Universidad de las Ciencias Informáticas
Vertex, Interactive 3D Environments, Facultad 4

Algoritmo de identificación automática de imágenes utilizando Álgebra Lineal

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Michel Hernández Rivera

Tutores: Ing. Yanet Liliana Garbey Gainza

DrC. Hassán Lombera Rodríguez

La Habana, 2018

Se pueden adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida.

Fidel Castro

Dedicatoria

A mis padres, mi hermano, a toda mi familia que siempre me apoyó y a mi amada.

Agradecimientos

Agradezco a mi familia, a todos mis amigos, a mis exelentes tutores Ing. Yanet Liliana Garbey Gainza y DrC. Hassán Lombera Rodríguez y a todas esas personas que de una forma u otra me ayudaron estos cinco años.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Michel Hernández Rivera
Autor

Ing. Yanet Liliana Garbey Gainza
Tutora

DrC. Hassán Lombera Rodríguez
Tutor

Resumen

En el Centro de Entornos Interactivos 3D existe una línea de trabajo con centros de rehabilitación de la capital en la que se utilizan videojuegos serios con fines terapéuticos. Dentro de ellos se encuentran los videojuegos destinados a la rehabilitación cognitiva en pacientes con algún trastorno cerebral. Entre los ejercicios desarrollados por los especialistas con los pacientes se encuentra el *test* de visoconstrucción. El objetivo es evaluar el control de las funciones motoras del paciente mediante la comparación de una imagen de muestra y la realizada por este. Para poder realizar un videojuego en el área de visoconstrucción, es fundamental lograr una identificación automática de una imagen en un conjunto dado. Aunque en el Centro de Entornos Interactivos 3D se han realizado trabajos referentes al tratamiento de imágenes, no se conoce alguno que haya abordado el problema de la clasificación entre ellas. El objetivo de este trabajo es desarrollar un algoritmo que permita clasificar automáticamente una imagen realizada manualmente en un conjunto de entrenamiento. La solución se basa en la Descomposición en Valores Singulares de una matriz rectangular correspondiente a un conjunto de entrenamiento dado. Asimismo, se utiliza la distancia euclidiana entre los conjuntos de entrenamiento y la imagen a clasificar. Al finalizar este trabajo, basado en los resultados obtenidos, se concluye que el algoritmo propuesto resulta adecuado cuando los distintos patrones que conforman la base de datos de entrenamiento no tienen parecidos entre sí.

Palabras clave: clasificar, distancia euclidiana, entrenamiento, imagen, Descomposición en Valores Singulares.

Índice general

Índice de figuras

Índice de tablas

Introducción

En el campo del procesamiento de imágenes, resulta de gran interés encontrar una solución óptima al problema de reconocimiento de estas. En la actualidad, varias empresas han invertido esfuerzo y recursos en darle una solución a este problema motivando una continua evolución en este aspecto. Ejemplo de esto es la inversión de Amadeus Ventures, el brazo corporativo de inversiones de Amadeus, en CrowdVision. Esta es una empresa especializada cuyo objetivo es ayudar a los aeropuertos a gestionar el incremento de viajeros para beneficiarse de un proceso de facturación más rápido y unas colas más cortas en los controles de seguridad mediante un software de seguimiento de personas basado en video. Otro ejemplo es la inversión de grandes empresas como Google, Facebook y Yahoo. Google ha invertido en la tecnología de código abierto *Facenet*, Facebook ha invertido en el *app Moments* y Yahoo ha adquirido empresas especializadas en estas tecnologías como *LookFlow*. Como resultado se puede observar en los últimos años un gran esfuerzo de modernización en conceptos tales como extracción de características para el posterior análisis de imágenes.

El análisis de imágenes es el proceso mediante el cual a partir de una imagen se obtiene una medición [1] e interpretación de sus características, las cuales se tendrán como punto de partida para el análisis de similitud entre imágenes (Poveda, 2013).

En el Centro de Entornos Interactivos 3D (VERTEX) existe una línea de trabajo con centros de rehabilitación de la capital en la que se utilizan videojuegos serios con fines terapéuticos. Estos representan una forma amena para evaluar al paciente, de manera que este se entretiene con el contenido del videojuego, presta mayor atención al ejercicio y colabora con el proceso de evaluación. Se han desarrollado videojuegos en colaboración con el Instituto Cubano de Oftalmología Ramón Pando Ferrer, para niños con dificultades en el campo visual. Un ejemplo de este tipo de videojuego es Meteorix, cuyo objetivo es la rehabilitación de la agudeza visual (Correa-Madrigal, 2015). Actualmente, se encuentran en desarrollo una serie de videojuegos destinados a la rehabilitación cognitiva en pacientes con algún trastorno cerebral, luego de haber sufrido un accidente. Dicho desarrollo se realiza en colaboración con el Centro Nacional de Rehabilitación Hospital Julio Díaz González ubicado en La Habana, Cuba.

Entre los ejercicios desarrollados por los especialistas con los pacientes se encuentra el *test* de visoconstrucción. El *test* consiste en reproducir una imagen, generalmente un dibujo con figuras geométricas básicas, en una hoja de papel. El objetivo es evaluar el control de las funciones motoras del paciente mediante la comparación de la imagen de muestra y la realizada por el paciente. Mientras exista mayor correspondencia entre ambas imágenes, mayor será el control de las funciones motoras que demostrará el paciente.

[1] Los datos de interés suelen ser casi siempre numéricos

En la actualidad se cuenta con varias formas de aplicar y de evaluar estos *test*. Una de ellas fue aplicada por el neurólogo José María Manubens Beltrán y consiste en solicitar la copia de cuatro figuras: círculo, rombo, rectángulos superpuestos y cubo (Beltrán, s.f.). Este procedimiento se le aplica a un grupo reducido de personas con características homogéneas, siendo posible la adaptación de la actividad en función de sus necesidades. Se dispone de datos normativos para la población de más de 70 años, procedentes de un estudio poblacional, para tener una base con el objetivo de fundamentar el criterio de evaluación de los especialistas. A estos *test* se le otorga una puntuación para evaluar el nivel de afectación de los pacientes. El valor mínimo es de 0 puntos y significa que esos pacientes presentan alteración grave, con figuras no reconocibles o graves distorsiones del esquema básico. Un máximo de 3 puntos cuando reproducen perfectamente las figuras con mínimas distorsiones.

Vale destacar que el proceso de aplicación y evaluación de los *test* se realiza en grupos reducidos y de forma manual por el médico. Esto provoca que en un día se puedan atender un número limitado de personas y dificulta la aplicación de los *test* a los pacientes.

Para poder realizar un videojuego en el área de Visoconstrucción, es fundamental lograr una identificación correcta de una imagen en un conjunto de forma automática. Aunque en VERTEX se han realizado trabajos referentes al tratamiento de imágenes, no se conoce alguno que haya abordado la identificación o el reconocimiento de patrones entre dos de ellas.

Teniendo en cuenta lo antes mencionado se define como **problema a resolver**: ¿Cómo identificar automáticamente una aproximación realizada manualmente dado un conjunto de entrenamiento? Para ello se tiene como **objeto de estudio** el procesamiento y reconocimiento de imágenes. El **campo de acción** de la investigación se enmarca en el reconocimiento de patrones en imágenes basado en álgebra lineal. La presente investigación tiene como **objetivo general** implementar un algoritmo que permita clasificar automáticamente una imagen realizada manualmente en un conjunto de entrenamiento.

Se pretende obtener como **posibles resultados** un algoritmo que:

- Permita el reconocimiento de niveles de similitud entre imágenes, resolviendo los problemas de rotación y traslación de objetos dentro de una imagen.
- Sea reutilizable para la posterior implementación de un videojuego interactivo.

Para dar cumplimiento al objetivo planteado se definen las siguientes **tareas de investigación**:

- Elaborar el Marco Teórico de la investigación a través de un estudio del estado del arte de las técnicas existentes para reconocimiento de patrones y comparación entre imágenes.
- Caracterización de herramientas y tecnologías. Selección de las que se ajustan a las necesidades de la solución propuesta.
- Análisis y diseño de un algoritmo de procesamiento de imágenes para dar solución al problema planteado.

Para el desarrollo del trabajo se utilizaron **métodos de investigación científica** dentro de los que se incluyen los métodos teóricos y empíricos que se muestran a continuación:

Métodos Teóricos:

- **Analítico-sintético:** se utilizó en el estudio a fondo de las características de diferentes tipos de bibliotecas para el tratamiento de imágenes. Además, se estudiaron las diferentes técnicas de reconocimiento de patrones existentes con el objetivo de resumir y exponer el resultado del estudio realizado.
- **Histórico-Lógicos:** se empleó para el estudio crítico de los trabajos realizados con anterioridad y la utilización de estos como punto de referencia y comparación de los resultados analizados.

Métodos Empíricos:

- **Consulta de la información:** en todo tipo de fuente: Se utiliza para la realización del marco teórico de la investigación.

El trabajo de diploma está estructurado como sigue: Introducción, tres Capítulos, Conclusiones, Recomendaciones, Glosario, Acrónimos y Referencias bibliográficas. En el Capítulo 1 se expone la fundamentación teórica de la investigación, y se incluyen las características principales de las técnicas para el procesamiento de imágenes. En el Capítulo 2 se expone la fundamentación de la propuesta de solución. Se realiza la descripción detallada del algoritmo. El Capítulo 3 aborda la Validación de la propuesta de solución mediante la técnica de un prototipo funcional.

CAPÍTULO 1

Fundamentación Teórica

En este capítulo se expone la fundamentación teórica de la investigación. Se expondrán los principales métodos existentes para el reconocimiento de similitud entre imágenes. Se identificarán además las tecnologías y herramientas para el desarrollo del algoritmo.

1.1. Técnicas para el reconocimiento de imágenes

Tradicionalmente el reconocimiento automático de imágenes ha sido una tarea difícil debido a su tasa de errores. Existen varias técnicas para el reconocimiento de imágenes de manera automática. Cada una de ellas enfoca el proceso de reconocimiento de imágenes de manera diferente. Poseen como característica común el preprocesamiento de la imagen. En este proceso extraen las características de la imagen que necesitan para realizar la clasificación y eliminan los elementos distorsionantes como el ruido. Las diferencias radican principalmente en el método que utilizan para comparar o clasificar las imágenes.

Entre las técnicas más utilizadas actualmente se encuentran: Clasificadores en Cascada, Redes Neuronales Artificiales (ANN, por sus siglas en inglés), Transformada de Wavelet, Histogramas, Eigenconjugación, Descomposición en Valores Singulares (SVD, por sus siglas en inglés). Para determinar la técnica más factible para el desarrollo de la propuesta de solución se analizarán las antes mencionadas.

1.1.1. Clasificadores en cascada

Las etapas de un clasificador en cascada están dispuestas en orden de complejidad creciente. Por lo general, en las imágenes predominan las ventanas negativas; la estructura en cascada permite rechazar, de manera eficiente, estas ventanas que no son de interés y enfocar los recursos en las ventanas positivas (Cheung y Medina, 2013). En un primer paso, se genera una detección con algún clasificador. Una vez hecho esto los elementos mal clasificados aumentan su peso para que el siguiente clasificador usado, dé más importancia a la clasificación de los elementos con mayor peso. Una vez realizado este último paso con diferentes clasificadores, obtenemos un único clasificador como combinación de los anteriores y que

clasifica todos los elementos correctamente con un valor de error aceptable (Delgado-Rodríguez, 2012). Cada nodo de un clasificador en cascada se basa en algunas características o algoritmos como son, la haar-like (Lienhart y Maydt, 2002) o AdaBoost (Weiming Hu, Wei Hu y Maybank, 2008) entre otras.

