una red profunda es una operación de destilación de información en varias etapas, en la que la información pasa por filtros sucesivos y surge cada vez más purificada (es decir, útil con respecto a alguna tarea). DL es una forma de aprender representaciones de datos en varias etapas[44].

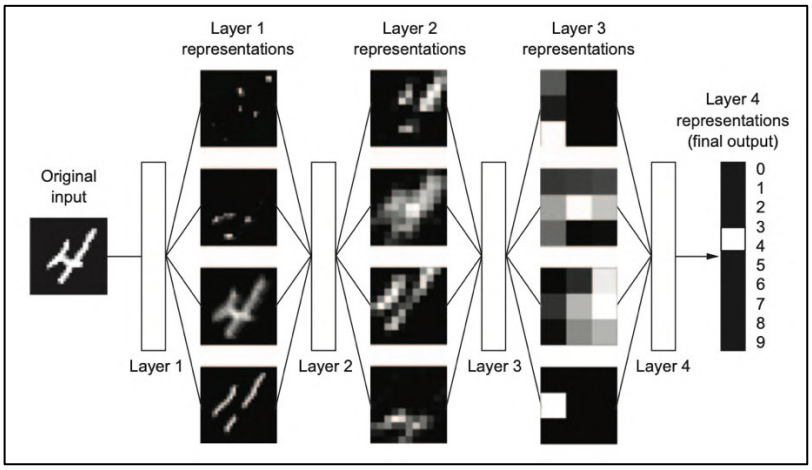


Figura 25. *Representaciones profundas aprendidas por un modelo de clasificación de dígitos.*

- **Estructura básica del modelo de DL**

La estructura de los modelos de aprendizaje profundo suele ser tal que tienen capas de unidades no lineales que procesan datos, o neuronas, y las múltiples capas de estos modelos procesan diferentes niveles de abstracción de los datos. La Figura 26 muestra una visualización de las capas de las redes neuronales[45].

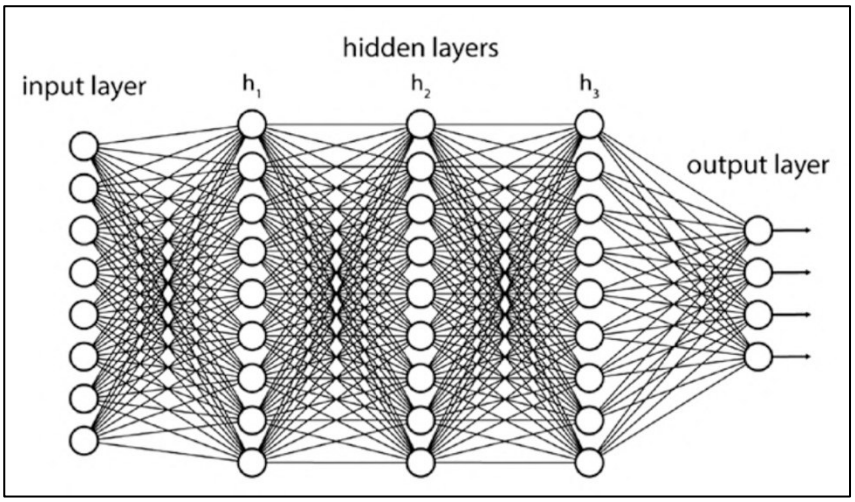


Figura 26. *Red neuronal profunda.*

Las redes neuronales profundas se distinguen por tener muchas capas ocultas, que se denominan "ocultas" porque no vemos necesariamente cuáles son las entradas y salidas de estas neuronas de forma explícita, más allá de saber que son la salida de la capa anterior. Más concretamente, los niveles inferiores de estos modelos explican el "cómo", y los niveles superiores de las redes neuronales procesan el "por qué"[45][46].

Tener múltiples capas permite al modelo procesar los datos de tal manera que construye una comprensión desde aspectos simples hasta construcciones más amplias. El objetivo de estos modelos es realizar tareas sin el mismo grado de instrucción explícita que necesitan muchos algoritmos de ML[45].

1.4.2.3. Redes Neuronales Artificiales

Las redes neuronales artificiales o ANN, también conocidas como redes neurales, son una técnica de ML para procesar datos mediante capas de análisis. El nombre de las redes neuronales artificiales se inspiró en el parecido del algoritmo con el cerebro humano. Una red neuronal está formada por unidades de procesamiento simples que tiene una propensión natural a almacenar el conocimiento de la experiencia y ponerlo a disposición para su uso. ANN se asemeja al cerebro en dos aspectos[47]:

- El conocimiento es adquirido por la red a partir de su entorno mediante un proceso de aprendizaje.

- La fuerza de las conexiones entre las neuronas, conocida como peso de las sinapsis, se utiliza para almacenar los conocimientos adquiridos.

Una neurona artificial es un modelo computacional inspirado en las neuronas naturales. Las neuronas naturales reciben señales o entradas a través de sinapsis situadas en las dendritas o la membrana de la neurona. Cuando las señales recibidas son lo suficientemente fuertes (superan un determinado umbral), la neurona se activa y emite una señal a través del axón. Esta señal puede ser enviada a otra sinapsis y puede activar otras neuronas[37][48], véase Figura 27. Al igual que las neuronas del cerebro humano, las ANN están formadas por neuronas interconectadas, también llamados nodos que interactúan entre sí a través de los axones llamados bordes[49].

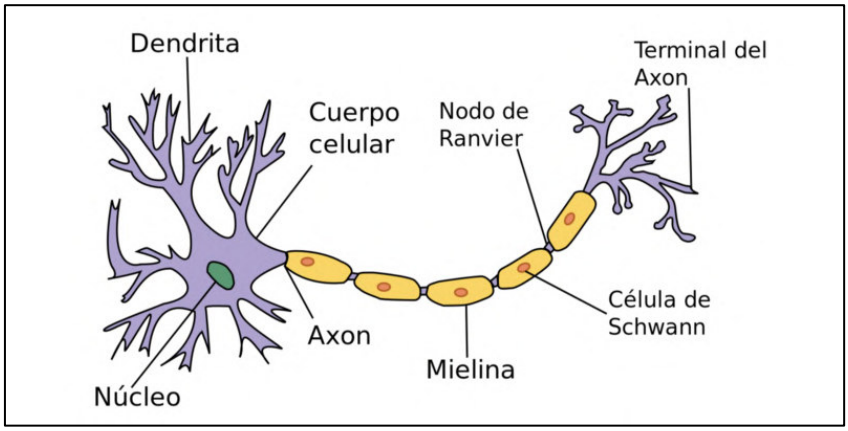


Figura 27. *Neurona biológica.*

En una red neuronal, los nodos se apilan en capas, empezando con una capa amplia. La primera capa está formada por datos brutos, como valores numéricos, texto, imágenes o sonido, que se dividen en nodos. Cada nodo envía información a la siguiente capa atreves de los bordes de la red[37].

Cada borde tiene un peso numérico, obtenido a partir de un algoritmo, el cual se puede modificar y es formulado en base a la experiencia. Si la suma de los bordes conectados satisface un umbral establecido, conocido como función de activación, se activará una neurona en la siguiente capa. Sin embargo, si la suma de los bordes conectado no llega al umbral, la función de activación no se encenderá. Los pesos a lo largo de cada arista son únicos para garantizar que los nodos se disparen de forma diferente y no devuelvan todo el mismo resultado[47][50].

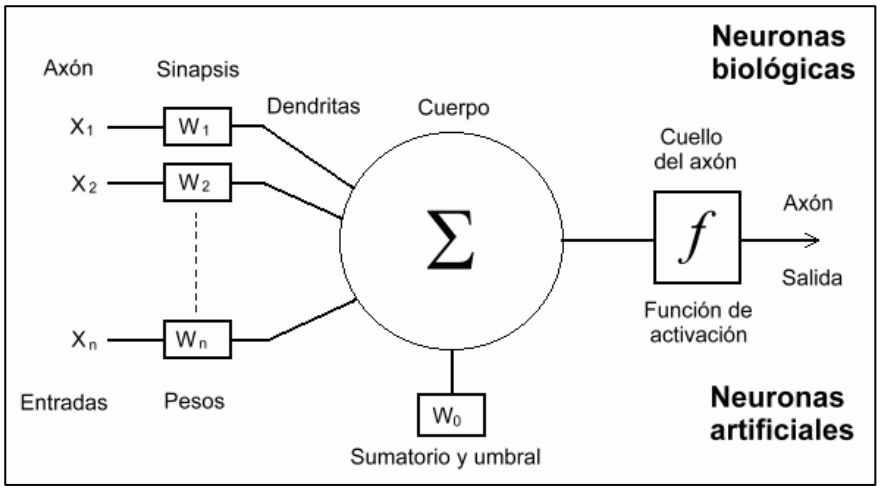


Figura 28. *Red neuronal artificial.*

El procedimiento utilizado para llevar a cabo el proceso de aprendizaje se denomina algoritmo de aprendizaje, cuya función es modificar los pesos de las sinapsis de la red de forma ordenada para alcanzar un objetivo de diseño deseado. La modificación de los pesos de las sinapsis constituye el método tradicional para el descompilado de redes neuronales. Este enfoque es el más cercano a la teoría del filtro adaptativo lineal, que ya está bien establecido y se aplica con éxito en muchos campos diversos[51][52]. Sin embargo, también es posible que una red neuronal modifique su propia topología, lo que está motivado por el hecho de que las neuronas del cerebro humano pueden morir y crecer nuevas conexiones sinápticas[47]. La Figura 28 ilustra una red neuronal artificial.

- **Neuronas como funciones**

Las neuronas se comportan como funciones. Las neuronas transducen una activación de entrada no limitada $x(t)$ en un tiempo t en una señal de salida limitada $f(x(t))$. Regularmente, una curva sigmoidal o en forma de S, como se puede ver en la Figura 29, describe la transducción. Esta función (f) se llama función de activación o de señal.

La función más utilizada es la función de señal logística:

$$f(a) = \frac{1}{1 + e^{-ca}}$$

Ecuación 9. *Ecuación de la función de señal logística.*

La cual es sigmoidal y aumenta estrictamente para una constante de escala positiva $c > 0$. La monotonicidad estricta implica que la derivada de activación de f es positiva:

$$f' = \frac{df}{da} = cf(1-f) > 0$$

Ecuación 10. *Derivada de función de f.*

La función de señal del umbral, representada con línea discontinua en la Figura 29, ilustra una función de señal no diferenciable. La familia de la función de señal logística, indexada por c, se aproxima asintóticamente a la función umbral cuando $c \to +\infty$. Entonces f transduce las señales de activaciones positivas a señales de unidad y las activaciones negativas a señales de cero.

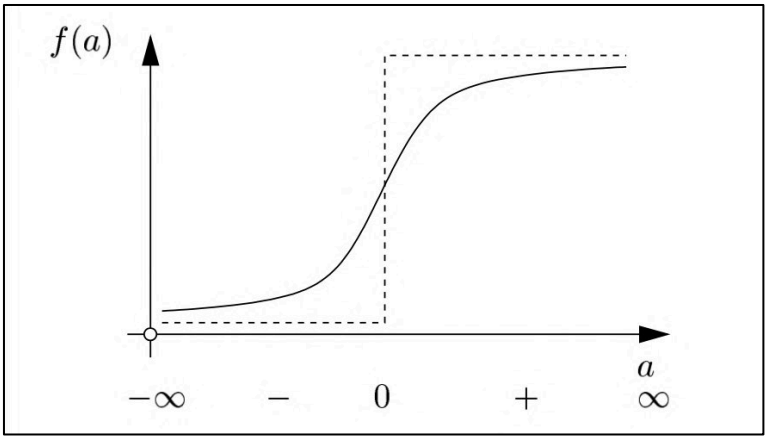


Figura 29. *La señal $f(a)$ como función acotada monótona-no decreciente de la activación a. La curva discontinua define el umbral de la función de señal umbral.*

- **Funciones de activación comunes**

Existen varios tipos de fórmulas funcionales diferentes (ver Tabla 1), adicionales a la función sigmoidal, que afectan a la unión de las neuronas (lineal, sinusoidal, gaussiana, etc.). Se puede definir una función de activación diferente para cada capa.

Función	Fórmula
Linear	$f(x) = x$ *Ecuación **11**. Función de activación lineal.*

Logística simétrica	$f(x) = \frac{(1+e^{-x})}{2}$ ***Ecuación 12.*** *Función de activación logística simétrica.*
Tangente hiperbólica	$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$ ***Ecuación 13.*** *Función de activación hiperbólica.*
Tangente corregida	$f(x) = tanh\,(c \cdot x)$ ***Ecuación 14.*** *Función de activación tangente corregida.*
Sinusoidal	$f(x) = sin\,(x)$ ***Ecuación 15.*** *Función de activación sinusoidal.*
Gaussiana	$f(x) = e^{-x^2}$ ***Ecuación 16****. Función de activación gaussiana.*
Gaussiana invertida	$f(x) = 1 - e^{-x^2}$ ***Ecuación 17.*** *Función de activación gaussiana invertida.*

Tabla 1. *Funciones de activación.*

No existe una regla para definir la función de activación ideal para cada una de las posibles distintas capas. Aunque en algunos casos se presentan diferentes funciones para las capas individuales, al contrario, de mantener una misma función para las capas de entrada, oculta y de salida. La función lineal se suele utilizar para capas de salida. Sin embargo, su uso es menos eficaz en las capas ocultas, especialmente si éstas se caracterizan por un elevado número de neuronas[53].

Las funciones logísticas y logísticas simétricas tienen la característica de variar, respectivamente, en los intervalos [0, 1] y [-1, 1]. Algunos problemas tienen características dinámicas que son captadas en una medida más precisa por la función simétrica, especialmente en las capas de entrada y ocultas. La mayor parte de la literatura empírica muestra el uso de esta función en la capa oculta, aunque sin una fuerte motivación teórica.

La función tangente hiperbólica permite adaptar la red de forma fiable en las capas ocultas (sobre todo en las redes de tres capas). La función sinusoidal se utiliza generalmente en la investigación y se sugiere normalizar la entrada y la salida en el rango [-1, 1]. La función gaussiana se presta a la identificación de procesos dinámicos específicos captados con arquitecturas de dos capas ocultas paralelas, con una función tangente en la segunda capa[53].

1.4.2.3.1. Estructura y Arquitectura de ANNs

La estructura básica de una neurona puede modelarse teóricamente como se muestra en la Figura 1, donde $X\{xi, i = 1,2,\cdots,n\}$ representa las entradas a la neurona e Y representa la salida. Cada entrada se multiplica por su peso w_i, se asocia un sesgo b a cada neurona y su suma pasa por una función de activación f. La Ecuación 18, describe la relación entre las entradas y salida de una red neuronal[54]. En general, las ANN se construyen colocando las neuronas en capas y conectando las salidas de las neuronas de una capa con las entradas de las neuronas de la capa siguiente[55], como se muestra en la Figura 30.

$$Y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Ecuación 18. *Ecuación de una red neuronal.*

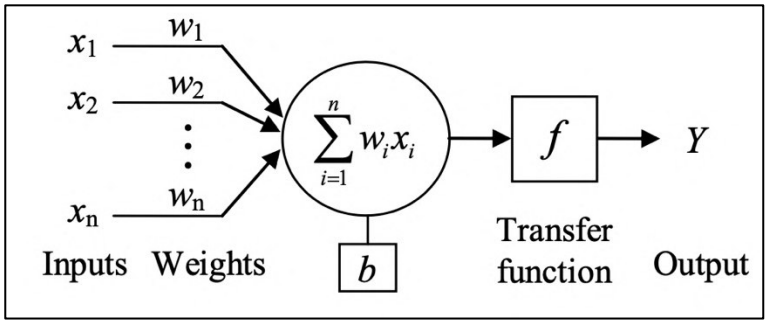


Figura 30. *Relación entre entradas y salidas de una red neuronal.*

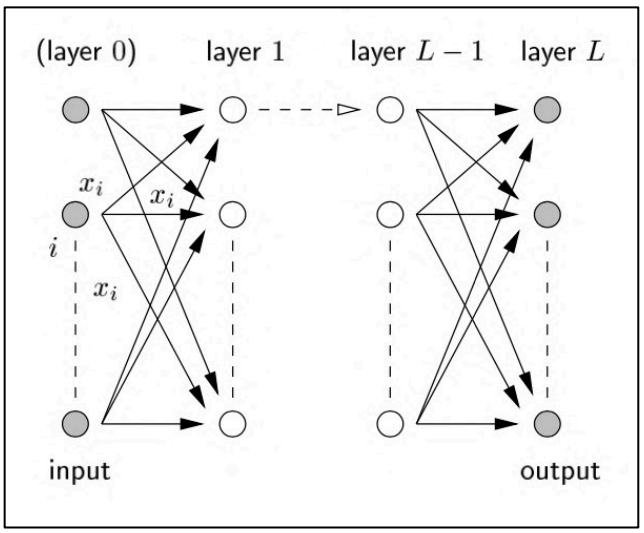


Figura 31. *Estructura básica de una red neuronal feedforward. La capa 0 distribuye la entrada a la capa de entrada 1. La salida de la red es (generalmente) la salida de la capa de salida L (última capa).*

- **Arquitecturas de ANNs**

Las arquitecturas de ANN pueden verse como grafos dirigidos ponderados en los que las neuronas artificiales son nodos y las aristas dirigidas (con pesos) son conexiones entre las salidas y las entradas de las neuronas. En función del patrón de conexión (arquitectura), las ANNs pueden agruparse en dos categorías[56]. La Figura 32 presenta los 2 tipos de arquitecturas:

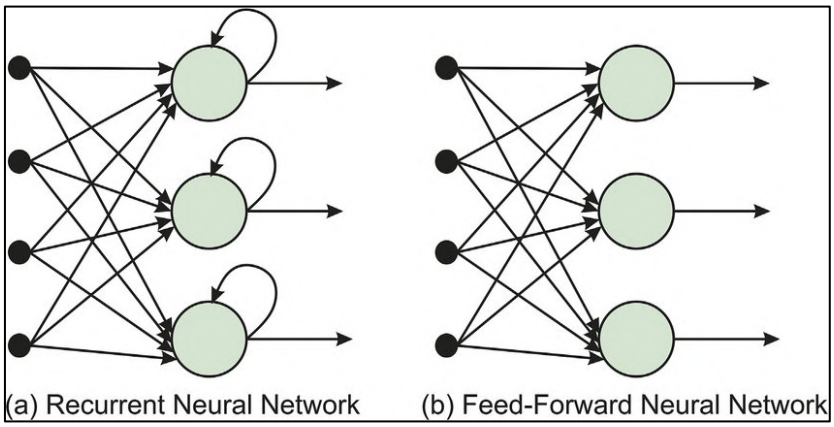


Figura 32. *Tipos de arquitecturas de una red neuronal artificial.*

- **Redes feed-forward**, en las que los gráficos no tienen bucles. Por ejemplo: Single-layer perceptron, Multilayer perceptron y Radial Basis Functions Nets.
- **Redes recurrentes** (o de retroalimentación), en las que los bucles se producen debido a las conexiones de retroalimentación. Por ejemplo: Competitive networks, Kohonen's SOM, Hopfield network y ART models.

A continuación, se presenta la Figura 33, en donde se ilustra gráficamente la arquitectura de los ejemplos de cada tipo de arquitectura de ANNs[56].

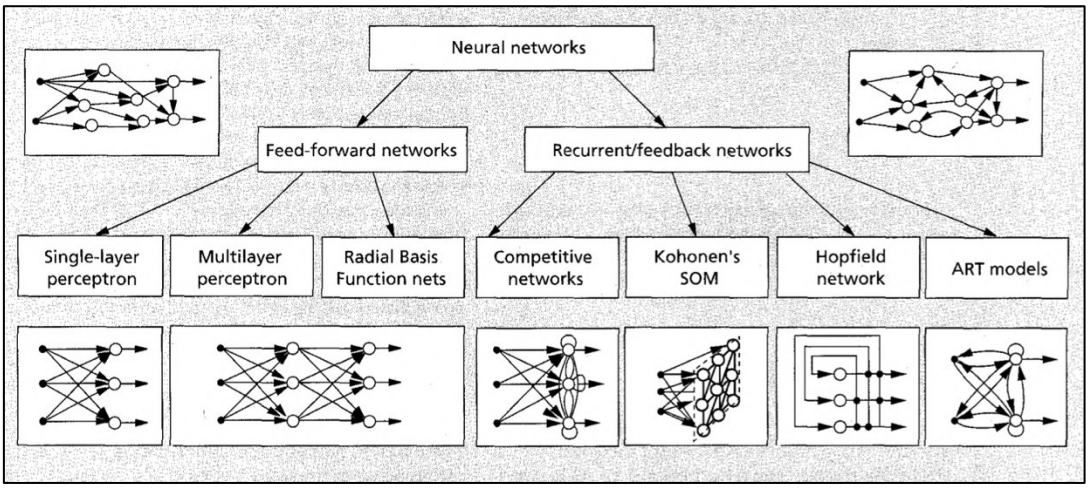


Figura 33. Diferentes topologías de las arquitecturas de redes feed-forward y recurrentes.

1.4.2.3.2. Problemáticas a resolver y consideraciones

- **Problemas de regresión:** son aquellos en los que se intenta predecir un valor numérico en respuesta al vector de entrada dado (como predecir la edad de una persona a partir de su peso, altura y sexo). Como se trata de predecir un solo número, la capa de salida sólo contiene una neurona. La función de activación de la identidad de la salida se suele utilizar para la neurona de la capa de salida y el cuadrado de la diferencia entre la salida de la red y la salida deseada se suele utilizar para la función de error[57].

- **Problemas de clasificación:** son aquellos en los que se intenta predecir si un vector de entrada dado pertenece a una de varias clases (como predecir la especie de una planta dadas varias de sus medidas). A diferencia de los problemas de regresión, en los que hay una sola neurona en la capa de salida, ésta suele tener una neurona por cada clase posible. La función de activación softmax, en combinación con la función de error de entropía cruzada, suele utilizarse para las neuronas de la capa de salida, ya que el valor de salida puede interpretarse como la probabilidad de que un vector de entrada determinado pertenezca a la clase representada por cada neurona de salida[57].

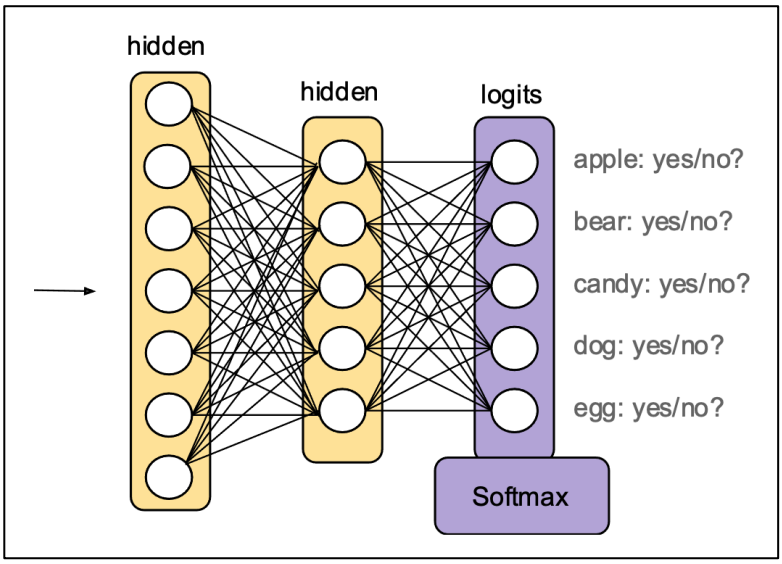


Figura 34. *Problema de clasificación con ANN.*

Algunas consideraciones clave para trabajar con redes neuronales son:

- **Valores iniciales.** Cuando los pesos en una red neuronal están cerca de cero, el modelo se aproxima a un modelo de regresión lineal. Luego de muchas iteraciones, el modelo de la red neuronal se vuelve cada vez más no lineal a medida que los pesos cambian y ayudan a definir las relaciones no lineales no observadas[57]. Si los valores iniciales están demasiado cerca de cero, el algoritmo no converge a los valores reales. Del mismo modo, empezar con pesos grandes suele conducir a soluciones pobres.
- **Overfitting.** El exceso de ajuste denota que el modelo ha sido entrenado para ajustarse a un conjunto específico de observaciones en el entrenamiento, pero no predice bien los resultados de las observaciones no utilizadas para el entrenamiento. A diferencia de la regresión lineal o logística, en las que se utilizan datos históricos para calcular sólo unos pocos parámetros del modelo, con redes neuronales, dependiendo del número de variables de entrada y del número de capas ocultas, se pueden ajustar los datos pasados con cientos o incluso miles de parámetros del modelo (pesos)[57]. Dado que las redes neuronales son aproximadores universales, si se añaden suficientes redes ocultas y se ejecuta durante un número suficiente de iteraciones, serán capaces de predecir perfectamente los resultados de las observaciones de entrenamiento.

