INSTITUTO TECNOLOGICO SUPERIOR DE MISANTLA

Traducción del lenguaje de señas usando visión por computadoras

TESIS

QUE PARA OBTENER EL GRADO DE:

Maestro en Sistemas Computacionales

P R E S E N T A:

Eduardo Mancilla Morales

DIRECTOR DE TESIS:

Dr. Simón Pedro Arguijo Hernández.

Misantla, Ver. Noviembre 2019

INSTITUTO TECNOLÓGICO SUPERIOR DE MISANTLA
DIVISIÓN DE ESTUDIOS PROFESIONALES
AUTORIZACIÓN DE IMPRESIÓN DE TRABAJO DE TITULACIÓN MAESTRÍA

FECHA: 19 de Noviembre de 2019.

ASUNTO: **AUTORIZACIÓN DE IMPRESIÓN DE TESIS.**

A QUIEN CORRESPONDA:

Por medio de la presente se hace constar que el (la) C:

EDUARDO MANCILLA MORALES

estudiante de la maestría en SISTEMAS COMPUTACIONALES con No. de Control 152T0736 ha cumplido satisfactoriamente con lo estipulado por el **Lineamiento de Posgrado para la obtención del grado de Maestría** mediante **Tesis.**

Por tal motivo se **Autoriza** la impresión del **Tema** titulado:

TRADUCCIÓN DEL LENGUAJE DE SEÑAS USANDO VISIÓN POR COMPUTADORA

Dándose un plazo no mayor de un mes de la expedición de la presente a la solicitud del examen para la obtención del grado de maestría.

ATENTAMENTE

Dr. Simón Pedro Arguijo Hernández
Presidente

DIVISIÓN DE ESTUDIOS PROFESIONALES

M.I.A. Roberto Ángel Meléndez Armenta
Secretario

M.S.I. Ana Lilia Sosa y Durán
Vocal

Archivo.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca recibida durante la maestría.

Agradezco a mi tutor de tesis Dr. Simón Pedro Arguijo Hernández por su apoyo y colaboración durante el desarrollo de este trabajo, por dedicar tiempo a revisar mi trabajo y resolver las dudas referentes a este.

Agradezco a los docentes miembros del jurado de corección de tesis y examen de grado, les agradezco por su tiempo y dedicación por corregir el documento final y ser miembros del jurado evaluador.

Agradezco a mis padres que me han apoyado en las metas que me he propuesto, y me han acompañado a lo largo de este logro profesional.

Resumen

Las personas con problemas auditivos y/o de habla, se pueden comunicar mediante el lenguaje de señas. Sin embargo, surgen problemas cuando se comunican con personas que desconocen el lenguaje de señas. Se propone un sistema que realiza la traducción automática de las señas estáticas del lenguaje de señas mexicano (LSM). El sistema hace uso del enfoque basado en apariencia usando una cámara. En primer lugar, la imagen se captura con la cámara, posteriormente se realiza el pre-procesamiento, escalando la imagen a un tamaño de 200×200 pixeles, reduciendo el ruido mediante el filtro bilateral, y eliminando parte del fondo mediante segmentación por color de piel basado en el espacio de color YCbCr, con el objetivo de mejorar la extracción de las características. Se realiza el análisis del desempeño de redes neuronales multicapa y SVM como clasificador para el reconocimiento de señas del LSM considerando tres conjuntos de características: Momentos de Hu, Momentos de Zernike e Histogramas de orientación del gradiente (HOG). Se creó un dataset de 21 señas del alfabeto LSM y se le aplicó un preprocesamiento a las imágenes antes de extraer el conjunto de características. Así mismo, se analizó el efecto de la reducción de dimensionalidad del conjunto de características con los algoritmos PCA y LDA. Se analizó las combinaciones de las técnicas mencionadas anteriormente mediante validación cruzada. Los resultados indican que los momentos de Zernike junto con LDA ofrece la mejor exactitud en los resultados, también ofrecen mayor velocidad de procesamiento comparadas con otras combinaciones de técnicas. La metodología utilizada en este trabajo puede utilizarse como base para el desarrollo e implementación de un sistema más completo de traducción.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Generalidades

1.1. Introducción

La comunicación oral se debe a la propagación de ondas mecánicas que se originan al momento de expulsar aire y lo cual da origen a diversos sonidos, cabe mencionar que este tipo de comunicación es una de las principales formas de interactuar entre las personas. Las personas con problemas auditivos han desarrollado su propio lenguaje de comunicación, el cual está basado en señas, este incluye movimientos de las manos, orientación, expresiones faciales, patrones en los labios y movimiento de los brazos o cuerpo, este lenguaje se conoce como lenguaje de señas y es diferente de un país a otro, así mismo el alfabeto también es diferente. En nuestro país se ha desarrollado el lenguaje de señas mexicano (LSM). Este lenguaje permite la comunicación entre personas con problemas auditivos.

Rautaray y Agrawal [1] señalan que la semántica de los gestos depende del país en donde se encuentra, el significado depende del lugar, la misma seña puede tener diferentes significados en diferentes lugares. El alfabeto mexicano es parecido al americano, pero tiene variaciones, como la incorporación de las letras ñ y ll, también cambian en la representación algunas letras. Se han tenido grandes avances en el desarrollo de sistemas de traducción automatizada de las señas a texto mediante técnicas de visión computacional. Sin embargo, dicho avance se ha dado principalmente para el lenguaje de señas americano, el japones y el de la India.

El uso de sistemas automatizados para la traducción de señas a texto requiere algoritmos avanzados de reconocimiento. Hay varias formas de realizar la traducción, mediante uso de guantes con sensores que permiten detectar los movimientos y posiciones. Otra forma de resolver el problemas, es el uso de varias cámaras como Kinect u otras cámaras 3D. También hay un enfoque que utiliza una cámara normal.

El presente trabajo utiliza el enfoque basado en apariencia con una cámara como medio de captura, la imagen capturada se procesa para realizar un reconocimiento de la seña y se convierte a texto. Cabe mencionar que la parte de reconocimiento requiere algoritmos muy complejos y con gran capacidad computacional, por lo tanto es necesario seleccionar los algoritmos que se adapten mejor al problema.

1.2. Justificación

La comunicación es fundamental para el desarrollo social del ser humano. Entre las diferentes formas de comunicación, la oral es la más común. Cuando el habla se ve impedida por algún motivo, se disminuye la capacidad de interacción social del individuo, por consecuencia su desarrollo educativo, profesional y humano quedan restringidos seriamente, lo que limita las oportunidades de inclusión.

Existen muchas personas con este problema, de acuerdo con los resultados de la Encuesta Nacional de la Dinámica Demográfica (ENADID) realizada en el año 2014, de los 119.9 millones de personas que habitan el país, 6 % (7.2 millones) tienen discapacidad, de las cuales el 33.5 % (2.4 millones) su problema es auditivo. Mencionan en [2]: "Las dificultades severas o graves para bañarse, vestirse o comer, las derivadas de problemas emocionales o mentales y para hablar o comunicarse fueron reportadas por 23.7, 19.6 y 18 % de la población con discapacidad, respectivamente.", por lo tanto de los 119.9 millones de personas que habitan el país, 6 % (7.2 millones) tienen discapacidad, de las cuales el 18 % (1.2 millones) su problema es auditivo.

El lenguaje de señas facilita la comunicación entre personas que conocen el lenguaje. Sin embargo, hay problemas al comunicarse con personas que no conoce el lenguaje de señas. Agregando que hay pocas personas que conocen ambos lenguajes: el lenguaje oral y el lenguaje de señas. Enseñar el lenguaje de señas a todas las personas requiere una gran inversión económica como de tiempo. Además de problemas de disponibilidad de las personas, por mencionar algunas limitaciones.

El problema puede solucionarse mediante la traducción automatizada de las señas a texto, mediante algoritmos y técnicas computacionales. Este problema ya se ha abordado con otros lenguajes de señas, como el americano, el japones y el hindú logrando grandes avances, pero en el lenguaje de señas mexicano aún hay pocos avances. Cabe mencionar que el lenguaje de señas no es universal, esto significa que las señas de un lenguaje a otro son diferentes. De manera computacional esto requiere diferentes configuraciones y diferentes algoritmos, debido a que el problema cambia. Por lo tanto es necesario desarrollar un sistema de traducción del lenguaje de señas mexicano(LSM).

El presente trabajo hace uso del enfoque basado en apariencia usando una cámara, por lo que la arquitectura permitirá una mayor cantidad de dispositivos, dado que no va a requerir sensores para usar el sistema, ni tampoco dispositivos como el Kinect, reduciendo los requerimientos y costos en hardware, haciéndolo ideal para los dispositivos actuales, como: laptops, smartphones, tablets y algunas minicomputadoras como la Raspberry Pi 3 con cámara.

Rautaray y Agrawal [1] mencionan que hay dos tipos de gestos con las manos: los estáticos y dinámicos. Los movimientos estáticos son definidos como orientación y posición de la mano, durante algún tiempo sin mover la mano. Si la mano varía durante ese tiempo, es llamado gesto dinámico. El reconocimiento de gestos dinámicos requiere reconocer secuencias de imágenes en el tiempo, debe detectarse que las secuencias de imágenes tengan un patrón de moviento, esto necesita conocimiento de las secuencias anteriores, también requiere de algoritmos que permitan este tipo de reconcimiento, además de un mayor tiempo de diseño, programación e investigación. Debido a esto, se ha limitado el presente trabajo al reconocimiento de señas estáticas, aplicado al reconocimiento del alfabeto del lenguaje de señas

mexicano. Sin embargo, las técnicas empleadas y desarrolladas en este trabajo pueden aplicarse en el diseño de un sistema más completo. De acuerdo a lo mencionado, en este trabajo se desarrolla e implementa un sistema que permita la traducción de las letras estáticas del alfabeto del lenguaje de señas mexicano (LSM).

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un sistema de tiempo real para el reconocimiento del alfabeto del lenguaje de señas mexicano utilizando algoritmos y técnicas de visión artificial y reconocimiento de patrones.

1.3.2. Objetivos Específicos

- Elaborar un dataset de imágenes del alfabeto estático del lenguaje de señas mexicano.
- Implementar técnicas de pre-procesamiento a las imágenes del dataset.
- Implementar un algoritmo de segmentación de piel.
- Realizar la umbralización de la imagen para obtener la forma de la seña estática.
- Implementar técnicas de extracción de características de forma.
- Crear un dataset de características numéricas del alfabeto estático del lenguaje de señas mexicano.
- Implementar clasificadores para analizar su desempeño con el dataset generado.

1.4. Metodología

Los pasos a seguir son los siguientes:

- Analizar, seleccionar e implementar algoritmos de pre-procesamiento de imágenes.
- Establecer los métodos de segmentación tanto de fondo como de piel para eliminar aquellas zonas que puedan afectar el desempeño de los clasificadores.
- Extraer las características de las señas que realiza la mano considerando que para el buen desempeño del clasificador la seña debe ser invariante a traslación, rotación y escalamiento. Se analizan e implementan los momentos de Hu, momentos de Zernike y los Histogramas de Gradientes Orientados (HOG, por sus siglas en inglés). Esto da paso a la creación del dataset.
- Reconocer la imagen ingresando las características extraídas en el paso anterior y entrenando un clasificador multiclase. Los algoritmos a analizar son Máquina de Soporte Vectorial y Perceptron multicapa con backpropagation.

- Aplicar técnicas de reducción de dimensionalidad a las características extraídas, entrenar el clasificador y comparar en incremento en la exactitud y velocidad. Las técnicas analizadas son PCA y LDA.
- Analizar los resultados para determinar las mejores combinaciones de los algoritmos, comparando en velocidad de procesamiento y exactitud.

Capítulo 2

Marco teórico

2.1. Lenguaje de señas mexicano

La Lengua de Señas Mexicana (LSM), es la lengua que utilizan las personas sordas en México. Como toda lengua, posee su propia sintaxis, gramática y léxico. Se compone de signos visuales con estructura lingüística propia. Para la gran mayoría de quienes han nacido sordos o han quedado sordos desde la infancia o la juventud, esta es la lengua en que articulan sus pensamientos y sus emociones, la que les permite satisfacer sus necesidades de comunicación así como desarrollar sus capacidades cognitivas al máximo mientras interactúan con el mundo que les rodea.

En la figura 2.1, se muestra el alfabeto del lenguaje de señas mexicano [3], el cual consta de 27 señas, de las cuales 21 son estáticas y las restantes dinámicas. Las señas dinámicas se realizan con el movimiento marcado con color rojo. Cabe mencionar que este trabajo está enfocado a las señas estáticas.

2.2. Definición de imagen

Una imagen se puede representar como una cuadricula rectangular de pixeles, ver Fig. 2.2. Cada posición en la imagen se localiza utilizando valores enteros positivos en un sistema de coordenadas cartesianas. En cada posición de la imagen un pixel representa el brillo y el color de un punto.

Además de la resolución espacial, o sea el ancho y alto de la imagen en pixeles, las imágenes también se describen utilizando bits de profundidad. Los bits de profundidad se refieren al número total de bits que describen el color o la intensidad de cada pixel. El número de bits utilizados para almacenar la información de color indica cuántos diferentes valores pueden almacenarse. Por ejemplo, una imagen con un byte de profundidad permite almacenar imágenes de intensidad también llamadas imágenes de niveles de gris. Así mismo, una imagen con tres bytes de profundidad almacena imágenes a color empleando un byte para cada uno de los colores primarios: Rojo, Verde y Azul.

Figura 2.1 Imagen del alfabeto LSM.

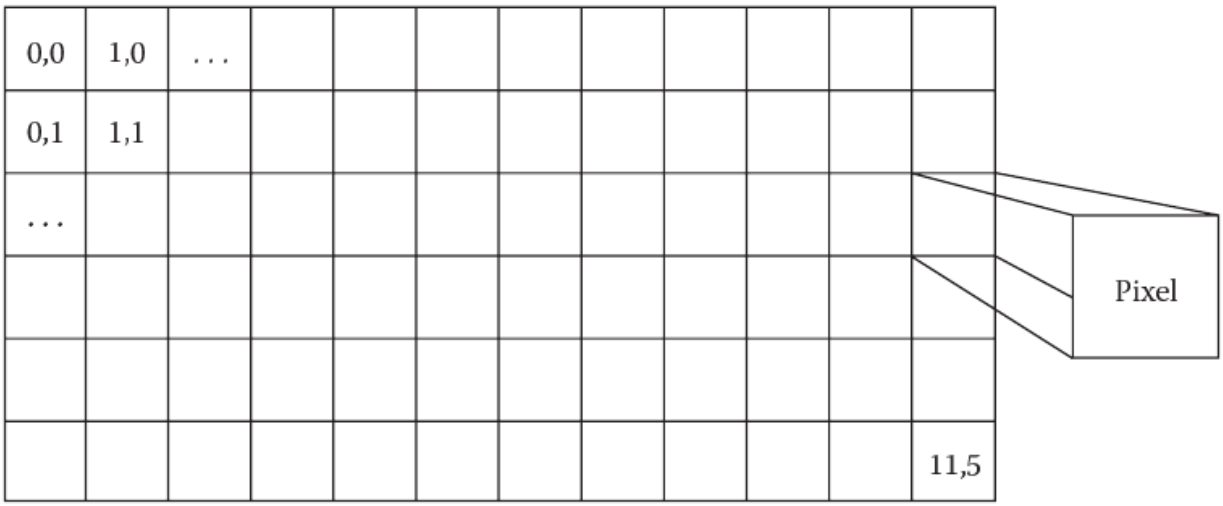


Figura 2.2 Imagen de 12 pixeles de ancho y 6 pixeles de alto.

2.3. Análisis de imágenes

El análisis de imágenes es el proceso de trabajar directamente con los píxeles de la imagen para obtener cierta información útil de ésta, una medición de los objetos que se encuentran representados. Si bien algunos algoritmos de análisis de imágenes pueden ser bastante complejos, hay una variedad de técnicas simples que se utilizan ampliamente debido a su variada aplicabilidad. Por ejemplo, el filtrado de imágenes se utiliza a menudo para reducir o eliminar el ruido de la imagen antes de procesarla. La segmentación de imágenes se puede utilizar para localizar regiones de interés, separar el objeto de interés del fondo, y varía en complejidad desde técnicas muy sencillas hasta muy complejas.

2.4. Reconocimiento de imágenes

El reconocimiento de imágenes se utiliza para realizar un gran número de tareas visuales, como etiquetar el contenido de las imágenes con meta etiquetas, realizar búsquedas de contenido de imágenes y guiar robots autónomos para evitar accidentes, por ejemplo. Un algoritmo de reconocimiento de imágenes toma una imagen como entrada e indica a la salida los objetos que contiene la imagen. En otras palabras, la salida es una etiqueta de la imagen. La pregunta obligada es, ¿cómo el algoritmo identifica el contenido de la imagen? El algoritmo debe entrenarse para aprender las diferencias entre las diferentes clases. Dicho entrenamiento debe realizarse con un gran número de imágenes que contengan los objetos a identificar. Cabe mencionar que el algoritmo solamente reconocerá los objetos que ha aprendido. La identificación de objetos cae ente dos tipos: binario (solamente dos clases) o multiclase (más de dos clases).

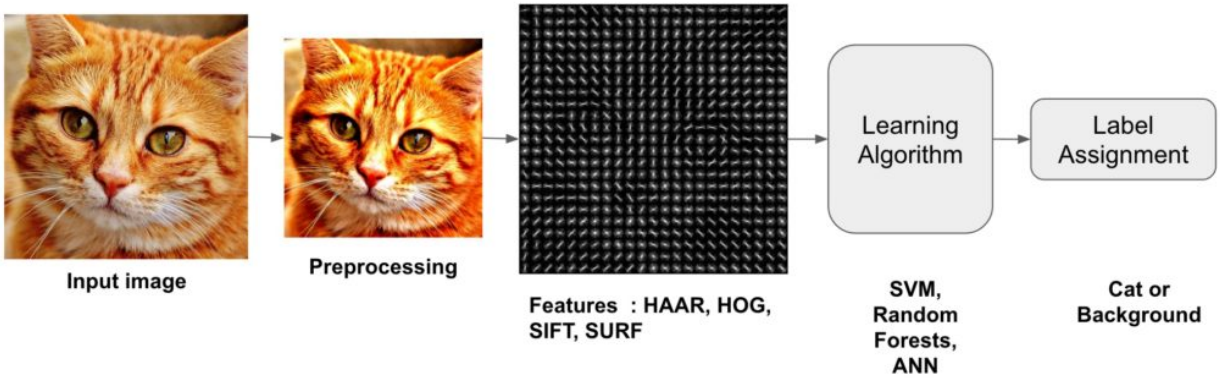


Figura 2.3 Proceso de reconocimiento de una imagen.

En la Figura 2.3, se muestra el proceso de reconocimiento de una imagen [4]. El proceso consiste en tomar una imagen como entrada, preprocesar la imagen, extraer las características que permitan diferenciar el objeto de los demás eliminando información innecesaria, y clasificar que imagen es mediante un clasificador que necesita entrenarse, una vez entrenado será capaz de clasificar nuevas imágenes.

2.5. Preprocesamiento

La creciente necesidad de desarrollar sistemas automatizados para la interpretación de imágenes requiere que éstas estén libres de ruido y otras aberraciones. El pre-procesamiento consiste en mejorar; para el observador, la visualización de la imagen, en términos de un mejor contraste y visibilidad de características de interés, o bien deshacer los efectos de degradación que pudieron haber sido causados por el sistema de adquisición. Es por demás importante realizar operaciones de pre-procesamiento en la imagen para que la imagen procesada sea más adecuada para su interpretación automática y así mejorar la tasa de precisión y su consecuente interpretación. Debido a que el objetivo del mejoramiento depende del contexto de la aplicación y frecuentemente está pobremente definido, y el criterio frecuentemente subjetivo, las técnicas de mejoramiento tienden a ser ad hoc. Las técnicas de mejoramiento incluyen operaciones puntuales, donde el valor el pixel de salida depende únicamente de su correspondiente valor de entrada, y operaciones locales o de vecindad, donde la salida eventual del pixel depende del valor de sus pixeles vecinos. Esta última operación incluye la convolución, la cual utiliza apropiadas máscaras para realizar un suavizado o detección de bordes de una imagen.

2.6. Ruido en imágenes digitales

El ruido ocurre en todas las imágenes capturadas, independientemente de la calidad del sensor. El ruido se debe a una serie de factores, pero principalmente por el ruido electrónico. Las diferentes causas del ruido producen variaciones no deseadas en el color o la intensidad de los píxeles, alejados del color verdadero del objeto que se está visualizando. Estas variaciones se obtienen de diferentes distribuciones de probabilidad, dependiendo de la naturaleza del ruido. Por ejemplo, la mayoría del ruido generado por el sensor de una cámara seguirá una distribución gaussiana.

Figura 2.4 Izquierda: Imagen con ruido. Derecha: Imagen sin ruido.

El filtrado de imágenes es una forma eficaz de reducir el ruido en una imagen preservando aspectos importantes de ésta. La naturaleza del filtro utilizado debe depender de la naturaleza del ruido, y se pueden utilizar varios filtros donde hay múltiples fuentes de ruido. En la Figura 2.4, se muestra un ejemplo de ruido en una imagen y tal como se puede apreciar éste afecta de manera significativa la información incluida haciendo más difícil reconocer los objetos presentes.