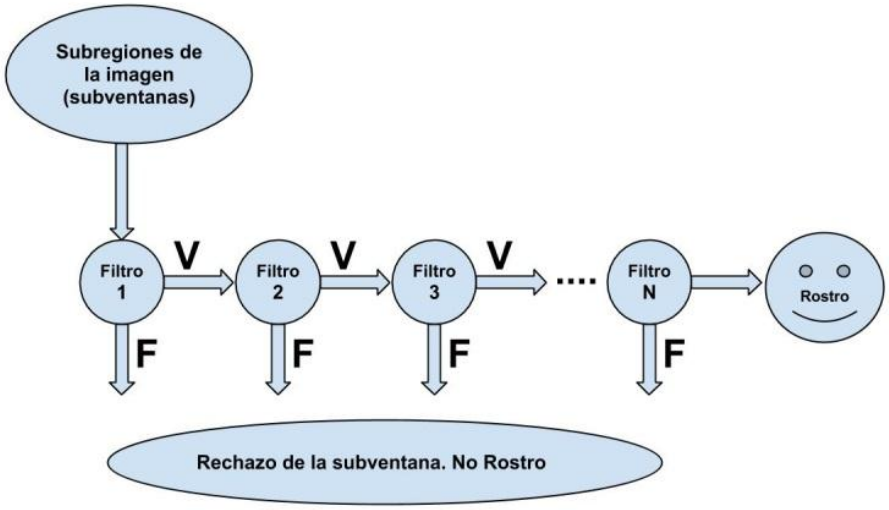


Figura 1.1. Diagrama de clasificación en cascada.

En la Figura 1.1 se muestra un ejemplo de clasificador en cascada que utiliza AdaBoost como filtros encadenados o en serie para la detección de rostros en imágenes digitales. El primer filtro elimina un gran número de ejemplos negativos (falsos F) con una cantidad de procesamiento muy pequeño. Se determina que es un objeto (Rostro) si los filtros producen la decisión de verdad (V) (García-Villanueva y Ramírez-Zavala, 2013).

1.1.2. Redes Neuronales Artificiales

El campo de procesamiento de imágenes está continuamente evolucionando. Durante los últimos años ha surgido un interés significativo en campos como la morfología de imágenes, ANN, procesamiento de imágenes en color y/o en escala de grises, comprensión de datos de imágenes, reconocimiento de imágenes y sistemas de análisis basados en conocimiento (García, 2013).

En la actualidad se pueden encontrar varias aplicaciones de las ANN en el reconocimiento de imágenes. Se puede citar el análisis y el procesado de señales, control de Procesos, diagnósticos Médicos, reconocimiento de Imágenes, entre otras (Olabe, 1998).

Las ANN son un paradigma de aprendizaje ampliamente tratado en la literatura de la inteligencia artificial (Alpaydin, 2014).

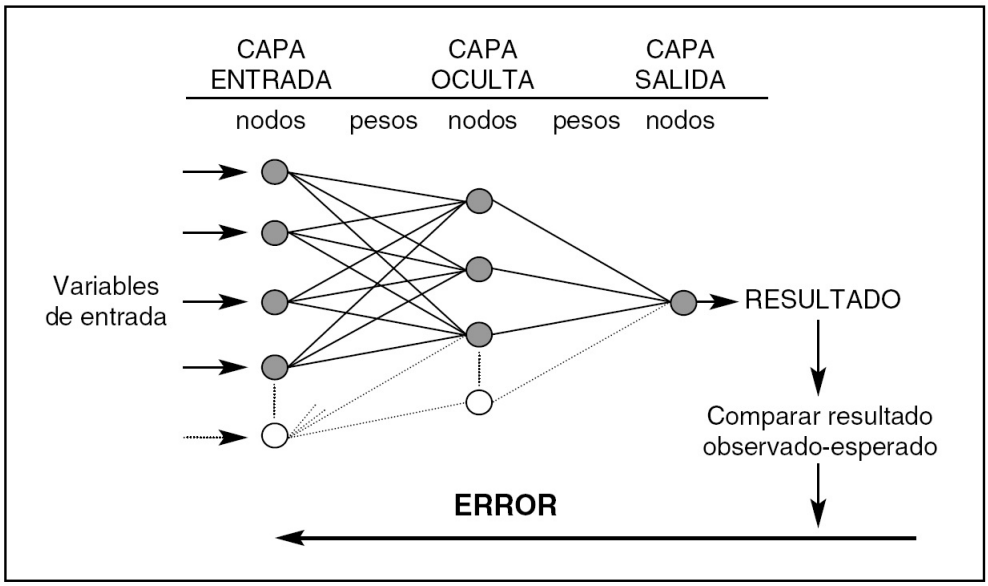


Figura 1.2. Perceptrón de tres capas.

En la Figura 1.2 se muestra la arquitectura de una ANN (perceptrón de tres capas). Los pesos son los parámetros que deben ajustarse durante el proceso de entrenamiento para conseguir que la capa de salida de resultados coincida con los observados. El error entre lo observado y el resultado de la red se propaga hacia atrás [1] y debe ir disminuyendo en las iteraciones sucesivas en las que se presentan los valores de las variables predictoras. La complejidad de una red depende del número de nodos de su capa oculta (Trujillano-Cabello et al., 2005).

Características

Las ANN están inspiradas en las redes neuronales biológicas del cerebro humano. Están constituidas por elementos que se comportan de forma similar a la neurona biológica en sus funciones más comunes. Estos elementos están organizados de una forma parecida a la que presenta el cerebro humano. Además, poseen características muy similares a las del cerebro, por ejemplo, aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos (Olabe, 1998).

Para poder extraer las características principales de una imagen utilizando una ANN es necesario procesarla para lograr diferenciar de mejor forma los objetos a clasificar. Dentro de los procesamientos que se le pueden aplicar a imágenes se encuentran: llevar la imagen a escala grises, realizarle filtrado para eliminar el ruido y la binarización (García, 2013).

[1] conocido como *backpropagation*

1.1.3. Transformada de *Wavelet*

Las *wavelets* [2] son funciones que satisfacen ciertos requerimientos matemáticos y son utilizadas como base para la representación de datos o de otras funciones. Permiten describir una función en términos de un contorno general más detallado, sin importar si la función de interés es una imagen no curva o una superficie. Las *wavelets* ofrecen una técnica elegante para representar los niveles de detalle presentes. Las *wavelets* son muy adecuadas para aproximación de datos con variaciones o con discontinuidades abruptas. La idea fundamental detrás de las *wavelets* es analizar funciones de acuerdo a escalas. En el análisis por *wavelets* la escala que se utiliza para analizar los datos juega un papel especial. Las *wavelets* son robustas con respecto a cambios de intensidad de color, y pueden capturar de manera eficiente información de textura y forma (L. R. Castro y S. M. Castro, 1995).

El procedimiento general del análisis utilizando *wavelets* es adoptar una función prototipo, llamada generalmente *wavelet* madre. La señal temporal se realiza entonces utilizando dilataciones y traslaciones de dicha función. La señal original puede entonces representarse como combinación lineal de la función original, sus trasladadas y dilatadas. Esto se denomina una expansión en *wavelets* (ibíd.).

La elección de la *wavelet* madre (y de este modo de la base o del marco de *wavelets*) no es única y depende del tipo de función o de datos a analizar. Una elección adecuada o la eliminación de coeficientes por debajo de un umbral prefijado, hace que las *wavelets* sean una herramienta excelente, entre otras, para la compresión de datos (ibíd.).

Similar a la transformada de Fourier, la transformada *wavelet* descompone una señal en términos de funciones $\psi(t)$ denominadas *wavelets*. El potencial del análisis de imágenes con *wavelets* está justamente en su capacidad de localizar cambios abruptos en las señales 1D o 2D, lo cual es aprovechado para la detección de los bacilos presentes en una imagen. El proceso consiste en aplicar sobre las imágenes la Transformada de Wavelet Discreta (DWT, por sus siglas en inglés) y extraer de sus coeficientes las características más sobresalientes de los objetos que conforman la imagen, cuantificarlas y obtener los descriptores respectivos al patrón seleccionado (Romo, Ramírez y Valdivieso, 2007).

La Transformada *Wavelet* es una herramienta usada en el procesamiento de señales y la detección de disturbios en sistemas de energía (Socorro-Borges, 2017).

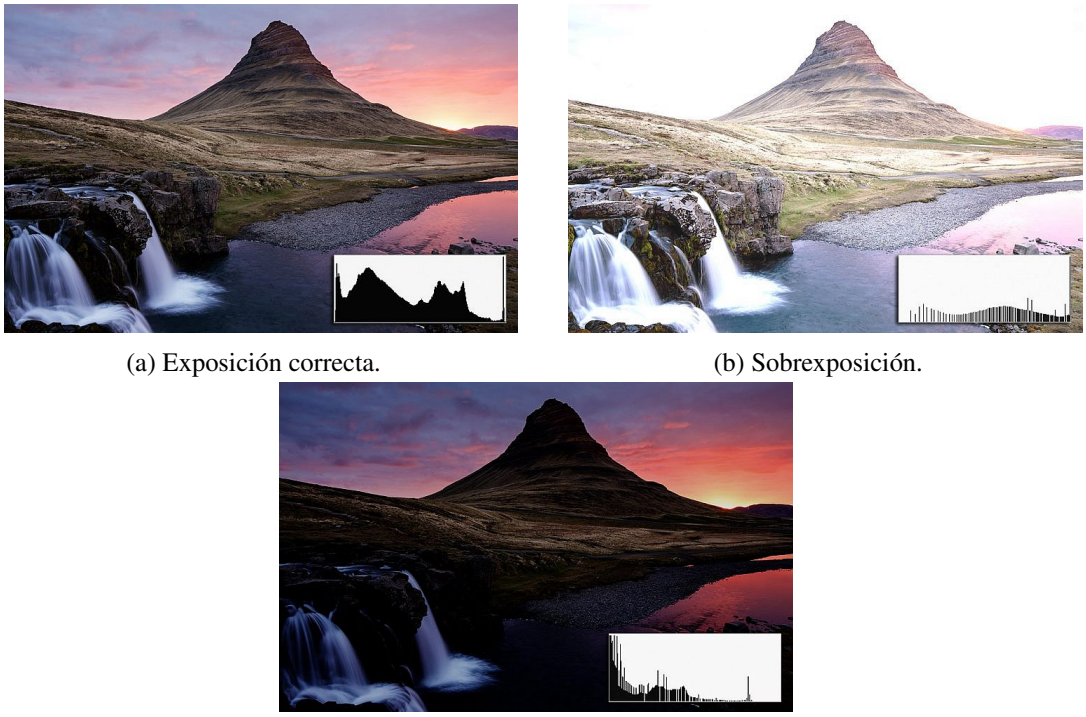
1.1.4. Histogramas

El histograma de una imagen es un arreglo unidimensional, donde cada elemento contiene el número de píxeles existentes para un nivel de gris en específico (Flores, 2015). La intensidad de un histograma muestra cómo los niveles de brillo están ocupados en la imagen; el contraste de la imagen se mide con el rango de brillo de la imagen. El histograma grafica el número de píxeles con un nivel de brillo particular. Otra de las características que se pueden detectar con el histograma es la existencia de ruido en la imagen, mediante el reconocimiento del histograma ideal. La eliminación de este ruido puede mejorar la calidad de la imagen, lo

[2] también llamadas *ondelettes* u onditas

que en consecuencia facilita que las técnicas que se le apliquen a dicha imagen puedan realizarse de manera más fácil y con mejores resultados (Enriquez-Vasquez, 2013).

Los picos en el histograma se utilizan para localizar los grupos en la imagen [3]. A veces, una iluminación deficiente o desigual en la imagen, hace que no sea conveniente el uso de un umbral global para toda la imagen. Una mejor aplicación de esta técnica consiste en aplicar de forma recursiva el método de búsqueda de histograma a los *clusters* de la imagen con el fin de dividirlos en grupos más pequeños. Esto se repite con las agrupaciones, cada vez más pequeñas hasta que no se puedan formar más de ellas (Gil-Rodríguez, 2008).



(a) Exposición correcta.

(b) Sobrexposición.

(c) Subexposición.

Figura 1.3. Histogramas de los diferentes valores de exposición para una imagen.

En la Figura 1.3 se muestran ejemplos del histograma de una imagen. Se puede apreciar que es la misma imagen en los tres casos, lo que cambia es la exposición, quedando en todas diferentes histogramas.

