INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERIA MECANICA Y ELECTRICA UNIDAD CULHUACAN

SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION

AUTENTICACION DE IMAGENES USANDO IMAGE HASHING

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIAS

DE INGENIERIA EN MICROELECTRONICA

P R E S E N T A:

RICARDO ANTONIO PARRAO HERNANDEZ*

TUTORES:

Dra. MARIKO NAKANO MIYATAKE

Dr. BRIAN M. KURKOSKI

MEXICO D.F Agosto 2011

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 11:00 horas del día 19 del mes de septiembre del 2011 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de SEPI-ESIME-CULHUACAN para examinar la tesis titulada:

"Autenticación de Imágenes Usando Image Hashing"

Presentada por el alumno:

Parrao	**Hernández**	**Ricardo Antonio**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: | B | 0 | 9 | 1 | 7 | 7 | 3 |

aspirante de:

MAESTRÍA EN CIENCIAS DE INGENIERÍA EN MICROELECTRÓNICA

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dra. Mariko Nakano Miyatake

Dr. Brian M. Kurkoski

Dr. Volodymyr Ponomaryov

Dr. Héctor Manuel Pérez Meana

Dr. Manuel Cedillo Hernández

INSTITUTO POLITECNICO NACIONAL
ESTADOS UNIDOS MEXICANOS
S.E.P.
SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Gonzalo Isaac Duchen Sánchez

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

México

En la Ciudad de México el día 28 del mes Septiembre del año 2011, el (la) que suscribe C. Ricardo Antonio Parrao Hernandez alumno (a) del Programa de Maestria en Ciencias de Ingenieria en Micaroelectronica con número de registro B091773, adscrito a SEPI ESIME CULHUAN, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dra. Mariko Nakano Miyatake y cede los derechos del trabajo intitulado Autenticaccion de Iamgenes usando Image Hashing, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección mariko@calmecac.esimecu.ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Ricardo Parrao Hernandez

Nombre y firma

Agradecimientos

A mis padres Lourdes Hernández y Ricardo Parrao por todo su apoyo y cariño. A mi familia que sin ellos este trabajo no pudo haber sido realizado.

A Jessica Sánchez Cruz por todo su cariño, paciencia, apoyo y por no dejarme rendirme en momentos difíciles.

A mis profesores Dra. Mariko Nakano y Dr. Brian Kuskoski por el apoyo en este trabajo que es fruto de su asesoría y consejos.

Al Consejo Nacional de Ciencias y Tecnología (CONACYT) y al Instituto Politécnico Nacional (IPN). Por el apoyo brindado para la realización de este proyecto.

Resumen

En los últimos años el uso de imágenes digitales ha crecido enormemente al mismo tiempo las herramientas para manipular archivos multimedia, como Paint, Photoshop, Corel Draw, etc, también están aumentando y se han vuelto muy fáciles de usar. Para garantizar la confiabilidad, las técnicas de autenticación de imágenes, como Image hashing y marcas de agua han surgido para verificar la integridad del contenido y evitar la falsificación.

Tradicionalmente los problemas de integridad de datos son tratados por Hashes criptográficos o funciones de autenticación de mensajes, que son clave-dependientes y sensibles a todos los bits del mensaje de entrada. Como resultado, la integridad del mensaje sólo se puede validar si cada bit del mensaje no ha cambiado.

La definición de la autenticidad multimedia no es tan directa. Por lo que este tipo de hash no se puede utilizar para autenticar el contenido, porque una función hash de imagen tiene en cuenta los cambios en el dominio visual. En particular, una función de Image Hashing debe tener la propiedad de que sí dos imágenes que tienen el mismo contenido, en terminos de la visión humana, deben generar un valor Hash similar, incluso si las imágenes tienen diferentes representaciones digitales. Por lo tanto, una validadción bit a bit ya no es adecuada para autenticar los datos multimedia y una herramienta de autenticación multimedia que valide el contenido es más deseada. Una aplicación inmediatamente obvia para Image Hashing es la identificación y búsqueda de imágenes en grandes bases de datos. Varias otras aplicaciones han sido identificadas recientemente en la autenticación de contenido, marcas de agua, y en la lucha contra la piratería.

Este trabajo presenta el uso del algoritmo de normalización de imágenes y la función hash de descomposición SVD para generar el vector hash de las imágenes digitales. La descomposición SVD es una herramienta importante de álgebra lineal, que se utiliza a menudo en el reconocimiento de patrones, marcas de agua digitales y otros campos de procesamiento de señales, por otro lado la normalización imágenes es un algoritmo que ha demostrado ser robusto a diferentes tipos de ataques geométricos tales como: rotación, escalamiento y Shearing. Utilizando el algoritmo de normalización de imagenes como pre-procesamiento se puede aumentar la robustez del valor hash como se muestra por medio de resultados numéricos.

Para medir que tan similares son dos imagen digitales, elegimos la distancia de Hamming normalizada, con respecto a la longitud (L), entre los valores hash binarios, como una métrica de rendimiento. Esta distancia se espera que sea cercana a 0 para imágenes similares y cercana a 1 para imágenes distintas. A medida que más partes de una imagen cambien, la imagen manipulada y la imagen original serán más disimilares. Este trabajo propone a partir de resultados numéricos el uso de 0,1 en términos de distancia de Hamming como umbral.

Abstract

In the last few years the use of digital image has a grown tremendously at the same time tools to manipulate digital multimedia are also increasing such as Paint, Photoshop, Corel Draw, etc., and became very easy to use. To ensure trustworthiness, image authentication techniques, such as image hashing, watermarking have emerged to verify content integrity and prevent forgery.

Traditionally data integrity issues are addressed by cryptographic hashes or message authentication functions, which are key-dependent and sensitive to every bit of the input message. As a result, the message integrity can be validated when every bit of the message is unchanged. The definition of authenticity for multimedia is not as straightforward. So this type of hashing cannot be use for authenticate content because an image hash function takes into account changes in the visual domain. In particular, a perceptual image hash function should have the property that two images that look the same to the human eye map to the same hash value, even if the images have different digital representations. Therefore bit-by-bit verification is no longer a suitable way to authenticate multimedia data and a media authentication tool that validates the content is more desired. An immediately obvious application for a perceptual image hash is identification/search of images in large databases. Several other applications have been identified recently in content authentication, watermarking, and anti-piracy search.

This work introduces the use of image normalization algorithm and SVD decomposition Hash function to generate the Hash vector for digital images. SVD decomposition is an important linear algebra tool, which is often used in pattern recognition, digital watermarking and other signal processing fields, on the other hand image normalization is an algorithm that has been shown to be robust to different kinds of geometric attacks such like, rotation, scaling and flipping. Using the image normalization algorithm as preprocessing step we can increase the robustness of the hash value as it shown by numerical results.

To measure the performance of image hashing, we choose the Hamming distance between the binary hashes, normalized with respect to the length (L) of the hash as a performance metric. This is expected to be close to 0 for similar images and close to 1 for dissimilar ones. As more parts of a picture are changed, the manipulated image and the original image become more dissimilar. This work proposes by numerical results the use of 0,1 in terms of Hamming distance as a threshold.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Durante la era de la información, el intercambio de contenido digital ha tenido un incremento muy considerable en los últimos años, de la misma manera herramientas para manipular dicho contenido como Photoshop, Corel Draw, Illustrator, etc, se han vuelto mas populares y fáciles de utilizar, lo que ha aumentado el uso de estas herramientas para manipular imágenes digitales para fines ilegales o desacreditación de personas. Para poder asegurar la autenticidad del contenido multimedia técnicas como marcas de agua e $Image\,hashing$ han surgido para verificar la integridad y prevenir la piratería. Tradicionalmente la integridad de la información se ha realizado a través de funciones de autentificación de mensajes o Hashes criptográficos, que son llave dependientes y sensibles a cambios de bit. La integridad del mensaje transmitido solo puede verificarse cuando cada uno de los bits no ha sido modificado[4]. Esta sensibilidad es usualmente aplicada a la autentificación de mensajes de textos.

Un $Image\,Hash$ es una firma digital basada en el contenido de la imagen. Para generar el $Hash$ una llave es usada para extraer algunas características del contenido de la imagen, estas características son procesadas para generar el valor $Hash$ final.

Entre las aplicaciones que se encuentran para este tipo de firmas digitales[5] son:

- Filtros para compartir archivos: Por filtrar nos referimos a una intervención activa en la distribución de contenido. Un ejemplo de esto es $NAPSTER$ que tuvo una disputa legal con la industria musical, haciendo que $NAPSTER$ introdujera el mecanismo de filtrado de audio que restringía las descarga de canciones protegidas por derechos de autor. La demanda de estos mecanismos de filtrado crece mientras la gente descarga imágenes y videos protegidos por derechos de autor a través de servicios de intercambio de archivos. Image hashing es un buen candidato para ser un buen mecanismo de filtrado[6][7]

- Índices automáticos de librerías multimedia: Hoy en día muchos usuarios tienen un librería multimedia con cientos y algunas veces miles de archivos multimedia (archivos video, imágenes y audio), cuando estos archivos son obtenidos de diferentes fuentes como CD, imágenes escaneadas ó a través de servicio de intercambio de archivos, usualmente estas librerías no se encuentran bien organizadas. Identificando estos archivos con este tipo de firmas digitales pueden ser automáticamente identificados con el metadato correcto, permitiendo una organización sencilla por ejemplo a través de autor, álbum o genero.

- Relación de contenido: La relación de contenido es un término general para aplicaciones de consumo donde el contenido esta relacionado de alguna forma con información adicional y de soporte. El reconocimiento de audio en teléfonos móviles es un ejemplo típico. Propongamos la siguiente situación: Cuando se escucha una canción en la radio y por diferentes razones no podemos identificar el nombre de la canción ó alguna otra información referente a ella, pero en la canción se encuentra embebido un valor hash, por lo qué podemos acceder a una base

de datos donde a través de dicho valor hash podamos obtener toda la información necesaria de la canción reproducida[6] . Una aplicación similar puede aplicarse para cualquier tipo de contenido.

- Autenticación: Hoy en día el software para manipular contenido digital se ha vuelto muy sencillo de utilizar y la autenticación verifica la originalidad del contenido detectando alguna modificación maliciosa. Respecto a la autenticación multimedia, el asunto clave es proteger el contenido del mensaje, no sú particular representación del contenido . Los métodos de autenticación de contenido pueden clasificarse en métodos basados en firmas digitales [8][9] y métodos basados en marcas de agua[10]. En términos generales los métodos basados en firma digital son mas robustos que los métodos basados en marcas de agua. Sin embargo los métodos basados en firma digital necesitan un canal adicional o almacenarse para checar la autenticidad del contenido; los métodos basados en marcas de agua no lo necesitan.

1.1. Objetivo

Desarrollar un algoritmo de Image Hashing que permita la autenticación de imágenes digitales, utilizando el método de la descomposición SVD para la extracción de características y el algoritmo de normalización de imágenes para proporcionar robustez ante ataques geométricos tales como rotación, escalamiento y shearing.

1.2. Justificación

En los últimos años ha habido un incremento en el uso de aplicaciones para la edición de material multimedia como Photoshop, Corel Draw, Movie Maker, Paint, se han vuelto mas fáciles en su uso ya que actualmente cualquier usuario puede manipular dicho material, logrando resultados donde no se pueda notar que el material multimedia haya sido manipulado.

Datos de la AMIPCI(Asociación Mexicana de Internet) revela que en los últimos años en México ha habido un incremento considerable de usuarios que se dedican al intercambio de información digital ya sea de video o de imágenes digitales publicándolas en sitios de internet como redes sociales(Facebook, Google+), blogs(Blogger, typepad), paginas personales, por lo que se vuelve una necesidad que exista algún método que nos permita saber la autenticidad de dicho material ya que estos pueden ser utilizados en casos legales como evidencia, además que la modificación de dicho material puede afectar no solo en el ámbito legal si no en el ámbito personal, desencadenando un posible escandalo que afecte en la reputación y desacredite tambien a un individuo ó empresa.

1.3. Metodología

Se realizarán pruebas sobre el algoritmo propuesto por Kozat [2] donde se utiliza la descomposición SVD como método de extracción de características para comprobar su robustez ante diferentes tipos de ataques tales como rotación, compresión JPEG, escalamiento, shearing, ruido Gaussiano.

Posteriormente se usara el algoritmo de normalización de imágenes [11] para incrementar la robustez de dicho algoritmo y se comparara con el algoritmo de Kozat para comprobar las mejoras en rendimiento en términos de la distancia de haming.

1.4. Contribución

Este trabajo justifica un mejor desempeño del algoritmo propuesto por Kozat utilizando el algoritmo de normalización de imágenes como pre-procesamiento. Dando como resultado un mejor desempeño en términos de robustez dieminuyendo la distancia de Hamming. Así como de la

utilización de cuantización y compresión del valor Hash para convertirlo a un vector Hash binario, debido a que el algoritmo propuesto por Kozat da como resultado un vector Hash de números reales.

Además este trabajo presenta la propuesta de umbral para la autentificación de imágenes a partir de observaciones experimentales obteniendo como valor umbral 0,1 en términos de la distancia de Hamming.

1.5. Organización de Tesis

En el capítulo 2 se revisara una descripción de los conocimientos previos y de las técnicas existentes que se han aplicado para resolver el problema de Image Hashing. En el capítulo 3 se mostrara la descomposición SVD y la normalización de imágenes que son dos conceptos fundamentales en este trabajo y muestra la teoría detrás de la solución propuesta. En el capítulo 4 se dará la descripción del sistema propuesto, en el capítulo 5 los resultados obtenidos y en el capítulo 6 las conclusiones y el trabajo futuro.

Capítulo 2

Estado del arte

En este capítulo se muestra el estado del arte así como las generalidades del problema de Image hahsing, y las investigaciones relevantes que se han hecho al respecto para resolver este problema

2.1. Imagen digital

Es una representación bidimensional de una fotografía, la representación mas usual es una matriz bidimensional donde cada uno de las casillas representan un pixel que es la representación mas pequeña de un color o tonalidad como se muestra en la figura 2.1

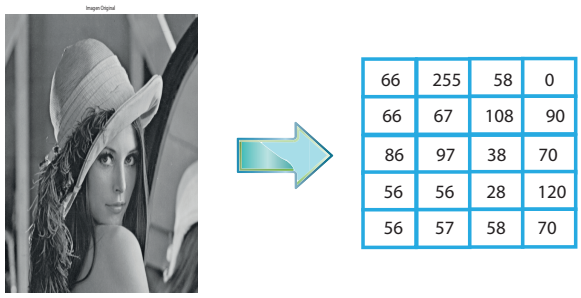


Figura 2.1: Representación de una imagen digital.

2.2. Función hash

Una función hash es un procedimiento bien definido o una función matemática que convierte una gran cantidad de información a un dato de menor dimensión, que usualmente sirve para indexar, el valor que regresa la función hash se le conoce como valor Hash, checksum o solamente hash.
Una función hash en términos generales debe tener las siguientes características:

- Bajo costo: El costo computacional debe ser lo suficientemente pequeño para hacer que la solución basado en hash sea más eficiente que otro tipo de enfoque.
- Determinista: Para una entrada conocida siempre debe de dar la misma salida.
- Uniformidad: Las salidas deben ser lo más uniforme posible sobre el rango de salida, donde cada valor posible en la salida debe tener la misma probabilidad.

Dentro de las funciones hash podemos clasificar en dos grandes grupos hashes criptográficos y hashes basado en contenido.

2.3. Hash criptográfico

Una función hash criptográfica $H(X)$ mapea un objeto X(usualmente grande) a un valor hash(usualmente pequeño), esto permite comparar dos objetos grandes X y Y, sólo comparando sus valores hash $H(X)$ y $H(Y)$, una igualdad matemática de $H(X)$ y $H(Y)$ implica que X y Y sean iguales con poca probabilidad de error. Usando funciones hash criptográficas podemos tener un método eficiente para saber si un objeto X esta contenido en set grade $Y = Y_i$, en lugar de comparar X con todo el set Y es suficiente con comparar el set de valores hash $\{h_i = H(Y_i)\}$ con el valor $H(X)$, este método es mas eficiente porque la cantidad de bits que se deben comparar es menor. Las propiedades de las funciones hash criptográficas son:

- Un solo sentido: dado $H(X)$ debe ser difícil de encontrar el objeto original X, y dado X y $H(X)$ es difícil encontrar un objeto $X' donde X' \neq X$ tal que $H(X') = H(X)$.
- Libre de colisiones: Es difícil encontrar dos objetos X y Y tales que sus valores hash sean iguales $(H(X) = H(Y))$.
- Uso de llaves: Sin el conocimiento de una llave K, es casi imposible determinar $HK(X)$ de cualquier objeto X.

Aunque este tipo de esquemas son muy útiles debido a que el mensaje debe de estar intacto para poder generar el mismo valor hash. Sin embargo para el problema de Image hashing este método falla cuando se utilizan este tipo de funciones hash criptográficas. Para contenido multimedia no estamos interesados en igualdades matemáticas sino en igualdad de contenido, ya que existen manipulaciones sin alteración de contenido como compresión, realces, conversión analógica-digital, ataques geométricos, etc. Las funciones hash criptográficas no podrían procesar dichas manipulaciones como el archivo original. Este problema podría resolverse si las funciones hash criptográficas tuvieran un comportamiento lineal por ejemplo si contenido perceptualmente similar generara al menos matemáticamente valores hash similares Sin embargo las funciones hash criptográficas tienen la propiedad contraria en el sentido en que son $bit - sensitive$ es decir si un solo bit en el contenido cambia, el valor hash es totalmente diferente como se muestra en la figura 2.2. Por esta observación podemos concluir que para la autenticación de contenido multimedia no podemos utilizar este tipo de funciones hash por lo que se busca funciones basadas en contenido.