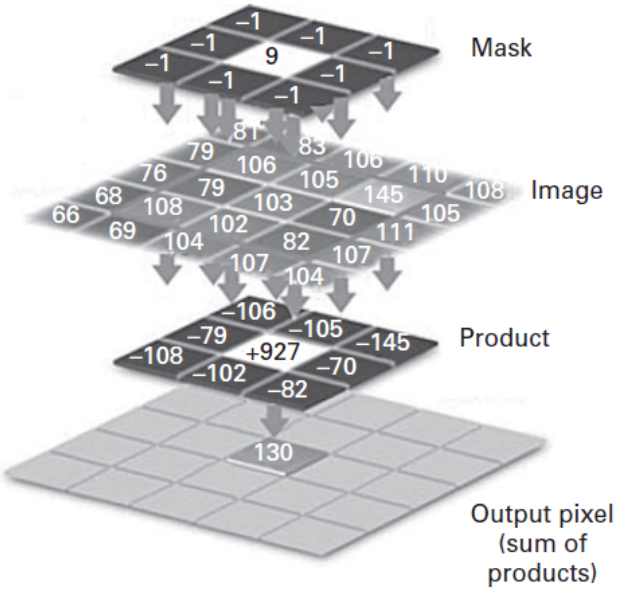


Figura 2.5 Proceso de convolución

2.7. Convolución

La convolución discreta en el dominio espacial es un proceso fundamental en procesamiento de imágenes, y se utiliza tanto para suavizar o resaltar una imagen. Consiste en una suma de productos y una operación de desplazamiento. El concepto básico es que una máscara o kernel, esencialmente una pequeña imagen o matriz de $k \times k$ elementos, se rota 180° y se desplaza a través de la imagen de $M \times M$ pixeles; k es usualmente un número impar, mucho más pequeño que el tamaño de la imagen. Cada pixel de la imagen de salida es una suma ponderada de los pixeles de entrada dentro la región definida por la máscara, con los elementos de la mascara definiendo los pesos.

Si la imagen de entrada es F (de tamaño $M \times M$) y la máscara de convolución es H (de tamaño $k \times k$), la imagen de salida G está dada como:

$$G = F * H \tag{2.1}$$

El valor de cada pixel de la imagen de salida se calcula:

$$g(i,j) = \sum_{m=-a}^{a} \sum_{n=-a}^{a} f(m,n)h(i-m, j-n) \tag{2.2}$$

donde $a = (k-1)/2$.

El proceso se ilusta en la Figura 2.5. La máscara se gira 180° y se coloca en la parte superior de la imagen de entrada, comenzando en la posición superior izquierda. Los elementos de la máscara se multiplican por los valores de píxel correspondientes en la imagen, y los productos se suman y normalizan para formar una respuesta ponderada que es el valor del píxel de salida en la posición correspondiente al centro de la máscara. En la mayoría de

las máscaras k es impar, de modo que el centro de la máscara sea fácilmente evidente. A continuación, la máscara se mueve una posición hacia la derecha, la suma de productos se recalcula y normaliza para dar el siguiente valor de píxel en la imagen de salida. Este proceso se repite al desplazarse la máscara a través de la imagen de entrada.

2.7.1. Filtro bilateral

El filtro bilateral es no lineal a diferencia del filtro gaussiano, éste toma en consideración la variación de intensidades para preservar los bordes. Se basa en la idea de que 2 pixeles que se encuentran cerca el uno del otro, no sólo si ocupan localizaciones espaciales cercanas, sino que también tienen alguna similaridad en el rango fotométrico (intensidad del color, distancia de profundidad, etc.). De acuerdo a lo mencionado por Paris y colaboradores [5] este filtro usa la convolución gaussiana. La acción de la convolución gaussiana es independiente del contenido de la imagen. La influencia que tiene un pixel en otro pixel depende sólo de la distancia en la imagen y no del valor actual de la imagen. El filtro está definido de la siguiente manera:

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G\sigma_s(\|p - q\|) G\sigma_r(I_p - I_q) I_q \tag{2.3}$$

Donde W_p es el factor de normalización que se asegura que los pesos de los pixeles dan una suma de 1.0:

$$W_p = \frac{1}{W_p} \sum_{q \in S} G\sigma_s(\|p - q\|) G\sigma_r(I_p - I_q) \tag{2.4}$$

Donde:

- $G_\sigma(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x^2+y^2}{2\sigma^2})$ es el kernel gausiano 2D.
- σ es el diámetro de cada pixel vecino que se utiliza durante el filtrado.
- I es una imagen en escala de grises.
- I_p es el pixel de la imagen en la posicion p.
- S es el conjunto de todas las posibles localizaciones de la imagen.
- p es el indice del pixel de la imagen.
- $\|p - q\|$ es la distancia euclideana entre las posiciones de los pixeles p y q.
- σ_r es el núcleo espacial para suavizar las diferencias de coordenadas. Esta función puede ser una función gaussiana.
- σ_s es el núcleo gama para suavizar las diferencias en las intensidades. Esta función puede ser una función gaussiana.

- $G\sigma_s$ es una gausiana espacial ponderada que disminuye la influencia de los pixeles distantes.
- $G\sigma_r$ es un rango gausiano que disminuye la influencia de los pixeles q cuando sus valores de intensidad difieren de I_p

Es muy efectivo para remover ruido, preservando los bordes. Pero sus operaciones son más lentas comparadas con otros filtros. El filtro bilateral es controlado por dos parámetros σ_s y σ_r:

- Cuando el rango σ_r incrementa, el filtro bilateral llega a ser más cercano al difuminado gausiano, porque el rango gaussiano es plano.
- Incrementando el parámetro espacial σ_s suaviza características más grandes.

2.8. Segmentación de la imagen

La segmentación es el proceso de dividir los píxeles de una imagen en grupos, donde cada grupo tiene una distinción significativa de los demás. De manera general la segmentación divide los píxeles en dos clases, donde una clase representa las áreas de interés en la imagen y la otra representa el fondo. La segmentación en dos grupos de píxeles se logra usando el umbral, aunque también se puede lograr la segmentación agrupado píxeles en función de su intensidad o color, la cual se realiza en el espacio de color adecuado a la aplicación.

2.9. Espacios de color

Un espacio de color, también conocido como modelo de color (o sistema de color), es un modelo matemático que describe el rango de colores como tuplas de números, típicamente como 3 o 4 valores o componentes de color. Por ejemplo, las imágenes de color típicas, en particular las generadas por un sistema de imagen digital, se representan como rojo, verde, azul y normalmente se denominan imágenes RGB; o sea, están definidas en el espacio de color RGB. Básicamente, el espacio de color es una elaboración del sistema de coordenadas y subespacio. Cada espacio de color tiene un sistema de coordenadas de color específico y cada punto en el espacio de color representa sólo un color específico. Hay una variedad de espacios de color, y cada uno de ellos puede ser útil para aplicaciones específicas. A continuación se mencionan algunos espacios de color.

2.9.1. RGB

El espacio de color RGB divide cada píxel en tres colores, los cuales representan los tres componentes primarios del color, rojo, verde y azul. RGB se puede visualizar como un cubo tridimensional (ver Fig. 2.6), donde cada eje representa uno de los canales de color. El color negro se encuentra en el origen, con las coordenadas (0, 0, 0). En la esquina opuesta se encuentre el blanco, con valores (255, 255, 255) para una imagen de 8 bits. Los píxeles en escala de grises se encuentran a lo largo de la línea entre las esquinas en blanco y negro,

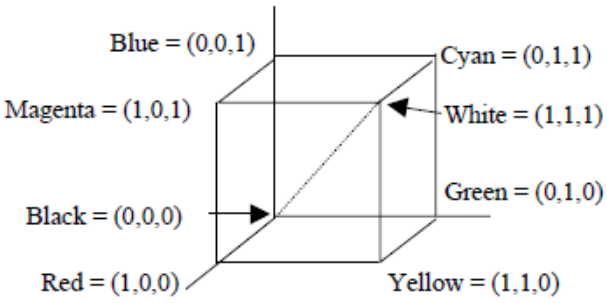


Figura 2.6 Espacio de color RGB.

donde R, G y B tienen el mismo valor. El formato RGB es popular porque coincide con la estructura de los píxeles en monitores y otras pantallas. Sin embargo, la principal desventaja notable de RGB es que combina el color y el brillo en el mismo espacio. Existen conversiones entre RGB y todos los demás espacios de color; por lo tanto, el componente de color requerido se puede separar del componente de brillo convirtiéndolo en un espacio de color que hace esa distinción.

2.9.2. HSV

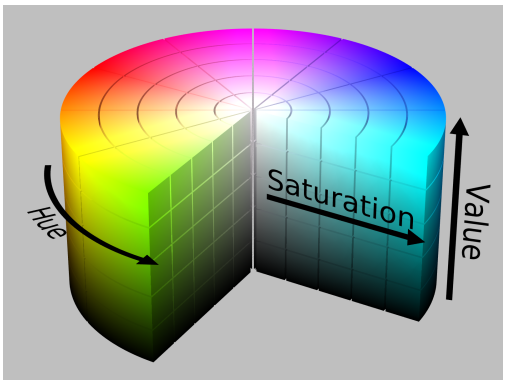


Figura 2.7 Espacio de color HSV.

El espacio de color HSV representa los componentes de tono, saturación y valor de un píxel. Este espacio de color se ve más fácilmente como un cilindro, como se muestra en la Figura 2.7, con el tono como la posición alrededor del borde, la saturación es la distancia desde el centro al borde y el valor es la posición de arriba hacia abajo.

El tono; también denominado matiz, representa el color del píxel, y usualmente su valor va de 0° a 360°. A 0° el color es rojo; al rotar sobre la rueda del tono, los colores pasarán a través del azul y el verde, y finalmente volverá al rojo. La saturación representa la intensidad del color, desde oscuro hasta un nivel de gris. Cuanto más cerca del centro de la sección transversal del cilindro se encuentre el valor de la saturación, menos se expresará el valor de matiz. Finalmente, la posición hacia arriba y hacia abajo del cilindro representa el valor o el brillo. Hacia abajo habrá píxeles más oscuros, con píxeles más claros arriba. Cualquier

posición en el cilindro de HSV puede coincidir con un píxel en RGB, y la conversión entre los dos espacios de color es simple. Debido a que el HSV separa el color (H + S) de la intensidad (V), el espacio de color HSV separa los componentes de color de un píxel del componente de escala de grises de manera que RGB no lo hace. Por lo tanto, HSV es más estable durante el cambio de condiciones de iluminación, que puede ocurrir al analizar imágenes con el tiempo.

2.9.3. HSL

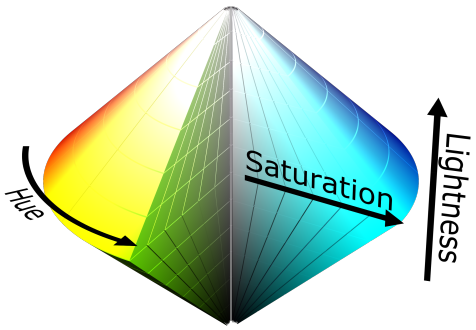


Figura 2.8 Espacio de color HSL.

Al igual que en HSV, el espacio de color HSL (H, tono o matiz; S, saturación; L, luminosidad) convierte RGB en distintos componentes de color y brillo. Sin embargo, hay pequeñas diferencias, como se puede ver en la Figura 2.8. Mientras que el valor de saturación aún influye en si un color es vibrante u opaco, el valor de luminosidad asigna la negrura y la blancura de un color. Para los propósitos del análisis de la imagen, HSV y HSL son similares. Sin embargo, a menudo se piensa que HSL es el espacio más intuitivo, ya que un alto valor de luminosidad producirá un píxel blanco, en lugar de un píxel cuya blancura depende de la variable de saturación adicional.

2.9.4. YCbCr

YCbCr (a veces denominado YUV) existe para reducir la redundancia inherente a las señales enviadas mediante RGB. El componente Y, escalado entre 0 y 1, representa la luminancia de un píxel. Los componentes de color Cb y Cr representan los valores de crominancia, o sea, la diferencia azul y la diferencia roja. Cualquier color RGB se puede encontrar en el eje Cb y Cr, con la luminancia Y especificando el tono de ese color. En la Figura 2.9, se muestra un diagrama del espacio de color; sin embargo, a menudo es más fácil de visualizar como sólo CbCr utilizando valores Y constantes, como en la figura. Si bien la cantidad de información por píxel en YCbCr no es mayor que en RGB, al separar la información de luminancia y color, al igual que con HSV, se pueden aplicar diferentes algoritmos de compresión al color o a la intensidad de una imagen. El formato de archivo JPEG utiliza el espacio de color YCbCr para comprimir la información de color. Los observadores humanos no pueden resolver la información de color con tanta precisión como la intensidad de escala de grises, por lo que la codificación de video para TV también se procesa como YCbCr.

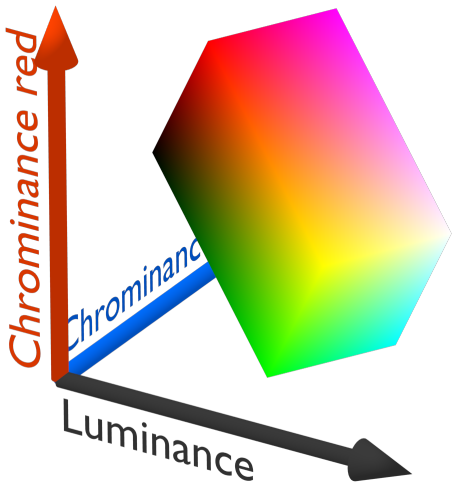


Figura 2.9 Espacio de color YCbCr.

Si bien la compresión de la información de color no es ideal para las imágenes destinadas al análisis de imágenes, se debe tener en cuenta que muchos algoritmos, como la segmentación y la reconstrucción estéreo, pueden funcionar bien en las imágenes en escala de grises. Otros algoritmos, como los filtros de bordes, exclusivamente operan en imágenes en escala de grises. Por lo tanto, es más importante preservar la información de intensidad que la información de color cuando se capturan imágenes.

2.9.5. Espacio de color para detección de la piel

El color juega un importante rol en la detección, seguimiento y reconocimiento de objetos. Algunos espacios de color se han propuesto para la detección de la piel humana, entre los más utilizados están YCbCr y HSI. Aunque, cabe mencionar que se acepta, no hay un único espacio de color conveniente para la detección de la piel en todas la imágenes a color. Este trabajo utiliza YCbCr para la detección del color de la piel. Las siguientes expresiones se utilizan para realizar la conversión de RGB a YCbCr:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}169 & -0{,}331 & 0{,}500 \\ 0{,}500 & -0{,}419 & 0{,}081 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Las personas tienen diferente color de piel en apariencia, numerosos estudios han demostrado que la principal diferencia esta en la intensidad, más que en el color. Comparado con RGB, el espacio de color YCbCr es lumina-dependiente, así que es uno de los más populares espacios de color para detección de la piel. Se menciona que la agrupación del color de la piel, es más compacto en YCbCr que en otro espacio de color. Para realizar la segmentación del color se aplica el algoritmo de detección de piel usando un valor umbral para cada componente.

En este trabajo se utiliza el método de detección de piel basado en el espacio de color YCbCr, el cual se basa en un valor umbral de cada componente del espacio de color [6]. Los

valores del los umbrales considerados son:

$$150 < Cr < 200 \tag{2.5}$$

$$100 < Cb < 150 \tag{2.6}$$

Se realiza un filtrado por toda la imagen en el espacio de color YCbCr, tomando el valor del pixel y comparando su valor con los valores de los umbrales. Si el pixel considerado está dentro del rango, se considera como un pixel con el color de la piel y se le asigna un 1 en una matriz binaria del mismo tamaño, de lo contrario se considera como no piel y el matriz se le asigna un 0.

El resultado es una imagen binaria donde todos los píxeles están marcados como piel o no piel. Algún ruido puede tomar lugar en ambas áreas de piel y no piel.

2.10. Extracción de características

La extracción de las características es un paso importante en la construcción de cualquier sistema de clasificación de patrones y tiene como objetivo la extracción de la información relevante que caracteriza a cada clase. En este proceso, las características relevantes se extraen de los objetos para formar vectores de características. Estos vectores se utilizan por los clasificadores para la unidad de entrada con una unidad de salida objetivo. El objetivo principal de la extracción de características consiste en extraer un conjunto de características relevantes, las cuales maximicen la tasa de reconocimiento con la menor cantidad de elementos y generar un conjunto de características similares, denominado patrón, para diversas observaciones de un mismo objeto. Es decir, se obtiene la información más relevante de los datos originales.

Para el caso de imágenes digitales, en ocasiones, se vuelve imprescindible utilizar técnicas de pre-procesamiento con la finalidad de mejorar la imagen. Inmediatamente después se aplican técnicas apropiadas de extracción de características de forma, de textura o la indicada que dependen del problema a resolver. Cabe mencionar que tanto en reconocimiento de patrones y en procesamiento de imágenes, la extracción de características es una forma especial de reducir la dimensionalidad.

La tarea principal del reconocimiento de patrones consiste en tomar y asignar los vectores de características correctamente como una de las posibles clases de salida. Este proceso se puede dividir en dos etapas generales: Selección de características y Clasificación. La selección de características en fundamental en todo el proceso puesto que el clasificador no será capaz de reconocer características mal seleccionadas. Los criterios para elegir características dadas por [7] son:

“Las características deben contener la información requerida para distinguir entre clases, ser insensibles a la variabilidad irrelevante a la entrada, y también debe ser un número limitados, para permitir un cálculo eficiente de las funciones discriminantes y limitar la cantidad de datos de entrenamiento requeridos”.

2.10.1. Momentos de Hu

La forma es una propiedad fundamental de un objeto. Hay dos tipos de descriptores de forma: descriptores de forma basados en contorno y descriptores de forma basados en región. Los momentos regulares invariantes son uno de los descriptores de formas basados en contornos más populares y más utilizados derivado por Hu. Hu utilizó momentos regulares para desarrollar un conjunto de funciones no lineales llamadas momentos invariantes que son invariables a la traslación, el escalamiento y la rotación los cuales aplicó a un problema de reconocimiento de caracteres. Utilizó momentos centrales y la técnica de normalización masiva para obtener la invariabilidad de la traslación y el escalamiento. También propuso dos métodos diferentes para obtener invariancia a la rotación. El primer método se basa en combinaciones de momentos regulares utilizando invariantes algebraicos, que él llamó como invariantes del momento absoluto. El otro método se derivó utilizando el método del eje principal.

Si $f(x, y)$ es una imagen digital, el orden del momento $p + q$ se define como:

$$\mu_{pq} = \sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q f(x, y) \quad (p, q) = 0, 1, 2, ... \tag{2.7}$$

En donde:

$$\overline{x} = \frac{m_{10}}{\overline{m_{00}}} \quad \overline{y} = \frac{m_{01}}{\overline{m_{00}}} \tag{2.8}$$

El $(p + q)$ orden normalizado de momentos centrales esta definido como:

$$n_{pq} = \frac{\mu_{pq}}{\mu_{00}^r} \quad r = \frac{p+q}{2} \quad p + q = 2, 3, 4 \tag{2.9}$$

constituidos por la combinación lineal del segundo y tercer orden de los momentos centrales, las siguientes son las expresiones especificas de los 7 momentos invariantes:

$$\phi(1) = (\mu_{20} + \mu_{02})$$

$$\phi(2) = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

$$\phi(3) = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$$

$$\phi(4) = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

$$\begin{aligned}\phi(5) = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] \\ +(3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]\end{aligned}$$

$$\begin{aligned}\phi(6) = (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \\ +4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})\end{aligned}$$

$$\phi(7) = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] \\ +(\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]$$

Los momentos de Hu tienen el siguiente significado fisico: los momentos de bajo orden describen de manera general las características de la imagen, el momento de orden cero refleja el área objetivo, el primer momento refleja el centro de masa del objetivo, el momento de orden 2 refleja la longitud principal, eje auxiliar y el ángulo de orientación del eje principal. Los momentos más grandes describen los detalles de la imagen, por ejempo la distorsión.

Los momentos son mediciones estadísticas que permanecen constantes después de transformaciones como traslación, rotación y escala. Los momentos de Hu se extraen de los pixeles de una imagen binaria.

Este algoritmo es invariante a traslación, escala y rotación, esto ofrece una gran ventaja, porque aunque una imagen se encuentre girada, tendrá valores similares, haciendo el proceso de reconocimiento más exacto.

2.10.2. Momentos de Zernike

Según Khotanzad y Hong [8], Zernike introdujo un conjunto de polinomios complejos los cuales forman un conjunto ortogonal completo sobre el interior de un circulo unitario, $x^2 + y^2 = 1$. El conjunto de esos polinomios son denotados como $\{V_{nm}(x, y)\}$. La forma de esos polinomios es:

$$V_{nm}(x, y) = V_{nm}(p, \theta) = R_{nm}(p) \exp(jm\theta) \tag{2.10}$$

Donde:

- n: entero positivo o cero.
- m: entero positivo y negativo sujeto a las siguientes restricciones $n - |m|$ es par, $|m| \leq n$.
- p: longitud del vector desde el origen a (x, y) pixel.
- θ: angulo entre el vector p y el eje x en sentido contrario a las agujas del reloj.
- $R_{nm}(p)$: polinomio radial definido como:

$$R_{nm}(p) = \sum_{s=0}^{n-|m|/2} (-1)^s . \frac{(n-s)!}{s!(\frac{n+|m|}{2})!(\frac{n-|m|}{2})!} p^{n-2s} \tag{2.11}$$

Se debe señalar que al ser momentos ortogonales $R_{n,-m}(p) = R_{nm}(p)$. Esos polinomios son ortogonales y satisfacen

$$\iint_{x^2+y^2 \leq 1} [V_{nm}(x, y)]^* \quad V_{pq}(x, y)\, dx\, dy = \frac{\pi}{n+1} \delta_{np} \delta_{mq} \tag{2.12}$$

con

$$\delta ab = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

Los momentos de Zernike son la proyección de la función de la imagen en esas funciones de base ortogonal. Los momentos de Zernike de orden n con repetición m para una función de imagen continua $f(x, y)$ que desaparece fuera del circulo unitario es

$$A_{nm} = \frac{n+1}{\pi} \iint_{x^2+y^2 \leq 1} f(x, y)\, V^*_{nm}(p, \theta)\, dx\, dy \tag{2.14}$$

Para una imagen digital, las integrales son reemplazadas por una sumatoria para obtener:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x, y)\, V^*_{nm}(p, \theta) \qquad x^2 + y^2 \leq 1 \tag{2.15}$$

Para calcular los momentos de Zernike de una imagen, el centro de la imagen es tomado como el origen y las coordenadas del pixel son mapeadas al rango del circulo unitario:

$$x^2 + y^2 \leq 1 \tag{2.16}$$

Aquellos pixeles que caen fuera del circulo unitario no son utilizados en el cálculo. También nota que $A^*_{nm} = A_{n,-m}$

2.10.3. Histograma de Gradientes Orientados (HOG)

El objetivo de este método es describir una imagen mediante un conjunto de histogramas locales. Esos histogramas cuentan las ocurrencias de la orientación del gradiente en una parte local de la imagen. Los pasos propuestos por Suard y colaboradores [9] son los siguientes:

1. Calcular los gradientes de la imagen.
2. Construir histogramas de orientación para cada celda.
3. Normalizar los histogramas dentro de cada bloque de celdas.