1.1.5. Eigenconjugación

La Eigenconjugación está basada en la información aportada por los valores y vectores propios (conocidos también como eigenvalores y eigenvectores) de una imagen, que genera a partir de estos una secuencia característica. La idea es evaluar el efecto de la conjugación de las matrices originales, los vectores

[3] el color o la intensidad pueden ser usados como medida

y valores en cada una de las otras. Se ejemplifica su utilidad para el cálculo de similitud entre imágenes (Ortega-Gonzalez, 2007).

La técnica de la Eigenconjugación es un método para extraer características, valida las relaciones de similitud entre matrices de datos ofreciendo una solución polinomial que emplea métodos numéricos que pueden ser fácilmente implementados en una computadora. Es una técnica que permite hacer análisis de similitud al más bajo nivel que emplea la información completa y directa de los datos originales sin necesidad de decisiones humanas ni de reducción o simplificaciones arbitrarias (Carrasco-Corona et al., 2012).

En Algebra Lineal, los vectores y valores propios son empleados para caracterizar la transformación definida por una matriz cuadrada. La determinación de esos vectores y valores es un problema ampliamente estudiado y existen varios procedimientos numéricos para resolverlo (ibíd.).

Un ejemplo de la relación entre Eigenconjugación y la similitud entre imágenes se describe en Ortega-Gonzalez, 2007, p.46.

Los conceptos de valores propios, vectores propios y su descomposición han tenido un impacto considerable en distintos dominios de la ciencia moderna. En Estadística las Eigenconjugación se emplean como descriptores de características de conjuntos de datos. En Física cuántica se utiliza una variante interesante del concepto conocido como estados propios o eigenestados que representan las bases de los estados posible para un sistema específico.

1.1.6. Descomposición en Valores Singulares

La SVD se ha convertido en una herramienta muy importante en las ciencias de la computación y ha encontrado una amplia gama de aplicaciones en el procesamiento de imágenes, tales como: filtrado y compresión. Con esta técnica se resuelve el problema de rotación y traslación de objetos dentro de una imagen (Waldemar y Ramstad, 1996).

En Álgebra Lineal la SVD es un método para la factorización de matrices rectangulares y está basada en la alineación de las secuencias características obtenidas. La matriz original se descompone en los siguientes tres conjuntos: Espacio (U), Luminosidad (S), Valores nulos (V). Los valores singulares (S) y los vectores singulares (U, V) describen completa y unívocamente la información intrínseca de una matriz rectangular. Esto posibilita que dos matrices nunca tengan la misma descomposición en vectores y valores singulares, aprovechando estas características para representar estas matrices sin pérdida de información ni ambigüedades (Ortega-Gonzalez, 2007).

La SVD trata las filas y columnas en forma simétrica, por lo tanto, proporciona más información sobre la matriz. Eso también ordena la información contenida en la matriz para que, en términos generales, la parte dominante se haga visible. Esta es la propiedad que hace que la SVD sea útil en la minería de datos y muchas otras áreas (Eldén, 2007).

1.2. Procesamiento de Imágenes

El procesamiento, análisis y la interpretación de imágenes es un campo de especialización muy importante de la visión artificial, que hace posible que un ordenador procese imágenes o fotografías bidimensionales (aisladas o bien conectadas en secuencias dinámicas o temporales). La finalidad del procesamiento digital de imágenes es procesar, comprender y descifrar características o rastrear objetos, con un objetivo claro, en una imagen (*Procesamiento de imagenes* 2018).

La extracción de características hace posible la reducción de la complejidad (dimensionalidad) de una imagen mediante la transformación de los datos de entrada en un conjunto de rasgos que representan las características esenciales de la imagen (ibíd.). Estos rasgos están compuestos por: píxel, resolución, profundidad de color, compresión (Gómez, 2018).

La similitud entre imágenes es un tema complejo de abordar en las ciencias de la computación. Esto se debe a que la capacidad conceptual del ser humano para establecer relaciones de semejanza entre objetos es representada en espacios métricos con propiedades geométricas (Yreta y Alberto, 2006). Se establece analizando aquellas propiedades que se consideran como importantes para establecer, hasta cierto nivel, la semejanza entre dos imágenes.

Los primeros esfuerzos para entender el problema de similitud los dio Tversky (1977), a continuación, se representan los problemas de similitud entre imágenes identificados por Tversky separándoles en cuatro enfoques:

a) La comparación de entidades usando como criterio los elementos comunes (*common elements approach*).
b) Realizar comparaciones en donde se persigue emparejar una entidad con respecto a la otra (*template approach*).
c) Realizar comparaciones de entidades por su geometría (*geometric approach*).
d) Por último, el enfoque de características (*feature approach*). Este enfoque es similar al de elementos comunes, aunque éste último no requiere identificar a priori las propiedades involucradas en la comparación.

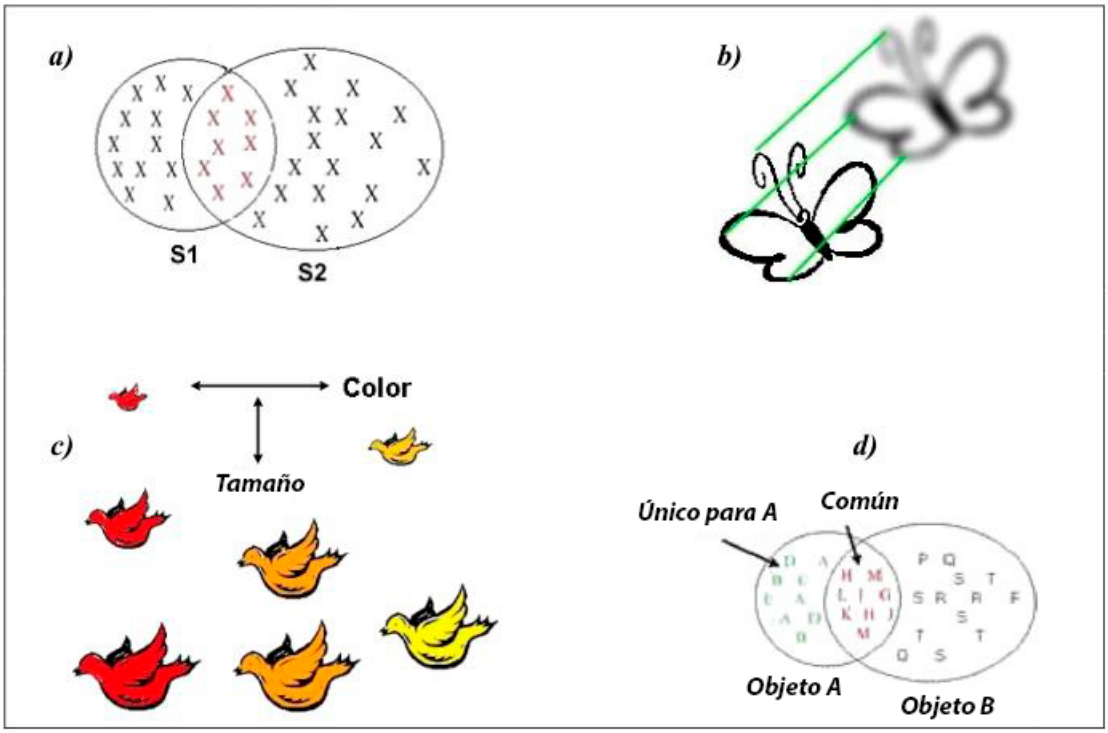


Figura 1.4. Representación de los problemas de similitud entre imágenes identificados por (Tversky, 1977).

Luego del análisis de los enfoques propuestos por Tversky, en el presente trabajo se pretende analizar el tercero que trata de la comparación de entidades por su geometría.

1.3. Herramientas para el desarrollo

1.3.1. Entorno de Desarrollo Integrado

Visual Studio

Microsoft Visual Studio es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, a lo cual hay que sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco. El software en sí es gratis lo que las características lujosas tienen un precio. Posee un enfoque renovado para mejorar la eficiencia de las tareas fundamentales que los desarrolladores desempeñan diariamente. Posee una instalación ligera y modular adaptada a las necesidades del desarrollador, un IDE rápido desde el arranque al apagado, una nueva manera de ver, editar y depurar cualquier código sin proyectos ni soluciones. Se utilizará Visual Studio 2017 en su versión 4.7.02556.

1.3.2. Bibliotecas de Álgebra Lineal y procesamiento de imágenes

Las bibliotecas son archivos (no siempre externos) que nos permiten llevar a cabo diferentes tareas sin necesidad de preocuparnos por cómo se hacen sino simplemente entender cómo usarlas. Las bibliotecas en C# permiten hacer nuestros programas más reutilizables facilitando además crear programas con funcionalidades bastante complejas en unas pocas líneas de código. Es la base sobre la que se compilan aplicaciones, componentes y controles de .NET Framework.

Para el desarrollo de la propuesta de solución se seleccionaron las bibliotecas Accord.net y Math.NET Numerics. La primera es un framework de aprendizaje automático .NET, presenta entre muchas otras prestaciones, el procesado de imágenes posee un conjunto completo de aplicaciones de muestra, proporciona un inicio rápido para comenzar a funcionar rápidamente, y una extensa documentación y wiki ayuda a completar los detalles. La segunda tiene como objetivo proporcionar métodos y algoritmos para cálculos numéricos en ciencias, ingeniería y uso diario; los temas cubiertos incluyen funciones especiales de álgebra lineal como SVD(método principal utilizado en el algoritmo), modelos de probabilidad, números aleatorios, interpolación, integración, regresión, problemas de optimización y más.

1.4. Conclusiones parciales

El estudio de las diferentes técnicas de procesamiento de imágenes permitió esclarecer los pasos a seguir y tener en cuenta para desarrollar la propuesta de solución. Vale destacar que cualesquiera de las estrategias revisadas podían ser elegidas para el desarrollo de la solución. Debido a que las imágenes tratadas contienen trazos sencillos y no poseen similitud entre ellas se selecciona SVD. Esta propone una solución sencilla, elegante y robusta, desde el punto de vista matemático, acorde al problema en cuestión.

Se seleccionaron las bibliotecas de álgebra lineal y procesamiento de imágenes Accord.net y Math.NET Numerics. Estas permitirán realizar el trabajo algebraico necesario para lograr el trabajo con las imágenes. Además, poseen funciones definidas para calcular SVD.

CAPÍTULO 2

Fundamentación de la propuesta de solución

En este capítulo se explica el funcionamiento de la SVD y cómo se realiza el entrenamiento y la clasificación en el algoritmo.

2.1. Descomposición en valores singulares

En aras de hacer autocontenida la tesis en relación a la factorizacion SVD se enuncia el Teorema 2.1.1, ampliamente descrito en la literatura (Eldén, 2007; Golub y Van-Loan, 1996).

Teorema 2.1.1. *Si A es una matriz real m-por-n entonces existen matrices ortogonales*

$$U = [\mu_1, ..., \mu_m] \in \mathbb{R}^{mxm} \quad y \quad V = [\upsilon_1, ..., \upsilon_n] \in \mathbb{R}^{nxn},$$

tal que

$$U^T AV = diag(\sigma_1, ..., \sigma_p) \in \mathbb{R}^{mxn} \quad p = min\{m, n\},$$

donde

$$\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_p \geqslant 0.$$

Prueba. La suposición $m \geq n$ no es una restricción: en el otro caso, simplemente se aplica A^t. La demostración del teorema anterior puede ser verificada en (Golub y Van-Loan, 1996, p.97)

Entre sus principales aplicaciones se encuentran: cálculo del rango, procesamiento de señales, obtención del núcleo y aproximación de matrices.

El cálculo de la SVD consiste en encontrar los valores propios y vectores propios de AA^T y A^TA. Los vectores propios de A^TA forman las columnas de V, los vectores propios de AA^T forman las columnas de U. Además, los valores singulares en S son raíces cuadradas de valores propios de AA^T o A^TA. Los valores singulares son las entradas diagonales de la matriz S y están dispuestos en orden descendente. Los valores singulares son siempre números reales. Si la matriz A es una matriz real, entonces U y V también son reales (Strang, 2006).