2.4. Hash basado en contenido

Reconocimiento de contenido usando características puede caracterizarse en dos grupos, métodos basados en características semánticas y no semánticas[7], las semánticas son por ejemplo extracción de límites o histograma de la imagen, que tienen un significado importante en la imágen, sin embargo aquellas que no tiene un significado semántico son robustas a diferentes transformaciones no basadas en contenido. Ambos grupos pueden ser usados para establecer perceptualmente similitudes en imágenes digitales, cabe notar que las características a extraer para Image hashing son un poco diferentes a las usadas en métodos de detección. Para los métodos de detección las

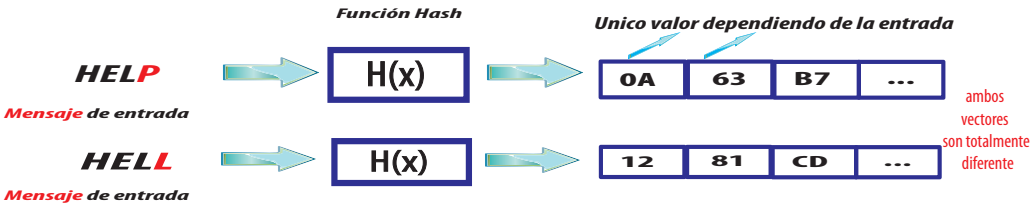


Figura 2.2: Función hash criptográfica, un sólo bit cambia, el valor hash es totalmente diferente

características deben facilitar las búsqueda de imágenes que de alguna manera parecen similares o de aquellas que poseen algún objeto similar, mientras que para Image hashing se requiere identificar imágenes perceptualmente iguales, excepto por diferencia de calidad o algún efecto de procesamiento de señales.

Construir una función hash basado en contenido no es una tarea trivial, por ejemplo los humanos no podemos diferenciar entre una imagen original o una versión comprimida o cambiando sus dimensiones en un monto pequeño. En general las funciones hash basadas en contenido tienen las siguientes características[12]:

- Robustez: Por robustez queremos decir que cuando la misma llave es usada, imágenes que perceptualmente son similares deben producir hashes similares.
- Libre de colisiones: Si dos imágenes perceptualmente diferentes sus valores hash deben ser totalmente diferentes o no deben tener ninguna correlación entre estos dos valores
- Seguridad: Se logra a través de una llave para generar el valor hash, sin el conocimiento de esta llave el valor hash debe ser imposible de estimar o falsificar.

2.5. Extracción de características

Existen diferentes formas de extraer características que pueden ser utilizadas para generar valores hash.

2.5.1. Detectores de esquinas

La detección de bordes es una herramienta fundamental en el procesamiento de imágenes y en la visión por computadora, particularmente en la extracción de características, lo que hace es poder identificar puntos en la imagen digital donde el brillo presenta discontinuidades como se muestra en la figura 2.3.

.
Para aplicar los detectores de bordes en el problema de Image hashing, se realiza a través del siguiente sistema como se muestra en la figura 2.4, el primer bloque extrae el vector de caracterśticas de la imagen, el paso dos comprime el vector final para obtener el valor hash final. En la extracción de características la imagen en 2-D se mapea a un vector en 1-D. Este vector debe capturar la esencia perceptual de la imagen.

Existen diferentes algoritmos para la detección de esquinas que se han propuesto[13], a continuación se mencionaran algunos de los más conocidos que son apropiados para el problema de Image hashing:

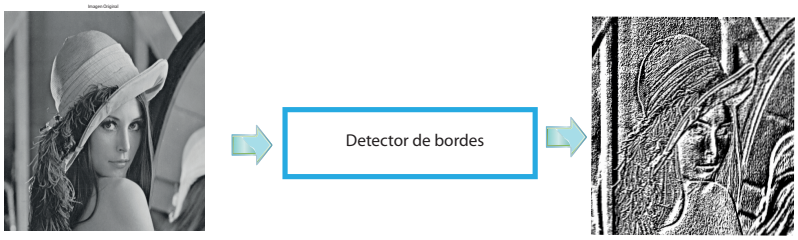


Figura 2.3: Entrada y salida de un detector de bordes

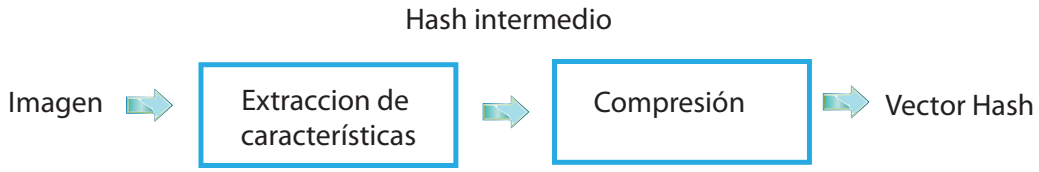


Figura 2.4: Diagrama de bloques de una función hash

2.5.1.1. Detector de Harris

Posiblemente el detector más conocido, el detector de Harris[14], usa diferentes características de la imagen. La construcción de este detector esta basado en el detector de bordes creado por Moravec. La función de respuesta $E_{x,y}$ es calculando el cambio (x, y) desde el punto central (u, v)

$$E_{x,y} = \sum_{u,v} w_{u,v} |I_{x+u,y+v} - I_{x,y}|^2 \tag{2.1}$$

Donde $I_{u,v}$ representa la luminancia de la imagen en la coordenada (u, v) y la función $w_{u,v}$ representa una ventana gaussiana con centro en (u, v). En esencia este detector funciona considerando una ventana en la imagen y determinando el cambio promedio de intensidad moviendo la ventana en pequeños cambios, el borde puede ser detectado cuando el cambio es grande. Harris reformuló la función de detección usando una representación matricial.

$$\underline{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \; I_x = \frac{\partial I}{\partial x}, \; I_y = \frac{\partial I}{\partial y} \tag{2.2}$$

$$A = (I_x)^2 * wB = (I_y)^2 * wC = I_x I_y * w. \tag{2.3}$$

La función de detección $E_{x,y}$ esta dada por $x^T M x$ y

$$M = \begin{bmatrix} A_{x,y} & C_{x,y} \\ C_{x,y} & B_{x,y} \end{bmatrix}. \tag{2.4}$$

$E_{x,y}$ puede interpretarse como la auto-correlación de la imagen con la forma de la función M. Harris y Stephens dieron una nueva definición usando eigenvalores α y β de la matriz M. Estos valores son invariantes a rotación y si sus magnitudes son grandes la auto-correlación es representado como un pico. Para evitar el computo de los eigenvalores de M, un nuevo criterio basado en el trazo y en los determinantes de M.

$$\begin{aligned} Tr(M) &= \alpha + \beta = A + B \\ M = det(M) &= \alpha \cdot \beta = A \cdot B - C^2 \\ R_H &= det(M) - k \cdot (Tr(M))^2 \end{aligned} \tag{2.5}$$

Donde k es una constante arbitraria. La extracción de características se logra aplicando un umbral a la respuesta de R_H y buscando el máximo local.

2.5.1.2. Detector de Hessian Affine

Una idea similar es la base para el detector Hessian Affine[15] la diferencia es en la matriz M que ahora esta dada por:

$$M = \begin{bmatrix} I_{x,x} & I_{x,y} \\ I_{x,y} & I_{y,y} \end{bmatrix} * w \tag{2.6}$$

Donde $I_{xx} = (\partial^2/\partial x^2)$ y $Iyy = (\partial^2/\partial y^2)$.
Las segundas derivadas que se utilizan en esta matriz da una fuerte respuesta a manchas y crestas. El punto interesante es que es similar a los detectores basados en operadores laplacianos pero con funciones en los determinantes de la matriz Hessian. Un máximo local en el determinante indica la presencia de un borde.

2.5.1.3. Detector de región máxima estable(MSER)

El detector de región máxima estable MSER [16] por sus siglas en inglés *Maximally Stable Extremal Regions*. La idea básica trata de que todos los pixeles en la región MSER tienen alta (regiones brillantes) o bajas(regiones oscuras) intensidades que los pixeles afuera de su frontera. Es decir todos los pixeles por debajo de un umbral dado son negros y todos los pixeles por encima del umbral son blancos. El máximo estable en la región MSER describe la propiedad de facilitar la selección de un umbral.

Las regiones MSER tienen dos importantes características:

- Continuidad: que es invariante a transformaciones Affine.
- Transformaciones mono tónicas de intensidad de una imagen: es sensible al cambio de iluminación provocado por la luz o por el movimiento de sombras

2.5.1.4. Detector End-stopped wavelet

Se aplica el operador de la primera derivada de filtro Gauseano FDoG a la wavelet de Morlet[17].

$$\psi_E(x, y, \theta) = (FDoG)o(\psi_M(x, y, \theta)). \tag{2.7}$$

La orientación esta dada por $\theta = \tan^{-1}(k_1/k_0)$. Hagamos que el rango de la orientaciń $[0, \pi]$ se dicretiza en M intervalos y el parámetro de la escala α se muestrea exponenciamente como $\alpha^i, i \in Z$. Este resultado en la familia de las wavelets es:

$$(\psi_E(\alpha^i(x, y, \theta_k))), \alpha \in \Re, i \in Z. \tag{2.8}$$

Donde $\theta_k = (k\pi)/M, k = 0, \ldots, M-1$ la transformada wavelet es :

$$W_i(x, y, \theta) = \int\int f(x_1, y_1)\psi_E(\alpha^i(x - x_1, y - y_1), \theta)dx_1dy_1 \tag{2.9}$$

El algoritmo para la extracción de características es:

1. Se computa la transformada wavelet como se muestra en la ecuación (2.9) con una escala adecuada de i para diferentes orientaciones. No se escoge i=1 porque es muy sensible a variaciones globales. Mientras mas fina sea la escala, es mas sensible a distorsiones.

2. Las posiciones (x,y) en la imagen que son identificadas como posibles punto de características satisface:

$$W_i(x, y, \theta) = \underset{(x, y') \in N_{(x,y)}}{max} |W_i(x', y', \theta)| \tag{2.10}$$

   Donde $N_{(x,y)}$ representa un vecino local de (x, y) donde se esta buscando.

3. De los candidatos seleccionados en el paso 2, se dice si es un punto si

$$\underset{\theta}{max} W_i(x, y, \theta > T) \tag{2.11}$$

   Donde T es el umbral determinado por el usuario.

En este método el paso uno computa la transformada wavelet para cada posición de la imagen. El paso dos identifica características significantes buscando el máximo local en los coeficientes wavelets en la ventana seleccionada. El paso tres elimina caracterśticas no significativas a través de un umbral.

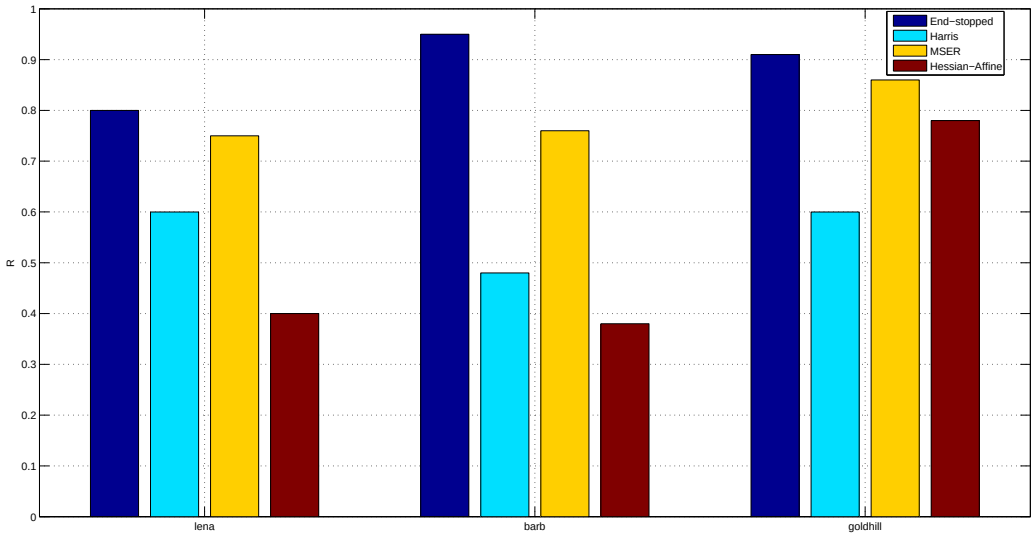


Figura 2.5: Benchmark de la función R de cuatro diferentes detectores de bordes aplicado a la imagen de Lena y diferentes tipos de ataques a dicha imagen, Mientras mas cercano a 1 mejor es el desempeño

2.5.1.5. Evaluación de los detectores

Para evaluar robustez de los detectores en las aplicaciones hash, se emplea una función de robustez R definida como:

$$R = \frac{N_{ret} - (N_{new} + N_{rem})}{N_{ret} + N_{rem}} \tag{2.12}$$

Donde $N_{tot} = N_{ret} + N_{rem}$ representa el numero total de puntos detectados en la imagen original. N_{ret} son los puntos detectados después de aplicar una transformada pero sin alteración de contenido, N_{rem} representa el numero de puntos que son removidos como resultado de la transformación y N_{new} es el numero de puntos que aparecen después de la transformada. Podemos observar que el valor máximo de esta función es 1 y esto sucede cuando $N_{new} = N_{rem} = 0$, que indica que en ambas imágenes se detectaron los mismos puntos.

En la figura 2.5 muestra el promedio de la función R después de aplicar diferentes tipos de ataques como compresión JPEG, rotación, escalamiento, ruido Gaussiano, filtros lineales y no lineales a la imagen de Lena.

Se puede observar claramente en la figura 2.5 que los valores más acertados de la función R son los obtenidos del detector End-stopped wavelet seguido del detector MSER. En la practica, se observa que la mayoría de los detectores tienen un comportamiento muy similar bajo ataques geométricos como rotación, translación y escalamiento. La verdadera ventaja del detector End-stopped wavelet es bajo ataques como compresión, adición de ruido y filtrados no lineales.

2.5.2. Algoritmo basado en la transformada Fourier-Mellin

La transformada de Fourier-Mellin ha mostrado ser invariante a diferentes tipos de transformada Affine[18], Swainathat propuso un algoritmo utilizando esta propiedad para atacar el problema de

Image Hashing[1].

2.5.2.1. Transformada Fourier-Mellin

Consideramos una imagen $i(x, y)$ y su transformada de Fourier 2-D como $I(f_x, f_y)$, donde f_x y f_y son frecuencias normalizadas en el rango $[0, 1]$. Y llamamos a la versión rotada, escalada y trasladada de $i(x, y)$ como $i'(x, y)$ y las podemos relacionarlas con:

$$i'(x, y) = i(\sigma(x\cos\alpha + y\sin\alpha), -x_0\sigma(-x\sin\alpha + y\cos\alpha) - y_0). \tag{2.13}$$

Donde los parámetros de rotación, escalamiento y translación son α, σ y (x_0, y_0) respectivamente y la magnitud de la transformada de Fourier de $i'(x, y)$ puede ser escrita como:

$$|I'(f_x, f_y)| = |\sigma|^{-2}|I(\sigma^{-1}(f_x\cos\alpha + f_y\sin\alpha), -\sigma^{-1}(-f_x\sin\alpha + f_y\cos\alpha) - x_0)|. \tag{2.14}$$

Ahora consideramos la representación polar en el dominio de Fourier, esto es $f_x = \rho\cos\theta$ y $f_y = \rho\sin\theta$ donde $\rho \in [0, 1]$ es el radio normalizado y $\theta \in [0, 2\pi]$ es el parámetro del ángulo. La ecuación (2.14) puede escribirse usando coordenadas polares como:

$$|I'(\rho, \theta)| = |\sigma|^{-2}|I(\rho\sigma^{-1}, \theta - \alpha)| \tag{2.15}$$

En la ecuación (2.15) podemos observar que la magnitud de la transformada de Fourier es independiente de los parámetros de traslación(x_0, y_0). Observando que la rotación de la imagen conlleva a una rotación en el mismo ángulo en el dominio de Fourier.

Integramos la magnitud de $I'(\rho, \theta)$ a lo largo de la circunferencia con centro en la frecuencia cero con un radio fijo ρ para obtener:

$$h(\rho) = \int_0^{2\pi} |I'(\rho, \theta)|d\theta \approx \int_0^{2\pi} |I(\rho, \theta - \alpha)|d\theta \approx \int_0^{2\pi} |I(\rho, \theta)|d\theta. \tag{2.16}$$

Esta propiedad de la transformada de Fourier nos es útil para usarla como una estructura robusta para extraer características.

Swaminathan propone el siguiente algoritmo[1] usando la transformada de Fourier-Mellin dividiendo le proceso en tres bloques como lo muestra la figura 2.6:

1. Preprocesamiento: primero se aplica un filtro pasa bajas. Posteriormente se aplica una normalización de histograma para obtener $i(x, y)$,posteriormente se aplica la transformada de Fourier 2-D para obtener $I(f_x, f_y)$, y esta es mapeada a $I'(\rho, \theta)$ como en la ecuación (2.15).
2. Extracción de características: se realiza una sumatoria alrededor de $I'(\rho, \theta)$ a lo largo del eje θ cada K puntos equidistantes en el rango $[0, 2\pi]$ (por ejemplo para $\theta \in \{\pi/K, 3\pi/K, \ldots, (2k - 1)\pi/K\}$) para obtener el vector de características h_ρ. Swaminathan propone dos esquemas para generar el valor hash intermedio[1].