De manera mas detalla son:

1. Cálculo del gradiente: El gradiente de una imagen se obtiene por medio de fitrado con dos filtros de una dimensión:
   - $horizontal : (-101)$
   - $vertical : (-101)$

   El gradiente puede tener o no tener signo. Este ultimo caso es justificado por el hecho de que la direccion de el contraste no tenga importancia. En otras palabras, se podría tener el mismo resultado con un objeto blanco colocado en un fondo negro, comparado con un objeto negro colocado en un fondo blanco.
2. Celda y descriptores de bloque: la particularidad de este método es dividir la imagen en diferentes celdas. Una celdas se puede definir como una región espacial, como un cuadrado con un tamaño predefinido en píxeles. Para cada celda, luego se calcula el histograma de gradientes, acumulando votos en contenedores para cada orientación.

Los votos se pueden ponderar por la magnitud de un gradiente, de modo que el histograma tome en cuenta la importancia del gradiente en un punto dado. Esto se puede justificar por el hecho de que una orientación de gradiente alrededor de un borde debe ser más significativa que la de un punto en una región casi uniforme. Cuanto mayor sea el número de intervalos, más detallado será el histograma. Cuando todos los histogramas se han calculado para cada celda, se construye el vector descriptor de una imagen que concatene todos los histogramas en un sólo vector. Sin embargo, debido a la variabilidad en las imágenes, es necesario normalizar los histogramas de las celdas. Los histogramas de celdas se normalizan localmente, de acuerdo con los valores de los histogramas de celdas vecinas. La normalización se realiza entre un grupo de celdas, lo que se denomina bloque. Luego se celdas un factor de normalización sobre el bloque y todos los histogramas dentro de este bloque se normalizan de acuerdo con este factor de normalización. Una vez que se ha realizado este paso de normalización, todos los histogramas se pueden concatenar en un único vector de características.

Son posibles diferentes esquemas de normalización para un vector V que contiene todos los histogramas de un bloque dado. El factor de normalización nf podría obtenerse a lo largo de estos esquemas:

- ninguno: ninguna normalización es aplicada a la celda. $nf = 1$.
- $L1 - norm : nf = \frac{v}{(\|v\|_1 + e)}$
- $L2 - norm : nf = \frac{v}{(\|v\|_2^2 + e^2)}$

donde

- v: es el vector no normalizado que contiene todos los histogramas en un bloque dado
- e: es una pequeña constante de regularización. Es necesario ya que en algún momento evaluamos gradientes vacíos. El valor de e no tiene influencia en los resultados. Según la forma en que se haya construido cada bloque, un histograma de una celda determinada puede participar en la normalización de varios bloques. En este caso, el vector de características final contiene algunas informaciones redundantes que se han normalizado de una manera diferente. Este es especialmente el caso si los bloques de celdas se superponen.
- $\|v\|_k$: es k-normal para $k = 1, 2$.

2.11. Aprendizaje no supervisado y supervisado.

Hay dos enfoques principales de reconocimiento de patrones: sin supervisión y supervisados. En la categoría no supervisada (también denominada aprendizaje no supervisado), el problema es descubrir la estructura del conjunto de datos, si es que existe alguna. Esto significa que se quiere saber si hay grupos en los datos, y qué características hacen que los objetos sean similares dentro del grupo y diferentes entre los grupos. Se han desarrollado y

se están desarrollando muchos algoritmos de agrupamiento para el aprendizaje no supervisado. La elección de un algoritmo es una cuestión de preferencia del diseñador. Diferentes algoritmos pueden crear diferentes estructuras para el mismo conjunto de datos. La maldición y la bendición de esta rama de reconocimiento de patrones es que no hay una verdad fundamental contra la cual comparar los resultados. La única indicación de qué tan bueno es el resultado es, probablemente, la estimación subjetiva del usuario.

En la categoría supervisada (también llamada aprendizaje supervisado), cada objeto del conjunto de datos viene con una etiqueta de clase preasignada. La tarea del diseñador consiste en entrenar a un clasificador para que haga el etiquetado "sensatamente". En la mayoría de los casos, el proceso de etiquetado no se puede describir de forma algorítmica. Por lo tanto, se le proporciona a la máquina habilidades de aprendizaje y se le presentan los datos etiquetados. El conocimiento de la clasificación aprendido por la máquina en este proceso puede ser oscuro, pero la precisión de reconocimiento del clasificador será el juez de su idoneidad.

La selección, entrenamiento y prueba de un modelo de clasificador forman el núcleo del reconocimiento de patrones supervisado. Se puede decidir utilizar un mismo modelo clasificador y repetir el entrenamiento con diferentes parámetros, o también cambiar el modelo del clasificador.

2.12. Algoritmos de clasificación

2.12.1. Máquina de soporte vectorial

Las máquinas de soporte vectorial (SVM por su siglas en inglés) pertenece al tipo de aprendizaje supervisado y se utiliza tanto para problemas de clasificación como de regresión. Es un sistema de aprendizaje que separa un conjunto de vectores de entrada (patrones) en dos clases con un hiperplano de separación óptimo. Se dice que el conjunto de vectores está separado de manera óptima por el hiperplano si está separado sin error y la distancia entre los vectores más cercanos al hiperplano es máxima. El clasificador SVM se obtiene aplicando una variedad de funciones de núcleo, también conocido como kernel; lineal, polinomial, funciones de base radial, como posibles conjuntos de funciones de optimización. Su base se encuentra en la solución de un problema de programación cuadrática dual y utilizando la minimización del riesgo estructural como principio inductivo, a diferencia de los algoritmos estadísticos clásicos que maximizan el valor absoluto de un error o de un error al cuadrado. Las principales ventajas de la máquina de soporte vectorial son:

- Efectivo en espacios de alta dimensionalidad.
- Efectivo en los casos donde el número de dimensiones es mayor al número de ejemplos.
- Utiliza un subconjunto de puntos de entrenamiento (vectores de soporte) en la función de decisión.
- Diferentes funciones del kernel pueden especificarse para la función de decisión. Kernels comunes son incluidos, pero es posible especificar kernels personalizados.

Figura 2.10 Dos diferentes parámetros de regularización. Izq: Menor valor de regularización. Der: Mayor valor de regularización .

Las desventajas de este clasificador son:

- Si el número de características es más grande que el número de ejemplos, para evitar el sobre entrenamiento es crucial la selección del kernel y el valor de regularización.
- No provee estimación de probabilidades directamente, son calculadas usando una costosa validación cruzada de 5 partes.

En una clasificación binaria la idea principal de SVM consiste en dividir los datos de la mejor manera. La clasificación binaria se utiliza cuando se requiere separar dos conjuntos de datos. Inicialmente, SVM se diseño para problemas de clasificación binaria. SVM considera dos enfoques:

- Cuando los datos son linealmente separables.
- Cuando los datos no son linealmente separables.

Considerando el primer caso; hay muchos límites de decisión lineales que dividen los datos. Pero sólo uno de ellos alcanza la máxima división. El límite de decisión para clasificar, puede terminar más cerca de un conjunto de conjuntos de datos en comparación con otros, por lo tanto, el concepto de clasificador de margen máximo o hiperplano es evidente solución. Los vectores de soporte son los puntos de datos que se encuentran más cerca de la superficie de decisión. El principal problema aquí es encontrar el único margen óptimo del hiperplano de separación $\boldsymbol{W^T x + b = 0}$, que proporciona el margen máximo entre las clases. Este margen garantiza la más baja de clasificación errónea.

Para permitir cierta flexibilidad en la separación de los límites de decisión, los modelos SVM tienen un parámetro de costo C que controla la compensación entre permitir errores de entrenamiento y forzar márgenes estrictos. Crea un margen suave que permite algunas clasificaciones erróneas. El parámetro de regularización C le dice al clasificador cuánto quiere evitar el error en la clasificación de cada ejemplo de entrenamiento. Para valores grandes de C, la optimización selecciona un pequeño margen del hiperplano, que hace un mejor trabajo obteniendo todos lo puntos de entrenamiento correctamente clasificados. Por el contrario, un pequeño valor de C va a provocar que se busque un grande margen que separe los hiperplanos, aún si clasifica erróneamente más puntos. En la Figura 2.10, se muestra cómo el parámetro de regularización afecta la clasificación de los ejemplos.

Cuando se utiliza un kernel no lineal, en el caso de un kernel de base radial. El parámetro gamma define hasta dónde llega la influencia de un sólo ejemplo de entrenamiento, con valores bajos que significan lejos y valores altos que significan cercanos. En otras palabras,

Figura 2.11 Dos diferentes parámetros de gamma. Izq: Gamma baja, los puntos lejanos también son considerados. Der: Gamma alta, sólo los valores cercanos son tomados en cuenta.

con gamma baja, los puntos lejos de la línea de separación plausible se consideran en el cálculo de la línea de separación. Donde como alto gamma significa que los puntos cercanos a la línea plausible se consideran en el cálculo. En la Figura 2.11, se muestra cómo el parámetro gamma afecta la clasificación de los ejemplos.

Ahora, en el segundo caso, los datos no son linealmente separables, es decir, en tales casos no se puede encontrar una línea recta que divida las clases. Los SVM lineales se pueden extender para generar SVM no lineales para la clasificación de datos no separables linealmente.

2.12.1.1. Clasificación multiclase

Los SVM se propusieron principalmente para tratar con la clasificación binaria, pero en los problemas reales hay una gran cantidad de datos que clasificar. Por lo tanto, esto crea la necesidad de una clasificación multiclase. Clasificación multiclase significa clasificación con más de dos clases. Las clasificaciones multiclase a través de una clasificación binaria se puede realizar mediante estos dos enfoques:

- Uno-contra-uno.
- Uno contra todos.

En general, el método más frecuente consiste en construir clasificadores de uno contra el resto o clasificación OVA, donde cada categoría se divide y todas las otras categorías se combinan para elegir la clase que clasifica los datos de prueba con el mayor margen. Divide un problema de clase en m problemas binarios. El paso de aprendizaje de los clasificadores se hace por toda la información de entrenamiento, considerando los patrones de la clase particular como positivos y todos los demás ejemplos como negativos. En la fase de validación, se presenta un patrón a cada uno de los clasificadores binarios y luego el clasificador que proporciona un resultado positivo indica la clase de salida. En numerosos casos, el resultado positivo no es único y algunas técnicas de desempate son obligatorias. El enfoque más familiar utiliza la confianza de los clasificadores para decidir el resultado final, prediciendo la clase del clasificador con la máxima confianza.

Otra estrategia se basa en construir un conjunto de clasificadores de uno contra uno dividiendo un problema de clase m en $m(m-1)/2$ problemas binarios. Cada problema surge por un clasificador binario que es responsable de distinguir entre un par diferente de clases. La fase de aprendizaje de los clasificadores se realiza utilizando como datos de entrenamiento sólo un subconjunto de instancias del conjunto de datos original que contiene cualquiera de

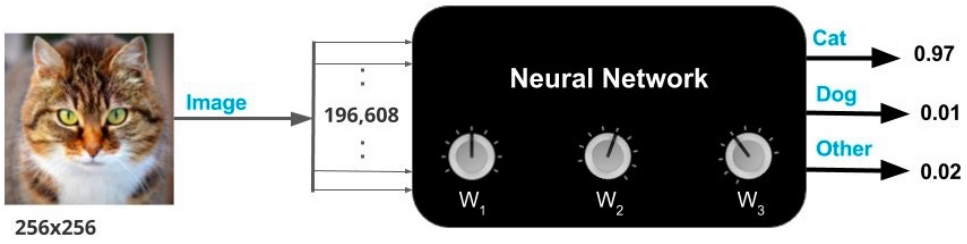


Figura 2.12 Red neuronal representada como una caja negra.

las dos etiquetas de clase correspondientes, mientras que las instancias con etiquetas de clase diferentes simplemente se ignoran.

En la fase de validación, se presenta un patrón a cada uno de los clasificadores binarios. El resultado de un clasificador dado por r_{ij} en [0, 1] es la confianza del clasificador binario que discrimina las clases i y j en favor de la clase anterior. La clase con la mayor confianza es la clase de salida de un clasificador. Estos resultados están representados por una matriz de puntaje R:

$$R = \begin{bmatrix} - & r_{12} & \dots & r_{1m} \\ r_{21} & - & \dots & r_{2m} \\ \dots & & & \dots \\ r_{m1} & r_{m2} & \dots & - \end{bmatrix} \tag{2.17}$$

Hay tanta diferencia entre estos dos enfoques. Construcciones de estrategia de uno contra todos al ajustar un clasificador por clase donde una estrategia de uno contra uno construye un clasificador por par de clases. El beneficio del enfoque uno contra todos es su interpretabilidad. Esta es la estrategia más comúnmente utilizada y es una opción justa por defecto. El clasificador uno contra uno no escala bien con n muestras.

2.12.2. Red neuronal

Una red neuronal, vista como una caja negra, funciona de la siguiente manera: cuando se ingresa una imagen (normalmente transformada en forma de vector) a la red neuronal, se obtiene la probabilidad de pertenencia de los datos respecto a las diferentes clases, dando como resultado un vector con la probabilidad por cada clase. Se toma como resultado final la más alta probabilidad de todas las clases. Para que la red pueda reconocer la imagen, se debe ajustar la red mediante entrenamiento, usando ejemplos de diferentes imágenes para que la red pueda ajustar sus parámetros. En la figura 2.12 se muestra un ejemplo de clasificación de una imagen visto como una caja negra.

Una red neuronal es una colección de neuronas conectadas por sinapsis. Esta colección está organizada en tres capas principales: la capa de entrada, la capa oculta y la capa de salida. Se pueden tener muchas capas ocultas. En una red neuronal artificial, hay varias entradas y producen una única salida, que se llama etiqueta. En la figura 2.13 se muestra una red neuronal con dos capas ocultas.

Cada neurona produce una salida, o activación, basada en las salidas de la capa previa y un conjunto de pesos. La función de activación se usa para determinar la salida de la red

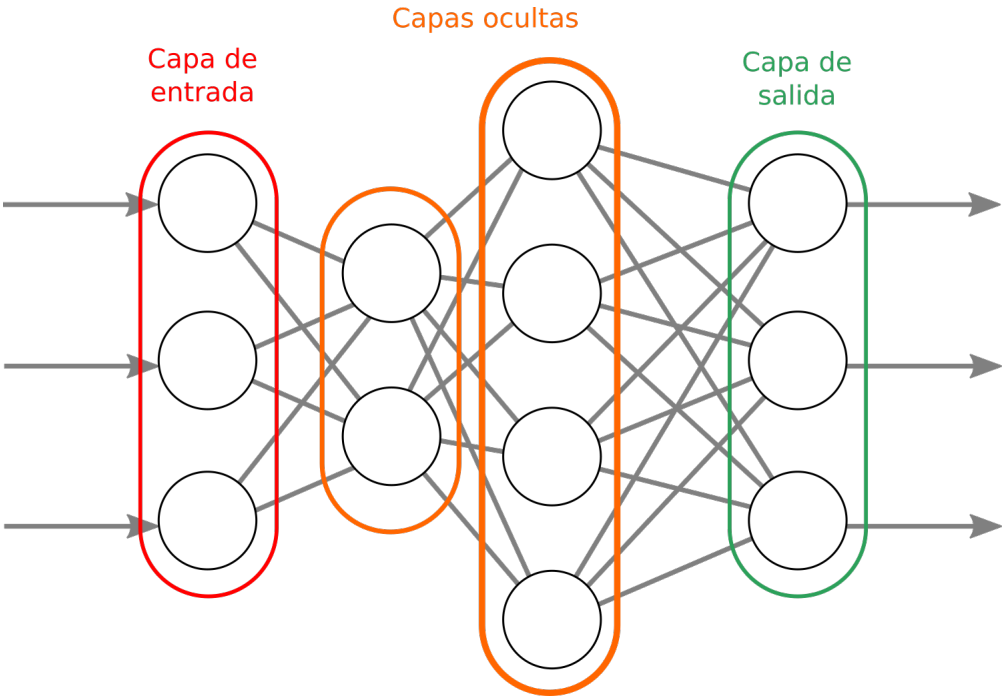


Figura 2.13 Red neuronal representada con dos capas ocultas.

neuronal, correlaciona los valores resultantes de 0 a 1 o -1 a 1, etc. (dependiendo de la función de activación).

Las funciones de activación no lineal son las funciones de activación más utilizadas. La no linealidad ayuda a que se puedan aprender patrones en forma de curvas como los puntos en la imagen del gráfico de la figura 2.14.

2.12.2.1. Funciones de activación

Función sigmoidal o logística Función de activación sigmoidal o logística, la cual se muestra en la Figura 2.15 es una función de activación de la forma $f(x) = \frac{1}{1+e^{-x}}$. Su rango está entre 0 y 1; y es una curva en forma de S. Es fácil de entender y aplicar, pero tiene

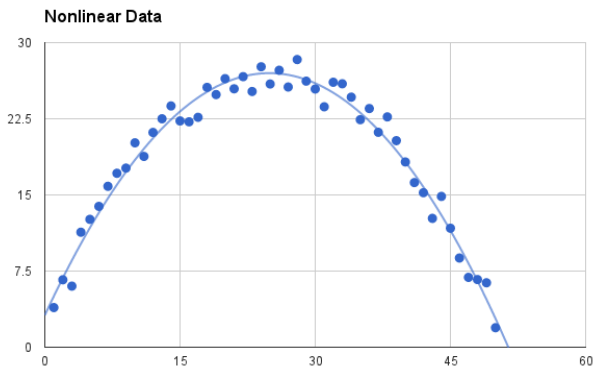


Figura 2.14 Función de activacion no lineal.

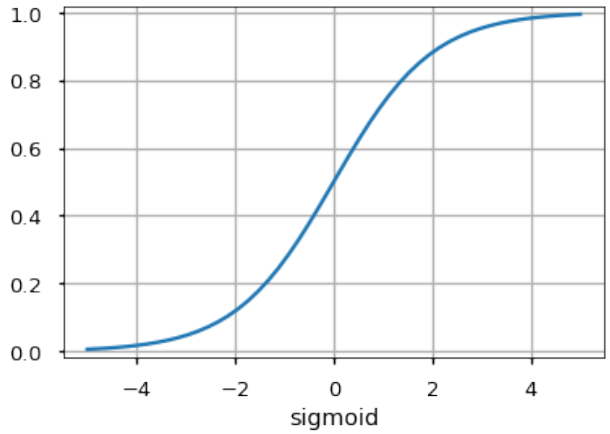


Figura 2.15 Función de activacion sigmoidal.

importantes razones que la han hecho disminuir su popularidad:

- Sufre del problema de Vanishing gradient, que es un problema que surge en las redes neuronales que son entrenadas con métodos basadas en gradientes. Este problema hace difícil que la red neuronal aprenda y resulta difícil de ajustar los parámetros de las primeras capas de la red. El problema se vuelve peor cuando se incrementa el número de capas en la arquitectura.
- Su salida no está centrada en cero. O sea, las actualizaciones de gradiente van demasiado lejos en diferentes direcciones y hace la optimización más difícil.
- La función logística sigmoidea puede hacer que una red neuronal se atasque en el tiempo de entrenamiento y, además, su convergencia es lenta..

Función softmax La función softmax es una función de activación logística más generalizada que se utiliza para la clasificación multiclase. Se expresa como: $\frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}}$ para $j = 1, .., K$ donde: z es un vector de K dimensiones. Esta función se utiliza en la capa de salida, para poder realizar la clasificación multiclase, a diferencia de la función sigmoidal, que se utiliza en la capa de salida para la clasificación binaria.

Función tangente hiperbólica (tanh) Se expresa como $f(x) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$. Su salida está centrada en cero porque su rango está entre -1 y 1. Por lo tanto, la optimización es más fácil, por lo que en la práctica siempre se prefiere este sobre la función Sigmoidal. Pero aún sufre de un problema de Vanishing gradient.

En la figura 2.16 se muestra la función tangente hiperbólica.

Unidad de rectificación lineal (ReLu) Se ha vuelto muy popular en los últimos años. Recientemente se demostró que la convergencia es 6 veces más rápida que la función tanh. Es sólo $R(x) = max(0, x)$, es decir, si $x < 0, R(x) = 0$ y si $x >= 0, R(x) = x$. Por lo tanto, al ver la forma matemática de esta función, se puede ver que es muy simple y eficaz. Esta función

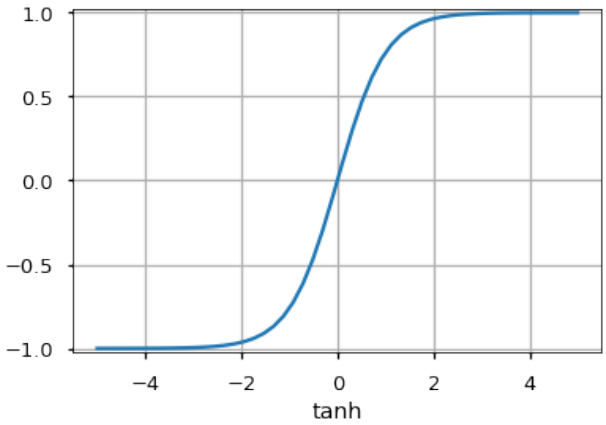


Figura 2.16 Función de activacion tangente hiperbólica.