Para entender mejor como resolver la SVD, se tomó un ejemplo de una matriz proporcionada en (Kuruvilla et al., 2002).

$$A = \begin{pmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

En este ejemplo la matriz A es de dimensión 4 x 2. Para encontrar los valores propios de A calculamos las matrices AA^T y A^TA. Como se menciona anteriormente, los vectores propios de AA^T forman las columnas de U, de modo que se realiza el análisis siguiente para encontrar U.

$$AA^T = \begin{pmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 & 0 \\ 4 & 3 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 20 & 14 & 0 & 0 \\ 14 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = W.$$

Ahora se tiene una matriz W^{mxm} y se puede determinar los valores propios de esta. Como $Wx = \lambda x$ entonces $(W - \lambda I)x = 0$

$$\begin{pmatrix} 20-\lambda & 14 & 0 & 0 \\ 14 & 10-\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{pmatrix} x = (W - \lambda I)x = 0.$$

El determinante de la matriz $(W - \lambda I)$ para un conjunto único de valores propios debe ser igual a cero. Por lo tanto, a partir de la solución del polinomio característico, $\mid W - \lambda I \mid = 0$ se obtiene: $\lambda = 0, \lambda = 0; \lambda = 29.883; \lambda = 0.117$ (cuatro valores propios ya que es un polinomio de cuarto grado). Este valor se puede usar para determinar el vector propio que se puede colocar en las columnas de U. Por lo tanto, se obtiene las siguientes ecuaciones:

$$\begin{aligned} 19.883x_1 + 14x_2 &= 0, \\ 14x_1 + 9.883x_2 &= 0, \\ x_3 &= 0, \\ x_4 &= 0. \end{aligned}$$

Al simplificar las dos primeras ecuaciones, se obtiene una relación que relaciona el valor de *x1* y *x2*. Estos valores se eligen de modo que los elementos de S sean las raíces cuadradas de los valores propios. Por lo tanto, una solución que satisfaga el conjunto de ecuación anterior es: x1 = -0.58 y x2 = 0.82 y x3 = x4 =

0 (esta es la segunda columna de la matriz U).

Sustituyendo el otro valor propio que se obtiene:

$$\begin{aligned} -9.883x_1 + 14x_2 &= 0, \\ 14x_1 + 19.883x_2 &= 0, \\ x_3 &= 0, \\ x_4 &= 0. \end{aligned}$$

Por lo tanto, una solución que satisfaga este conjunto de ecuaciones es x1 = 0.82 y x2 = -0.58 y x3 = x4 = 0 (esta es la primera columna de la matriz U). Combinando estos se obtiene:

$$U = \begin{pmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Del mismo modo, A_TA compone las columnas de V para que se pueda realizar un análisis similar para encontrar el valor de V.

$$A_TA = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 4 & 3 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 4 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

De manera similar se obtiene la expresión:

$$V = \begin{pmatrix} 0.40 & -0.91 \\ 0.91 & 0.40 \end{pmatrix}.$$

Finalmente, como se mencionó anteriormente, S es la raíz cuadrada de los autovalores de AA_T o A_TA y se puede obtener directamente, dando como resultado:

$$S = \begin{pmatrix} 5.47 & 0 \\ 0 & 0.37 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

2.1.1. Compresión de la imagen en la solución propuesta

El objetivo de la compresión de una imagen es reducir datos redundantes e irrelevantes con la menor pérdida posible de información para optimizar el posterior procesado de la misma. Existen dos tipos de compresión: compresión sin pérdida y compresión con pérdida. La segunda es una técnica que modifica la

estructura y la radiometría de la imagen para que ocupe menor volumen. Produce pérdida de la información menos significativa. La reconstrucción de un fichero comprimido por esta técnica no es exactamente el fichero original. Se consiguen tasas más elevados de compresión que en la compresión sin pérdida (Pérez, Aguilera y Muñoz, 2003). Para el desarrollo de la propuesta de solución se selecciona la técnica de compresión con pérdida. Se utiliza esta porque al aplicarle SVD a una matriz, las matrices U, S y V contienen la información más importante de la matriz en sus primeras columnas.

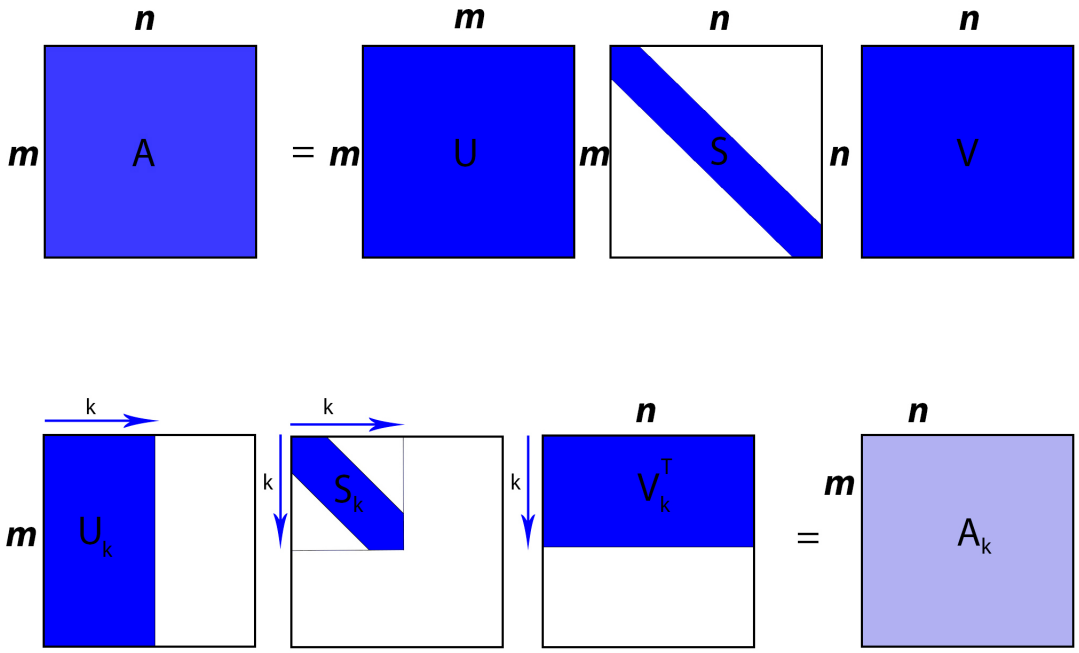


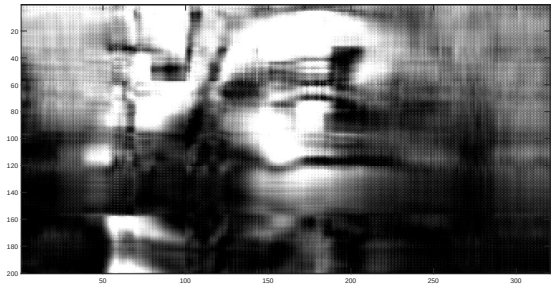
Figura 2.1. Compresión de una matriz.

En la Figura 2.1 se muestra el proceso de descomposición, compresión y reconstrucción de una imagen. El conjunto de imágenes superior muestra cómo quedan las matrices U, S, V al aplicarle SVD a la matriz A. En el conjunto inferior se muestra el resultado obtenido al aplicarle la compresión, para ello se define el valor de compresión(k) que constituye el valor por el cual se va a delimitar el factor de compresión, es decir cuántas columnas o filas se conservan. Para realizar la reconstrucción de la matriz de la imagen A, se realiza la multiplicación de las matrices comprimidas, obteniéndose A_k, similar a la original.

(a) Imagen original del *clown*.

(b) Imagen del *clown* con un *k=100*.

(c) Imagen del *clown* con un *k=50*.



(d) Imagen del *clown* con un *k=10*.

Figura 2.2. Imagen Clown de Matlab con diferentes valores de *k*.

Al analizar las imágenes contenidas en la Figura 2.2 se observa que la imagen es reconocible a simple vista con un *k=10*. Esto equivale a un 190 % de compresión con respecto a la imagen original la cual posee un *k=200*. Además la imagen original es de tamaño m x n (m-filas, n-columnas), al aplicarle la compresión, la cantidad de elementos resultantes en las matriz U al aplicar SVD es de $(m+n)k$. La reducción significativa de los elementos de la matriz como resultado de la aplicación de la compresión permite la optimización del algoritmo, reduciendo así el costo computacional sin afectar la precisión del algoritmo.

2.2. Clasificación de una imagen utilizando Descomposición en Valores Singulares

Si el objeto es una matriz o un vector, entonces las normas en álgebra lineal son útiles para evaluar la precisión de la respuesta o para medir progreso durante una iteración. Si el objetivo es calcular semejanza, es necesario poder cuantificar la distancia entre dos subespacios. Las proyecciones ortogonales son críticas en este respecto (Golub y Van-Loan, 1996).

2.2.1. Ángulo entre subespacios

Las imágenes de un patrón dado se consideran los vectores para una base que abarca un subespacio. La pregunta es: ¿qué tan cerca está un subespacio de entrenamiento de otro? La respuesta es relevante, esta daría información referente a la cercanía de las regiones en el espacio de alta dimensión, en el que los patrones son agrupados. Por ejemplo, los números 3, 5 y 8 tienen formas muy similares, se esperaría que sus regiones en este espacio de alta dimensión, sean cercanas. Esto ayudaría a entender por qué el algoritmo funciona como lo hace o quizás por qué no funciona como se espera (Savas, 2003). Los ángulos principales se obtienen midiendo cuan separados están los subespacios. El método de cálculo de los ángulos principales entre dos subespacios se describe a continuación: Sean F y G subespacios en $\mathbb{R}^m$

$$p = dim(F) \geq dim(G) = q \geq 1.$$

Los ángulos principales $\sigma_1, \cdots, \sigma_q \in [0, \pi/2]$ entre F y G se define recursivamente por

$$cos(\sigma_k) = \max_{\mu \in F} \max_{\upsilon \in G} \mu^t \upsilon = \mu_k^t \upsilon_k,$$

sujeto a:

$$\begin{aligned} \|\mu\| = \|\upsilon\| &= 1 \\ \mu^t \upsilon_i &= 0 \quad \text{i= 1:k-1} \\ \upsilon^t \mu_i &= 0 \quad \text{i= 1:k-1} \end{aligned}$$

Teniendo en cuenta que los ángulos principales satisfacen $0 \leq \sigma_1 \leq \cdots \leq \sigma_q \leq \pi/2$. Los vectores $\{\mu_1, \cdots, \mu_q\}$ y $\{\upsilon_1, \cdots, \upsilon_q\}$ son conocidos como los vectores principales entre los subespacios F y G. Si $p = q$ entonces $dist(F, G) = \sqrt{1 - \cos(\sigma_p)^2} = \sin(\sigma_p)$.

El valor de calcular el ángulo entre subespacios varía de $0 - \pi/2$. Mientras el valor obtenido este más cercano a $\pi/2$ los subespacios a comparar de dos patrones van a estar más alejados el uno del otro. Sucediendo lo contrario en caso de que converjan a 0. El caso óptimo es que el valor del ángulo sea exactamente $\pi/2$ convirtiendo a los subespacios en ortogonales. Si los resultados obtenidos son cercanos a $\pi/2$, se podrán obtener resultados favorables al aplicar la clasificación a una nueva imagen con el algoritmo desarrollado y la base de datos utilizada.

2.2.2. Energía acumulada

A pesar de que los datos se encuentran en un alto espacio dimensional R^{1400}, en el caso de la base de datos TCLR, la mayoría de la información se puede representar utilizando unos pocos vectores de base. La

matriz A^{mxn} con valores singulares $\sigma_1 \cdots \sigma_r$ posee una energía acumulada definida por la fórmula

$$\frac{\| \sum_{i=1}^{k} \sigma_i^2 \|}{\| \sum_{i=1}^{r} \sigma_i^2 \|}.$$

En general, supongamos que los primeros k elementos dejan vectores singulares de la matriz de datos, $U_1, ..., U_k$, captura el 97 % de la energía acumulada. En la práctica, se toma un umbral del 95 % cuando se desea diseñar un algoritmo eficiente y se toma un umbral del 99 %, cuando la precisión es la mayor preocupación. Dado que en este caso ambos indicadores son relevantes se utiliza un umbral del 97 %. Estos vectores son los que forman la base para el subespacio de cada patrón (Pacheco, 2011).