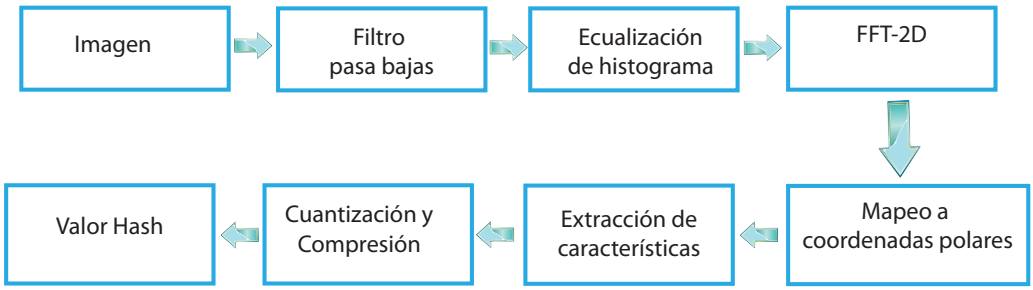


Figura 2.6: Diagrama de bloques del algoritmo propuesto por Swaminathan[1]

- Esquema 1: Se obtiene $|I'(\rho,\theta)|$ como se muestra en la ecuación (2.15) y se computa la sumatoria a lo largo del eje θ para obtener el jth valor hash,

$$h_j = \sum_{i=0}^{k-1} \beta_{\rho j,i} |I'(\rho_j, \frac{(2i+1)\pi}{K})| \tag{2.17}$$

  donde $\beta_{\rho j,i}$ son números pseudoaleatorio normalmente distribuidos con media m y varianza σ^2.
- Esquema 2: Con un llave se generan aleatoriamente el set de radios $\{\Gamma_j\}$. Se toma $|I'(\rho,\theta)|$ obtenida de la ecuación (2.15) y se realiza uan sumatoria en el eje θ para cada radio en el set. Una combinación lineal da como resultado el jth valor hash, que puede ser representado como:

$$h_j = \sum_{\rho\in\Gamma_j} \beta_\rho \sum_{i=0}^{k-1} |I'(\rho, \frac{(2i+1)\pi}{K})| \tag{2.18}$$

  donde β_ρ son números pseudoaleatorio normalmente distribuido con media m y varianza σ^2. Este método es mostrado en la figura 2.7:

3. Postprocesamiento: Se cuantiza el vector resultante y se le aplica código de gray para obtener el valor hash final. Este se pasa por un decodificador de orden 3 Reed-Muller para compresión.

Este algoritmo presenta robustez en contra de ataques como rotación hasta 10° y un 20 % de cropping, así como un desempeño adecuado sobre ataques de filtrado y compresión, debido a que los coeficientes de baja frecuencia de la transformada de Fourier que contribuyen en el valor hash, haciendo que dicho valor no varie mucho bajo este tipo de ataques.

2.5.3. Transformada de Radon

Las características más importante de una imagen digital son el contenido y la luminancia , existen diferentes métodos para extraer el contenido como son los detectores de bordes, sin embargo estas características no son invariantes a diferentes tipos de ataques. Por lo tanto usamos la luminancia y su relación en las diferentes áreas y esta no cambia en la mayoría de ataques que no afectan el contenido, y la transformada de Radon es un método muy efectivo que calcula la distribución de la luminancia en la imagen. La forma general de la transformada de Radon es:

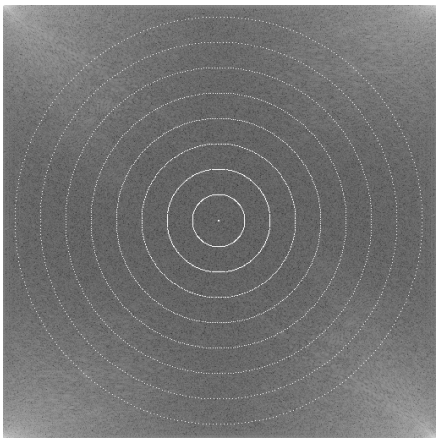


Figura 2.7: Transformada 2-D de la imagen de Lena. El valor jth valor hash de h_j, se logra sumando los pesos a través de la circunferencia escogida del set $\rho \in \Gamma_j$.

$$g(\rho, \theta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y)\delta(\rho - x\cos\theta - y\sin\theta)dxdy, \tag{2.19}$$

donde $\delta()$ es la función delta de Dirac. La distribución de Radon $g(\rho, \theta)$ es la proyección de cada θ en la imagen.

La propiedad más interesante de la transformada de Radon son la invariancia rotación y el escalamiento:

$$f(x\cos\theta_0 - y\sin\theta_0 x\cos\theta_0 + y\sin\theta_0) \leftrightarrow g(\rho, \theta - \theta 0) \tag{2.20}$$

$$f(sx, sy) \leftrightarrow \frac{1}{s}g(s\rho, \theta) \tag{2.21}$$

En la ecuación (2.20) θ_0 denota el ángulo de rotación. En la transformada de Radon esto aparece como un desplazamiento. Un escalamiento en la imagen por un factor, en la trasformada de Radon causa un escalamiento por el mismo factor. Desafortunadamente la transformada de Radon no es invariante a la mayoría de las transformaciones geométricas. Sin embargo el algoritmo presentado por Di Wu[19] puede compensar alguna de estas transformaciones.

Los coeficientes de la transformada de Radon cambian después de un ajuste de luminancia, mientras que la relación entre los coeficientes no cambia, por lo que podemos usar esta relación como característica a extraer.

El algoritmo para la extracción se muestra en la figura 2.8. Primero se aplica la transformada de Radon a la imagen en el campo discreto.

Antes de extraer las características, para compensar los efectos de la rotación y shift, hacemos algunos pasos como preprocesamiento:

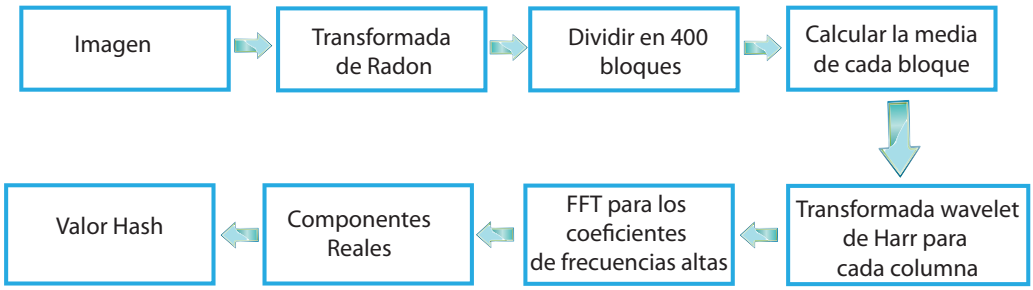


Figura 2.8: Algoritmo para la extracción de características utilizando la transformada de Radon

1. Se eliminan las filas cuyo valor sea cero.
2. Se localizan dos puntos A y B en la matriz de Radon.
3. Cíclicamente se mueve la matriz verticalmente hasta que A y B estén en la fila de en medio.

Se realiza la descomposición de wavelet de segundo nivel para extraer las características de los coeficientes de Radon. Se utiliza la wavelet de Haar por simplicidad. Antes de aplicar transformada wavelet, la distribución de Radon $g(\rho, \theta)$ se divide en bloques de 40×20 y la media de cada bloques es calculada. Después de la transformada los coeficientes de alta frecuencia son considerados como característica.

Finalmente los coeficientes de FFT en cada proyección radial son binarizados tomando la media de los coeficientes como umbral de cubanización.

Este algoritmo presenta algunas desventajas como[19]:

- Para ataques de rotación mayores a 45º hace que se pierda la sincronía, por lo que causa perdida de información.
- Como las formas en una imagen no pueden ser representadas explícitamente por la transformada de Radon, el algoritmo no es muy sensible a manipulaciones de forma sí la luminancia no cambia.

2.6. Conclusión

En este capťulo se presentarón las caracteriśticas que debe presentar una función hash basada en contenido como es la robustez y la seguridad ademas se introdujó diferentes técnicas en las que se ha abordado el problema de Image Hashing, usando diferentes formas para la extracci´on de características, donde se pudo observar que en la mayoría de los casos se presenta una buena conservación del contenido de la imagen sin embargo en la presencia de ataques geomÃ©tricos se presenta poca robustez.

Capítulo 3

Descomposición SVD y Normalización de imágenes digitales

En este capítulo revisaremos la teoría detrás del algoritmo propuesto para resolver el problema de Image hashing. En este algoritmo se basa en dos importantes tópicos. Uno es descomposición SVD(Singular Value Decomposition), que es una importante herramienta de algebra lineal, que es utilizada usualmente en compresión de imágenes, marcas de agua y en otros campos de procesamiento de imágenes. Por otra parte el algoritmo de normalización de imágenes, es un algoritmo basado en los momentos invariantes de una imagen. Estos momentos son usados para normalizar la imagen geométricamente en términos de orientación y escalamiento . Este algoritmo es muy bien conocido en el reconocimiento de patrones.

3.1. Descomposición SVD

La descomposición SVD de sus siglas en inglés Singular Value Deomposition, es uno de los conceptos básicos de algebra lineal. La descomposición SVD fue introducida para matrices cuadradas reales por Beltrami y Jordan en 1870 y para matrices imaginarias por Autonne en 1902 y se generalizó para matrices reales o complejas, cuadradas o no cuadadas por Eckhart y Young en 1939.

La descomposición SVD utiliza los vectores propios y valores propios.

3.1.1. Vectores propios y valores propios

Los vectores propios o eigenvectores, son los vectores no nulos, que al ser transformados por un operador lineal dan como resultado un múltiplo escalar de si mismo, con lo que no cambia su dirección. Este escalar λ recibe el nombre de valor propio o eigenvalue. El prefijo eigen viene de la palabra alemana eigen que significa propio.

Los vectores propios y valores propios tienen las siguientes características:

- Los vectores propios de las transformaciones lineales son vectores que no se ven afectados por la transformación o se ven multiplicados por un escalar y por lo tanto no varían su dirección.
- El valor propio de un vector propio es el factor de escala por el que se ha multiplicado.
- Un espacio propio es un espacio formado por todos los vectores propios del mismo valor propio, además del vector nulo que no es un vector propio.
- La multiplicidad geométrica de un valor propio es la dimensión del espacio propio asociado.

- El espectro de una transformación es espacios vectoriales finitos es el conjunto de todos sus valores propios.

Formalmente, definimos los valores y vectores propios como:

$$Av = \lambda v \tag{3.1}$$

El escalar λ se dice que es un valor propio de A correspondiente a v.

3.1.1.1. Cálculo de valores propio

Para poder encontrar los valores propios de matrices cuadradas utilizamos la técnica del polinomio característico. Es decir que λ es un valor propio de A que es equivalente al sistema de ecuaciones lineales $Av = \lambda v \rightarrow Av - \lambda v$ donde factorizando podemos escribir:

$$det(A - \lambda I) = 0. \tag{3.2}$$

Los valores propios de una matriz son los ceros de su polinomio característico.

3.1.1.2. Cálculo de vectores propios

Una vez que se conocen los valores propios λ, los vectores propios se pueden hallar resolviendo el sistema de ecuaciones homogéneo:

$$(A - \lambda I)v = 0. \tag{3.3}$$

3.1.1.3. Ejemplo del cálculo de valores y vectores propios

Para la matriz:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}. \tag{3.4}$$

La ecuación característica es:

$$det(A - \lambda I) = det \begin{bmatrix} 2 - \lambda & 1 \\ 1 & 2 - \lambda \end{bmatrix}. \tag{3.5}$$

El determinante nos da como resultado la ecuación cuadrática:

$$\lambda^2 - 4\lambda + 3 = 0 \tag{3.6}$$

Donde las soluciones son $\lambda = 1$ y $\lambda = 3$. Los vectores propios para el valor propio $\lambda = 3$ los calculamos usando la ecuación (3.1), que nos da:

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 3 \begin{bmatrix} x \\ y \end{bmatrix}.$$

Donde obtenemos:

$$\begin{aligned} 2x + y &= 3x \\ x + 2y &= 3y \end{aligned} \tag{3.7}$$

Nos queda que $x = y$. Esto quiere decir que cualquier vector de la forma (x, y) con $x = y$ es un vector propio del valor propio $\lambda = 3$. De la misma manera hacemos el cálculo para $\lambda = 1$ que nos da que son los vectores (x, y) tal que $y = -x$. Estos vectores con colocados como columnas de una matriz.

3.2. Descomposición SVD aplicado a imágenes digitales

Una de las características de la descomposición SVD es que puede realizarse sobre cualquier matriz (m, n) . Una matriz A con m columnas y n filas, con rango r donde $r < m$, A puede factorizarse en tres submatrices.

$$A = USV^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \tag{3.8}$$

donde la matriz U es una matriz de $m \times m$ ortogonal

$$U = u_1, u_2, \ldots, u_{r+1}, \ldots, u_m, \tag{3.9}$$

el vector columna u, para $i = 1, 2, \ldots, m$ forma un conjunto ortogonal

$$u_i^T u_j = \delta_{ij} = \begin{cases} 1, \ldots, i = j \\ 0, \ldots, i \neq j \end{cases}, \tag{3.10}$$

la matriz v de $n \times n$ forma un conjunto ortogonal.

$$V = v_1, v_2, \ldots, v_{r+1}, \ldots, v_n. \tag{3.11}$$

El vector columna v_i para $i = 1, 2, \ldots, n$ forma un set ortogonal:

$$v_i^T v_j = \delta_{ij} = \begin{cases} 1, \ldots, i = j \\ 0, \ldots, i \neq j \end{cases}. \tag{3.12}$$

Donde S es una matriz diagonal con los valores singulares en la diagonal como se muestra:

$$S = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & s & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & \sigma_{r+1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & \sigma_n \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}$$

Para $i = 1, 2, \dots, n$ σ_i son llamados valores singulares de la matriz A, v_i y u_i son llamados vectores singulares derechos e izquierdos de A respectivamente[20].

3.3. Propiedades de SVD

Existen muchas propiedades y atributos que tiene la descomposición SVD[21]:

- Los valores singulares $\sigma_1, \sigma_2, \dots, \sigma_n$ son únicos, sin embargo las matrices U y V no son únicas.
- Los valores singulares cambian poco cuando se aplica pequeñas perturbaciones a la matriz.
- Los valores singulares representan las características de iluminación en la imagen, mientras que los vectores singulares representan las características geométricas.
- La calidad de la imagen regenerada no se ve afectada si removemos los valores propios menores. Como se muestra en la figura 3.1 donde se utilizan diferentes números de valores propios y vectores propios para la reconstrucción.
- El rango de la matriz a descomponer es igual al número de valores propios con valor distinto de cero[22].

height=4cm, width=4cm]lenar10.eps height=4cm, width=4cm]lenar50.eps

(a) r=10 (b) r=50

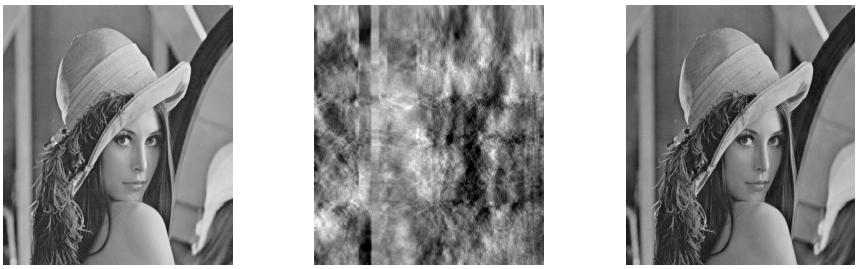
height=4cm, width=4cm]lena.eps height=4cm, width=4cm]lena.eps

(c) r=180 (d) Imagen original

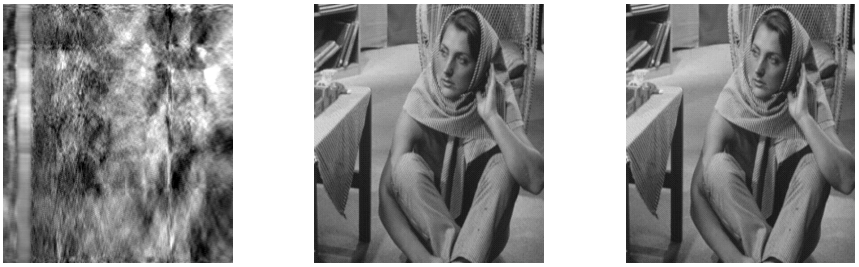
Figura 3.1: Diferente numero de valores propios y vectores propios para la reconstrucción de una imagen usando descomposición SVD.

Para el problema de Image Hashing utilizaremos los vectores singulares, debido a que estos guardan las características geométricas, además por resultados experimentales[23] estos guardan las características de contenido como se muestra en la figura **??**, donde podemos observar las diferentes combinaciones de las matrices U, S, V, Donde U_1, V_1, S_1 son los valores de la descomposición SVD de la imagen de Lena y U_2, V_2, S_2 son los valores de la descomposición SVD de la imagen de Barb, si sólo cambiamos la matriz S, la imagen no se ve afectada, sin embargo si cambiamos alguna de las matrices U ó V, la imagen no se puede regenrar.

Con esto podemos establecer que las matrices U y V son las que nos interesan ya que preservan las características del contenido de la imagen digital, por lo que podemos utilizarla para generar las características en el problema de Image Hashing.



(a) U1*S1*V1 (b) U1*S1*V2 (c) U1*S2*V1



(d) U2*S1*V1 (e) U2*S1*V2 (f) U2*S2*V2

Figura 3.2: Resultado de cambiar valores propios y vectores propios

3.4. Normalización de Imágenes

3.4.1. Transformada Affine

Las transformaciones geométricas modifican la relación espacial entre pixeles. Es decir que es una transformación espacial que define la reubicación de los pixeles en el plano y los interpolan.