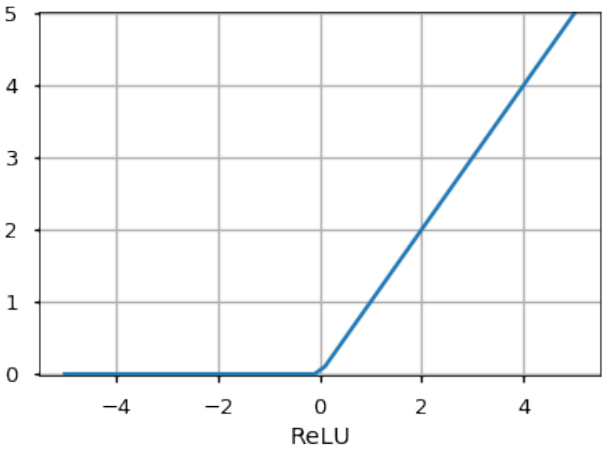


Figura 2.17 Función de unidad de rectificación lineal.

evita y rectifica el problema de Vanishing gradient. Casi todos los modelos de aprendizaje profundo usan ReLu hoy en día. Pero su limitación es que sólo debe usarse dentro de capas ocultas de un modelo de red neuronal. Por lo tanto, para las capas de salida, se debe utilizar una función Softmax para un problema de clasificación, para calcular las probabilidades de las clases, y para un problema de regresión, simplemente debería usar una función lineal.

Un problema con ReLu es que algunos gradientes pueden ser frágiles durante el entrenamiento y pueden bloquearse. Puede causar una actualización de peso que hará que nunca vuelva a activarse en ningún punto de datos. Simplemente diciendo que ReLu podría resultar en Neuronas bloqueadas.

En la figura 2.17, se muestra la función unidad de rectificación lineal.

2.12.2.2. Función de costo

La función de costo analiza la función que la red ha inferido y la utiliza para estimar los valores de los puntos de datos en el conjunto de entrenamiento. Las diferencias entre los resultados en las estimaciones y los puntos de datos del conjunto de entrenamiento son los

valores principales para la función de costo. Al entrenar la red, el objetivo será conseguir que el valor de esta función de costo sea lo más bajo posible.

El papel de una sinapsis es tomar la multiplicación de las entradas y los pesos. Los pesos son la conexión entre las neuronas. Los pesos definen principalmente la salida de una red neuronal. Sin embargo, son altamente flexibles. Después, se aplica una función de activación para devolver una salida.

2.12.2.3. Red neuronal de propagación hacia adelante

A continuación se describe como funciona una simple red neuronal feedforward:

- Toma entradas como una matriz (matriz 2D de números).
- Multiplica la entrada por un conjunto de pesos (realiza un producto de puntos, también conocido como multiplicación de matrices).
- Aplica una función de activación.
- Devuelve una salida.
- El error se calcula tomando la diferencia de la salida deseada de los datos y la salida pronosticada. Esto crea el descenso de gradiente, se puede emplear para alterar los pesos.
- Los pesos se alteran ligeramente según el error.
- Para entrenar, este proceso se repite n veces. Cuanto más entrenados los datos, más precisos serán los resultados.

2.12.2.4. Red neuronal de propagación hacia atrás

Debido a que se tiene un conjunto aleatorio de ponderaciones, estas se deben modificar para que las entradas sean iguales a las salidas correspondientes del conjunto de datos. Esto se hace a través de un método llamado backpropagation.

Backpropagation funciona mediante el uso de una función de pérdida para calcular qué tan lejos estaba la red del resultado objetivo.

2.12.2.5. Cálculo de error

Una forma de representar la función de pérdida es mediante el uso de la función del error cuadrático medio: $Loss = \sum(0{,}5)(o - y)^2$. Donde: o es el resultado predicho, y y es el resultado real. Ahora que se tiene la función de pérdida, el objetivo es acercarla lo más posible a 0. Eso significa que se debe tener prácticamente ninguna pérdida. Mientras se entrena la red, todo lo que se hace es minimizar la pérdida

Para determinar en qué dirección modificar las ponderaciones, se necesita encontrar la tasa de cambio de la pérdida con respecto a las ponderaciones. En otras palabras, se necesita usar la derivada de la función de pérdida para comprender cómo los pesos afectan la entrada.

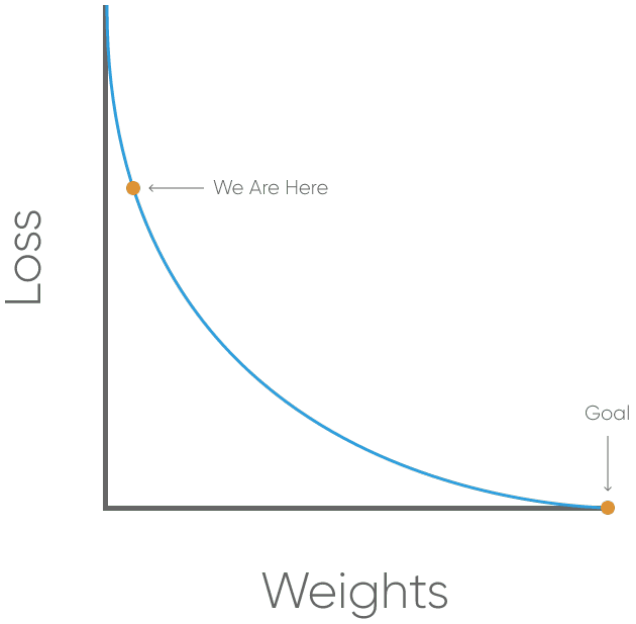


Figura 2.18 Descenso del gradiente.

Este método se conoce como descenso de gradiente. Al saber en qué forma alterar los pesos, los resultados pueden ser más precisos, como ese muestra en la Figura 2.18.

Así es como se calcula el cambio incremental de los pesos:

1. Localizar el margen de error de la capa de salida (o) tomando la diferencia de la salida pronosticada y la salida real (y)

2. Aplicar la derivada de la función de activación sigmoidal al error de la capa de salida. Nombrando a este resultado la suma de salida delta.

3. Se utiliza la suma de salida delta del error de la capa de salida para determinar cuánto contribuyó la capa z_2 (oculta) al error de salida mediante la ejecución de un producto de puntos con la segunda matriz de ponderación. Llamando a esto el error z_2.

4. Calcular la suma de salida delta para la capa z_2 aplicando la derivada de la función de activación sigmoidal (como en el paso 2).

5. Ajustar los pesos para la primera capa realizando un producto de puntos de la capa de entrada con la suma de salida delta oculta (z_2). Para el segundo peso, realizar un producto de puntos de la capa oculta (z_2) y la salida (o) delta output sum.

Calcular la suma de salida delta y luego aplicar la derivada de la función sigmoidea son muy importantes para la propagación inversa. La derivada del sigmoide dará como resultado la tasa de cambio, o pendiente, de la función de activación en la suma de salida.

La tasa de aprendizaje es uno de los hiperparámetros más importantes para la red neuronal. Si la tasa de aprendizaje es demasiado alta, podría ir demasiado lejos en la otra dirección, y nunca llegar al mínimo que se está buscando. Si se establece demasiado bajo, la red tardará mucho tiempo en encontrar los pesos correctos, o se quedará atascada en un mínimo local.

No hay un "número mágico"para usar, cuando se trata de una tasa de aprendizaje, y generalmente es mejor probar varios y seleccionar el que mejor funcione para la red y conjunto de datos individuales. En la práctica, se opta por recobrar la velocidad de aprendizaje a lo largo del tiempo: comienza alto, porque está más alejado de la solución y decae a medida que se acerca.

2.12.2.6. Razón de aprendizaje

Los modelos de aprendizaje profundo generalmente son entrenados por un optimizador de descenso del gradiente estocástico. Hay muchas variaciones del descenso del gradiente estocástica: Adam, RMSProp, Adagrad, etc. Todas permiten establecer el valor de la tasa de aprendizaje. Este parámetro le dice al optimizador qué tan lejos debe mover los pesos en la dirección del gradiente para un mini lote.

Si la tasa de aprendizaje es baja, el entrenamiento es más confiable, pero la optimización llevará mucho tiempo porque los pasos hacia el mínimo de la función de pérdida son pequeños. Si la tasa de aprendizaje es alta, el entrenamiento puede no converger o incluso divergir. Los cambios de peso pueden ser tan grandes que el optimizador sobrepasa el mínimo y empeora la pérdida.

El entrenamiento debe comenzar desde una tasa de aprendizaje relativamente grande porque, al principio, los pesos aleatorios están lejos de ser óptimos, y luego la tasa de aprendizaje puede disminuir durante el entrenamiento para permitir actualizaciones de peso más finas.

Cuando se comienza a entrenar con una gran tasa de aprendizaje, la pérdida no mejora y probablemente incluso crece mientras corren las primeras iteraciones de formación. Al entrenar con una tasa de aprendizaje más pequeña, en algún punto el valor de la función de pérdida comienza a disminuir en las primeras iteraciones.

Hay varias formas de seleccionar un buen punto de partida para la tasa de aprendizaje. Un enfoque sencillo es probar algunos valores diferentes y ver cuál le da la mejor pérdida sin sacrificar la velocidad de entrenamiento. Se puede comenzar con un valor grande como 0.1, luego tratar con valores exponencialmente más bajos: 0.01, 0.001, etc.

2.13. Reducción de dimensionalidad

En el aprendizaje automático, "dimensionalidad"simplemente se refiere al número de características (es decir, variables de entrada) en su conjunto de datos.

Cuando el número de funciones es muy grande en relación con el número de observaciones en su conjunto de datos, ciertos algoritmos les resulta difícil formar modelos efectivos. Esto se denomina "Maldición de la dimensionalidad", y es especialmente relevante para los algoritmos de agrupamiento que se basan en cálculos de distancia. La dificultad de buscar en el espacio se vuelve mucho más difícil ya que se tienen más dimensiones.

Existen varios métodos de reducción de dimensionalidad, pero los que se van a tratar son los algoritmos de extracción de características. Estos algoritmos crean un conjunto nuevo y más pequeño de características que aún captura la mayor parte de la información útil.

2.13.1. Análisis de componentes principales (PCA)

El análisis de componentes principales (PCA) es un algoritmo no supervisado que crea combinaciones lineales de las características originales. Las nuevas características son ortogonales, lo que significa que no están correlacionadas. Además, se clasifican por orden de su "varianza explicada". El primer componente principal (PC1) explica la mayor variabilidad en su conjunto de datos, PC2 explica la segunda variante más, y así sucesivamente.

Por lo tanto, se puede reducir la dimensionalidad al limitar el número de componentes principales para mantener en función de la varianza acumulativa. Por ejemplo, se puede decidir mantener sólo tantos componentes principales como sea necesario para alcanzar una varianza acumulativa del 90 %.

Siempre se debe normalizar su conjunto de datos antes de realizar PCA porque la transformación depende de la escala. Si no se realiza la normalización, las características que están en la escala más grande predominaran como componentes principales. Las fortalezas de este algoritmo son:

- Es una técnica versátil.
- Es rápido y simple de implementar, lo que significa que se pueden probar fácilmente algoritmos con y sin PCA para comparar el rendimiento.
- Ofrece variaciones y extensiones para abordar obstáculos específicos.

Las debilidades de este algoritmo son:

- Los nuevos componentes principales no son interpretables, lo que puede ser un factor decisivo en algunos entornos.
- Se debe establecer manualmente o ajustar un umbral para la varianza acumulativa.

2.13.2. Análisis discriminante lineal (LDA)

El análisis discriminante lineal (LDA), también crea combinaciones lineales de sus características originales. Sin embargo, a diferencia de PCA, LDA no maximiza la varianza acumulada. En cambio, maximiza la separabilidad entre clases. Por lo tanto, LDA es un método supervisado que sólo se puede usar con datos etiquetados. Los resultados de los algoritmos LDA y PCA varían de un problema a otro, por lo tanto es necesario probar los dos algoritmos. La transformación LDA también depende de la escala, por lo que primero se debe normalizar su conjunto de datos. Las fortalezas de este algoritmo son:

- Es supervisado, lo que puede (pero no siempre) mejorar el rendimiento predictivo de las características extraídas.
- Ofrece variaciones para abordar obstáculos específicos.

Las debilidades de este algoritmo son:

- Al igual que con PCA, las nuevas características no se pueden interpretar fácilmente, y todavía se debe configurar o ajustar manualmente la cantidad de componentes que desea conservar.
- LDA también requiere datos etiquetados, lo que lo hace más situacional.

Capítulo 3

Trabajos Previos

3.1. Mexican sign language recognition using movement sensor

R. Galicia y colaboradores [10], diseñaron un sistema que utiliza un sensor de movimiento para capturar imágenes de profundidad y el esqueleto de la mano humana para rastrear movimientos considerando la altura y profundidad de la mano.

3.2. Automatic Mexican Sign Language Recognition Using Normalized Moments and Artificial Neural Networks

Francisco Solís y colaboradores [11], propusieron un sistema de visión artificial el cual utiliza cuatro reflectores led para mejorar la iluminación durante la adquisición de la imagen. Este sistema realiza el reconocimiento de veintiuna señas estáticas cuya descripción está dada por momentos centrales normalizados invariantes a transformaciones de escala y rotación. Obtienen un 93 % de exactitud con un perceptrón multicapa para la etapa de reconocimiento.

3.3. Real-Time Mexican Sign Language Recognition

Sosa-Jiménez y colaboradores [12], proponen un sistema para la interpretación de palabras usando un sensor Kinect para capturar los gestos, usando momentos geométricos como extractor de características y usando modelos ocultos de Markov como clasificador para encontrar las relaciones entre la secuencia de imágenes. Se realizaron experimentos con 10 personas, obteniendo una sensibilidad promedio de 80 % y especificidad de 80 %.

3.4. Automatic Translation System from Mexican Sign Language to Text

Aguilar [13], proponen un sistema para la traducción automatica de lenguaje de señas mexicano a texto, para facilitar la comunicación entre la gente sorda y la que no es sorda. Usan una webcam para capturar las imagenes, con el algoritmo Theo Pavlidis obtienen el contorno de la imagen, inmediatamente obtienen un polígono convexo que envuelve el contorno de la mano a través del algoritmo convex hull, también buscan el menor círculo que cubra el polígono y utilizan el algoritmo Wen and Niu para localizar los dedos. Con los algoritmos mencionados extraen las características de la mano pero no utilizan ningún clasificador, solamente comparan el vector de características obtenido con vectores previamente almacenados. El porcentaje de exactitud que reportan el del 80 %.

3.5. Hand Gesture Recognition Using PCA

Ahuja y Singh [14], abordan el problema del reconocimiento de gestos para controlar un brazo robótico, con el objetivo de usarlo en aplicaciones quirúrgicas. Proponen un método para reconocer gestos de las manos, utilizando segmentación basada en el color de la piel en el espacio de color YCbCr. Posteriormente emplean un umbral para extraer el fondo de la imagen aplicando una técnica de coincidencia de plantilla, y utilizan PCA para el reconocimiento. El sistema permite dar instrucciones a la mano robótica remotamente, lo que permite agregar exactitud a las operaciones, pero el modelo propuesto no es capaz de trabajar con imágenes de manos que tengan otro color de piel ni con imágenes que tengan otro color de luz, sólo con gestos estáticos predefinidos.

3.6. A Framework for Recognition of Hand Gesture in Static Postures

Vishwakarma y colaboradores [15], abordan de manera general el tema del reconocimiento de gestos de las manos para la interacción hombre-máquina. El método propuesto inicia con la extracción de la región de la piel en el espacio de color YCbCr, posteriormente se aplica un algoritmo de mapa de prominencia y un filtro de Gabor para extraer características de las texturas, en varias orientaciones y escalas, posteriormente emplean PHOG para extraer la figura de la mano, calculando la distribución espacial de la imagen de la piel de rasgos sobresalientes. Por último las características extraídas son clasificadas mediante SVM. Los resultados que obtienen son mejores frente al mismo conjunto de imágenes que otros métodos como descriptores espacio temporales, descriptores de Fourier y patrones locales binarios. El sistema es independiente de la variación de la luz, también es invariante a rotación y escala. Sin embargo, el sistema tiene algunos errores, si el fondo contiene grandes áreas parecidas al color de la piel humana.

3.7. Indian Sign Language Gesture Recognition

Sanil Jain y Kadi Vinay Sameer Raja [16], abordan el reconocimiento del lenguaje de señas indio. Se enfocan en el reconocimiento del alfabeto que, a diferencia del lenguaje de señas americano, utiliza las dos manos. El método inicia con la creación de un dataset extraído de video y que incluye todas las señas del alfabeto realizado por distintas personas. Las imágenes obtenidas del video se procesan en los espacios de color YIQ y YUB para realizar segmentación de piel. Para la extracción de características aplican el enfoque "bolsa de palabras" utilizando el algoritmo SURF como detector de características y descriptor. Posteriormente recurren a k-means para agrupar y así mapear la imagen al clúster más cercano. Partes similares son representadas con la misma palabra clave. Realizan un entrenamiento con 25 imágenes por alfabeto de tres personas y de prueba emplean 25 imágenes por alfabeto de una sola persona. Reportan sus resultados con un porcentaje de 33.84 % de exactitud, las observaciones que hace el autor, es que las señas del alfabeto son muy similares, y una mano puede cubrir características de la otra, lo que provoca una clasificación incorrecta. Una de las 3 personas era zurda lo que provocó que hubiera imágenes inversas.

3.8. Improved Face and Hand Tracking for Sign Language Recognition

El problema que tiene la gente sorda para comunicarse con personas que escuchan también se ha abordado [17]. Soontranon propone un método para la detección y seguimiento de las manos y cara, para un sistema de reconocimiento de lenguaje de señas, utilizando detección del color de la piel, luego separan la cara y las manos usando tamaño y características faciales. Basándose en que el movimiento de la cara es menor al de las manos, proponen un mapa de estimación de movimiento sólo en el seguimiento de las manos. Para aumentar la precisión, el método de seguimiento puede compensar el error considerando un área de búsqueda adaptativa, para recalcular el balanceo de las manos iterativamente. Obteniendo resultados que indican que el algoritmo propuesto puede seguir la cara y la mano con gran precisión, con insignificante incremento en la complejidad computacional. Este trabajo sólo realiza la detección de la mano, pero no realiza el reconocimiento.

Capítulo 4

Desarrollo de la solución

4.1. Análisis del lenguaje de señas

Para reconocer el alfabeto del lenguaje de señas es necesario sólo enfocarse en reconocer la configuración de la mano. Hay aspectos que se deben considerar cuando de trata de detectar y reconocer una mano, considerando que la mano es un modelo dinámico, debido a que la mano puede tener muchas formas, tomando en cuenta que la forma de la mano cambia de una persona a otra. Si varias personas, realizan la misma seña, suelen haber pequeñas variaciones al realizar una seña con la mano. Estas variaciones en las señas se aumentan si la mano se captura desde otro punto vista. Para simplificar este problema, se utilizan imágenes capturadas desde un sólo punto de vista. O sea, la cámara enfoca de frente a la persona.

Existen otros factores que afectan a las imágenes:

- La variación en el color de la piel, porque cada individuo tiene su propio tono de piel, y puede contener varios tonos. También los tonos de la piel están afectados por la iluminación, provocando que tengan otro color diferente. Es importante señalar que no se pretende abarcar todos los posibles colores de piel ni todas las condiciones de iluminación, solamente para los colores de piel con que son entrenados los algoritmos.
- La calidad de la cámara también influye en la calidad de la detección de la forma de la mano, debido a que la imagen puede contener ruido, entre menor calidad tenga la cámara, más notable es el ruido en la imagen. Por este motivo se utilizó una cámara de calidad media con una resolución de 1280 píxeles de ancho y 720 píxeles de alto.

Tomando en cuenta lo anteriormente mencionado el sistema de reconocimiento de señas requiere los siguientes pasos:

- Captura de las imágenes para entrenar y probar el sistema.
- Pre-procesamiento de la imagen, para eliminar información innecesaria.
- Extracción de características para encontrar patrones que permitan describir el objeto y diferenciarlo de otros objetos.
- Si las características son demasiadas, aplicar algoritmos de reducción de dimensionalidad a las características, para evitar problemas al entrenar los clasificadores.

- Entrenar un clasificador, para aprender los patrones y reconocer nuevas imágenes.

Dentro de estos pasos hay diferentes algoritmos que deben evaluarse y seleccionarse, se desea seleccionar la combinación de técnicas que tenga mayor exactitud, y la combinación que tenga mejor velocidad. Debido a que diferentes aplicaciones, requieren diferentes requerimientos de los algoritmos.

4.2. Captura de las imágenes

Para realizar el reconocimiento de un objeto, primero se obtiene un conjunto de imágenes que describan el problema, también conocido como dataset. Este conjunto de imágenes debe describir el ambiente real, lo más aproximado posible, agregando variaciones a la forma de la mano, cambiando la posición de la mano, girándola en algunas direcciones o incluso acercándola o alejándola. Este tipo de variaciones permiten comprobar, que el algoritmo de aprendizaje puede reconocer imágenes en posiciones con las que no fue entrenado, denominado aprendizaje.

En la búsqueda de un dataset para el lenguaje de señas mexicano, solamente se encontró uno [18], pero las imágenes les faltaba variación en traslación y escala, por lo cual se decidió realizar un dataset. Para crear un dataset se requiere obtener imágenes que pueden provenir de diferentes lugares de almacenamiento, ya sea en internet o en dispositivos de almacenamientos, en diferentes formatos. Estas imágenes son utilizadas para entrenar los algoritmos y para probarlos. Para crear un dataset del alfabeto del lenguaje de señas se requieren los siguientes pasos que permiten obtener imágenes de diferentes fuentes de información:

- Capturar una imagen desde una webcam o cámara y guardarlas en el almacenamiento del dispositivo.
- Extraer las imágenes desde un archivo de video y cargarlas en el programa.
- Leer una imagen desde una ruta y cargarla en el programa.
- Leer todas las imágenes dentro de una carpeta y cargarlas en el programa.
- Leer una imagen directamente desde una webcam o cámara y cargarla en el programa, sin guardarla.
- Guardar las imágenes en imagen o en video.