El proceso de clasificación se divide en dos etapas: entrenamiento y clasificación.

2.2.3. Entrenamiento

El entrenamiento en general, es la preparación para perfeccionar el desarrollo de una actividad, especialmente para la práctica de un deporte. Adecuando este concepto al entrenamiento de un algoritmo, es el proceso por el cual pasan las imágenes de los patrones de entrenamiento para su posterior utilización en la clasificación. Para el desarrollo de la propuesta de solución deseada se tuvieron en cuenta una serie de factores como las características de una imagen, color (escala de grises) y la resolución. Además, se tomaron los valores matriciales obtenidos luego de aplicar la factorización SVD (Geometría, Luminosidad, Valores Nulos). Se siguieron 3 pasos fundamentales para realizar el entrenamiento del algoritmo de identificación automática de imágenes. Su objetivo es clasificar una imagen de prueba, a partir del análisis y procesamiento de patrones predefinidos. Estos pasos se mencionan a continuación:

- Rasterizar cada imagen de entrenamiento en una matriz de 35x40.
- Linealizar cada imagen, estas conformarán las columnas de la matriz de entrenamiento de cada patrón, en otras palabras, se construye una matriz donde cada elemento columna representa una imagen de entrenamiento para un patrón específico.
- Aplicar la técnica SVD a las matrices obtenidas en el paso anterior, con el objetivo de obtener las bases ortogonales de cada patrón.

Una condición necesaria para realizar el preprocesamiento de la imagen es que esta se encuentre en escala de grises.

Para una mejor comprensión del proceso realizado a continuación se detallan cada uno de los pasos mencionados anteriormente con respecto a la base de datos TCLR. El primer paso fue realizar el preprocesamiento de las imágenes de prueba. Para ello se redimensionaron a una resolución de 35 x 40. Se tomaron estos valores debido a que las imágenes no deben poseer una resolución muy elevada. Debido a que la matriz de entrenamiento quedaría $\mathbb{R}^{1400xn}$ para cada patrón, donde n representa la cantidad de elementos contenidos en la base de datos del patrón seleccionado en este caso n=*20*.

Lo segundo es obtener la matriz de la imagen con el método `ImageToMatrix` de la Biblioteca Accord.Net. El siguiente paso es linealizar la matriz obtenida. Luego, se guarda en una matriz llamada contenedora (B), donde cada elemento columna recrea una imagen de entrenamiento para un patrón específico. Un ejemplo de esto se muestra en la Figura 2.3.

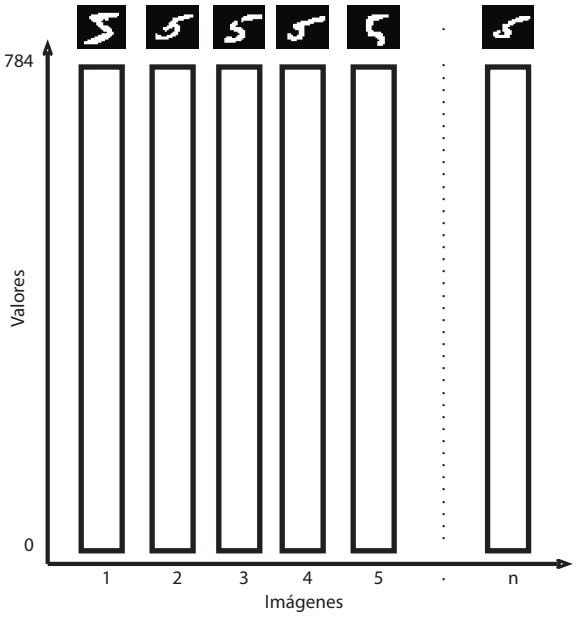


Figura 2.3. Gráfica de la matriz de entrenamiento B con n dígitos 5.

Este proceso se repite para cada patrón de manera que al final se obtiene tantas matrices como patrones existan. El tercer paso es aplicar la factorización SVD a la matriz B de cada patrón para obtener sus valores y vectores singulares. Luego de realizado el preprocesamiento de la imagen, y obtener sus valores SVD, se procede a realizar la clasificación.

2.2.4. Clasificación

Este procedimiento se realizará a partir del cálculo del residuo de la imagen. Este valor expresa cuán cerca está la imagen a comparar, al espacio vectorial de cada patrón. Si se tiene una imagen desconocida z, se debe calcular la distancia euclidiana (residuo) desde el subespacio de z hasta el subespacio de cada patrón, esto se define como:

$$\min_{\alpha} ||U_k\alpha - z||_2,$$

donde $\alpha \in \mathbb{R}^k$ y U_k es la matriz cuyas columnas son $U_1, ..., U_k$. La solución a este problema es equivalente a resolver el mínimo del cuadrado de la norma-2 o norma euclidiana.

$$\begin{aligned}\min_{\alpha} \|U_k\alpha - z\|_2^2 &= \min_{\alpha}(U_k\alpha - z)^t(U_k\alpha - z),\\ &= \min_{\alpha}(\alpha^t U_k^t - z^t)(U_k\alpha - z),\\ &= \min_{\alpha}(\alpha^t\alpha - \alpha^t U_k^t z - z^t U_k\alpha + z^t z).\end{aligned}$$

Se toma la derivada de la última expresión con respecto a α y se iguala a cero obteniendo:

$$\begin{aligned}2\alpha^t - 2z^t U_k &= 0,\\ \alpha^t &= z^t U_k,\\ \alpha &= U_k^t z.\end{aligned}$$

Intuitivamente, $U_k^t z$ es la proyección de z en el subespacio del patrón analizado y la distancia entre ambos es la norma euclidiana del vector residual.

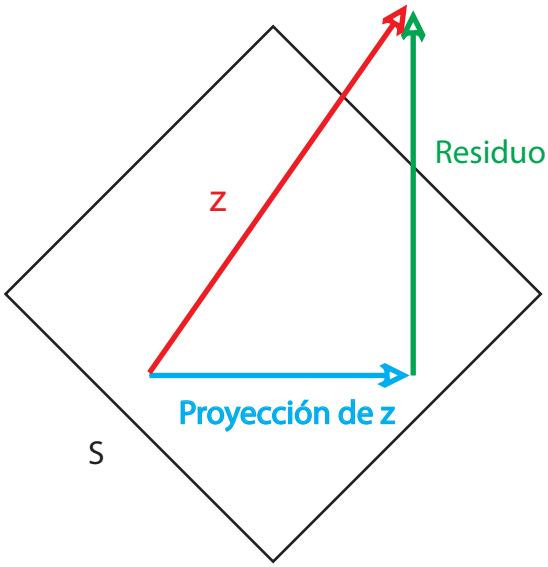


Figura 2.4. El subespacio z, su proyección sobre el subespacio S y el valor del residuo.

en la Figura 2.4 se muestra esta proyección. Luego se sustituyen los elementos en la ecuación inicial

$$\|U_k U_k^t z - z\|_2,$$

se toma factor común z,

$$\|(I - U_k U_k^t)z\|_2.$$

En dicha fórmula, I representa la matriz Identidad. Luego de aplicar SVD a la matriz de entrenamiento de un patrón determinado se obtienen los valores de U, S, V. Se toma la matriz U de dimensión 1400 x 1400,

se le aplica el factor de compresión *k=15* y se obtiene U_k de dimensión 1400 x 15. Para obtener U_k^t se halla la transpuesta de U_k, quedando de dimensión 15 x 1400. La z representa la matriz candidata linealizada, o sea la matriz candidata que va a ser comparada con todos los patrones para ser clasificada. El valor que se obtiene como resultado representa que tan cerca está la imagen candidata al subespacio vectorial del patrón analizado. Esto se realiza para cada patrón a comparar. Para el desarrollo de este algoritmo se toma $(I - U_k U_k^t)$ como L quedando la fórmula $\|L * z\|_2$. Luego de calcular L se procede a terminar de calcular el residuo utilizando la imagen candidata a comparar. Una vez calculado el residuo para cada patrón con z, se comparan los resultados y se toma el menor valor del residuo. Esto significa que es más probable que z esté más cercano al subespacio vectorial del patrón con el menor valor de residuo calculado.

2.3. Diagrama de clases

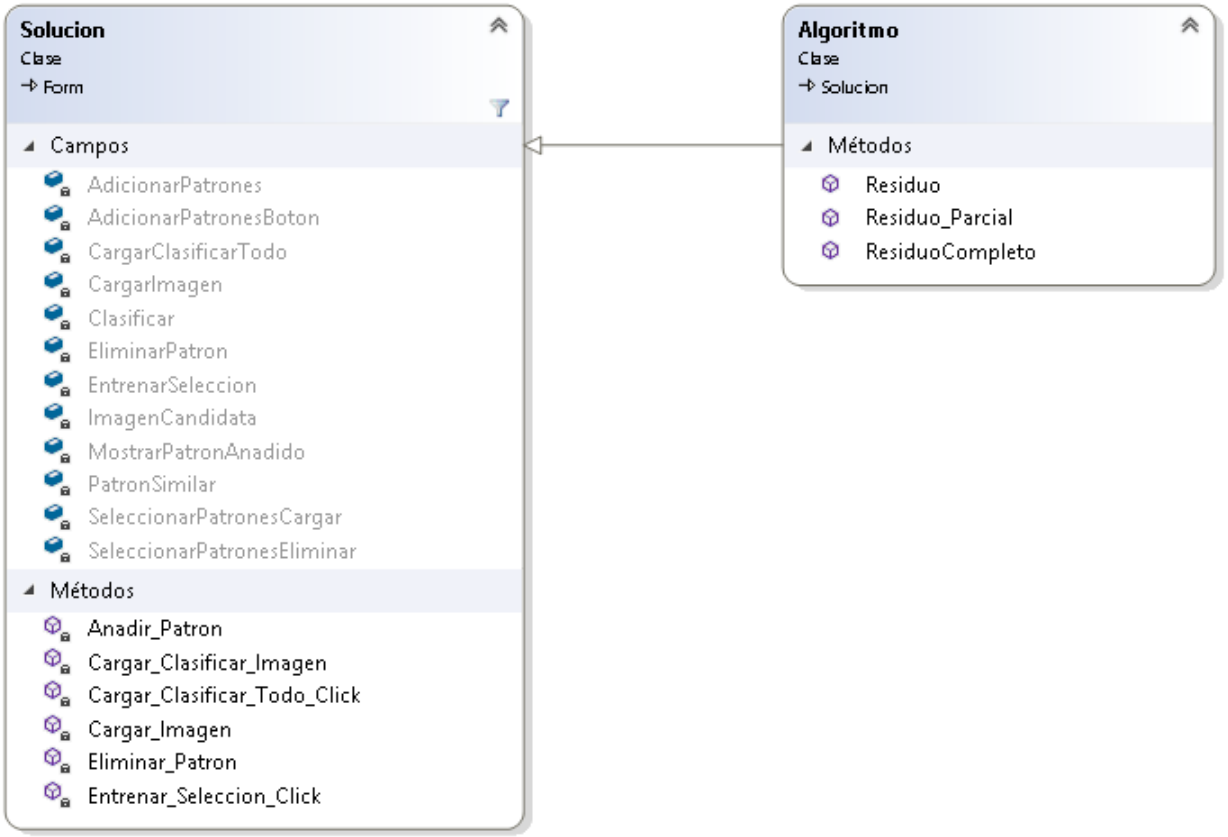


Figura 2.5. Diagrama de clases de la solución.