Una transformación Affine es aquella que las coordenadas (x', y') son expresadas en términos de las coordenadas originales (x, y), lo podemos escribir como:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.13}$$

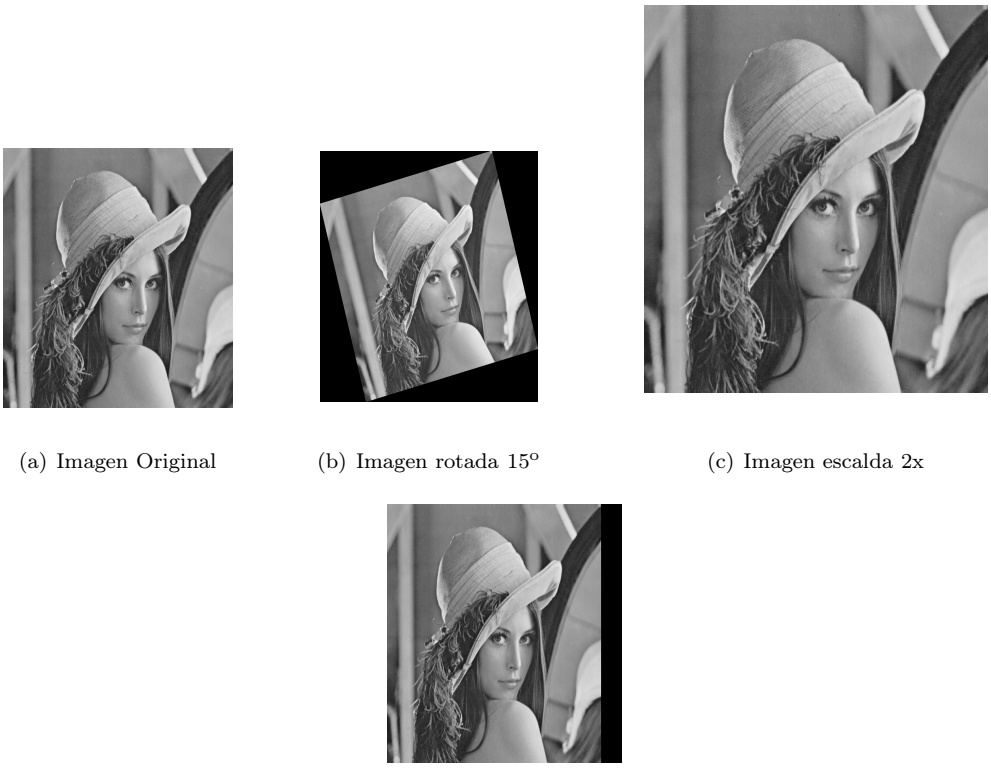
El sistema tiene solución si y solo si $\Delta = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, es decir, si la matriz es no singular. Con la transformada Affine podemos rotar, escalar y trasladar una imagen digital. Para poder rotar una imagen en un ángulo θ aplicamos

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.14}$$

Para poder escalar una imagen tanto en el eje horizontal como en el vertical utilizamos

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.15}$$

Como se muestra en la figura 3.3, vemos el resultado de aplicar diferentes transformadas Affine.



(a) Imagen Original (b) Imagen rotada 15º (c) Imagen escalda 2x

(d) Imagen trasladada

Figura 3.3: Diferentes transformadas Affine aplicada a la imagen de Lena.

Dentro de las propiedades de la transformada Afinne tenemos:

- Existe una transformada inversa.
- Puede representar una composición de transformadas básicas.
- La transformada Affine puede separarse en traslación, rotación y shearing.

3.4.2. Principios de la normalización de Imágenes

Hagamos que $f(x, y)$ denote una imagen digital de tamaño M y sus momentos geométricos $m_{p,q}$ y sus momentos centrales $\mu_{p,q}$, $p, q = 0, 1, 2, \ldots$, son definidos como:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y) \tag{3.16}$$

y

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \tag{3.17}$$

donde

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}}, \quad \bar{y} = \frac{m_{0,1}}{m_{0,0}}. \tag{3.18}$$

Una imagen $g(x, y)$ se dice que es una transformación Affine de la imagen $f(x, y)$ si existe una matriz $A = \begin{pmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{pmatrix}$ y un vector $d = (d_1, d_2)^T$ talque $g(x, y) = f(x_a, y_a)$ donde

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} - d. \tag{3.19}$$

Recordemos que la rotación, translación y escalamiento son casos especiales de la transformada Affine.

3.4.3. Momentos geométricos y centrales

Las características más útilies que podemos extraer de estos, es que son invariantes a transformaciones gerométricas. Para calcular los momentos centrales se usa el centroide de la figura.

El centroide o centro de masa de una figura determinado por las coordenadas $(\bar{x}, \bar{y})$, de forma que el área que queda a la derecha e izquierda del punto $\bar{x}$ es la misma, igual que el área que queda por encima y por debajo del punto $\bar{y}$.

3.4.3.1. Momentos de orden 0

Momento simple de orden 0, $m_{0,0}$. Suma todos los pixeles cuyo valor es distinto de 0.

$$m_{0,0} = \sum_x \sum_y f(x, y) \tag{3.20}$$

3.4.3.2. Momentos de orden 1

Los momentos de orden 1, $m_{1,0}$ y $m_{0,1}$ como se ha mencionado se usan para determinar el centro de masa de una figura.

$$m_{1,0} = \sum_x \sum_y x f(x, y), \quad m_{0,1} = \sum_x \sum_y y f(x, y). \tag{3.21}$$

Por definición los momentos centrales $\mu_{1,0}$ y $\mu_{0,1}$ valen 0

$$\begin{aligned} \mu_{1,0} &= \sum_x \sum_y (x - \bar{x})^1 (y - \bar{y})^0 f(x, y) \\ &= m_{1,0} - \frac{m_{1,0}}{m_{0,0}} m_{0,0} \\ \mu_{0,1} &= \sum_x \sum_y (x - \bar{x})^0 (y - \bar{y})^1 f(x, y) \\ &= m_{0,1} - \frac{m_{0,1}}{m_{0,0}} m_{0,0} \end{aligned} \tag{3.22}$$

3.4.3.3. Momentos de orden 2

Son de vital importancia en el cálculo de momentos centrales.

$\mu_{2,0}$: su valor aumenta cuanto mayor sea la componente horizontal en la imagen.

$\mu_{0,2}$: su valor aumenta cuanto mayor sea la componente vertical en la imagen.

$\mu_{1,1}$: usa las componentes horizontal y vertical, puede ser positivo o negativo dependiendo de donde se encuentre la componente vertical; si la componente vertical se encuentra en los cuadrantes 2 o 4, entonces será negativo, si se encuentra en los cuadrantes 1 y 3 entonces es positivo, sí la imagen es simétrica con respecto a los ejes $\mu_{1,1}$ será 0.

3.4.4. Algoritmo de normalización

Como se muestra en la figura 3.4 tenemos las entradas figura 3.4(a), (b) ó (c) y la salida figura 3.4(d) del algoritmo de normalización de imágenes cuando (a), (b) ó (c) es la entrada. Dong propusó un algoritmo para el procedimiento de normalización que consiste en los siguientes pasos[11]:

1. Centramos la imagen $f(x, y)$, esto lo logramos haciendo la matriz $A = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ y el vector $d = (d_1\ d_2)^T$ en la ecuación (3.19) con:

   $$d_1 = \frac{m_{1,0}}{m_{0,0}}, \quad d_2 = \frac{m_{0,1}}{m_{0,0}}. \tag{3.23}$$

   Donde m_{10}, m_{01} y m_{00} son los momentos de $f(x, y)$ definidos en la ecuación (3.16), con este paso hacemos que sea invariante a traslación. El resultado de este paso lo denotaremos como $f_1(x, y)$.

2. Aplicamos shearing en la dirección x a $f_1(x, y)$ con la matriz $A = \left(\begin{smallmatrix} 1 & \beta \\ 0 & 1 \end{smallmatrix}\right)$ y el vector $d = 0$ y denotamos la salida como $f_2(x, y)$. Para determinar beta usamos:

   $$\mu_{30} + 3\beta\mu_{21} + 3\beta^2\mu_{12} + \beta^3\mu_{03} = 0. \tag{3.24}$$

   De esta ecuación podemos tener hasta tres raíces, si una de ellas es real y las otras complejas, utilizamos la raíz real. Si dos o las tres raíces son reales utilizamos la media de las raíces reales. Si todas las raíces son complejas $\beta = 0$.

   En casos muy raros β puede tener un numero infinito de soluciones, si todos los momentos son cero, esto pasa cuando la imagen es rotacionalmente simétrica[24][25].

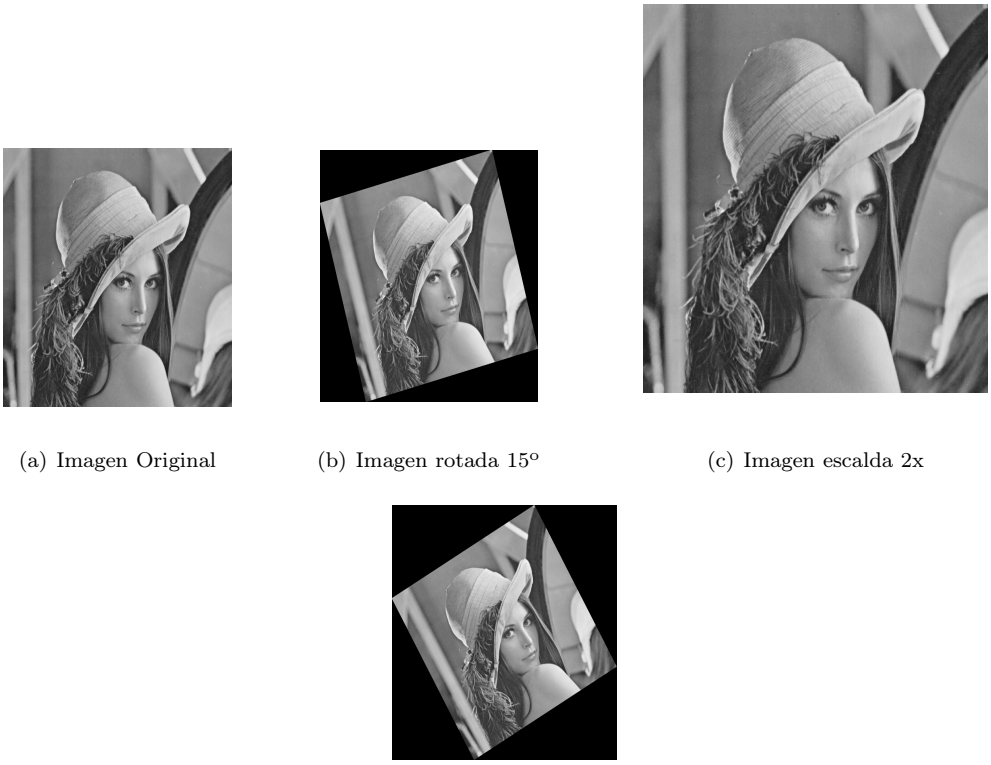
3. Aplicamos shearing en la dirección y a $f_2(x, y)$ con la matriz $A = \left(\begin{smallmatrix} 1 & 0 \\ \lambda & 1 \end{smallmatrix}\right)$ y el vector $d = 0$ y denotamos la salida como $f_3(x, y)$. λ es definida con la ecuación

   $$\mu_{11} = \lambda\mu_{20} + \mu_{11}, \tag{3.25}$$

   y hacemos $\mu_{11} = 0$ obtenemos

   $$\lambda = \frac{\mu_{11}}{\mu_{20}}. \tag{3.26}$$

4. Finalmente se le aplica escalamiento a $f_3(x, y)$ en las dimensiones deseadas para tener la imagen normalizada $g(x, y)$



(a) Imagen Original

(b) Imagen rotada 15º

(c) Imagen escalda 2x

(d) Salida del algoritmo de normalización

Figura 3.4: entradas y salidas del algoritmo de normalización de imágenes.

3.5. Conclusión

En este capítulo se presento la descomposición SVD como un método para la extracción de características, debido a la posibilidad de poder extraer características de iluminación y de contenido. Por lo que se ha escogido como el método a utilizar para la generación del vector Hash. De la misma forma se presento la utilización del algoritmo de normalización de imágenes como un algoritmo que nos permita incrementar la robustez ante ataques geométricos comprendidos en la transformada Affine.

Capítulo 4

Algoritmo de Autenticación de Imágenes Digitales

En este capítulo veremos las características del algoritmo propuesto para la resolución del problema de Image Hashing.

El algoritmo propuesto se vasa en la descomposición SVD para la extracción de características asi como de el algoritmo de normalización de Imágenes para hacerlo robusto a diferentes tipos de ataques geométricos, como rotación, escalamiento, shearing.

El algoritmo propuesto por Kozat et. All [2] propone un algoritmo basado en la descomposición SVD donde hace la descomposición SVD dos veces. Debido a que si se hace mas de dos veces esta descomposición, el algoritmo ya no es lo suficientemente robusto y seguro [4].

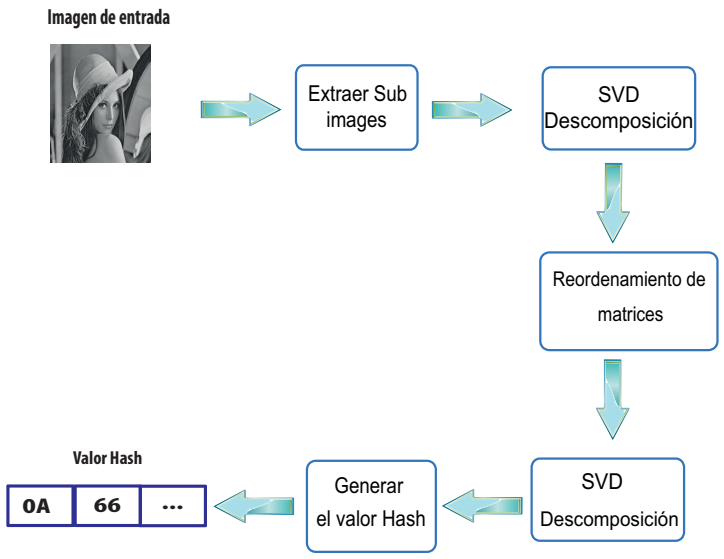


Figura 4.1: Diagrama de Bloques del algoritmo propuesto por Kozat[2]

Como se muestra en la figura 4.1 el valor Hash final es una vector de numero reales. Debido a que la mayoría de los sistemas, para la resolución del problema de Image hashing, la salida de estos se compone un vector binario. Podemos decir que es una desventaja de dicho algoritmo.

También por resultados experimentales este algoritmo aunque presenta una buena preservación de contenido, no presenta robustez a diferente tipo de ataques geométricos, debido a que al cambiar las dimensiones de la imagen, el valor Hash generado es muy diferente al valor Hash original.

4.1. Esquema Propuesto

4.1.1. Algoritmo de Partición Aleatoria

Para incrementar la seguridad en el valor Hash y agregar algo de aleatoriedad a dicho valor, usamos el algoritmo de partición aleatoria donde seleccionamos un numero p de sub-imágenes como se muestra en la figura 4.2.

Los pasos de dicho algoritmo son los siguientes:

1. Con una llave k generamos un set W, donde $W = ((x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p),)$, que son el par coordenado de la esquina superior izquierda de la sub-imagen $A_1, A_2 \ldots, A_p$.
2. Por cada coordenada generar un cuadrado de tamaño m, y cada cuadrado es la salida del algoritmo que llamamos sub-imagen A.

Con este algoritmo no solo agregamos seguridad al seleccionar aleatoriamente un número p de sub-imágenes, si no que agregamos redundancia en la extracción de características debido a que las sub-imágenes se traslapan por lo que las características extraídas de las sub-imágenes presentan redundancia.

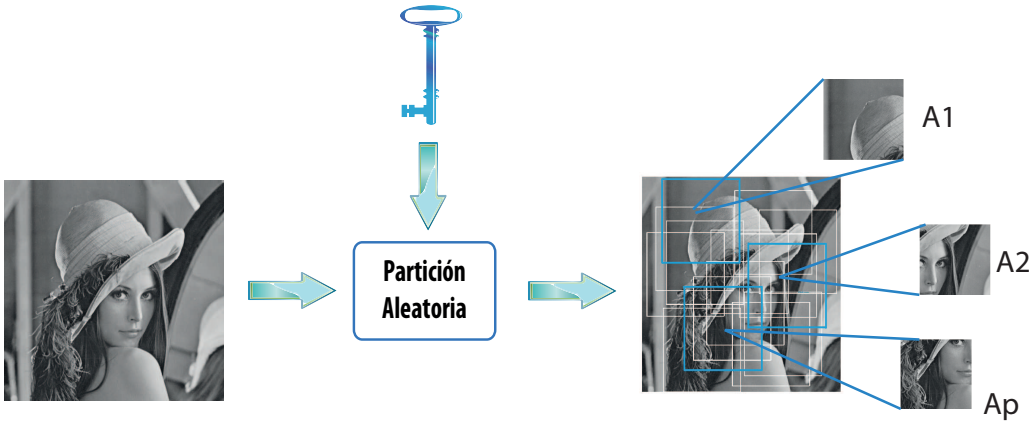


Figura 4.2: Entrada y salida de el algoritmo de partición aleatoria

4.1.2. Códigos de REED-MULLER

Los códigos de Reed-Muller conforman una familia de códigos lineales que permiten su decodificación por lógica mayoritaria. El código ReedâMuller de orden r y longitud 2^m se denota $RM(r, m)$ donde $0 \leq r \leq m$. Se define recursivamente como:

1. $RM(0,m) = 00..,0,1..,1, RM(m,m) = k^{2^m}; k^n =$ Conjunto de dígitos binarios de longitud n.
2. $RM(r,m) = (x, x+y)|x \ni RM(r,m-1), y \ni RM(r-1,m-1), 0 \leq r \leq m.$

De esta forma RM(m, m) son todas las palabras de longitud 2^m y RM(0, m) contiene a palabras de todos ceros o todos unos.

La matriz generadora se construye exclusivamente como:

$$G(r,m) = \begin{bmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{bmatrix} \tag{4.1}$$

donde $0 < r < m$.

Para $r = 0$ se define $G(0,m) = [11..,1]$.

Para $r = m$ se define $G(m,m) = \begin{bmatrix} G(m-1,m) \\ 0 \ldots 01 \end{bmatrix}$

Así las matrices generadoras de $RM(0,1)$ y $RM(1,1)$ son:

$G(0,1) = (1,1)$ y $G(1,1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$.

