UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE COMPUTACIÓN GRÁFICA

Verificación de Firmas offline utilizando Morfología Matemática con K-Vecinos y Distancia Euclídea con Centros Geométricos

Trabajo Especial de Grado presentado ante la Ilustre
Universidad Central de Venezuela

Por la Bachiller
Marjorie E. Figueroa D

Para optar al título de
Licenciada en Computación

Tutor: Prof. Rhadamés Carmona S.

Caracas, 14 de Mayo de 2015

Les dedico este trabajo a mi padre y a mi madre
quienes desde pequeña han sido mis maestros y héroes.
Y también a todos los que han estado presentes
y han creído en mí durante todos estos años.

Agradecimientos

En primer lugar le agradezco a mi padre quien ha sido una constante en mi vida, quién me ha cuidado y se ha sacrificado para que yo pudiera estudiar y ser una mejor persona. Gracias papá por madrugar conmigo y esperarme tantas veces. Gracias a mi madre quien me ha apoyado en todo lo que he querido hacer y ha confiado en mí.

A mis padrinos, quienes me cuidaron desde pequeña y aún siguen ahí apoyándome y queriéndome.

A mi novio Jorge, quién ha sido mi soporte en los momentos en que he estado a punto de caer. Quién se ha asegurado de que no me rinda y me ha dado la fuerza que necesitaba en los momentos cruciales.

A mis amigos, Gabriela, Odette, Johana, Grindeliz, María Francis, Alexis, Juan Andrés, Alejandro por estar conmigo cuando más los he necesitado. Gracias por cuidarme y cargar mi bolso tantas veces. Ustedes son los mejores amigos que podría desear.

A mis profesores, quienes me guiaron en el camino de convertirme en una profesional y me dieron las herramientas para obtener el conocimiento y la experiencia que tengo ahora.

A mi tutor Rhádames Carmona, quien me guio durante este proceso y me brindó su conocimiento. A la Profesora Haydemar Núñez por su apoyo en este trabajo.

A mi doctora, María Gollo quién se aseguró de acomodar mi espalda para que yo pudiera seguir mi vida de la manera más normal posible.

A la vida, por darme todas las experiencias buenas y malas, las cuales me han moldeado y me han hecho ser más fuerte, y por poner en mi camino personas tan especiales.

Gracias a todos y cada uno de ustedes por darme su apoyo, su amistad y su cariño. Gracias por ayudar a construir la persona que soy ahora.

Resumen

El uso de firmas para identificar a una persona es una práctica común desde hace muchos años, por lo que hoy en día son aceptadas en distintas situaciones como, transacciones bancarias, validación de documentos, entre otras. Existen distintas maneras de autenticar a una persona entre ellas están, las huellas digitales, el iris, la retina, voz, etc. Cada una de estas, tiene un nivel de fiabilidad, facilidad de uso, aceptación y estabilidad. Las firmas son sumamente fáciles de usar, y son aceptadas en la mayoría de las situaciones, pero son poco estables dado que estas cambian dependiendo de muchos factores.

Durante muchos años se han realizado investigaciones para la identificación de firmas falsificadas. Existe una gran cantidad de técnicas que ayudan a solventar este problema. En este trabajo se implementan dos enfoques distintos para la verificación de firmas a partir de imágenes de la misma. Uno de ellos está basado en operaciones morfológicas y algoritmos de minería de datos. Este, aplica 4 niveles de dilataciones en las imágenes para obtener un patrón. Dado que las firmas de una misma persona poseen variaciones, se establece un rango de que tan grande pueden ser haciendo uso de estos patrones. Una vez obtenidos los datos, se entrena un algoritmo de K-Vecinos, el cual se encargará de verificar las firmas a partir de las características extraídas al comparar un patrón contra la firma cuestionada. Aquí, se realizó una mejora al enfoque original, al agregar al conjunto de datos nuevas variables que aportan información global de las firmas.

También, se implementó otro enfoque basado en distancias. Este, utiliza la Distancia Euclídea y una serie de centros geométricos para obtener un conjunto de características que identifiquen a las firmas y además calcular umbrales para su posterior verificación. De igual forma, se creó una interfaz donde se puede analizar un conjunto de firmas y verificar una firma a partir de los modelos entrenados.

Palabras Claves: Verificación de firmas offline, morfología, Distancia Euclídea, minería de datos, procesamiento digital de imágenes.

Índice

Introducción

El uso de firmas para permitir que se realice una acción o un conjunto de acciones, se ha vuelto muy común en el día a día de las personas. Las firmas son mundialmente aceptadas dado que desde hace mucho tiempo atrás se establecieron como una identificación personal en diferentes tipos de situaciones. El problema aparece cuando alguien trata de falsificar nuestra firma, por lo tanto es necesario validar las firmas. He aquí donde entra el campo del reconocimiento de firmas. Hilton (1992) hablar sobre qué es una firma y cómo se produce, destaca que una firma posee al menos tres atributos, forma, movimiento y variación. Dado que una firma es creada al mover un lápiz en un papel, el movimiento es uno de los atributos más importantes. Una vez que la persona se acostumbra a realizar la firma, el movimiento es controlado por el cerebro sin prestar atención a los detalles.

En general, existen tres tipos de firmas: Simples, cursivas y gráficas. Las simples son aquellas donde la persona solo escribe su nombre de una forma elegante. Las cursivas a pesar de que poseen los caracteres del nombre, son escritas de forma corrida y usualmente en un solo trazo. Por último, las firmas gráficas, las cuales son una combinación de formas sin un sentido claro, normalmente son una representación burda del nombre del signatario.

Una gran cantidad de sistemas han sido desarrollados para resolver la problemática que representan las falsificaciones de firmas, estos sistemas se dividen en dos grandes conjuntos, los sistemas *online* (en línea) y *offline* (fuera de línea o sin conexión). La diferencia entre ambos radica en la forma de adquirir los datos, en los sistemas *online* se capturan los datos en vivo haciendo uso de dispositivos digitalizadores. Y al contrario de los *online*, los sistemas *offline* utilizan datos obtenidos mediante cámaras o *scanners*. En ambos casos, se utilizan técnicas que van desde el campo de la inteligencia artificial hasta el uso de métodos estadísticos para la verificación de las firmas.

En este trabajo se implementan dos algoritmos basados en métodos existentes, uno de ellos se basa en una serie de dilataciones morfológicas [1] las cuales se encargan de medir la variación permitida en una firma. Utilizando esta información e información global sobre la firma, se entrena un algoritmo de K-Vecinos el cual se encarga de verificar si una firma es falsificada o no. El segundo algoritmo implementado, utiliza los centros geométricos de la imagen y la Distancia Euclídea [2] para el cálculo de un vector de características que represente a las firmas de una persona y también, para calcular dos umbrales. Al momento de verificar una nueva firma se comparan los nuevos valores contra los umbrales para determinar su autenticidad.

Este documento está organizado de la siguiente forma: En el Capítulo 1 se plantea el problema que se quiere resolver, así como también la solución propuesta, objetivos generales y específicos y las características de la plataforma utilizada. Los conceptos básicos del Procesamiento Digital de Imágenes, del proceso de obtención de conocimiento a partir de datos, así como también la teoría sobre firmas y falsificaciones se presentan en el Capítulo 2. En el Capítulo 3 se detallan los métodos desarrollados, los distintos algoritmos utilizados y la interfaz creada. También, se presenta un análisis de los resultados obtenidos al realizar distintas pruebas sobre los algoritmos implementados (Capítulo 4) y por último las conclusiones y trabajos futuros (Capítulo 5).

Capítulo 1 – Descripción del Problema

La autenticación y verificación de una persona es un problema que ha generado muchas investigaciones. Los métodos son abundantes y se basan en diferentes características de los individuos, tales como la voz, los movimientos, las manos, las huellas digitales, la retina, el iris, entre otros. Uno de los métodos más comunes a la hora de identificar una persona es el uso de firmas. Estas, a lo largo del tiempo han tenido diferentes aplicaciones. La función tradicional de una firma es dar evidencia de algo; puede funcionar como identidad, es decir, para demostrar la procedencia de un documento u objeto, o para demostrar la intención o voluntad de alguien con respecto a algún documento. En la actualidad, puede ser utilizada en los sistemas bancarios o de seguridad, validación de tarjetas de crédito, etc.

El problema surge cuando la firma de un individuo es duplicada por otra persona o algún dispositivo. Sin un sistema que pueda identificar y validar la firma de una persona, la misma podría ser usada para realizar acciones ilegales o distintas a las deseadas por el dueño de la firma. Muchos entes que utilizan firmas como una forma de identificación, poseen distintas formas para validarlas pero estas pueden resultar ineficientes o propensas a ser afectadas por factores externos. Este es el caso en el que se utiliza a personas para determinar la autenticidad de una firma. Dicha persona puede presentar problemas de salud o no encontrarse en las condiciones óptimas, lo cual puede afectar el proceso de validación. En otros casos, los sistemas pueden ser poco fiables o utilizar técnicas que no cumplen con lo necesario para un sistema de este tipo.

En la actualidad existe una gran cantidad de sistemas cuyo propósito es solventar el problema anteriormente expuesto. Los investigadores utilizan distintos enfoques y están continuamente introduciendo nuevas ideas, conceptos y algoritmos. La verificación de firmas es un área importante de investigación en el campo de la autenticación de personas. Al realizar una verificación de firmas se tiene como objetivo distinguir entre una firma original y una falsificada. En los últimos años, se han realizado muchas investigaciones en esta área y se han creado una gran variedad de enfoques. Estos enfoques están dividido en dos tipos, *Online* (en línea) u *Offline* (sin conexión). En un sistema *online* los datos son obtenidos a través de tabletas electrónicas y se guarda información sobre la actividad a la hora de escribir; por ejemplo, la rapidez, la presión, etc. Por otro lado, en un sistema *offline* las firmas son escritas en papel y digitalizadas utilizando cámaras o *scanners*. Dado que el volumen de información disponible es menor, la verificación de firmas utilizando técnicas *offline* es más retador. Otro aspecto importante es la complejidad de las técnicas, estas pueden ser simples como el uso de Distancias o mucho más complejas

como los Modelos Ocultos de Markov o HMM por sus siglas en inglés. En específico existen dos métodos los cuales abordan el problema de la verificación de firmas de manera distinta.

Uno de ellos está basado en distancias haciendo uso de la Distancia Euclídea para la extracción de las distintas características de la firma, este es un método relativamente sencillo. El otro, utiliza operaciones morfológicas y minería de datos para la extracción y verificación de las firmas respectivamente; en comparación con el primero, es mucho más complejo. Estos enfoques son muy distintos entre sí, manejan las firmas y las verifican de forma diferente.

1.2 Solución Propuesta

Se plantea desarrollar y modificar dos métodos de verificación de firmas offline existentes. Se busca comparar los resultados entre un método simple basado en distancias y otro más complejo que involucra la minería de datos para realizar la verificación.

Para el desarrollo de este trabajo se cuenta con un conjunto de datos de firmas el cual está compuesto por 9 usuarios con 24 firmas genuinas y 12 falsificadas cada uno. Este, pertenece a los conjuntos de datos de la Conferencia Internacional en Análisis y Reconocimiento de Documentos 2011 (ICDAR).

En primera instancia, se desarrollarán los algoritmos necesarios para el pre procesamiento de las imágenes. El primer método a implementar toma como base que las firmas de cada persona poseen variaciones naturales, pero una falsificación se sale del rango de variaciones naturales de las firmas de esa persona. Este método, a partir de un patrón mide las variaciones en una firma al compararla con dicho patrón. Utilizando estas guías y aplicando una operación de XOR con imágenes binarizadas, se extraen las características que aportan información sobre estas variaciones. Luego, se utiliza un algoritmo de minería de datos para la verificación, por esta razón es necesario conocer el proceso de extracción de conocimiento a partir de datos el cual será explicado en el próximo capítulo.

El segundo método, utiliza los Centros Geométricos de la firma y la Distancia Euclídea para la obtención de una serie de puntos. Con estos, se procede a calcular un umbral el cual será utilizado al momento de verificar. Adicionalmente utilizando estos puntos, se calcula un vector base que representa a todas las firmas de un usuario.

Se toman estos dos enfoques muy distintos entre sí, con el fin de comparar su rendimiento. Por un lado, se utilizan algoritmos de minería de datos y operaciones morfológicas sobre las imágenes y por otro, se hace uso de las características geométricas de las imágenes y un esquema de distancias para obtener las características y la posterior verificación de una firma.

Adicionalmente se desarrollará una aplicación en la cual se puedan verificar firmas utilizando los métodos desarrollados y que permita crear nuevos modelos a partir de otro conjunto de datos. Esta aplicación contará con dos módulos principales:

- **Módulo de Morfología:** En este módulo se podrán realizar distintas acciones. Una de ellas es el pre procesamiento de un conjunto de datos nuevo. También, se pueden extraer las características del conjunto pre procesado anteriormente o de algún conjunto que se tenga disponible. Por último, se podrá verificar una firma utilizando el modelo creado en este proyecto o uno creado en los pasos anteriores.
- **Módulo de Distancia Euclídea:** Se podrán realizar las mismas acciones que en el módulo anterior, es decir, pre procesar un conjunto de datos, extraer sus características y verificar una firma utilizando el método de Distancia Euclídea.

1.3 Objetivo General

Implementar y modificar dos métodos de verificación de firmas offline ya existentes, tomando como base imágenes digitales de firmas y luego comparar sus resultados.

1.4 Objetivos Específicos

- Recolectar los datos necesarios para el entrenamiento de los métodos los cuales están conformados por imágenes digitales de firmas auténticas y falsificadas.
- Realizar el pre procesamiento de las imágenes obtenidas. Este pre procesamiento puede variar dependiendo del método utilizado.
- Extraer las características que se consideren necesarias que sirvan para identificar la imagen.
- Evaluar el desempeño de los métodos midiendo la tasa de rechazo y aceptación. Esto se realizará utilizando una parte del conjunto de firmas para probar los métodos y medir la cantidad de falsos positivos y falsos negativos que arroja cada uno.
- Desarrollar una aplicación la cual sea capaz de verificar firmas a partir de los métodos implementados y pueda ser utilizada para nuevos conjuntos de datos.

1.5 Metodología de desarrollo a emplear

Para el desarrollo de estos métodos, se usará una metodología Ad-Hoc. Esta metodología está diseñada y será aplicada específicamente para este proyecto. Los procedimientos a emplear están adaptados a los requerimientos del proyecto. Básicamente, consisten en el desarrollo de cada uno de los módulos por separado. Dado que un sistema de verificación de firmas posee distintas etapas: Adquisición

de datos, pre procesamiento, extracción de características, verificación y evaluación, se utilizará una parte de éstas como guía para el desarrollo.

En primera instancia, se desarrollarán los algoritmos para el pre procesamiento de imágenes. Una vez finalizados, se implementará la clase que contiene las funciones para la extracción de las características de las imágenes. Estos algoritmos serán utilizados por ambos enfoques. Una vez terminada esta clase, se procederá a revisar cada uno de las funciones anteriormente desarrolladas con la finalidad de mejorarlas y tratar de encapsular lo mejor posible cada una de las clases.

Luego, se creará una clase por método las cuales utilizarán las implementadas anteriormente. Paralelamente, se irá desarrollando una interfaz en otro proyecto. Una vez terminados los algoritmos y la interfaz, se mezclaran ambos proyectos para obtener así el producto final.

1.6 Plataforma de Software y Hardware

El proyecto será desarrollado y evaluado en la plataforma del Sistema Operativo Windows 8.1, se utilizará Microsoft Visual Studio 2013 como entorno de desarrollo y C# como lenguaje de programación. Adicionalmente se utilizarán las siguientes librerías:

- **AForge.NET**: Es un framework de C# de código abierto, posee un conjunto de librerías de las cuales se utilizaron AForge.Imaging y Aforge.Math
- **WEKA**: Por sus siglas en inglés, *Waikato Enviroment for Knowledge Analysis*. Es una colección de algoritmos utilizados para las tareas de minería de datos. Es una librería de código abierto nativa de JAVA, por lo cual se realizó una conversión para poder ser usada en C#. Este proceso se realizó con la ayuda de IKVM.NET [3]

Las características del computador a usar para el desarrollo y las pruebas son las siguientes:

- **Procesador**: Intel Core i5-2310. 2.90GHz
- **Memoria RAM**: 8GB DDR3
- **Tarjeta Gráfica**: Nvidia GeForce GTX 550 Ti

Las pruebas a realizar consisten en la creación de distintos conjuntos de datos, variando sus tamaños y atributos en el caso del primer enfoque. Para el segundo se variaran los tamaños de las imágenes y las cantidades utilizadas a la hora de entrenar el algoritmo.

Capítulo 2 – Marco Teórico

La creación de un sistema de verificación y reconocimiento de firmas se divide en distintas partes. Una de las más importantes, es la aplicación de distintos algoritmos a las imágenes de las firmas con la finalidad de prepararlas para su posterior análisis. Para esto, es necesario conocer los conceptos básicos y las técnicas más importantes que pueden ser utilizadas.

El entendimiento del Procesamiento Digital de Imágenes es parte crucial en el proceso de desarrollo en un sistema de este tipo, así como también lo es la comprensión de las distintas características de las firmas y las falsificaciones; es necesario conocer y entender los aspectos que componen una firma para poder así identificar una falsificación. Por su parte, muchos sistemas de verificación de firmas utilizan algoritmos de minería de datos en su desarrollo. A continuación, se presentan los conceptos básicos y técnicas sobre el Procesamiento Digital de Imágenes, Minería de Datos y la teoría básica para la comprensión de las firmas y las falsificaciones.

2.1 Procesamiento Digital de Imágenes

Previo a la definición de Procesamiento Digital de Imágenes, es necesario definir el concepto de imagen. Una imagen es una representación gráfica de un objeto, un hecho, una persona o de cualquier elemento físico, la cual contiene información descriptiva sobre el objeto que representa. Las imágenes pueden ser clasificadas en distintos tipos, basándose en su forma o método de generación.

Dentro del conjunto de imágenes existe un subconjunto muy importante, el cual contiene todas las imágenes visibles, aquellas que pueden ser vistas y percibidas por el ojo humano, las cuales incluyen fotos, pinturas o dibujos. También existe otro subconjunto el cual contiene las imágenes ópticas, las cuales son formadas por lentes u hologramas.