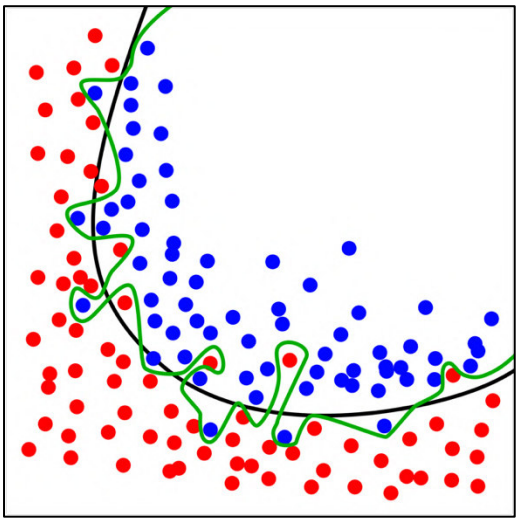


Figura 35. *Representación gráfica de overfitting.*

- **Escalamiento en entradas.** La escala de las variables de entrada afecta a los pesos iniciales. Es una buena práctica escalar las entradas a una media de 0 y una desviación estándar de 1 antes de empezar a construir su modelo. Esto garantizará que todas las variables de entrada reciban un tratamiento justo y equitativo[57].
- **Aseguramiento del mínimo global.** Debido a que el objetivo es minimizar el error es no convexo (lo que significa que, en el hiperespacio, cualquier línea entre dos puntos no es siempre una solución válida; es posible quedarse atascado en un mínimo local y nunca alcanzar el mínimo global. La capacidad de encontrar el mínimo global se basa en gran medida en los valores de partida[57].

1.4.2.3.3. Redes Feed-Fordward Multicapa

Se trata de una subclase de redes no cíclicas en las que se permite una conexión desde un nodo de la capa actual sólo a los nodos de la capa actual + 1, como se muestra en la Figura 36. Estas redes se describen abreviadamente mediante una secuencia de números que indican el número de nodos en cada capa[49]. Por ejemplo, la red mostrada en la Figura 36 es una red feed-forward 3-5-5-3; contiene tres nodos en la capa de entrada (capa 1), cinco nodos en la primera capa oculta (capa 2), cinco nodos en la segunda capa oculta (capa 3) y tres nodos en la capa de salida (capa 4).

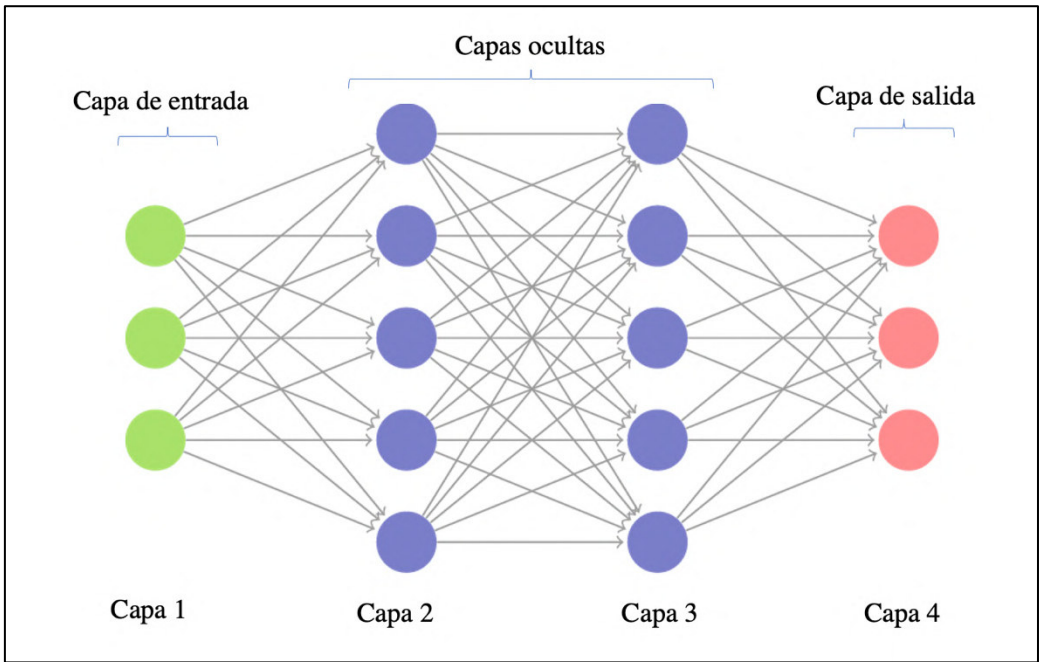


Figura 36. *Red neuronal feed forward.*

Estas redes, generalmente con no más de cuatro capas, se encuentran entre las redes neuronales más comunes en uso, hasta el punto de que se identifican la frase "redes neuronales" para referirse únicamente a las redes feed-forward. Conceptualmente, los nodos de las capas sucesivamente superiores extraen características de nivel sucesivamente superior de las capas precedentes[49].

- **Perceptrón multicapa**

Dentro de las clases de redes feed-forward multicapa, el perceptrón es el más general u utilizado de todos. El autor Rosenblatt [56] define un perceptrón como una máquina que aprende, mediante ejemplos, asignando vectores de entrada (muestras) a diferentes clases, utilizando una función lineal de las entradas.

Los siguientes tres puntos destacan las características básicas de los perceptrones multicapa:

- El modelo de cada neurona de la red incluye una función de activación no lineal que es diferenciable.
- La red contiene una o más capas que están ocultas tanto para los nodos de entrada como de salida.

- La red presenta un alto grado de conectividad, cuya extensión está determinada por los pesos sinápticos de la red.

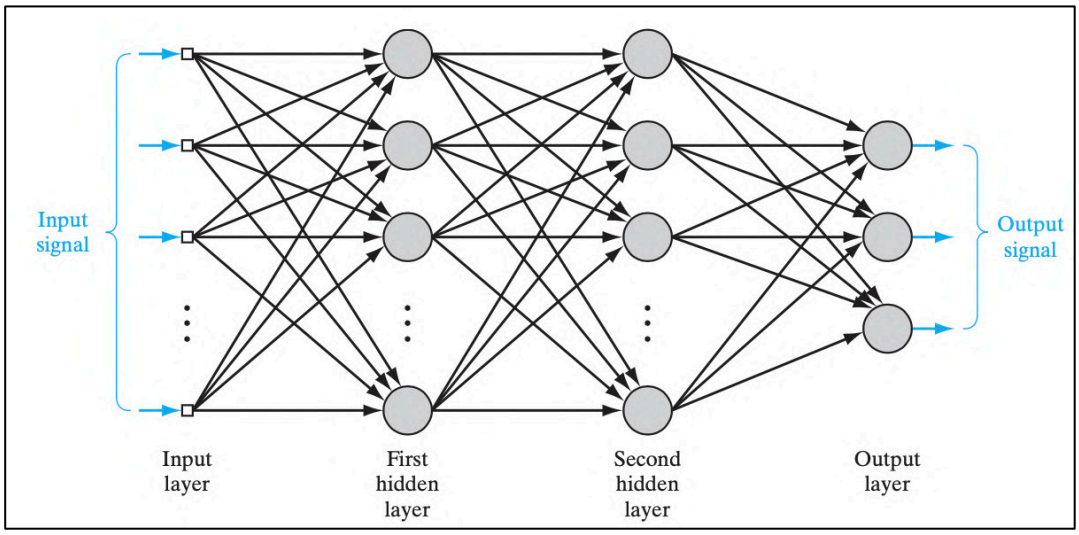


Figura 37. *Arquitectura perceptron multicapa.*

La Figura 37 muestra una parte del perceptrón multicapa, en el cual pueden existir dos tipos de señales identificadas en esta red esta red[47], las señales de activación descritas en la sección ∂1.4.2.3 y las señales de error, la cual se origina en una neurona de salida de la red y se propaga hacia atrás (capa por capa) a través de la red. Nos referimos a ella como una "señal de error" porque su cálculo por cada neurona de la red implica una función dependiente del error de una forma u otra (véase Figura 38).

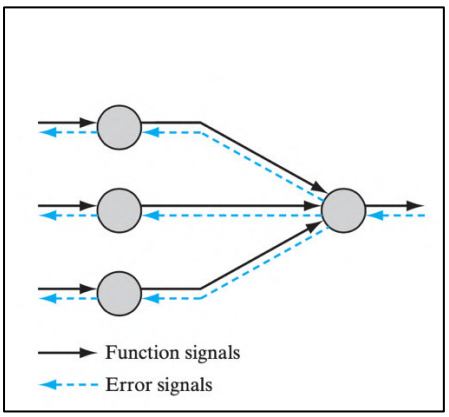


Figura 38. *Tipos de señales de un perceptron multicapa.*

El poder del perceptrón multicapa proviene precisamente de funciones de activación no lineales. Casi cualquier función no lineal se puede utilizar para este propósito, excepto las funciones polinomiales. Actualmente, las funciones más utilizadas en la actualidad son la

sigmoide unipolar (o logístico), que se formula en la Tabla 1 descrita en la sección ∂1.4.2.3 y se muestra en la Figura 39, y la sigmoide bipolar, que se formula a continuación:

$$f(s) = \frac{1 + e^{-a.s}}{1 + e^{-a.s}}$$

Ecuación 19. *Función de activación sigmoide bipolar.*

Cabe señalar que las funciones sigmoideas actúan aproximadamente lineales para valores absolutos pequeños y están saturados, asumiendo de alguna manera el papel de umbral para valores absolutos altos del argumento. Se ha demostrado [4] que una red (posiblemente infinita) con una capa oculta es capaz de aproximarse a cualquier función continua.

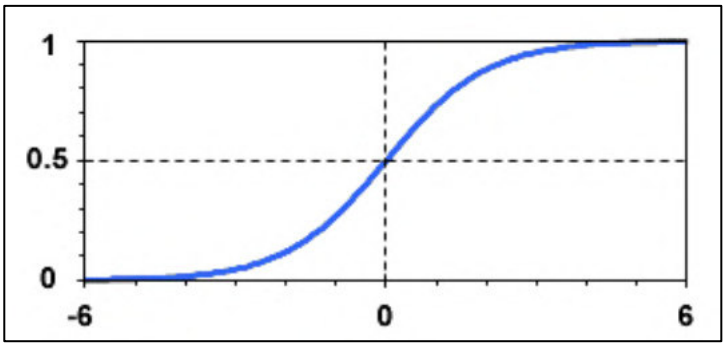


Figura 39*. Función de activación unipolar sigmoidea.*

- **Algoritmo back propagation**

Este algoritmo se utiliza en ANNs multicapas, por lo que las neuronas artificiales se organizan en capas y envían sus señales "hacia adelante", y luego los errores se propagan hacia atrás. El algoritmo de retropropagación modifica los pesos de la red para minimizar el error cuadrático medio entre las salidas deseadas y reales de la red. La retropropagación utiliza un aprendizaje supervisado en el que la red se entrena utilizando datos de los que se conocen tanto las entradas como las salidas deseadas. Una vez entrenada, los pesos de la red se congelan y pueden utilizarse para calcular los valores de salida de las nuevas muestras de entrada[50][56][58] (véase Figura 40).

La idea del algoritmo es reducir este error, hasta que la ANN aprenda los datos de entrenamiento. El entrenamiento comienza con pesos aleatorios, y el objetivo es ajustarlos para que el error sea mínimo.

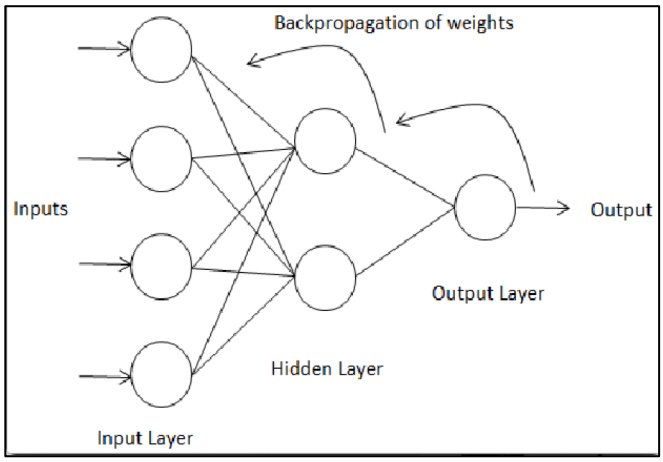


Figura 40. *Retropropagación de pesos feed forward.*

El algoritmo de retropropagación puede dividirse en dos pasos[56].

- Paso de avance: Los datos de entrada a la red se propagan al siguiente nivel y así sucesivamente.
- Paso hacia atrás: El error cometido por la red se propaga hacia atrás y los pesos se actualizan adecuadamente.

- **Entrenamiento de ANNs**

Las ANNs no se programan directamente, sino que se entrenan explícitamente a través de un algoritmo de aprendizaje para resolver una tarea determinada, un proceso que conduce al "aprendizaje a través de la experiencia". El algoritmo de aprendizaje ayuda a definir la configuración específica de una red neuronal y, por tanto, condiciona y determina la capacidad de la propia red para dar respuestas correctas a problemas concretos[56]. Existen al menos tres tipos de aprendizaje: no supervisado, supervisado y por refuerzo, dos de ellos mencionados en la sección δ1.4.2.2. El conjunto de datos utilizado para el aprendizaje de la red constituye el llamado conjunto de aprendizaje o entrenamiento.

1.4.2.4. Redes Neuronales Recurrentes

Las redes recurrentes son sistemas dinámicos con representaciones temporales de estados, desde el punto de vista computacional pueden ser utilizados en varios modelos y aplicaciones de procesamiento temporal, además son adecuadas para la implementación digital[59].

La característica fundamental de una red neuronal recurrente es que la red contiene al menos una conexón de retroalimentación, por lo que las actividades fluyen en bucle, permitiendo a las redes aprender secuencias y realizar procesamientos temporales. Una arquitectura de red recurrente puede tener diferentes formas, el tipo más común consiste en un perceptón multicapa (MLP), algunas tienen estructuras más uniformes potencialmente con cada neurona conectada a todas las demás y también pueden tener funciones de activación estocásticas[59][60].

Truncar la red desplegada a un solo paso de tiempo la reduce a una red recurrente simple, comúnmente conocida como Red Elman[59] (vease Figura 41):

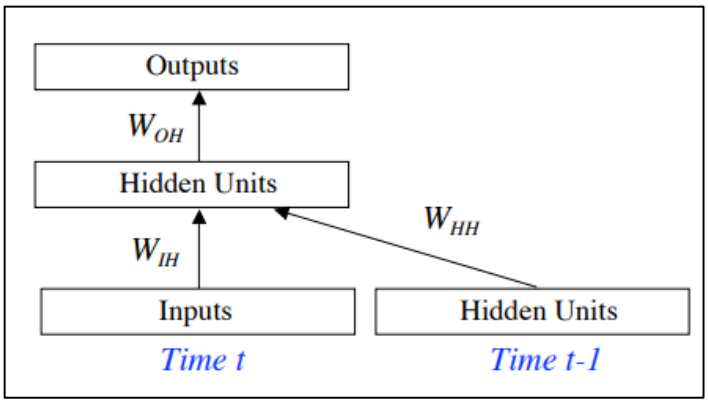


Figura 41. Red recurrente simple.

En este caso, cada conjunto de pesos aparece sólo una vez, por lo que es posible aplicar el descenso del gradiente utilizando el algoritmo estándar de retro propagación, con lo cual la señal de error no se propagará muy atrás, y será difícil que la red aprenda a utilizar la información de tiempos muy anteriores[59].

Aprovechando el poder de la computación cerebral, las redes neuronales tienen al menos una conexión de retroalimentación, también pueden utilizarse como memorias asociativas para construir atractores a partir de la asociación entrada-salida. La forma más sencilla de red neuronal totalmente recurrente es una MLP con el conjunto anterior de activaciones de las unidades ocultas que se retroalimentan en la red junto con las entradas[60] (véase Figura 43):

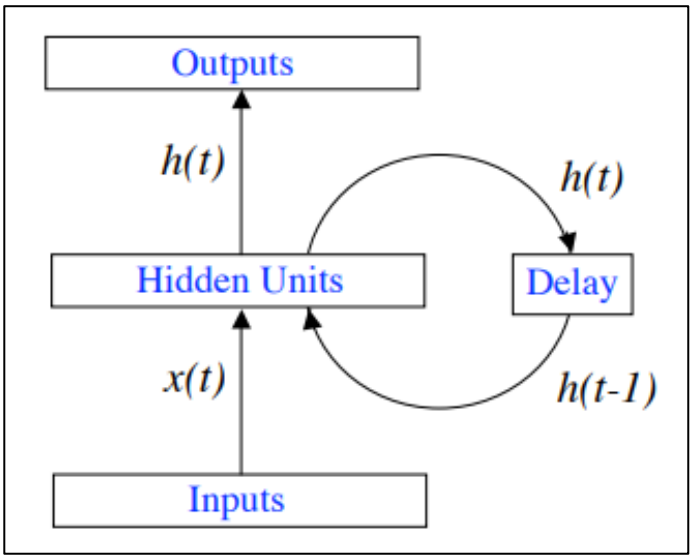


Figura 43. *Red neuronal totamente recurrente.*

Existen dos clasificaciones de redes recurrentes las redes globalmente recurrentes, la cual permite conexiones de retroalimentación entre cada neurona, y redes localmente recurrentes alimentadas con la dinámica global realizada dentro de los modelos neuronales. Ambas pueden ser aproximadores universales de sistemas dinámicos. Pero, las redes globalmente recurrentes tienen problemas de estabilidad durante el entrenamiento por lo cual requieren de un algoritmo de entrenamiento complicado, por el contrario, las redes localmente recurrentes se diseñan con modelos neuronales dinámicos que tienen una estructura menos complicada permitiendo que su entrenamiento sea más sencillo[59][60].

Dos formulaciones en tiempo discreto de redes recurrentes son la red recurrente retardada la cual se entrena para minimizar el error de predicción, por el contrario, la red recurrente simultánea no proporciona una mejor predicción en el tiempo, sino que utiliza la recurrencia para proporcionar una capacidad general de aproximación de funciones, basada en conceptos de la teoría de Turing y de la teoría de la complejidad[60].

Dentro de la clasificación de redes recurrentes existen[59]:

- Máquina de Boltzmann.
- Redes de Hopfield.
- Máquina de Boltzmann restringido.

1.4.2.5. Matriz de confusión

La matriz de confusión contiene información sobre clasificaciones reales y previstas realizadas por un sistema de clasificación, el rendimiento de estos sistemas se evalúa utilizando los datos de la matriz[61]. En la siguiente tabla se muestra la matriz de confusión para un clasificador de dos clases:

En la Tabla X se indica el significado de las entradas de la matriz de confusión:

Predecido			
Negativo	**Positivo**		
a	b	**Negativo**	**Actual**
c	d	**Positivo**	

Tabla 2. *Significado de entradas en una matriz de confusión.*

Donde:

- a es el número de predicciones correctas de que una instancia sea negativa
- b es el número de predicciones incorrectas de que una instancia sea positiva
- c es el número de predicciones incorrectas de que una instancia sea negativa
- d es el número de predicciones correctas de que una instancia sea positiva

A continuación, se detalla los términos estándar para una matriz de dos clases:

- **Exactitud (AC)** es la proporción del número total de predicciones que fueron correctas.

$$\mathrm{AC} = \frac{\mathrm{a} + \mathrm{d}}{\mathrm{a} + \mathrm{b} + \mathrm{c} + \mathrm{d}}$$

Ecuación 20. *Ecuación de exactitud del número total de predicciones correctas.*

- **Recuperación o tasa de verdaderos positivos (TP**) es la proporción de casos positivos identificados.

$$TP = \frac{d}{c + d}$$

Ecuación 21. *Ecuación de la tasa de verdaderos positivos.*

- **Tasa de falsos positivos (FP**) es la proporción de casos negativos que se clasificaron incorrectamente como positivos.

$$FP = \frac{b}{a + b}$$

Ecuación 22. Ecuación de la tasa falsos positivos.

- **Tasa de verdaderos negativos (TN**) es la proporción de casos negativos que se clasificaron como positivos.

$$TN = \frac{a}{a + b}$$

Ecuación 23. *Ecuación de la tasa de verdaderos negativos.*

- **Tasa de falsos negativos (FN)** es la proporción de casos positivos que se clasificaron incorrectamente como negativos.

$$FN = \frac{c}{c + d}$$

Ecuación 24. *Ecuación de la tasa de falsos negativos.*

- **Precisión (P)** es la proporción de casos positivos predichos que fueron correctos

$$P = \frac{d}{b + d}$$

Ecuación 25. *Ecuación de precisión.*

1.4.2.6. Cross Validation y Ajuste K-fold

La validación cruzada es una técnica tomada del análisis de regresión en estadística[62] para el entrenamiento y validación de un conjunto de datos. Primero, el conjunto de datos se divide aleatoriamente en una muestra de entrenamiento y otra de testeo. La muestra de entrenamiento se divide además en dos subconjuntos disjuntos[51]:

- Subconjunto de estimación, utilizado para seleccionar el modelo.
- Subconjunto de validación, utilizado para probar o testear modelo.

En la validación cruzada típica, los conjuntos de entrenamiento y validación deben cruzarse en rondas sucesivas de modo que cada punto de datos tenga la posibilidad de ser validado. La forma básica de validación cruzada es la validación cruzada de k veces[63].

- **K Fold Cross Validation**

En la validación cruzada de k pliegues, los datos se dividen primero en k segmentos o pliegues de tamaño igual (o casi igual). Posteriormente, se realizan k iteraciones de entrenamiento y validación de modo que dentro de cada iteración se mantiene un pliegue diferente de los datos para su validación, mientras que los k 1 pliegues restantes se utilizan para el aprendizaje. Los datos se estratifican comúnmente antes de dividirse en k pliegues. La estratificación es el proceso de reorganizar los datos para garantizar que cada pliegue sea un buen representante del todo[63]. La Figura 44 muestra un ejemplo con k = 3. La sección más oscura de los datos se usa para el entrenamiento, mientras que las secciones más claras se usan para la validación. En la minería de datos y el aprendizaje automático, la validación cruzada de 10 veces (k = 10) es la más común.

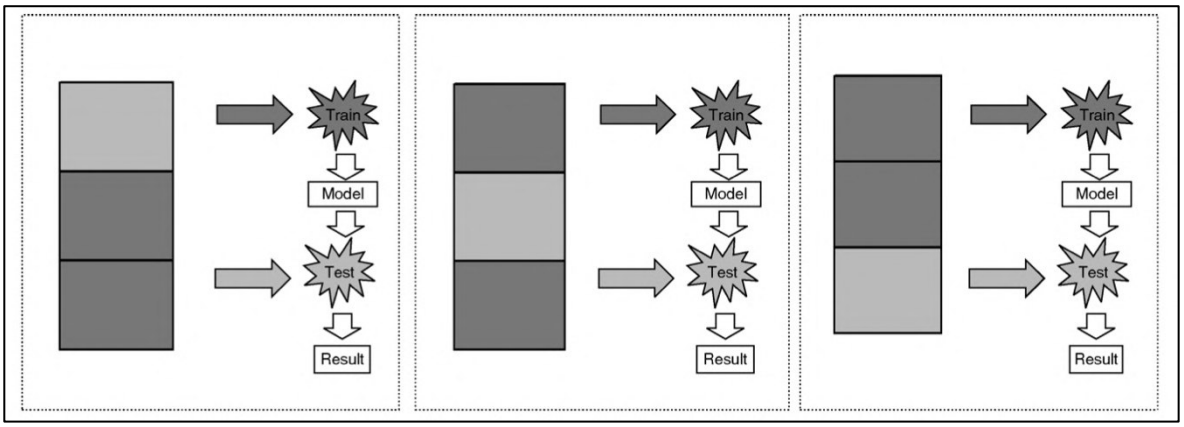


***Figura 44**. Procedimiento de cross-validation.*

1.4.2.7. t-SNE (T-distributed Stochastic Neighbor Embedding)

Es un algoritmo diseñado para la visualización de conjunto de datos de alta dimensionalidad si el número es muy alto Scikit-Learn, recomienda utilizar previamente un método de reducción de dimensiones para reducir el conjunto de datos a un número de dimensiones razonables lo que permitirá reducir el ruido aligerando la ejecución de t-SNE[64][65].

t-SNE se ejecuta en dos pasos: primero se construye una distribución de probabilidad sobre parejas de muestras en el espacio original, tal que las muestras semejantes reciben alta probabilidad de ser escogidas, mientras que las muestras muy diferentes reciben baja probabilidad de ser escogidas. Segundo t-SNE lleva los puntos de alta dimensionalidad al espacio de baja dimensionalidad de manera aleatoria, definiendo una distribución de probabilidad semejante al espacio de baja dimensionalidad lo cual permite minimizar la divergencia Kullback-Leibler (divergencia que mide la similitud o diferencia entre dos funciones de distribución de probabilidad) [64][65]. La Figura 45 muestra un ejemplo de utilización de este algoritmo.