Los pasos anteriores solamente son para leer imágenes desde video, cámara web o desde imágenes de algún dispositivo de almacenamiento, por que son las fuentes más comunes de almacenamiento de imágenes.

4.3. Creación del dataset

El dataset debe contener imágenes de las diferentes letras del alfabeto pero con las siguientes variaciones:

- Traslación: La mano puede encontrarse en diferentes lugares de la imagen, o sea, en la parte de arriba, en la esquina, en el centro, abajo, derecha. Las diferentes posiciones deben capturarse, con pequeños movimientos de izquierda a derecha, arriba a abajo, con un cambio en la posición de la mano de pocos centímetros, hasta llegar al otro extremo. Se debe tratar de tener la mano con la misma forma, para que el clasificador generalice la variación en el cambio de posición, porque aleatoriamente se seleccionan algunas imágenes para entrenamiento y otras para prueba. Si no se aplica lo anterior, causará que el algoritmo se entrene con determinadas formas, entonces al predecir una nueva imagen, la forma será muy diferente a las formas con que se entrenó el clasificador, aumentando del número de errores.

- Rotación: Es necesario agregar variaciones en la rotación de la mano, para que el clasificador aprenda las posibles variaciones de cada seña cuando se gira la mano. La rotación de la mano en una imagen con respecto a otra, debe ser con un cambio muy pequeño, entre dos o tres grados de rotación entre cada captura, tratando de que no se mueva la posición de la mano, que solamente se rote la mano.

- Escala: La imagen puede ser capturarse desde cerca o lejos de la cámara, por esto se debe agregar variaciones en la distancia de la mano, para reconocer manos más cerca o lejos. Deben ser capturadas con variaciones de pocos centímetros de cambio entre cada captura.

Para hacer el proceso más rápido se capturó las imágenes desde la webcam y se guardaron como imagen en el dispositivo de almacenamiento, sin embargo, es recomendable agregar más imágenes al dataset desde otras fuentes, para aumentar la cantidad de información proporcionada al algoritmo de clasificación. Por cuestiones de facilidad se utilizó un fondo de color uniforme que no fuera similar al de la piel, en este caso de color verde, para ayudar al proceso de segmentación y tener solamente el objeto sin el fondo. Esto facilita la segmentación de las imágenes utilizadas en el entrenamiento. Las letras a reconocer son 21 y se tomaron 300 imágenes por cada letra del alfabeto, aplicando variaciones en traslación, escala y rotación.

En la Figura 4.1, se muestran 8 imágenes del dataset, realizando la letra A, todas la imágenes son diferentes y tienen pequeños cambios en la posición de la mano.

En la Figura 4.2, se muestran variaciones en la posición de la mano en traslación y rotación. La traslación se logra moviendo la mano de un lado hacia otro y la rotación se logra al girar la mano, en las imágenes de muestra con una flecha el movimiento que se realizó al capturar las imágenes. Al agregar estas variaciones se pretende que el algoritmo aprenda las posibles variaciones en la posición de la mano.

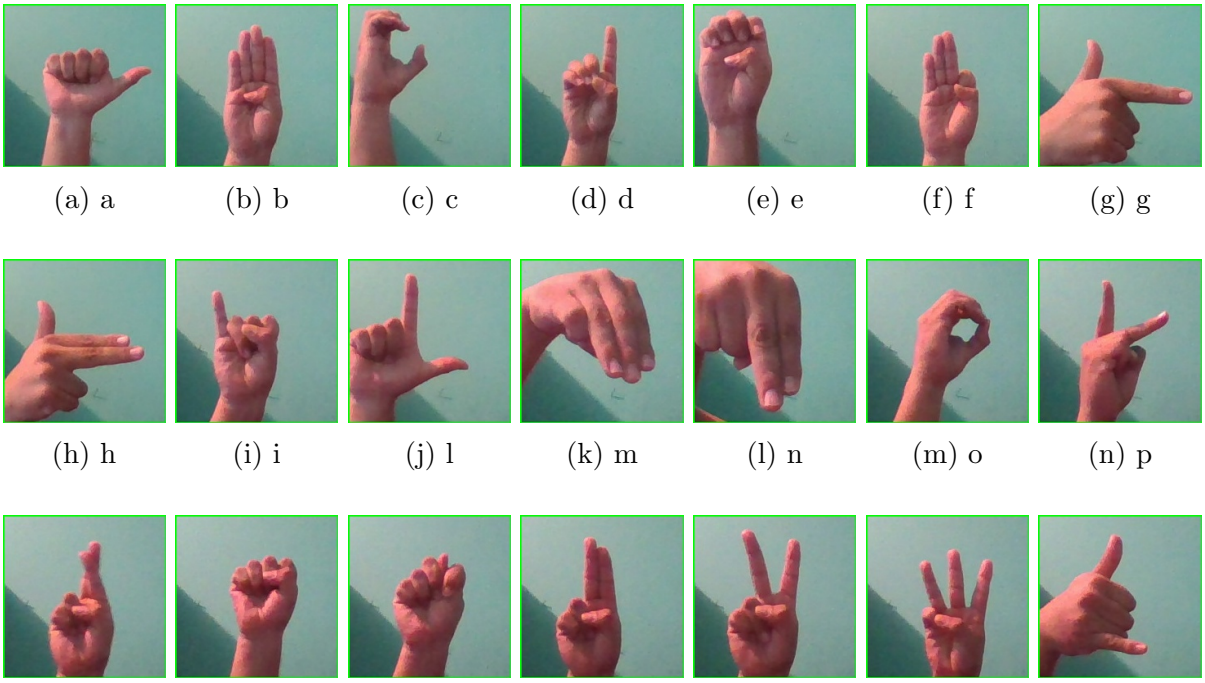
Las mismas variaciones son aplicadas a las demás letras del alfabeto. En la Figura 4.3, se muestra una imagen de ejemplo, para cada letra del alfabeto que se utilizó para crear el dataset, hay 300 imágenes en el dataset por cada letra, con las variaciones en la posición antes mencionadas.

Se capturaron 300 imágenes por cada seña, de las 21 que se consideraron, dando un total de 6300 imágenes. Las imágenes están separadas en 21 carpetas, como se muestra en la figura 4.4a. Hay una carpeta para cada letra y dentro se encuentras las imágenes para esa letra.

En la figura 4.4b se muestra que dentro de la carpeta *a* están todas la imágenes que corresponden a la seña de la letra a. Esto se realiza para facilitar la carga de las imágenes

Figura 4.1 Imágenes de ejemplo del dataset con la letra A con variaciones en la posición de la mano.

Figura 4.2 Imágenes de la letra A con variaciones en traslación, rotación.

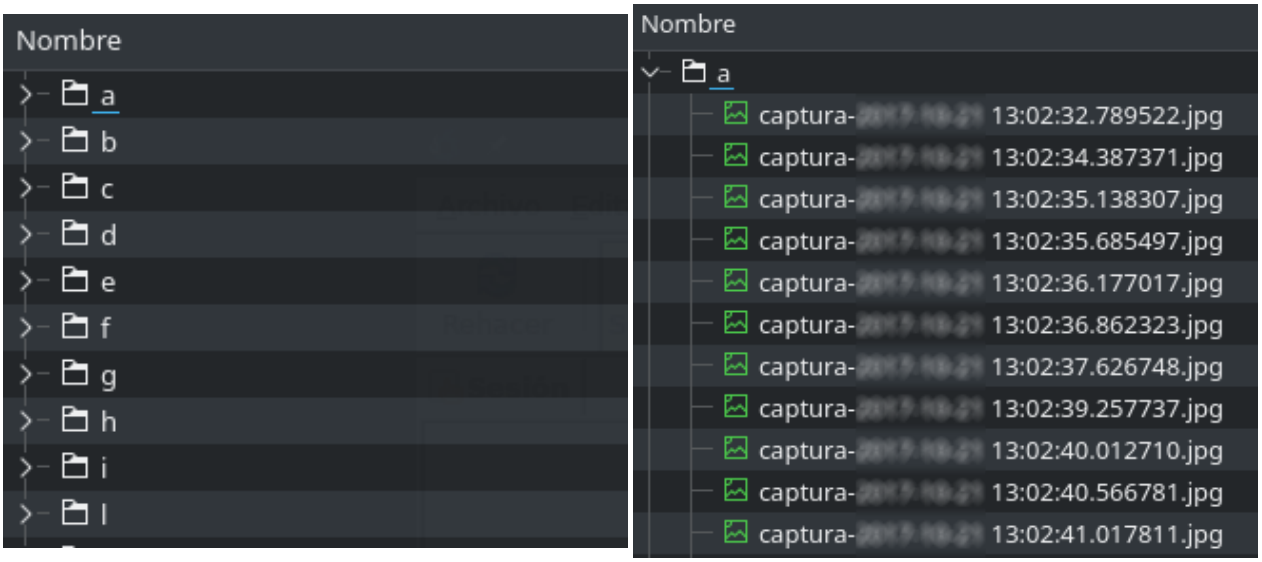


(a) a (b) b (c) c (d) d (e) e (f) f (g) g

(h) h (i) i (j) l (k) m (l) n (m) o (n) p

(ñ) r (o) s (p) t (q) u (r) v (s) w (t) y

Figura 4.3 Ejemplos de las imágenes de todas las señas usadas en el dataset. Sólo se incluyenlas letras estáticas.



(a) Ejemplos de carpetas con imagenes del alfabeto.

(b) Ejemplos de las imagenes dentro de la carpeta de la letra A.

Figura 4.4 Estructura del dataset organizado en carpetas.

al programa, evitando ponerles un nombre que este relacionado a la letra que representan las imágenes, por ejemplo $imagenA1, .., imagenA300$. Debido a que la carpeta contiene la información relacionada al tipo de letra que representan, esto permite que el nombre de la imagen sea independiente.

Para realizar la carga de las imágenes en el programa se necesita el nombre de la carpeta que contiene las carpetas con las imágenes. El algoritmo debe buscar todas las carpetas que se encuentren dentro de la carpeta principal y obtener el nombre de esta carpeta, ejemplo a, b hasta z. Por cada carpeta encontrada buscar las imágenes que hay dentro y cargarla una por una, el nombre de la carpeta que contienen las imágenes sirve como una etiqueta indicando la letra a la que pertenece.

Existen técnicas de extracción de características que permiten describir la imagen, y estas se ven afectadas por la forma de capturar el dataset, más adelante se explica como influye la captura del dataset en la extracción de características.

4.4. Análisis y selección de técnicas de preprocesamiento de imágenes

El pre-procesamiento de imágenes es un paso esencial en el mejoramiento de la información que contiene la imagen, debido a que permite eliminar información innecesaria que puede afectar el reconocimiento. Existen varios métodos para realizar esto, pero los métodos que se analizan en este trabajo son la eliminación de ruido y la segmentación.

El ruido afecta a las imágenes digitales debido a que es generado por los dispositivos de captura y se encuentra presente en todas la imágenes, la cantidad de ruido presente depende de la calidad de la cámara. Este tipo de ruido es conocido como ruido gausiano y afecta el procesamiento de las imágenes, debido a que muchos pixeles cambian de valor con cada captura, provocando que las imágenes del mismo objeto en la misma posición, tengan diferentes valores en algunos pixeles.

Las imágenes tienen diferente tamaño dependiendo del dispositivo de captura, por esto es necesario reajustar el tamaño de la imagen, para disminuir el tiempo de procesamiento, además de unificar el tamaño de las imágenes a procesar, que es un requerimiento del algoritmo de entrenamiento. Por tal motivo se debe seleccionar un tamaño de imagen que aumente la velocidad de procesamiento, pero que no pierda demasiada información, ya que puede provocar aumento en los errores en el reconocimiento, debido a la perdida de información al realizar el cambio de tamaño.

En el presente trabajo se utiliza una cámara web de calidad media, usando las técnicas para la reducción del ruido basadas en filtros (procesos de convolución).

4.4.1. Escalamiento de la imagen

Las imágenes que se utilizan en el entrenamiento de los algoritmos de reconocimiento de objetos suelen tener diferentes resoluciones, es decir un ancho y alto diferente. Se debe cambiar el tamaño de la imagen a un tamaño que permita mayor velocidad de procesamiento, pero sin perder demasiada información, que haga que el error en el reconocimiento aumente. Un tamaño grande de una imagen suele tomar mucho tiempo de procesamiento debido al

número de pixeles que se están manipulando. En la Tabla 4.1, se muestra el número de pixeles que tiene la imagen basado en el ancho y alto de un imagen, esto sin considerar la profundidad de color..

ancho	alto	total de pixeles
11	11	121
20	20	400
50	50	2500
100	100	10,000
200	200	40,000
500	500	250,000
1000	1000	1,000,000
3264	2448	7,990,272

Tabla 4.1 Incremento en el número de pixeles basados en el ancho y alto.

Se puede notar en la Tabla 4.1, que entre más grande sea el ancho y alto de la imagen, el número de pixeles aumenta de una manera significativa, esto provoca que la imagen requiera un mayor tiempo de procesamiento y consecuentemente un mayor tiempo de respuesta. Se seleccionó un tamaño de imagen de 200 de ancho por 200 de alto, para la presente aplicación debido a que conserva la suficiente información de la mano, disminuyendo el tiempo de procesamiento.

Es importante seleccionar los filtros que mejor se ajusten al problema, ya que algunos filtros eliminan información importante de la imagen, como son los bordes o la estructura de los objetos, debido a esto es necesario realizar un análisis para elegir los mejores filtros con sus mejores valores para presente aplicación.

4.4.2. Filtro promedio

El kernel de este filtro puede tener tamaños impares, por ejemplo 3, 5, 7. Entre más grande sea el tamaño del kernel, más suavizada será la imagen, perdiendo detalles de la imagen.

En la figura 4.5, se muestra la aplicación de los filtros de tamaño 3 y 5. Se puede notar que la imagen pierde en gran medida los bordes, cuando se aumenta el tamaño del filtro a 5×5. De manera visual, se obtienen mejores resultados con el filtro de 3×3.

4.4.3. Filtro gausiano

Los parámetros necesarios de este filtro son: tamaño de kernel y valor de sigma, que es la desviación estándar.

En la figura 4.6, se muestra el filtro gausiano de tamaño 5×5 con el valor de $\sigma = 1,2$. Cuando se aumenta el valor de σ, es muy efectivo reduciendo ruido gausiano. Sin embargo la imagen empieza a suavizarse perdiendo los bordes y detalles de la imagen. Por lo tanto el mejor valor para la imagen es $\sigma = 1$.

(a) original (b) filtro 3x3 (c) filtro 5x5

Figura 4.5 Resultados de filtro promedio al aumentar el tamaño del filtro.

(a) original (b) $\sigma = 1$ (c) $\sigma = 2$

Figura 4.6 Resultados de filtro gausiano al aumentar el tamaño en los valores de $\sigma = 1, 2$.

(a) original (b) mediana de 3x3 (c) mediana de 5x5

Figura 4.7 Resultados de filtro mediana al aumentar el tamaño del filtro.

4.4.4. Filtro mediana

Tiene la ventaja de que el valor final del pixel es un valor real presente en la imagen y no un promedio, de este modo se reduce el efecto borroso que tienen las imágenes que han sufrido un filtro de media. Pero es más complejo de calcular ya que hay que ordenar los diferentes valores que aparecen en los pixeles de la ventana y determinar cual es el valor central.

En la figura 4.7, se muestra los resultados al aumentar el tamaño del kernel de 3 y 5. Los mejores resultados se obtienen con el kernel de tamaño 5. La imagen se ve menos borrosa comparada con el uso de otros filtros, pero cuando el kernel es muy grande se pierden los bordes.

4.4.5. Filtro bilateral

Es un filtro muy efectivo para remover ruido, preservando los bordes. Pero sus operaciones son más lentas comparadas con otros filtros. El filtro bilateral es controlado por dos parámetros σ_s y σ_r:

- Cuando el rango σ_r incrementa, el filtro bilateral llega a ser más cercano al difuminado gaussiano, porque el rango gaussiano es plano.
- Incrementando el parámetro espacial σ_s suaviza características más grandes.

En la Figura 4.8, se muestra como el aumento de d, se elimina una mayor cantidad de ruido, pero hace la textura más suave perdiendo detalles de la imagen cuando el valor aumenta, también aumenta el tiempo de procesamiento. Por lo tanto el valor de $d = 5$ elimina suficiente ruido conservando los detalles de la mano y permitiendo un tiempo de procesamiento menor.

En la figura 4.9, se muestra el aumento de σ. Esto suaviza la imagen conservan los bordes de la imagen, pero aumenta el efecto de caricatura. El valor de $\sigma = 130$ elimina suficiente cantidad de ruido, ya que valores mayores suavizan mucho la imagen. Por lo tanto, los mejores valores para las imágenes se obtiene con los parámetros de $d = 5$ y $\sigma = 130$.

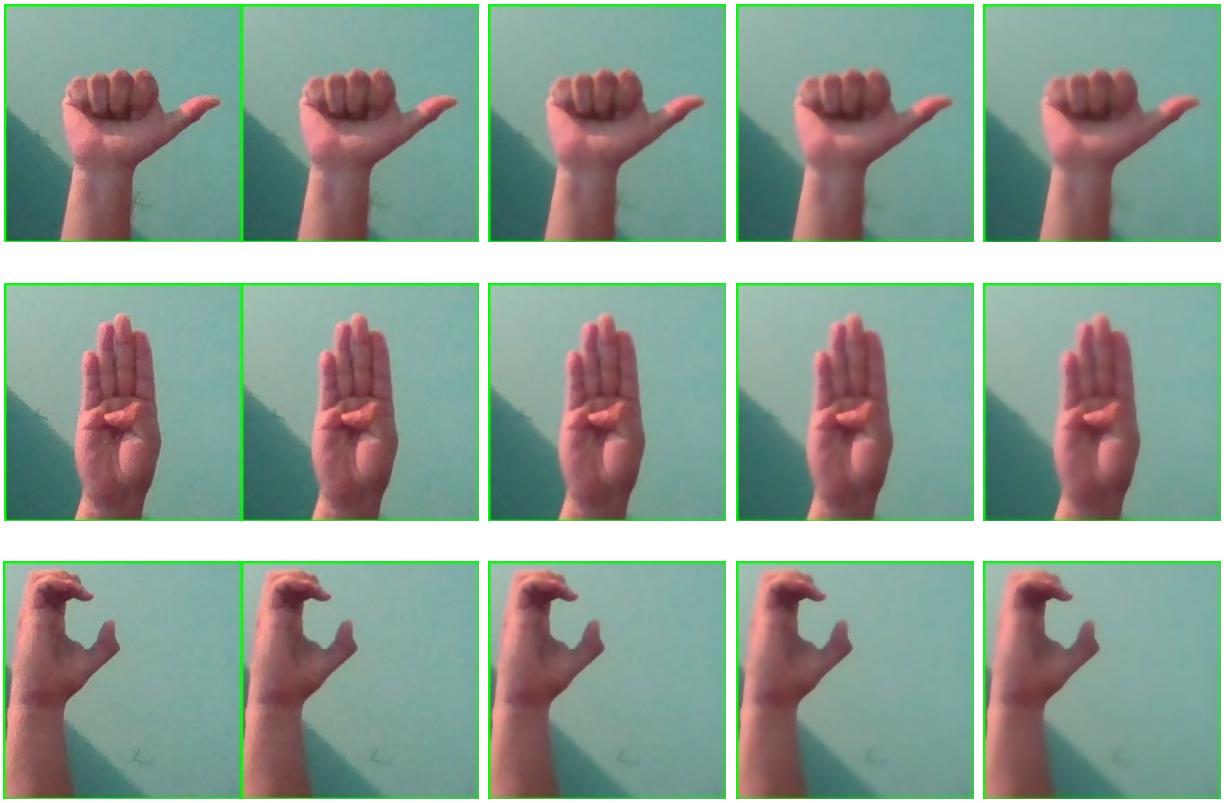
Los mejores resultados se obtienen con el filtro bilateral, debido a que conserva mejor los bordes, pero es más lento, debido a que realiza más operaciones. Sin embargo, ajustando los valores se puede aumentar la velocidad, logrando buenos resultados en eliminación de ruido.

4.5. Análisis y comparación de técnicas de eliminación del fondo

Se necesita eliminar el fondo de la imagen, para evitar tener información innecesaria, que puede interferir en los resultados de la detección de la mano. Debido a esto se requiere hacer un análisis de los métodos que tengan mejores resultados en el presente problema.