El hecho de que una imagen pueda ser apreciada por el ojo humano no implica que puede ser procesada por un computador. Para que esto suceda, la imagen tiene que ser digitalizada. Una imagen digital, ya sea generada por un computador o tomada a través de algún dispositivo, puede definirse como una representación numérica de una imagen comúnmente de forma binaria, donde estos bits contienen información de color, posición, tamaño y resolución que estructura a la imagen.

Dependiendo de la organización de los bits, se pueden definir dos tipos de imágenes, como se observa en la Figura 1:

- **Imágenes Vectoriales:** Las imágenes vectoriales poseen información que describe líneas y curvas. Estas imágenes pueden ser escaladas y aun así mantener su forma
- **Imágenes de Mapa de Bits:** Estas imágenes poseen datos que describen los colores de cada píxel. Este tipo de imágenes al contrario de las vectoriales, pierden detalle y definición cuando son escaladas [4]

La palabra píxel es una combinación de las palabras “picture” y “element”. Un píxel no es más que un punto de muestreo. Para una imagen a color, un píxel puede tener tres muestras, una para cada componente de color que contribuye a la imagen [5]. El píxel de una imagen queda definido por un par de coordenadas específicas pertenecientes al plano de ésta [6]. Mientras más cantidad de píxeles tenga una imagen, más definida será. Cada píxel se codifica mediante un conjunto de bits de cierta longitud, este tamaño va desde 1bit (2 colores) en adelante.

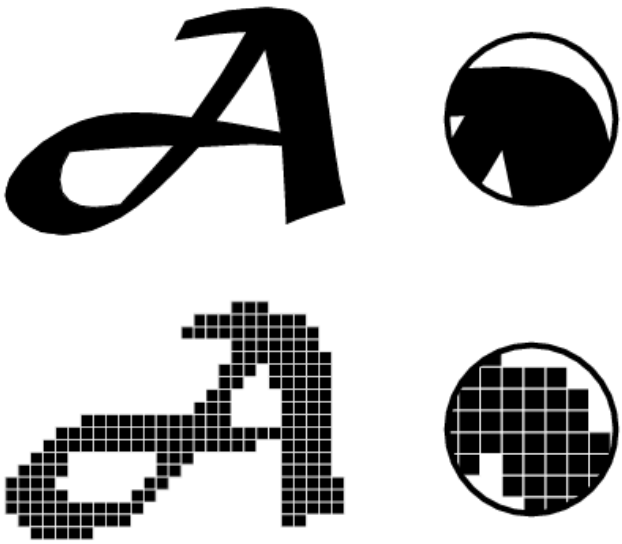

Figura 1: Imagen Vectorial (Arriba), imagen de Mapa de Bits (Abajo)

Asimismo, para el procesamiento de un píxel, se debe conocer el modelo de color, que no es más que el sistema ordenado con el que se crea todo un rango de colores tomando un pequeño grupo de colores como punto inicial. Existen dos tipos de modelos de color, el primero de ellos es el modelo aditivo. En éste, los colores que se obtienen mediante la mezcla de la luz entre dos o más componentes de color. El segundo modelo, es el sustractivo, con el cual los colores se obtienen mediante la sustracción de la luz de los colores. Entre los distintos modelos de color más comunes están (Figura 2):

- **RGB:** Este modelo crea su gama de colores de forma aditiva, usando Rojo, Verde y Azul. Las computadoras usualmente usan este modelo. Los valores por cada componente van de 0 a 255.
- **CMYK:** Este modelo utiliza los colores cian, magenta y amarillo para crear todo el rango de colores, pero dado que con iguales cantidades de CMY se obtiene un color marrón, suele usarse K la cual representa el color negro.
- **HSV:** Viene de las siglas en inglés *Hue*, *Saturation*, *Value* (Matiz, Saturación y Valor). Se define en términos de sus componentes. Trata de modelar la forma en la que los artistas trabajan [7].
- **HSL:** Por sus siglas en inglés, *Hue*, *Saturation*, *Lightness* (Matiz, Saturación y Luminosidad). Es visto como un círculo. El matiz representa el ángulo en el círculo; en pocas palabras, simboliza los colores. La saturación nos dice que tan puro es un color. Un color completamente puro esta 100% saturado, mientras que los colores grises no poseen ninguna saturación. La luminosidad representa que tanta luz u oscuridad hay en el matiz. El color blanco tiene un 100% de luminosidad y el negro un 0%.

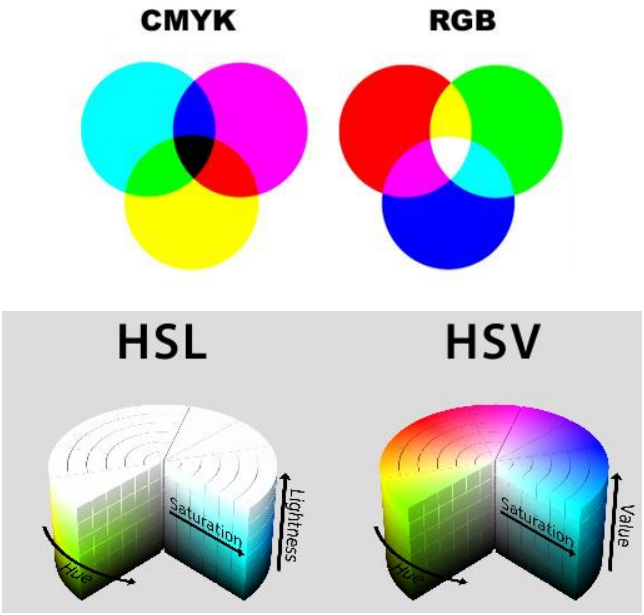

Figura 2: Modelos de Color

Cuando se captura una imagen haciendo uso de algún dispositivo, ya sea una cámara, un scanner, etc., pueden ocurrir distintos fenómenos externos o internos que afecten de manera negativa a la imagen

en el proceso de captura. Si esto ocurre, suele existir la necesidad de preparar la imagen para su posterior procesamiento, mediante la aplicación de distintas técnicas de procesamiento digital de imágenes. Dentro de esta perspectiva, se puede definir el procesamiento digital de imágenes como el acto de aplicar distintos algoritmos a imágenes digitales con el objetivo de mejorar la calidad, resaltar formas, eliminar imperfecciones, analizar la imagen, extraer información relevante, entre otros. Existe una gran variedad de técnicas que se pueden utilizar en una imagen digital. A continuación se detallan algunas técnicas relevantes para este trabajo.

2.1.1 Técnicas de Procesamiento Digital de Imágenes

Existe una gran cantidad de métodos o técnicas utilizados para modificar o analizar las imágenes, desde aclarar la imagen, obtener las distintas formas que esta contiene y extraer las características más importantes de esta. A continuación se presentan algunas de las técnicas más comunes:

A. Binarización o Thresholding

Entre las distintas técnicas que se utilizan en el procesamiento digital de imágenes se encuentra la binarización. Es la forma más simple de segmentar una imagen. También, es comúnmente utilizada para saber cuáles píxeles en una imagen están fuera de un rango de intensidades. Se utiliza al menos un canal de color o una imagen en escala de grises para el proceso y como resultado se obtiene una imagen en blanco y negro que representa la segmentación como se observa en la Figura 3 [8]. Dependiendo de la aplicación, un primer plano puede ser representado por un nivel de gris de 0 y el fondo por la mayor luminosidad, lo que quiere decir que los objetos son de color negro y el fondo de color blanco lo cual en una imagen de 8-bits estaría representado por el valor de 255. También se puede representar el fondo con el color negro y los objetos con blanco [9] e incluso utilizar dos colores arbitrarios para diferenciar el fondo del objeto.

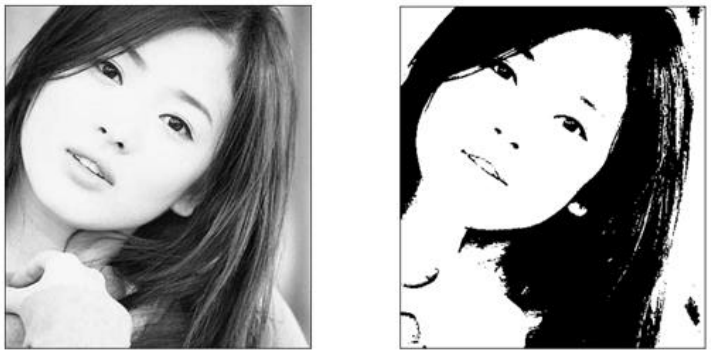

Figura 3: Izquierda, imagen original. Derecha Imagen después de la binarización

B. Filtros Digitales

Cuando se captura una imagen existen diversos factores tales como la luz, fallas en el dispositivo o en la transmisión que pueden alterar el resultado de dicha captura. Esto es comúnmente conocido como ruido. Un píxel se considera como ruido cuando la diferencia entre su intensidad y la media de las intensidades de sus vecinos se encuentra por encima de un umbral. Cuando se habla de píxeles vecinos de un píxel con coordenadas i, j ($P_{i, j}$) se hace referencia a todos los píxeles que se encuentran dentro de una máscara de n x n píxeles donde n es un número impar mayor a 1 y el centro de esta mascara se encuentra en $P_{i, j}$ [10]. Además del ruido, existen otros factores que pueden modificar las imágenes como es el caso de la distorsión que provoca una falta de precisión debido a la falta de información [11].

Para solucionar estos problemas existe una gran cantidad de filtros digitales, los cuales permiten de manera limitada solventar estos inconvenientes en las imágenes. Un filtro, puede definirse como un operador L que transforma una señal X en una señal Y. Estos pueden ser divididos en dos categorías.

a. Filtros Lineales

En estos filtros el operador utilizado (L) es un operador lineal, por lo tanto cumple el principio de superposición. Este principio engloba las propiedades de escalado o proporcionalidad y aditividad. Es decir, el resultado de cada píxel se obtiene con la combinación lineal de sus vecinos. Para aplicar un filtro lineal se multiplica el entorno de cada píxel por una máscara de tamaño fijo la cual contiene los pesos con que cada uno de los vecinos contribuye al valor final del píxel.

Esta máscara es denominada *kernel* o máscara de convolución. El concepto de *kernel* se entiende como una matriz de coeficientes la cual se desplaza por cada píxel de la imagen para así obtener el valor del nuevo píxel. Dicho valor es calculado a partir de todos los valores de los píxeles que estén dentro de ella. Los coeficientes y tamaño de esta matriz afectarán el resultado del píxel final; por esta razón, el tipo de filtro es establecido por el contenido del *kernel* utilizado.

Entre los filtros lineales se encuentran:

- **Filtros de paso bajo:** Atenúan las frecuencias altas y mantiene las bajas sin variación. Se utilizan para suavizar la imagen o reducir el ruido.
    - Media: Se obtiene el valor del píxel calculando la media de sus vecinos (Figura 4)
    - Gaussiano: Simulan una distribución gaussiana. El valor máximo aparece en el píxel central y disminuye hacia los extremos.

- **Filtros de paso alto:** Atenúa las frecuencias bajas sin cambiar las altas. Realzan los detalles en la imagen, su objetivo es resaltar las zonas con mayor variabilidad. Otros filtros de realce de bordes son:
    - **Realce de bordes mediante desplazamiento:** Se toma la imagen original y se sustrae de esta una copia desplazada con lo que se obtienen los bordes existentes.
    - **Realce de bordes mediante Laplace:** Realza las áreas con cambios bruscos en la intensidad, por esta razón es utilizado para la detección de bordes (Figura 5).
    - **Detección de bordes y filtros de contorno:** Se utilizan para detectar los contornos de las imágenes y así poder clasificar las formas que están dentro de la imagen. Entre estos se encuentran los filtros de Sobel y Prewitt (Figura 5).

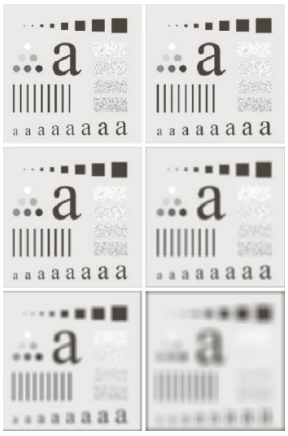

Figura 4: Filtro de Media con máscaras de 3, 5, 9, 15 y 35

Figura 5: Imagen Original, Filtro Laplaciano, Filtro de Sobel

b. Filtros no Lineales

En estos filtros, en lugar de utilizar un promedio ponderado se emplean otras técnicas que combinan los valores del grupo de píxeles de entrada. Algunos de los tipos de filtros no lineales son los siguientes:

- **Filtros de orden estadístico:** Estos filtros están diseñados para preservar la información de los contornos y conservar los detalles de la imagen. El principal filtro de este tipo es el de la mediana. Existen otros filtros de orden estadístico y en su mayoría buscan optimizar el filtro de la mediana o combinarlo con alguna otra técnica.
- **Filtros morfológicos:** Son aplicados a imágenes binarias. Están basados en una serie de operaciones morfológicas que actúan tomando como base las características geométricas y topológicas de la imagen. Con estos filtros se puede eliminar cualquier tipo de ruido de forma satisfactoria [12]. Aquí, se utiliza un elemento estructurante, el cual juega el mismo rol que una matriz de convolución en los filtros lineales. Este elemento estructurante está compuesto de 0's y 1's, las posiciones que poseen un valor de 0 son ignoradas. Los filtros morfológicos se basan en dos operaciones básicas (Figura 6):
    - Erosión: Esta operación, utilizando un elemento estructurante pequeño reduce la imagen al eliminar una capa de píxeles de los bordes interiores y exteriores. Los espacios o huecos entre las regiones aumentan, y los detalles pequeños son eliminados
    - Dilatación: Tiene el efecto contrario a la erosión, dado que añade una capa de píxeles a los bordes internos y externos de cada región. Los espacios dentro de una región y entre regiones se vuelven más pequeños.

  Los filtros morfológicos más comunes son Apertura y Cierre. La apertura consiste en aplicar una erosión seguida de una dilatación y el cierre consiste en una dilatación seguida de una erosión. Estos filtros son idempotentes, es decir la aplicación sucesiva de estos operadores no cambia el resultado. Otra operación morfológica importante es el *Thinning* o adelgazamiento, este procedimiento es utilizado para eliminar las diferencias de grosor en los objetos de la imagen, haciendo que tengan un píxel de grueso.
- **Filtros adaptativos:** Estos filtros analizan características de la imagen y su ruido y aplican el método más conveniente para eliminarlo. Pueden llegar a utilizar distintos métodos en las diferentes regiones de la imagen (Figura 7) [12].

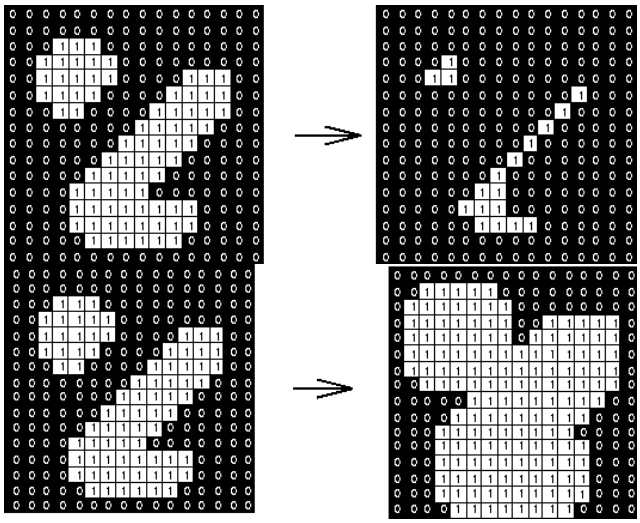

Figura 6: Arriba, Erosión 3x3. Abajo, Dilatación 3x3

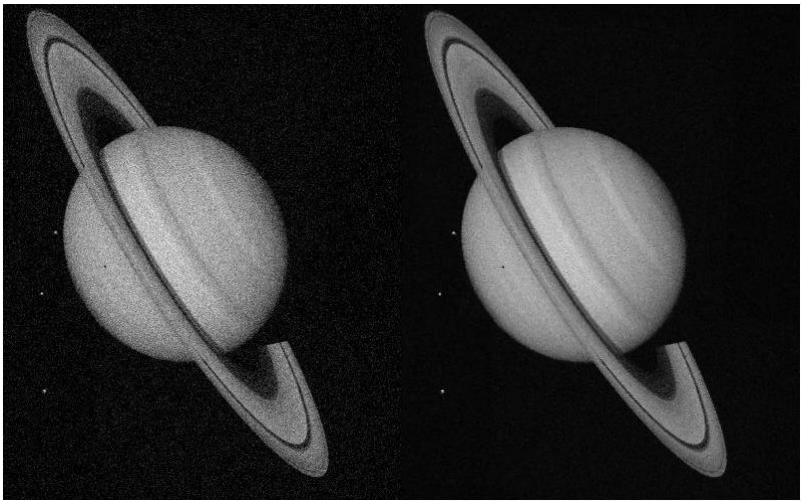

Figura 7: Izquierda: Imagen con ruido. Derecha: Imagen recuperada usando filtros adaptativos

2.2 Proceso de descubrimiento de conocimiento a partir de datos

A lo largo de una extensa variedad de campos, datos son recolectados y acumulados a una velocidad dramática. Hay una necesidad urgente por una nueva generación de teorías computacionales y herramientas que asistan a los humanos en el proceso de extraer información útil (conocimiento) de estos volúmenes de datos digitales que crecen de forma extremadamente rápida. Estas teorías y herramientas pertenecen al campo del KDD (*Knowledge Discovery in Databases*) por sus siglas en inglés o descubrimiento de conocimiento a partir de datos [13].

Históricamente, la noción de encontrar patrones útiles en los datos ha tenido distintos nombres, entre estos minería de datos, extracción de conocimiento, descubrimiento de información, cosecha de información, etc. El termino minería de datos ha sido mayormente utilizado por estadísticos, analistas de datos y además ha ganado popularidad en el cambo de base de datos.