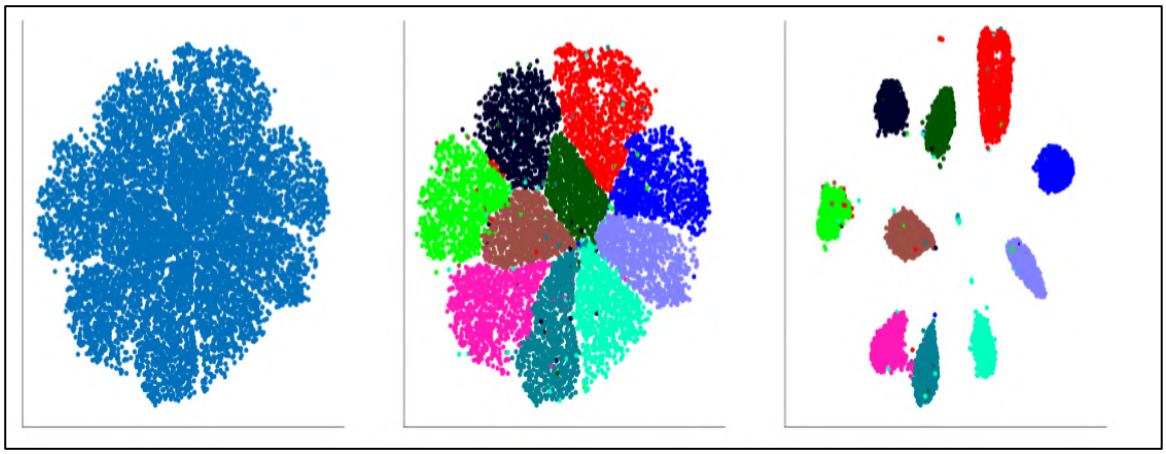


Figura 45. *Ejemplo gráfico de la aplicación del algoritmo t-SNE en el reconocimiento y clasificación de dígitos del dataset MNIST.*

1.4.2.8. Factorización de la Trayectoria Espacial

En esta sección se describe los conceptos teóricos de la Factorización de la trayectoria espacial aplicada al reconocimiento de señales. La motivación para adoptar este enfoque es utilizar mejor la información codificada en las trayectorias espaciotemporales de las manos de un sujeto en cámara.

Al adoptar un punto de vista espacial 2D más restrictivo con visibilidad temporal limitada, se pierde información que facilite el reconocimiento y clasificación de señas. Por lo tanto, se propone utilizar un método similar al de factorización descrito por Mark Borg [6] con el fin de factorizar los movimientos significativos del firmante directamente a partir de los datos de la imagen. En particular, se emplea una variante no rígida de la técnica de estructura a partir del movimiento (SfM) [66] para separar los movimientos 3D del sujeto en cámara, de la forma deformable de su cuerpo. Y la forma 3D deformable en cualquier momento se expresa como una combinación lineal ponderada de un pequeño número de formas clave predeterminadas: estas formas clave constituyen una estructura que abarca el espacio de formas de todas las maneras posibles en que el cuerpo 3D del firmante puede deformarse.

1.4.2.8.1. El método de factorización

Los métodos de factorización son una clase de algoritmos que pueden extraer "factores" significativos de los datos de la imagen, donde sólo las complejas y entrelazadas

interacciones de estos factores son directamente observables. Algunos ejemplos son la separación del estilo del contenido, la descomposición de la expresión facial y el SfM. Cuando aplicamos el método de factorización a la SfM, somos capaces de recuperar y factorizar el movimiento de un objeto en movimiento (o de una cámara en movimiento) a partir de la estructura o la forma 3D del objeto. Esto puede hacerse incluso cuando el objeto se observa con una cámara monocular no calibrada, y cuando no tenemos conocimiento previo sobre la forma del objeto y el movimiento de la cámara (o del objeto). En otras palabras, el método de factorización puede recuperar la forma 3D de un objeto directamente, sin necesidad de calcular la profundidad como paso intermedio, lo que se denomina enfoque de reconstrucción en un solo paso [6]

A continuación, se describe cómo funciona el método de factorización estándar para la SfM para un cuerpo rígido. El método, introducido por primera vez por Tomasi y Kanade [66], requiere un conjunto de puntos característicos pertenecientes al objeto de interés, y que estos puntos característicos sean rastreados con éxito a lo largo de un segmento de vídeo de cierta longitud F.

Dados los P puntos característicos, se construye la matriz de trayectoria W (también llamada matriz de medición) como sigue: Dadas las coordenadas $(\mathcal{U}_{t,p}\mathcal{U}_{t,p})$ sean las coordenadas 2D basadas en la imagen del punto de característica p en el tiempo t, donde $p \in \{1, \dots, P\}$ y $t \in \{1, \dots, F\}$. Apilamos estas coordenadas en una matriz W de tamaño $\mathbb{R}^{2F \times P}$ como sigue:

$$W_{2t-1,p} = u_{t,p}$$

$$\approx$$

$$W_{2t,p} = u_{t,p}$$

Ecuación 26. *Matriz W expresada para cada termino* $u_{t,p}$**.**

W se puede factorizar en M, una matriz que describe el movimiento 3D del objeto, y S, una matriz que describe la forma 3D del objeto:

$$\boldsymbol{W = MS}$$

Ecuación 27. *W expresado en términos de M y S.*

Tomasi y Kanade [66] emplean el algoritmo de descomposición de valores singulares

(SVD) para factorizar W en el producto de las dos submatrices de bajo rango M y S. Aparte de su simplicidad y elegancia matemática, SVD presenta un alto grado de estabilidad y robustez frente al ruido en W. Es necesario aclarar que en la Ecuación 26 se tiene el movimiento observable de los puntos característicos en W, y el movimiento 3D real en M. El movimiento en M consta de componentes de rotación y traslación. La matriz de movimiento M de la Ecuación 26 se simplifica a una matriz de rotación R. Por lo tanto, se obtene:

$$\boldsymbol{W = RS}$$

***Ecuación 28**. W expresado en términos de R y S.*

La Figura 46 ilustra el método de factorización para un cuerpo rígido.

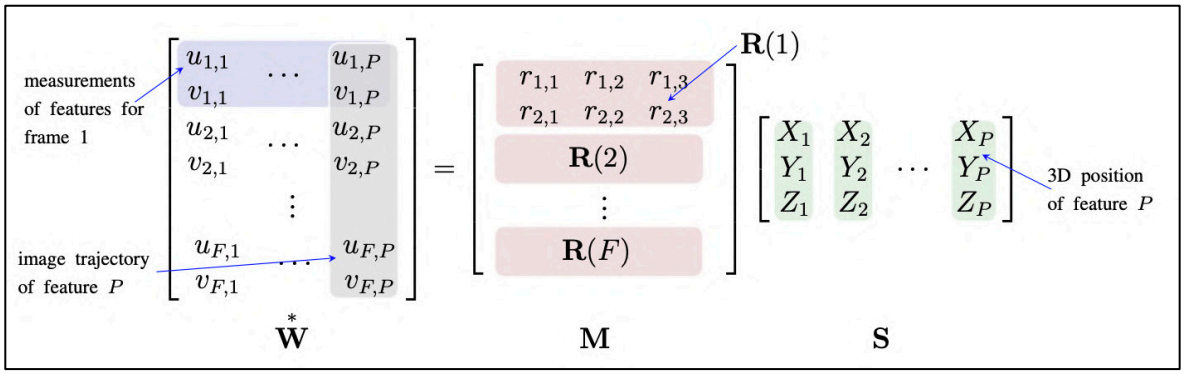


***Figura 46**. El método de factorización para un cuerpo rígido. La matriz de trayectoria W* se factoriza en una matriz de movimiento M y una matriz de forma rígida S.*

1.4.2.8.2. Factorización de estructuras no rígidas

La utilización del método de factorización para estructuras no rígidas a partir del movimiento (NRSfM) es más difícil, especialmente si no se conoce previamente el tipo de deformaciones 3D por las que pasa el objeto de interés. El autor Bregler aborda el problema NRSfM representando las complejas deformaciones 3D de un objeto en términos de una combinación lineal ponderada de un número limitado de formas clave. Denotando la forma 3D de un objeto en cualquier instante de tiempo t como $S(t)$, entonces ésta puede ser expresada en términos de K formas base $\{S_j\}_{j=1}^{k}$, con los correspondientes pesos variables en el tiempo $a_j(t)$ como sigue[6]:

$$S(t) = \sum_{j=1}^{k} a_j(t)\, S_j, \qquad S_j \in \mathbb{R}^{3K \times P}, \qquad a_j(t) \in \mathbb{R}$$

Ecuación 29. *Forma de un objeto 3D en un instante de tiempo expresada en términos de K bases de trayectoria.*

La Ecuación 28 puede escribirse en forma matricial: apilamos todos los $S(t)$ para obtener la matriz S, los pesos $a_j(t)$ a se agrupan en la matriz de pesos A de tamaño $\mathbb{R}^{3F \times 3K}$, y las K formas base S_j en la matriz base Θ de tamaño $\mathbb{R}^{3K \times P}$, dando lugar a[6]:

$$S = \Theta\, A$$

Ecuación 30. *Matriz S en términos de* $\Theta\ y\ A$.

La factorización de S en Θ y A se determina automáticamente. Sustituyendo lo anterior en la Ecuación 27, obtenemos lo siguiente:

$$W = R\, \Theta\, A$$

Ecuación 31. *Matriz W en términos de* $R, \Theta\ y\ A$.

1.4.2.8.3. Factorización de la trayectoria espacial

Existe una dualidad entre espacio y tiempo. En lugar de describir la deformación del objeto en términos de una combinación lineal ponderada de bases de forma (según la Ecuación 28), se puede también describir la deformación como una combinación lineal ponderada de K bases de trayectoria θ_t[6]:

$$\mathbf{\Phi}(i) = \sum_{j=1}^{K} a_j(i)\, \theta_j, \quad \theta_j \in \mathbb{R}^{3 \times F}, \quad a_j(i) \in \mathbb{R}$$

Ecuación 32. *Trayectoria de deformación en un punto de característica.*

donde $\mathbf{\Phi}(i)$ es la trayectoria de deformación de un punto de característica particular i, a_j son los pesos, y el espacio de trayectoria de baja dimensión es abarcado por la base de trayectoria $\mathbf{\Theta} = \{\theta_j\}_{j=1}^{K}$. El autor Akhter[66] ha demostrado que estas dos representaciones de la deformación del objeto son equivalentes en términos de su poder expresivo. Las coordenadas en el espacio de la forma se convierten en las bases en el espacio de la

trayectoria, y viceversa. Las Figura 47 presentada en la siguiente página ilustran esta dualidad.

En términos de los factores recuperados, el resultado de la factorización del espacio de trayectoria sigue estando dado por la Ecuación 30, salvo que la matriz base $\boldsymbol{\Theta}$ representa ahora la base de la trayectoria.

Una de las principales ventajas de utilizar la factorización del espacio de trayectoria sobre la factorización basada en la forma es que las bases de la primera tienden a ser más independientes del objeto - las bases de trayectoria pueden ser más fácilmente "recicladas" a través de las secuencias de vídeo, ya que son independientes de la forma 3D del objeto, y dependen sólo de la forma en que las deformaciones cambian en el tiempo. Este método define una base de trayectoria genérica a priori, en lugar de recuperarla automáticamente mediante la factorización[6]. Esto supone una reducción significativa del número de incógnitas, una mayor estabilidad en la reconstrucción y la posibilidad de manejar un mayor grado de deformación que la factorización basada en la forma. En cambio, los métodos de factorización basados en la forma necesitan aprender desde cero la base de forma específica para cada flujo de vídeo concreto. La base de la transformada discreta de coseno (DCT) [66]se adopta universalmente en la factorización del espacio de trayectoria, principalmente debido a su idoneidad para representar movimientos suaves, así como a sus propiedades de compactación de energía.

Una limitación del método de factorización del espacio de la trayectoria es[6]:

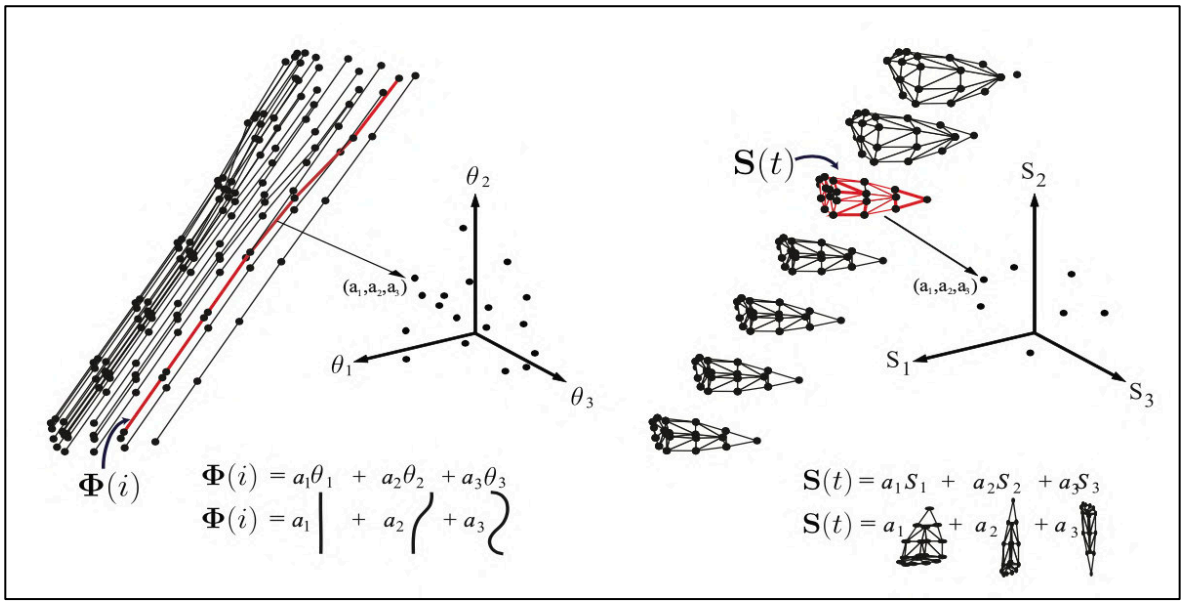


Figura 47: *La dualidad del espacio de la forma y el espacio de la trayectoria. (A) La trayectoria* $\mathbf{\Phi}(i)$ *de la i-ésima partícula de características (resaltada en rojo), se representa como una suma ponderada de trayectorias básicas, siendo los pesos* (a_1, a_2, a_3) *las coordenadas en el espacio de trayectorias que abarcan las bases* $\{\theta_1, \theta_2, \theta_3\}$. *(B) La forma* $S(t)$ *de todas las características en el tiempo t se representa como una combinación lineal de bases de forma* $\{S_1, S_2, S_3\}$., *siendo los pesos (a1, a2, a3) las coordenadas correspondientes en el espacio de forma. Nótese que en (A), las trayectorias reflejan la deformación del objeto, una boca en este caso. (Adaptado de Akhter et al. [264]. Los derechos de autor de la imagen original pertenecen a Ijaz Akhter y Yaser Sheikh*).

i. La trayectoria $\mathbf{\Phi}(i)$ de un punto de característica i (véase la Ecuación 31) se trata independientemente de las de los demás puntos. No se explota ninguna restricción o dependencia sobre cómo se deforman colectivamente los puntos de característica.

1.4.2.8.4. Algoritmo de TSF

El algoritmo la aplicación de la factorización espacial utiliza técnicas como la minimización del error mínimo cuadrado, norma de Frobenius, combinaciones lineales, SVD, mínimos globales, matrices ortogonales[6]. La Ecuación 29 indica la formula de recuperación la matriz de rotación R.

Pasando a la recuperación de la estructura $S = \Theta A$, primero se crea la matriz base DCT conocida $\mathbf{\Theta}$ de la siguiente manera. La base DCT unidimensional de la j-ésima base θ_j viene dada por [6]:

$$\theta_j(i) = \frac{\sigma_j}{\sqrt{F}} \cos\left(\frac{\pi\,(i-1)\,(j-1)}{F}\right), \quad i \in \{1, \ldots, F\}, j \in \{1, \ldots, K\} \qquad (6.21)$$

Ecuación 33. *Ecuación la de base DCT unidimensional de la j-ésima base* θ_j.

donde σ1 = 1, σj = √2 para j ≥ 2. A continuación, se apila las bases vectoriales unidimensionales de K como columnas de la matriz $\Phi_1 \in \mathbb{R}^{F\,X\,K}$:

$$\Phi_1 = [\theta_1, \theta_2 \cdots \theta_k]$$

Ecuación 34. *Ecuación la de base DCT unidimensional de la j-ésima base* θ_j.

obteniendo la matriz base de la trayectoria 3D completa Φ con la estructura requerida como se indica a continuación, donde ⊗ es el producto de Kronecker[6]:

$$\boldsymbol{\Theta} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \otimes \boldsymbol{\Theta_1} \otimes \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \otimes \boldsymbol{\Theta_1} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \otimes \boldsymbol{\Theta_1} \otimes \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ecuación 35. *Matriz de base de la trayectoria 3D completa* Φ.

Partiendo de la Ecuación 30 original, se tiene que $G = R\,\Theta$, ambas conocidas ahora:

$$\mathrm{W} = \mathrm{R\Theta A} = \mathrm{GA}$$

Ecuación 36. *Matriz W en términos de* G y A.

Como no se puede tomar la matriz inversa de G, ya que no es una matriz cuadrada, se multiplica por G^T en su lugar [6]:

$$(G^T G)^{-1}\, G^T W = (G^T G)^{-1}\, (G^T G)\, A$$

$$(G^T G)^{-1}\, G^T W = A$$

Ecuación 37. *Matriz A en términos de G y W.*

Lo que da como resultado la matriz de pesos A, y por tanto la estructura S (véase también la Figura 48). Esto completa el proceso de factorización de la trayectoria espacial.

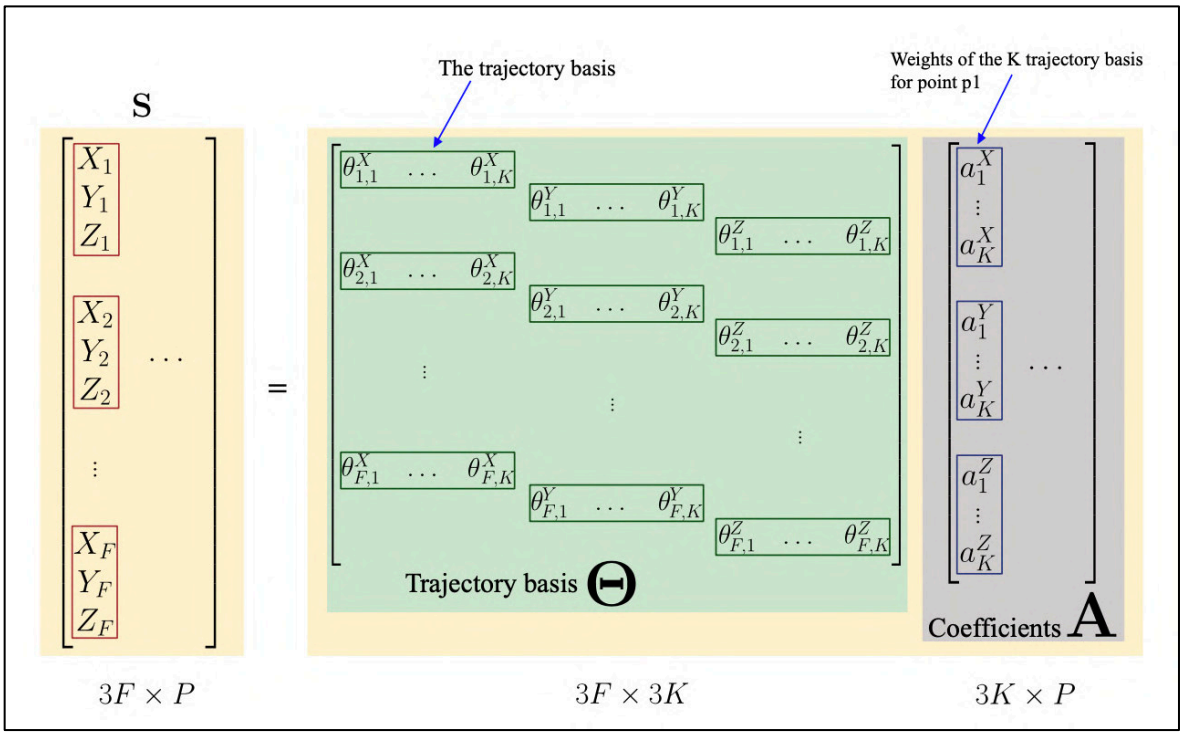


Figura 48. *Factorización de la estructura S en una combinación lineal de bases de trayectoria Θ ponderada por A. La matriz Θ se expande en la forma diagonal de bloque intercalada dada, y se rellena con ceros, para poder multiplicarla correctamente con A.*

1.4.2.8.5. Aplicación de la factorización de la trayectoria espacial en el reconocimiento de señas

Como se ha mencionado en la introducción de esta sección, la idea clave del enfoque por aplicar, es considerar el movimiento de las manos del sujeto en cámara, como constitutivo para las deformaciones de la forma 3D del firmante (véase Figura 49).

Se aplica el método de factorización de la trayectoria espacial, factorizando la matriz W de trayectorias KLT en el producto $R\ \Theta\ A$ [6], donde:

(i) La matriz R representa las rotaciones de las manos del sujeto en cámara.