En la Figura 2.5 se muestra el diagrama de clases de la solución propuesta, describiéndose las clases `Solucion` y `Algoritmo`. La clase `Solucion` incluye los métodos referentes a la interfaz visual del prototipo funcional desarrollado para la solución y el procedimiento final del algoritmo que consiste en identificar el

patrón con menor valor del residuo. La clase `Algoritmo` incluye los métodos que implementan los cálculos para hallar el residuo. En la clase `Solucion` se hacen las llamadas a los métodos de la clase `Algoritmo` en dependencia de la etapa del cálculo de la SVD y la distancia euclidiana para realizar el algoritmo.

CAPÍTULO 3

Validación de la propuesta de solución

En este capítulo se valida la propuesta de solución mediante un prototipo funcional. Se comparan los resultados obtenidos del algoritmo con los obtenidos en la herramienta Matlab.

3.1. Validación del conjunto de datos

El algoritmo propuesto se validó utilizando la base de datos internacional MNIST. Esta base de datos cuenta con un conjunto de dígitos escritos a mano. Posee un conjunto de entrenamiento de 60,000 ejemplares y un conjunto de prueba de 10,000. Es una parte de un conjunto más grande disponible de NIST. Los dígitos están normalizados por tamaño y se han centrado en una imagen de tamaño fijo. Es recomendable para personas que deseen probar técnicas de aprendizaje y métodos de reconocimiento de patrones en datos del mundo real a la vez que gastan un mínimo esfuerzo en el preprocesamiento y el formateo. La base de datos MNIST se construyó a partir de la base de datos especial 1 y 3 de NIST, que contienen imágenes binarias de dígitos escritos a mano. El conjunto de entrenamiento MNIST se compone de 30,000 patrones de las dos bases de datos especiales mencionadas. El conjunto de prueba se compone de 5.000 patrones de estas dos. El conjunto de 60,000 patrones de entrenamiento contiene ejemplos de aproximadamente 250 escritores (LeCun, 1998).

Como el propósito final del algoritmo es incluirlo en el desarrollo de un videojuego interactivo que pretende simular un *test* de visoconstrucción, se hace necesario probar el mismo con una base de datos que contenga imágenes representativas de los patrones que el paciente debe reproducir en ese tipo de test. Como parte de la investigación se elaboró una base de datos con una pequeña muestra de imágenes. La misma contiene imágenes similares a objetos o figuras geométricas. La base de datos TCLR fue realizada por estudiantes de la Universidad de las Ciencias Informáticas (UCI) y consta con 4 patrones (triángulos, cuadrados, lápices y raquetas). Esta base de datos está compuesta por 20 elementos de entrenamiento y 10 de prueba para cada patrón.

3.1.1. Ángulos entre subespacios

A continuación, se muestra el resultado de calcular los ángulos entre subespacios para la base datos MNIST y TCLR. En ambas tablas, los valores de la diagonal son 0, esto se debe a que se calcula la distancia de un subespacio hasta él mismo. Solo aparecen representados los valores por encima de la diagonal debido a que, calcular la distancia de un subespacio Q hasta uno P es la misma que de P hasta Q. En la Tabla 3.1 se muestran los valores del ángulo entre subespacios de todas las bases de los dígitos de MNIST.

Tabla 3.1. Ángulos entre subespacios de todas las bases de los dígitos.

Dígitos	0	1	2	3	4	5	6	7	8	9
0	0	1.5706	1.5698	1.5684	1.5560	1.5569	1.5675	1.5689	1.5638	1.5707
1	_	0	1.5680	1.5703	1.5651	1.5675	1.5704	1.5701	1.5694	1.5692
2	_	_	0	1.5686	1.5698	1.5671	1.5644	1.5685	1.5581	1.5677
3	_	_	_	0	1.5697	1.5680	1.5582	1.5707	1.5623	1.5695
4	_	_	_	_	0	1.5655	1.5686	1.5676	1.5674	1.5655
5	_	_	_	_	_	0	1.5662	1.5700	1.5695	1.5578
6	_	_	_	_	_	_	0	1.5680	1.5686	1.5701
7	_	_	_	_	_	_	_	0	1.5687	1.5699
8	_	_	_	_	_	_	_	_	0	1.5626
9	_	_	_	_	_	_	_	_	_	0

La Tabla 3.2 muestra los resultados de los ángulos entre subespacios de los patrones de las figuras de TCLR.

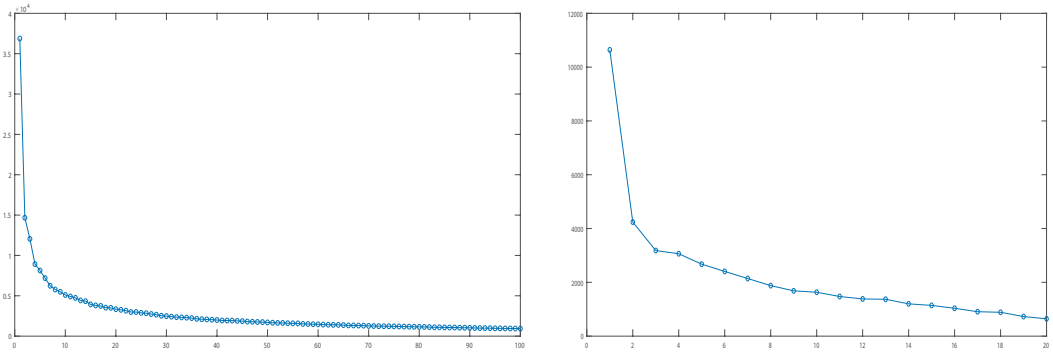
Tabla 3.2. Ángulos entre subespacios de todas las bases de los patrones de las figuras.

Patrones	Cuadrado	Triángulo	Lápiz	Raqueta
Cuadrado	0	1.5682	1.5701	1.5624
Triángulo	_	0	1.5700	1.5681
Lápiz	_	_	0	1.5672
Raqueta	_	_	_	0

En ambas tablas se puede observar que los resultados del cálculo de los ángulos entre subespacios en ambas bases de datos es bastante cercano a $\pi/2$. Esto demuestra que los patrones almacenados en dichas bases de datos cuentan con características geométricas diferentes, por lo que los subespacios de cada patrón están bien separados los unos de los otros.

3.1.2. Energía Acumulada

En la Figura 3.1a se muestran los valores singulares del conjunto de dígitos 3 y en la Figura 3.1b se muestran los valores singulares del patrón de lápices. Los puntos representados en ambas figuras muestran los valores singulares. Mientras más rápido converjan los valores por el eje *y* a 0, los primeros valores por el eje *x* poseerán mayor información intrínseca de la imagen. Esto permite la reconstrucción de la imagen con pocos valores siendo aún reconocible. Este cálculo facilita la optimización del algoritmo, debido que se analiza la cantidad de datos suficiente para representar la imagen sin pérdida de información relevante, de esta manera se obtiene el valor correcto de *k*. Para que la base de datos sea correcta los últimos valores singulares deben de estar cercanos a 0. Si esto no ocurre, la base de datos no está suficientemente poblada para alcanzar buenos resultados en la clasificación. Como se observa en la Figura 3.1a los valores singulares finales están cercanos a 0 por lo que es una base correcta, sin embargo en la Figura 3.1b los valores singulares están lejos aún de 0 por lo que debe poblarse más la base de datos TCLR.



(a) Valores singulares de la base del 3.

(b) Valores singulares del patrón de los lápices.

Figura 3.1. Comparación entre los valores singulares de las bases de los 3 y los lápices.

Para obtener una energía acumulada del 97 % en cada subespacio, con el objetivo de lograr diseñar un algoritmo eficiente con alto nivel de precisión, se realizó el cálculo probando diferentes valores de *k* hasta alcanzar el por ciento requerido.

3.2. Validación del algoritmo

3.2.1. Curvas de residuo

Para la validación del algoritmo se utilizó la herramienta Matlab, en conjunto con la base de datos MNIST. Se realizó el procedimiento con las imágenes de entrenamiento y prueba que brinda dicha base de datos. En general, aumentar el número de imágenes matrices básicas da mejores resultados, pero el error para diferentes clases no necesariamente disminuye (Savas, 2003). Las curvas de residuo muestran gráficamente cómo se comportan los residuos de los elementos de un mismo patrón. Al aproximar un *dígito*

desconocido a todas las bases del espacio superior, se puede afirmar que este dígito se parece a la base de imágenes que tenga menor distancia euclidiana con respecto a él. Por ejemplo al clasificar una nueva imagen, si su distancia euclidiana con respecto a la base del *3* es menor que a las demás, es muy probable que este nuevo dígito sea un *3*. Por lo tanto, debemos identificar qué tan bien se puede representar un dígito desconocido en las 10 bases diferentes. Estaríamos hablando de ejecutar varias veces el procedimiento de clasificación, cada vez con una entrada diferente, con el fin de verificar su capacidad de clasificación. En la Figura 3.2 se representa las curvas de residuo de 107 dígitos *3* en términos de todas las bases. Se utilizaron 500 imágenes de entrenamiento para cada base y un *k=40*. De los 107 casos analizados, el algoritmo clasificó correctamente 99 para un 92.5 % de efectividad. De igual manera clasificó incorrectamente 8 imágenes correspondientes a ejemplos difíciles de distinguir por el ojo humano como un *3*.

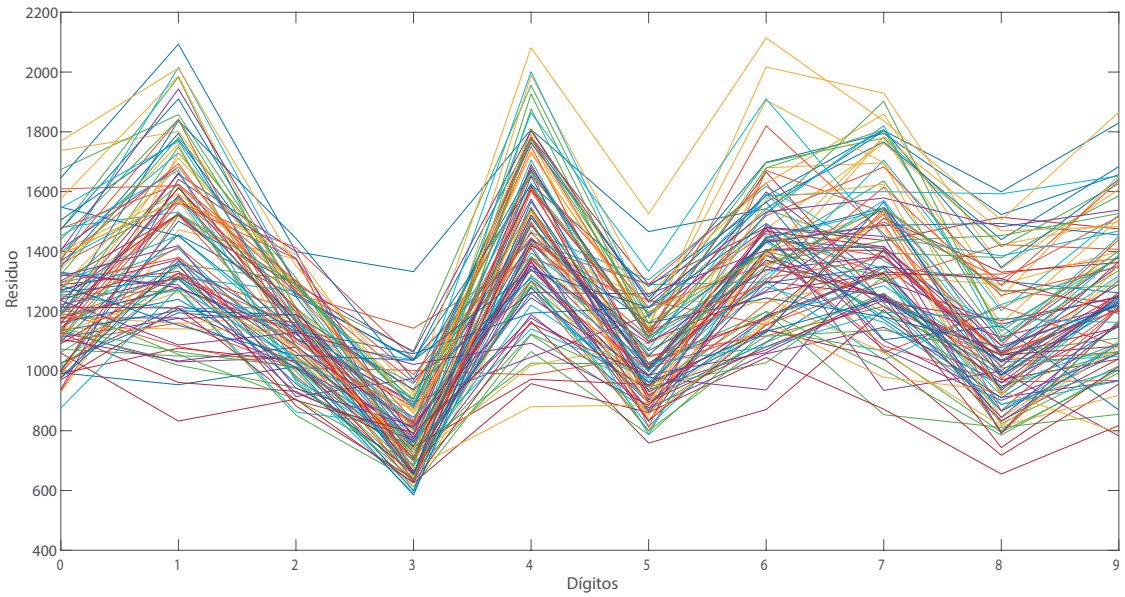


Figura 3.2. Representación de los residuos relativos de dígitos 3.

Como se aprecia en la Figura 3.2 las bases del 3, 5 y 8 son similares. Esto es debido a su geometría por lo que al realizar la clasificación para un dígito 3, mal escrito, puede ser representada como combinación lineal de las bases del 5 u 8. Por lo que los patrones que componen el entrenamiento deben ser lo más distintos posibles para un mejor desempeño del algoritmo. En la Figura 3.3 se pueden observar ejemplos de casos fallidos.

Figura 3.3. Algunas de las imágenes que el algoritmo no identificó como 3.