Si $m = 2$, la longitud es $4 = 2^2$ y para $r = 1, 2$ se tiene:

G(1,2)=$\begin{bmatrix} G(1,1) & G(1,1) \\ 0 & G(0,1) \end{bmatrix}$ G(1,2)=$\begin{bmatrix} G(1,2) \\ 0001 \end{bmatrix}$

G(1,2)=$\begin{bmatrix} 11 & 11 \\ 01 & 01 \\ 00 & 11 \end{bmatrix}$ G(2,2)=$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Para $m = 3$:

G(0,3)=(11 11 11 11) G(3,3)=$\begin{bmatrix} & & G(2,3) & \\ 00 & 00 & 00 & 01 \end{bmatrix}$

G(1,3)=$\begin{bmatrix} G(1,2) & G(1,2) \\ 0 & G(0,2) \end{bmatrix}$ G(2,3)=$\begin{bmatrix} G(2,2) & G(2,2) \\ 0 & G(1,2) \end{bmatrix}$

Y de esta forma:

$$G(1,3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

La distancia de Hamming es de $d = 2m - r$.

4.1.2.1. algoritmo de decodificación

Antes de decodificar, se define el producto de Kronecker como $A = [a_{ij}B]$, esto es cada componente de la matriz resultante es la matriz $[aijB]$.

Por ejemplo, si $H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $M = \begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix}$, $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, entonces:

$$\text{M} \times H = \begin{bmatrix} 2 & 2 & -1 & -1 \\ 2 & -2 & -1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad \text{I}_2 x H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$\text{H} \times I_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Ahora sean las matrices:

$H_m^i = I_{2m-i} \times H \times I_{2i-1}$

Para $i = 1, 2, \ldots m$, donde H es de la forma del ejemplo anterior. Así para $m = 2$:

$$\begin{aligned} & i = 1 \\ H_2^1 = I_2 \times Htimes I_1 = I_2 \times H \\ H_2^2 = I_1 \times H \times I_2 = H \times I_2 \end{aligned} \tag{4.2}$$

Supongamos que recibe w y $G(1, m)$ en la matriz generadora de $R(1, m)$:

1. Sustituir a 0 por 1 en w para formar w.
2. Calcular $w_1 = w \times H_m^1$ y $w_i = w_{i-1} H_m^i$ para $i = 2, 3, \ldots, m$.
3. Encontrar la posición j de la mayor componente (en valor absoluto) de w_m.

Sea $v(j) \in km$ la representación binaria d_j (en PCM) inicia en posición 0.

Entonces si j^{TH} componente de w_m es positivo, el posible mensaje es $(1, v(j))$ y si la componente es negativa el posible mensaje es $(0, v(j))$.

4.1.2.2. Código de Gray

Este es un código binario no ponderado y tiene la propiedad de que los códigos para dígitos decimales sucesivos difiere en un sólo bit. El código Gray también es conocido como autorreflejado, o cíclico.

Para n=1 se tiene solo dos dígitos por lo que es trivial.

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Para n = 2 notamos que la mitad superior de la secuencia difiere de la inferior en el bit más significativo. Además el orden del bit menos significativo de la mitad inferior de la secuencia se invierte con respecto a la primera mitad.

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Si abajo del código Gray para n = 2 añadimos una copia del mismo invertida en el eje vertical se cumple la propiedad de cambio en un solo bit, salvo en las dos filas centrales y entre la primera y la última fila. Entonces añadimos un 0 al comienzo de cada fila de la primera mitad y un 1 a cada fila de la segunda mitad. El resultado es el código binario Gray para n = 3.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Podemos entonces definir el código binario Gray de manera recursiva. Si decimos que Γ_n representa la secuencia binaria Gray de n bits, entonces podemos definir a Γ_n con [26]:

$$\Gamma_n = \in \qquad \Gamma_{n+1} = 0\,\Gamma_n, 1\,\Gamma_n^R \tag{4.3}$$

Donde $\in$ representa una cadena vacía, $0\Gamma_n$ denota la secuencia Γ_n con prefijo 0 añadido a cada cadena, y $1\Gamma_n^R$ simboliza el reverso de la secuencia Γ_n con prefijo 1 agregado a cada cadena.

4.2. Algoritmo Propuesto

El algoritmo propuesto de Image hashing tiene dos entradas una imagen I y una llave secreta k y tiene una salida que es un vector binario $\vec{h} = H_k(I)$.Como se muestra en la figura 4.3.

La función Hash debe tener las siguientes características perceptuales:

1. El valor Hash para todas las imágenes perceptualmente iguales, esto significa cuando una persona no puede ver ninguna diferencia en el contenido, debe tener el mismo valor hash.

2. Imágenes perceptualmente diferentes, cuando dos imágenes no contienen el mismo contenido en términos de percepción visual, deben producir un valor Hash diferente.

Además este valor sin el uso de una llave debe ser casi imposible de estimar, por lo que el valor Hash generado tiene que ser seguro. Y conservar la información de contenido de la imagen digital.

Los pasos del algoritmo propuesto son los siguientes:

1. Hagamos que las entradas estén compuestas por la imagen I de tamaño M, $I \in [0, 255]^M$, el numero de sub-imágenes denotado por p de tama no m, donde $1 < m < M, N$ y la llave secreta para seleccionar dichas sub-imágenes denotada por k. Y la salida el vector $\vec{h}$ es el valor hash final.

2. A la imagen I le aplicamos el algoritmo de normalización de imágenes descrito en el capitulo anterior, para generar la imagen normalizada I_1.

3. A la imagen I_1, con el uso de una llave k, pseudo-aleatoriamente seleccionamos p posibles sub-imágenes $A_1, A_2, \ldots, A_p$ de tamaño m con el algoritmo de partición aleatoria.

4. A cada sub-imagen A_i, se le aplica la descomposcion SVD para obtener $\{U_1S_1V_1, U_2S_2V_2, \ldots, U_pS_pV_p\}$ donde U_i y V_i son los vectores singulares que guardan la información de contenido de cada sub-imagen y S_i son los valores singulares.

5. La primera columna de cada U_i y la primera fila de V_i^T, son los componentes de una nueva matriz B,

$$B = [U_1^{(1)}U_2^{(1)} \ldots U_p^{(1)}, V_1^{(1)T}V_2^{(1)T} \ldots V_p^{(1)T}] \tag{4.4}$$

   De tamaño $m \times 2p$.

6. Le aplicamos la descomposición SVD a la nueva matriz B, para obtener U',V' y S'.

7. La primera columna de U' y la primera fila de V'^T son el valor hash intermedio con valores reales de longitud $m + 2p$.

8. El valor hash intermedio es cuantificado y se le aplica código de gray para obtener un una secuencia binaria.

9. La secuencia de bits es pasada por el decodificador de tercer orden de Reed Muller para compresión.

10. Se obtiene la salida $\vec{h}$ que es el valor hash final.

4.3. Conclusión

Se presentó el algoritmo para el cálculo del vector Hash donde se introdujeron diferentes algoritmos como el de partición aleatoria para incrementar la seguridad y agregar redundancia al vector Hash final, de la misma manera se presento la forma en que se obtiene el vector Hash binario usando los codigos de Reed-Muller, al binarizar el vector Hash se vuelve una salida mas común. En el algoritmo propuesto se puede observar que al agregar una parte de pre-procesamiento como es el algoritmo de normalización de imágenes no afecta las características del vector Hash.

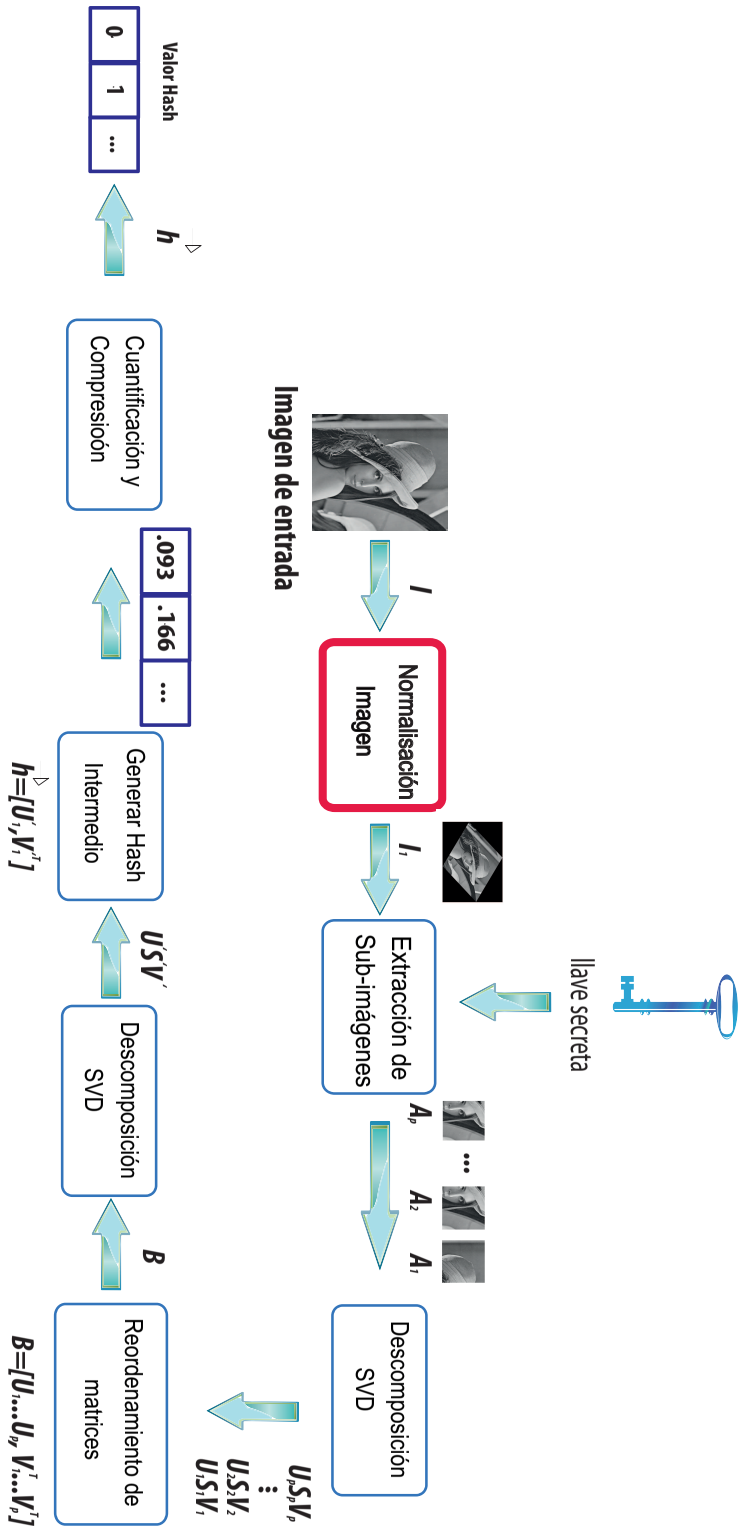


Figura 4.3: Diagrama de Bloques del algoritmo propuesto

Capítulo 5

Resultados

5.1. Características de las pruebas

Las pruebas se realizaron con imágenes en escala de grises de 512×512 pixeles muestreadas a 8 bits. Se realizaron sobre una base de datos de 1000 imágenes, que contiene imágenes conocidas como *Lena*, *Goldenhills*, *bird*, asi como imágenes de paisajes como se muestra en la figura 5.1, y sus versiones atacadas en rotación, escalamiento, recortadas, ruido Gauseano, estos ataques se realizaron con la herramienta Stirmark 4.0 [3].

(a) bird (b) Paisaje (c) Lena

(d) Barb

Figura 5.1: Ejemplo de imágenes de referencia que utilizamos como imágenes originales, es decir sin modificación.

Las imágenes que se utilizaron para la realización de las pruebas son como se muestra en la tabla 5.1.

Ataque	Parametros	Número de imágenes
Rotación	Grados 0,25 – 45	299
Escalamiento	25 % - 200 %	65
Cropping	1 % - 20 %	65
Compresión JPG	Factor de compresion en porcentaje 1 %-99 %	195
Paisajes		300
Total		924

Tabla 5.1: Características de las imágenes de prueba.

5.2. Política de evaluación

Para poder lograr la autenticación de imágenes a través de Image Hashing, generamos el vector Hash de la imagen sin modificación u original, y se calcula el vector Hash de la imagen atacada o modificada, posteriormente se calcula la distancia de Hamming entre estos dos vectores para poder ver que tan similares son como se muestra en la figura 5.2.

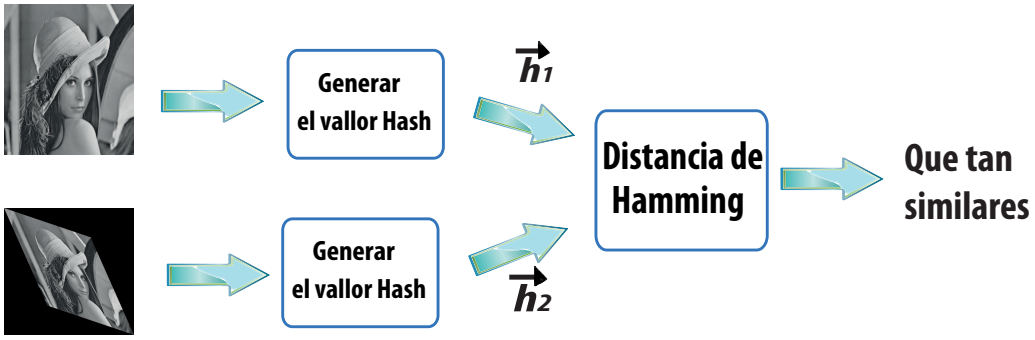


Figura 5.2: Politica de evaluacion para la autenticacion de imágenes.

5.2.1. Distancia de Hamming

Sean h_1 y h_2 dos vectores de longitud L, la distancia Hamming (o simplemente distancia) entre h_1 y h_2, que se denota como $d_h(h_1, h_2)$, se define como el número de dígitos en el mismo sitio que tienen diferentes. Por ejemplo, la distancia Hamming entre h_1 = (1 0 0 1 0 1 1) y h_2 = (0 1 0 0 0 1 1) es 3; tienen diferentes las posiciones cero, uno y tres. Podemos definir la distancia de Hamming normalizada como:

$$d_h(h_1, h_2) = \frac{1}{L} \sum_{k=1}^{L} \|h_1(k) - h_2(k)\| \tag{5.1}$$

Cuando el valor de la distancia es cercano a 0 podemos decir que los vectores son similares, pero si el valor de la distancia es cercano a 1 podemos decir que los vectores son totalmente diferentes. Con este principio podemos establecer que si la distancia de haming es cercana a 0 las dos imágenes tienen contenido similar, pero si la distancia de Hamming es cercana a 1 son totalmente diferentes.

5.3. Resultados

Para el ataque de rotación como se muestra en la figura 5.3 donde se muestra el ángulo de rotación contra la distancia de Hamming, y cada punto es un promedio de diferentes imágenes atacadas, podemos observar que en el caso del algoritmo propuesto, se reduce la distancia de Hamming en un 50 % tomando como 100 % el valor máximo que alcanza la distancia de Hamming en el algoritmo propuesto por Kozat. Es decir pasa de 0,14 a ,07. Con esto tenemos una mejor autenticación en contra de la rotación.

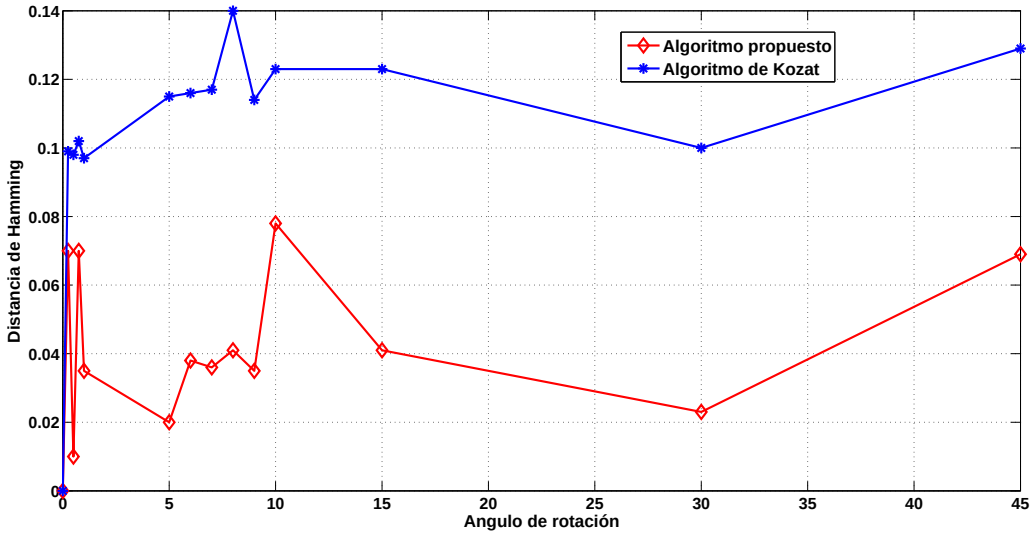


Figura 5.3: Ataque de rotación, ángulo de rotación vs distancia de Haming, donde 0 es la imagen sin rotación.

Para el ataque de escalamiento, como se puede observar en la figura 5.4 donde se muestra el porcentaje de escalamiento contra la distancia de Hamming, siendo 100 % la imagen con dimensiones originales, y cada punto es el promedio de diferentes imágenes atacadas.

Aunque no se haya una mejora muy significativa, se reduce la distancia de Haaming para escalamientos entre el 60 % y 200 %, esto es debido a que si la imagen es muy grande,en estae caso mayor a 1024×1024, las sub-imágenes extraídas no abarcan todo el contenido de la imagen por lo que no hay una buena preservación. De la misma manera si la imagen se reduce a un tamaño muy pequeño, en este caso a un tamaño menor 128×128 es decir un 25 % del tamaño original, las sub-imágenes extraídas, tienden a ser del tamaño de toda la imagen ó mayores a estas, por lo tanto no se puede generar un valor Hash.