(ii) El producto $\Theta\ A$ representa la forma evolutiva del firmante, es decir, los movimientos de las manos en relación con el centroide del sujeto en cámara. Esto se expresa en términos de una base de trayectoria Θ que abarca un espacio de trayectoria, y los correspondientes coeficientes del espacio de trayectoria

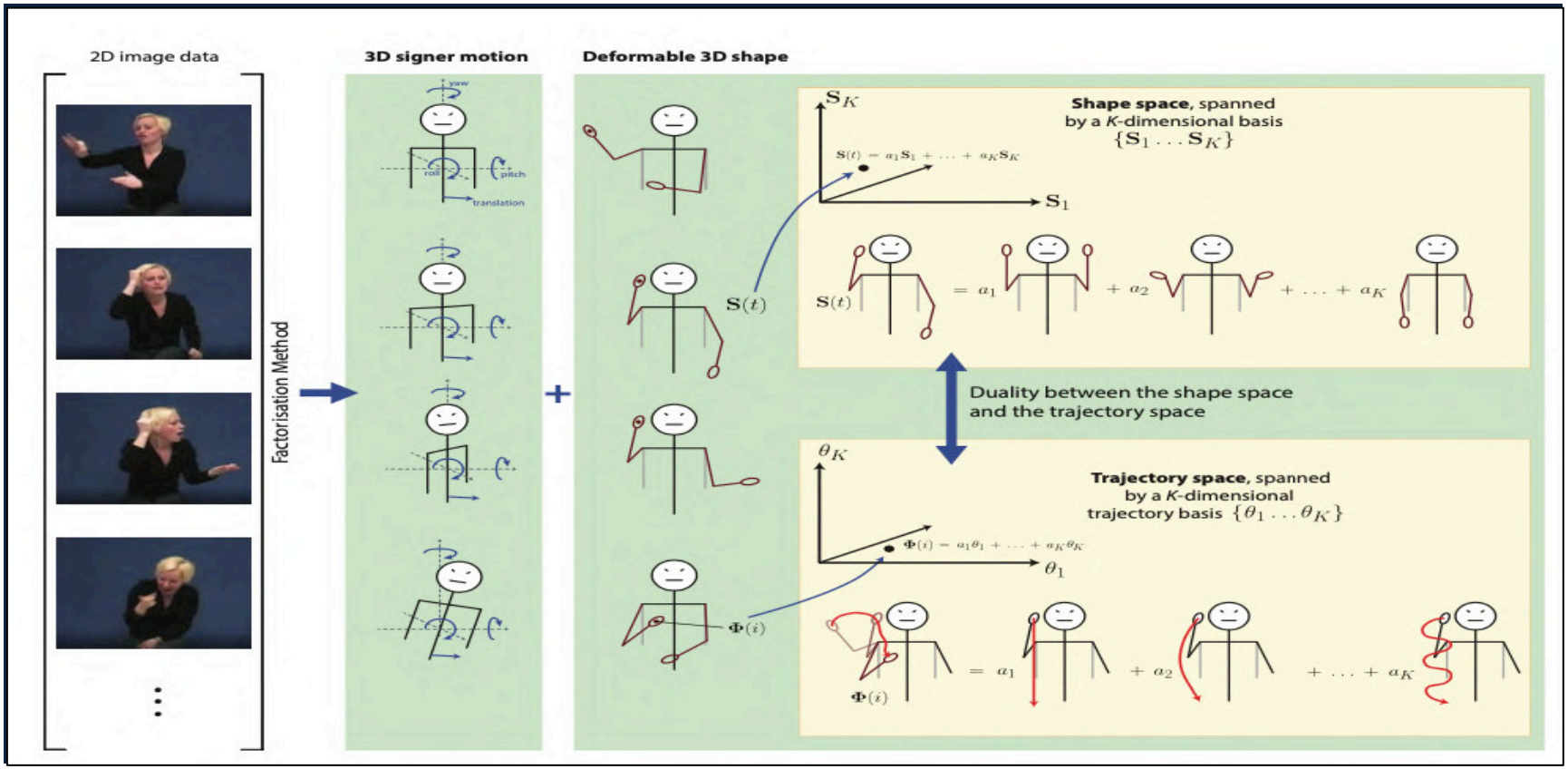


Figura 49. Ilustración esquemática del uso de la factorización del Espacio de Trayectoria para reconocimiento de señales. En primer lugar, los datos de la imagen 2D se factorizan en movimientos 3D del firmante (expresados en términos de rotaciones y traslaciones globales del cuerpo del firmante), y en formas 3D deformables. La forma del firmante se deforma debido a los movimientos de las manos con respecto al centro del cuerpo, y en el momento t se expresa como una combinación lineal ponderada de K formas clave en el espacio de la forma (véase la parte superior derecha). Una interpretación dual consiste en considerar el movimiento de deformación de una característica particular de la mano $\mathbf{\Phi}(i)$ (por ejemplo, el centro de la mano derecha) como una combinación lineal ponderada de de K trayectorias clave en el espacio de trayectorias (parte inferior derecha). Dada una base de trayectoria predefinida, la factorización puede expresar la firma en términos de movimientos globales del cuerpo, así como las trayectorias de la mano en 3D expresadas en términos de la base de trayectoria elegida.

1.4.2.8.6. La matriz de trayectoria W

En la matriz de trayectoria W, se centra únicamente en representar únicamente los puntos de característica de las manos del sujeto en cámara, a diferencia de otras aplicaciones en la que se recupera todo el cuerpo presentado en la Figura 50 [6]. Las características de las manos izquierda y derecha del firmante, se pueden utilizar de manera aislada W_{LH} , W_{RH} o se agrupan en W_{LH} y W_{RH} respectivamente, y luego se apilan como sigue[66]:

$$W = [W_{RH} \mid W_{LH}]$$

Ecuación 38. *Matriz de trayectoria W conformada por la representación ambas manos del sujeto en cámara.*

Se debe registrar W con respecto al centroide del cuerpo del firmante x_0. Si no se realiza este procedimiento, los movimientos de las manos se describirán con respecto a su centro de masa común, lo que creará movimientos ilegítimos de las manos[66].

$$W = W - x_0(t)$$

Ecuación 39. *Matriz de trayectoria W con respecto al centroide del cuerpo del sujeto en cámara.*

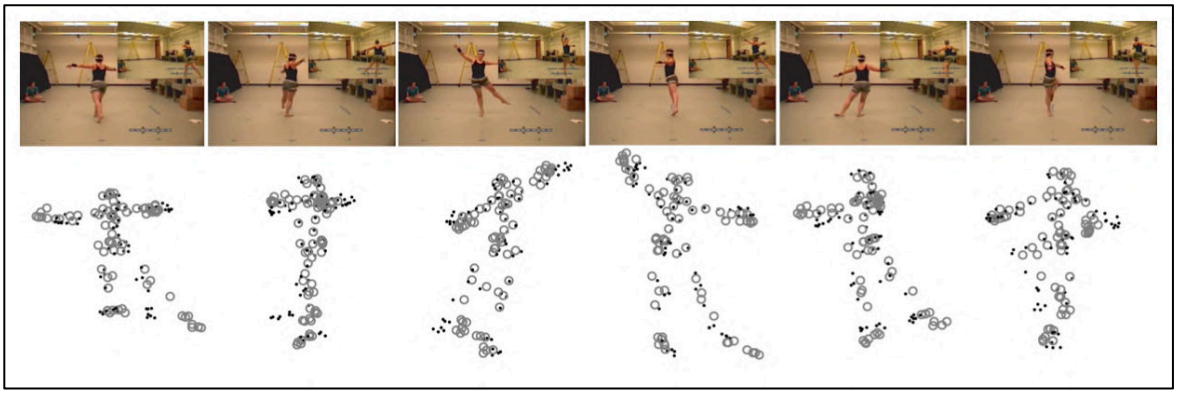


Figura 50: *Factorización del espacio de trayectorias aplicada a la secuencia de baile del conjunto de datos Mocap de la CMU. A diferencia de nuestro enfoque, es típico considerar las deformaciones del objeto completo en muchas aplicaciones no relacionadas con las señas. (Fuente: Akhter et al. [264]. Derechos de autor de Ijaz Akhter y Yaser Sheikh. Reproducido con permiso).*

1.4.2.8.7. Rotaciones globales

El método de factorización recupera el movimiento global del firmante en la matriz R, que consiste en una secuencia de submatrices de rotación $\{R(t)\}_{t=1}^{F}$ para cada paso de tiempo t vista en la Ecuación 30. Para comprender mejor la rotación del firmante, se expresa cada una de las matrices de rotación $R(t)\,\epsilon\,\mathbb{R}^{2\times 3}$ en términos de ángulos de Euler $(\phi_t, \theta_t, \psi_t)$ utilizando la ecuación siguiente[6]:

$$R(t) = \begin{bmatrix} i_1 & i_2 & i_3 \\ j_1 & j_2 & j_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \; R_Y(\psi_t)R_X(\theta_t)R_Z(\phi_t)$$

Ecuación 40. *Representación de rotaciones en términos de ángulos de Euler.*

Esto significa que ahora $R(t)$ puede interpretarse como una secuencia de tres rotaciones elementales sucesivas alrededor de los ejes $Y - X - Z$. Y si se fija correctamente la orientación inicial del firmante según la sección anterior, entonces el ángulo de Euler ϕ_t (la rotación alrededor del eje Z) puede interpretarse como el ángulo de balanceo en el tiempo t, el ángulo θ_t como el cabeceo y ψ_t como la guiñada[6].

- **Coeficientes DCT**

Se utiliza los coeficientes DCT de la matriz A para determinar cómo se está "codificando" el movimiento de las manos del sujeto en cámara, y si lleva alguna información significativa que pueda ser útil para el reconocimiento de señales. La base DCT es la base elegida en la aplicación de la factorización de la trayectoria espacial, debido a su capacidad para representar movimientos suaves y sus propiedades de compactación de energía, a la vez que es independiente de los datos[6].

1.4.2.9. Modelo Cliente Servidor

Para la realización de este proyecto, se desea aplicar un modelo de computación distribuida para la comunicación entre computadores, al dividir las funciones de procesamiento de VA en conjunto con la aplicación de modelos de DL y modelos matemáticos utilizados en la aplicación TSF, a diferencia de la captura y envió de imágenes en tiempo real. Este enfoque permite utilizar hardware y software a la medida para cada función utilizada en el desarrollo del prototipo del caso de estudio.

El Internet conjunto con la tecnología Web son usados hoy en día para casi todas las actividades que involucran al ser humano, mayoritariamente en el desarrollo de sus comunicaciones; lo que demuestra la estrecha relación que mantiene, siendo este un elemento esencial del siglo XXI. Al mismo tiempo, el Internet nunca se ha limitado a los ordenadores; se ha abierto a todo tipo de dispositivos digitales inteligentes, como los móviles, y dispositivos terminales IoT, siendo la arquitectura Cliente-Servidor (CS) el modelo de comunicación de la Web[67].

El término Cliente/Servidor se utiliza para describir una arquitectura informática para el desarrollo de sistemas informáticos. Esta arquitectura se basa en la distribución de funciones entre dos tipos de procesos independientes y autónomos: Servidor y Cliente. La red une al servidor y al cliente, proporcionando el medio a través del cual los clientes y el servidor se comunican [67][68]. La Figura 51 muestra un modelo básico de la arquitectura CS. A continuación, se describe los elementos básicos de su arquitectura[68][69]:

- Un Cliente es cualquier proceso que solicita servicios específicos al proceso servidor.
- Un Servidor es un proceso que proporciona los servicios solicitados por el Cliente. Los procesos Cliente y Servidor pueden residir en el mismo ordenador o en diferentes ordenadores conectados por una red. (client-server computing)
- Protocolos de la capa de red, como TCP/IP.

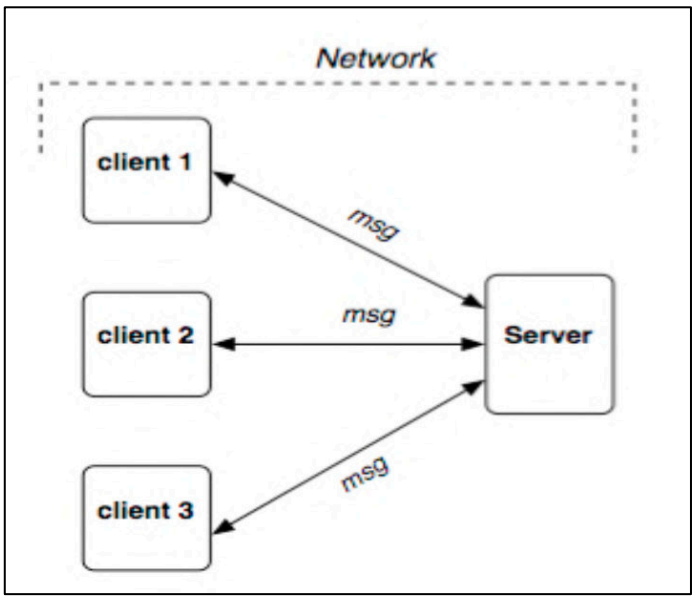


Figura 51. *Representación de la arquitectura Cliente-Servidor.*

Cuando los procesos Cliente y Servidor residen en dos o más ordenadores independientes en una red, el Servidor puede proporcionar servicios para más de un Cliente. Además, un cliente puede solicitar servicios a varios servidores de la red sin tener en cuenta la ubicación o las características físicas del ordenador en el que reside el proceso del Servidor [68].

Para implementar la comunicación de red mediante programación, el método más común es el socket, que es una interfaz de red, un punto final de un flujo de comunicación entre procesos a través de una red. Un socket contiene la dirección del socket y los datos. La dirección del socket incluye la dirección IP local y la dirección IP remota que son las direcciones de cliente y el servidor, además el número de puerto que se utiliza para enviar y recibir datos. En cuanto a los datos incluye también cabeceras utilizadas para sincronización[70]. La figura 52 indica las diferentes funciones utilizadas en comunicación entre un Cliente y Servidor por medio de sockets.

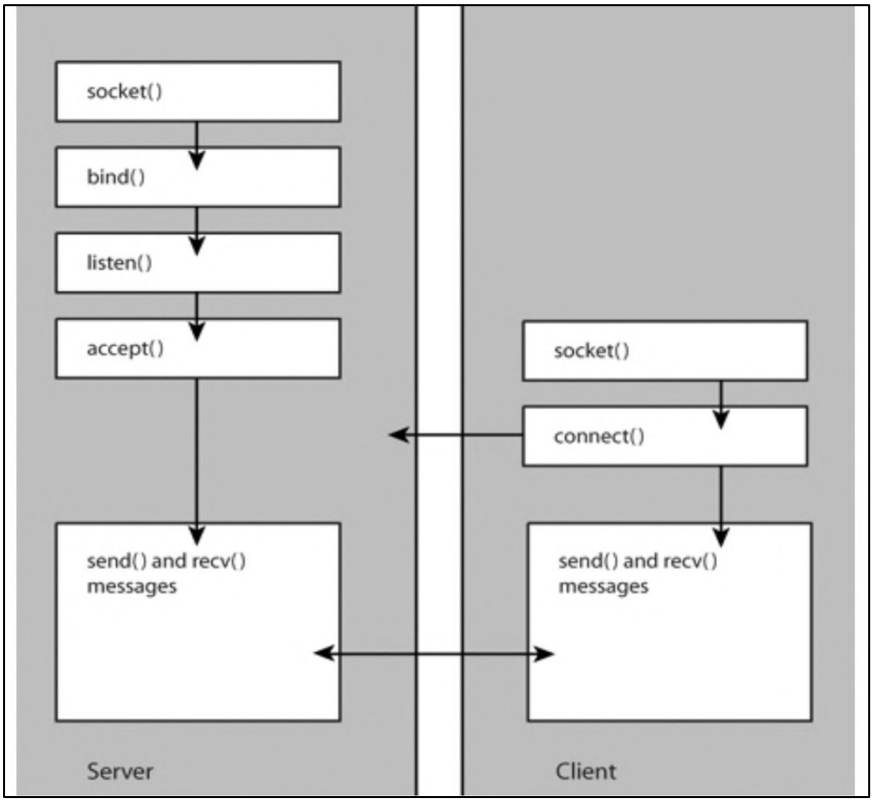


Figura 52. *Arquitectura de comunicación entre un Cliente y Servidor por medio de sockets.*

Algunos de los beneficios de aplicar la arquitectura CS son [71]:

- División del procesamiento de la aplicación en varias máquinas:
    - Los datos y funciones no críticas se procesan en el cliente

    - Las funciones críticas se procesan en el servidor
- Optimiza el servidor para el procesamiento y almacenamiento de datos (por ejemplo, gran cantidad de memoria y espacio en disco)
- Escalamiento horizontal - Se pueden añadir múltiples servidores, cada uno con capacidades y potencia de procesamiento, para distribuir la carga de procesamiento.
- Escalamiento vertical - Puede trasladarse a máquinas más potentes, como un miniordenador o un mainframe para aprovechar el rendimiento del sistema más grande.

1.4.3. Herramientas y librerías de desarrollo

En esta sección del documento se detalla todas las herramientas utilizadas, como frameworks, lenguajes de programación y plataformas, así como librerías de desarrollo específicas.

1.4.3.1. Herramientas

- **Python**

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con una semántica dinámica. Las estructuras de datos de alto nivel que incorpora, combinadas con la tipificación y vinculación dinámicas, hacen muy atractivo este lenguaje para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripting o pegamento para conectar componentes existentes, además admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código[72]. Python es el lenguaje de programación escogido para el desarrollo del prototipo del sistema de control de dispositivos domóticos.

- **Matlab**

MATLAB® es una plataforma de programación diseñada específicamente para ingenieros y científicos para analizar y diseñar sistemas y productos que transforman el mundo. El corazón de MATLAB es el lenguaje MATLAB, un lenguaje basado en matrices que permite la expresión más natural de las matemáticas computacionales[73]. MATLAB es la plataforma de programación escogida para la ejecución de algoritmos relacionados con la aplicación de la factorización de la trayectoria espacial.

- **Anaconda Individual Edition**

Anaconda® es un gestor de paquetes, un gestor de entornos, una distribución de ciencia de datos Python/R y una colección de más de 7.500 paquetes de código abierto[74]. Este gestor permitirá utilizar la herramienta Jupyter Notebook.

- **Jupyter Notebook**

Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Sus usos incluyen: limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos, y aprendizaje automático[75]. La aplicación será utilizada, para generar, analizar y depurar el conjunto de datos utilizados en el modelo de RN, además de contener el código y ejecuciones del entrenamiento, validación y testeo de la RN a utilizar.

- **Visual Studio Code**

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity)[76]. Este editor permitirá programar todo el código utilizado en el prototipo del sistema de control de dispositivos domóticos.

- **JSON**

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. JSON es de fácil lectura y escritura para los usuarios. JSON es fácil de analizar y generar por parte de las máquinas. JSON se basa en un subconjunto del lenguaje de programación JavaScript, Estándar ECMA-262 3a Edición - diciembre de 1999. JSON es un formato de texto completamente independiente del lenguaje, pero que utiliza convenios que resultan familiares a los programadores de lenguajes de la familia C, incluidos C, C++, C#, Java™, JavaScript, Perl, Python y mucho otros. Estas características hacen de JSON un lenguaje de intercambio de datos ideal[77]. JSON será el código de comunicación entre Amazon Alexa y dispositivos de IoT.

1.4.3.1.1. Componentes de AWS

- **Amazon Web Services (AWS)**

Amazon Web Services (AWS) es la plataforma en la nube más completa y ampliamente adoptada del mundo, que ofrece más de 200 servicios completos desde centros de datos de todo el mundo[78]. Esta plataforma será utilizada para alojar en la nube el backend del skill de Amazon Alexa.

- **AWS IAM**

AWS Identity and Access Management (IAM) permite administrar el acceso a los servicios y recursos de AWS de forma segura. Con IAM, es posible crear y administrar usuarios y grupos de AWS, y utilizar permisos para permitir y denegar su acceso a los recursos de AWS[79], de esta forma se puede acceder de manera remota a los recursos utilizados en el desarrollo del skill de Amazon Alexa.

- **AWS IoT Core**

AWS IoT Core es un servicio administrado en la nube que permite que los dispositivos conectados interactúen de manera fácil y segura con aplicaciones en la nube y otros dispositivos. El núcleo de AWS IoT admite miles de millones de dispositivos y billones de mensajes, y puede procesar y enrutar esos mensajes a los puntos de enlace de AWS y a otros dispositivos de manera fiable y segura[80]. Cuando los dispositivos IoT conectados estén administrados por AWS IoT Core es posible que sean manejados por el backend del skill de Amazon Alexa desarrollado.

- **AWS Lambda**

Lambda es un servicio de computación que permite ejecutar código sin necesidad de aprovisionar o gestionar servidores. Lambda ejecuta el código en una infraestructura informática de alta disponibilidad y realiza toda la administración de los recursos informáticos, incluido el mantenimiento del servidor y del sistema operativo, el aprovisionamiento de capacidad y el escalado automático, la supervisión del código y el registro de operaciones[81]. El servicio Lambda contiene todo el código utilizado en el skill de Amazon Alexa, además del código de enrutamiento de endpoints utilizados por los dispositivos IoT.

1.4.3.1.2. Componentes de Amazon Alexa

- **Alexa Skills Kit**

Alexa Skills Kit (ASK) es un marco de desarrollo de software que permite crear componentes, llamados skills. Las skills son como aplicaciones para Alexa[82]. En este caso se desea crear una aplicación del tipo Smart Home Skill con el fin de ser la tecnología por utilizar para controlar dispositivos domóticos.

- **Smart Home Skill API**

Smart Home Skill API es un tipo de Alexa skill que brinda una solución de automatización que admite la programación y configuración de dispositivos que interpretan órdenes de voz y transmiten mensajes a dispositivos habilitados en la nube[83], sin embargo, se desea que la skill no responda a órdenes de voz, sino a una invocación por línea de comandos.

- **Alexa Skills Kit Command Line Interface (ASK CLI)**

Alexa Skills Kit Command Line Interface (ASK CLI) (ASK CLI) es una herramienta que permite administrar las habilidades de Alexa y sus recursos relacionados, como las funciones de AWS Lambda. Con ASK CLI, se tiene acceso a la API de administración de skills, que permite administrar las skills de Alexa mediante programación desde la línea de comandos[84]. ASK CLI es el medio por el cual el backend del servidor del sistema de control de dispositivos domóticos se puede comunicar desde línea de comandos con la skill de Amazon Alexa desarrollado.

- **Amazon Smart Plug**

Amazon Smart Plug permite usar Alexa para controlar luces, ventiladores, cafeteras y mucho más. Es posible programar el encendido y apagado automático de luces, ventiladores y electrodomésticos, o controlarlos de forma remota[85]. Este dispositivo IoT es el escogido para ser utilizado en el prototipo del sistema de control de dispositivos domóticos.

1.4.3.2. Librerías de Desarrollo

- **ImageZMQ**

imageZMQ es un conjunto de clases de Python que transmiten imágenes de OpenCV de un ordenador a otro utilizando la mensajería PyZMQ[86].

- **OpenCV**

OpenCV (Open Source Computer Vision Library) es una librería de software de visión por computador y aprendizaje automático de código abierto. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión por computador y para acelerar el uso de la percepción de la máquina en los productos comerciales[87].

- **Mediapipe**

MediaPipe es un marco de trabajo para construir pipelines de ML para aplicación multimedia (por ejemplo, vídeo, audio, cualquier serie de datos de tiempo) y multiplataforma (es decir, Android, iOS, web)[88].

- **MATLAB Engine**

MATLAB Engine para Python proporciona un paquete que hace posible que Python llame a MATLAB como motor de cálculo[89], es decir permite llamar a funciones de MATLAB desde Python.

- **Tensorflow**

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático y los desarrolladores creen e implementen aplicaciones con tecnología de AA fácilmente.[90]

- **Scikit-learn**

Scikit-learn es un módulo de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas supervisados y no supervisados de mediana escala.[91]

Proporciona una selección de herramientas de modelado estadístico incluyendo clasificación, regresión, clustering y reducción de dimensionalidad[92].

- **Keras**

Keras es una librería de código abierto de componentes de redes neuronales escrita en Python. Keras es capaz de funcionar sobre TensorFlow, Theano, PlaidML y otros. La biblioteca fue desarrollada para ser modular y fácil de usar.[93]

2. METODOLOGÍA

Este capítulo contiene las diferentes etapas que se realizarán para el desarrollo del prototipo del caso de estudio propuesto en base a los conceptos técnicos enunciados en el capítulo anterior. El capítulo empieza con una breve descripción de la metodología Action-Research (AC) del autor Ernest T. Stringer[94], seguido de una planificación de acción en base a esta metodología.

La primera etapa describe el conjunto de datos utilizado para la realización del presente proyecto, seguido de la descripción del proceso de reconocimiento de manos. Posteriormente, se describe la aplicación de TSF en el reconocimiento de señales. Además, se describe el conjunto de datos de entrada y el modelo de clasificador utilizado en la traducción de señales en instrucciones definidas.

Finalmente se presenta la implementación del caso de estudio, correspondiente al desarrollo del sistema de control de sistemas domóticos CODDO, los diseños de las arquitecturas utilizadas en el sistema, y descripción de procesos y diagramas de flujo.

2.1. Metodología Action Research

La metodología Action-Research (AC) se conoce con muchos otros nombres, entre ellos investigación participativa, investigación colaborativa, investigación emancipadora, aprendizaje en acción e investigación de acción contextual, pero todas son variaciones de el mismo tema. En pocas palabras, la investigación de acción es "aprender haciendo": se identifica un problema, se hace algo para resolverlo, ve cuán exitosos fueron los esfuerzos y, si no es satisfactorio, se repite el proceso[95]; tal y como se ilustra en la Figura 53. Cabe destacar que esta metodología se utiliza en situaciones reales, más que en estudios artificiales y experimentales, ya que se centra en la solución de problemas reales.