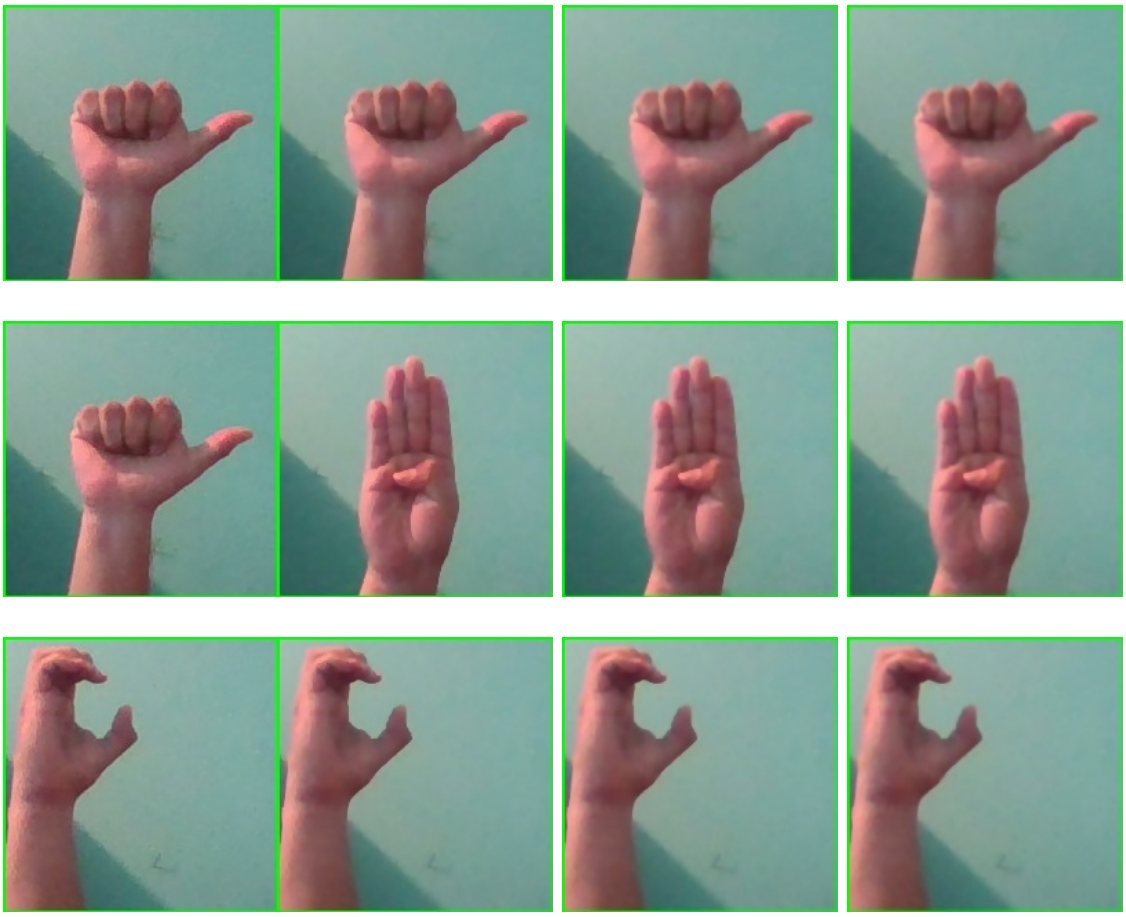
4.5.1. Sustracción del fondo

Es un método de eliminación del fondo que calcula la diferencia entre la imagen actual y la imagen de fondo. Este método elimina una gran cantidad del fondo, proporcionando



(a) imagen original (b) $d = 3$ y $\sigma = 50$ (c) $d = 5$ y $\sigma = 50$ (d) $d = 7$ y $\sigma = 50$ (e) $d = 9$ y $\sigma = 50$

Figura 4.8 Resultados de filtro bilateral al aumentar el valor de d.



(a) imagen original (b) $d = 5$ y $\sigma = 50$ (c) $d = 5$ y $\sigma = 90$ (d) $d = 5$ y $\sigma = 130$

Figura 4.9 Resultados de filtro bilateral al aumentar el valor de σ.

una imagen que contenga los objetos que son considerados como fondo. Sin embargo, este algoritmo se ve afectado por cambios en la iluminación, ruido en la imagen y por objetos moviéndose en el fondo. Es útil en ambientes donde el único movimiento realizado es la mano, se debe evitar que los objetos en el fondo se muevan o que haya cambios bruscos en la iluminación. Es conveniente en ambientes de oficina, casa o donde no haya movimiento en el fondo, y se tenga una iluminación estática.

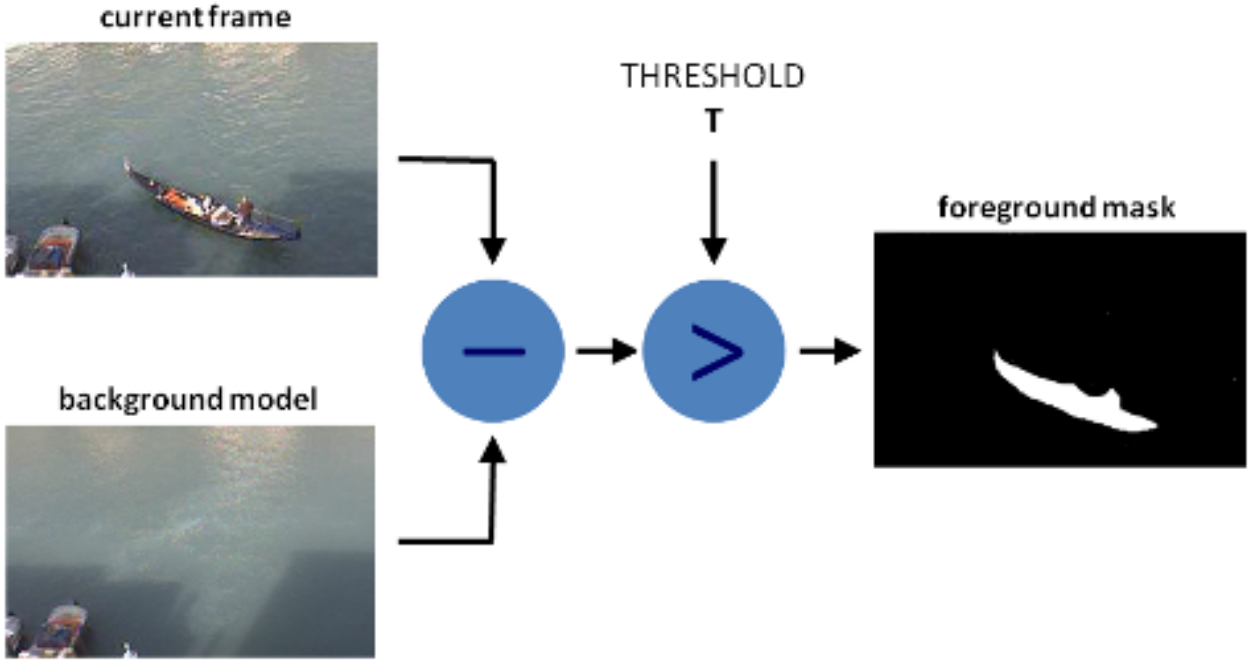


Figura 4.10 Funcionamiento de la técnica de sustacción de fondo.

En la Figura 4.10, se muestra de manera general el funcionamiento de la sustracción del fondo, como se aprecia se requiere ajustar un valor llamado threshold (umbral), que permite regular la diferencia aceptada entre la imagen del fondo y la imagen actual.

4.5.2. Segmentación basada en el color de la piel

Este algoritmo consiste en eliminar las partes de la imagen que no tengan un color similar al color de la piel, eliminando el fondo. Este funciona de la siguiente manera, dado un rango de valores, se busca en todos los pixeles de la imagen los valores que estén dentro de este rango y se marcan con uno, de lo contrario se marca con 0. Teniendo una imagen binaria, con valores de 1 la parte de la piel y con 0 el fondo. Cuando el fondo tiene cualquier color que no sea parecido al color de la piel, este algoritmo ofrece excelentes resultados. Sin embargo, existen situaciones donde no se elimina el fondo, debido a que puede tener un color parecido al de la piel. Otro problema es la selección de los valores que son reconocidos como color de piel, ya que dependen de las personas, razas e iluminación, por lo tanto elegir un valor para el color de la piel, es difícil, debido a que puede reconocer bien algunos tipos de piel, mientras que fallar en reconocer otros, o reconocer gran parte del fondo como color de la piel. En la Figura 4.11, se muestra la segmentación basada en el color de la piel. Se puede

notar que no toda la mano es considerada como piel, debido a los valores establecidos y a la iluminación presente al capturar la imagen, en este caso la parte de la mano que no se tomó como color de la piel, es debido a que tiene mucho brillo.

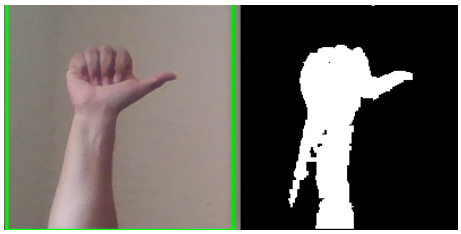


Figura 4.11 Segmentación basada en el color de la piel.

Según Shaik y colaboradores [6], se han usado una gran variedad de espacios de color, sin embargo el espacio de color RGB no es muy preferible para la detección basada en color, porque la información de color e intensidad está mezclada y no tiene características uniformes. Llegó a la conclusión que el espacio de color HSV es mejor para imágenes simples con fondo uniforme, pero YCbCr puede ser aplicado a imágenes con colores complejos y con iluminación irregular. Los valores que recomienda son los siguientes:

$$150 < Cr < 200 \tag{4.1}$$

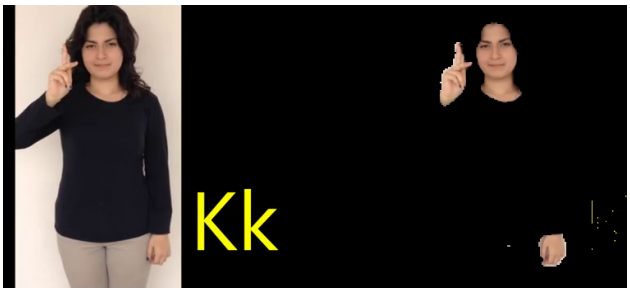
$$100 < Cb < 150 \tag{4.2}$$

En la figura 4.12, se muestran los resultados de aplicar la segmentación basada en el color de la piel usando diferentes fondos, la imagen de la derecha muestra algunos problemas que presenta el color del fondo.

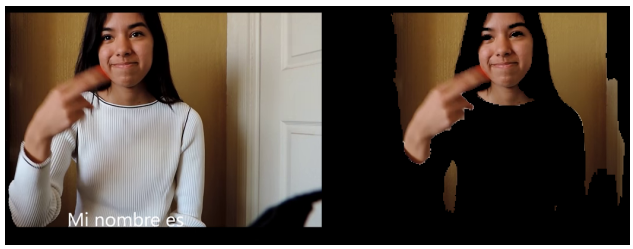
Para obtener la imagen a color sin el fondo, se multiplica la mascara obtenida, por la imagen original, obteniendo la imagen segmentada.

4.5.3. Análisis de la segmentación

Debido a lo anterior se seleccionó el método de segmentación basado en el color de la piel, puesto que es más robusto a cambios en la iluminación y que no le afecta el movimiento de la cámara o del fondo. El problema que presenta el método de segmentación basado en el color



(a) imagen con fondo diferente al color de la piel



(b) fondo parecido al color de la piel

Figura 4.12 Resultados de la segmentación por color de la piel usando diferentes fondos.

de la piel, es la selección de los valores que son reconocidos como color de la piel, para ello se pretende desarrollar un método para seleccionar de manera automática, los valores que permitan reconocer la mayor cantidad de pixeles de color de la piel y minimizar el número de pixeles del fondo. Este método que se menciona a continuación.

4.5.4. Valores adecuados para la segmentación de piel

El método de segmentación basado en el color de la piel obtiene buenos resultados, sin embargo, el paso más importante en este método es la selección del valor adecuado para los valores Y, Cb y Cr, que permitan obtener la mayor parte de la mano que no pertenezca al fondo. Para esto se propone el siguiente método de seleccionar los valores basados en imágenes de entrenamiento.

Se propone una técnica de selección automatica usando un clasificador que busque los patrones en las imagenes de ejemplo y que a partir de este clasificador, se pueda predecir que pixeles de una imagen pertenecen al color de la piel. El dataset utilizado es el 'Skin Segmentation Dataset' [19], contiene 4 valores, 3 valores correspondientes a un pixel en espacio de color RGB y un valor 1 para cuando pertenece al color de la piel y 0 cuando no pertenece. El dataset no es balanceado, debido a que contiene más ejemplos de pixeles que son color de fondo comparado a los pixeles de color de la piel. Es necesario convertir los pixeles al espacio de color YCbCr, debido a que como se mencionó antes, este espacio de color obtiene mejores resultados en la segmentación del color de la piel. También es necesario convertirlos de YCbCr a RGB para mostrar la imágenes de resultado una vez realizada la segmentación con un clasificador.

r	g	b	tipo
74	85	123	1
72	83	121	1
70	81	119	1
70	81	119	1
69	80	118	1
70	81	119	1
70	81	119	1
255	255	255	2
253	255	255	2
185	179	136	2
183	177	134	2

Tabla 4.2 Tabla que muestra ejemplos del dataset de piel.

La tabla 4.2, muestra ejemplos del dataset de piel, este contiene 245057 datos con 4 atributos. El clasificador que se propone es un árbol de decisiones, debido a que el conjunto de datos es no balanceado y binario, las dos clases son piel (valor 1) y fondo (valor 0).

El procedimiento general de entrenamiento del clasificador es el siguiente:

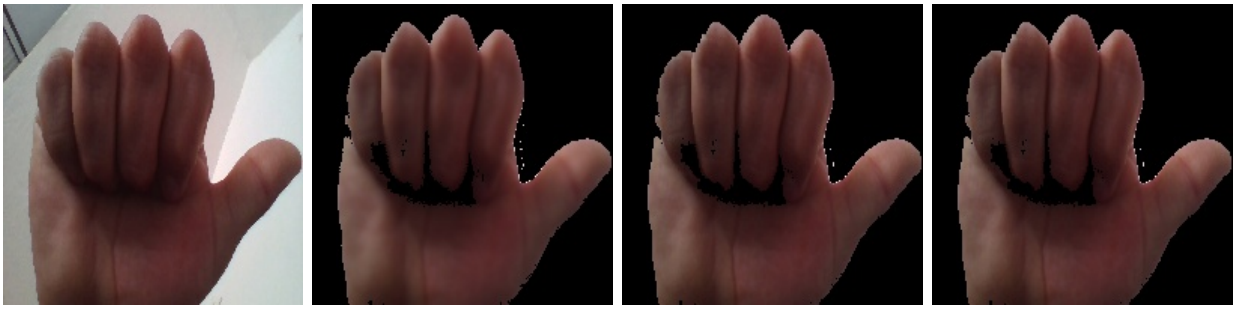
- Se abre el archivo de texto que contiene la 4 características del datatet: R, G, B, piel.
- Se convierte del espacio de color RGB al espacio YCbCr.
- En el entrenamiento del clasificador se divide R, G, B como la variable X y piel como la variable Y. Esto quiere decir que cuando se ingrese esa combinación de valores R, G, B se debe obtener el resultado especificado en Y. No es necesario hacer pre-procesamiento de los datos, debido a que los valores son enteros.
- Seleccionar los parámetros del clasificador como el criterio entropía, gini o error. La máxima profundidad del árbol, el número de ramas a dividir. Estos parámetros permiten que el clasificador tenga una mejor segmentación.
- Se debe entrenar el clasificador con todos los datos, usando los parámetros seleccionados anteriormente.
- Se procede probar el porcentaje de exactitud del algoritmo, realizando una predicción de todos lo datos. Lo más lógico es que no permita predecir todos los pixeles, debido a que algunos pixeles pertenecen al fondo y al color de la piel al mismo tiempo. Esto hace que sea imposible obtener el 100 % de exactitud, lo mejor es seleccionar los parámetros que tengan menor errores en ambas clases.
- Para cada pixel de la imagen que se quiere segmentar, se ingresa al clasificador y se obtiene el resultado de si es piel o no, estos valores se almacenan en una nueva matriz. Obteniendo una matriz con valores de 1 donde hay piel y de 0 donde hay fondo.
- Usando esta matriz, que es una imagen binaria, se puede aplicar enmascaramiento a la imagen y obtener la imagen ya segmentada
- Se transforma del espacio de color YCbCr a RGB.

Con el procedimiento anterior se puede ajustar el clasificador a las necesidades de la aplicación, personalizando la detección al color de piel requerido para el usuario o generalizar los valores del color de piel.

4.6. Selección del orden de los pasos del pre-procesamiento de la imagen

Es necesario seleccionar el orden en que se aplicarán las técnicas de pre-procesamiento de imágenes para obtener mejores resultados. Para realizar el análisis, se capturó algunas imágenes de la letra a, b, c y v. Mediante esas imágenes se analizó el resultado que se tenia al aplicar las técnicas de pre-procesamiento de las imágenes en diferente orden. Las combinaciones que se analizaron fueron las indicadas:

- Método 1:
  - Eliminar ruido,



(a) imagen original (b) método 1 (c) método 2 (d) método 3

Figura 4.13 Resultados obtenidos al aplicar las técnicas en diferente orden.

- Escalar la imagen,
- Segmentación por color de piel.

- Método 2:
  - Escalar la imagen,
  - Eliminar ruido,
  - Segmentación por color de piel.
- Método 3:
  - Eliminar ruido,
  - Segmentación por color de piel,
  - Escalar la imagen.

Se obtienen mejores resultados visualmente, cuando se elimina el ruido primero, ya que se obtiene una mejor segmentación que conserva los bordes, y también conserva una mayor cantidad de la piel. Cuando se escala la imagen y luego se elimina el ruido, la imagen se suaviza más, perdiendo los bordes y eliminando pequeñas partes de la piel comparado con el método anterior. También hace que se pierda más información de la mano. El orden de la segmentación y la escala no importa. Ya que da los mismo resultados aplicar un método antes que otro, sin embargo el orden de la reducción de ruido y la escala si afectan. Es necesario realizar el análisis de los resultados, cuando se combinan con la extracción de características y los algoritmos de clasificación.

En la figura 4.13, se muestra los resultados de aplicar las diferentes técnicas para el pre-procesamiento de imágenes en diferente orden. En 4.13b, se aplica primero la escala y luego el filtro de ruido, la imagen pierde información durante el proceso de escala y cuando se aplica el filtro de reducción de ruido, tiene menos información, por lo tanto suaviza la imagen y los bordes se ven afectados, aun cuando el algoritmo de reducción de ruido debería conservar los bordes. En 4.13c, se aplica primero el filtro de ruido y posteriormente la escala, se obtienen mejores resultados, debido a que los bordes no se pierden, también la imagen no se ve suavizada, conservando más información comparado con el método anterior. También hay pequeñas diferencias en la imagen, por ejemplo que este método hace que tenga menos

agujeros negros en la imagen, ya que ayuda a la segmentación por color de piel, en la imagen 4.13c se puede ver que en los dedos hay menos agujeros comparados con la imagen 4.13b. En 4.13d se obtiene la misma imagen que el método anterior, demostrando que el orden de la escala y la segmentación no afectan el resultado, ya que se obtiene la misma imagen. Sin embargo, es mejor aplicar primero la escala y luego la segmentación debido a que disminuye el procesamiento requerido, porque cuando se escala la imagen el tamaño es menor, entonces la segmentación se aplica a una imagen más pequeña, haciendo la segmentación más rápida que aplicarla a toda la imagen. Por lo tanto el orden de los algoritmos de procesamiento de imágenes seleccionado es:

- Eliminar ruido,
- Escalar la imagen,
- Segmentación por color de piel.

4.7. Transformación de imagen a datos

A continuación se debe transformar el conjunto de imágenes en datos para el clasificador, debido a que los clasificadores que se van a utilizar, no están preparados para aceptar imágenes directamente. Una vez que se eliminó el ruido de la imagen, se escaló la imagen y se segmentó, es necesario extraer las características de la imagen mediante alguno de los siguientes algoritmos:

- Momentos de Hu
- Momentos de Zernike
- Histogramas de Orientación del Gradiente

Estos algoritmos dan como salida, un vector con números, y dependiendo del algoritmo, el tamaño del vector puede variar, según los parámetros que usen en el algoritmo. El algoritmo de Hu siempre da como resultado un vector de longitud igual a 7, la longitud del vector de características que produce el algoritmo de Zernike depende del grado del polinomio usado, y la longitud del vector de características para el algoritmo de Histograma de Orientación del Gradiente depende del número de pixeles por celda y el número de orientaciones que sean seleccionadas. El algoritmo que da como salida mayor número de características es el Histograma de Orientación del Gradiente. El vector de datos correspondiente a cada uno de estos métodos, es el que se ingresa al clasificador.

4.8. Algoritmos de extracción de características

Una vez obtenida la parte segmentada de la imagen usando el color de la piel, se procede a extraer las características de la imagen que permitan describir el objeto, mediante los algoritmos de extracción de características. Estos algoritmos ayudan a eliminar información innecesaria, que disminuye la velocidad de procesamiento y afecta la exactitud del clasificador. Algunos algoritmos tienen ventajas como:

- Ser invariantes a rotación.
- Ser menos afectados por el ruido y la iluminación.
- Ser más exactos.
- Ser más rápidos.
- Ser menos complejos.

Tomando en consideración lo mencionado por Patil y Subbaraman [20], se analizan tres algoritmos de extracción de características que son: momentos de Hu, histograma de orientación del gradiente y momentos de Zernike.

4.8.1. Momentos de Hu

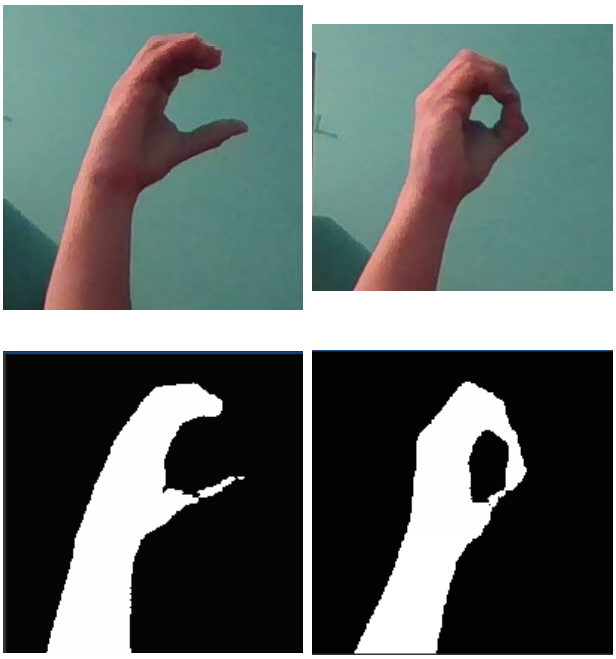
Este algoritmo es invariante a traslación, escala y rotación, esto ofrece una gran ventaja, porque aunque una imagen se encuentre girada, tendrá valores similares, haciendo el proceso de reconocimiento más exacto. Este algoritmo no requiere ningún hiperparámetro para funcionar, y siempre da como resultado un vector de características de tamaño 7. Sin embargo este algoritmo sólo utiliza información del contorno de la imagen, generando errores cuando las imágenes tienen la misma silueta. Hay algunas ocasiones en las que se confunde debido al parecido que tienen algunas imágenes en su silueta cuando son vistos desde un punto de vista diferente. En la figura 4.14, se muestra un ejemplo donde la silueta permite diferenciar un objeto de otro. En la figura 4.15, se muestra las imagenes y sus siluetas de las letras m, n, o y p capturadas desde un punto de vista frontal. Desde este punto de vista la silueta no permite diferenciar la letra m de la letra n, ni tampoco la letra o de la letra c, debido a que la imagen de la silueta, no proporciona información de la forma de los dedos, por lo tanto se espera que este algoritmo en combinación con el algoritmo de clasificación tenga más errores.