KDD se refiere al proceso de descubrimiento de conocimiento a partir de datos y la minería de datos es un paso particular en este proceso. En términos generales, la minería de datos se define como la aplicación de distintos algoritmos para la extracción de patrones en datos. Es importante diferenciar el proceso de KDD de la etapa de minería de datos la cual está incluida en este proceso. Los patrones encontrados con KDD deben ser válidos, novedosos, potencialmente útiles y en última instancia comprensibles [13].

Cuando se habla de patrones válidos, se refiere a que estos deben ser precisos para los datos nuevos y no solo para los que se usaron en su obtención. Un patrón es novedoso cuando aporta nuevos conocimientos para el sistema y para el usuario. A su vez, que sea potencialmente útil quiere decir que debe llevar a acciones que posean algún beneficio, como la ayuda a la toma decisiones. Por último, los patrones deben facilitar su interpretación, lo cual les otorga la característica de comprensibles.

El proceso KDD tiene una serie de fases que deben ser cumplidas (Figura 8); este proceso es iterativo; es posible volver a una fase anterior las veces que sean necesarias hasta que se obtengan los resultados deseados. A continuación se detallan cada uno de los pasos en el proceso KDD.

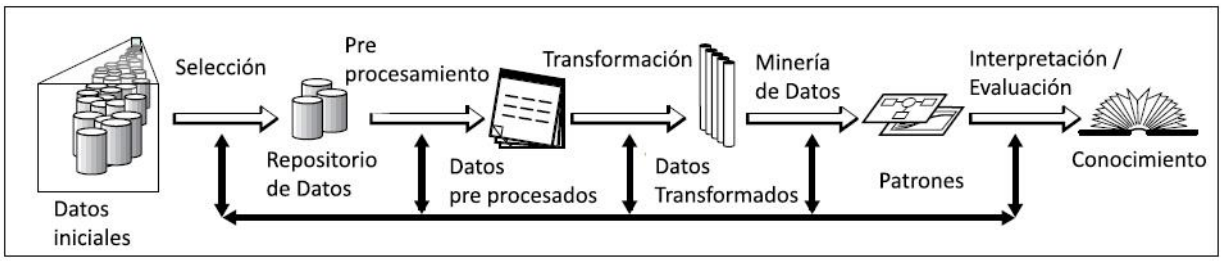

Figura 8: Fases del proceso KDD

2.2.1 Integración y Recopilación de Datos

Se deben definir los distintos objetivos y determinar los datos que serán usados para descubrir el conocimiento. Esto incluye saber cuáles datos están disponibles, obtener los datos necesarios, para luego integrarlos en un solo conjunto con un único formato, incluyendo los atributos que se tomarán en cuenta durante el proceso.

2.2.2 Selección, limpieza y transformación

Durante esta etapa se aumenta la confiabilidad y exactitud de los datos. Dado que en ciertas ocasiones los datos provienen de distintas fuentes, estos pueden contener valores errados o faltantes. El objetivo de esta fase es eliminar y corregir los datos que se detecten con errores o que no estén presentes. Adicionalmente, se escogen los atributos que serán relevantes, es decir que hagan la tarea de minería más sencilla. Por último, dependiendo del proyecto, estos datos pueden ser transformados, por ejemplo discretizar los atributos numéricos, cambiar el tipo de atributo, entre otros.

2.2.3 Minería de Datos

Como se mencionó anteriormente, la minería de datos es un paso en el proceso KDD, el cual consiste en obtener patrones o modelos a partir de los datos recopilados. Durante esta fase, se deben tomar ciertas decisiones, entre ellas está escoger la tarea de minería de datos a utilizar; también se debe decidir cuál algoritmo o técnica de minería será utilizada.

Las tareas de minería, pueden ser descriptivas o predictivas. Las predictivas son aquellas donde el sistema encuentra patrones para predecir el comportamiento futuro de algunas entidades. Por otra parte, con las tareas descriptivas, el sistema busca patrones para ser presentados a un usuario en una forma que pueda ser leída por un humano [13]. Las predicciones se usan para prever el comportamiento futuro de una entidad, mientras que una descripción puede ayudar a su comprensión. En realidad, un modelo predictivo puede ser descriptivo mientras sea comprensible por personas, adicionalmente, los modelos descriptivos pueden ser utilizados para realizar predicciones [14]. Entre las tareas predictivas se tienen la regresión y la clasificación.

En la clasificación se tiene un atributo que representa una clase, cuyo valor está dentro de un conjunto finito de etiquetas. La regresión se encarga de asignar a cada instancia un valor numérico en vez de una etiqueta.

Entre las tareas descriptivas se encuentran la agrupación y la asociación. La agrupación se encarga de buscar grupos en el conjunto de datos, teniendo en cuenta la similitud de los valores de las instancias. La asociación busca relaciones entre los atributos del conjunto de datos.

Por su parte, las técnicas de minería de datos se clasifican en dos grandes categorías: Supervisadas y no supervisadas. La Figura 9, representa para qué propósito son más utilizadas estas técnicas. Las técnicas no supervisadas se refieren a aquellas que agrupan instancias sin un atributo dependiente previamente especificado. Por el contrario, las técnicas supervisadas, intentan descubrir la relación entre los atributos de entrada y un atributo llamado objetivo o *target*; esta relación es representada como un modelo. Usualmente, los modelos describen y explican fenómenos, los cuales están ocultos en el conjunto

de datos y pueden ser usados para predecir valores del atributo *target* conociendo los valores de los atributos de entrada [15].

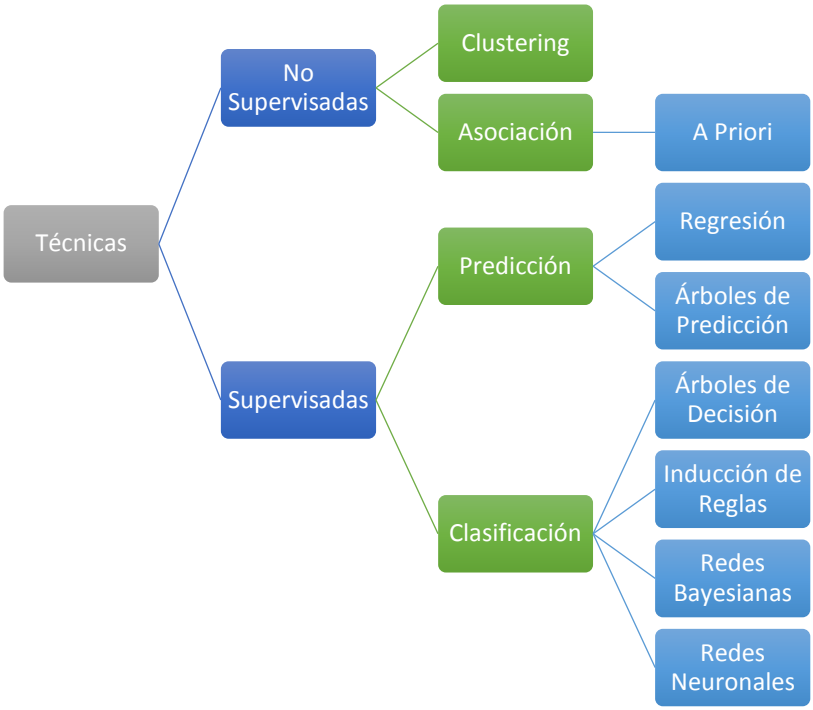

Figura 9: Técnicas de minería de datos

La mayoría de las técnicas de minería están basadas en aprendizaje inductivo, donde se construye un modelo, ya sea de manera implícita o explícita, por medio de la generalización de una cantidad de ejemplos de entrenamiento. A continuación se detallan las técnicas que influenciaron este trabajo.

2.2.4 Técnicas de Minería de Datos

A. Árboles de decisión

Estos están englobados dentro del aprendizaje supervisado, son un conjunto de condiciones organizadas en una estructura jerárquica, de manera que la decisión final se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas. Se tienen tres tipos de nodos: raíz, interno y hoja. El nodo raíz y el interno realizan pruebas sobre un atributo, a su vez el nodo hoja representa los valores que retorna el árbol; estos nodos están interconectados por arcos, los cuales representan los diferentes valores que un atributo puede tener [16].

B. K-Vecinos (KNN)

Esta técnica busca dentro del conjunto de datos el grupo de las K instancias o vecinos con la menor distancia con respecto a un objeto sobre el que se hará la predicción. Es utilizado para la clasificación en la cual la predicción estará basada en la clase que predomina dentro del grupo mencionado anteriormente. También puede ser utilizado para la regresión, en este caso la predicción es el valor medio de las K instancias más cercanas. La fase de entrenamiento de este algoritmo consiste solamente en guardar los vectores característicos y las clases de las instancias del conjunto de entrenamiento. Para la fase clasificación, K es una constante definida por el usuario y un vector sin clase es clasificado por medio de la asignación de la clase que es más frecuente entre las K instancias de entrenamiento más cercanas (Figura 10).

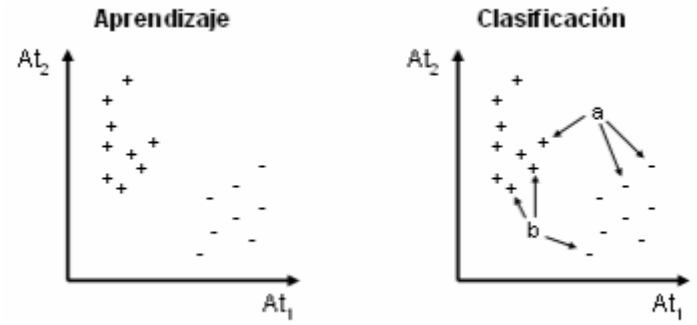

Figura 10: Ejemplo de aprendizaje y clasificación con KNN

C. Redes neuronales

Las redes neuronales o redes neuronales artificiales, NN por sus siglas en inglés, han disfrutado de una gran popularidad entre los investigadores durante los últimos 20 años y han sido aplicadas de forma satisfactoria en la resolución de distintos problemas en casi todas las áreas de negocios, industrias y ciencia. Hoy en día, las redes neuronales son tratadas como herramientas de minería de datos y usadas en varias tareas de minería tales como la clasificación de patrones, predicciones, etc.

Las redes neuronales son modelos computacionales para el procesamiento de información, y son particularmente útiles para la identificación de relaciones fundamentales entre un conjunto de variables o patrones de datos. Específicamente, tratan de imitar el aprendizaje de las redes neuronales biológicas, especialmente aquellas que se encuentran en el cerebro humano. A pesar de que las NN artificiales son abstracciones muy simples de los sistemas biológicos, estas comparten dos

características muy importantes; el procesamiento paralelo de información y el aprendizaje y generalización por medio de la experiencia [15].

Como se mencionó anteriormente, las NN se comportan de forma similar a nuestro cerebro, aprendiendo de la experiencia y del pasado, y aplicando ese conocimiento a la resolución de nuevos problemas. Este aprendizaje se obtiene como resultado de un entrenamiento. Una vez que las NN han sido entrenadas, estas pueden hacer predicciones y clasificaciones. Por desgracia, uno de los mayores inconvenientes de las NN es la dificultad de acceder y comprender los modelos que generan, además de que presentan dificultades para extraer las reglas creadas por el modelo. Otra de sus características es que pueden trabajar con datos incompletos, lo cual puede resultar en una ventaja o no dependiendo del problema. Las redes neuronales se construyen estructurando una serie de niveles o capas compuestas por nodos o neuronas [14]. Una NN debe disponer de mínimo tres capas, como se muestra en la Figura 11; en caso de tener más, estas forman parte de la Capa Oculta.

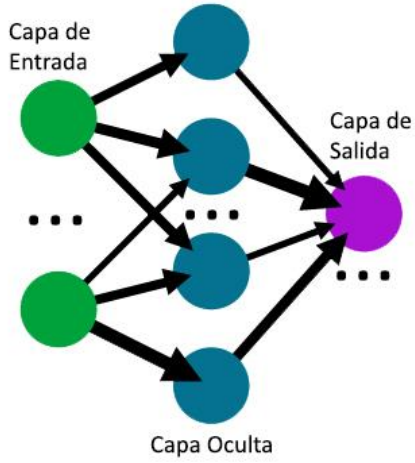

Figura 11: Estructura de una red neuronal

2.2.5 Evaluación e interpretación

Una vez que se obtiene el modelo, este debe ser validado para comprobar que las conclusiones que arroja son válidas y satisfactorias. De no ser así, se debe ir a una fase anterior y repetir el proceso. Para realizar una evaluación se debe utilizar una técnica y una medida de evaluación la cual depende de la tarea seleccionada. Una de las técnicas de evaluación más común es la validación cruzada o *Cross Validation*, esta divide el conjunto de datos en N particiones disjuntas de manera aleatoria. Estas

particiones se usarán para evaluar el modelo N veces y el resto para el entrenamiento. Los errores calculados son el promedio de todas las ejecuciones.

Adicionalmente, es necesario utilizar una medida de evaluación la cual, al igual que la técnica de evaluación, depende de la tarea que se utilizó anteriormente. Esta medida, determina que tan bien un patrón en particular cumple los objetivos del proceso de KDD.

2.2.6 Difusión y uso

El conocimiento se obtiene para realizar acciones, ya sea para incorporarlo en un sistema o para almacenarlo y reportarlo a las personas interesadas. Este proceso también incluye la resolución de conflictos con conocimientos previamente extraídos [13].

2.3 Firmas y Falsificaciones

La firma de una persona posee numerosas características por las cuales puede ser identificada. Estas características pueden ser divididas en dos grupos: características de clase y características individuales. Las características de clase son el resultado del estilo o forma de escritura que emplea el signatario el cual ha sido enseñado o adoptado. Por otro lado, las características individuales son resultado de numerosos factores, tales como la coordinación muscular, salud, edad, temperamento, personalidad o carácter del signatario [17].

Para una persona que escribe todos los días, el firmar se vuelve un hábito, lo que resulta en que posea una cierta rapidez, ritmo y movimiento. Por el contrario una persona que escribe ocasionalmente, creara firmas menos uniformes [17]. Pero por más que el signatario esté acostumbrado a firmar, la firma siempre poseerá cierta variación, de hecho, esta variación es en sí una prueba de que la firma es genuina. Sin embargo, una falsificación también poseerá variaciones, pero estas no forman parte de las variaciones naturales que puede poseer una firma auténtica, en la Figura 12 se pueden observar ejemplos de firmas genuinas y falsificadas.

Las falsificaciones de firmas han sido categorizadas basándose en sus rasgos más característicos [18]. Los tres principales tipos de falsificaciones son:

- **Falsificación Aleatoria:** El signatario usa el nombre de la víctima con su propia forma de escribir. Son las más fáciles de detectar.
- **Falsificación Inexperta:** El signatario imita la firma en su propio estilo sin tener ningún conocimiento ni previa experiencia. Esta imitación se puede notar al observar las firmas detenidamente.

- **Falsificación Experta:** Esta falsificación es creada por profesionales o personas que tienen experiencia copiando firmas. Es la más difícil de detectar de todas las falsificaciones.

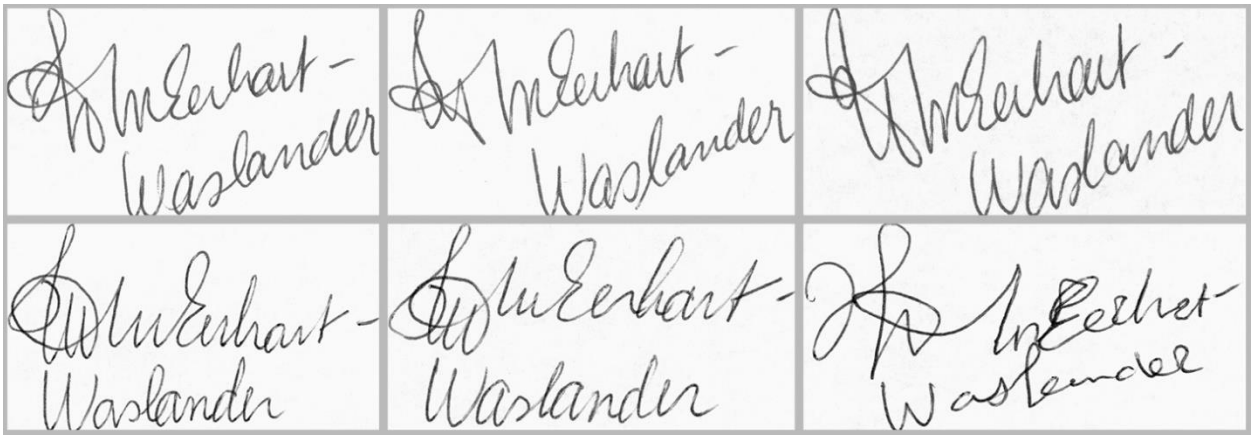

Figura 12: Arriba: Firmas Genuinas. Abajo: Firmas Falsificadas

Tanto las firmas genuinas como las falsificadas poseen una serie de características. El entendimiento de estas es crucial para determinar qué aspectos o rasgos de las firmas son los más importantes para la verificación automática de firmas. Madasu & Lovell [19] mencionan algunas de estas caracteristicas:

- **Alargamiento de los caracteres:** Una falsificación es usualmente más larga que una firma original. Un falsificador se toma más tiempo a la hora de realizar la firma que el autor original, esto hace que la firma falsificada sea más larga, no solo las letras por separado sino también la firma en sí.
- **Tendencia a que las curvas posean un mayor ángulo:** El falsificador normalmente tiene cuidado para así obtener la forma correcta de la letra, usando una velocidad lenta para imitar perfectamente la curva. Mientras más tiempo se pase haciendo la curva, la letra se tornara más angulosa y viceversa.
- **Retoque:** En muchas ocasiones el falsificador hace retoques en la firma una vez que ya ha sido finalizada o en partes que ya ha escrito. Por esta razón, se pueden encontrar líneas que no siguen el flujo continuo de la firma o que son más gruesas en ciertos puntos.
- **Líneas con baja calidad:** La presión hecha en el papel por el bolígrafo no es la misma en el caso de una firma genuina y una falsificada. Se ha encontrado, que la presión puesta en una firma falsa es mayor a la puesta en una original. La tinta puede revelar distintas variaciones en la presión, rapidez o cantidad de ésta en el papel. En algunos casos, se encuentra menos presión en una firma falsificada, lo cual puede ser ocasionado por dudas o temblores a la hora de imitar la firma [4].