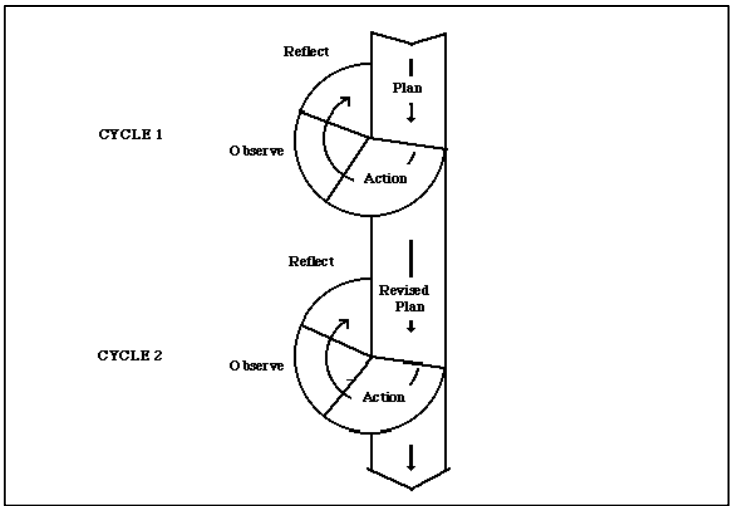


Figura 53. *Modelo básico de la metodología Action-Research.*

Actualmente la metodología es utilizada con diversos enfoques y perspectivas, depende de la problemática a abordar. Por su parte Ernest Stringer propone su propio enfoque y modelo, en donde define AC como un enfoque sistemático de la investigación que permite a las personas encontrar soluciones efectivas a los problemas que confrontan en su vida diaria, asimismo caracteriza la investigación como el establecimiento un problema o cuestión a investigar, seguido de un proceso de investigación y finalmente se brinda explicaciones que permiten a los individuos comprender la naturaleza del problema [94].

El modelo de Stringer es un enfoque de colaboración para la indagación o investigación que proporciona a las personas los medios para tomar medidas sistemáticas para resolver problemas específicos[96]. Stringer señala que a medida que los investigadores implementan un estudio, estos procesos pueden ser descritos de la siguiente manera:

- **Diseñar el estudio** refinando cuidadosamente el tema a investigar, planificando procesos sistemáticos de investigación, comprobando la ética, y la validez del trabajo que se realizará.
- **Recopilar datos** incluida información de diversas fuentes verídicas y comprobables con fundamento científico.
- **Analizar los datos** para identificar características clave y patrones, con el fin de tener una síntesis de los datos y desarrollar conclusiones.
- **Usar** los resultados del estudio para trabajar hacia la resolución del problema investigado.

- **Comunicar** los resultados del estudio al público pertinente.

El plan de investigación propuesto por Stringer proporciona un marco simple pero poderoso (Ver > Pensar > Actuar), permite a las personas comenzar sus indagaciones de manera sencilla y dar mayor detalle a los procedimientos a medida que aumenta la complejidad de las problemáticas [97]. La siguiente tabla muestra cómo las fases de las rutinas se relacionan con las prácticas de investigación tradicionales.

Rutina básica de Action-Research
Ver • Recopilar información relevante (Recopilar datos) • Construir una imagen: Describir la situación (Definir y describir)
Pensar • Explorar y analizar: ¿Qué está pasando aquí? (Analizar) • Interpretar y explicar: ¿Cómo/por qué son las cosas como están? (Teorizar)
Actuar • Plan (Informe) • Implementar • Evaluar

***Tabla 3**. Las fases de la rutina se relacionan con las prácticas de investigación tradicionales* [1].

2.2. Definición del conjunto de datos

Los datos por utilizar en el sistema de reconocimiento automático de señales corresponden a una secuencia de imágenes que representan la gesticulación de un sujeto en cámara de inicio a fin ilustrado en la Figura 54a. Cada imagen o frame (Fi) es representado computacionalmente como una matriz de dimensionalidad $m \times n$, donde m corresponde a las columnas o anchura de la imagen y n a las filas o altura de la imagen, ilustrado en la Figura 54b. Cada elemento de la matriz corresponde a un píxel con valores de 0 entre 255.

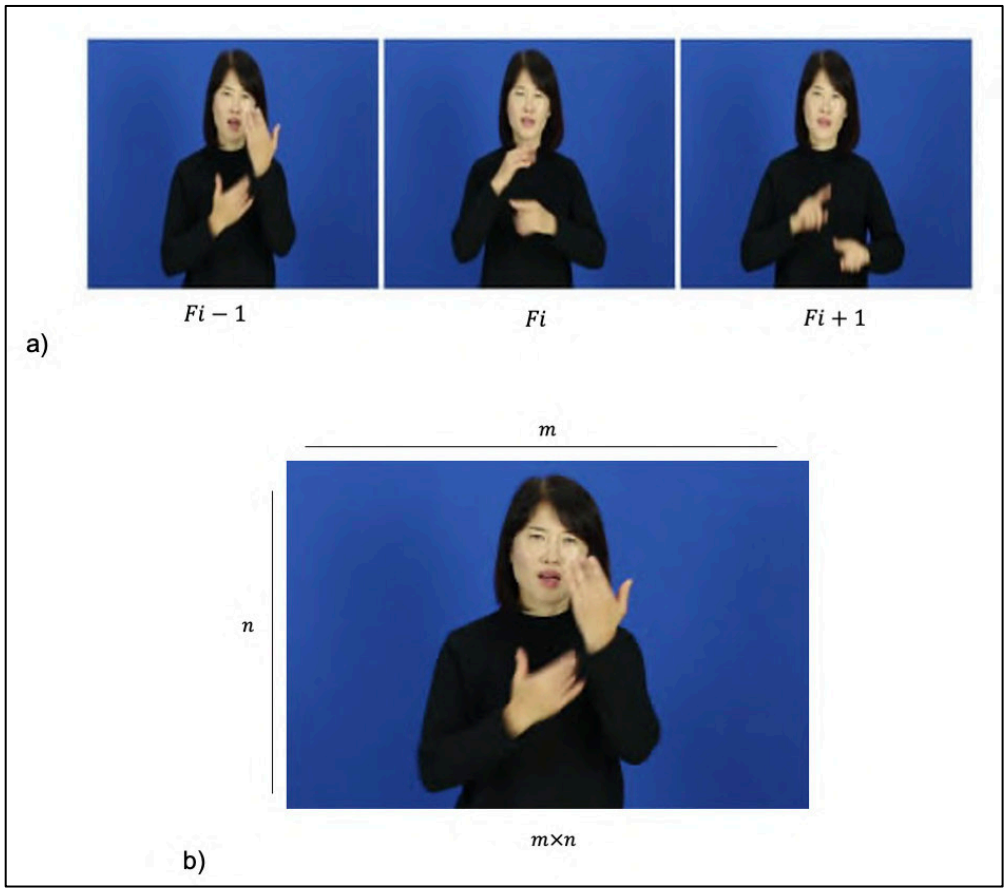


Figura 54. *Representación de un frame de Imagen.*

El conjunto de datos es utilizado como entrada para el modelo de reconocimiento de manos y la reconstrucción de su estructura en 3D por medio de la aplicación de TSF, posteriormente en el entrenamiento y testeo del modelo de CNN, por lo cual se expone en futuras secciones la creación de un propio conjunto de datos para la realización de este proyecto.

2.2.1 Definición de señas

A partir de esta sección, las señas lingüísticas de ASL serán conocidas como señales, las señales escogidas a ser reconocidas forman parte del diccionario de ASL mencionado en la sección 1.6.1, para fines de uso en el prototipo propuesto se ha seleccionado 3 señales de uso cardinal, es decir indica tanto dirección como un movimiento. La Figura 55, indica una ejemplificación del movimiento de las manos a realizar por el sujeto en cámara. Las señales se describen a continuación en la siguiente tabla:

Señal	Mano utilizada	Significado
LEFT	Izquierda	Indicación de dirección hacia la izquierda
RIGHT	Derecha	Indicación de dirección hacia la derecha
TURN	Izquierda	Indicación de movimiento de giro

Tabla 4. *Descripción de señas utilizadas.*

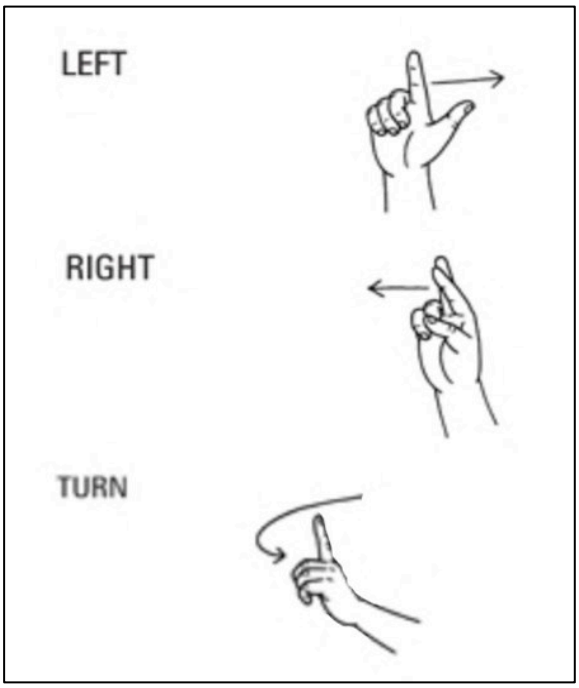


Figura 55*. Ejemplificación del movimiento de manos de las señales escogidas.*

2.2.2 Construcción del conjunto de videos

Con el objetivo de obtener un eficiente modelo de CNN, es necesario tener una cantidad considerable de datos que permitan capturar la mayor cantidad de casos en las que se puedan realizar las señales anteriormente descritas. En el Internet a día de hoy no existen gran variedad de conjuntos de videos que presenten sujetos en cámara gesticulando señales con las manos, sin embargo, podemos rescatar uno de ellos; realizado por el autor Mark Borg y utilizado en el dataset “Signing in the Wild” [98], contiene videos obtenidos de la plataforma YouTube con un total de 1120 videos, la Figura 56 muestra una vista del conjunto de videos.

***Figura 56**. Dataset "Signing in the Wild".*

Al no adaptarse a las necesidades de este proyecto, se ha decidido crear un propio conjunto de videos llamado "Directional Signs ASL" en el que un sujeto en cámara gesticula las 3 señales deseadas A continuación se describe los resultados de la construcción del conjunto de videos presentado en la Tabla 5. Al finalizar la construcción del conjunto de videos se obtienen un total de 2432 videos entre las señales LEFT, RIGHT, y TURN. Se muestra una vista del conjunto de videos en la Figura 57.

Señal	Número de Videos
LEFT	809
RIGHT	822
TURN	801
Total	2432

***Tabla 5.** Descripción de elementos en el conjunto de videos construido.*

Figura 57. *Vista del conjunto de videos "Directional Signs ASL".*

2.2.3 Consideraciones de grabación

Al preguntarse qué condiciones deben existir para que un sujeto en cámara pueda gesticular señales efectuadas por las manos y estas puedan ser reconocidas, es necesario reflexionar que restricciones necesitan aplicarse a la grabación de cada video. De ser el caso de no enlistar estas restricciones se tiene un conjunto de videos con más variabilidad y por lo tanto más difícil de cumplir el objetivo. En consecuencia, se enuncia las consideraciones tomadas en cuenta en el conjunto de videos "Directional Signs ASL":

1. Se debe tomar una distancia de 1 metro frente a la cámara.
2. Se debe utilizar prendas en las que se pueda distinguir fácilmente brazos y manos (preferiblemente camisetas, blusas).
3. La duración de la realización de la señal es de entre 3 a 4 segundos (se debe tomar como referencia contar hasta 4).
4. Se debe empezar el video con posición inicial de las manos ocultas de la cámara o detrás de la espalda, realizar la señal y finalizarlo con la posición inicial. (Si una mano no está siendo utilizada al momento de realizar la señal, mantenerla oculta de la cámara o detrás de la espalda)
5. Se debe asegurar que la posición las manos durante la realización de la señal deben ubicarse en la franja central de la cámara.
6. La cámara debe estar inclinada en posición horizontal.

2.2.4 Análisis del conjunto de videos

Una vez que se obtiene el conjunto de videos, se desea que cada clase, en este caso cada señal tenga un mismo número de elementos como datos; al construir el conjunto de videos por cuenta propia se lo puede garantizar y se lo demuestra en la Figura 58 que ilustra una similitud entre el número de elementos de cada clase. Cada elemento pasa por un proceso de preprocesamiento de cada uno de sus fotogramas de imagen o frames (Fi) para más adelante capturar la secuencia que corresponda a la duración de la gesticulación de una señal, no obstante, no todos pueden llegar a contar como parte de la secuencia de la señal ya que tendrán que pasar por un modelo de reconocimiento de manos en el que puede existir el caso que no sea reconocido y no cuente dentro de la secuencia.

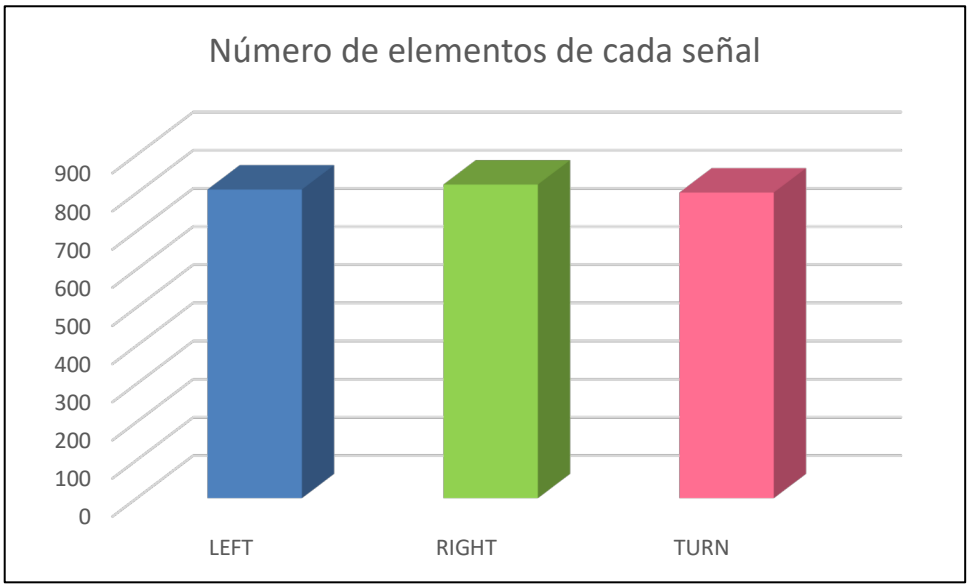


Figura 58. *Representación gráfica del número de elementos de cada señal.*

Una secuencia de frames para poder ser entrada del método de la aplicación de TSF debe tener una longitud mínima ≥ 120 frames, de llegar a tener una longitud menor la secuencia no cuenta como reconocida, por lo tanto, el video ha sido perdido. La Tabla 6 muestra cuantas secuencias de frames fueron reconocidos y al contrario cuantos se perdieron.

Señal	Número de secuencias	Reconocidos	Perdidos
LEFT	809	803	6
RIGHT	822	801	21

TURN	801	794	7
Total	2432	2398	34

Tabla 6. *Secuencias reconocidas y perdidas.*

El porcentaje de secuencias perdidas corresponden al 1,4% del total del conjunto de videos, por lo que las restricciones aplicadas a la grabación de cada video favorecieron en reducir el número de casos en los cuales una secuencia es descartada. La figura 59 muestra la proporción entre secuencias reconocidas y perdidas.

Figura 59. *Proporción entre secuencias reconocidas y perdidas del conjunto de videos.*

2.3. Reconocimiento de manos

Partiendo del hecho que en una secuencia de frames se captura el movimiento de las manos y resto del cuerpo, es imprescindible contar con un modelo que permita reconocer manos y resto de cuerpo, además de dar un seguimiento a cada objeto. El modelo de ML de Google “MediaPipe”[88] es el escogido para el reconocimiento de manos de un sujeto en cámara, debido a que logra un rendimiento optimo aplicado en tiempo real.

- **ML MediaPipe Holistico**

MediaPipe ofrece soluciones rápidas y precisas, aunque independientes, para este tipo de tareas en donde, combina las solucionas todas en tiempo real en una solución integral que requiere la inferencia simultánea de múltiples redes neuronales dependientes. MediaPipe Holistico integra modelos separados para los componentes de pose, cara y mano, los cuales está optimizado para su uso particular. Sin embargo, debido a sus diferentes enfoques de solución la entrada de un componente no es adecuada para otro. El modelo tiene un proceso de varias etapas, que trata las diferentes regiones utilizando una resolución de imagen apropiada para la región[99]. La Figura 60 ilustra el proceso de reconocimiento holístico con MediaPipe. A continuación, se presenta una breve descripción del modelo:

1. Se estima la pose humana (parte superior de la Figura 60 presentada en la siguiente página) con el detector de pose de BlazePose y se consigue un modelo de referencia.
2. Utilizando los puntos de referencia de la postura deducida, se deriva tres regiones de interés (ROI) para cada mano y cara.
3. Se recorta el fotograma de entrada de resolución completa en las regiones ROI y se aplica modelos de cara y mano específicos, con la tarea de estimar sus correspondientes puntos de referencia.
4. Se combina todos los puntos de referencia con los del modelo de postura para para generar un total de 543 puntos de referencia (33 puntos de referencia de la pose, 468 puntos de referencia de la cara y 21 puntos de referencia de la mano).

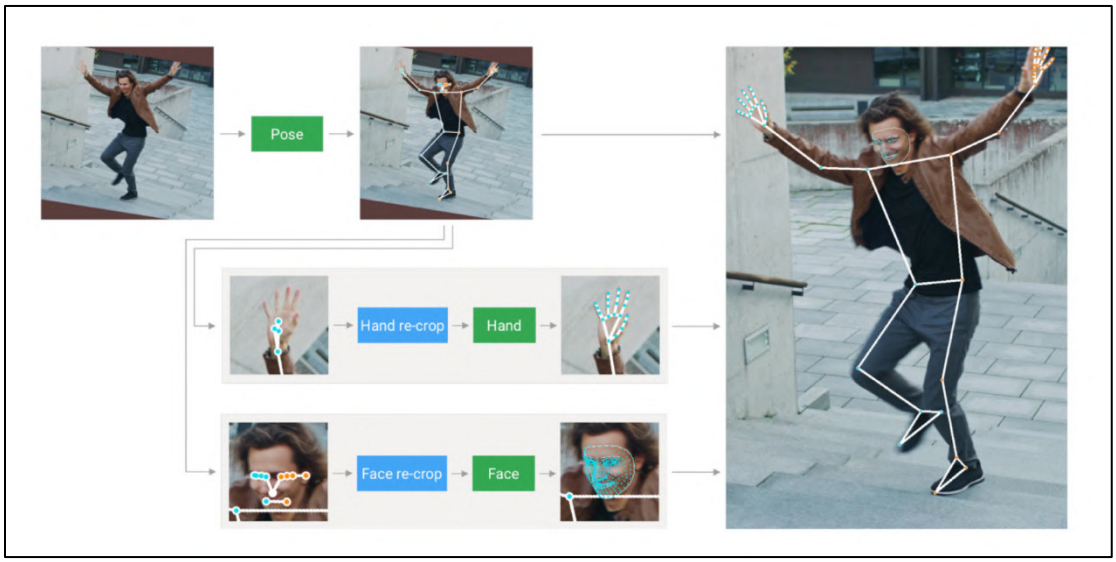


Figura 60. *Representación gráfica de proceso de ejecución del algoritmo MediaPipe Holístico.*

- **MediaPipe Hands**

MediaPipe Hands funciona en una variedad de tamaños de manos con un gran rango de escala (~20x) en relación con el marco de la imagen véase Figura 61. Lograr tener la percepción completa de las manos en tiempo real es una tarea de VA decididamente desafiante, ya que las manos a menudo se ocluyen a sí mismas o entre sí (por ejemplo, oclusiones de dedos/palmas y apretones de manos) y carecen de patrones de alto contraste[100].

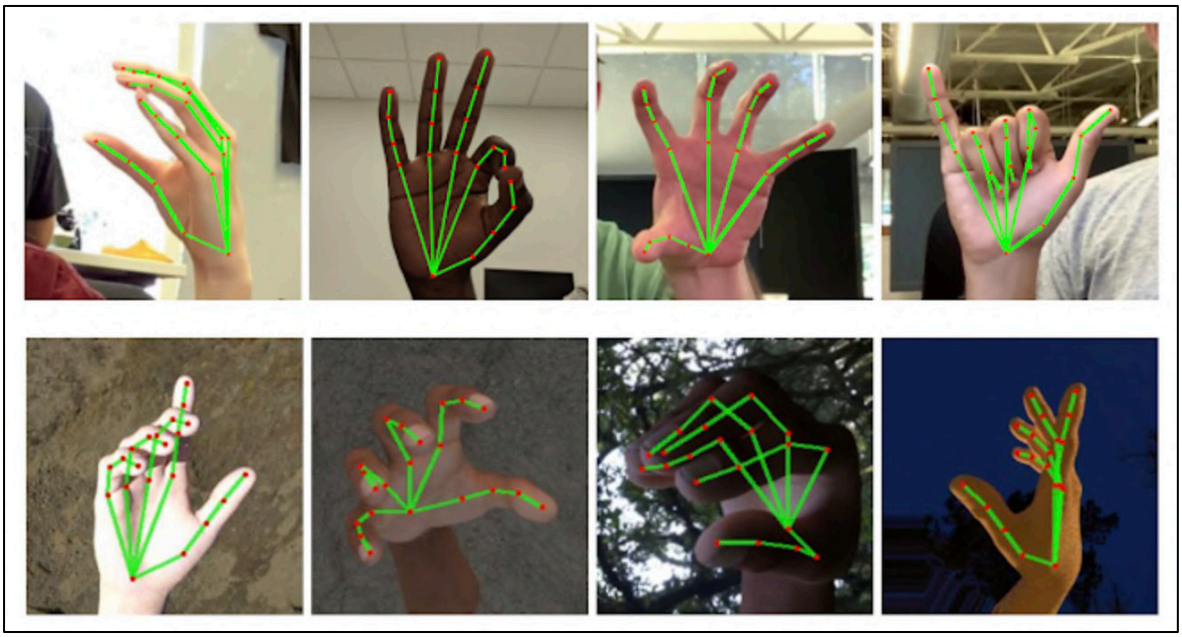


Figura 61. *Reconocimiento de manos con MediaPipe Hands.*

MediaPipe Hands es una solución de seguimiento de manos y dedos, que emplea el aprendizaje automático (ML) para inferir 21 puntos de referencia de una mano a partir de un solo fotograma. MediaPipe Hands utiliza un enfoque de tuberías de ML que consiste en múltiples modelos que trabajan entre si[100]:

1. Un modelo de detección de la palma de la mano opera sobre la imagen completa y devuelve un cuadro delimitador de la mano.
2. Un modelo de referencia de la mano trabaja sobre la región de la imagen recortada definida por el detector de la palma y devuelve los 21 puntos de referencia de los nudillos del mano ilustrado en la Figura 62, detectadas mediante regresión, es decir, predicción directa de coordenadas

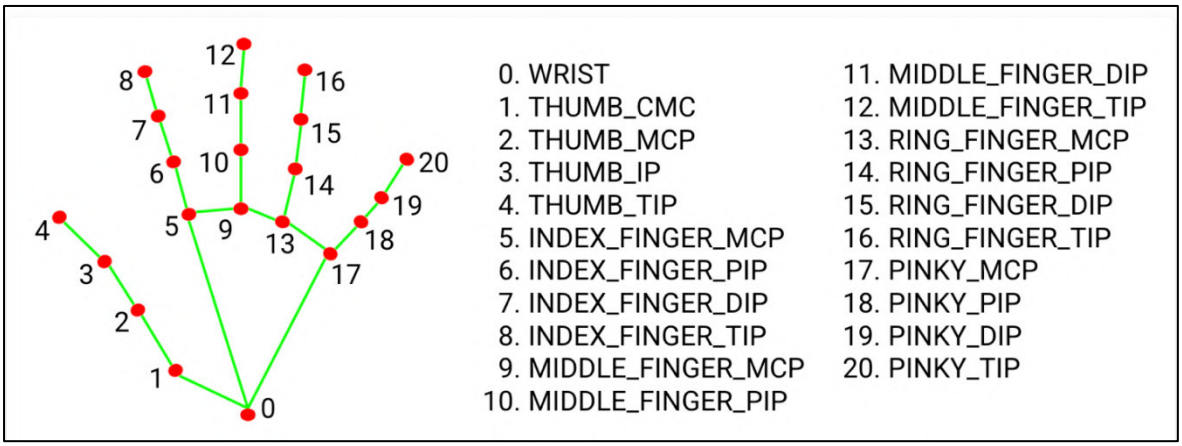


Figura 62. *21 puntos de referencia de los manos reconocidos por MediaPipe Hands.*

2.3.1. Aplicación del modelo MediaPipe Holístico

Dada una secuencia de fotogramas $\{F0, \dots Fn\}$, para cada frame previo a la aplicación del modelo se utiliza técnicas de preprocesamiento de imágenes, incluyen un redimensionamiento de imagen a dimensiones 640x480, un volteado de imagen, y una aplicación de capa de color COLOR_BGR2RGB.