Este algoritmo se aplica a imágenes binarias, por lo tanto se va aplicar este algoritmo a la máscara binaria, que se obtiene en el paso de la segmentación por color de piel.

0
0.0009258529700307316
3.2686172476550545e-07
2.44893411537584e-10
1.0196143788898641e-10
1.6101107210287263e-20
5.793661913876255e-14
5.852791087430051e-22
r

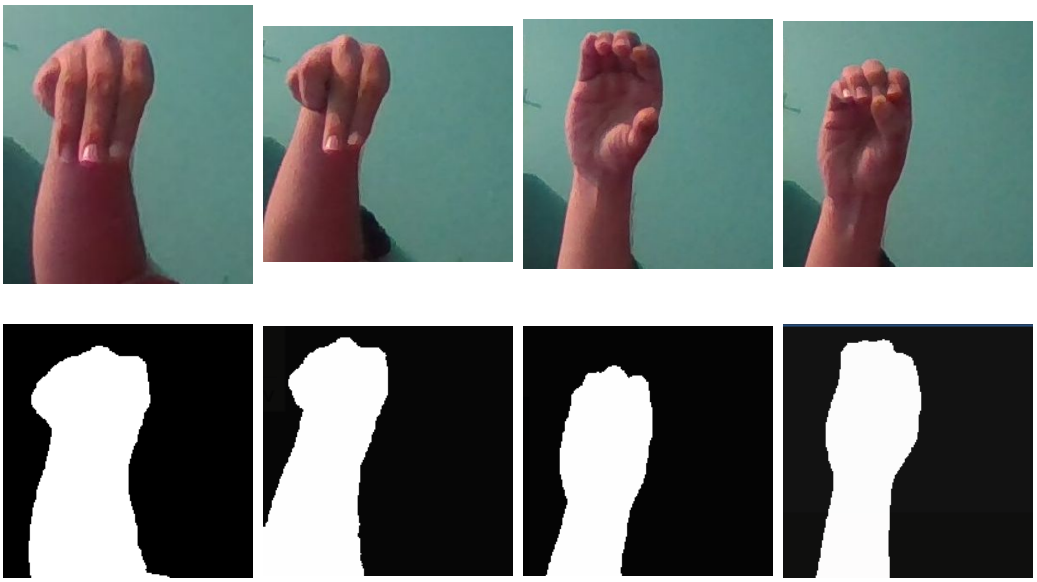
Tabla 4.3 Vector de características de los momentos de Hu.

En la Tabla 4.3, se muestra un ejemplo del vector de características del algoritmo de momentos de Hu para la letra r. El archivo CSV contiene 6300 vectores como este. El primer



(a) letra c (b) letra o

Figura 4.14 Siluetas de la letra c y o, tomadas del dataset.



(a) letra m (b) letra n (c) letra o (d) letra c

Figura 4.15 Siluetas de la letra m, n, o y c, realizadas desde otro punto de vista.

valor es un número que se crea automáticamente por la librería, por esto debe omitirse cuando se carguen los datos desde el CSV, los siguientes 7 valores son los resultados del algoritmo de momentos de Hu, y el último valor es la clase a la que pertenece, en este caso es la letra r.

4.8.2. Momentos de Zernike

Los momentos de Zernike son invariantes a rotación y pueden ser invariantes a escala y translación mediante una modificación al algoritmo. Los momentos de Zernike hacen una aproximación a la forma de la imagen, describiendo la parte interna de la mano, a diferencia de los momentos de Hu que sólo describen la silueta, esto permite tener una mejor descripción del objeto. Como menciona Sabhara y colaboradores [21], el algoritmo de Zernike tiene mayor exactitud conforme se aumenta el grado del polinomio. Debido a que los momentos de Zernike permiten adaptar el grado del polinomio a las aplicaciones, este algoritmo tiene una gran ventaja con respecto a los momentos de Hu.

Cuando se aumenta el grado del polinomio de Zernike se mejora la descripción de la imagen, pero la longitud del vector de características aumenta. Este incremento en la longitud del vector de características puede causar el problema de la maldición de la dimensionalidad, que hace que el clasificador necesite un número mayor de ejemplos para entrenamiento del clasificador, de lo contrario se tendrá malos resultados en el reconocimiento. Por lo tanto, se debe elegir el grado del polinomio que tenga la mayor exactitud, pero sin aumentar mucho el número de características. Otro parámetro que se debe configurar es el radio, se debe ajustar de manera manual, y depende del problema.

Este algoritmo trabaja sobre imágenes en escala de grises, por tal motivo se debe convertir la imagen segmentada por color de la piel, del espacio de color RGB a escala de grises y posteriormente se debe aplicar el algoritmo.

Por lo tanto los principales parámetros a configurar en el algoritmo son:

- Radio: Es el máximo radio para el polinomio de Zernike en pixeles.
- Grado: Es el máximo grado a considerar.
- Centro de masa: Por default es el centro de masa de la imagen.

En la Tabla 4.4, se muestra un ejemplo del vector de características del algoritmo de momentos de Zernike para la letra r. El archivo CSV contiene 6300 vectores como este. El primer valor es un número que se crea automáticamente por la librería, por esto debe omitirse cuando se carguen los datos desde el CSV, los siguientes 12 valores son los resultados del algoritmo de momentos de Zernike, y el ultimo valor es la clase a la que pertenece, en este caso es la letra r.

4.8.3. Histograma de Gradientes Orientados (HOG)

Este algoritmo sólo es invariante a traslación, pero tiene la ventaja que es menos afectado por el ruido y la iluminación. Tiene una alta dimensionalidad, es decir tiene un vector grande de características, dependiendo de la configuración puede tener una longitud de hasta 4000

0
0.3183098861837879
0.054575124095770346
0.3946869806264319
0.14412542947765777
0.09016317039280948
0.030949762478741014
0.07975846670337478
0.1342022746177698
0.07846004231516572
0.06816471509257606
0.013136121260190037
0.06279479393370553
r

Tabla 4.4 Vector de características de los momentos de Zernike.

características. Este algoritmo no es invariante escala, por lo tanto requiere que la imagen de entrenamiento y las de pruebas sean del mismo tamaño, debido a que una imagen de diferente tamaño, producirá un vector de diferente tamaño. Este algoritmo trabaja sobre imágenes en escala de grises, por tal motivo se debe convertir la imagen de RGB a escala de grises. Este algoritmo de extracción de características ha sido usado en la deteccion de peatones en las calles, obteniendo buenos resultados. Además según Sabhara y colaboradores [21], se ha utilizado este enfoque en el reconocimiento de lenguaje de señas. Los parámetros a configurar son:

- Pixeles por celda: Las imágenes son de tamaño $200x200$, dando como resultado 40000 pixeles, el tamaño del descriptor es mucho menor que el número de pixeles en la imagen. Este parámetro se seleccionó basado en la escala de importancia de las características para hacer la clasificación. Un valor muy pequeño debería aumentar el tamaño del vector de características, y un tamaño muy grande podría no capturar información relevante.

- Celdas por bloque: La noción de bloque existe para abordar la variación en la ilumina-ción. Un bloque grande hace los cambios locales menos significativos, mientras que un bloque pequeño añade más peso a los cambios locales. Un típico valor es colocar este valor como 2 veces el tamaño de los pixeles por celda.

- Orientaciones: Asigna el número de contenedores en el histograma de gradientes. El autor del articulo de HOG recomienda el valor de 9, para calcular gradientes entre 0 y 180 grados en incrementos de 20 grados.

El uso de un clasificador puede ayudar a este algoritmo a aprender las variaciones en la rotación y la escala, mediante la agregación de ejemplos al dataset con variaciones en la rotación y la escala.

No se va a motrar un ejemplo del vector de características del algoritmo de Histograma de Gradientes Orientados, debido a la longitud de 1024, que es demasiado para el presente documento, sin embargo, tiene el mismo formato que los vectores de momentos de Hu y momentos de Zernike, pero la diferencia es que el vector el número de características es 1022.

4.9. Algoritmos de clasificación multiclase

Los algoritmos de extracción de características permiten describir el objeto, y cuando se combinan con un clasificador se logra el reconocimiento de objetos, debido a que el clasificador encuentra patrones en las características de cada objeto, que diferencian a este de los demás objetos. Por lo tanto, es necesario probar la combinación de los algoritmos de extracción de características con los clasificadores y evaluar su exactitud, para elegir el que mejor se adapte a este problema. Un algoritmo de clasificación necesita lo siguiente para funcionar:

- Dataset: Requiere un conjunto de datos para entrenar el clasificador. Se utiliza el vector de características obtenido con el algoritmo de extracción de características descrito anteriormente.
- Pre-procesamiento: Los datos requieren ser procesados antes de ser ingresados en el clasificador, en este paso, se transforman el vector de características, a un formato que sea válido para el clasificador y que mejore la exactitud del clasificador.
- Evaluación del clasificador: Es necesario evaluar la exactitud del clasificador, entrenando el clasificador con un conjunto de datos y con este clasificador entrenado predecir un conjunto de datos diferente al de entrenamiento, con el objetivo de medir la capacidad de generalizar los datos, también para detectar problemas como el sobre-entrenamiento o la falta de ejemplos en el entrenamiento.

Se realiza un preprocesamiento debido a que un clasificador requiere que los datos cumplan lo siguiente para funcionar correctamente:

- Datos únicamente numéricos: Los datos que se ingresan al clasificador deben ser números, cualquier otro tipo de dato debe ser transformados en una representación numérica. La información de tipo texto es convertida a un formato llamado one hot encoding.
- Normalización de los datos: Los datos suelen estar en rangos muy diferentes, lo que afecta la exactitud de algunos clasificadores, por lo tanto es necesario reajustar el rango de los datos.

Lo anterior hace notar la importancia de realizar un preprocesamiento de los datos, debido a que no siempre se encuentran los datos en forma numérica, ejemplo de esto, es la categoría de cada imagen, es decir, la letra que representa cada imagen: $a, b, c, ..., z$ debe ser transformada a una representación numérica.

Es importante dividir el conjunto de datos en dos partes, una de entrenamiento y otra de prueba, con el objetivo de que el algoritmo aprenda del conjunto de entrenamiento, y se mida su exactitud con el conjunto de prueba. De esta forma se verifica que desempeño tiene

con ciertos parámetros, encontrando una mejor configuración para el problema actual. Las formas de evaluación más comunes son:

- Dividir los datos en conjunto de entrenamiento y conjunto de prueba.
- Usar el método de validación cruzada.
- Dividir los datos en conjunto de entrenamiento y conjunto de prueba; y posteriormente aplicar validación cruzada al conjunto de entrenamiento.

En el presente trabajo se utiliza el tercer método, debido a que da una mejor aproximación la exactitud real.

Para entrenar los clasificadores se utiliza el vector de características, y también se necesita una etiqueta para ese vector, indicando que letra representa ese vector y pertenece a una de las letras del alfabeto $a, b, c, ..., z$. Esto sirve para entrenar el clasificador, ya que es un requerimiento para el aprendizaje supervisado. Por lo tanto, se obtiene un vector de características representado como X con la etiqueta representada como Y. Todas las características de la imágenes deben tener su etiqueta.

4.10. Preprocesamiento de los datos

Una vez obtenidos los valores del algoritmo de extracción de características, es necesario transformar los datos a un formato válido para el clasificador. Los clasificadores a comparar son dos, máquina de soporte vectorial y una red neuronal perceptron multicapa con backpropagation, debido a que estas técnicas han tenido buenos resultados en aplicaciones similares de reconocimiento de objetos.

Los algoritmos de extracción de características dan como resultado un vector con números decimales, por lo tanto, no es necesario transformar estos valores. Sin embargo, es necesario transformar la etiqueta o clase a la que pertenece el vector de las características, a un tipo de dato numérico, debido a que es de tipo categórico. Debido a que son 21 letras del alfabeto, cuando se transforme el dato categórico al formato one hot encoding, se va a tener un vector de longitud 21. Por lo tanto para la letra a se tiene el siguiente vector codificado: $[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$, para la letra b se tiene el siguiente vector codificado: $[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$ y así sucesivamente para todas la etiquetas del alfabeto.

Por lo tanto X y Y del algoritmo de momentos de Hu, ya codificados podría quedar así:
$X = [0,0,0,0,0,0,0]$
$Y = [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$

También es necesario reajustar los valores mediante el escalamiento de los valores, para evitar que los valores más grandes dominen sobre los valores menores. Esto mejora la exactitud de los algoritmos de clasificación.

Hay dos enfoque comunes para traer las diferentes características en la misma escala:

- Normalización

- Estandarización

La normalización se refiere a re-escalar las características a un rango de [0, 1]. Para normalizar los datos, se aplica el algoritmo de min-max scaling a cada característica del vector, donde el valor nuevo del vector $x(i)$ es calculado como se indica:

$$x(i) = \frac{x(i) - x_{min}}{x_{max} - x_{min}} \tag{4.3}$$

donde:

- $x(i)$ es un ejemplo en particular.
- x_{min} es el valor más pequeño en la columna de características.
- x_{max} es el valor más grande en la columna de características.

Aunque la normalización sea una técnica muy útil cuando se requieren tener los valores en un determinado rango, la estandarización es más práctica en muchos algoritmos de aprendizaje máquina. La razón es que muchos modelos como la máquina de soporte vectorial, inician los pesos a cero o a un valor aleatorio cercano a cero. Mediante la estandarización se centra la columna de características a un promedio de cero y una desviación estándar de 1, así que la columna de características toma la forma de una distribución normal, lo que hace mas fácil aprender los pesos. De ahí que la estandarización mantiene información útil sobre valores atípicos y hace el algoritmo menos sensible a ellos, en comparación con el algoritmo min-max scaling. En el presente trabajo se utiliza la estandarización para re-escalar las características.

El proceso de estandarización puede ser expresado con la siguiente ecuación:

$$x(i)_{std} = \frac{x(i) - \mu_x}{\sigma_x} \tag{4.4}$$

donde:

- $x(i)$ es un ejemplo en particular.
- μ_x es el promedio de una columna de características.
- σ_x es la desviación estándar de la columna de características.

En la Tabla 4.5, se muestra la diferencia entre los dos enfoque en un conjunto de números, se puede notar que en la normalización los valores son todos positivos, mientras que en la estandarización se incluyen valores negativos y están centrados a cero:

entrada	estandarización	normalización
0.0	-1.336306	0.0
1.0	-0.801784	0.2
2.0	-0.267261	0.4
3.0	0.267261	0.6
4.0	0.801784	0.8
5.0	1.336306	1.0

Tabla 4.5 Diferencia entre normalización y estandarización

4.11. Datos de entrenamiento y prueba

Dividir los datos en entrenamiento y prueba permite saber cuando hay sobre-entrenamiento, debido a que si un clasificador tiene gran exactitud en los datos de entrenamiento, pero tiene poca exactitud en los valores de prueba, se dice que esta sobre-entrenado. Existen técnicas para evitar este problema, dependiendo del algoritmo que se use.

Los datos obtenidos del algoritmo de extracción de características son divididos en una parte para entrenar el algoritmo y otra parte para probar el desempeño del clasificador, midiendo el porcentaje de errores, al predecir las etiquetas de los datos de pruebas. La parte de entrenamiento se evaluá usando validación cruzada, porque es uno de los métodos que dan una mejor evaluación del clasificador, usando k = 4, debido al tamaño del dataset. Por lo tanto primero se dividen los datos en 70 % para entrenamiento y 30 % para prueba, este 70 % de entrenamiento posteriormente es dividido en k particiones y evaluado con validación cruzada. Se puede aplicar validación cruzada para la evaluación del algoritmo debido a que las técnicas usadas no toman demasiado tiempo de procesamiento. De lo contrario tardaría mucho tiempo en realizar la evaluación.

4.11.1. Preprocesamiento de datos para los clasificadores

El algoritmo de extracción de características, produce como resultado, un vector de números, por lo tanto no se requiere transformar los datos a formato numérico. Sin embargo, los datos suelen estar en rangos muy diferentes. Se utilizará el algoritmo de estandarización, para ajustar el rango de los datos, esto mejora la exactitud del clasificador.

Lo primero que se realiza, es entrenar el algoritmo con los datos de entrenamiento, a continuación, se transforman los datos de entrenamiento con el algoritmo entrenado, y por último se transforma el conjunto de prueba, con el algoritmo. Es importante sólo entrenar el algoritmo con el conjunto de entrenamiento, y transformar ambos conjuntos con el algoritmo.

Hasta ahora se tiene preprocesados los valores de las características extraídas de las imágenes, esto es representado como X. Pero aun falta procesar la etiqueta a la que pertenece la imagen, ejemplo: "a", "b", "c", etc. Estos valores son de tipo texto, también conocidos como categóricos. Sin embargo el clasificador sólo acepta valores numéricos, entonces se debe transformar los datos. Lo que se realiza es transformar cada texto a un número entero. Quedando de la siguiente manera: "a" se representa como 0, "b" se representa como 1, y así sucesivamente. Sin embargo, esta forma agrega información extra al algoritmo de clasifica-

ción, debido a que debería ser interpretado, como si estuvieran ordenadas las categorías. La solución es utilizar one hot encoding, para transformar los datos categóricos a numéricos. quedando la letra "a" como $[1, 0, 0, 0, 0, ..., n]$ donde n es el número de letras del alfabeto. y "b" se representa como $[0, 1, 0, 0, 0, ..., n]$ hasta "z", que se representa como $[0, 0, 0, 0, 0, ..., 1]$.

4.12. Configuración de la red neuronal

La entrada de la red neuronal depende del algoritmo de extracción de características, en el caso de los momentos de Hu, la red neuronal tiene como entrada 7 características, y para los otros algoritmos dependerá de la configuración que tengan, porque la salida depende de los parámetros de configuración del algoritmo de extracción de características. La capa de neuronas de salida de la red neuronal para todos los algoritmos será de 21 neuronas, ya que son 21 tipos de señas, que corresponden a las diferentes letras del alfabeto. El número de neuronas en las capas internas deberá ser ajustada durante el entrenamiento, también sus hiperparámetros:

- Función de activación: Permite a la red la capacidad de procesar datos que no sean lineales.
- Función de inicialización: Permite encontrar los pesos correctos de la neuronas de una forma más rápida.
- Número de épocas.
- Razón de aprendizaje.
- Tamaño de lote.
- Función de aprendizaje.

Para evitar el sobre-entrenamiento de la red se utiliza Dropout, que consiste en bloquear algunas neuronas, para permitir que las otras neuronas aprendan patrones independientes. La función de activación de la última capa de las neuronas debe tener la función de activación softmax, debido a la clasificación del tipo multiclase. La función de activación de las demás neuronas es RELU, que tiene ventajas sobre las otras funciones. La función de optimización que se utiliza es Adam, utilizando decay = learning rate/número de épocas, permitiendo a la red converger más rápido, si se ajusta de la manera correcta. La métrica que se utilizan para evaluar el clasificador es: exactitud (acuraccy).

Estos parámetros permiten ajustar la red al problema, la selección de estos parámetros, permite que la red tenga mayor exactitud para clasificar las letras.

Cuando se agregan más neuronas o se agregan más capas se debe reajustar los parámetros e incrementar la razón de aprendizaje. El número de capas y neuronas se obtiene mediante experimentación. Se realizó la de la siguiente manera:

- Agregar neuronas a la red cuando se tiene una sola capa. Es conocido como agregar neuronas a lo ancho.

- Varias capas con una neurona. Es conocido como agregar neuronas en profundidad.
- Hacer combinaciones de las anteriores, es decir agregar varias capas con variaciones en el número de neuronas.

Mayor número de capas permite mejor oportunidad de recomposición jerárquica de características abstractas aprendidas de los datos. Sin embargo, requieren más entrenamiento, en el número de épocas y la razón de aprendizaje.

4.13. Configuración de la Máquina de Soporte Vectorial

Los parámetros que se deben configurar son los siguientes:

- C es el parámetro de penalización del error.
- Kernel.
- Gamma es un parámetro que se agrega cuando utiliza un kernel de tipo RBF.

Se utilizó un kernel de función de base radial RBF que obtiene mejores resultados que un kernel lineal. Este clasificador es más fácil de configurar que una red neuronal. Se utiliza el enfoque uno contra el resto para realizar la clasificación multiclase, debido a que este clasificador esta diseñado para la clasificación binaria.

4.14. Selección de la configuración de los algoritmos

Los mejores parámetros se buscan por medio de grid search, que consisten en buscar las combinaciones de los diferentes parámetros, y mediante la técnica de validación cruzada, seleccionar la mejor combinación de parámetros. Este método permite la paralelización debido a que las variables son independientes. La paralelización permite dividir las tareas para que pueden ser ejecutadas al mismo tiempo en diferentes equipos, esto permite reducir el tiempo de las operaciones. Sin embargo, es difícil probar todas las posibles combinaciones, debido a que lleva mucho tiempo. Por esto se va a establecer los parámetros de los algoritmos de extracción de carácteristicas que se van a utilizar en conjunto con los algoritmos de reducción de dimensionalidad y los algoritmos de clasificación, para disminuir el espacio de busqueda.