- **Indecisión:** Cuando está imitando una firma, el falsificador puede pararse a consultar la firma original y luego continuar con el proceso, lo cual hace que aparezcan puntos más gruesos o manchas.
- **Puntuación:** En las falsificaciones se pueden encontrar puntos en las letras 'i' que están mal puestos, faltan o que sobran.
- **Variación en la presión:** Es difícil variar la presión del bolígrafo igual que lo hace el signatario original. Los falsificadores no pueden imitar la presión, esta puede ser muy fuerte o muy suave, dependiendo del estilo del falsificador.
- **Terminaciones repentinas:** Esta es una característica importante en una falsificación. En muchos casos la firma original termina de forma suave mientras que la falsa simplemente se detiene.
- **Características del Falsificador:** Inconscientemente la persona que imita la firma, revela características de su propia escritura en esta. El falsificador no puede evitar esto, y se ve reflejado en la forma, espaciado y posición de las letras.
- **Error de Línea Base o *Baseline Error*:** Esa línea imaginaria que va debajo de la firma, varía entre la firma original y la falsificada. Esta línea normalmente no es horizontal por lo que cualquier variación notable en ella es un indicador de falsificación.
- **Espaciado:** Imitar el espacio que existe entre letras, palabras completas y entre las puntuaciones y las palabras es un trabajo muy difícil, este espaciado puede ser tan grande o tan pequeño que no puede ser copiado al escribir la firma.
- **Caracteres que no están en la firma:** Cuando los falsificadores conocen el nombre del autor de la firma original, inconscientemente incluyen letras que no están en la firma genuina. Por otra parte, si el falsificador esta inseguro del nombre de la persona, pueden aparecer letras que no son correctas en la firma falsificada.

2.3.1 Conceptos básicos en verificación de firmas

Un sistema de reconocimiento y verificación de firmas (SRVF) es un sistema capacitado para alcanzar de forma eficiente dos tareas principales: la primera, identificar a quién pertenece la firma y segundo, decidir si esta es genuina o no. Dependiendo de las necesidades del problema, estos sistemas son comúnmente categorizados en dos principales clases, online SRVF y offline SRVF. La diferencia entre estas dos clases recae en como son obtenidos los datos. En un sistema online los datos son obtenidos a través de una Tablet o algún otro dispositivo. En los sistemas offline, los datos están constituidos por imágenes de las firmas obtenidas usando un scanner o una cámara. El objetivo de un SRVF offline es

decidir si una firma pertenece a una persona utilizando la imagen de la firma y algunas otras imágenes de la firma original. Muchos sistemas se enfocan solo en el reconocimiento o solo en la verificación de firmas.

Durante el desarrollo de un sistema de verificación de firmas, se tienen diferentes etapas las cuales siguen la siguiente secuencia.

- Adquisición de datos y pre procesamiento
- Extracción de características
- Verificación de la firma
- Evaluación del sistema

A. Adquisición de datos y pre procesamiento

La adquisición de datos es una etapa crucial en cualquier sistema de verificación de firmas. Dependiendo del tipo de sistema la forma en que se capturan los datos varía. En el caso de los sistemas offline, se obtienen imágenes digitales de las firmas mediante algún dispositivo luego de que el proceso de escribir la firma ha terminado. Estas imágenes pueden ser adquiridas haciendo uso de una cámara, un escáner, etc. Asimismo, los sistemas online realizan la captura de datos en tiempo real. Para esto, se utilizan dispositivos como tabletas digitalizadoras.

Luego de que se obtienen los datos, se procede a realizar el pre procesamiento de las imágenes o datos recopilados. El pre procesamiento consiste en el conjunto de operaciones subsecuentes las cuales son aplicadas para mejorar la calidad de la imagen de la firma. Esta mejora incrementa la precisión de las etapas siguientes sin perder información relevante. Estas operaciones son aplicadas a los píxeles de la imagen. Algunos de los distintos sub-procesos los cuales pertenecen al área de procesamiento de imágenes son: Binarización, adelgazamiento, reducción de ruido, entre otros. Estos sub-procesos pueden variar dependiendo del enfoque utilizado [20].

B. Extracción de características

La firma de una persona puede cambiar con el tiempo, lo que hace que un sistema de verificación de firmas pueda tener una tasa de errores alta, por lo tanto el desempeño del sistema depende principalmente de esta etapa. Las técnicas de extracción de características deben ser rápidas y fáciles de procesar. Las características elegidas deben diferenciar entre una firma genuina y una falsificación [21].

Existen dos tipos de características que validan una firma. Las características estáticas, las cuales se extraen de la imagen de la firma, y características dinámicas, que son extraídas de las firmas adquiridas en tiempo real. Estas últimas proporcionan información sobre la cantidad de trazos, rapidez, la presión del lápiz en cada punto, etc. Se tienen dos enfoques en el proceso de extracción de características, estos

son basados en parámetros y basados en funciones. Cuando las características basadas en parámetros son utilizadas, la firma es representada como un vector de elementos, en el cual cada uno representa un valor de una característica. Por otra parte, cuando las basadas en funciones son utilizadas, la firma es usualmente representada en términos de una función de tiempo cuyos valores constituyen el conjunto de características [21].

Además, las características basadas en parámetros se clasifican generalmente en dos categorías principales:

- **Globales:** Son aquellas que se extraen de la firma completa. Estas pueden ser extraídas fácilmente, pero aportan información limitada sobre la firma, no son afectadas por el ruido ni por las pequeñas distorsiones. Sin embargo, pueden ser afectadas por la posición y las variaciones en los estilos. Algunas son: Tamaño de la firma, *Aspect Ratio* (Relación de Aspecto), cantidad de levantamientos del lápiz, etc.
- **Locales:** Estas son extraídas de partes específicas de la firma. Dependiendo de las características obtenidas, pueden ser dividas en:
    - Parámetros orientados a componentes, los cuales son extraídos al nivel de cada componente. Tal es el caso de la posición de los trazos, orientación, *Aspect Ratio* de los trazos, etc.
    - Parámetros orientados a píxeles, son extraídos a nivel de píxeles, estos incluyen la densidad de píxeles, intensidad de niveles de grises, etc.

  Las características locales son computacionalmente más costosas, pero más precisas que las globales [21].

La técnica ideal para obtener las características de la firma, se encarga de extraer el conjunto mínimo de características que maximicen la distancia interpersonal entre las firmas de varias personas mientras minimiza la distancia intrapersonal para las firmas que pertenecen a una misma persona [22].

Se han usado distintos tipos de características en sistemas de verificación de firmas y cada elección tiene su propia tasa de aciertos y fallas. Muchas características locales poseen la misma naturaleza que las globales, la diferencia es que se aplican a partes de la firma obtenidas, ya sea por segmentación o luego de dividirla en celdas. Cabe destacar que, si se combinan características globales con las locales se pueden obtener resultados mucho mejores, dado que se aprovechan las ventajas de ambos enfoques y al mismo tiempo se pueden evitar sus deficiencias creando un conjunto de características más discriminativo y efectivo.

C. Verificación de la firma

En el proceso de verificación, se prueba la autenticidad de la firma evaluando sus características contra las almacenadas en la base de datos. Este proceso devuelve un solo valor, verdadero o falso. Esta fase comprende varios aspectos críticos que van de la elección de la técnica para comparar firmas a la estrategia utilizada para el desarrollo de la base de conocimientos. Existen diversos enfoques en la verificación de firmas y en muchos casos se pueden adoptar soluciones mezcladas [23].

D. Evaluación del sistema

Un sistema de verificación de firmas puede tener dos tipos de errores:

- **Error Tipo I:** Cuando una firma genuina se clasifica como falsificada se considera como un falso negativo.
- **Error Tipo II:** Cuando una firma falsificada se clasifica como genuina se considera como un falso positivo.

Además de esto, existen dos valores más que pueden ser medidos para calcular el rendimiento del sistema:

- **Verdaderos Positivos:** Mide la cantidad de firmas genuinas que son aceptadas
- **Verdaderos Negativos:** Mide la cantidad de firmas falsificadas que son rechazadas

Teniendo en cuenta los errores explicados anteriormente el rendimiento del sistema es evaluado en términos de la tasa de falsos negativos y falsos positivos o también conocidos como FRR (*False Rejection Rate*) y FAR (*False Acceptance Rate*) por sus siglas en inglés.

2.3.2 Métodos de Verificación de Firmas

En los últimos años se ha realizado una gran cantidad de investigaciones en esta área y se han investigado una gran variedad de enfoques. En la Figura 13, se muestran algunos de los enfoques más relevantes. Cuando se consideran las técnicas de *Template Matching*, la instancia a ser comprobada es comparada contra plantillas de las firmas genuinas o falsas. Este es uno de los enfoques más simples en el reconocimiento de patrones. El algoritmo de *Dynamic Time Warping* (DTW) es el más utilizado en las técnicas de *Template Matching* en la verificación de firmas offline.

Cuando se habla de enfoques estadísticos, los clasificadores basados en distancia son considerados. Por otra parte, las redes neuronales han sido ampliamente usadas para la verificación de firmas, debido a su capacidad de aprender y generalizar. Recientemente, se ha prestado atención al uso de los Modelos Ocultos de Markov o HMMs por sus siglas en inglés, tanto para verificación de firmas offline como online. Los enfoques sintácticos están relacionados con la representación estructural de las

firmas, la cual es descrita a través de sus elementos principales, llamados también primitivas, y son comparados mediante grafos o árboles [23]. A continuación se detallan los enfoques que influenciaron este trabajo, así como también distintas investigaciones y sistemas desarrollados con estas técnicas.

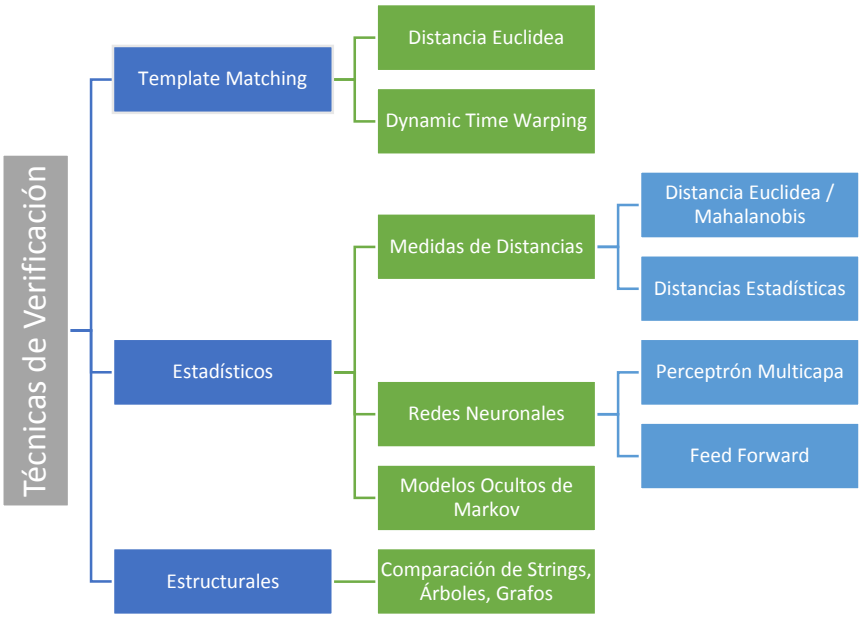

Figura 13: Técnicas de Verificación de Firmas

A. Redes Neuronales

Este enfoque es ampliamente utilizado; las principales razones son su capacidad de aprender y generalizar, la facilidad de uso (dependiendo de la red) y su poder. Cuando se utiliza este enfoque, se obtiene la estructura de la NN extrayendo las características de las firmas y aprendiendo la relación entre la firma y su clase. Baltzakisa y Papamarkos [24] desarrollan una técnica de verificación basada en un clasificador de redes neuronales de dos etapas. Este sistema utilizó características globales y de cuadrículas. Por cada conjunto de características, implementaron una estructura de Perceptron OCON (*one-class-one-network*) de dos etapas. En la primera etapa, el clasificador combina los resultados de la decisión de la NN y de la Distancia Euclídea utilizando tres conjuntos de datos. Los resultados del primer clasificador alimentan una red neuronal RBF (*Radial Basis Function*), la cual toma la decisión final. El rendimiento de este sistema fue evaluado con un subconjunto compuesto por 500 firmas. Este sistema arrojó un 9.81% de FAR y un 3% de FRR con una eficacia total del 90.09%.

Kumar, Raja, Chhoraray y Pattanaik [25] proponen un modelo de verificación de firmas offline basado en la fusión de características globales y de cuadrículas. Este sistema está dividido en tres

fases: Pre procesamiento, extracción de características y verificación. La base de datos de firmas que utilizaron contenía 600 firmas divididas igualmente en genuinas y falsificadas. En la fase de pre procesamiento realizaron una reducción de ruido, normalización de tamaño y adelgazamiento. Luego de pre procesar y extraer las características entre ellas el *Aspect Ratio*, área de la imagen, ancho y alto de la imagen pasaron a la etapa de verificación, aquí utilizaron una NN multicapa *feed-forward*. La NN consiste en 30 variables de entrada y está diseñada para verificar una firma a la vez. Para el entrenamiento utilizan el algoritmo de *Back Propagation*. Este sistema reportó un 7.51% de FRR y un 4.16% de FAR.

Algunos de los modelos de NN que han sido usados en distintas investigaciones son: Redes Bayesianas [26] [27], Perceptrón Multicapa [28] [29] [30], *Back Propagation* NN [31] [29] [32], *Radial Basis Functions* (RBF) o funciones de base radial [29] [33] [34].

B. Template Matching (Comparación de Patrones)

El proceso de comparar patrones es denominado *Template Matching*; estos patrones pueden ser una curva o una imagen. Madasu [35] desarrolló un sistema para la verificación de firmas utilizando modelos difusos. En la fase de extracción de características, la imagen fue dividida en 8 partes, y cada una tenía aproximadamente la misma cantidad de píxeles negros. Cada una de estas partes se dividió en 4 filas y 3 columnas, para luego calcular la suma de los ángulos de todos los puntos en cada parte con respecto a la esquina inferior izquierda, luego esta suma se normalizó con el número de píxeles en cada caja. La comparación se hizo mediante modelos, utilizando 1200 firmas de 40 personas. Este sistema reportó un 21% de FAR para falsificaciones expertas y un 8% de FRR.

Otros enfoques de *Template Matching* pueden usar medidas de similitud [36], comparación de patrones [37] [38], funciones de desplazamiento [39] [40], entre otros.

C. Distancias Simples

Un clasificador de distancias simples modela cada patrón de clase con una función de densidad de probabilidad, PDF por sus siglas en inglés, y luego se apoya en una distancia calculada entre un patrón de prueba y la PDF para realizar la clasificación.

Mahji [2] implementa un método de extracción de características basado en los centros geométricos. Estas características son obtenidas mediante la división de la firma de forma horizontal y vertical por sus centros geométricos. Estos centros, conforman el vector de características. Un modelo de Distancia Euclídea es utilizado para la clasificación en una base de datos que contiene 30 firmas genuinas, 10 falsificaciones aleatorias, 10 falsificaciones simples y 10 expertas, por cada usuario. Los autores alegan

obtener 2.08% de FAR para las falsificaciones aleatorias, 9.75% para las simples y 16.36% para las expertas y un 14.58% de FRR.

Por otra parte, Viriri y Schafer [41] presentaron un sistema de verificación de firmas offline basado en características globales y de transición. En esa investigación se utilizó una base de datos de 2016 firmas. Para entrenar el sistema usaron un subconjunto de la base de datos. Luego, promediaron las características para todas las firmas en la forma de un vector de características cuyos valores era los centroides de las imágenes. Cuando una firma llegaba para ser verificada, se comparaba contra los vectores anteriormente mencionados. En la fase de pruebas, usaron la Distancia Euclídea, si el valor de la firma nueva contra el patrón era menor a un umbral esta era aceptada. Utilizando un umbral global lograron un FAR de 18.5% y usando un umbral local, lograron un FAR de 17.8%.

Capítulo 3 – Diseño e Implementación

Para el sistema desarrollado en este trabajo se implementaron dos métodos de verificación de firmas. Cada uno de estos métodos cuenta con su propia clase dentro del proyecto y su uso se hace por medio de métodos internos a cada clase, estos métodos cumplen con las distintas etapas de un sistema de verificación de firmas. El primer método desarrollado utiliza una serie de dilataciones para la obtención un patrón el cual al ser comparado contra otra firma mide las distintas variaciones que esta posee. Estas variaciones están expresadas en cantidades de píxeles de ciertos colores. Para la verificación se utilizan técnicas de minería de datos. El segundo método desarrollado, hace uso de los Centros Geométricos de la firma y de la Distancia Euclídea para obtener un vector que representen a las firmas y para calcular dos umbrales que serán utilizados en la fase de verificación. A continuación se detalla la implementación de ambos métodos.