Se aplica un modelo de MediaPipe Holístico y posteriormente el modelo MediaPipe Hands con la finalidad de obtener las coordenadas (x, y) de cada uno de los 21 puntos de referencia, siendo x el número de fila e y el número de la columna del punto con respecto a la matriz de pixeles, teniendo como salida la matriz de trayectoria W como entrada del modelo de TSF. En la Figura 63 ilustra el flujo de fotogramas a través del modelo con la presentación de los puntos de interés de cara, torso y manos.

Figura 63. *Aplicación del modelo MediaPipe Holistico en una secuencia de frames correspondientes a la gesticulación de la señal LEFT***.**

2.4. Aplicación de la factorización de la trayectoria espacial aplicada en el reconocimiento de señales

Luego de cumplir todas las etapas anteriormente mencionadas y con la matriz de trayectoria W conformada, con dimensión 2F x P donde F es la cantidad de frames de la secuencia y P el número los puntos de interés, se aplica el método de factorización de la trayectoria espacial de autor Akhter, factorizando W en el producto $R\ \Theta\ A$. Todo el procesamiento se lo realiza en un engine de la aplicación Matlab.

2.4.1. Análisis de la aplicación de TSF en la señal LEFT

El resultado de la aplicación de TSF es la recuperación total de la estructura 3D de la mano con sus 21 puntos de referencia, a continuación, se presentan las siguientes figuras que ilustran los resultados obtenidos:

- La Figura 64 muestra la trayectoria formada por el conjunto de coordenadas de la matriz W en una línea de tiempo al momento de realizar la señal LEFT, la gráfica demuestra que la señal mantiene un recorrido uniforme a lo largo de toda la secuencia.

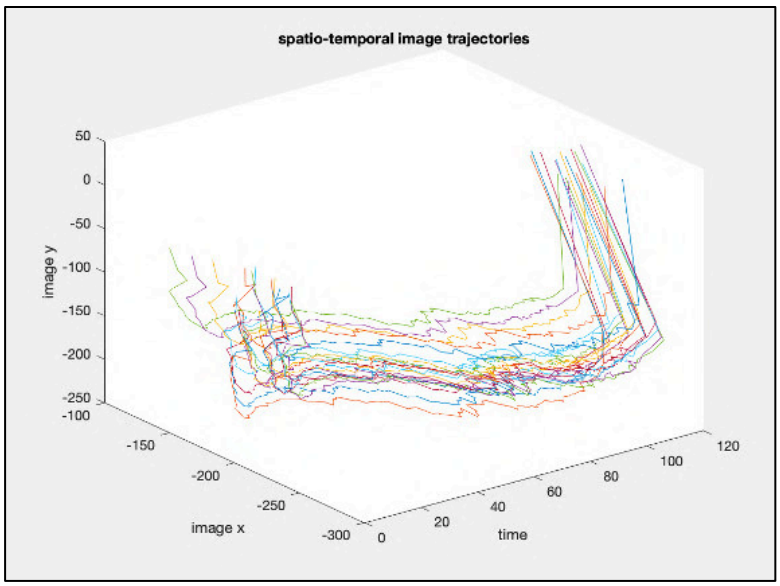


Figura 64. Trayectoria 2D de la señal LEFT.

- La Figura 65b ilustra una serie de imágenes que muestran las perspectivas de la estructura recuperada de la mano en 3D al momento de realizar la señal LEFT, al no tener oclusiones entre dedos se logra una mayor precisión en la recuperación de la forma. Se puede comparar las siluetas con la Figura 65a que indica la forma original de la señal en dos dimensiones.

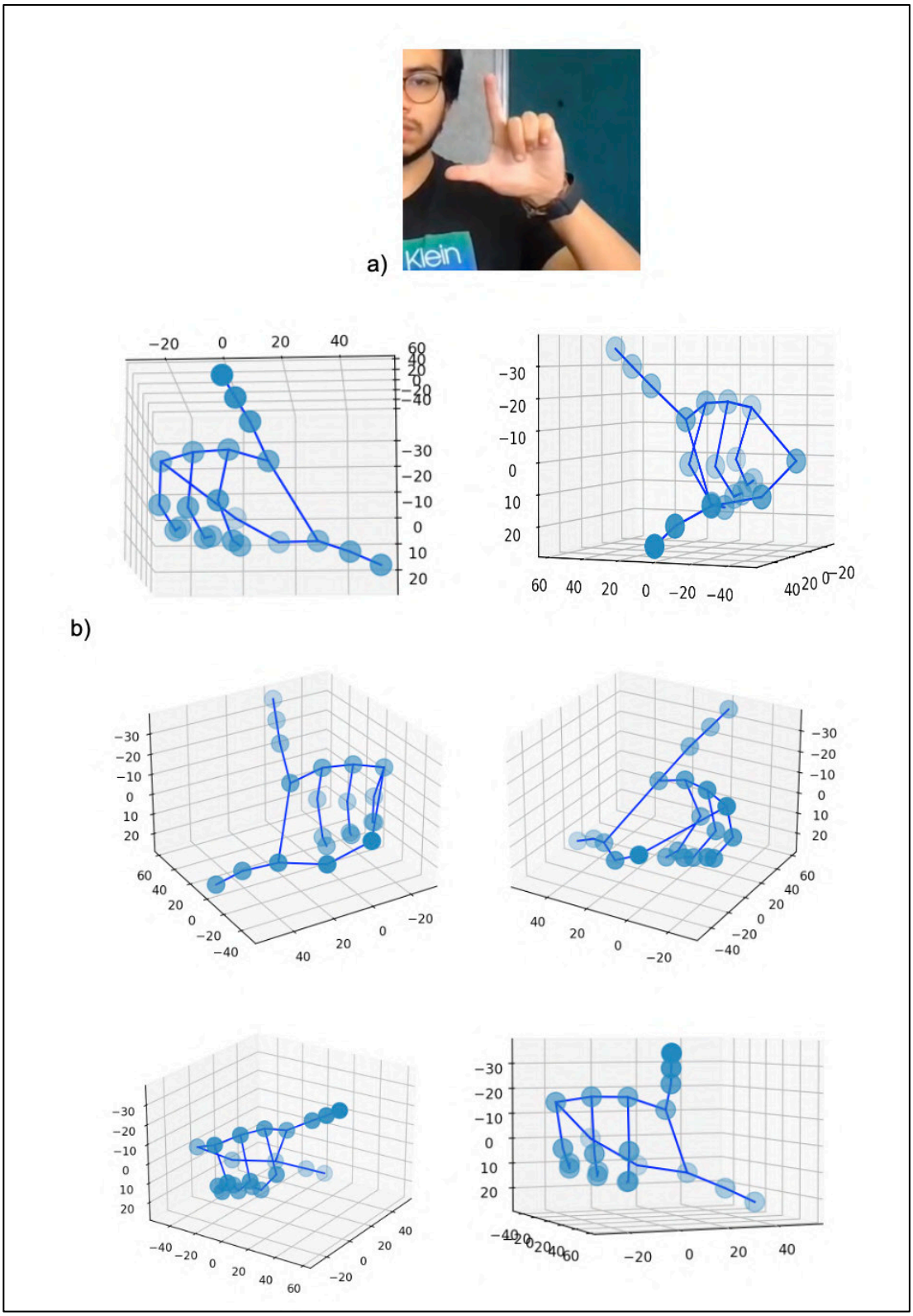


***Figura 65**. Perspectivas de estructura recuperada de la mano en 3D de la señal LEFT.*

2.4.2. Análisis de la aplicación de TSF en la señal RIGHT

El resultado de la aplicación de TSF es la recuperación total de la estructura 3D de la mano con sus 21 puntos de referencia, a continuación, se presentan las siguientes figuras que ilustran los resultados obtenidos:

- La Figura 66 muestra la trayectoria formada por el conjunto de coordenadas de la matriz W en una línea de tiempo al momento de realizar la señal RIGHT, la gráfica demuestra que la señal mantiene a lo largo de su recorrido, varios puntos intersectados debido a que la señal tiene varias coordenadas similares entre dedos que se encuentra solapados.

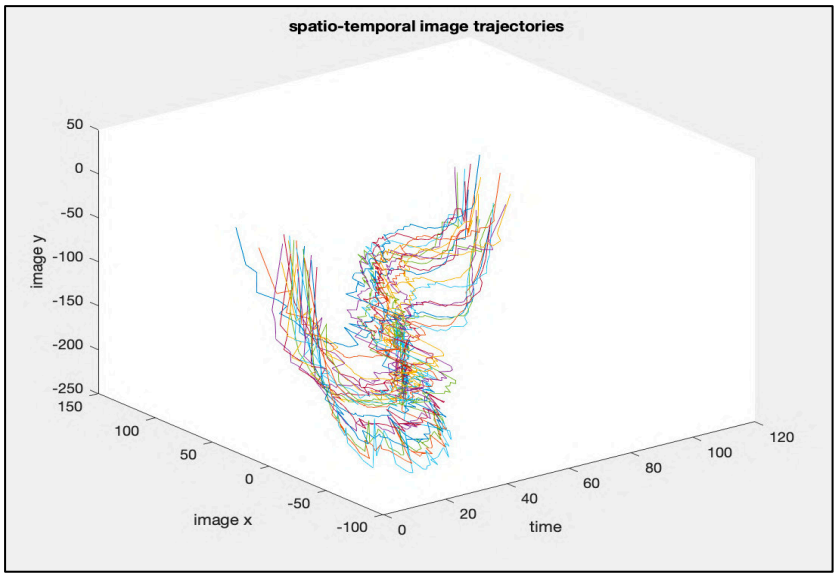
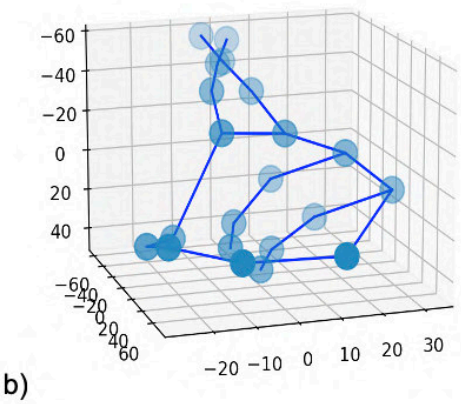


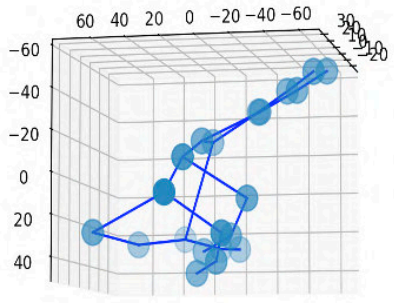
Figura 66. Trayectoria 2D de la señal RIGHT

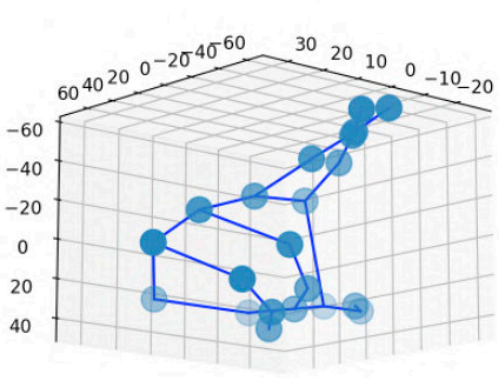
- La Figura 67b ilustra una serie de imágenes que muestran las perspectivas de estructura de la mano en 3D al momento de realizar la señal RIGHT, esta señal representa un reto de reconocimiento al presentar oclusiones entre el dedo índice y medio, además del solapamiento de los dedos restantes en la palma de la mano, lo cual dificultó la recuperación de la estructura, a pesar de esto la Figura 67a muestra que las siluetas tienen gran equivalencia con a la estructura original.

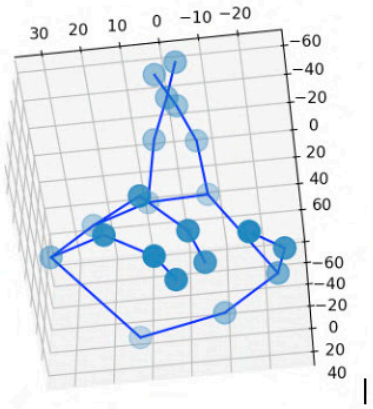
a)

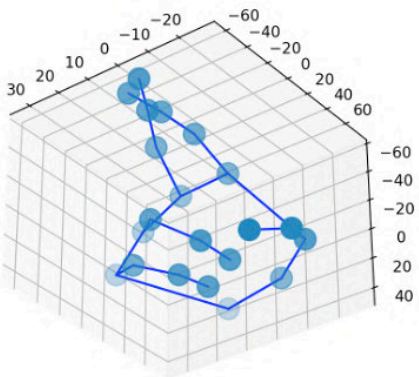
b)


−60
−40
−20
0
20
40
−60
−40
−20
0
20
40
60
−20 −10 0 10 20 30


60 40 20 0 −20 −40 −60
30
−60
−40
−20
0
20
40


60 40 20 0 −20 −40 −60
30 20 10 0 −10 −20
−60
−40
−20
0
20
40


30 20 10 0 −10 −20
−60
−40
−20
0
20
40
60
−60
−40
−20
0
20
40


30 20 10 0 −10 −20
−60 −40 −20 0 20 40 60
−60
−40
−20
0
20
40

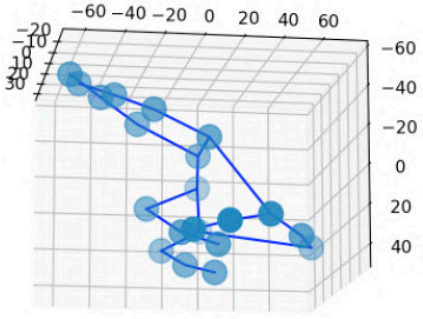

−60 −40 −20 0 20 40 60
−20
−10
0
10
20
30
−60
−40
−20
0
20
40

Figura 67. Perspectivas de estructura recuperada de la mano en 3D de la señal RIGHT.

2.4.3. Análisis de la aplicación de TSF en la señal TURN

El resultado de la aplicación de TSF es la recuperación total de la estructura 3D de la mano con sus 21 puntos de referencia, a continuación, se presentan las siguientes figuras que ilustran los resultados obtenidos:

- La Figura 68 muestra la trayectoria formada por el conjunto de coordenadas de la matriz W en una línea de tiempo al momento de realizar la señal TURN, la gráfica demuestra que al tener la señal un movimiento en giro circular se puede distinguir la forma de espiral en la línea de tiempo, además los puntos de coordenadas están intersectados debido a la oclusión del dedo pulgar con los dedos medio y anular, lo que ocasiona que las coordenadas mantengan valores en común.

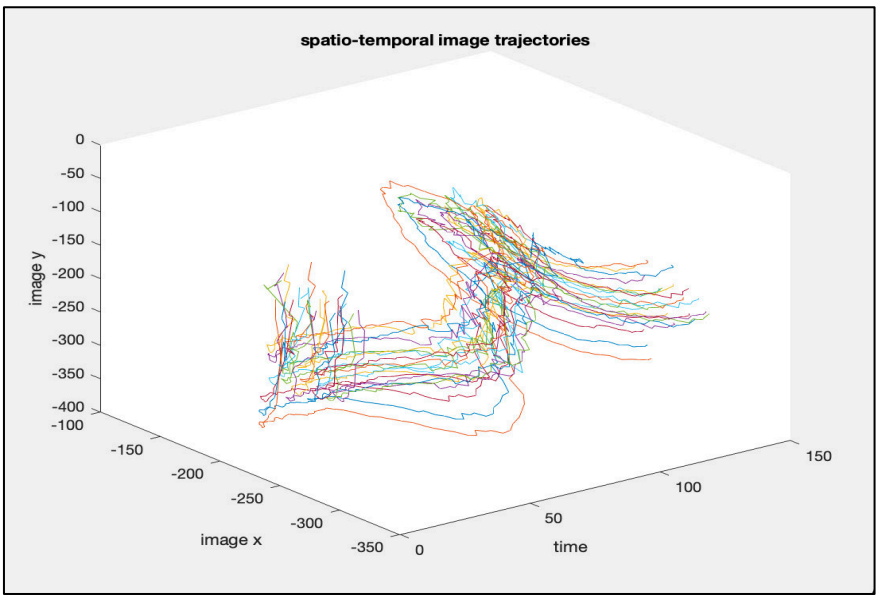
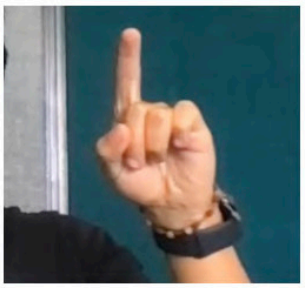


Figura 68. Trayectoria 2D de la señal TURN.

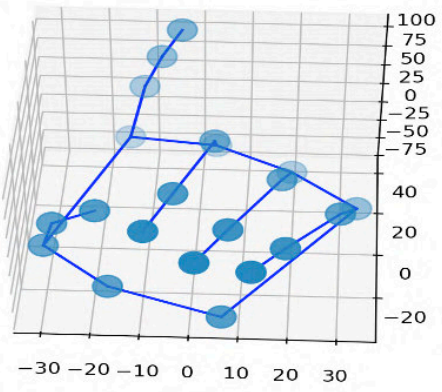
- La Figura 69b ilustra una serie de imágenes que muestran las perspectivas de la estructura recuperada de la mano en 3D al momento de realizar la señal TURN. Esta señal presento igual un reto de reconocimiento debido a

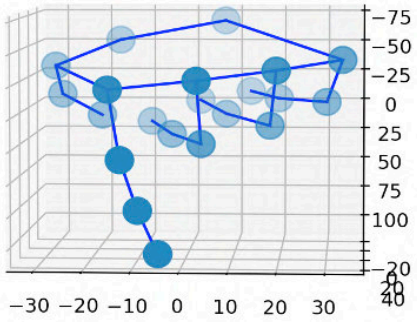
que la mano mantiene el puño casi cerrado y los dedos en su mayoría están juntos y solapados. Se puede comparar las siluetas con la Figura 69a, que verifica una buena recuperación en su forma equivalente a la original, a pesar de tener las dificultades anteriormente mencionadas.

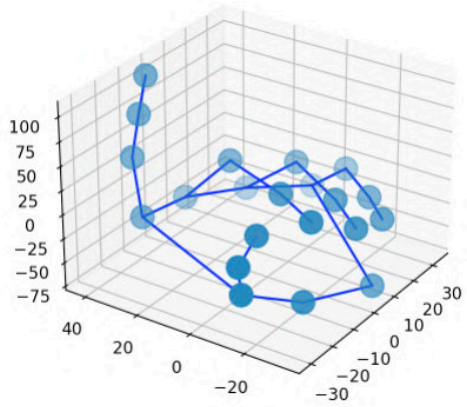
a)

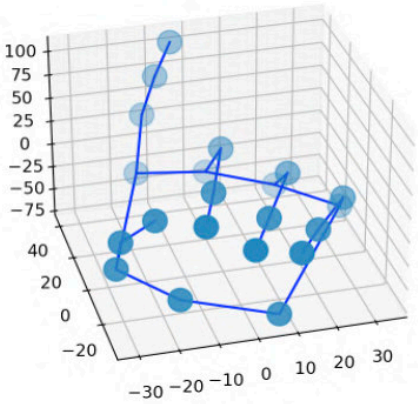


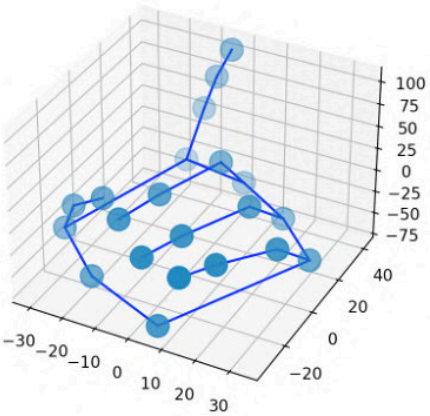
b)


100
75
50
25
0
−25
−50
−75
40
20
0
−20
−30 −20 −10 0 10 20 30


−75
−50
−25
0
25
50
75
100
−20
0
20
40
−30 −20 −10 0 10 20 30


100
75
50
25
0
−25
−50
−75
40
20
0
−20
−30
−20
−10
0
10
20
30


100
75
50
25
0
−25
−50
−75
40
20
0
−20
−30 −20 −10 0 10 20 30


100
75
50
25
0
−25
−50
−75
40
20
0
−20
−30 −20 −10 0 10 20 30

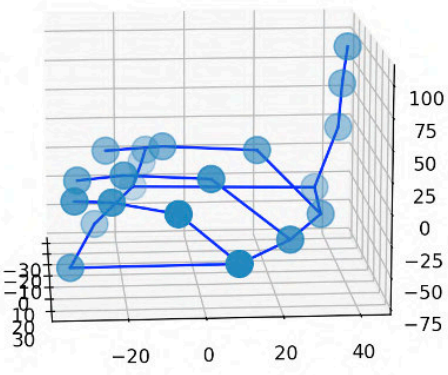

100
75
50
25
0
−25
−50
−75
−20
0
20
40

Figura 69. *Perspectivas de estructura recuperada de la mano en 3D de la señal TURN.*

2.4.4. Consideraciones de la aplicación TSF en señales

Al hacer casos repetitivos, las estructuras 3D recuperadas de cada señal pueden llegar ligeramente a variar, ya que dependen del recorrido de la trayectoria en el que se lo realizo, el tamaño de los dedos de cada persona, el tiempo de duración de la señal y si las manos del sujeto en cámara mantuvieron la forma correcta de la señal. Esta variación beneficia al reconocimiento de las señales ya que permite cubrir un mayor rango de posibilidades en el que una persona gesticule estas señales.

En secciones siguientes se intenta clasificar cada estructura 3D recuperada a partir de estas variaciones, con el objetivo de que el computador pueda automáticamente reconocerlas. Cada señal al momento de ser recuperada es representada computacionalmente por una matriz de dimensiones $p \times d$, en donde p es el número de puntos de referencia y del número de dimensiones. Al tener valores finitos la matriz para cada señal mantiene los valores $(21\ x\ 3)$, ilustrado en la Figura 70.

$$\begin{pmatrix} p1, d1 & \cdots & p1d3 \\ \vdots & \ddots & \vdots \\ p21, d1 & \cdots & p21, d3 \end{pmatrix}$$

Figura 70. *Representación computacional de la estructura 3D recuperada de una señal.*

2.5. Reducción de dimensionalidad

Como requerimiento de entradas en un modelo de CNN, así como para definir cada ítem del dataset utilizado en el entrenamiento y validación del modelo, es necesario mapear la representación computacional de la estructura 3D de cada señal ilustrada

en la Figura X(anterior), en una sola dimensión 1D, por lo cual es preciso utilizar un método de reducción de dimensiones.

El algoritmo t-SNE definido en la sección δ1.4.2.7, lleva los puntos en el espacio con alta dimensionalidad al espacio de baja dimensionalidad de forma aleatoria, definiendo una distribución de probabilidad sobre parejas de muestras en el espacio original, de forma que tal que las muestras semejantes reciben alta probabilidad de ser escogidas, mientras que las muestras muy diferentes reciben baja probabilidad de ser escogidas. El concepto de "semejanza" se basa en la distancia entre puntos y densidad en las proximidades de un punto. (cita) t-SNE intenta reproducir la distribución que existía en el espacio original en el espacio final. El resultado de la aplicación del algoritmo para cada señal es la obtención de un arreglo de la forma $[p1, \cdots p21] \in \mathbb{R}$. A continuación, en la Figura 71 se ilustra gráficamente la reducción de la dimensionalidad de los 21 puntos de referencia para cada señal.