4.14.1. Configuración de algoritmos de extracción de carácteristicas.

El algoritmo de Hu no requiere configuración, así que no es necesaria una selección de los mejores parámetros.

La mejor configuración del algoritmo de extracción de características Zernike para el dataset es: Zernike grado 5 radio 80, debido a que da los mejores resultados sin aumentar mucho el número de características, el algoritmo da como resultado un vector con 13 características.

La mejor configuración del algoritmo de histograma de Orientación del gradiente para el dataset es: orientaciones = 8, pixeles por celda = (16,16), celdas por bloque = (1,1). Da como resultado un vector de salida de 1153 características.

4.14.2. Configuración de la máquina de soporte vectorial.

Los parametros a buscar mediante grid search para el clasificador SVM, que son C y gamma. Se buscan de la siguiente manera:

Primero se busca el número de componentes para PCA o para LDA. El máximo valor a tomar es el largo del vector de características. El vector de características de los momentos de Zernike es de tamaño 12, por lo tanto, se buscan algunos valores menores a 12, porque se busca reducir el número de características, no se toman todos por cuestión de tiempo de procesamiento, pero se comparan algunos para verificar si un menor número de componentes obtiene mejor exactitud. Se consideraron los siguientes valores en los momentos de Zernike: $[2, 5, 10, 12]$, para los momentos de Hu se toman los valores: $[4, 5, 6, 7]$, porque el número de características es 7.

El valor de gamma y C se buscan en pequeños incrementos, para encontrar la tendencia a los valores que producen mejores resultados. Los valores de gamma en la primer busqueda se consideran los siguientes: $[0{,}01, 0{,}1, 1, 10, 100, 1000, 1000]$ y los valores de C: $[0{,}1, 1, 10, 100, 1000]$

Despues de obtener los mejores parámetros en la primera prueba, se procede a buscar valores cercanos a los mejores encontrados, es decir si el mejor parametro encontrado para el valor de gamma fue 0.1, la siguiente prueba se realiza con valores de gamma cercanos a este: $[0{,}5, 0{,}1, 0{,}2]$. De la misma manera se procede con el valor de C.

4.14.3. Configuración de la red neuronal.

Para buscar los valores de los parámetros en la red neuronal se procede a definir que la red neuronal siempre tiene el número de neuronas de entrada igual al tamaño del vector de características obtenido del algoritmo de extracción de características o en el caso de que se haya aplicado un algoritmo de reducción de dimensionalidad, se debe crear la capa de neuronas igual al tamaño de la salida del algoritmo de reducción de dimensionalidad.

Así mismo el número de capas de salida es igual al número de letras a reconocer y esto define que es la misma cantidad para todas la configuraciones. Lo que se debe configurar a medida del problema es el número de capas de neuronas ocultas, así mismo también establecer la cantidad de neuronas en cada capa oculta, existen varias formas de configurar la red, pero las que dan mejores resultados es la configuración escalonada, es decir poner 5 neuronas en la primera capa, 4 en la segunda y así sucesivamente, por lo tanto se va a utilizar esta configuración en la busqueda de los parámetros. También se va a configurar la función de activación como RELU para todas la neuronas ocultas y la inicialización de tipo uniforme.

4.15. Resultados

Para la implementación de las configuraciones mencionadas en la sección anterior se utilizó Python 2.7, para tener la compatibilidad con mayor número de librerias, con la distribución Anaconda para facilitar la instalación de OPENCV. Se utilizó las librerías:

- Pandas: Para analisis de datos, guardar y leer archivos CSV.
- Numpy: Para operaciones con arreglos.
- OpenCV: Procesamiento de imagenes.
- Scikit-learn: Para preprocesamiento y clasificación de los datos con SVM.
- Mahotas: Para extraer características con el algoritmo de momentos de Zernike.
- Scikit-image: Para extraer características con el algoritmo de momentos de HOG.
- Keras: Para preprocesamiento y clasificación de los datos con la red neuronal.
- Matplotlib: Para visualizar las imagenes.
- Pickle: Para guardar los clasificadores despues del entrenamiento.

El equipo de cómputo utilizado fue un ordenador portátil con un procesador Intel core i5, 8gb de memoria RAM con sistema operativo macOs Mojave 10.14.

Se obtuvo como resultado un dataset de las imágenes del alfabeto de señas estáticas, se encuentran organizadas en carpetas con el nombre de la letra del alfabeto. También se obtuvo tres archivos en formato CSV, uno por cada algoritmo de extracción de características, los datos son obtenidos de las imágenes despues de ser preprocesadas y posteriormentes se obtienen los vectores de características extraidas con los tres algoritmos: momentos de Hu, momentos de Zernike y Histograma de gradientes orientados.

También se obtuvo el mejor orden de los pasos en el preprocesamiento de las imágenes ,para optimizar el tiempo de procesamiento y obtener los mejores resultados. Los pasos que se analizaron fueron: reducción de ruido, escalamiento de la imagen y segmentación. El mejor orden de los paso del preprocesamiento de las imágenes es: Eliminar ruido, luego escalar la imagen y por último segmentar por color de piel.

Se realizó una prueba con validación cruzada con k = 4. Los resultados de todas las configuraciones se muestran en la Tabla 4.6, en esta se resumen las exactitudes obtenidas. El algoritmos de extracción de características influye mucho en los resultados obtenidos del clasificador. El algoritmo de Hu obtiene la menor exactitud, aún así el entrenamiento es más rápido, debido a que el vector de características es menor. El algoritmo de HOG obtiene mejores resultados que el algoritmo de Hu, sin embargo, el entrenamiento y el cálculo de las características son procedimientos muy lentos, esto se debe a que el número de características es mayor que los otros algoritmos de extracción de características. El algoritmo de Zernike obtiene una mejor exactitud, comparado a HOG y Hu, también tiene menor número de características, y tarda menos tiempo de entrenamiento que HOG. Los algoritmos de reducción de dimensionalidad ayudan al algoritmo en algunos casos, pero en otros disminuyen la exactitud, en algunas situaciones se obtiene mejores resultados con PCA y en otras situaciones se

obtiene mejores resultados con LDA, por este motivo es mejor evaluar la combinación completa y no sólo al clasificador. El algoritmo de clasificación SVM y la red neuronal obtienen resultados parecidos en exactitud de la clasificación, sin embargo, la red neuronal requiere un configuración mayor de sus párametros y mayor tiempo de entrenamiento, y sólo para obtener una exactitud parecida.

La búsqueda de los parámetros se realizó mediante gridseach, que consiste en probar combinaciones de valores y entrenar la red, y mediante validación cruzada se evalua los resultados de la red, esto se realiza a todas la combinaciones que se configuren. Para validación cruzada se utilizó el número de particiones es 4.

Extracción de características	Preprocesamiento de datos	Reducción de dimensionalidad	Clasificación	Exactitud
Zernike	std scaler		SVM	97.17 %
Zernike	std scaler	PCA	SVM	96.69 %
Zernike	std scaler	LDA	SVM	98.34 %
HU	std scaler		SVM	93.11 %
HU	std scaler	PCA	SVM	86.85 %
HU	std scaler	LDA	SVM	92.56 %
HOG	std scaler		SVM	94.22 %
HOG	std scaler	PCA	SVM	97.44 %
HOG	std scaler	LDA	SVM	92.88 %
Zernike	std scaler		NN	96.78 %
Zernike	std scaler	PCA	NN	80.93 %
Zernike	std scaler	LDA	NN	96.89 %
HU	std scaler		NN	93.06 %
HU	std scaler	PCA	NN	91.06 %
HU	std scaler	LDA	NN	92.95 %
HOG	std scaler		NN	93.31 %
HOG	std scaler	PCA	NN	81.72 %
HOG	std scaler	LDA	NN	88.59 %

Tabla 4.6 Resultados de la exactitud de los algoritmos evaluados con validación cruzada con $k = 4$.

Se realizó otra prueba con validación cruzada con k = 10. Como se esperaba la exactitud mejoró, sin embargo la mejor combinación de algoritmos fue la misma que en la tabla anterior. Los resultados de todas las configuraciones se muestran en la Tabla 4.8, en esta se resumen las exactitudes obtenidas.

El proceso de estracción de características para los diferentes algoritmos es el siguiente:

- Momentos de Hu: 68.64 segundos en crear dataset.
- Histograma de orientación del gradiente: 1319.21 segundos en crear dataset.
- Momentos de Zernike: 134.29 segundos en crear dataset.

Extracción de características	Preprocesamiento de datos	Reducción de dimensionalidad	Clasificación	Exactitud
Zernike	std scaler		SVM	98.16 %
Zernike	std scaler	PCA	SVM	98.37 %
Zernike	std scaler	LDA	SVM	98.96 %
HU	std scaler		SVM	93.81 %
HU	std scaler	PCA	SVM	86.98 %
HU	std scaler	LDA	SVM	93.24 %
HOG	std scaler		SVM	95.24 %
HOG	std scaler	PCA	SVM	98.41 %
HOG	std scaler	LDA	SVM	93.58 %
Zernike	std scaler		NN	97.35 %
Zernike	std scaler	PCA	NN	90.34 %
Zernike	std scaler	LDA	NN	97.57 %
HU	std scaler		NN	93.17 %
HU	std scaler	PCA	NN	91.84 %
HU	std scaler	LDA	NN	92.99 %
HOG	std scaler		NN	94.58 %
HOG	std scaler	PCA	NN	82.58 %
HOG	std scaler	LDA	NN	90.50 %

Tabla 4.7 Resultados de la exactitud de los algoritmos evaluados con validación cruzada con $k = 10$.

La mejor combinación considerando la exactitud, la velocidad y la cantidad de datos en memoria, es la siguiente:

- Creación del dataset de las 21 señas con 300 imagenes cada una.
- Eliminar el ruido de la imagen con el filtro bilateral con la configuración: $\sigma = 130$ y $d = 5$.
- Escalar la imagen a un tamaño de 200×200 pixeles.
- Segmentar la imagen por color de piel mediante el espacio de color YCbCr, para eliminar el fondo de la imagen mediante los umbrales: $150 < Cr < 200$ y $100 < Cb < 150$.
- Transformar la imagen a un vector de características mediante el algoritmo de los momentos de Zernike con la configuración: $radio = 80$ y $grado = 5$.
- Aplicar el algoritmo de reducción de dimencionalidad LDA con los primeros 11 componentes, para reducir el número de características en el vector obtenido en el paso anterior, pero conserva la información que permite diferenciar las señas.
- Preprocesamiento del vector de características antes de ser ingresado al clasificador:

Extracción de características	Preprocesamiento de datos	Reducción de dimensionalidad	Clasificación	Tiempo de entrenamiento
Zernike	std scaler		SVM	0.19 segundos
Zernike	std scaler	PCA	SVM	0.20 segundos
Zernike	std scaler	LDA	SVM	0.32 segundos
HU	std scaler		SVM	0.37 segundos
HU	std scaler	PCA	SVM	0.44 segundos
HU	std scaler	LDA	SVM	0.23 segundos
HOG	std scaler		SVM	32.92 segundos
HOG	std scaler	PCA	SVM	0.95 segundos
HOG	std scaler	LDA	SVM	1.92 segundos
Zernike	std scaler		NN	65.69 segundos
Zernike	std scaler	PCA	NN	66.60 segundos
Zernike	std scaler	LDA	NN	55.26 segundos
HU	std scaler		NN	47.90 segundos
HU	std scaler	PCA	NN	46.69 segundos
HU	std scaler	LDA	NN	48.50 segundos
HOG	std scaler		NN	89.79 segundos
HOG	std scaler	PCA	NN	118.69 segundos
HOG	std scaler	LDA	NN	50.15 segundos

Tabla 4.8 Resultados del tiempo de entrenamiento de los algoritmos.

  - Codificación del nombre de la clase a la que pertenece el vector de características mediante el algoritmo one hot encoding.
  - Estandarización del vector de características.

- Buscar con el algoritmo de Grid search los mejores parámetros para el clasificador mediante el metodo de evaluación: validación cruzada.
- Dividir los datos en entrenamiento y prueba. Entrenar el clasificador SVM con la parte de los datos de entrenamiento usando los parámetros: $gamma = 0{,}05$ y $C = 19$.. Y predecir con el clasificador entrenado la parte del conjunto de datos de prueba.
- Evaluar el desempeño del clasificador.

4.15.1. Análisis de resultados

Analizando los resultados del clasificador en conjunto con los diferentes algoritmos de clasificación y los algoritmos de reducción de dimensionalidad, se obtiene que el clasificador SVM obtiene mejores resultados que la red neuronal, incluso en un tiempo menor de entrenamiento. La red neuronal requiere una mayor cantidad de parámetros a configurar para una mayor exactitud en el problema específico, por lo tanto requiere una mayor cantidad

de combinaciones a probar, esto produce que la cantidad de tiempo requerido para entrenar una red neuronal sea mayor que el tiempo que requiere el entrenamiento de una máquina de soporte vectorial. La red neuronal es un clasificador más dificíl de configurar debido a su mayor cantidad de parámetros en cada neurona, esto aumenta el tiempo de entrenamiento muchas veces más, por lo tanto la máquina de soporte vectorial es más conveniente para el objetivo del problema.

La combinación del algoritmo de reducción de dimensionalidad LDA junto con el algoritmo de extracción de características de los momentos de Zernike, y el clasificador SVM obtuvo la mejor exactitud. Otra combinación buena es el algoritmo de extracción de características HOG junto al algoritmo de reducción de dimensionalidad PCA y el algoritmo de clasificación SVM, obtuvieron una buena exactitud, pero a un costo de tiempo de entrenamiento y memoria mayor a las demás combinaciones de algoritmos. Por esta razón no se seleccionó esta combinación como la mejor para este problema, por la limitación de esta arquitectura para ser utilizada por dispositivos con poca memoria o capacidad de procesamiento, sin embargo, la combinación de momentos de Zernike con LDA y SVM permite tener una buena exactitud en un menor tiempo de entrenamiento, con menor memoria, debido a su vector de características de longitud 12 comparado con la longitud del vector de características de HOG de longitud 1153 (por cada imagen). Esto permite que se pueda usar en aplicaciones casi en tiempo real.

Capítulo 5

Conclusión y trabajos futuros

5.1. Conclusión y discusión

En esta investigación se propuso un sistema de reconocimiento de las señas estáticas del LSM basado en imágenes a color. Se generó una base de conocimiento (dataset) el cual consiste de 21 señas con 300 imagenes cada una, dando un total de 6300 imáges en las cuales están representadas las señas estáticas.

Se presentaron dos clasificadores para las señas: redes neuronales y SVM junto con una base de datos que consta de 6300 imágenes en las cuales se consideraron variaciones de escala, rotación y traslación.

Se consideraron varios conjuntos de características obtenidas mediante los siguientes algoritmos:

- Momentos de Hu.
- Momentos de Zernike.
- Histogramas de orientación del gradiente.

Se consideraron dos algoritmos de reducción de dimensionalidad:

- Análisis discriminante lineal (LDA).
- Analisis de componentes principales (PCA).

La tasa de reconocimiento para las imágenes estáticas en las condiciones descritas exhibieron una exactitud del 98.7 %. Para lograr estos resultados, en primer lugar se toma la imagen de la seña, se elimina el ruido con el filtro bilateral con la configuración: $\sigma = 130$ y $d = 5$, se escala la imagen a 200×200 y se aplica la segmentación por color de piel en el espacio de color YCbCr mediante los umbrales: $150 < Cr < 200$ y $100 < Cb < 150$, es necesario aplicarlos en este orden por cuestion de eficiencia en el tiempo de procesamiento, como se mencionó anteriormente en el documento. A continuación se aplica el algoritmo de extracción de características de momentos de Zernike con la configuración: $radio = 80$ y $grado = 5$, dando como resultado un vector de características de longitud 12. Después se le aplica reducción de dimensionalidad con el algoritmo LDA con los primeros 11 componentes.

Es necesario aplicar el algoritmo de estandarización al vector de características y el algoritmo de one hot encoding al nombre de la clase. Por último se entrena el clasificador SVM con el vector de características obtenido, con la configuración de los parametros: $gamma = 0{,}05$ y $C = 19$.

Los pasos para resolver la detección del alfabeto de lenguaje de señas son dependientes uno del otro. Por lo tanto si un paso no se realiza correctamente puede afectar el rendimiento de los demás pasos. La segmentación basada en el color resulta muy buena, ya que remueve gran parte del fondo de una manera muy eficiente, sin embargo, cuando en el fondo se encuentran objetos de color parecidos a la piel, surgen problemas, para solucionar esto, se debe ajustar el rango de valor para la detección del color o usar algún manera de diferenciar el fondo. En el presente trabajo se menciona un método para encontrar los valores adecuados mediante un clasificador, este método se puede utilizar para ajustar los valores a determinado color de piel, dependiendo la persona que haga uso del sistema, pudiendo aumentar la exactitud ante la variación de la iluminación.

5.2. Trabajo futuro

El presente trabajo sirve como base para las siguientes actividades:

- Reconocer las letras del abecedario que son dinámicas.
- Implementar seguimiento de la mano, y reconocer signos dinámicos.
- Reconocer los gestos de la cara.
- Reconocer movimientos del cuerpo.
- Reconocer movimientos involuntarios del usuario.
- Implementar la traducción de voz a lenguaje de señas.
- Implementar mensajes de error al no reconocer la seña, pero en lenguaje de señas, para que pueda entenderlo el sordo-mudo.

Anexo

Anexo A

Archivos de datos del dataset

El presente trabajo ha dado como resultado un dataset del alfabeto del lenguaje de señas mexicano de las señas estáticas. El dataset de imágenes se incluye en el disco, debido a la gran cantidad de información, las imágenes se encuentran organizadas en carpetas con el nombre de la letra del alfabeto. También se incluyen tres archivos en formato CSV con los vectores de características despues del preprocesamiento de las imágenes y la extracción de características. El primer valor del archivo CSV es un índice creado por la librería y debe ser ignorado al cargar el dataset.

Bibliografía

[1] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[2] INEGI, *La discapacidad en México, datos al 2014*, (accesado Enero 10, 2019). `http://internet.contenidos.inegi.org.mx/contenidos/productos/prod_serv/contenidos/espanol/bvinegi/productos/nueva_estruc/702825090203.pdf`.

[3] Escuelaparasordos, *Escuela virtual para sordos*, (accesado Febrero 21, 2018). `http://www.escuelaparasordos.com/lsm.php`.

[4] S. Mallic, *Image Recognition and Object Detection : Part 1*, 2018 (accessed Marzo 21, 2018). `https://www.learnopencv.com/image-recognition-and-object-detection-part1/`.

[5] S. Paris, P. Kornprobst, J. Tumblin, F. Durand, *et al.*, "Bilateral filtering: Theory and applications," *Foundations and Trends® in Computer Graphics and Vision*, vol. 4, no. 1, pp. 1–73, 2009.

[6] K. B. Shaik, P. Ganesan, V. Kalist, B. Sathish, and J. M. M. Jenitha, "Comparative study of skin color detection and segmentation in hsv and ycbcr color space," *Procedia Computer Science*, vol. 57, pp. 41–48, 2015.

[7] G. Kumar and P. K. Bhatia, "A detailed review of feature extraction in image processing systems," *Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on*, pp. 5–12, 2014.

[8] A. Khotanzad and Y. H. Hong, "Invariant image recognition by zernike moments," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 5, pp. 489–497, 1990.

[9] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi, "Pedestrian detection using infrared images and histograms of oriented gradients," in *Intelligent Vehicles Symposium, 2006 IEEE*, pp. 206–212, IEEE, 2006.

[10] R. Galicia, O. Carranza, E. Jiménez, and G. Rivera, "Mexican sign language recognition using movement sensor," in *Industrial Electronics (ISIE), 2015 IEEE 24th International Symposium on*, pp. 573–578, IEEE, 2015.

[11] F. Solís, D. Martínez, and O. Espinoza, "Automatic mexican sign language recognition using normalized moments and artificial neural networks," *Engineering*, vol. 8, no. 10, p. 733, 2016.

[12] C. O. Sosa-Jiménez, H. V. Ríos-Figueroa, E. J. Rechy-Ramírez, A. Marin-Hernandez, and A. L. S. González-Cosío, "Real-time mexican sign language recognition," in *Power, Electronics and Computing (ROPEC), 2017 IEEE International Autumn Meeting on*, pp. 1–6, IEEE, 2017.

[13] L. A. F. Montaño and R. M. Rodríguez-Aguilar, "Automatic translation system from mexican sign language to text," *Advances in Computational Linguistics*, p. 53, 2011.

[14] M. K. Ahuja and A. Singh, "Hand gesture recognition using pca," *International Journal of Computer Science Engineering and Technology (IJCSET)*, vol. 5, no. 7, pp. 267–27, 2015.

[15] D. Vishwakarma, K. Singh, *et al.*, "A framework for recognition of hand gesture in static postures," in *Computing, Communication and Automation (ICCCA), 2016 International Conference on*, pp. 294–298, IEEE, 2016.

[16] S. Jain and K. V. S. Raja, "Indian sign language gesture recognition," 2015.

[17] N. Soontranon, S. Aramvith, and T. H. Chalidabhongse, "Improved face and hand tracking for sign language recognition," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 2, pp. 141–146, IEEE, 2005.

[18] E. Ibarra, *Generación de dataset para problema de visión computarizada*, (accesado Marzo 31, 2019). `https://medium.com/inteligencia-artificial-itesm-cq/generaci%C3%B3n-de-dataset-para-problema-de-visi%C3%B3n-computarizada-a90c77a0dc9a`.

[19] R. Bhatt and A. Dhall, "Skin segmentation dataset," *UCI Machine Learning Repository*, 2010.

[20] A. Patil and S. Subbaraman, "A review on vision based hand gesture recognition approach using support vector machines," *Journal of Electronicsl and Communication Engineering*, 2012.

[21] R. K. Sabhara, C.-P. Lee, and K.-M. Lim, "Comparative study of hu moments and zernike moments in object recognition," *SmartCR*, vol. 3, no. 3, pp. 166–173, 2013.