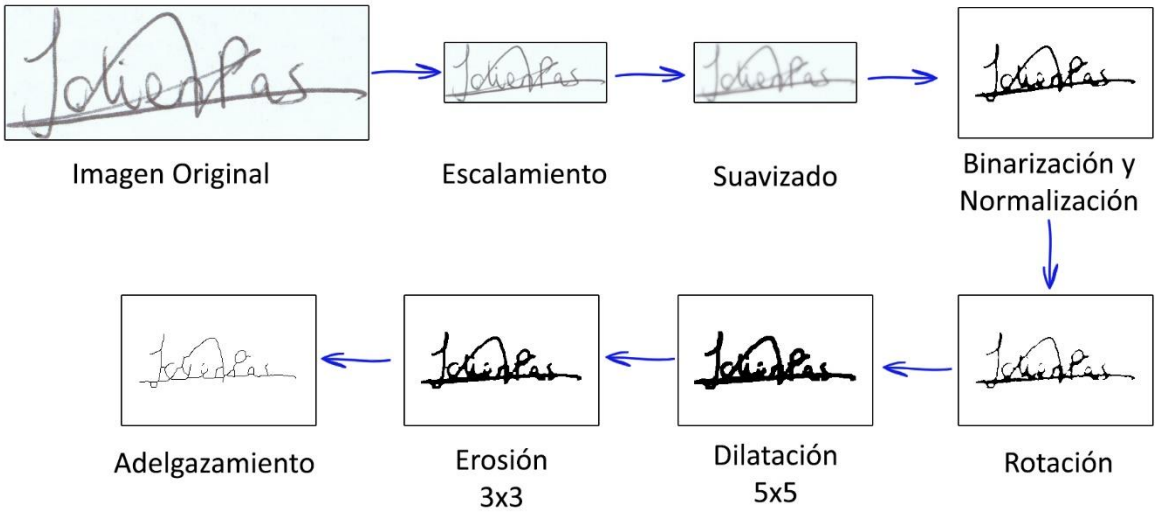

Figura 14: Pasos en el Pre procesamiento de una firma

3.1 Morfología y K-Vecinos

Durante el proceso de firmar, existen distintos factores que pueden afectar el resultado de la firma; dichos factores pueden estar relacionados con la edad, el humor, la situación en la que se firma, etc. Es por esto, que las firmas de una persona no son siempre iguales, existen ciertas variaciones entre ellas. Estas variaciones son llamadas variaciones intrapersonales [1]. Este método busca medir estas variaciones utilizando operaciones morfológicas en distintos niveles y a partir de ahí extraer las características para luego utilizar k-vecinos como método de clasificación. A continuación se explican los distintos pasos de este método:

3.1.1 Pre procesamiento

La etapa de pre procesamiento es común en su mayoría para ambos métodos. Los algoritmos aplicados a la imagen son los siguientes:

- **Escalamiento:** Esta operación se realiza con el fin de reducir el tamaño de la imagen, manteniendo su *Aspect Ratio*. Se reduce para que el tiempo de procesamiento de los algoritmos siguientes sea menor
- **Suavizado:** Se aplica un filtro de suavizado bilateral, este remueve el ruido y los artefactos mientras preserva los bordes
- **Binarización y Normalización de tamaño:** Se aplica una binarización a la imagen, y se copia a una imagen de tamaño fijo. Esto con la finalidad de que todas las imágenes tengan el mismo tamaño
- **Rotación:** Se calcula el ángulo de inclinación de la firma con respecto a una línea horizontal y se rota tantos grados el ángulo indique
- **Dilatación:** Se realiza una dilatación con un elemento estructurante cuadrado de 5x5 para compensar por la información perdida en el proceso anterior
- **Erosión:** Se aplica una erosión con un elemento estructurante de 3x3 con el fin de eliminar los artefactos que no pertenezcan a la firma o aquellos que son muy pequeños para aportar alguna información
- ***Thinning* o Adelgazamiento:** Se reduce el grosor de la firma a 1 píxel

En la Figura 14 se pueden observar los distintos pasos aplicados a una firma. A continuación se detallará cada método utilizando el esquema descrito en el Capítulo 2.

3.1.2 Extracción de Características

Las firmas de una misma persona poseen ciertas variaciones pero estas son limitadas. Las falsificaciones poseen variaciones mayores a las que una firma genuina posee. Para clasificar una firma se trata de detectar estas variaciones y para cuantificarlas se utiliza la técnica morfológica desarrollada en este trabajo. Primero, se genera el contorno de la firma, el cual no es más que los bordes externos de esta. Para lograrlo se utilizan diferentes niveles de dilatación, donde los elementos estructurantes (EE) son círculos con radios r_1, r_2, r_3 y r_4. Al aplicar estos EE se obtienen 4 conjuntos dilatados.

Al realizar la dilatación se rellena el contorno generado con un color distinto a cada nivel, estos niveles son manejados por los EE los cuales poseen un radio que cumple con que $r_1 < r_2 < r_3 < r_4$. Como

resultado, se tiene una estructura con 4 bandas de colores, donde cada banda representa la extensión de la variación de cada píxel y por lo tanto los segmentos de la firma. Dentro del sistema, las bandas fueron generadas con los radios r_1=3, r_2=6 r_3=10 y r_4=16 y rellenadas con los colores Negro, Rojo, Verde y Azul respectivamente (Figura 15).

Figura 15: Firma con los distintos niveles de dilatación

Una vez que se generan los patrones de las firmas, se procede a intersectar cada patrón con cada firma disponible por usuario. Para encontrar la intersección de ambas imágenes se evalúa cada banda con la operación XOR u OR exclusivo. Esta operación se realiza con los colores RGB de cada píxel, por ejemplo R (255, 0,0) XOR G (0, 255,0) es igual a (255, 255,0). Cada una de las bandas de colores, representa las variaciones permitidas para un píxel y si este entra dentro del rango de variación su color será exclusivamente R, G, o B o cualquier combinación de ellos.

Los píxeles negros son considerados píxeles sin variación. La operación de XOR genera una nueva imagen la cual contiene los colores que se encuentran en la Tabla 1. En la Figura 16 se observa el resultado de la intersección entre dos firmas genuinas y entre una genuina y una falsificada.

Figura 16: Izquierda: Comparación contra firma genuina. Derecha: Comparación contra firma falsificada

Color	R	G	B
Negro	0	0	0
Rojo	255	0	0
Verde	0	255	0
Azul	0	0	255
Color de Fondo	255	155	159
Color de Fondo 2	0	100	96
Blanco	255	255	255
Cian	0	255	255
Magenta	255	0	255
Amarillo	255	255	0

Tabla 1: Colores posibles de una intersección entre una imagen dilatada y otra normal

El vector de características de este método está compuesto por la cantidad de píxeles de cada color en la Tabla 1 y cuatro variables adicionales. Estas cuatro variables son:

- ***Aspect Ratio*:** Es la proporción entre el ancho y alto de la firma (
- Ecuación 1)

$$AR = \frac{Alto\ de\ la\ imagen}{Ancho\ de\ la\ imagen}$$

Ecuación 1: Aspect Ratio

- ***Density Ratio*:** Para calcularlo se divide la imagen en dos partes iguales de forma vertical y se cuenta la cantidad de píxeles negros de cada lado. Provee información sobre la densidad de las mitades de la firma [42]. Se calcula utilizando la
- Ecuación 2.

$$DR = \frac{Cantidad\ de\ píxeles\ del\ lado\ izquierdo}{Cantidad\ de\ píxeles\ del\ lado\ derecho}$$

Ecuación 2: Density Ratio

- ***Occupancy Ratio*:** Representa la cantidad de píxeles que pertenecen a la firma entre la cantidad de píxeles de la imagen (
- Ecuación 3).

$$OR = \frac{Cantidad\ de\ píxeles\ de\ la\ firma}{Cantidad\ de\ píxeles\ de\ la\ imagen}$$

Ecuación 3: Occupancy Ratio

- **Puntos Críticos:** Las esquinas de la imagen son regiones con una gran variación en su intensidad en todas las direcciones. Estas esquinas son consideradas puntos críticos para

efectos de este trabajo. Se calcula la cantidad de esquinas utilizando el algoritmo de *Harris' Corners.*

De esta manera, el vector está compuesto por 14 valores. Dicho vector es calculado para cada imagen generada por las intersecciones, para luego crear un archivo en formato *arff* el cual contiene cada uno de los vectores y adicionalmente una variable de clase la cual indica si la instancia pertenece a una firma genuina (*True*) o falsificada (*False*). Dado de que no se dispone de la misma cantidad de firmas que genuinas y falsificadas la clase False esta desbalanceada, lo que dificulta la generación de un buen modelo de clasificación. Para resolver esto, se aplica el algoritmo SMOTE (*Synthetic Minority Over-sampling Technique*) [43] (Figura 17), el cual realiza un sobre muestreo de la clase minoritaria para así nivelar la proporción de ésta en el conjunto de datos. Se toman instancias de la clase minoritaria y se generan nuevas instancias "sintéticas", utilizando cierta cantidad de vecinos más cercanos, los cuales se eligen al azar. Estas nuevas instancias se generan de la siguiente manera: Se toma la diferencia entre el vector de características a utilizar y su vecino más cercano, se multiplica esta diferencia por un número aleatorio entre 0 y 1 y se añade al conjunto.

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100) * T$ synthetic minority class samples

1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. $numattrs$ = Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the $nnarray$
15. Populate(N, i, $nnarray$)
16. **endfor**

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)

17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** $numattrs$
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][$
21. Compute: gap = random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (∗ *End of Populate.* ∗)

End of Pseudo-Code.

Figura 17: Pseudocódigo del algoritmo SMOTE

3.1.3 Verificación de las firmas

Una vez obtenido el archivo con todas las instancias, se procede a elegir una técnica de minería de datos, la cual es el algoritmo de K-Vecinos donde K = 1. Aquí, se utilizaron los algoritmos que provee la librería de WEKA (*Waikato Environment for Knowledge Analysis*) [44]. Una vez entrenado el modelo, se realizó la evaluación utilizando *Cross Validation* con 10 particiones.

Al momento de verificar una firma, se le proporcionan al sistema 2 o 3 firmas una o dos originales y la cuestionada. Se aplica el procedimiento descrito anteriormente para obtener los valores del vector y se clasifica utilizando el modelo generado por el algoritmo K-Vecinos. Esto devolverá un valor *True* si es genuina y *False* si es falsificada.

3.2 Distancia Euclídea

Adicionalmente, se implementó un segundo método. Este utiliza algoritmos para la división de la imagen y la Distancia Euclídea para la extracción de características, el cálculo de umbrales de comparación y para la verificación de una nueva firma. A continuación se detallan cada uno de los pasos que comprenden este método.

3.2.1 Pre procesamiento

Los procedimientos utilizados para este método son exactamente iguales a los presentados anteriormente, con la excepción de que se omite el *Thinning* de la firma.

3.2.2 Extracción de Características

En este enfoque las características usadas son de tipo geométrico. Se tienen dos conjuntos con 30 puntos los cuales representan la distribución de píxeles en la imagen. Estos puntos no son más que los Centros Geométricos. Se aplican algoritmos de divisiones verticales y horizontales para obtener estos puntos. El procedimiento para encontrarlos se explica a continuación.

En el algoritmo de divisiones verticales la imagen de la firma se divide con una línea vertical que pasa por el centro geométrico, para así obtener una parte del lado derecho y otra del lado izquierdo. El centro geométrico se calcula localizando un punto donde el número de píxeles negros es la mitad del total de los píxeles en la firma. Luego, se obtienen los centros geométricos v_1 y v_2 de las dos partes. Se procede a dividir el lado izquierdo con una línea horizontal que pasa por v_1 para así obtener los centros v_3 y v_4 correspondientes a la sección superior e inferior del lado izquierdo de la imagen. De igual forma, el lado derecho se divide con una línea horizontal que pasa por v_2 para obtener los centros v_5 y v_6. Cada una de las secciones obtenidas se vuelve a dividir a través de sus respectivos centros geométricos hasta obtener la cantidad de puntos necesarios.

En el caso de las divisiones horizontales el procedimiento es exactamente igual al descrito anteriormente, solo que en vez de empezar a dividir de forma vertical se empieza con una línea horizontal, para luego obtener los centros geométricos de la parte superior e inferior de la imagen h_1 y h_2. Luego se divide la sección superior de forma vertical para obtener los centros h_3 y h_4 correspondientes al lado izquierdo y derecho de la parte superior de la firma. En la Figura 18 se pueden observar las distintas divisiones tanto horizontales como verticales.

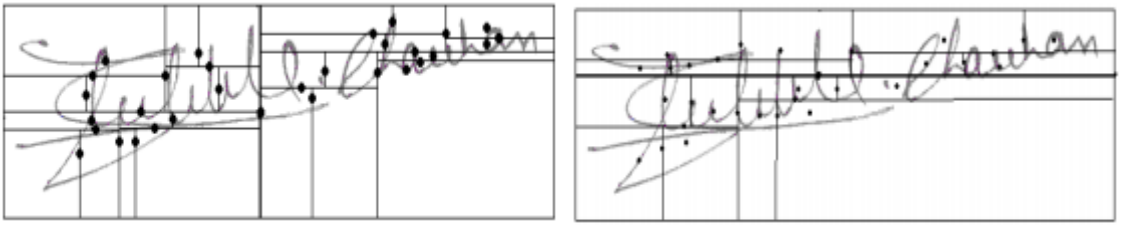

Figura 18: Izquierda: Divisiones Verticales. Derecha: Divisiones Horizontales

Luego de que se obtienen los 60 puntos (30 verticales y 30 horizontales) se procede a calcular un umbral que represente a las firmas de cada usuario. Siendo *n* la cantidad de firmas para el entrenamiento y x_1, x_2,…, x_n los puntos individuales de estas firmas, se toma un punto de cada firma y se calcula la mediana de estos puntos ($x_{mediana}$). Después, se calculan las distancias entre cada uno de los puntos y $x_{mediana}$ para así obtener *n* distancias [2]. Esto se repite para los 60 puntos de las *n* firmas.

$$d1 = distance(x_{mediana}, x_1)$$

$$d2 = distance(x_{mediana}, x_2)$$

$$\dots$$

$$dn = distance(x_{mediana}, x_n)$$

Ecuación 4: Cálculo de distancias

Para el cálculo del umbral se utilizan dos parámetros, d_{avg} y σ (desviación estándar). En la Ecuación 5 se muestra el cálculo de estos parámetros.

$$d_{avg} = promedio(d_1, d_2, \dots, d_n)$$

$$\sigma = \mathrm{SD}\ (d_1, d_2, \dots, d_n)$$

Ecuación 5: Cálculo de la distancia promedio y la desviación estándar

Los parámetros anteriormente mencionados se calculan para cada punto, es decir, se tendrán 30 valores de d_{avg} y 30 de σ. Una vez obtenidos, se procede a calcular el umbral mediante la fórmula de la Ecuación 6.

$$umbral(t) = \sqrt{\sum_{i=1}^{30} (d_{avg,i} + \sigma_i)^2}$$

Ecuación 6: Cálculo del umbral

Para el entrenamiento, se tomaron *n* firmas de cada persona. Teniendo 60 puntos por cada firma genuina, de los cuales 30 son obtenidas mediante divisiones horizontales y 30 mediante divisiones verticales, se procede a calcular los umbrales verticales y horizontales (Ecuación 7).

$$umbralV = \sqrt{\sum_{i=1}^{30}(vd_{avg,i} + v\sigma_i)^2}$$

$$umbralH = \sqrt{\sum_{i=1}^{30}(hd_{avg,i} + h\sigma_i)^2}$$

Ecuación 7: Cálculo del umbral horizontal y vertical

Adicionalmente, se calculan dos conjuntos de puntos que representaran a las *n* firmas de un usuario (Ecuación 8), donde $v_{i,1}$, $v_{i,2}$, … , $v_{i,30}$ son los puntos obtenidos de la firma *i* mediante divisiones verticales y $h_{i,1}$, $h_{i,2}$, … , $h_{i,30}$ son los puntos obtenidos de la firma *i* mediante divisiones horizontales. Estos valores son almacenados para ser usados en el próximo paso.

$$v_{pattern,1} = mediana(v_{1,1}, v_{2,1}, \dots, v_{n,1})$$
$$v_{pattern,2} = mediana(v_{1,2}, v_{2,2}, \dots, v_{n,2})$$
$$\dots$$
$$v_{pattern,30} = mediana(v_{1,30}, v_{2,30}, \dots, v_{n,30})$$

$$h_{pattern,1} = mediana(h_{1,1}, h_{2,1}, \dots, h_{n,1})$$
$$h_{pattern,2} = mediana(h_{1,2}, h_{2,2}, \dots, h_{n,2})$$
$$\dots$$
$$h_{pattern,30} = mediana(h_{1,30}, h_{2,30}, \dots, h_{n,30})$$

Ecuación 8: Cálculo de los patrones que representan a las firmas de un usuario

Cada uno de estos pasos se realiza para cada usuario en el conjunto de datos. Al final, por usuario se tienen 60 puntos que representan sus firmas y dos umbrales, uno horizontal y uno vertical.

3.2.3 Verificación de las firmas

Cuando una firma necesita ser verificada, hay que calcular los puntos característicos horizontales y verticales. Los puntos verticales son ahora $v_{new,1}$, $v_{new,2}$, … , $v_{new,30}$ y los horizontales, $h_{new,1}$, $h_{new,2}$, … , $h_{new,30}$. Luego, se calcula la Distancia Euclídea entre los nuevos puntos y los patrones almacenados (Ecuación 9).

$$vd_{new,1} = distancia(v_{pattern,1}, v_{new,1})$$
$$vd_{new,2} = distancia(v_{pattern,2}, v_{new,2})$$
$$\dots$$
$$vd_{new,30} = distancia(v_{pattern,30}, v_{new,30})$$

Ecuación 9: Cálculo de las nuevas distancias

$$hd_{new,1} = distancia(h_{pattern,1}, h_{new,1})$$
$$hd_{new,2} = distancia(h_{pattern,2}, h_{new,2})$$
$$\dots$$
$$hd_{new,30} = distancia(h_{pattern,30}, h_{new,30})$$

Con esto, se calcula dos valores los cuales serán comparados contra los umbrales obtenidos anteriormente usando la Ecuación 10 y la Ecuación 11.

$$v_m = \sqrt{\sum_{i=1}^{30} vd^2{}_{new,i}}$$

Ecuación 10: Magnitud de las distancias verticales

$$h_m = \sqrt{\sum_{i=1}^{30} hd^2{}_{new,i}}$$

Ecuación 11: Magnitud de las distancias horizontales

Si v_m es menor o igual a *umbralV* la firma nueva es aceptada por divisiones verticales. Adicionalmente si h_m es menor o igual a *umbralH* entonces la firma es aceptada por divisiones horizontales. Para que la nueva firma sea clasificada como genuina, esta tiene satisfacer ambas condiciones.

3.3 Detalles de Implementación

Se desarrollaron y utilizaron distintas clases y métodos, entre ellas está la clase *LockBitmap*. Las operaciones de *GetPixel* y *SetPixel* de C# son computacionalmente costosas, lo que aumenta el tiempo de procesamiento. En este proyecto estos métodos son sustituidos por los de la clase *LockBitmap*, la cual se encarga de bloquear y desbloquear los datos del *Bitmap* para agilizar el procesamiento de la imagen.