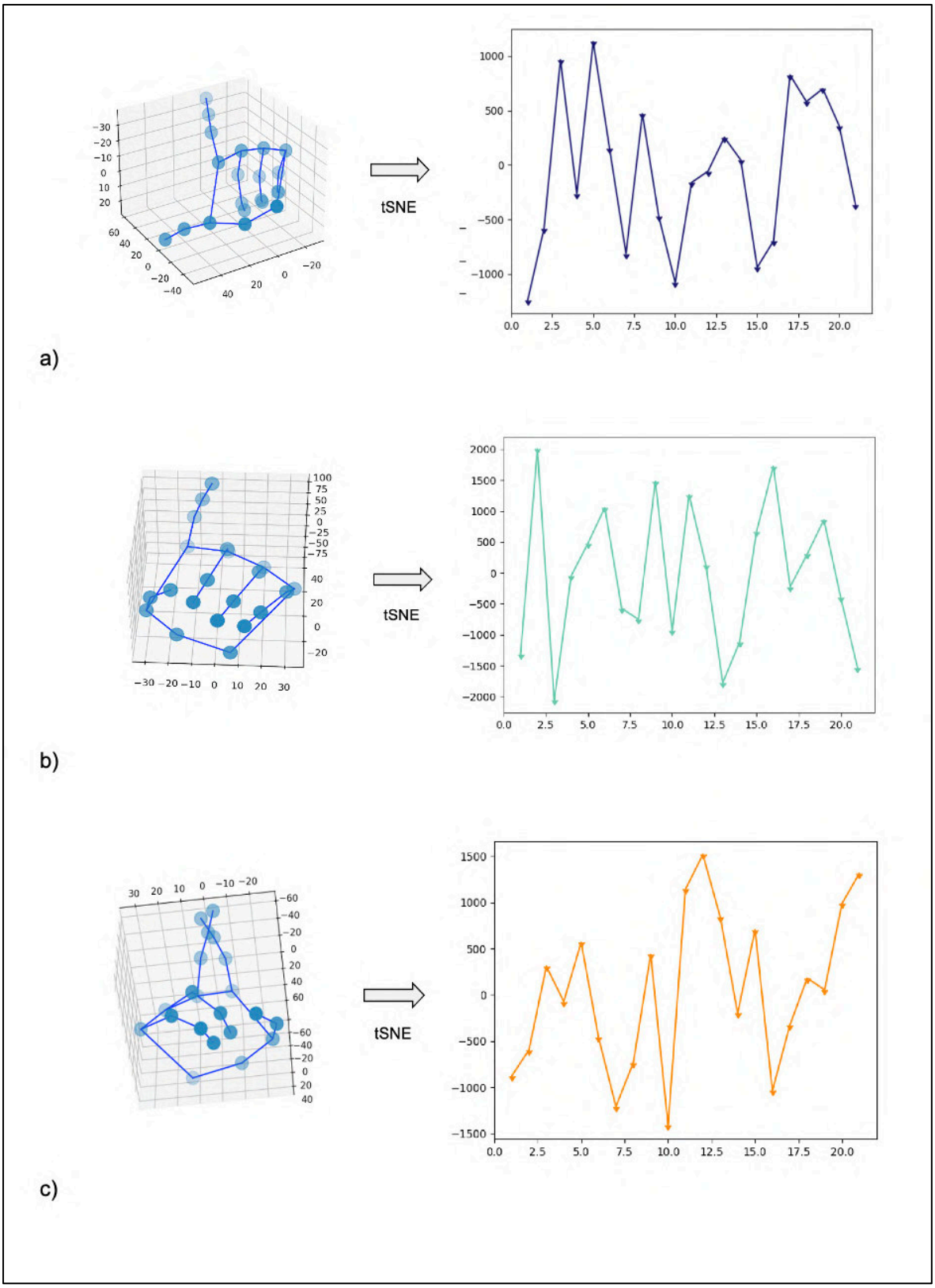

tSNE
a)
tSNE
b)
tSNE
c)

Figura 71. *a) Representación gráfica de la señal LEFT en una dimensión, b) representación gráfica de la señal RIGHT en una dimensión, c) representación gráfica de la señal TURN.*

2.6. Definición del dataset

Una vez reducida la dimensionalidad para cada representación computacional de la estructura 3D de las manos, los 21 puntos de referencia corresponden a las 21 columnas iniciales de cada ítem del dataset, en el cual se hará una adición de las siguientes variables:

1. Hand (hand): Indica la mano con la que fue realizada la señal $\{L, R\}$.
2. Label (label): Indica la etiqueta de la señal $\{LEFT, RIGHT, TURN\}$.

El dataset cuenta con dimensionalidad (23, 2398) respectivo a 23 columnas y 2398 filas, se encuentra en formato CSV y tiene un peso de 529.4 Kb. A continuación, la tabla 7 describe las columnas del dataset como variables y su descripción:

Columna	Variable	Descripción
1	hand	Indica si los puntos corresponden a la mano derecha o mano izquierda
2	palm_1	Representación unidimensional de la posición tridimensional del punto de referencia de la Muñeca 1
3	palm_2	Representación unidimensional de la posición tridimensional del punto de referencia de la Muñeca 2
4	thumb_1	Representación unidimensional de la posición tridimensional del punto de referencia Dedo pulgar: raíz
5	thumb_2	Representación unidimensional de la posición tridimensional del punto de referencia Dedo pulgar: primera articulación

6	thumb_3	Representación unidimensional de la posición tridimensional del punto de referencia Dedo pulgar: yema del dedo
7	index_1	Representación unidimensional de la posición tridimensional del punto de referencia Dedo índice: raíz
8	index_2	Representación unidimensional de la posición tridimensional del punto de referencia Dedo índice: segunda articulación
9	index_3	Representación unidimensional de la posición tridimensional del punto de referencia Dedo índice: primera articulación
10	index_4	Representación unidimensional de la posición tridimensional del punto de referencia Dedo índice: yema del dedo
11	middle_1	Representación unidimensional de la posición tridimensional del punto de referencia Dedo medio: raíz
12	middle_2	Representación unidimensional de la posición tridimensional del punto de referencia Dedo medio: segunda articulación

13	middle_3	Representación unidimensional de la posición tridimensional del punto de referencia Dedo medio: primera articulación
14	middle_4	Representación unidimensional de la posición tridimensional del punto de referencia Dedo medio: yema del dedo
15	ring_1	Representación unidimensional de la posición tridimensional del punto de referencia Dedo anular: raíz
16	ring_2	Representación unidimensional de la posición tridimensional del punto de referencia Dedo anular: segunda articulación
17	ring_3	Representación unidimensional de la posición tridimensional del punto de referencia Dedo anular: primera articulación
18	ring_4	Representación unidimensional de la posición tridimensional del punto de referencia Dedo anular: yema del dedo
19	little_1	Representación unidimensional de la posición tridimensional del punto de referencia Dedo meñique: raíz

20	little_2	Representación unidimensional de la posición tridimensional del punto de referencia Dedo meñique: segunda articulación
21	little_3	Representación unidimensional de la posición tridimensional del punto de referencia Dedo meñique: primera articulación
22	little_4	Representación unidimensional de la posición tridimensional del punto de referencia Dedo meñique: yema del dedo
23	label	Etiqueta de la señal correspondiente al entrenamiento supervisado

Tabla 7. *Descripción del dataset "Directional Signs ASL"*

2.7. Entrenamiento y testeo

La generación de un modelo idóneo para clasificar la representación unidimensional de la estructura 3D de las manos, debe afinarse hasta lograr tener un buen valor de clasificación, mientras más se aproximen al 100% de clasificación será considerado el adecuado para ser usado como clasificador. Los modelos de clasificación a comparar son una red neuronal recurrente (RNN) y máquinas de soporte de vectores (SVM). El porcentaje de exactitud de los dos modelos verifica cuan efectivo es cada uno. El número de ítems para los sets de entrenamiento y validación son:

- Número de muestras de entrenamiento = 1678.
- Número de muestras de validación = 720.

Las arquitecturas para el modelo de RNN es presentado en la Tabla 8:

Capa	Función de activación	Número de neuronas
0 (Input)	-	-
1	relu	20
2	relu	16
3	relu	12
4	relu	8
5	softmax	3
6 (output)	Softmax	-

Tabla 8. *Arquitectura del modelo de RNN propuesto.*

2.7.1. Entrenamiento

El entrenamiento de los dos modelos se realiza con los siguientes hiperparámetros iniciales para cada caso (véase Tabla 9 y 10)

- **RNN**

Hiperparámetro	Valor
Taza de aprendizaje	0.05
momentum	0.6
Tamaño de batch de entrada	5
Pérdida de peso	categorical_crossentropy

Total de épocas	150

Tabla 9. *Hiperparámetros del modelo de RNN.*

- **SVM**

Hiperparámetro	Valor
kernel	Lineal

Tabla 10. *Hiperparámetros del modelo de SVM.*

2.7.2. Testeo

Los resultados de la aplicación de cada uno de los modelos con los hiperparámetros mencionados anteriormente, muestran los siguientes resultados de exactitud (véase Tabla 11):

Modelo clasificador	Acurracy
CNN	72.50%
SVM	65.41%

Tabla 11. *Exactitudes de los modelos clasificadores propuestos.*

El modelo de RNN al ser el de mayor exactitud con un **72.50%**, es el escogido como el modelo clasificador del presente proyecto.

2.8. Implementación Sistema de Control de Dispositivos Domóticos

El sistema de visión artificial para el control de dispositivos domóticos o llamado en este documento como el sistema “CODDO” toma en cuenta las herramientas definidas en la sección (), esta implementado en el lenguaje de programación Python y utiliza su motor interprete para su ejecución, adicionalmente cuenta con un submódulo escrito y ejecutado en la plataforma MATLAB.

El módulo de control de dispositivos domóticos corresponde plenamente al uso del kit de desarrollo de las habilidades de Amazon Alexa del tipo SmartHome, y su llamada por medio de línea de comandos por medio de ASK CLI, utilizando un backend en la nube de AWS.

El sistema utiliza el modelo Cliente-Servidor, en donde el cliente es quien captura las imágenes de la secuencia de gesticulación de señas y las envía al servidor dedicado a VA con su módulo de TSF que procesa y envía una instrucción de control a la skill de Amazon Alexa. Debido a que los procesos del cliente y del servidor son realizados en tiempo real todos los pasos mencionados en la sección anterior deberán ser ejecutar indefinidamente en un bucle infinito hasta su interrupción. A continuación, se presenta un diagrama de flujo que detalla todas las actividades des sistema de visión artificial A continuación, se define el diseño del sistema con sus arquitecturas y se define los procesos y diagramas de flujo utilizados en la implementación.

2.8.1.1. Diseño de la arquitectura del sistema

- **Arquitectura General**

La Figura 72 muestra la arquitectura general del sistema CODDO aplicando el modelo Cliente-Servidor, con los procesos mencionados en la sección ∂2.8

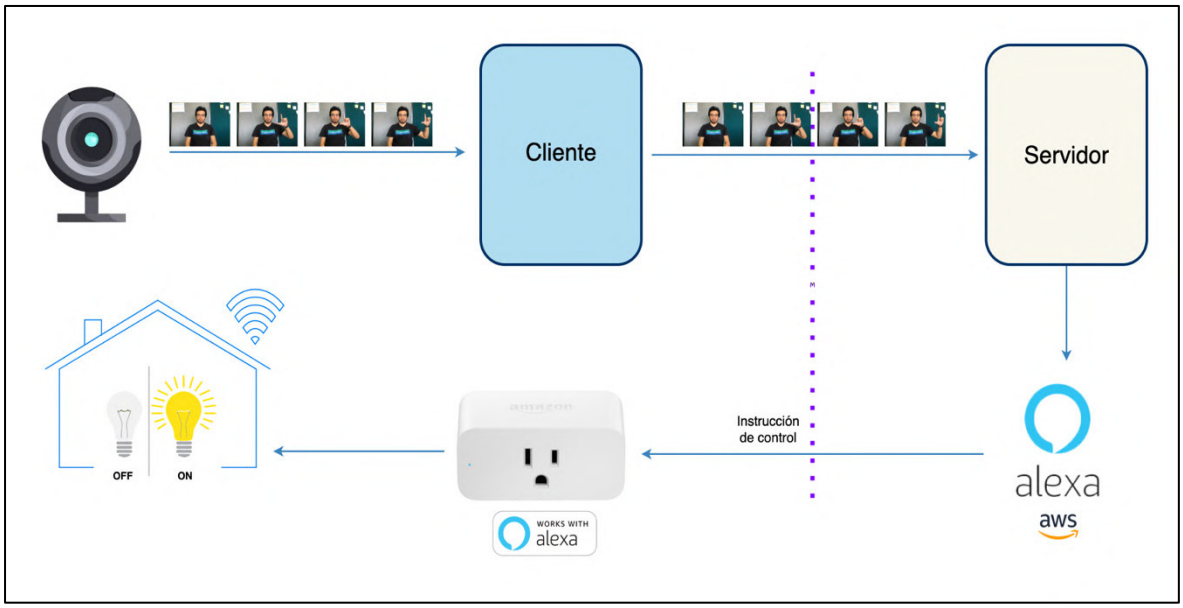


Figura 72. *Arquitectura general del Sistema CODDO.*

- **Arquitectura Cliente Servidor**

La Figura 73 muestra la arquitectura interna del cliente y del servidor del Sistema CODDO, que contienen librerías utilizadas y submódulos que contienen una mayor carga de procesamiento. Entre ellos destacan el Módulo de VA, y el módulo de RNN.

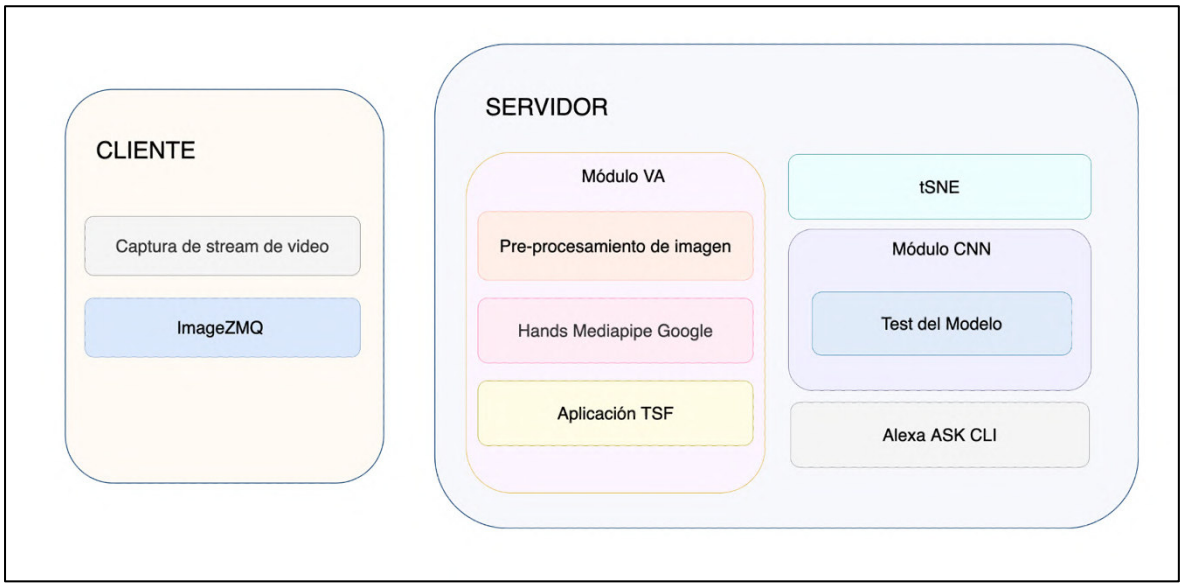


Figura 73. *Arquitectura interna del cliente y servidor del Sistema CODDO.*

- **Arquitectura Alexa Smart Home Skill**

La Figura 74 indica un modelo genérico de la arquitectura de una Skill de Amazon Alexa, con su backend alojado en la nube de AWS. La presente arquitectura es la utilizada para el desarrollo de este proyecto.

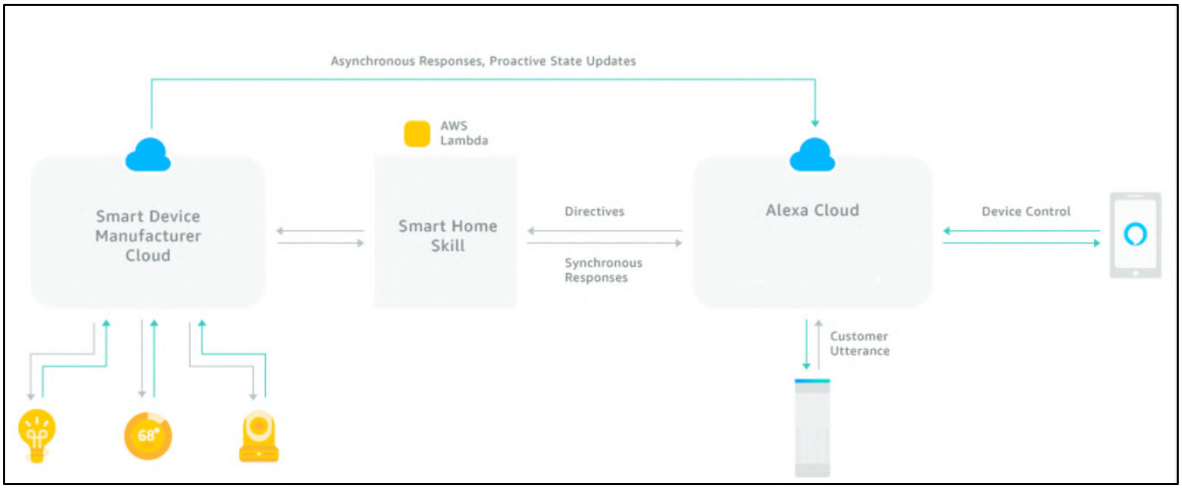


Figura 74. *Arquitectura Genérica de Alexa Smart Home Skill.*

2.8.1.2. Descripción de procesos y diagramas de flujos

A continuación, se describe las entradas y salidas de cada proceso del sistema CODDO, con una breve descripción de las tareas que se realizan. La tabla 12 muestra los procesos utilizados.

Proceso	Nombre	Descripción	Input	Output
1	Pre-procesamiento de Imagen	De una serie de frames (F) por cada frame (Fi) se hace un redimensionamiento de imagen a dimensiones 640x480, volteado de la imagen con cv2.split() y aplicación de capa de color COLOR_BGR2RGB	Numpy array de Fx1280 x 720	Numpy array Fx640x480

2	Detección y rastreo de puntos de referencias de las manos	Detección y rastreo de puntos de referencias de las manos P (x,y) por cada frame Fi por medio de librería "mediapipe" de Google, siendo un total de 21 puntos Pi	Numpy array Fx640x480	Numpy array FxPx2
	Generación de la matriz W	Generación de la Matriz W, en donde por cada frame Fi se duplica la fila correspondiente al punto de referencia 'x' y 'y'.	Numpy array FxPx2	Numpy array F*2xP
3	Aplicación de la factorización de la trayectoria espacial (Non-rigid TSF)	Aplicación de la factorización de la trayectoria espacial (Non-rigid TSF), en donde la entrada es una matriz W y el número decoeficiente DCT, siendo el resultado un arreglo multidimensional de tres dimensiones (R*3), con la aproximación a la estructura de la mano izquierda/derecha.	Numpy array F*2xP	Numpy array Rx3
4	Reducción de dimensiones con t-SNE	Reducción de cada conjunto de 21 puntos tridimensionales Rx3 en una sola dimensión D (21x1), utilizando Incrustación Stochastic Neighbor-t distribuida (tSNE , la cual es una técnica no lineal no supervisada utilizada principalmente para la exploración de datos y la visualización de datos de alta dimensión.	Numpy array Rx3	Numpy array D
5	Agregación de información	Agregado de información al array D, con los campos (number, hand, label)	Numpy array D	Numpy array D (21+3,1)

7	Clasificador RNN	Clasificador RNN que da como resultado el reconocimiento de instrucción gesticulada, entre las señales {LEFT, TURN, RIGHT}	Numpy array D (21+3,1)	String de instrucción
8	Alexa ASK CLI	Envio de instrucción de control a Alexa Smart Home Skill por medio de Alexa ASK CLI	String de instrucción	-

Tabla 12. *Descripción de procesos del Sistema CODDO.*

- **Diagrama de Flujo del Sistema de Visión Artificial CODDO**

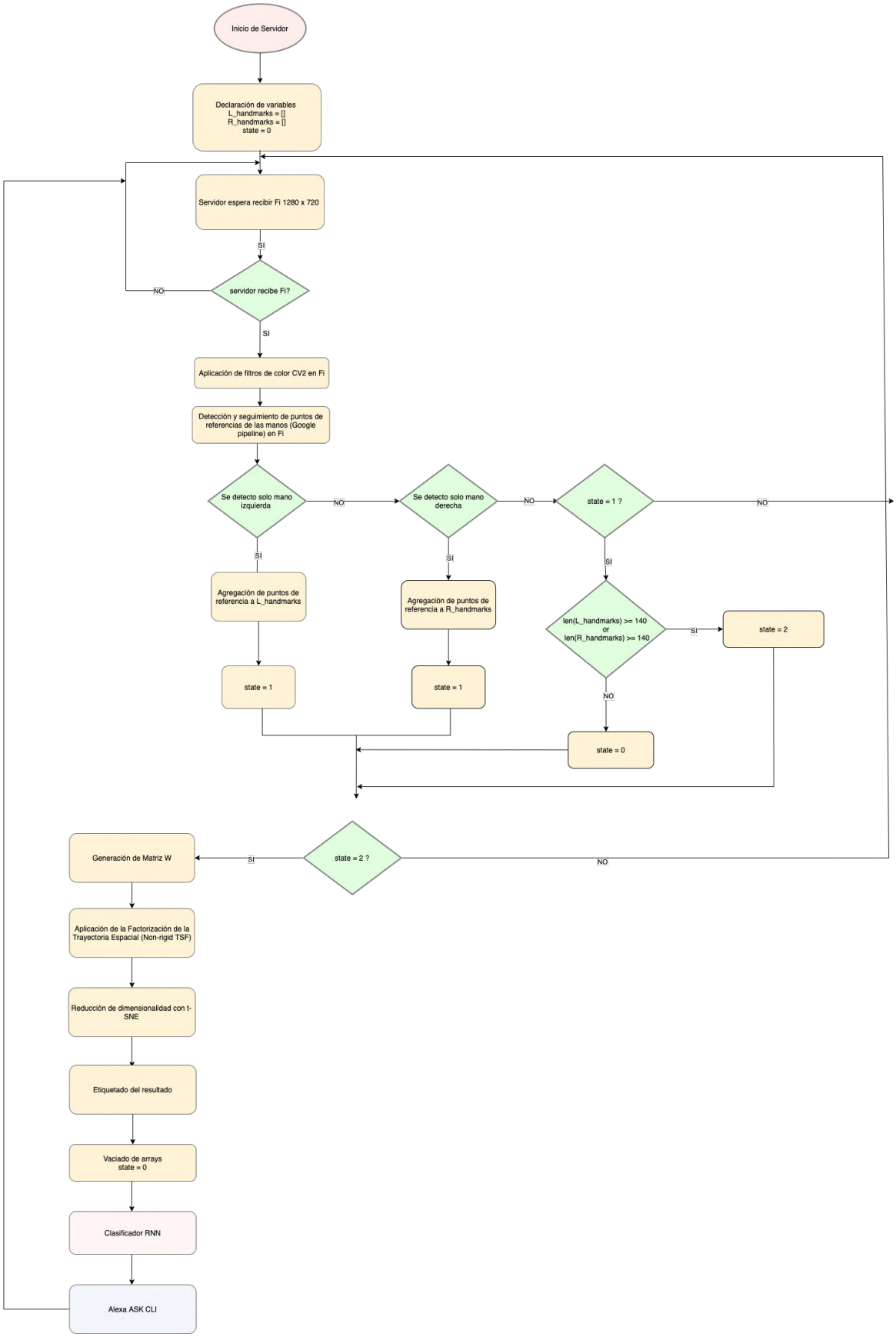

Inicio de Servidor
Declaración de variables
L_handmarks = []
R_handmarks = []
state = 0
Servidor espera recibir Fi 1280 x 720
SI
servidor recibe Fi?
NO
SI
Aplicación de filtros de color CV2 en Fi
Detección y seguimiento de puntos de referencias de las manos (Google pipeline) en Fi
Se detecto solo mano izquierda
NO
Se detecto solo mano derecha
NO
state = 1 ?
NO
SI
SI
SI
Agregación de puntos de referencia a L_handmarks
Agregación de puntos de referencia a R_handmarks
len(L_handmarks) >= 140
or
len(R_handmarks) >= 140
SI
state = 2
state = 1
state = 1
NO
state = 0
Generación de Matriz W
SI
state = 2 ?
NO
Aplicación de la Factorización de la Trayectoria Espacial (Non-rigid TSF)
Reducción de dimensionalidad con t-SNE
Etiquetado del resultado
Vaciado de arrays
state = 0
Clasificador RNN
Alexa ASK CLI

Figura 76. *Diagrama de flujo del servidor del sistema CODDO.*

En la Figura 75 se presenta el diagrama de flujo del servidor del sistema de visión artificial CODDO. En éste se muestran todas las acciones que se realizan desde su encendido e inicialización y demuestra el flujo indefinido que mantiene para atender las solicitudes del cliente.