Como se muestra en los ejemplos anteriores el algoritmo aceptó en la mayoría de los casos utilizando la base de datos MNIST. A continuación, se analizan los resultados obtenidos con la base de datos TCLR.

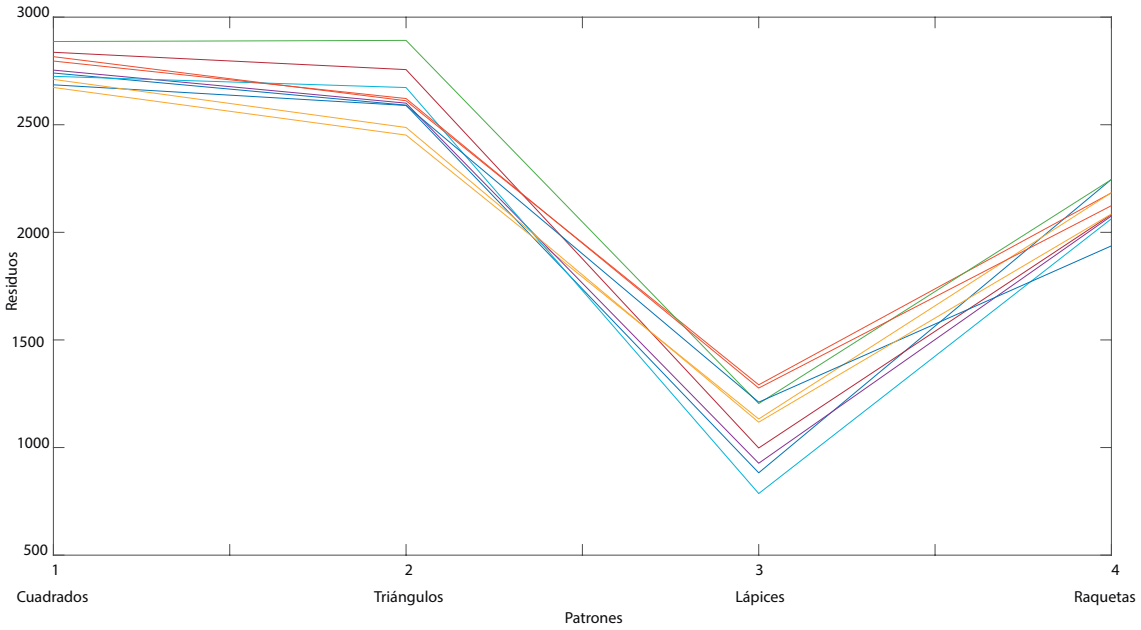


Figura 3.4. Representación de los residuos relativos de los lápices

En la Figura 3.4 se representa las curvas de residuo con 10 imágenes de lápices en términos de todas las bases. Se utilizaron 20 imágenes de entrenamiento para cada base y un *k=15*. El algoritmo clasificó correctamente los 10 casos.

3.2.2. Validación del algoritmo con prototipos.

El uso de cualquier producto de software tiene que estar justificado por las ventajas que brinda. Es por esto que una vez concluida la etapa de desarrollo se le realizan una serie de pruebas para justificar su valía. Los prototipos de *software* son implementaciones realizadas con el lenguaje o las técnicas de programación escogidas para desarrollar la aplicación. Su objetivo es probar el correcto funcionamiento de cada una de sus partes.

Clases de prototipos

La elaboración de prototipos se puede aplicar de manera conveniente ante una situación particular, en lugar de sintetizar todos sus usos en una sola definición. A continuación se muestran varias concepciones para la elaboración de prototipos.

- **Prototipo corregido**: Representa la construcción de un sistema que funciona pero se corrige simultáneamente. En la ingeniería a este enfoque se le denomina: elaboración de una tabla experimental.

- **Prototipo no funcional**: Es un modelo no funcional, configurado para probar ciertos aspectos del diseño.
- **Primer prototipo de una serie**: Involucra la creación de un primer modelo a escala completa de un sistema llamado piloto.
- **Prototipo de características seleccionadas o Prototipo funcional**: Es la creación de un modelo funcional que incluye algunas de las características que tendrá el sistema final.

La clase de prototipo que se utiliza para validar el algoritmo propuesto es el prototipo funcional. Utilizando este modelo se podrá simular el comportamiento del algoritmo en un entorno real.

Prototipo funcional

El prototipo funcional desarrollado para validar visualmente la correcta clasificación de las imágenes por el algoritmo cuenta con dos interfaces principales.

En la Figura 3.5 se muestra la interfaz de entrenamiento del algoritmo. En esta interfaz se brinda la posibilidad de adicionar un nuevo patrón a la base de datos. De igual manera puede adicionar imágenes de entrenamiento al nuevo patrón añadido o a los ya existentes. De manera adicional brinda la opción de eliminar un patrón. La opción *Entrenar Selección* realiza el entrenamiento del patrón seleccionado.

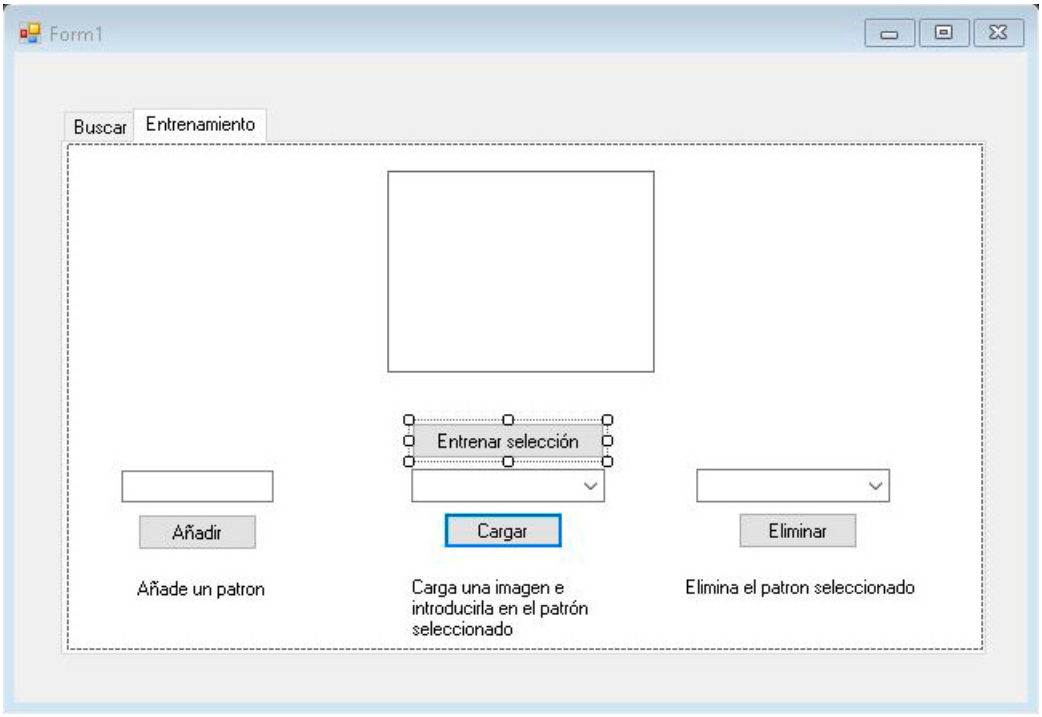


Figura 3.5. Interfaz de entrenamiento del algoritmo.

En la Figura 3.6 se muestra la primera interfaz del prototipo. En ella se realiza la clasificación de las imágenes. Se muestra la imagen a clasificar en el espacio de la izquierda. En el espacio de la derecha se

muestra la imagen del patrón a la cual pertenece. El botón *Cargar y Clasificar Todo* realiza el proceso de entrenamiento y clasificación de una imagen en un solo proceso. Mientras que el botón *Cargar y Clasificar* clasifica una imagen con el entrenamiento realizado previamente en la interfaz de la Figura 3.5.

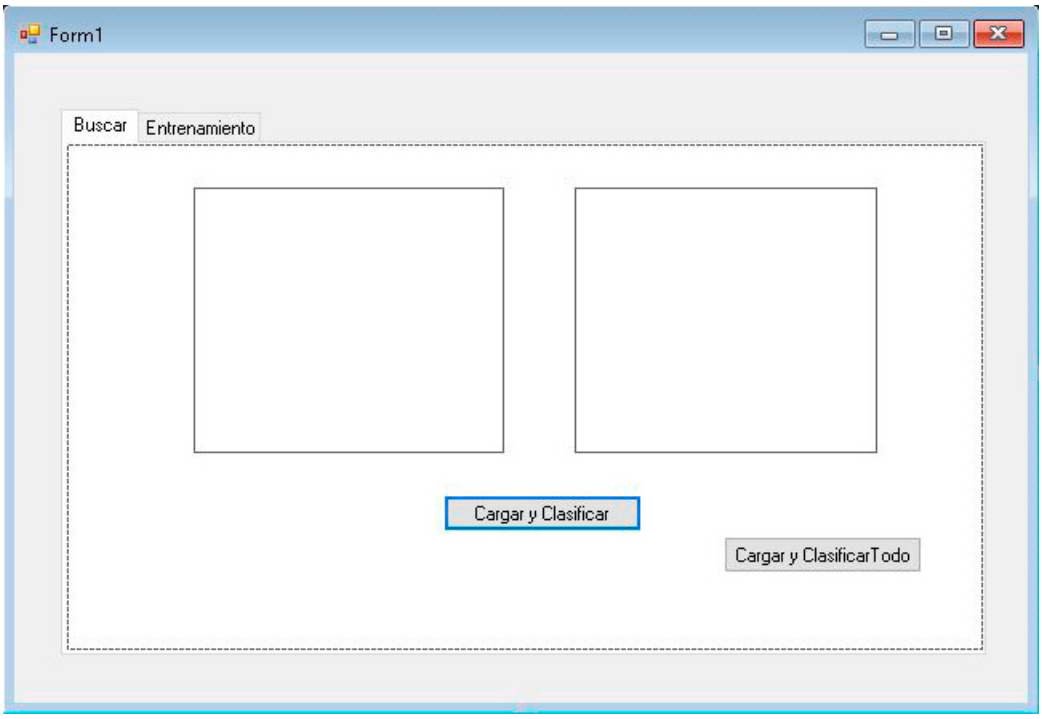


Figura 3.6. Interfaz de clasificación.

El prototipo funcional se realizó con el objetivo de comprobar de forma visual que el algoritmo propuesto clasifica correctamente las imágenes de prueba. Además permite la integración de su codificación con una interfaz visual. Esto permite que en desarrollos posteriores se pueda integrar en un videojuego interactivo o en otro tipo de software que requiera clasificar imágenes visualmente.

3.3. Reutilización del Algoritmo

Para reutilizar las funciones implementadas en un videojuego interactivo, se separaron las funcionalidades correspondientes al funcionamiento del algoritmo, de las funciones correspondientes a la interfaz visual desarrollada. Las funciones relacionadas con SVD y cálculo del residuo se encuentran en la clase `Algoritmo`. Esta utiliza funciones de las bibliotecas Accord.net y MathNet.

Código fuente 3.1. Referencias a las bibliotecas utilizadas

```
using Accord.Math;
using MathNet.Numerics.LinearAlgebra.Double;
```

A continuación se describen los métodos de la clase `Algoritmo`: En el Código fuente 3.2 se muestra el método en el cual se calcula *L*. El parámetro de entrada es la matriz de entrenamiento `matrizImagenes` y el parámetro de salida `residuoParcial` que devuelve la matriz *L*.