Una de las clases más importantes es *PDIMethods*, esta contiene todas las operaciones que alteran las imágenes de las firmas, aquí se encuentran los algoritmos que usan los métodos desarrollados para el pre procesamiento de las imágenes. Los métodos que destacan en esta clase son:

- Función *scaleImageMaintainingAspectRatio* (Algoritmo 1). Dada una imagen la escala dentro de un tamaño especificado sin alterar su *Aspect Ratio*. Los valores *newWidth* y *newHeight* constituyen el tamaño deseado. Primero, se calcula el *Aspect Ratio* con respecto a los valores dados y el nuevo tamaño se obtiene multiplicando el ancho y alto de la imagen por el *Aspect ratio* calculado. Retorna la imagen escalada.
- Se creó también una función que se encarga de colocar la firma ya escalada dentro de una imagen más grande.
- Función *rotateSignature* (Algoritmo 2), toma una imagen que contiene una firma, calcula su *bounding box* y rota la firma en un ángulo determinado. Esto genera una nueva imagen de tamaño variable dependiendo de la firma. Esta nueva imagen es copiada a una de tamaño más grande para asegurar que todas las imágenes posean el mismo tamaño.
- Función *Thin_ZhangSuen* (Algoritmo 3). Este algoritmo se encarga de realizar el adelgazamiento de una imagen binaria. Consiste en remover los puntos del contorno de la imagen excepto aquellos que pertenecen al esqueleto. Para poder preservar la conectividad de esqueleto se divide cada iteración en dos sub-iteraciones. En la primera el punto P_i perteneciente al contorno se elimina si satisface ciertas condiciones, si alguna no se cumple entonces P_i no es eliminado. En la segunda iteración, se cambian dos condiciones de la primera y el resto queda igual [45]. De esta forma se van eliminando los puntos que no pertenecen al esqueleto.
- También, se creó una función que a partir de una imagen realiza una erosión con un elemento estructurante de tamaño 3x3, seguida de una dilatación con un elemento estructurante de 5x5.

```
1  public static  Bitmap scaleImageMaintainingAspectRatio(Bitmap img, int newWidth, int newHeight){
2
3       float aspectW = newWidth / (float)img.Width;
4       float aspectH = newHeight / (float)img.Height;
5       int newW, newH;
6
7       float aspect = Math.Min (aspectW, aspectH);
8
9       newW = (int)(img.Width * aspect);
10      newH = (int)(img.Height * aspect);
11
12      ResizeBicubic filter = new ResizeBicubic (newW, newH);
13      Bitmap res = filter.Apply (img);
14
15      return res;
16 }
```

Algoritmo 1: Función para escalar una imagen manteniendo su Aspect Ratio

```
public static Bitmap rotateSignature(Bitmap img, float angle)
     {
     Bitmap peq;
     Bitmap r;
     BoundingBox boundingB = new BoundingBox (0, 0, 0, 0);
     getImageBoundingBox (img, ref boundingB);

     Bitmap b2O = new Bitmap (boundingB.Right - boundingB.Left, boundingB.Down - boundingB.Up);
     Bitmap b3O = new Bitmap (img.Width, img.Height);

     LockBitmap b2 = new LockBitmap (b2O);

     b2.LockBits ();

     using (Graphics grp = Graphics.FromImage (b3O))
     {
          grp.FillRectangle (
          Brushes.White, 0, 0, img.Width, img.Height);

     }


     for (int x = boundingB.Left; x < boundingB.Right; x++)
     {
         for (int y = boundingB.Up; y < boundingB.Down; y++)
         {
             b2.SetPixel (x - boundingB.Left, y - boundingB.Up, img.GetPixel (x, y));
         }
     }

     b2.UnlockBits ();

     //Rotar
     r = rotateBitmap (b2O, angle);
     LockBitmap b3 = new LockBitmap (b3O);
     b3.LockBits ();

     //Obtener Bounding Box nuevo
     peq = getImageBoundingBox (r, ref boundingB);

     //Copiar la imagen rotada a una del tamaño estándar
     for (int x = boundingB.Left; x < boundingB.Right; x++)
     {
         for (int y = boundingB.Up; y < boundingB.Down; y++)
         {
             if (r.GetPixel (x, y).R == 0)
             {
                 b3.SetPixel (x - boundingB.Left, y - boundingB.Up, r.GetPixel (x, y));
             }
         }
     }

     b3.UnlockBits ();

     //Centrar
     peq = getImageBoundingBox (b3O);
     b3O = centerSignature (b3O, peq);

     return b3O;
}
```

Algoritmo 2: Función para rotar una imagen

```
public static Bitmap Thin_ZhangSuen(Bitmap imgOr)
{
      int[,] nbrs = new int[,] { { 0, -1 }, { 1, -1 }, { 1, 0 }, { 1, 1 }, { 0, 1 }, { -1, 1 }, { -1, 0 }, { -1, -1 }, { 0, -1 } };
      int[, ,] nbrGroups = new int[,,] { { { 0, 2, 4 }, { 2, 4, 6 } }, { { 0, 2, 6 }, { 0, 4, 6 } } };

      bool firstStep = false;
      bool hasChanged;
      List<Point> toWhite = new List<Point> ();
      Bitmap imgo = new Bitmap (imgOr);
      LockBitmap img = new LockBitmap (imgo);

      img.LockBits ();
      do
      {
          hasChanged = false;
          firstStep = !firstStep;

          for (int r = 1; r < img.Width - 1; r++)
          {
              for (int c = 1; c < img.Height - 1; c++)
              {
                  if (img.GetPixel (r, c).R != 0)
                      continue;

                  int nn = 0;
                  for (int i = 0; i < 8; i++)
                      if (img.GetPixel (r + nbrs[i, 1], c + nbrs[i, 0]).R == 0)
                          nn++;

                  if (nn < 2 || nn > 6)
                      continue;

                  //Número de transiciones
                  int count = 0;
                  for (int i = 0; i < 8; i++)
                      if (img.GetPixel (r + nbrs[i, 1], c + nbrs[i, 0]).R != 0)
                          if (img.GetPixel (r + nbrs[i + 1, 1], c + nbrs[i + 1, 0]).R == 0)
                              count++;
                  if (count != 1)
                      continue:

                  //Al menos uno es blanco
                  count = 0;
                  int step = firstStep ? 0 : 1;
                  for (int i = 0; i < 2; i++)
                  {
                      for (int j = 0; j < 3; j++)
                      {
                          unsafe
                          {
                             if (img.GetPixel (r + nbrs[nbrGroups[step, i, j], 1], c + nbrs[nbrGroups[step, i, j], 0]).R != 0)
                              {
                                  count++;
                                  break;
                              }
                          }
                      }
                  }
                  if (!(count > 1))
                      continue;

                  toWhite.Add (new Point (r, c));
                  hasChanged = true;
              }
          }
          foreach (Point p in toWhite)
          {
              img.SetPixel (p.X, p.Y, Color.FromArgb (255, 255, 255));
          }
          toWhite.Clear ();

      } while (hasChanged || firstStep);

      img.UnlockBits ();
      return imgo;
}
```

Algoritmo 3: Algoritmo de adelgazamiento Zhang-Suen

Otra clase implementada que es usada por ambos métodos es la clase *FeatureExtraction*. En esta se encuentran todos los métodos que obtienen información de la imagen de una firma. Los algoritmos que destacan en esta clase se explicarán más adelante junto a sus respectivos métodos. Adicionalmente, se tiene una clase llamada *WekaClass*, la cual hace uso de la librería de Weka para la evaluación de las instancias y la creación del modelo usando K-Vecinos. En esta clase se tienen:

- La función *kNeighbors*, lee un archivo *arff* con las instancias para el entrenamiento, crea el clasificador de K-Vecinos, lo entrena, realiza la evaluación del clasificador y por último lo retorna.
- La función *evalInstance*, dado un *string* con las características de la firma, evalúa la misma utilizando el clasificador creado anteriormente.
- También se creó una función, la cual a partir de un conjunto de instancias aplica el filtro de SMOTE con un porcentaje especificado por parámetros.

Cada método posee su clase particular, la cual se encarga de mezclar y utilizar todas las funciones de las clases anteriormente descritas para la el entrenamiento y clasificación. El primer método es el de Morfología y K-Vecinos. Esta clase cuenta con los atributos mostrados en el Algoritmo 4 y los métodos involucrados son:

- La acción *preProcessing*, utiliza los métodos de la clase *PDIMethods* para pre procesar las imágenes de las firmas. Los pasos son los descritos anteriormente y cada uno de los resultados se guarda si así lo desea el usuario. Aquí, la imagen que se guarda por defecto es la de la variable *artifacts* dado que se necesita para futuras operaciones. No obstante esto puede ser cambiado por medio de la interfaz.
- La función *dilatacionMatch* (Algoritmo 5) perteneciente a la clase *PDIMethods*, la cual dada una imagen dilatada con las cuatro bandas de colores (Figura 15) y otra imagen binarizada, aplica la operación de XOR por cada píxel para obtener una imagen como la de la Figura 16.
- Las funciones *countPixels* y *getFeatureVector*, calculan las características de la imagen de una firma. Estas características son las mencionadas en la página 36. Estos métodos pertenecen a la clase *FeatureExtraction*.

Todas estas funciones se aplican dentro de ciclos, los cuales leen cada una de las firmas del conjunto de datos y realizan ya sea el pre procesamiento de las imágenes o la dilatación de las imágenes

y la comparación entre ellas. Adicionalmente en esta clase se generan los archivos que se utilizan para el entrenamiento del algoritmo k-Vecinos.

Luego, para el segundo método se implementó la clase *EuclideanSplitting*. En esta, al igual que en la anterior se mezclan los métodos de las clases explicadas anteriormente y se procesan las imágenes con el fin de obtener sus características y generar los respectivos archivos. Aquí se utilizan las siguientes funciones de la clase FeatureExtraction:

- *geometricCenter* calcula el centro geométrico de la imagen, en la sección delimitada por los parámetros *tl* y *bl* correspondientes a la esquina superior derecha y la inferior izquierda. Hace uso de la función *blackPixels*, la cual devuelve la cantidad de píxeles negros en dicha sección (Algoritmo 6).
- Los métodos *horizontalS* y *verticalS* hacen uso de la función anterior para calcular las divisiones horizontales y verticales de la imagen (Algoritmo 7). Aquí, se tiene una cola la cual maneja las secciones de la imagen que serán divididas, en cada iteración se saca un elemento de la cola correspondiente a un par de esquinas y se calcula el centro geométrico de esta parte. Luego, el corte que será utilizado ya sea horizontal o vertical, se determina dependiendo del número de iteración en el que se esté. Se encolan las dos secciones resultantes de este paso y se repite hasta que la cantidad de puntos ha sido alcanzada. En la etapa de entrenamiento estas funciones son llamadas por cada firma de la imagen y los puntos son almacenados para el cálculo de los umbrales.

```
1   //Medianas de los puntos Horizontales (Xmedian)
2   private List<Point> pMedianLH = new List<Point> ();
3   //Medianas de los puntos Verticales (Xmedian)
4   private List<Point> pMedianLV = new List<Point> ();
5
6   //Puntos Horizontales de cada una de las firmas de un usuario
7   private List<List<Point>> horizontalPoints = new List<List<Point>> ();
8   //Puntos Verticales de cada una de las firmas de un usuario
9   private List<List<Point>> verticalPoints = new List<List<Point>> ();
10
11  //Promedio de los puntos Horizontales
12  private List<Point> pAverageH = new List<Point> ();
13  //Promedio de los puntos Verticales
14  private List<Point> pAverageV = new List<Point> ();
15
16  //Distancia promedio entre los puntos Horizontales
17  private List<double> dAverageH = new List<double> ();
18  //Distancia promedio entre los puntos Verticales
19  private List<double> dAverageV = new List<double> ();
20
21  //Distancias entre los puntos Horizontales y las Xmedian
22  private List<List<double>> distancesH = new List<List<double>> ();
23  //Distancias entre los puntos Verticales y las Xmedian
24  private List<List<double>> distancesV = new List<List<double>> ();
25
26  //Desviación estándar de los puntos Horizontales y las Xmedian
27  private List<double> dStandardH = new List<double> ();
28  //Distancias entre los puntos Verticales y las Xmedian
29  private List<double> dStandardV = new List<double> ();
30
31  //Puntos Horizontales que representan a las firmas de un usuario
32  private List<Point> hPattern = new List<Point> ();
33  //Puntos Verticales que representan a las firmas de un usuario
34  private List<Point> vPattern = new List<Point> ();
35
36  //Puntos Horizontales que representan las las firmas de todos los usuarios
37  private List<List<Point>> hPatternTotal = new List<List<Point>> ();
38  //Puntos Verticales que representan las las firmas de todos los usuarios
39  private List<List<Point>> vPatternTotal = new List<List<Point>> ();
40  //Todos los umbrales de todos los usuarios
41  private List<double[]> thresholdsTotal = new List<double[]> ();
42
43  //Los puntos Vericales de todas las firmas de todos los usuarios
44  public List<List<List<Point>>> allVerticals = new List<List<List<Point>>> ();
45  //Los puntos Horizontales de todas las firmas de todos los usuarios
46  public List<List<List<Point>>> allHorizontals = new List<List<List<Point>>> ();
```

Algoritmo 4: Atributos de la clase FeatureExtraction

```
public Bitmap dilatacionMatch(Bitmap img1, Bitmap img2)
{
     Bitmap res = new Bitmap (img1.Width, img1.Height);

     for (int x = 0; x < img1.Width; x++)
     {
         for (int y = 0; y < img1.Height; y++)
         {
             Color c1, c2;
             int R, G, B;
             c1 = img1.GetPixel (x, y);
             c2 = img2.GetPixel (x, y);
             //EX-XOR
             R = (int)(c1.R ^ c2.R);
             G = (int)(c1.G ^ c2.G);
             B = (int)(c1.B ^ c2.B);

             res.SetPixel (x, y, Color.FromArgb (R, G, B));
         }
     }

     return res;
}
```

Algoritmo 5: Comparación entre dos imágenes para la futura obtención de características

```
public Point geometricCenter(Bitmap img, Point tl, Point br)
{
     Point gc = new Point ();
     int[] limits = { tl.X, br.X, tl.Y, br.Y };
     int half = blackPixels (img, limits) / 2;
     int count = 0;

     Bitmap img2 = (Bitmap)img.Clone ();
     for (int x = limits[0]; x < limits[1]; x++)
     {
         for (int y = limits[2]; y < limits[3]; y++)
         {
             Color color = img2.GetPixel (x, y);
             if (color.ToArgb () == Color.Black.ToArgb ())
             {
                 count++;
                 if (count == half)
                 {
                     gc.X = x;
                     break;
                 }
             }
         }
     }

     count = 0;
     for (int y = limits[2]; y < limits[3]; y++)
     {
         for (int x = limits[0]; x < limits[1]; x++)
         {
             Color color = img2.GetPixel (x, y);
             if (color.ToArgb () == Color.Black.ToArgb ())
             {
                 count++;
                 if (count == half)
                 {
                     gc.Y = y;
                     break;
                 }
             }
         }
     }
     return gc;
}
```

Algoritmo 6: Cálculo del Centro Geométrico

```
public void horizontalS(Bitmap img, int cant)
{
      Queue<Corner> dataProcess = new Queue<Corner> ();
      Queue<Corner> temp = new Queue<Corner> ();
      List<Point> hP = new List<Point> ();
      Corner c = new Corner ();

      c.p1.X = 0;
      c.p1.Y = 0;
      c.p2.X = img.Width;
      c.p2.Y = img.Height;
      dataProcess.Enqueue (c);
      int cantP = cant;
      int hv = 0;
      while (cantP != 0)
      {

          while (dataProcess.Count > 0)
          {
              Corner c1 = new Corner ();
              Corner c2 = new Corner ();
              Point p;
              Corner a = dataProcess.Dequeue ();
              p = geometricCenter (img, a.p1, a.p2);
              hP.Add (p);
              if (hv % 2 == 0)
              {
                  //Horizontal
                  c1.p1 = a.p1;
                  c1.p2.X = a.p2.X;
                  c1.p2.Y = p.Y;

                  c2.p1.X = a.p1.X;
                  c2.p1.Y = p.Y;
                  c2.p2 = a.p2;
              }
              else
              {
                  //Vertical
                  c1.p1 = a.p1;
                  c1.p2.X = p.X;
                  c1.p2.Y = a.p2.Y;

                  c2.p1.X = p.X;
                  c2.p1.Y = a.p1.Y;
                  c2.p2 = a.p2;
              }

              temp.Enqueue (c1);
              temp.Enqueue (c2);
          }

          foreach (Corner d in temp)
          {
              dataProcess.Enqueue (d);
          }
          temp = new Queue<Corner> ();

          hv++; cantP--;
      }
      hP.RemoveAt (0);
      HorizontalPoints.Add (hP);
}
```

```
public void verticalS(Bitmap img, int cant)
{
    Queue<Corner> dataProcess = new Queue<Corner> ();
    Queue<Corner> temp = new Queue<Corner> ();
    List<Point> hP = new List<Point> ();
    Corner c = new Corner ();

    c.p1.X = 0;
    c.p1.Y = 0;
    c.p2.X = img.Width;
    c.p2.Y = img.Height;
    dataProcess.Enqueue (c);
    int cantP = cant;
    int hv = 0;
    while (cantP != 0)
    {
        while (dataProcess.Count > 0)
        {
            Corner c1 = new Corner ();
            Corner c2 = new Corner ();
            Point p;
            Corner a = dataProcess.Dequeue ();
            p = geometricCenter (img, a.p1, a.p2);
            hP.Add (p);
            if (hv % 2 == 0)
            {
                //Vertical
                c1.p1 = a.p1;
                c1.p2.X = p.X;
                c1.p2.Y = a.p2.Y;
                c2.p1.X = p.X;
                c2.p1.Y = a.p1.Y;
                c2.p2 = a.p2;
            }
            else
            {
                //Horizontal
                c1.p1 = a.p1;
                c1.p2.X = a.p2.X;
                c1.p2.Y = p.Y;
                c2.p1.X = a.p1.X;
                c2.p1.Y = p.Y;
                c2.p2 = a.p2;
            }
            temp.Enqueue (c1); temp.Enqueue (c2);
        }
        foreach (Corner d in temp)
        {
            dataProcess.Enqueue (d);
        }
        temp = new Queue<Corner> ();

        hv++; cantP--;
    }
    hP.RemoveAt (0);
    VerticalPoints.Add (hP);
}
```

Algoritmo 7: Algoritmos que realizan las divisiones verticales y horizontales

3.4 Diseño de la Interfaz

La aplicación se denominó Sistema de Verificación de Firmas Offline. Se crearon pantallas para cada método, donde se puede cambiar el modelo si se dispone de un nuevo conjunto de datos y adicionalmente tiene una pestaña de verificación. Las distintas interfaces se detallan a continuación:

Cuando se inicia la aplicación se tienen dos opciones principales, ir a la pantalla del módulo de Distancia Euclídea o al de Morfología y K-Vecinos (Figura 19).