En el caso de compresión JPEG, como se muestra en la figura 5.5, donde se tiene el factor de compresión contra la distancia de Hamming, y cada valor representa el promedio de diferentes imágenes atacadas y 100 % es la imagen sin compresión. Observamos que la distancia de Hamming obtenemos una reduccion por debajo de 0,1, y en comparación con el algoritmo de Kozat, se reduce

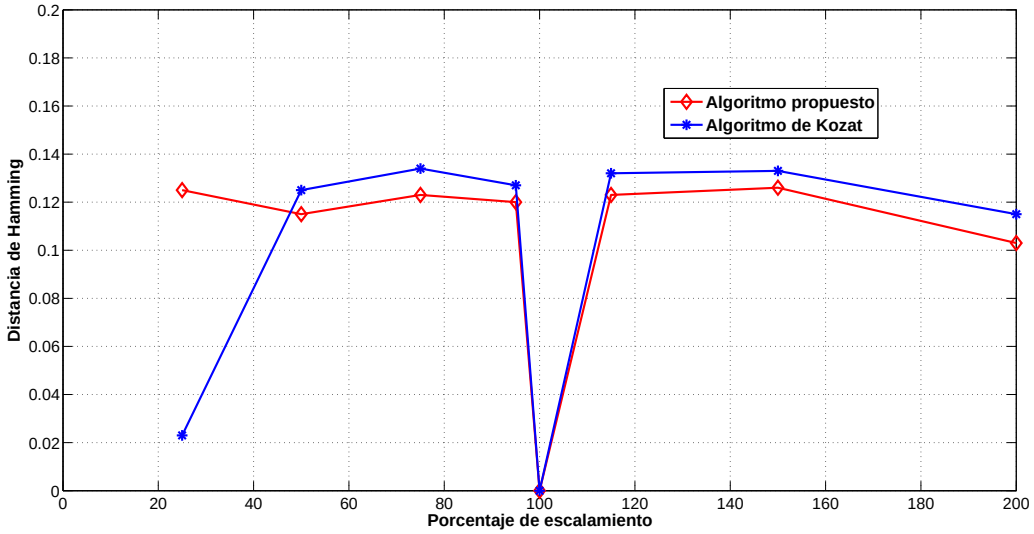


Figura 5.4: Ataque de escalamiento, porcentaje de escala vs distancia de Hamming, siendo 100 % la imagen con dimensiones originales.

la distancia de haming en promedio 0,04.

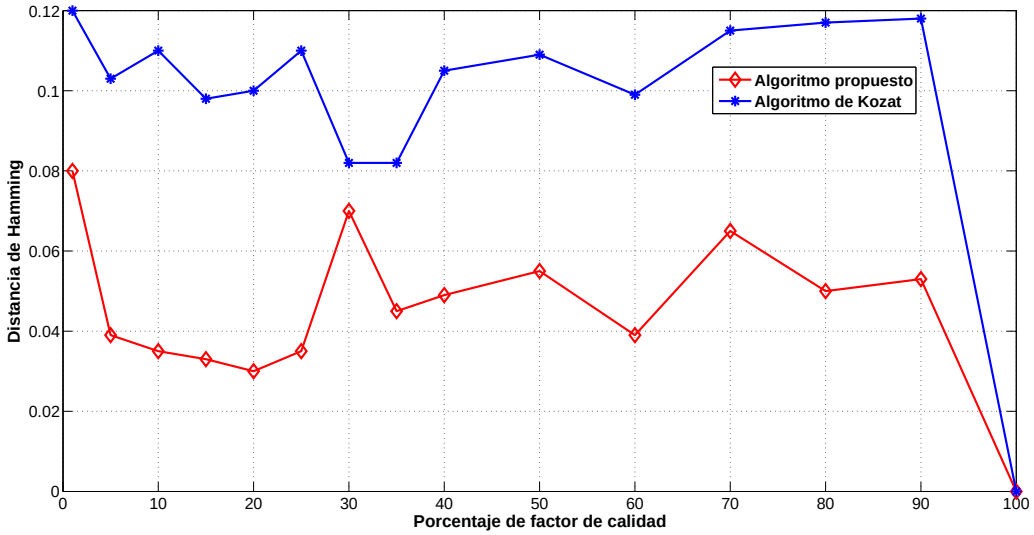


Figura 5.5: Ataque de compresión JPEG, factor de calidad en porcentaje vs distancia de Hamming, siendo 100 % la imagen sin compresión.

En el caso de Cropping como se observa en la figura 5.6, sólo es capaz de detectarlo correctamente hasta un 10 % de cropping siempre y cunado el contenido que se encuentra en el centro de la imagen no se vea afectada.

Usando diferentes parámetros para la compresión (decodificador de Reed-Muller), no sólo se ve afectada la longitud del vector Hash como se muestra en la tabla 5.2, si no que mientras menor cantidad de bits en el vector Hash se vuelve menos robusto a diferente tipo de ataques como se muestra en la figura 5.7 que es un ejemplo de ataque de rotación, pero en comparación con el

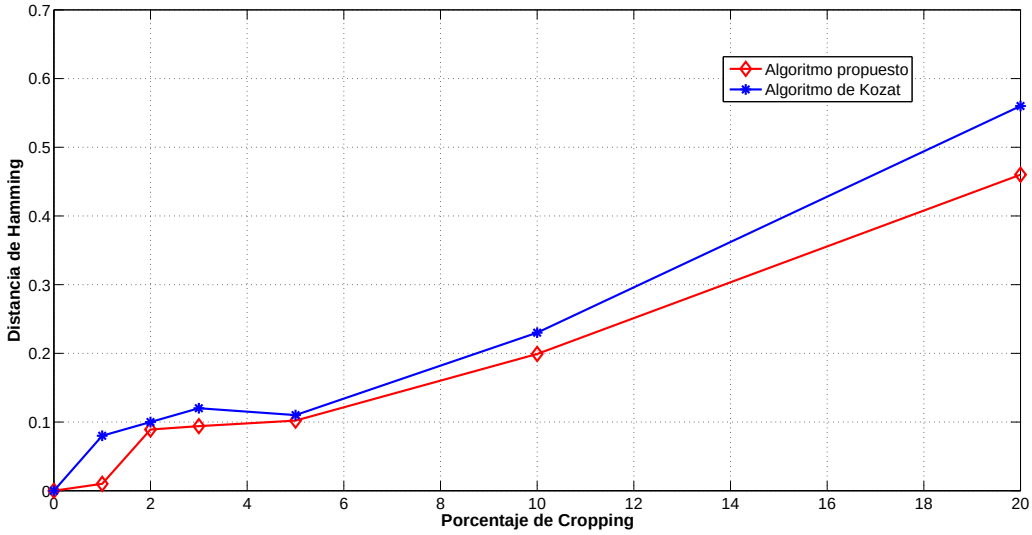


Figura 5.6: Ataque de cropping, porcentaje de cropping vs distancia de Hamming, siendo 0 % la imagen sin cropping.

algoritmo de Kozat se ve una mejora en términos de la distancia de Hamming.

Orden	m	k	Longitud Vector Hash
Primer	4	7	130
Segundo	4	3	420
Tercer	4	1	760

Tabla 5.2: Parámetros de usados para el decodificador Reed-Muller y la longitud del valor Hash obtenido.

Se realizaron pruebas con imágenes con cambio de contenido, como se muestra en la figura 5.8 se puede observar que conforme el porcentaje de pixeles modificados se incrementa la distancia de Hamming tiende hacia el valor de 1, como se puede observar que a partir del 10 % de pixeles modificados la distancia de Hamming sobre pasa el valor de 0,1.

5.4. Conclusión

De la figura 5.8 se puede observar que la distancia de Hamming aumenta significativamente después de un 10 % de pixeles modificados, y observando los diferentes resultados obtenidos con diversos ataques geométricos podemos establecer como valor umbral de autenticación de 0,1 aproximadamente, es decir que al comparar dos vectores Hash si la distancia de Hamming se encuentra por debajo de 0,1 podemos decir que ambas imágenes contienen el mismo contenido, sin embargo si el valor se encuentra por encima de 0,1 amabas imágenes son distintas.

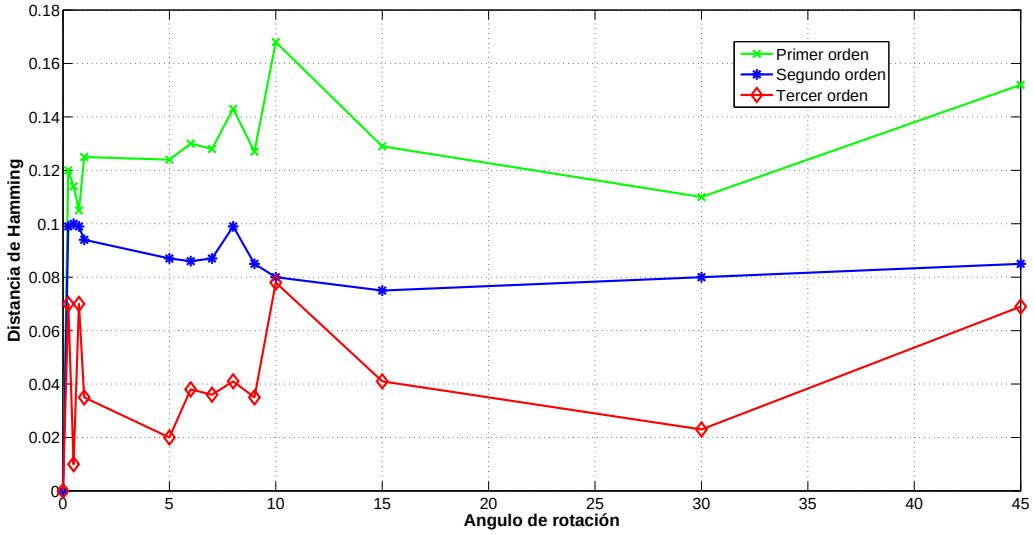


Figura 5.7: Comparación de diferentes parámetros de códigos de Reed-Muller sobre el ataque de rotación.

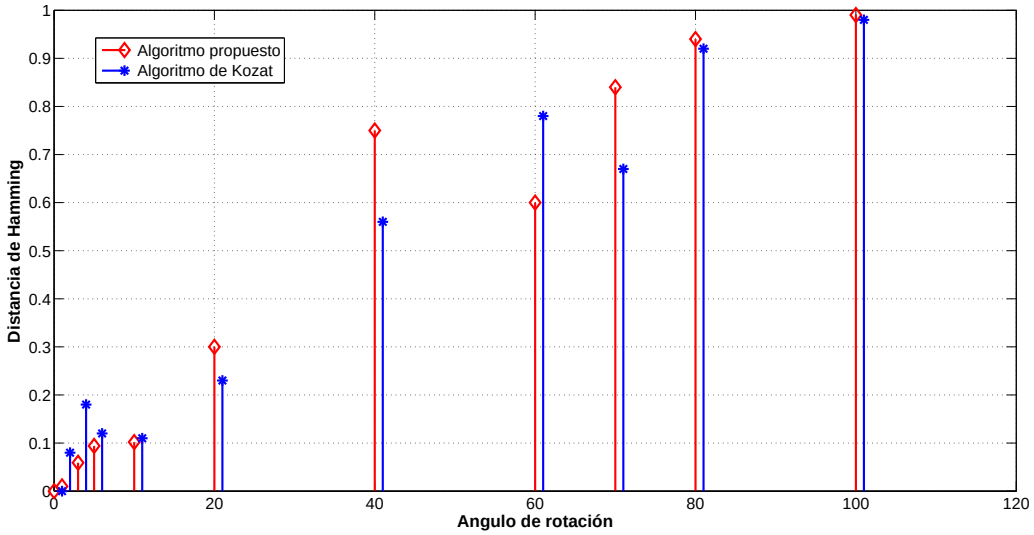


Figura 5.8: Comparación de diferentes parámetros de códigos de Reed-Muller sobre el ataque de rotación.

Capítulo 6

Conclusiones

6.1. Conclusiones

En este trabajo se presento el algoritmo de normalización de imágenes como pre-procesamiento en el algoritmo de Image Hashing que utiliza descomposición SVD como generador del vector Hash.

Se puede observar que este algoritmo presenta una buena conservación de características relevantes de contenido. Además de ser seguro al introducir el uso de una llave secreta.

Otras de las observaciones es que en caso de ataques geométricos, al usar el algoritmo de normalización de imágenes, hay una mejora significativa en términos de la distancia de Hamming. Por lo que este algoritmo presenta una buen desempeño como pre-procesamiento en el problema de Image Hashing, por lo cual resulta un esquema más robusto en contra de transformaciones Affine.

En este trabajo se propuso un valor umbral de 0,1 en términos de la distancia de Hamming para determinar si dos imágenes son idénticas o no.

Aunque en este trabajo no se enfoca en el análisis de la forma en que se cuantifica y se comprime el valor Hash final, añadiendo estos pasos en el algoritmo de Kozat y en el esquema propuesto, no modifica mucho en el desempeño de ambos algoritmos, sin embargo investigaciones posteriores pueden explorar diferentes parámetros asi como diferente tipos de códigos para poder realizar una mejor preservación de contenido y mejorar el desempeño de ambos esquemas.

6.2. Trabajo futuro

A continuación mencionaremos algunos trabajos que se pueden realizar para continuar con la investigación:

1. Usar el algoritmo de normalización de imágenes como pre-procesamiento para incrementar la robustez de los algoritmos de Image Hashing basados en detectores de puntos relevantes, algoritmos basados en diferentes transformaciones como Fourier-Melling.
2. Explorar diferentes parámetros para el numero de sub-imágenes, el tamaño de las sub-imágenes extraídas para observar que tanta información es preservada en la extracción de características.
3. Usar diferente tipos de códigos como lattice para la cuanización y compresión del vector Hash final.

Bibliografía

[1] M. Wu A. Swaminathan, Y. Mao. Robust and secure image hashing. *IEEE Trans. on Information Forensies and Security*, 1(2):215 –230, November 2006.

[2] M. Kivanc Mihcak Suleyman S. Kozat, Ramarathnam Venkatesan. Robust perceptual image hashing via matrix invariants. *International Conference on Image Processing, 2004. ICIP '04. 2004*, 5:3443–3446, 2004.

[3] Stirmark benchmark, Jannuary 1999.

[4] V. Monga and B.L. Evans. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. *Image Processing, IEEE Transactions on*, 15(11):3452 –3465, November 2006.

[5] J. Oostveen c T. Kalker, J.A. Haitsma. Issues with digital watermarking and perceptual hashing. In *SPIE 4518, Multimedia Systems ans Applications IV*, pages 189–197, November 2001.

[6] T. Kalker J.A. Haitsma. A highly robust audio finger printing system. In *Proceedings of the the international Conference on Music Information Retrieval(ISMIR)*, volume 1, October 2002.

[7] T. Kalker J.A. Haitsma J. Oostveen, c. Feature extraction and a database strategy for video fingerprinting. *Proceedings of the International Conference on Visual Information Systems*, March 2002.

[8] G.L. Friedman. The trustworthy digital camera: restoring credibility to the photographic image. *Image Processing, IEEE Transactions on*, 39:905–910, November 1993.

[9] M. Bhattacharjee, S. Kutter. Compression tolerant image authentication. *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference*, 1:435, October 1998.

[10] B. Girod J. Eggers. Blind watermarking applied to image authentication. In *Proceedings of the IEEE ICASSP 2001*, May 2001.

[11] Nikolas P. Galatsanos Yongyi Yang Franck Davoine Ping Dong, Jovan G. Brankov. Digital watermarking robust to geometric distortions. *IEEE transactions on image processing*, 14(12):2140–2150, 2005.

[12] R. Venkatesan, S.-M. Koon, M.H. Jakubowski, and P. Moulin. Robust image hashing. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 3, pages 664 –666 vol.3, February 2000.

[13] C. Schimd A. Zisserman J. MAtas . Schaffalitzky T. Kadir k. Mikolajczyk, T. Tuytelaars and L. Van Gool. A compasion of affine region detectors. *Int. Comput. Vis.*, 39:905–910, November 1993.

[14] M. Stephen C. Harris. A combined corner and edge detector. *Proc. Alvey Visual Conference*, pages 147–151, September 1998.

[15] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detector. *Int J. Comput Vis*, pages 63–68, October 204.

[16] M Urban J. Matas, O. Schum and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Proc. Brit. Mach. Vision Conf.*, pages 384–393, March 2002.

[17] J. P. Antoine and R. Murenzi. Two-dimensional directional wavelets and the scale-angle representation. *Signal process*, pages 259–281, March 2002.

[18] J. J. K. O'Ruanaid and T.Pun. Rotation, scale and translation invariant digital image watermarking. *Proc. IEEE Int. conf. Image Processing,*, 1(11):536–539, October 1997.

[19] Xiamu Niu Di Wu, Xuebing Zhou. A novel image hash algorithm resistant to print-scan. *ELSEVIER Signal procesing*, 89:2415–2424, 2009.

[20] Guoliang Zeng. Face recognition with singular value decomposition. *CISSE Procededing*, 2006.

[21] Lijie Cao. Singular value decomposition applied to digital image procesing. *Division of Computing Studies Arizona States University*, 2008.

[22] Steve J. Leon. Linear algebra with aplications. *Macmillan Publishing Company*, 1996.

[23] Pentland P. Alex Matthew A. Turk. Face recognition using eigenface method. *IEEE conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[24] D. Shen and H. S. Ip. Generalized affine invariant image normalization. *IEEE trans. Pattern Anal. Mach. Intell*, 18(5):366–376, 1997.

[25] K. K. T. Cheung E. K Teoh D. Shen, H. S. Ip. Symetry deteccion by generalized complex (gc) moments: A close-form solution. *IEEE trans. Pattern Anal. Mach. Intell*, 21(5):466–476, 1999.