- **Diagrama de Flujo del Sistema de Visión Artificial CODDO**

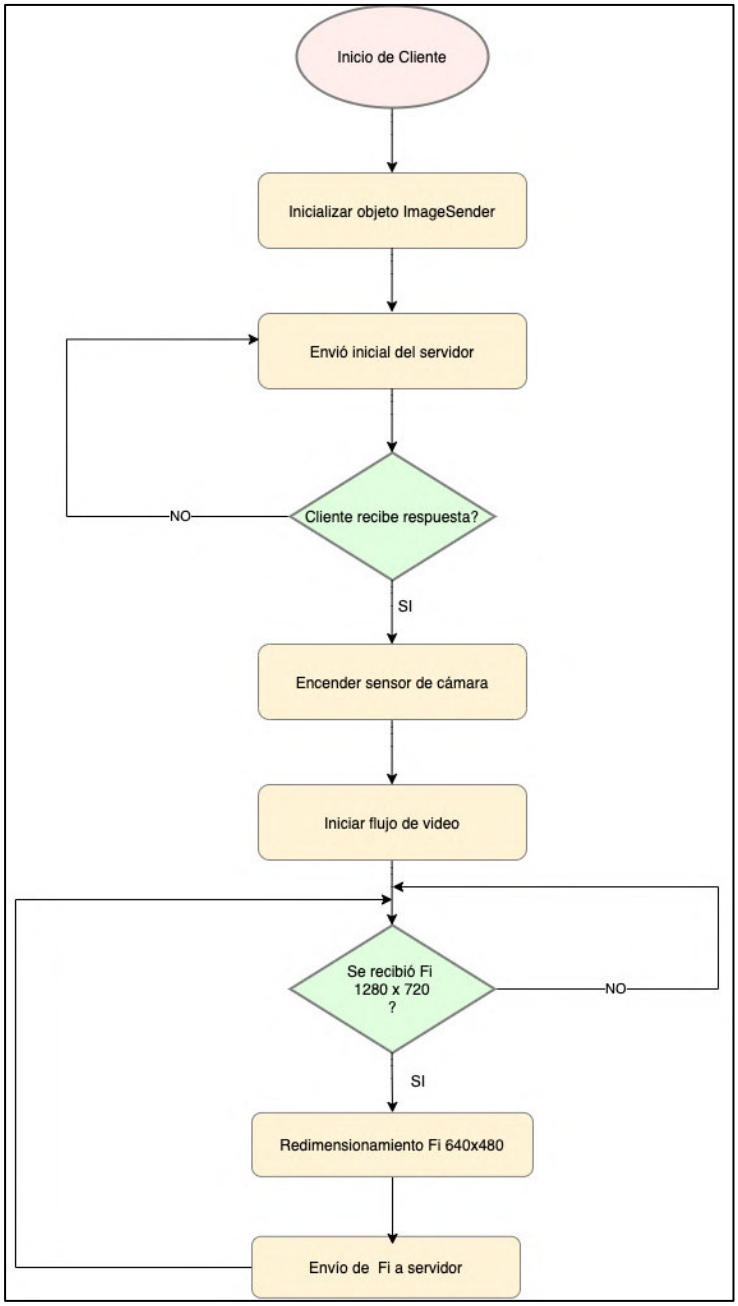


Figura 76. *Diagrama de flujo del cliente del sistema CODDO.*

En la Figura 76 se presenta el diagrama de flujo del cliente del sistema de visión artificial CODDO, en éste se muestran todas las acciones que se realizan desde su encendido e inicialización. El cliente presenta un comportamiento lineal secuencial indefinido, el cual recibe el flujo de imágenes desde el sensor de la cámara para enviarlas al servidor para su procesamiento.

RESULTADOS

La aplicación de la factorización de la trayectoria espacial fue exitosa, debido a que los puntos de referencia de manos reconocidos por medio de la aplicación del modelo de reconocimiento de manos de Google “Hands Mediapipe”[100] tiene una precisión media del 95,7% en la detección de manos. Sin embargo, después del uso de una pérdida de entropía cruzada normal y sin decodificador da una línea de base, tiene una exactitud del 86,22%.

La aplicación del modelo de red recurrente propuesto con la utilización de una validación cruzada de k plieges, dio como resultado una exactitud del 72.5%, por lo que se afirma la clasificación de señales en las instrucciones previamente definidas en la mayoría de los casos.

El sistema de control de dispositivos domóticos tiene en conjunto un rendimiento promedio del 78.73%, lo que significa que el sistema funciona correctamente en el 79.36% de los casos. El tiempo de respuesta desde la finalización de la gesticulación de la señal hasta la acción de la instrucción resultado del clasificador de RNN, con un promedio de 4.8 segundos.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se logro aplicar la factorización de la trayectoria espacial en el reconocimiento automático de señales basado en las consideraciones de la tesis doctoral del autor Mark Borg[6], en la que se reconstruye exitosamente la estructura 3D de las manos a partir de una secuencia de imágenes en 2D capturadas en tiempo real desde una cámara web, para cada instrucción del conjunto definido en la sección ∂2.2.
- Las condiciones definidas para la gesticulación de señas de un sujeto en cámara mencionadas en la sección ∂2.2.3 aseguraron un mayor umbral de reconocimiento para cada video del conjunto de datos de entrenamiento, validación y testeo.
- El dataset creado para la ejecución de este proyecto “Directional Signs ASL” tuvo un total de 2432 videos. Consiguió entrenar, validar y testear un modelo de red recurrente, usado para la clasificación de la representación de dimensionalidad reducida de la estructura 3D de las manos en las clases $\{LEFT, TURN, RIGHT\}$. El modelo de red recurrente tuvo mayor exactitud, en comparación con el modelo de máquinas de vectores de soporte.
- El sistema de control de dispositivos domóticos CODDO, logró el control de dispositivos domóticos por medio del asistente virtual Amazon Alexa con el uso de su funcionalidad Amazon Smart Home Skill, el cual recibe y ejecuta una instrucción a partir de la traducción automática de señales.
- La manipulación de interruptores de luz inteligentes fue posible mediante el control del dispositivo terminal inteligente Amazon Smart Home, el cual respondió exitosamente en cuanto se realizó una señal especifica de ejecución, específicamente su apagado con la traducción automática de la señal RIGHT y encendido con la traducción automática de la señal LEFT.

4.2. Recomendaciones

Para futuros trabajos, el autor del presente proyecto recomienda:

- La obtención de una masiva cantidad de imágenes necesarias para el entrenamiento de un modelo de ML puede llegar a ser muy demandante en tiempo e imposible para la realización de una persona, por lo que siempre es recomendable utilizar conjuntos de datos existentes y disponibles para su uso, siempre y cuando se ajusten a las necesidades del proyecto.
- La falta de compatibilidad entre versiones de diferentes frameworks utilizados en el presente proyecto pueden llegar a demandar un análisis exhaustivo del historial de versiones, y errores presentados en el transcurso de sus ejecuciones, por lo que utilizar versiones estables y de mayor compatibilidad asegura en conjunto con un código legible, un prototipo más estable y sin errores.
- El Hardware necesario para el entrenamiento, validación y ejecución de redes neuronales de gran procesamiento puede llegar a ser de alto costo, o de imposible comparación con computadores de gama media, o de uso estudiantil. Por lo que llegar a una cooperación con instituciones como la propia universidad, o exteriormente, las cuales brinden acceso a sus arquitecturas como supercomputadores o cluster, hace más alcanzable la ejecución de este tipo de proyectos.
- El prototipo del sistema fue evaluado en situaciones mayoritariamente ideales en las que solo existe una persona en cámara, o evitando la mayoría de las restricciones mencionadas en la sección δ2.2.3, por lo que un modelo más aplicable en el mundo real deberá tener un menor y reducido número de limitantes.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. J. Rueda Lopez, “LA TECNOLOGÍA EN LA SOCIEDAD DEL SIGLO XXI: ALBORES DE UNA NUEVA REVOLUCIÓN INDUSTRIAL,” *Aposta*, vol. 32, no. Ciencias Sociales, p. 29, 2007.

[2] A. C. Luis, T. E. M. Elizabeth, F. V. Jesús, R. U. M. Deyanira, and A. S. J., “Interacción Humano-Computadora,” 2017, pp. 195–232.

[3] L. Alba Montoya, “Comunicación Oral y Escrita,” *2015*, 2015. https://slideplayer.es/slide/5842541/ (accessed Sep. 07, 2021).

[4] Organización Mundial de la Salud, “Sordera y pérdida de la audición,” *2021*, Mar. 02, 2021. https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss (accessed Sep. 07, 2021).

[5] L. E. Sucar and G. Gómez, “Vision Computacional,” *Inst. Nac. Astrofísica, Óptica y Electrónica*, p. 185, 2011, [Online]. Available: http://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf.

[6] M. Borg, “Trajectory Space Factorisation for Vision-Based Automated Sign Language Recognition,” *Univ. Malta*, vol. One, no. Computer Vision, p. 350, 2020.

[7] I. G. S. Benavides, “Lenguaje De Señas Entre Niños Sordos De Padres Sordos Y Oyentes.”

[8] National Institute on Deafness and Other Communication Disorders, “What Is American Sign Language (ASL)? | NIDCD,” Mar. 2019. https://www.nidcd.nih.gov/health/american-sign-language (accessed Sep. 08, 2021).

[9] National Institute on Deafness and Other Communication Disorders, “American Sign Language,” *NIDCD*, Feb. 2014, Accessed: Sep. 08, 2021. [Online]. Available: http://www.nidcd.nih.gov/.

[10] D. Vernon, *Machine Vision*, vol. 37, no. 2. London: Prentice Hall International, 1991.

[11] G. De Catalunya, “Aplicación práctica de la visión artificial en el control de procesos

industriales," 2011.

[12] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, Analysis, and Machine Vision-Cengage Learning*, Fourth. CENGAGE Learning, 2014.

[13] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*. 1995.

[14] R. Szeliski, "Computer vision: Algorithms and Applications," *Comput. Sci. Handbook, Second Ed.*, pp. 43-1-43–23, 2004, doi: 10.22214/ijraset.2021.35926.

[15] V. Balicevic and P. Pavle, "Digital image processing and analysis Laboratory 6 : Detection of image features," pp. 3–8, 2014.

[16] S. Vijayaraghavan, "Digital Image Processing," *SCSVMV Deemed Univ.*, pp. 1–118, 1998.

[17] K. M. Iftekharuddin and A. Awwal, "Sampling and Quantization," *Sisu@UT*, no. i, pp. 5–5, 2012, doi: 10.1117/3.923354.ch5.

[18] T. Fukunaga, "Introduction to 2D image processing," *Nihon Hoshasen Gijutsu Gakkai Zasshi*, vol. 69, no. 1, pp. 129–134, 2013, doi: 10.6009/jjrt.2013_JSRT_69.1.129.

[19] L. Walkling, "Introduction to Digital Resolution," *Leswalking.Com*, p. 11, 2011.

[20] M. Kharinov, "Information quantity in a pixel of digital image," *Russ. Acad. Sci.*, no. February, 2014, [Online]. Available: http://arxiv.org/abs/1401.7517.

[21] P. West, "Fundamentals of machine vision lighting," *Proc. WESCON 1993 Conf. Rec.*, no. 408, pp. 44–52, 1993, doi: 10.1109/WESCON.1993.488407.

[22] S. Sangwine and R. Horne, *The Colour Image Processing Handbook*. 1998.

[23] F. Minardi, "Visión Artificial," 2017.

[24] F. R. Leta, F. F. Feliciano, I. L. d. Souza, and E. Cataldo, "Discussing accuracy in an automatic measurement system using computer vision techniques," *ABCM Symp. Ser. Mechatronics*, vol. 2, pp. 645–652, 2006, [Online]. Available: http://www.abcm.org.br/pt/wp-content/symposium-series/SSM_Vol2/Section_X_Computer_Vision/SSM2_X_03.pdf.

[25] G. J. Agin, “Computer Vision Systems for Industrial Inspection and Assembly.,” *Instrum. Pulp Pap. Ind. Proc.*, vol. 1, no. May, pp. 1–7, 1980.

[26] T. Lindeberg, “Scale-Space,” *Wiley Encycl. Comput. Sci. Eng.*, vol. 609, no. Sep 2008, pp. 2495–2504, 2008, doi: 10.1002/9780470050118.ecse609.

[27] A. Cunha, R. Teixeria, and L. Velho, “Discrete Scale Spaces via Heat Equation,” *IMPA–Instituto Matemática Pura e Apl.*

[28] B. Zhao, L. Fei-Fei, and E. P. Xing, “Image segmentation with topic random field,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6315 LNCS, no. PART 5, pp. 785–798, 2010, doi: 10.1007/978-3-642-15555-0_57.

[29] H. Xin, H. Ai, H. Chao, and D. Tretter, “Human head-shoulder segmentation,” *2011 IEEE Int. Conf. Autom. Face Gesture Recognit. Work. FG 2011*, no. May 2014, pp. 227–232, 2011, doi: 10.1109/FG.2011.5771402.

[30] L. J. Li, H. Su, E. P. Xing, and L. Fei-Fei, “Object Bank: A high-level image representation for scene classification & semantic feature sparsification,” *Adv. Neural Inf. Process. Syst. 23 24th Annu. Conf. Neural Inf. Process. Syst. 2010, NIPS 2010*, no. January, 2010.

[31] L. Liu, J. Xing, and H. Ai, “Multi-view vehicle detection and tracking in crossroads,” *1st Asian Conf. Pattern Recognition, ACPR 2011*, no. November 2015, pp. 608–612, 2011, doi: 10.1109/ACPR.2011.6166688.

[32] Y. Bengpeng and A. Haizhou, “MATCHING TEXTURE UNITS FOR FACE RECOGNITION,” *Sci. Technol.*, pp. 1920–1923, 2008.

[33] D. Walther, “Machine Vision,” *Cambridge Handb. Learn. Sci.*, pp. 1–3, 2015.

[34] L. J. Li and L. Fei-Fei, “What, where and who? Classifying events by scene and object recognition,” *Proc. IEEE Int. Conf. Comput. Vis.*, no. May, 2007, doi: 10.1109/ICCV.2007.4408872.

[35] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei, “Object-centric spatial pooling for image classification,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7573 LNCS, no. PART 2, pp. 1–15, 2012, doi:

10.1007/978-3-642-33709-3_1.

[36] S. Savarese and L. Fei-Fei, “3D generic object categorization, localization and pose estimation,” *Proc. IEEE Int. Conf. Comput. Vis.*, 2007, doi: 10.1109/ICCV.2007.4408987.

[37] O. Theobald, “Machine Learning for Absolute Beginners,” 2017.

[38] R. Russell, “Machine learning step-by-step guide to implement machine learning algorithms with Python,” p. 106, 2018, [Online]. Available: http://booksdescr.org/item/index.php?md5=D161EE832B8007A058CD006DD67E388E.

[39] A. Géron, *Hands-on Machine Learning*, vol. 53, no. 9. 2017.

[40] E. Alpaydin, *Introduction to Machine Learning*, Second. Cambridge, Massachusetts: The MIT Press, 2010.

[41] R. Sathya and A. Abraham, “Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification,” *Int. J. Adv. Res. Artif. Intell.*, vol. 2, no. 2, 2013, doi: 10.14569/ijarai.2013.020206.

[42] IBM, “Supervised vs. Unsupervised Learning: What’s the Difference? | IBM,” 2021. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning (accessed Oct. 24, 2021).

[43] D. Soni, “Supervised vs. Unsupervised Learning,” *towards data science*, 2018. https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d (accessed Oct. 24, 2021).

[44] F. Chollet, *Deep Learning with Python*. 2018.

[45] T. Beysolow II, *Introduction to Deep Learning Using R*. 2017.

[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016.

[47] S. Haykin, *Neural Networks and Learing Machines*, vol. 1–3. Ontario, Canada: Pearson Education, 2018.

[48] I. Livshin, *Artificial Neural Networks with Java*. 2019.

[49] K. Mehrotra, C. Mohan, and S. Ranka, “Elements of Artificial Neural Networks,” *Elem. Artif. Neural Networks*, no. May, 2019, doi: 10.7551/mitpress/2687.001.0001.

[50] C. Gershenson, “Artificial Neural Networks for Beginners,” no. September 2003, 2003, [Online]. Available: http://arxiv.org/abs/cs/0308031.

[51] S. Haykin, “Adaptive signal processing,” *Proceedings of the IEEE Ultrasonics Symposium*, vol. 1. pp. 635–638, 1997, doi: 10.1201/b14904-24.

[52] H. Simon and B. Van Veen, *Signal and Systems*. Jhon Wiley & Sons, Inc., 2002.

[53] C. Gallo, “Artificial neural networks,” *Stud. Syst. Decis. Control*, vol. 131, no. July, pp. 11–35, 2018, doi: 10.1007/978-3-319-75049-1_2.

[54] J. Jiang, P. Trundle, and J. Ren, “Medical image analysis with artificial neural networks,” *Comput. Med. Imaging Graph.*, vol. 34, no. 8, pp. 617–631, 2010, doi: 10.1016/j.compmedimag.2010.07.003.

[55] R. Hristev, *The ANN book*, 0 ed. 1998.

[56] J. Mao and A. Jain, “Artificial Neural Networks: A tutorial,” 1996.

[57] D. Jared, *Big data, data mining, and Machine Learning*. 2014.

[58] P. Marius-Constantin, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, “Multilayer perceptron and neural networks,” *WSEAS Trans. Circuits Syst.*, vol. 8, no. 7, pp. 579–588, 2009.

[59] J. A. Bullinaria, “Recurrent Neural Networks,” pp. 1–20, 2015, [Online]. Available: http://www.cs.bham.ac.uk/~jxb/INC/l12.pdf.

[60] K. L. Du and M. N. S. Swamy, *Recurrent Neural Networks*, vol. 9781447155, no. April 2016. 2014.

[61] K. M. Ting, “Confusion Matrix,” *Encycl. Mach. Learn. Data Min.*, pp. 1–1, 2016, doi: 10.1007/978-1-4899-7502-7_50-1.

[62] K. Gurney, *Introduction to neural networks*, vol. 346, no. 8982. 1995.

[63] L. Tang, H. Lu, and P. Refaeilzadeh, “Cross Validation,” pp. 1–8, 2008.

[64] Interactive Chaos, “t-SNE .” https://interactivechaos.com/es/manual/tutorial-de-machine-learning/t-sne (accessed Nov. 02, 2021).

[65] L. Van Der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Ann. Oper. Res.*, 2008.

[66] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade, “Nonrigid structure from motion in trajectory space,” *Adv. Neural Inf. Process. Syst. 21 - Proc. 2008 Conf.*, vol. 1, pp. 42–48, 2009.

[67] H. Shakirat Shakirat, “Client-Server Model,” *IOSR J. Comput. Eng.*, vol. 16, no. 1, pp. 57–71, 2014, doi: 10.9790/0661-16195771.

[68] Y. Subhash Chandra and S. Kumar Singh, *An Introduction to Client/Server Computing*, vol. 148. Varanasi: New Age International Limited Publishers, 2009.

[69] “Client-server Architecture,” [Online]. Available: https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/ClientServer.pdf.

[70] H. Zhang, “Architecture of Network and Client-Server model,” *Univ. Sci. Technol. China*, pp. 1–3, 2013, [Online]. Available: http://arxiv.org/abs/1307.6665.

[71] P. By and C. Kambalyal, “N-Tier Architecture,” *Pro LINQ Object Relational Mapp. with C# 2008*, pp. 311–345, 2008, doi: 10.1007/978-1-4302-0597-5_11.

[72] Python, “What is Python? Executive Summary | Python.org.” https://www.python.org/doc/essays/blurb/ (accessed Sep. 07, 2021).

[73] MATLAB, “What Is MATLAB? - MATLAB & Simulink.” https://www.mathworks.com/discovery/what-is-matlab.html (accessed Sep. 07, 2021).

[74] Anaconda, “Anaconda Individual Edition — Anaconda documentation.” https://docs.anaconda.com/anaconda/index.html (accessed Sep. 07, 2021).

[75] Jupyter, “The Jupyter Notebook.” https://jupyter.org/ (accessed Sep. 07, 2021).

[76] Microsoft, “Documentation for Visual Studio Code.”

https://code.visualstudio.com/docs (accessed Sep. 07, 2021).

[77] IBM, “Formato JSON (JavaScript Object Notation).” https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format (accessed Sep. 07, 2021).

[78] Amazon, “What is AWS.” https://aws.amazon.com/what-is-aws/ (accessed Sep. 07, 2021).

[79] Amazon, “AWS Identity & Access Management - Amazon Web Services.” https://aws.amazon.com/iam/ (accessed Sep. 08, 2021).

[80] Amazon, “Internet of Things (IoT) - Overview of Amazon Web Services.” https://docs.aws.amazon.com/whitepapers/latest/aws-overview/internet-of-things-services.html (accessed Sep. 08, 2021).

[81] Amazon, “What is AWS Lambda? - AWS Lambda.” https://docs.aws.amazon.com/lambda/latest/dg/welcome.html (accessed Sep. 08, 2021).

[82] Amazon, “What is the Alexa Skills Kit? | Alexa Skills Kit.” https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html (accessed Sep. 08, 2021).

[83] ProgrammableWeb, “Amazon Alexa Smart Home Skills REST API .” https://www.programmableweb.com/api/amazon-alexa-smart-home-skills-rest-api (accessed Sep. 08, 2021).

[84] Amazon, “Quick Start: Alexa Skills Kit Command Line Interface (ASK CLI) | Alexa Skills Kit.” https://developer.amazon.com/en-US/docs/alexa/smapi/quick-start-alexa-skills-kit-command-line-interface.html (accessed Sep. 08, 2021).

[85] Amazon, “How to use an Amazon Smart Plug.” https://www.amazon.com/b?ie=UTF8&node=21341313011 (accessed Sep. 08, 2021).

[86] J. Bass, “imageZMQ: Transporting OpenCV images.” https://github.com/jeffbass/imagezmq (accessed Sep. 08, 2021).

[87] OpenCV, “About - OpenCV.” https://opencv.org/about/ (accessed Sep. 08, 2021).

[88] Google Open Source, “MediaPipe.” https://opensource.google/projects/mediapipe (accessed Sep. 08, 2021).

[89] MATLAB, “Calling MATLAB from Python - MATLAB & Simulink.” https://www.mathworks.com/help/matlab/matlab-engine-for-python.html (accessed Sep. 08, 2021).

[90] TensorFlow, “Why TensorFlow.” https://www.tensorflow.org/?hl=en (accessed Sep. 08, 2021).

[91] F. Pedregosa, G. Varoquaux, V. Michel, and A. Gramfort, “ Scikit-learn: Machine Learning in Python.” https://www.researchgate.net/publication/51969319_Scikit-learn_Machine_Learning_in_Python (accessed Sep. 08, 2021).

[92] Tutorialspoints, “Scikit-Learn ,” 2019.

[93] DeepAI, “What is Keras?” https://deepai.org/machine-learning-glossary-and-terms/keras (accessed Sep. 08, 2021).

[94] Stringer T. Ernest, *Action Research Third Edition*. 2000.

[95] R. O’Brien, “An overview of the methodological approach of action Research,” *Univ. Toronto*, pp. 1–15, 1998.

[96] M. A. Nasrollahi, “A CLOSER LOOK AT USING STRINGER ’ S ACTION RESEARCH MODEL IN IMPROVING STUDENTS ’ RESEARCH ARTICLE A CLOSER LOOK AT USING STRINGER ’ S ACTION RESEARCH MODEL IN IMPROVING STUDENTS ’ LEARNING * Mohammad Ali Nasrollahi,” *Int. J. Curr. Res.*, no. November, 2015.

[97] C. Estay-Niculcar’s, “Investigación-Acción (action-research),” 2010. https://cestay.wordpress.com/2010/09/29/investigacion-accion-action-research-36-produccion-de-conocimiento-y-ciclicidad-conocimiento-practicas-y-ciclos/ (accessed Oct. 25, 2021).

[98] M. Borg, “Signing in the Wild dataset | IEEE DataPort,” 2019. https://ieee-dataport.org/documents/signing-wild-dataset (accessed Oct. 25, 2021).

[99] Google, “Holistic - Mediapipe,” 2020. https://google.github.io/mediapipe/solutions/holistic.html (accessed Oct. 25, 2021).

[100] Google, “Hands - mediapipe,” 2020. https://google.github.io/mediapipe/solutions/hands.html (accessed Oct. 25, 2021).