Figura 19: Pantalla de inicio de la aplicación

Si se selecciona la opción de Morfología y K-Vecinos se despliega la pantalla de la Figura 20, aquí se tienen dos pestañas. En la primera, llamada Pre Procesamiento divida en dos partes, del lado izquierdo se puede cargar un nuevo conjunto de datos y pre procesarlo, adicionalmente del lado derecho se puede cargar una carpeta donde se encuentren las imágenes dilatadas y otra donde se encuentren las imágenes binarizadas y generar un nuevo archivo a partir de esas imágenes. Por defecto tienen una carpeta asignada pero esta puede ser cambiada, al igual que si realiza el procesamiento del conjunto de datos estos directorios se actualizan para utilizar los generados en ese paso. Se tienen 3 opciones, donde se pueden configurar la cantidad de usuarios que se desean utilizar para generar el archivo, así como también la cantidad de firmas genuinas y falsas. Por último, se puede asignar un nombre al archivo donde serán generados los datos.

En la pestaña de Verificación (Figura 20), se cuenta con la opción para verificar una firma. Para esto se deben cargar 2 o 3 imágenes, donde la primera que se seleccionó es la firma cuestionada y el resto son firmas genuinas con las cuales se hará la comparación para obtener las características y así realizar la verificación. Si se cargan 3 imágenes se podrá validar la firma contra cualquiera de las dos por separado dependiendo de la que este seleccionada en el momento.

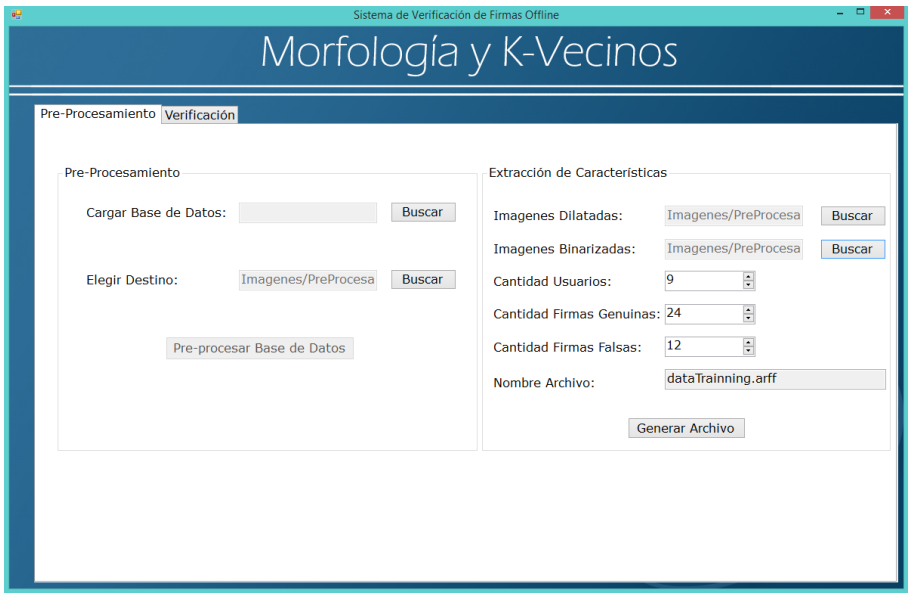

Figura 20: Pantalla de Pre procesamiento del método de Morfología y K-Vecinos

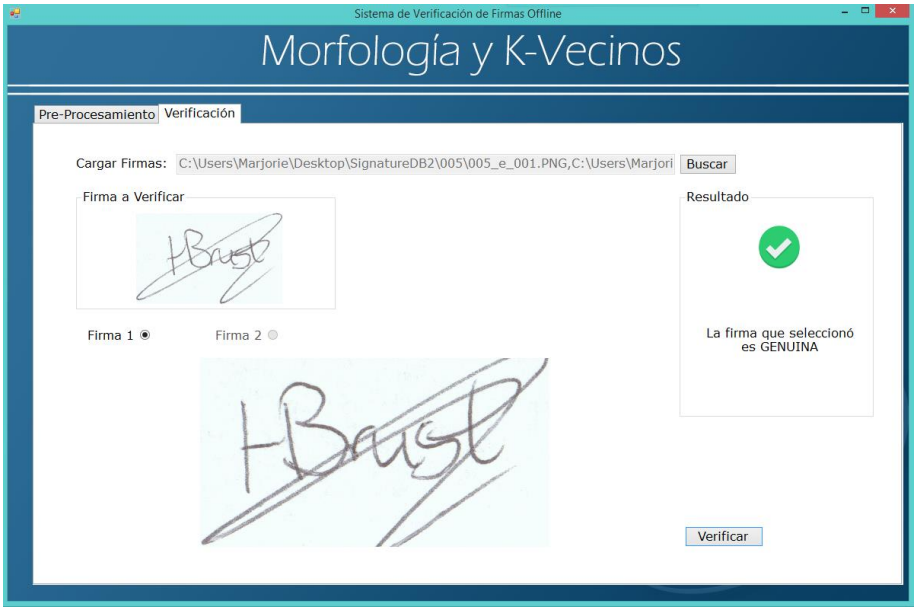

Figura 21: Pantalla de verificación del método de Morfología y K-Vecinos

Si se elige la opción de la Distancia Euclídea en la pantalla principal, se muestra la interfaz divida en dos secciones, el lado izquierdo para el Pre procesamiento del conjunto de datos en caso de que se quiera cambiar. En esta parte se puede elegir el conjunto de datos a procesar y el directorio de destino donde se guardarán las imágenes. Del lado derecho, se puede seleccionar un directorio que contenga

imágenes procesadas para realizar el proceso de entrenamiento, y se tiene la posibilidad de colocar el nombre del archivo a generar.

En la pestaña de Verificación, se tiene la opción de cargar una firma para que sea verificada. El nombre de la imagen debe ser en el formato especificado anteriormente. Una vez que se le da clic al botón verificar se realiza la verificación y el resultado se muestra en el lado derecho de la pantalla (Figura 23).

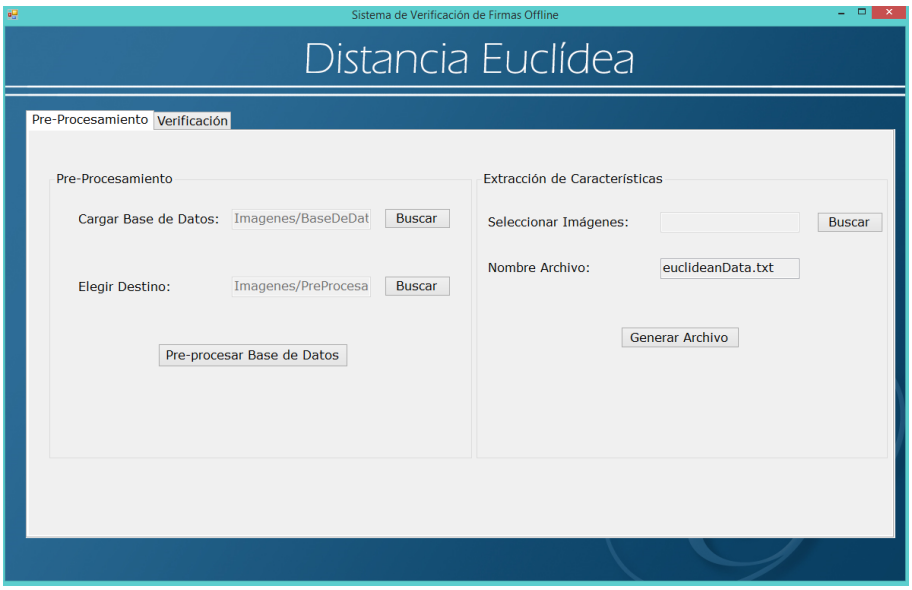

Figura 22: Pantalla para el Pre procesamiento del método de Distancia Euclídea

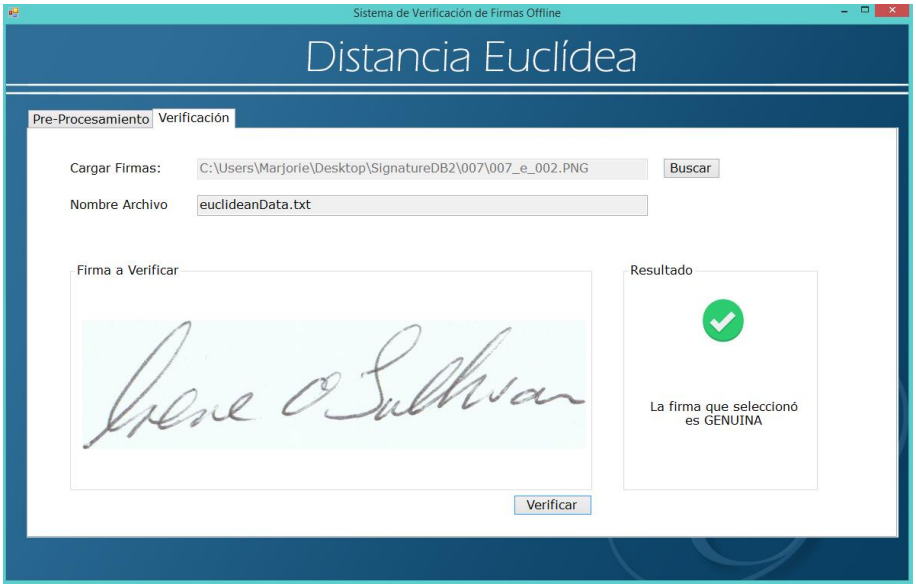

Figura 23: Pantalla para verificar una firma utilizando el método de Distancia Euclídea

La tercera opción es la pantalla principal es la de Configuración (Figura 24), aquí se pueden cambiar los parámetros del conjunto de datos y se pueden elegir las carpetas que se quieren crear a la hora de realizar algún pre procesamiento. Por defecto están marcadas la carpeta de Artefactos y Dilatación, las cuales son necesarias para el funcionamiento de ambos métodos anteriormente descritos.

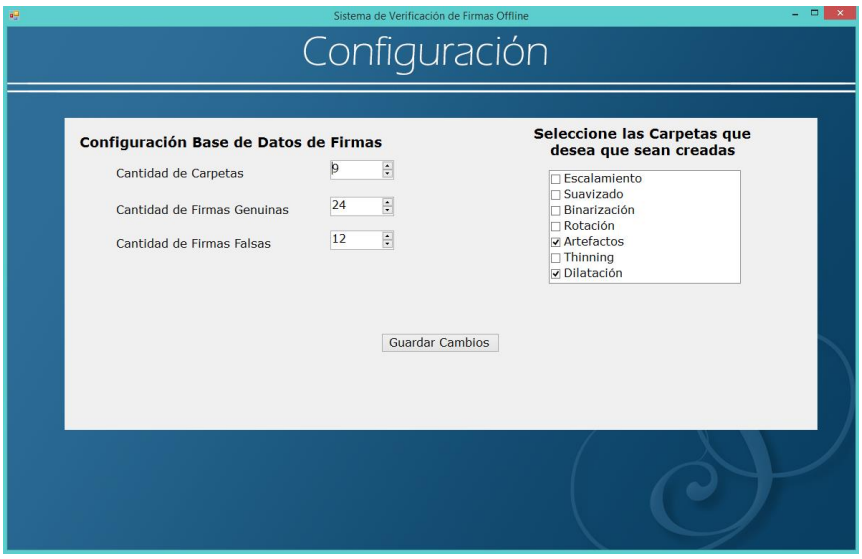

Figura 24: Pantalla de Configuración

Los dos iconos que se encuentran en la parte inferior de la pantalla de inicio se refieren a la pantalla de Información y la de Ayuda respectivamente. La pantalla de información (Figura 25) contiene el nombre del desarrollador y tutor. Por último, en la pantalla de Ayuda (Figura 26) se pueden ver las funcionalidades de cada una de las interfaces mostradas anteriormente.

Figura 25: Pantalla de Información

Figura 26: Pantalla de Ayuda

Capítulo 4 – Pruebas y Resultados

Las pruebas realizadas fueron específicas por cada método pero las medidas de error utilizadas son comunes, para así poder comparar ambos enfoques de una mejor forma. Estas medidas son, tasa de falsos positivos (FAR) y tasa de falsos negativos (FRR).

El conjunto de datos con el cual se realizaron las pruebas contiene firmas de 9 individuos alemanes donde cada persona cuenta con 24 firmas genuinas y 12 falsas, con un total de 324 firmas. Las imágenes de las firmas deben cumplir con un formato para la organización y los nombres de los archivos. Existe una carpeta por cada usuario, estas son nombradas en orden empezando desde 001, dentro de cada carpeta las imágenes de las firmas genuinas deben llamarse *idUsuario_e_idFirma* y las falsificaciones *id_Usuario_h_idFirma*, donde el *idUsuario* es el nombre de la carpeta y el *idFirma* es un número de tres dígitos que empieza en 001. En la Figura 27 se pueden observar algunas de las firmas pertenecientes al conjunto de datos.

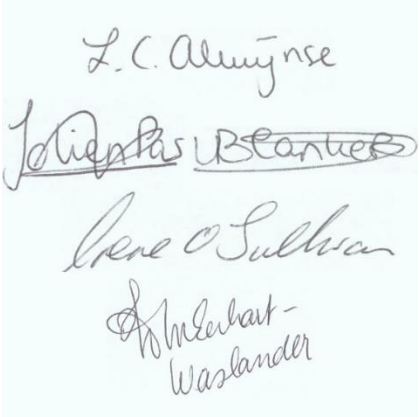

Figura 27: Firmas pertenecientes al conjunto de datos

En el primer método, se realizaron tanto pruebas con el conjunto de datos como con distintos clasificadores. Para la obtener las instancias que conforman a los conjuntos de datos, se compara cada uno de los patrones con el resto de las firmas binarizadas, es decir, la firma 001 del usuario 001 se compara con las otras 23 de ese usuario y así sucesivamente tanto para las genuinas como para las falsas.

Primero, se tomó un conjunto de datos (C1) con 10 atributos y la clase la cual puede ser *True* o *False*. Estos atributos representan la cantidad de píxeles de cada color en las imágenes resultantes al hacer XOR entre una imagen dilatada y una binarizada (Figura 16). El conjunto de datos utilizado consta de 9 usuarios con 24 firmas genuinas y 12 falsificadas cada uno. De las 24 firmas se utilizaron 18 y de las 12 se

tomaron 9 para crear los conjuntos descritos a continuación. El conjunto C1 contiene 4617 instancias donde 3726 son *True* y 891 *False*.

Dado que solo se tienen 9 firmas falsificadas por persona, la clase *False* esta desbalanceada, por lo que se aplicó el algoritmo de SMOTE al conjunto 1 aumentando la clase minoritaria (*False*) en un 100% y un 150%, obteniendo dos conjuntos nuevos. El conjunto C2, con 5508 instancias donde 3726 son *True* y 1782 *False* y C3, con 5953 instancias donde 3726 son *True* y 2227 son *False*. Se utilizaron diferentes técnicas de minería de datos tales como *Random Forest* perteneciente a las técnicas basadas en arboles de decisión, K-Vecinos y una red neuronal *Multilayer Perceptron*. Los modelos generados se evaluaron utilizando validación cruzada con 10 particiones.

Conjunto	Cantidad de Instancias	Algoritmo de Clasificación	FAR	FRR
C1	4617	Random Forest	42.5%	4.02%
C1	4617	Multilayer Perceptron	55%	3.6%
C1	4617	K-Vecinos	28.39%	3.99%
C2	5508	Random Forest	20.8%	6.1%
C2	5508	Multilayer Perceptron	39.6%	9.7%
C2	5508	K-Vecinos	6.22%	5.44%
C3	5953	Random Forest	16.7%	7.9%
C3	5953	Multilayer Perceptron	33.7%	11.7%
C3	**5953**	**K-Vecinos**	**4.9%**	**5.98%**

Tabla 2: Pruebas con los conjuntos de datos C1, C2 y C3

Los resultados de estas pruebas se muestran en la Tabla 2, donde se observa que los mejores resultados se obtienen utilizando el Conjunto C3 y K-Vecinos como clasificador. El resto de los resultados no son aceptables dado que la tasa de Falsos Positivos es muy alta. De igual forma, se creó un nuevo conjunto de datos C4, donde se agregaron 4 variables nuevas que aportan información sobre las características globales de la imagen de la firma. Este conjunto también esta desbalanceado, así que se aplicó SMOTE aumentando la clase minoritaria en 100% y 150%. Así, se generaron dos conjuntos nuevos C5 y C6, con la misma cantidad de instancias que los conjuntos C2 y C3.

Conjunto	Cantidad de Instancias	Algoritmo de Clasificación	FAR	FRR
C4	4617	Random Forest	24.4%	1.6%
C4	4617	Multilayer Perceptron	39.8%	4.2%
C4	4617	K-Vecinos	20.1%	1.4%
C5	5508	Random Forest	11.2%	2.8%
C5	5508	Multilayer Perceptron	25.6%	9.3%
C5	5508	K-Vecinos	4.9%	1.9%
C6	5953	Random Forest	8.3%	3.3%
C6	5953	Multilayer Perceptron	22.6%	9.8%
C6	**5953**	**K-Vecinos**	**2.7%**	**2.3%**

Tabla 3: Pruebas con los conjuntos C4, C5 y C6

En la Tabla 3 se pueden observar los resultados de las pruebas con los conjuntos C4, C5 y C6. Nuevamente, el clasificador con el que se obtienen mejores resultados es K-Vecinos y utilizando el conjunto C6. Teniendo esto en cuenta, se elige K-Vecinos como clasificador para este enfoque y se hacen nuevas pruebas variando el valor de K y la cantidad de vecinos en el algoritmo SMOTE. Se crea un nuevo conjunto C7 utilizando 1 vecino en el algoritmo SMOTE el cual contiene la misma cantidad de instancias que el conjunto C6.