[26] A. Gersho and R. M. Gray. Vector quantization and signal compression. *Kluwer Academic*, 1991.

Apéndices

Apéndice A

Software

Se muestra el código para la generación de valor hash de una imagen digital, se extrae tanto el valor hash final y el valor hash intermedio.

A.1. Generador de Hash

```
%% Generador de Image Hash
%Input
%I = la imagen en escala de grises a 8 bits
%Output
%hashVec =vector hash binario
%hashVec1= vector hash intermedio, numeros reales de longitud 2*numrects + rectsize

function [hashVec hashVec1]=genehashsvd1(I)
key=10; %llave secreta
numrects=15; %numero de sub-imagenes
rectsiz=100; %dimension de la sub-imagen
I1=I;
I1=round(imnorm(I1));
I1=histeq(uint8(I1));
hashVec = hashbySVD(double(I1), key, numrects, rectsiz);
hashVec1=hashVec;
hashVec=round(hashVec/max(hashVec)*255);
[y,map] = bin2gray(hashVec,'qam',256);
c1=dec2bin(y,8);
h1a=(double(c1)-48)';
h1=h1a(:);
hashVec=reedmu(h1');
return
```

A.2. Extracción de características

```
function [hashVec] = hashbySVD(imgin, key, numRects, rectsize);

% HASHBYSVD Image Hashing by singular value decomposition
%  Description
%  [hashVec] = hashbySVD(imgin, key, numRects, size) extracts
```

```
%  hash vector hashVec from input image "imgin". The basic approach
%  is to divide the image into random rectangles selected using
%  the secret key "key" and obtain a singular value decomposition (SVD)
%  of each rectangle. "numRects" is the number of input rectangles and
%  "rectsize" is the size of each rectangle. A random subset of the most
%  significant singular vectors and singular values forms the hash. Details
%  can be found in the reference below.
%
%   See also WAVELETHASH, FEATUREPOINTHASH
%
%   Ref:
%   1.) S. S. Kozat, K. Mihcak, and R. Venkatesan, "Robust
%   perceptual image hashing via matrix invariances", Proc. IEEE
%   Conf. on Image Processing, pp. 3443-3446, Oct. 2004.
%
% Authored 2005 by Vishal Monga
% Copyright (c) 1999-2005 The University of Texas
% All Rights Reserved.

[siz siz1]= size(imgin);

if siz>rectsize && siz1>rectsize

rand('state',key);
xcoors = ceil(rand(numRects,1)*(siz-rectsize+1));
ycoors = ceil(rand(numRects,1)*(siz1-rectsize+1));

umat = zeros(rectsize,2*numRects);
uumat1 = zeros(rectsize,numRects);
vvmat1 = zeros(rectsize,numRects);

for i=1:1:numRects,
   subim = imgin(xcoors(i):(xcoors(i)+rectsize-1),ycoors(i):(ycoors(i)+rectsize-1));
   tt = subim+randn(size(subim));
   [u1,s1,v1] = svds(tt,1);
   uumat1(:,i) = sign(sum(u1))*u1;
   vvmat1(:,i) = sign(sum(v1))*v1;
end;
umat = [uumat1 vvmat1];


[u,s,v] = svds(umat,1);
u = sign(sum(u))*u;
v = sign(sum(v))*v;
hashVec = [u;v];
else
    hashVec=ones(rectsize+ 2*numRects,1 );
end
return;
```

A.3. Normalización de imágenes

```
function normim = imnorm(im)
%
% Implementation of the image normalization part of:
%   1. P. Dong, J.G. Brankov, N.P. Galatsanos, Y. Yang, and F. Davoine,
%      "Digital Watermarking Robust to Geometric Distortions," IEEE Trans.
%      Image Processing, Vol. 14, No. 12, pp. 2140-2150, 2005.
%   2. P. Dong and N.P. Galatsanos, "Affine Transformation Resistant
%      Watermarking Based on Image Normalization," Int. Conf. Image
%      Processing, pp. 489-492, 2002.
%
% Input:
%   im: input grayscale image of class uint8
%
% Output:
%   normim: normalized grayscale image of class double
%
%
% Copyright (c), Yuan-Liang Tang
% Associate Professor
% Department of Information Management
% Chaoyang University of Technology
% Taichung, Taiwan
% http://www.cyut.edu.tw/~yltang
%
% Permission is hereby granted, free of charge, to any person obtaining
% a copy of this software without restriction, subject to the following
% conditions:
% The above copyright notice and this permission notice should be included in
% all copies or substantial portions of the Software.
%
% The Software is provided "as is", without warranty of any kind.
%
% Created: May 2, 2007
% Modified: Aug. 7, 2007
%

im = cutborder(double(im));   % Cut black borders, if any

% Normalization steps:
% 1. Translation invariance: translate coordinate to the image centroid
[cx cy] = imcentroid(im);
tform = maketform('affine', [1 0 0; 0 1 0; -cx -cy 1]);
[translateim xdata ydata] = imtransform(im, tform, 'bilinear');
 showim(translateim, xdata, ydata, 'Translate');

% 2. X-shearing invariance
u03 = immoment(translateim, 0, 3, cx, cy);
u12 = immoment(translateim, 1, 2, cx, cy);
u21 = immoment(translateim, 2, 1, cx, cy);
u30 = immoment(translateim, 3, 0, cx, cy);
rts = sort(roots([u03 3*u12 3*u21 u30]));
```

```
if isreal(rts)     % All roots are real: choose the medium one
  beta = rts(2);
else               % Not all roots are real: choose the real one
  for i = 1:3
    if isreal(rts(i))
      beta = rts(i);
      break
    end
  end
end
tform = maketform('affine', [1 beta 0; 0 1 0; 0 0 1]');
[xshearim xdata ydata] = imtransform(translateim, tform, 'bilinear', ...
  'udata', xdata, 'vdata', ydata);
[xshearim xdata ydata] = cutborder(xshearim, xdata, ydata);
showim(xshearim, xdata, ydata, 'Xshear');

% 3. Y-shearing invariance
[cx cy] = imcentroid(xshearim);
u11 = immoment(xshearim, 1, 1, cx, cy);
u20 = immoment(xshearim, 2, 0, cx, cy);
gamma = -u11/u20;
tform = maketform('affine', [1 0 0; gamma 1 0; 0 0 1]');
[yshearim xdata ydata] = imtransform(xshearim, tform, 'bilinear', ...
  'udata', xdata, 'vdata', ydata);
[yshearim xdata ydata] = cutborder(yshearim, xdata, ydata);
showim(yshearim, xdata, ydata, 'Yshear');

% 4. Anisotropic scaling invariance: scale to a standard size of
%    [256 256] and ensure u50>0 and u05>0
[r c] = size(yshearim);
alpha = 512/c;
delta = 512/r;
tform = maketform('affine', [alpha 0 0; 0 delta 0; 0 0 1]);
[scaleim xdata ydata] = imtransform(yshearim, tform, 'bilinear', ...
  'udata', xdata, 'vdata', ydata);
[cx cy] = imcentroid(scaleim);
if immoment(scaleim, 5, 0, cx, cy) < 0
  scaleim = fliplr(scaleim);   % Flip left and right
  xdata = -xdata(2:-1:1);  % Swap elements
  [cx cy] = imcentroid(scaleim);
end
if immoment(scaleim, 0, 5, cx, cy) < 0
  scaleim = flipud(scaleim);   % Flip up and down
  ydata = -ydata(2:-1:1);  % Swap elements
end
showim(scaleim, xdata, ydata, 'Scale');

% 5. Rotation invariance
[cx cy] = imcentroid(scaleim);
u30 = immoment(scaleim, 3, 0, cx, cy);
u12 = immoment(scaleim, 1, 2, cx, cy);
u03 = immoment(scaleim, 0, 3, cx, cy);
u21 = immoment(scaleim, 2, 1, cx, cy);
```

```
phi = atan(-(u30+u12)/(u03+u21+eps));
tform = maketform('affine', [cos(phi) sin(phi) 0; -sin(phi) cos(phi) 0; 0 0 1]);
[normim xd yd] = imtransform(scaleim, tform, 'bilinear', ...
  'udata', xdata, 'vdata', ydata);
[cx cy] = imcentroid(normim);
u03 = immoment(normim, 0, 3, cx, cy);
u21 = immoment(normim, 2, 1, cx, cy);
if u03+u21 < 0
  phi = phi + pi;
  tform = maketform('affine', [cos(phi) sin(phi) 0; -sin(phi) cos(phi) 0; 0 0 1]);
  [normim xd yd] = imtransform(scaleim, tform, 'bilinear', ...
    'udata', xdata, 'vdata', ydata);
end
xdata = xd;
ydata = yd;
[normim xdata ydata] = cutborder(normim, xdata, ydata);
showim(normim, xdata, ydata, 'Normalized');

function m = immoment(im, p, q, cx, cy)
% Compute image moment
[y x v] = find(im);
if nargin == 5
  x = x - cx;
  y = y - cy;
end
m = sum(x.^p .* y.^q .* v)/sum(v);


function [cx, cy] = imcentroid(im)
% Computer image centroid
cx = immoment(im, 1, 0);
cy = immoment(im, 0, 1);


function [imout, xdata, ydata] = cutborder(imin, xdata, ydata)
% Cut black borders of the image and adjust xdata and ydata values
[y x] = find(imin);
imout = imin(min(y):max(y), min(x):max(x));
if nargin == 3
  [r c] = size(imin);
  xdata(1) = xdata(1) + min(x) - 1;
  xdata(2) = xdata(2) - c + max(x);
  ydata(1) = ydata(1) + min(y) - 1;
  ydata(2) = ydata(2) - r + max(y);
end
```

Apéndice B

Demostración de la normalización de imágenes

Como se menciono en la sección 3.4 la normalización de imágenes se puede ver como una composicion de las transformaciones de traslación, shering y escalamiento.

Se puede ver que la normalización de imágenes es invariante a la traslación, esto es porque cualquier traslación en la imagen es removida en el paso donde se centra la imagen.

A continuación demostraremos que la normalización de una imagen $f(x, y)$ es invariante a las otras transformaciones. como se menciona en[11].

Usaremos $g(x, y)$ como la imagen transformada de $f(x, y)$ despuś de aplicarle una transformación affine. También usaremos $\mu_{pq}^{(g)}$ y μ_{pq} para denotar los momentos de $g(x, y)$ y $f(x, y)$ respectivamente.

B.1. Shearing en la dirección de x

En este caso $g(x, y) = f(x_a, y_a)$ donde:

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & \beta_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{B.1}$$

Si consideramos que β la resolvemos con

$$\mu_{30}^{(g)} + 3\beta\mu_{21}^{(g)} + 3\beta^2\mu_{12}^{(g)} + \beta^3\mu_{03}^{(g)} = 0. \tag{B.2}$$

Lema 1: si $g(x, y)$ es una transformación affine de $f(x, y)$ y es obtenida con una transformaciń $A = \left(\begin{smallmatrix} a_{1,1} & a1,2 \\ a_{2,1} & a_{2,2} \end{smallmatrix}\right)$y $d = 0$, entonces las siguientes identidades son ciertas:

$$\mathrm{m}_{p,q}^{(g)} = \sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} \binom{q}{j} a_{1,1}^{i} a_{1,2}^{p-i} a_{2,1}^{j} a_{2,2}^{q-j} m_{i+j,p+q-i-j} \tag{B.3}$$

$$\mu_{p,q}^{(g)} = \sum_{i=0}^{p} \sum_{j=0}^{q} \binom{p}{i} \binom{q}{j} a_{1,1}^{i} a_{1,2}^{p-i} a_{2,1}^{j} a_{2,2}^{q-j} \mu_{i+j,p+q-i-j} \tag{B.4}$$

Podemos escribir los momentos $\mu_{p,q}^{(g)}$ en (B.2) en términos de $\mu_{p,q}$ usando (B.4), podemos reescribir (B.2) despues de un poco de algebra como:

$$\mu_{30} + 3(\beta + \beta_0)\mu_{21} + 3(\beta + \beta_0)^2\mu_{12} + (\beta + \beta_0)^3\mu_{03} = 0. \tag{B.5}$$

Hagamos que $\beta' \cong \beta + \beta_0$, una vez que veamos que β' satisface la ecuación de shering con el parámetro β para normalizar la imagen original $f(x, y)$. Hagamos que A'_x denote la transformada de shearing correspondiente es $A'_x = \left(\begin{smallmatrix} 1 & \beta' \\ 0 & 1 \end{smallmatrix}\right)$. Observe que

$$A_x A = \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \beta_0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & \beta + \beta_0 \\ 0 & 1 \end{pmatrix} = A'_x. \tag{B.6}$$

Esto es la normalización de shearing en $g(x, y)$ dará la misma imagen de la normalización de shearing en $f(x, y)$.

B.2. Shearing en la dirección de y

En este caso $g(x, y) = f(x_a, y_a)$ donde:

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \gamma_0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{B.7}$$

Como se hizo anteriormente escribimos los momentos $\mu^{(g)}_{p,q}$ en (B.2) en términos de $\mu_{p,q}$ para obtener:

$$\mu_{30} + 3(\frac{\beta}{1 + \beta\gamma_0})\mu_{21} + 3(\frac{\beta}{1 + \beta\gamma_0})^2\mu_{12} + (\frac{\beta}{1 + \beta\gamma_0})^3\mu_{03} = 0. \tag{B.8}$$

Podemos escribir:

$$\mathrm{A}_y A_x A = \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \gamma_0 & 1 \end{pmatrix} = \begin{pmatrix} 1 + \beta\gamma_0 & 0 \\ \gamma + \gamma_0(1 + \beta\gamma) & 1 + \beta\gamma \end{pmatrix},$$

despues de un poco de algebra podemos escribir la ecuación anterior como:

$$\mathrm{A}_y A_x A = \begin{pmatrix} 1 + \beta\gamma_0 & 0 \\ 0 & \frac{1+\beta\gamma_0}{(1+\beta\gamma_0)^2\mu_{2,0}+2\beta(1+\beta\gamma_0)\mu_{1,1}+\beta^2\mu_{0,2}} \end{pmatrix} \times \begin{pmatrix} 1 & \beta' \\ -\mu_{1,1} - \beta'\mu_{0,2} & \mu_{2,0} - \beta'\mu_{1,1} \end{pmatrix}$$

Podemos observar que la segunda matriz corresponde a una transformación Affine que es independiente del parámetro γ_0, por lo que la imagen resultante $g(x, y)$ es invariante a la transformación Affine A que es parame trizada por γ_0

B.3. Escalamiento en la dirección x e y

En este caso $g(x, y) = f(x_a, y_a)$ donde:

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha_0 & 0 \\ & \delta_0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{B.9}$$

Una vez más escribimos los momentos $\mu^{(g)}_{p,q}$ en (B.2) en términos de $\mu_{p,q}$ para obtener:

$$\mu_{30} + 3(\frac{\delta_0}{\alpha_0}\beta)\mu_{21} + 3(\frac{\delta_0}{\alpha_0}\beta)^2\mu_{12} + (\frac{\delta_0}{\alpha_0}\beta)^3\mu_{03} = 0. \tag{B.10}$$

Hagamos que $\beta' \cong (\delta_0/\alpha_0)\beta$, una vez que veamos que β' satisface la ecuación de shering con el parámetro β para normalizar la imagen original $f(x, y)$.

$$\mathrm{A}_y A_x A = \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_0 & 0 \\ & \delta_0 \end{pmatrix} = \begin{pmatrix} \alpha_0 & \beta\delta_0 \\ \alpha_0\gamma & \delta_0(1+\beta\gamma) \end{pmatrix},$$

Despues de un poco de algebra tenemos:

$$\mathrm{A}_y A_x A = \begin{pmatrix} 0 & 0 \\ 0 & \frac{\alpha_0^2\delta}{alpha_0^2\mu_{2,0}+2\beta\alpha_0\delta_0\mu 1,1+\beta^2\delta_0^2\mu_{0,2}} \end{pmatrix} \times \begin{pmatrix} 1 & \beta' \\ -\mu_{1,1} - \beta'\mu_{0,2} & \mu_{2,0} - \beta'\mu_{1,1} \end{pmatrix}$$

Una vez mas la segunda matriz corresponde a una transformación Affine independiente de los par¡ametros α_0 y δ_0, y la primera matriz corresponde a una matriz de escalamiento. Sin embargo la imagen normalizada $g(x, y)$ es invariante a la transformación Affine A con parámetros α_0 y δ_0

Apéndice C

Glosario de términos

Algoritmo de partición aleatoria : Algoritmo que nos permite extraer sub-imágenes aleatoriamente de una imagen digital.

Descopmposicion SVD : De sus siglas en inglés Singular Value Deomposition, es uno de los conceptos básicos de algebra lineal. Utiliza los vectores propios y valores propios.

Detector de Borde : una herramienta que nos permite identificar puntos en la imagen digital donde el brillo presenta discontinuidades.

Funcion Hash : procedimiento bien definido o una función matemática que convierte una gran cantidad de información a un dato de menor dimensión.

Image Hash : es una firma digital basada en el contenido de la imagen.

Imagen Digital : Es una representación bidimensional de una fotografía.

Detector MSER : Maximally Stable Extremal Regions. detector de región máxima estable, la idea básica consiste en que todos los pixeles por debajo de un umbral dado son negros y todos los pixeles por encima del umbral son blancos.

Normalización de imágenes : algoritmo que nos permite obtener una imagen invariante a transformaciones Affine.

Pixel : la representación mas pequeña de un color o tonalidad .

Transformada Affine : es aquella que las coordenadas (x^2, y^2) son expresadas en términos de las coordenadas originales (x, y). Concentra las transformaciones geométricas como rotación, escalamiento y shearing.

Transformada de Fourier-Mellin : método que a mostrado ser invariante a diferentes tipos de transformada Affine.

Transformada de Radon : basada en el contenido y la luminancia de una imagen.