Código fuente 3.2. Residuo Parcial *L*

```
public static double[,] Residuo_Parcial(double[,]
    matrizImagenes)
{
    //matrizImagenes es la matriz de entrenamiento
    DenseMatrix matrixImagenes = DenseMatrix.OfArray(
        matrizImagenes);
    var u = matrixImagenes.Svd().U;
    //factor de compresión k
    int k = 15;
    double[,] UkA = new double[u.RowCount, k];
    for (int b = 0; b < k; b++)
    {
        for (int a = 0; a < u.RowCount; a++)
        {
            UkA[a, b] = u[a, b];
        }
    }
    double[,] UkTA = UkA.Transpose();
    double[,] resultado = Accord.Math.Matrix.Dot(UkA, UkTA);
    double[,] Inversa = Accord.Math.Matrix.Identity(u.RowCount)
        ;
    double[,] residuoParcial = resultado.Subtract(Inversa);
    return residuoParcial;
}
```

En el Código fuente 3.3 se calcula el resto de la fórmula del residuo, sus parámetros de entrada son el `residuoParcial` previamente calculado y la matriz de la imagen a clasificar linealizada `matrizNewLineal`. Su parámetro de salida es `residuo`, la distancia euclidiana entre el patrón seleccionado y la imagen candidata .

Código fuente 3.3. Residuo Parcial 2

```
public static double Residuo(double[,] residuoParcial, double
    [,] matrizNewLineal)
{
```

```
        //calcular segunda parte del residuo
        //residuo_Parcial es el la matriz que devuelve el método
            Residuo_Parcial
        //matrizNewLineal es la matriz de la imagen a clasificar
            linealizada
        double residuo = 0;
        double[,] multiplicacion = Accord.Math.Matrix.Dot(
            residuoParcial, matrizNewLineal);
        residuo = Norm.Euclidean(multiplicacion);
        return residuo;
    }
```

En el Código fuente 3.4 se unifican los dos primeros métodos. Se realiza todo el procedimiento para el cálculo del residuo en un solo método, sin necesidad de guardar datos en disco. De realizarse el proceso completo, entrenamiento y clasificación, este demora más y consume más memoria temporal. De realizarse por partes, calcular y guardar L en disco primero y luego clasificar, se consume espacio en disco, pero el proceso de clasificación se realiza más rápido. Cuando se realiza la clasificación de una nueva imagen el valor almacenado de L no varía, por lo que solamente se carga el resultado de este. Se proponen las dos variantes para que el interesado seleccione la que más se corresponda a sus requerimientos.

Código fuente 3.4. Residuo Total

```
    public static double ResiduoCompleto(double[,] matrizImagenes,
        double[,] matrizNewLineal)
    {
        // matrizImagenes son los valores de la matriz de
            entrenamiento
        // matrizNewLineal es la matriz de la imagen a clasificar
            linealizada
        DenseMatrix matrixImagenes = DenseMatrix.OfArray(
            matrizImagenes);
        var u = matrixImagenes.Svd().U;
        //comprimirla
        int k = 15;
        double[,] UkA = new double[u.RowCount, k];

        for (int b = 0; b < k; b++)
        {
            for (int a = 0; a < u.RowCount; a++)
```

```
            {
                UkA[a, b] = u[a, b];
            }
        }
        double[,] UkTA = UkA.Transpose();
        double[,] resultado = Accord.Math.Matrix.Dot(UkA, UkTA);
        double[,] Inversa = Accord.Math.Matrix.Identity(u.RowCount)
            ;
        resultado = resultado.Subtract(Inversa);
        double residuoCompleto = 0;
        double[,] multiplicacion = Accord.Math.Matrix.Dot(resultado
            , matrizNewLineal);
        residuoCompleto = Norm.Euclidean(multiplicacion); ;
        return residuoCompleto;
    }
```

Conclusiones

Con la realización de este trabajo se arribó a la siguiente conclusión:

- El algoritmo propuesto resulta adecuado cuando los distintos patrones que conforman la base de datos de entrenamiento no tienen parecidos entre sí, de lo contrario esto tendría como resultados falsos positivos. Esta afirmación se verifica con los residuos mostrados en la Figura 3.2.

Recomendaciones

Se recomienda para el desarrollo de futuros trabajos:

- Utilizar *Sparse* para el almacenamiento de las matrices con el objetivo de optimizar el algoritmo.
- Poblar con más elementos la base de datos TCLR.
- Utilizar el algoritmo en un videojuego interactivo para la rehabilitación en el área de la visoconstrucción.

Glosario

binarización Es una técnica que consiste en la realización de un barrido en la matriz de la imagen digital, por medio de bucles o recursividad, con el fin de que el proceso produzca la reducción de la escala de grises a dos únicos valores (0,1) (Magro, 2013).. 6

profundidad de color Es el número de bits utilizados para almacenar información sobre el color de cada píxel en una imagen digital.. 12

ruido Efectos espurios que pueden presentarse en una imagen a consecuencia del proceso de captura, digitalización, transmisión o almacenamiento (*Reducción del ruido en una imagen digital* 2007).. 6

ventanas Elementos es los que se subdividen una imagen los cuales mantienen siempre la relación de aspecto 1:1.. 4

Acrónimos

ANN Redes Neuronales Artificiales. 4–6

DWT Transformada de Wavelet Discreta. 7

IDE Entorno de Desarrollo Integrado. 10

SVD Descomposición en Valores Singulares. 4, 9–14, 16–18, 20–22, 29

UCI Universidad de las Ciencias Informáticas. 24

VERTEX Centro de Entornos Interactivos 3D. 1, 2

Referencias bibliográficas

Alpaydin, Ethem (2014). *Introduction to machine learning*. MIT press (vid. pág. 5).

Beltrán, Manubens J. M. (s.f.). «Localización de las lesiones». En: (vid. pág. 2).

Burger, W. y M. J. Burge (2009). *Principles of digital image processing*. Springer.

Carrasco-Corona, E. et al., (2012). «Sistema de verificación biométrico vascular». Tesis de mtría. Instituto Politécnico Nacional Escuela Superior de Cómputo (vid. pág. 9).

Castro, L. R. y S. M. Castro (1995). «Wavelets y sus Aplicaciones». En: *I Congreso Argentino de Ciencias de la Computación* (vid. pág. 7).

Cheung, L. y C. Medina (2013). «Implementación y análisis de un detector de manos basado en visión artificial». En: *Universidad Tecnológica de Panamá* (vid. pág. 4).

Correa-Madrigal, O. (2015). «Modelo de generación procedural de contenido para la rehabilitación de la agudeza visual con videojuegos». Tesis doct. Universidad de las Ciencias Informáticas (vid. pág. 1).

Delgado-Rodríguez, M. (2012). «Extracción automática de caras en imágenes captadas con móviles Android». En: *UPC* (vid. pág. 5).

Eldén, L. (2007). *Matrix Methods in Data Mining and Pattern Recognition*. Ed. por Nicholas J. Higham. Society for Industrial y Applied Mathematics (vid. págs. 9, 13).

Enriquez-Vasquez, R. A. (2013). «Sistema Móvil de Recuperación de Información Visual Utilizando Formas y Colores para el Reconocimiento de Obras Arquitectónicas». En: *Universidad de las Américas Puebla* (vid. pág. 8).

Flores, W. G. (2015). «Reconocimiento de objetos en fotografías». En: *CINVESTAV-LTI* (vid. pág. 7).

García, P. P. (2013). «Reconocimiento de imágenes utilizando redes neuronales artificiales». En: *Facultad de Informatica, Universidad Complutense de Madrid*, pág. 5 (vid. págs. 5, 6).

García-Villanueva, M. y Salvador Ramírez-Zavala (2013). «Integración de OpenCv en una Raspberry Pi: Sistema de Detección de Rostro vía WEB». En: (vid. pág. 5).

Gil-Rodríguez, J. L. (2008). «Estado Actual de la Representación y Análisis de Textura en Imágenes». En: *La Habana: CENATAV. Recuperado de http://catalogo. bnjm. cu/cgi-bin/koha/opac-detail. pl* (vid. pág. 8).

Golub, G. H. y C. F. Van-Loan (1996). «Matrix computations. 1996». En: *Johns Hopkins University, Press, Baltimore, MD, USA*, págs. 374-426 (vid. págs. 13, 17).

Gómez, F. (2018). «8 características de la imagen digital que debes conocer». En: *Deusto Formación*. URL: https://www.deustoformacion.com/blog/diseno-produccion-audiovisual/8-caracteristicas-imagen-digital-que-debes-conocer (vid. pág. 10).

Hu, Weiming, Wei Hu y S. Maybank (2008). «Adaboost-based algorithm for network intrusion detection». En: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.2, págs. 577-583 (vid. pág. 5).

Kuruvilla, F. G. et al., (2002). «Dissecting glucose signalling with diversity-oriented synthesis and small-molecule microarrays». En: *Nature* 416.6881, pág. 653 (vid. pág. 14).

LeCun, Y. (1998). «The MNIST database of handwritten digits». En: *http://yann. lecun. com/exdb/mnist/* (vid. pág. 24).

Lienhart, R. y J. Maydt (2002). «An extended set of haar like features for rapid object detection». En: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 1. IEEE, págs. I-I (vid. pág. 5).

Magro, R. (2013). «Binarización de imágenes digitales y su algoritmia como herramienta aplicada a la ilustración entomológica». En: *Boletín de la SEA* 53, págs. 443-464 (vid. pág. 36).

Olabe, X. B. (1998). «Redes neuronales artificiales y sus aplicaciones». En: *Publicaciones de la Escuela de Ingenieros 101pp* (vid. págs. 5, 6).

Ortega-Gonzalez, E. V. (2007). «Una técnica para el análisis de similitud entre imágenes». Tesis doct. Instituto Politécnico Nacional Centro de Investigación en Computación (vid. pág. 9).

Pacheco, J. I. (2011). «A comparative study for the handwritten digit recognition problem». Tesis doct. California State University, Long Beach (vid. pág. 19).

Pérez, C., D. G. Aguilera y A. L. Muñoz (2003). «Estudio de viabilidad del uso de imágenes comprimidas en procesos de clasificación». En: *Universidad de Salamanca* (vid. pág. 16).

Poveda, Enrique A Lopez (2013). *Las técnicas de análisis de imagen tienen aplicaciones en astronomía, teledetección, y también en neurociencias.* [Online; accessed 15. May 2018]. Universidad de Salamanca. URL: http://audiolab.usal.es/Documentos/Docencia/Master%20Neurociencias.pdf (vid. pág. 1).

Procesamiento de imagenes (2018). http://www.vicomtech.org/t1/e7/procesamiento-de-imagenes (vid. pág. 10).

Reducción del ruido en una imagen digital (2007). Departamento de Ingeniería electrónica, Telecomunicación y Automática. (vid. pág. 36).

Romo, H. A., F. Ramírez y R. Valdivieso (2007). «Detección del bacilo mycobacterium tuberculosis mediante reconocimiento de patrones». En: *Avances en sistemas e informatica* 4.3 (vid. pág. 7).

Savas, B. (2003). «Analyses and tests of handwritten digit recognition algorithms». En: *LiTH-MAT-EX-2003-01, Link ping University, Department of Mathematics* (vid. págs. 18, 26).

Socorro-Borges, M. A. (2017). «Propuesta de algoritmo para la identificación de perturbaciones que afectan la calidad de la energía eléctrica distribuida usando Transformada S y Máquinas de Soporte Vectorial.» Tesis doct. Universidad de La Habana (vid. pág. 7).

Strang, G. (2006). *Linear Algebra and Its Applications*. Thomson, Brooks/Cole. ISBN: 9780030105678. URL: https://books.google.com.cu/books?id=8QVdcRJyL2oC (vid. pág. 13).

Trujillano-Cabello, J. et al., (2005). «Redes neuronales artificiales en Medicina Intensiva. Ejemplo de aplicación con las variables del MPM II». En: *Med. Intensiva* 29.1. ISSN: 0210-5691 (vid. pág. 6).

Tversky, A. (1977). «Features of similarity». En: *Psychological review* 84.4, pág. 327 (vid. págs. 10, 11).

Waldemar, P. y T. A. Ramstad (1996). «Image compression using singular value decomposition with bit allocation and scalar quantization». En: *Proceedings of NORSIG Conference*, págs. 83-86 (vid. pág. 9).

Yreta, A. y M. Alberto (2006). «Cómputo de la similitud entre figuras geométricas». Tesis doct. Instituto Politécnico Nacional. Centro de Investigación en Computación (vid. pág. 10).

Generado con LaTeX: 25 de junio de 2018: 2:13pm