Los resultados de estas pruebas se pueden apreciar en la Tabla 4. Aquí podemos observar que los porcentajes de FRR se mantienen dentro de un rango y que los de FAR mejoran considerablemente con respecto a otros algoritmos. Es por esto que se decide utilizar K-Vecinos con 1 solo vecino en un conjunto de datos con 14 variables y aplicando SMOTE a la clase minoritaria en un 150% con 1 vecino.

Conjunto	Valor de K	FAR	FRR
C6	1	2.7%	2.3%
C7	**1**	**1.6%**	**2.3%**
C6	3	4.5%	2%
C7	3	2.9%	2%
C6	5	6.2%	2%
C7	5	4.2%	2.1%

Tabla 4: Pruebas variando cantidad de vecinos

Por otra parte, se realizó una prueba con el resto de las imágenes que no fueron utilizadas, se creó un conjunto C8 el cual contiene 1638 instancias de las cuales 1242 son *True* y 396 *False*. Utilizando el clasificador propuesto, se obtuvo un 6.1%% de FRR y un 20.45% de FAR.

Adicionalmente se realizaron nuevas pruebas creando otros conjuntos de datos. Primero, se varió la forma en la que se generaban las instancias *False*. Para las pruebas anteriores se hacía una comparación entre dos firmas falsas para obtener una instancia *False*; ahora se hace una comparación entre una firma

falsificada y una genuina para así generar un nuevo conjunto CN1 al cual se le aplicó SMOTE en un 50% a la clase minoritaria (*False*) obteniendo así 3726 instancias para cada clase. Por otra parte, dado que solo se tienen 12 firmas falsas y 24 genuinas por persona, se decidió balancear la clase *False* agregando firmas de otros usuarios hasta obtener 24 firmas falsas. De igual forma, se generan las instancias al comparar una firma falsa contra una genuina y se crea un conjunto CN2 el cual contiene 3726 *True* y 3888 *False*. Se utilizó K-Vecinos como clasificador y se evaluó con *Cross Validation*, los resultados se pueden apreciar en la Tabla 5.

Conjunto	FAR	FRR
CN1	2.3%	9.8%
CN2	**7.4%**	**7.6%**

Tabla 5: Pruebas variando la forma de obtener las instancias falsas

Para cada CN1 y CN2 se generaron dos conjuntos CP1 con 1242 *True* y 621 *False* y CP2 con 1242 *True* y 1296 *False*. Las instancias que conforman estos conjuntos no fueron utilizadas en el entrenamiento, los resultados obtenidos se pueden observar en la Tabla 6. En [1] aplican este método utilizando redes neuronales y sin agregar las 4 variables que se utilizaron en este trabajo. La base de datos que utilizaron contenía más de 1000 firmas genuinas y 350 falsas con distintos niveles de falsificaciones de 100 usuarios distintos logrando así obtener un 5.79% de FAR con falsificaciones expertas.

Conjunto	FAR	FRR
CP1	45.9%	18.8%
CP2	**5%**	**7.6%**

Tabla 6: Pruebas utilizando instancias no presentes en conjuntos anteriores

Para el segundo método basado en la Distancia Euclídea se realizaron pruebas variando la cantidad de firmas del entrenamiento y el tamaño de la imagen de la firma. Se utilizaron 8, 12 y 14 firmas para el entrenamiento y dos tamaños distintos de imágenes. Utilizando imágenes de mayor tamaño se puede observar que se obtienen mejores resultados, sin embargo, estos siguen siendo poco favorables. Lamentablemente, este enfoque es sensible a las pequeñas modificaciones en las firmas. En el conjunto de datos que se utilizó para este sistema las firmas de una misma persona poseen grandes variaciones, como se puede observar en la Figura 28, por lo que los cálculos de los umbrales se ven afectados de gran manera.

Cantidad Firmas Entrenamiento	Cantidad Firmas Genuinas Pruebas	Cantidad Firmas Falsas Pruebas	Tamaño de la Imagen	FAR	FRR
8	16	12	300x200	16.66%	53.47%
12	12	12	300x200	38.88%	29.62%
14	10	12	300x200	34.44%	30.55%
8	**16**	**12**	**500x400**	**12.96%**	**46.52%**
12	12	12	500x400	33.33%	28.70%
14	10	12	500x400	24.44%	28.70%

Tabla 7: Pruebas Método Distancia Euclídea

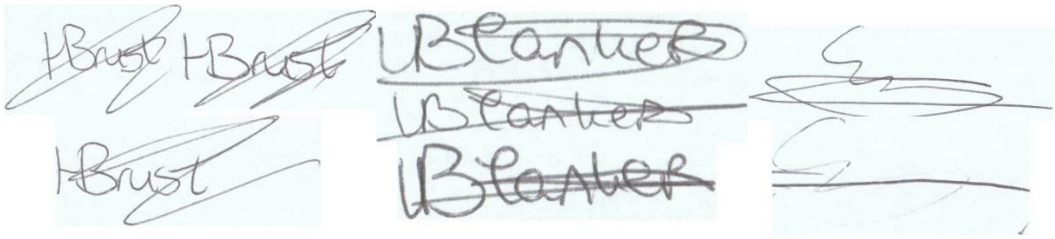

Figura 28: Variaciones en las firmas del conjunto de datos

Capítulo 5 – Conclusiones y Trabajos Futuros

En este trabajo se implementaron y modificaron dos métodos de verificación de firmas offline los cuales son capaces de comprobar la autenticidad de una firma. Se desarrolló una aplicación la cual cuenta con una interfaz que le permite realizar los distintos pasos que tiene un sistema de verificación de firmas, pre procesamiento, extracción de características y verificación.

En las pantallas de pre procesamiento es posible cargar y procesar un conjunto de datos y además de extraer las características de las imágenes procesadas. Por su parte en las interfaces de verificación se permite evaluar una firma para saber si esta es genuina o no.

Así mismo, la disponibilidad de un conjunto de datos de firmas que contenga una gran cantidad de imágenes, es una de las mayores limitantes de este trabajo. El conjunto de datos utilizado cuenta con firmas de 9 personas distintas, 24 genuinas y 12 falsificadas. Las falsificaciones son de tipo simple, no se contaron con falsificaciones expertas.

Para la creación del módulo de pre procesamiento se evaluaron distintas técnicas, hasta obtener una imagen que cumpliera con las expectativas. Se utilizaron algoritmos de binarización, erosión, dilatación, rotación y adelgazamiento para así mejorar la calidad de la imagen y obtener los mejores resultados en las fases posteriores. Por otra parte, los enfoques desarrollados son muy distintos tanto en la forma en que extraen las características como en la forma de verificar una firma. El primer método utiliza operaciones morfológicas para la obtención de una imagen a la cual se le puedan extraer una serie de características, las cuales son cantidades de píxeles de ciertos colores. Sin embargo, al momento de realizar las pruebas estas características resultaron insuficientes, es por esto que se decidió incluir nuevos valores que aportaran información global sobre la firma. Al hacer esto, se lograron obtener resultados bastante favorables, un 1.6% de Falsos Positivos y un 2.3% de Falsos negativos. Adicionalmente, se realizaron otras prueban donde se varió la forma en la que se obtienen las instancias pertenecientes a la clase *False*. Con esto, se lograron obtener resultados más estables al probar con un conjunto de datos que no fue utilizado en el entrenamiento. Dado que no se cuenta con un conjunto de datos balanceado y que la cantidad de instancias es poca, estos valores son satisfactorios.

En el segundo enfoque utiliza la Distancia Euclídea como media de distancia entre firmas. Esta es utilizada para la obtención de umbrales y para la verificación de una nueva firma. El algoritmo implementado utiliza 60 puntos característicos para calcular los umbrales, por esto una variación pequeña

en la firma tiene un gran impacto en el resultado de estos cálculos y del cálculo del centro geométrico. Una vez realizadas las pruebas los resultados fueron poco satisfactorios, esto se debe a la gran cantidad de variaciones que poseen las firmas de una misma persona dentro del conjunto de datos utilizado. El éxito de este método está condicionado a que el usuario firme con sumo cuidado, para que no exista una gran variación en las firmas que se utilizaran para el entrenamiento.

Como trabajos futuros se propone implementar los algoritmos de pre procesamiento de las imágenes del conjunto de datos de forma paralela dado que mientras mayor sea la cantidad de imágenes, mayor es el tiempo de procesamiento. De igual forma, se deben optimizar los métodos de extracción de características si van a ser utilizados con mayores cantidades de datos. También, se propone utilizar más de dos clases a la hora de clasificar las instancias. En este trabajo se utilizan solo *True* y *False* como clases, pero se pueden realizar pruebas utilizando las clases Genuina, Falsificación Experta y Falsificación Simple.

Teniendo en cuenta que el conjunto de datos es una de las mayores limitantes de este proyecto se sugiere crear un conjunto propio, la cual contenga una cantidad considerable de usuarios, firmas y falsificaciones tanto simples como profesionales.

Bibliografía

[1] S. Gupta, V. B. Kulkarni, A. A. Ambardekar, H. B. Kekre y V. A. Bharadi, «Off-Line Signature Recognition Using Morphological Pixel Variance Analysis,» de *International Conference and Workshop on Emerging Trends in Technology*, New york, 2010.

[2] B. Majhi, Y. S. Reddy y D. P. Babu, «Novel Features for Off-line Signature Verification,» *International Journal of Computers, Communications & Control,* vol. 11, nº 11, pp. 17-24, 2006.

[3] J. Frijters, «IKVM.NET,» 2002-2011. [En línea]. Available: http://www.ikvm.net/. [Último acceso: Abril 2015].

[4] M. Cofield, «Digital Imaging Basics,» 2005. [En línea]. Available: https://www.ischool.utexas.edu/technology/tutorials/graphics/digital/. [Último acceso: 2104 Enero 15].

[5] A. R. Smith, «A Pixel Is Not A Little Square, A Pixel Is Not A Little Square, A Pixel Is Not A Little Square!,» *Microsoft Computer Graphics, Technical Memo,* vol. 6, Julio 1995.

[6] A. Urkullu Villanueva, «Análisis automático de señales de atún obtenidas mediante sonar de largo alcance a bordo de buques pesqueros,» Universidad del País Vasco, Departamento de Ciencia de la Computación e Inteligencia Artificial, Tesis de Máster, 2011.

[7] M. K. Agoston, «Computer Graphics and Geometric Modeling», vol. 2, Heidelberg: Springer, 2005.

[8] D. Conesa, «Reconocimiento automático de números de placa de vehículos,» Universidad Central de Venezuela, Biblioteca Alonso Gamero, Tesis de Pregrado en la Licenciatura de Computación, Caracas, 2008.

[9] S. Mehmet y S. Bülent, «Survey over image thresholding techniques and quantitative performance evaluation,» *Journal of Electronic Imaging,* vol. 13, nº 1, pp. 146-165, 2004.

[10] J. Echeverri, J. Rudas y R. Toscano Cuello, «Eliminación de ruido impulsivo en imágenes a color, utilizando interpolación con funciones de base radial,» *Ingenieria,* vol. 16, nº 1, pp. 27-35, 2011.

[11] E. Escalona y E. Frias, «Módulo de adquisición y Pre-Procesamiento de imagenes digitales para un sistema de reconocimiento facial automatizado,» Universidad Central de Venezuela, Biblioteca Alonso Gamero, Tesis de Pregrado en la Licenciatura de Computación, Caracas, 2008.

[12] G. G. Aguilar, «Procesamiento digital de imágenes utilizando filtros morfologicos,» Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica, Tesis de Pregrado, Quito, 1995.

[13] U. Fayyad, G. Piatetsky-Shapiro y P. Smyth, «From Data Mining to Knowledge Discovery in Databases,» *AI Magazine,* vol. 17, nº 3, pp. 37-54, 1996.

[14] J. M. Molina y G. Jesús, «Técnicas de Análisis de Datos», Madrid, 2006, pp. 98-238.

[15] O. Maimon y L. Rokach, «Data Mining and Knowledge Discovery Handbook», Springer, 2010.

[16] J. Hernández, M. Ramírez y C. Ferri, «Introducción a la minería de datos», Madrid: Pearson - Prentice Hall, 2009.

[17] O. Hilton, «Detection of Forgery,» *Journal of Criminal Law and Criminology,* vol. 30, nº 4, 1939.

[18] C. Suen, Q. Xu y L. Lam, «Automatic recognition of handwritten data on cheques – Fact or Fiction?,» *Pattern Recognition Letters,* vol. 20, nº 11-13, p. 1287–1295, 1999.

[19] K. Madasu y B. Lovell, «An Automatic Offline Signature Verification and Forgery Detection System,» de *Pattern Recognition Technologies and Applications: Recent Advances*, New York, Information Science Reference, 2008, pp. 63-89.

[20] Y. Madhuri, K. Alok, P. Tushar y K. Bhupendra, «A Survey on Offline Signature Verification,» *International Journal of Engineering and Innovative Technology (IJEIT),* vol. 2, nº 7, 2013.

[21] M. Prathiba y L. Basavaraj, «Online handwritten signature verification system: A Review,» *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),* vol. 3, nº 2, 2014.

[22] H. Saikia y K. C. Sarma, «Approaches and Issues in Offline Signature Verification System,» *International Journal of Computer Applications,* vol. 42, pp. 45-52, 2012.

[23] D. Impedovo y G. Pirlo, «Automatic Signature Verification: The State of Art,» *IEEE Transactions on Systems, Man, and Cybernetics, Part C,* vol. 38, nº 5, pp. 609-635, 2008.

[24] H. Baltzakis y N. Papamarkos, «A new signature verification technique based on a two-stage neural network classifier,» *Engineering Applications of Artificial Intelligence,* vol. 14, nº 1, pp. 95-103, 2001.

[25] K. B. Raja, Chhotaray, R. K. y P. Sabyasachi, «Off-line Signature Verification Based on Fusion of Grid and Global Features Using Neural Networks.,» *International Journal of Engineering Science and Technology ,* vol. 2, nº 12, pp. 7035-7044, 2010.

[26] H. D. Chang, J. F. Wang y H. M. Suen, «Dynamic handwritten Chinese signature verification,» de *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, Tsukuba Science City, 1993.

[27] X. Xiao y G. Leedham, «Signature verification using a modified Bayesian network,» *Pattern recognition,* vol. 35, nº 5, pp. 983-995, 2002.

[28] A. I. Al-Shoshan, «Handwritten signature verification using image invariants and dynamic features,» de *Computer Graphics, Imaging and Visualisation*, Sydney, 2006.

[29] R. Bajaj y S. Chaudhury, «Signature verification using multiple neural classifiers,» *Pattern recognition,* vol. 30, nº 1, pp. 1-7, 1997.

[30] K. Huang y H. Yan, «Off-line signature verification based on geometric feature extraction and neural network classification,» *Pattern Recognition,* vol. 30, nº 1, pp. 9-17, 1997.

[31] S. Armand, M. Blumenstein y V. Muthukkumarasamy, «Off-line Signature Verification based on the Modified Direction Feature,» de *18th International Conference on Pattern Recognition*, Hong Kong, 2006.

[32] J. P. Drouhard, R. Sabourin y M. Godbout, «A comparative study of the k nearest neighbour, threshold and neural network classifiers for handwritten signature verification using an enhanced directional PDF,» de *roceedings of the Third International Conference on Document Analysis and Recognition*, Montreal, 1995.

[33] C. Gruber, M. Coduro y B. Sick, «Signature Verification with Dynamic RBF Networks and Time Series Motifs,» de *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule, 2006.

[34] N. F. O'Brien y S. C. Gustafson, «Real-time signature verification using neural network algorithms to process optically extracted features,» de *SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing*.

[35] M. Hanmandlu, M. H. M. Yusof y V. K. Madasu, «Off-line signature verification and forgery detection using fuzzy modeling,» *Pattern Recognition,* vol. 38, nº 3, pp. 341-356, 2005.

[36] Q. Z. Wu, S. Y. Lee y I. C. Jou, «On-line signature verification based on logarithmic spectrum,» *Pattern Recognition,* vol. 31, nº 12, pp. 1865-1871, 1998.

[37] R. Sabourin, J. P. Drouhard y E. S. Wah, «Shape matrices as a mixed shape factor for off-line signature verification,» de *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Ulm, 1997.

[38] K. Ueda, «Investigation of off-line Japanese signature verification using a pattern matching,» de *12th International Conference on Document Analysis and Recognition*, Edinburgh, 2003.

[39] Y. Mizukami, M. Yoshimura, H. Miike y I. Yoshimura, «An off-line signature verification system using an extracted displacement function,» *Pattern Recognition Letters,* vol. 23, nº 13, pp. 1569-1577, 2002.

[40] Y. Mizukami, Tadamura, Y. M. K. y I. Yoshimura, «Statistical displacement analysis for handwriting verification,» de *Image Analysis and Processing–ICIAP*, Cagliari, 2005.

[41] B. Schafer y S. Viriri, «An off-line signature verification system,» de *International Conference on Signal and Image Processing Applications (ICSIPA)*, Kuala Lumpur, 2009.

[42] R. S. R. D. D. Jana, «Offline Signature Verification using Euclidian Distance,» de *International Journal of Computer Science and Information Technologies*, India, 2014.

[43] N. Chawla, K. Bowyer, L. Hall y P. Kegelmeyer, «SMOTE: Synthetic Minority Over-sampling Technique,» *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, 2002.

[44] «Weka,» Machine Learning Group at the University of Waikato, [En línea]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Último acceso: Abril 2015].

[45] T. Y. Zhang y C. Y. Suen, «A Fast Parallel Algorithm for Thinning Digital Patterns,» *Communications of the ACM,* vol. 27, nº 3, pp. 236-239, 1984.

[46] S. Ahmed, A. Ramasamy, A. Khairuddin y J. Omar, «Automatic online signature verification, A prototype using neural networks,» de *TENCON*, Singapur, 2009.