Valor propio : es el factor de escala que se ha multiplicado a un vector propio.

Vectores propios o eigenvectores : son los vectores no nulos, que al ser transformados por un operador lineal dan como resultado un múltiplo escalar de si mismo.

Apéndice D

Publicaciones realizadas

1. Ricardo Antonio Parrao Hernández, Mariko Nakano Miayatake, Brian M. Kurkoski,"Increase robustness of the SVD decomposition hashing function with image normalization algorithm", Proceedings of UEC International conference for exchange students XXV, pp. 47-52, march 2011
2. Ricardo Antonio Parrao Hernández, Mariko Nakano Miayatake, Brian M. Kurkoski, " Robust image Hashing using image normalization and SVD decomposition", IEEE MWSCAS, Aug 2011
3. Ricardo Antonio Parrao Hernández, Mariko Nakano Miayatake, Brian M. Kurkoski, " Robust image hashing against geometric attacks using image normalization and SVD decomposition", IEICE conference, pp. 221-226, march 2011

Increase robustness of the SVD decomposition hashing function with image normalization algorithm

Mariko NAKANO MIYATAKE, Ricardo Antonio PARRAO HERNANDEZ *
UEC Student No.1095011
SEPI, ESIME Culhuacan, National Polytechnic Institute IPN
Av. Santa Ana Num. 1000, Mexico City, 04430 Mexico

Brian M. KURKOSKI
Department of Information and Communications Engineering
The University of Electro-Communications
Tokyo #182-8585. Japan

March 02-03, 2011

Abstract

This paper introduce the use of image normalization algorithm and SVD decomposition to generate a hash value of a digital image. SVD decomposition is an important linear algebra tool, which is often used in pattern recognition, digital watermarking and other signal processing fields, on the other hand image normalization is an algorithm that has been shown to be robust to different kinds of geometric attacks such like rotation, scaling and flipping. using the image normalization algorithm as a preprocessing step we can increase the robustnes of the hash value.

Keywords: Image hashing, SVD decomposition, Image normalization, Invariants moments, Geometric attacks

1 Introduction

During the last years ,the copyrights issues are becoming an important problem for multimedia data. Tools like watermarking and digital signatures has had tremendous growth to ensure trustworthines and prevent forgery.

An image hash is an image-based content digital signature. To generate the hash, a secret key is used to extract some features from the content of the image. These features are processed to generate the hash. In addition to content authentication, multimedia hashes are used in content-based retrieval from databases [6]. To search for multimedia content, methods such as sample-by-sample comparisons are computationally inefficient. Moreover, these methods compare the lowest level of content representation and do not offer robustness in such situations as geometric distortions or different kind of compresion. Robust image hash functions can be used to address this problem [9].

Image hash functions have been used also for image and video watermarking, instead of using the original image for embeding, we can use the image hash value[4]. The hash fucntions have also been used as independent key for watermarking [4].

There are two important criterias in the design of the image hash function, robustness and security [9]. By robustness, we mean that if the same key is used, perceptually similar images generate a similar hash. The similiraty of hashes is mesured in terms of metrics like Hamming or Euclidean distances. The security of the image hash function is by using a secret key to generate the hash. Without the knowledge of this key, the hash value should not be easy to estimated or forge.

The SVD decomposition is an important linear algebra tool, which is often used in image compression, digital watermarking and other signal processing fields. The SVD decomposes the image in to three matrices, U, S and V. The matrix S, is a diagonal matrix contaning the singular values of the image. The matrices U and V contain the singular vectors of the image, and by applying this decomposition we can reduce the dimensions of the data. On the other hand, image normalization is an algorithm based on the invariant moments of the image. These moments are

*the author is supported by JASSO Scholarship.

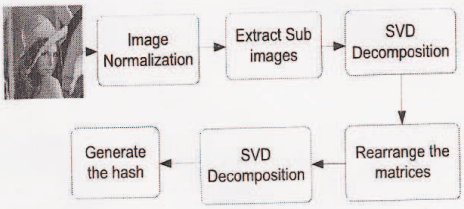


Figure 1: Proposed algorithm for generate the hash value

used to geometrically normalize the image in terms of scale, orientation, and flipping . This method is well-known in pattern recognition problems.

This paper introduces image normalization as a preprocessing step in the SVD decomposition hash function to increase robustness against geometric attacks. For increasing the security and introduce unpredictability in the hash value we randomlyselect sub-images, with a random partitioning algorithm [7], and apply the SVD decomposition hash function to each sub-image, after that we rearrange the matrices U and V and again apply SVD decomposition for generate the final hash value, in this case the hash value is a sequence of real numbers. The proposed algorithm is shown in Figure 1.

This paper contains five sections. Section1 introduces a brief description of why is important to have a hash based on the content of a digital image. Section 2 describes how to use SVD decomposition, the image normalization algorithm and the random partition algorithm, Section 3 describes the proposed algorithm, Section 4 present the results, Section 5 is the conclusion and the posible future work for this paper.

2 Definitions

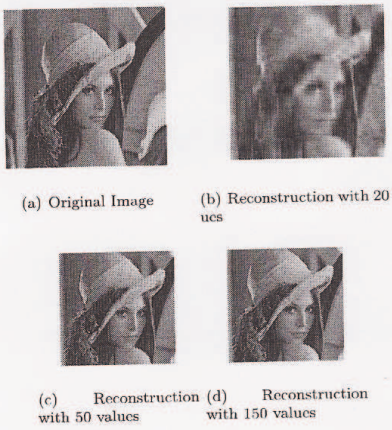
In this section we introduce the theory behind the singular value decomposition (SVD), the characteristics of the image normalization algorithm and the random partition algorithm.

2.1 SVD Decomposition

We can see an image as a M-by-N matrix, so we can apply the SVD decomposition. We have an image A and apply the SVD decomposition as:

$$A = USV^T, \tag{1}$$

where U and V are orthogonal matrices that contain the left and right singular vectors and S is a diagonal



(a) Original Image (b) Reconstruction with 20 values

(c) Reconstruction with 50 values (d) Reconstruction with 150 values

Figure 2: Use of different numbers of singulars values and vectors for reconstructing the original image

matrix $S = \text{diag}(s_1, s_2, \ldots)$ with the singular values of A. We can express the decomposition as the following:

$$A = \sum_{i=0}^{r} U_i s_i V_i^T, \tag{2}$$

$$A = U_1 s_1 V_1^T + U_2 s_2 V_2^T + \ldots + U_r s_r V_r^T, \tag{3}$$

where r is the rank of A.

The SVD decomposition for digital images has the following properties[3]:

1. The singular values change little when a small disturbance is applied.

2. The singular values represent the brightness features of the image, while the singular vectors represent the geometrical features.

3. The quality of the reconstructed image does not degenerate if we remove the small-valued singular values s_i. This property we can be seen it in Figure 2 were we do not need all the singular values to generate an image with a good quality.

2.2 Image normalization algorithm

Image normalization is an algorithm based on the geometric moments of the image. With this algorithm,

an image F is robust to different kinds of affine transforms [2]. The algorithm is as follows:

Let F denote a digital image of size M-by-N. Its geometric moments m_{pq} and central moment μ_{pq},where $p, q = 0, 1, 2, \ldots$ are defined as:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q F(x, y). \quad (4)$$

And,

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q F(x, y), \quad (5)$$

where

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}}, \quad \bar{y} = \frac{m_{0,1}}{m_{0,0}}. \quad (6)$$

An image G is said to be an affine transform of F if there is a matrix $H = \left(\begin{smallmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{smallmatrix}\right)$ and a vector $d = (d_1\ d_2)^T$ such that $G(x, y) = F(x_a, y_a)$, where

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = H \begin{pmatrix} x \\ y \end{pmatrix} - d. \quad (7)$$

The normalization procedure consists of the following steps for a given unnormalized image F:

1. Center the image F using the matrix $H = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and the vector $d = (d_1\ d_2)^T$ with:

$$d_1 = \frac{m_{1,0}}{\mu_{0,0}}, \quad d_2 = \frac{m_{0,1}}{\mu_{0,0}}. \quad (8)$$

   This step performs translation invariance. Let F_1 denote the resulting centered image.

2. Apply a shearing transform to F_1 in the x direction with matrix $A = \left(\begin{smallmatrix} 1 & \beta \\ 0 & 1 \end{smallmatrix}\right)$ so the resulting image is denoted by F_2 To determine β we use:

$$\mu_{30} + 3\beta\mu_{21} + 3\beta^2\mu_{12} + \beta^3\mu_{03} = 0. \quad (9)$$

   We can have up to three roots, if one of the roots is real and the others are complex, we take the real one, if two or three roots are real we pick the median of the real roots, if all roots are complex β is 0, on some very unusual conditions β will have infinite number of solutions if all the moments are zero. This happens when the image is rotationally symetric [8].

(a) Original Image

(b) Image Rotate 45 degrees

(c) Output of image normalization algorithm

Figure 3: Input and output of the image normalization algorithm

3. Apply a shearing transform to F_2 in the y direction with matrix $H = \left(\begin{smallmatrix} 1 & 0 \\ \lambda & 1 \end{smallmatrix}\right)$, the resulting image is denoted by F_3, where λ is defined with the equation:

$$\mu_{11} = \lambda\mu_{20} + \mu_{11}, \quad (10)$$

   and making $\mu_{11} = 0$ we obtain:

$$\lambda = \frac{\mu_{11}}{\mu_{20}}. \quad (11)$$

4. Finally, the image F_3 is resized to a specified size.

In Figure 3 we can see the result of the image normalization algorithm applied to the lena image. Figure 3(a) is the original image of lena, Figure 3(b) is the lena image rotated 45 degrees. Figure 3(c) is the output of the image normalization when either (a) or (b) is applied as an input. With this techniq we observe that this algorithm is robust to scaling, rotation and flipping.

2.3 Random partition algorithm

For increasing the security of the hash value and adding unpredictability, we add some randomless by generating sub-images. We form a number p pseudo-randomly selected sub-images with the random partition algorithm as shown in Figure 4. This consist of:

1. With a key k pseudo-randomly generate a set W where $W = ((x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p))$, which is the coordinate pair of the top-left corner of the sub-image $A_1, A_2, \ldots, A_p$.

2. For each coordinate pair generate a square of size m-by-m, and each square is the output that we call sub-image A_i.

3 The proposed algorithm

Robust image hash function has two inputs: an image I and a secret key k and has one output, a hash, which is a short vector $\vec{h} = H_k(I)$. The hash function should possess perceptual properties:

1. Hash values for all perceptually identical images, this means when a person cannot find any difference in the content of the image should have the same hash value.

2. Perceptually different images, when two images do not havet the same content in terms of visual perception, should produce different hash values.

Next, we present the description of our proposed algorithm:

1. Let the M-by-N input image be $I \in [0, 255]^{M \times N}$. The number of sub-images is denoted by p and the size of this sub-image is denoted by m where $1 < m < M, N$, and the secret key for generate the hash is denoted by k. And the output $\vec{h}$ is the final hash value.

2. To I we apply the image normalization described in Section 2.2 to generate the normalized image I_1.

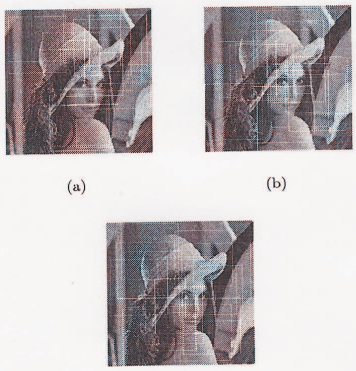
3. From I_1,with the use of a key k, pseudo-randomly form p possibly overlapping squares $A_1, A_2, \ldots, A_p$ each of size m-by-m as described in Section 2.3.

4. To each rectangle A_i apply the SVD decomposition to get $\{U_1 S_1 V_1, U_2 S_2 V_2, \ldots, U_p S_p V_p\}$ where matrices U_i and V_i are the singular vectors that have the content information of each sub-image and S_i are the singular values.

5. The first column of each U_i and the first row of each V_i^T compose a new matrix B.,

$$B = \left[U_1, \ldots U_p, V_1^T \ldots V_i^T\right], \qquad (12)$$

with size m-by-$2p$.



(a) (b) (c)

Figure 4: Examples of generating 15 sub-images of Lena, each image is partitioned with different key (a) key 1, (b) key 2 and (c) key 3

6. We apply the SVD decomposition to B to get the matrices U', S' and V'.

7. And finally, the first column of U' and the first row of V'^T are the image hash with length $m+2p$

8. output hash $\vec{h}$, $\vec{h} \in \mathbb{R}^{m+2p}$.

4 Results

To evaluate the performance of the algorithm we compare it with an algorithm without image normalization [5]. We apply different standard test images as an input and consider different kind of attacks such as rotation, compression, scaling and salt and pepper noise generating by the stirmark benchmark tool 4.0 [1]. These images are a 512-by-512 grayscale and also this is the size we use for the output in the image normalization algorithm. The parameters for the number of sub-images $p = 100$ and size $m = 50$.

For measure the difference d between two hashes $\vec{h}_1$ and $\vec{h}_2$ in the L_2 norm sense we use:

$$d = \left\| \frac{\vec{h}_1 - \vec{h}_2}{2 * \sqrt{\left\|\vec{h}_1\right\| \left\|\vec{h}_2\right\|}} \right\|. \qquad (13)$$

databases this characteristic is very useful because we can search based in the content of the image without taking care of the orientation or the size of the image.

We see the image normalization algorithm as a solution for geometric attacks, and this technique can be applied to different content-based image hashing methods to improve the performance, using it as a preprocessing step in the problem of the image hashing.

Acknowledgements: This research was sponsored by the UEC Japan through the JUSST program, and the CONACyT of Mexico.

References

[1] `http://www.petitcolas.net/fabien/watermarking/stirmark/`.

[2] Alghoniemy, M., and Tewfik, A. Geometric invariance in image watermarking. *Image Processing, IEEE Transactions on 13*, 2 (2004), 145 –153.

[3] Andrews, H., and Patterson, C. Singular value decompositions and digital image processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on 24*, 1 (Feb. 1976), 26 – 53.

[4] Cannons, J., and Moulin, P. Design and statistical analysis of a hash-aided image watermarking system. *Image Processing, IEEE Transactions on 13*, 10 (2004), 1393 –1408.

[5] Kozat, S., Venkatesan, R., and Mihcak, M. Robust perceptual image hashing via matrix invariants. In *Image Processing, 2004. ICIP '04. 2004 International Conference on* (2004), vol. 5, pp. 3443 – 3446 Vol. 5.

[6] Lin, S., Özsu, M. T., Oria, V., and Ng, R. T. An extendible hash for multi-precision similarity querying of image databases. In *Proceedings of the 27th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 2001), VLDB '01, Morgan Kaufmann Publishers Inc., pp. 221–230.

[7] Monga, V., and Evans, B. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. *Image Processing, IEEE Transactions on 15*, 11 (Nov. 2006), 3452 –3465.

[8] Shen, D., and Ip, H. Generalized affine invariant image normalization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*, 5 (May 1997), 431 –440.

[9] Venkatesan, R., Koon, S.-M., Jakubowski, M., and Moulin, P. Robust image hashing. In *Image Processing, 2000. Proceedings. 2000 International Conference on* (Feb. 2000), vol. 3, pp. 664 –666 vol.3.

http://www.mwscas2011.org

PRO CEEDIN GS

IEEE Catalog Number : CFP11MID-USB
ISBN : 978-1-61284-855-6

The 54 th IEEE International Midwest Symposium on Circuits and Systems

IEEE MWSCAS 2011

August 7-10, 2011
Yonsei University, Seoul, Korea

Robust Image Hashing Using Image Normalization and SVD Decomposition

Ricardo Antoio Parrao Hernandez
and Mariko Nakano Miyatake
SEPI, ESIME Culhuacan
National Polytechnic Institute IPN
Av. Santa Ana Num. 1000, Mexico City, 04430 Mexico
Email: parrao744@yahoo.com.mx, nakano-m@ice.uec.ac.jp

Brian M. Kurkoski
Department of Information and Communications Engineering,
University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585 Japan
Email: kurkoski@ice.uec.ac.jp

***Abstract*— This paper introduces the use of image normalization algorithm and SVD decomposition hash functions to generate a hash value for a digital image. Image normalization is a technique that has been shown to be robust to different kinds of geometric attacks such as rotation, scaling and flipping. Using the image normalization algorithm as a preprocessing step we can increase the robustness of the hash by 50% against rotation and scaling compared to without this algorithm.**

I. Introduction

During the information era, the interchange of digital content has had tremendous growth, but in the same way tools to manipulate digital content like Photoshop, Corel Draw and Illustrator, are becoming more popular and easier to handle, which leads an increase in the use of these tools for manipulating images for illegal purposes or to discredit people. To ensure trustworthiness, multimedia authentication techniques like watermarking and image hashing have emerged to verify content integrity and prevent forgery.

Traditionally, data integrity issues are addressed by message authentication functions or cryptographic hashes, which are key-dependent and bit sensitive. The integrity of the message can be validated only when every bit of the message is unchanged [2]. This sensitivity is usually applied to text message authentication. For digital images this sensitivity is not so straightforward because it should be able to tolerate some lossy representations with graceful degradation. Therefore bit-by-bit authentication is no longer suitable, and a tool that can authenticate content is more appropriate.

An image hash is an image-based content digital signature. To generate the hash, a secret key is used to extract some features from the content of the image. These features are processed to generate the hash. In addition to content authentication, multimedia hashes are used in content-based retrieval from databases [6]. To search for multimedia content, methods such as sample-by-sample comparisons are computationally inefficient. Moreover, these methods compare the lowest level of content representation and do not offer robustness in such situations as geometric distortions or different kinds of compression. Robust image hash functions can be used to address this problem [10].

There are two important criterias in the design of the image hash functions, robustness and security [10]. By robustness, we mean that if the same key is used, perceptually similar images generate a similar hash. The similarity of hashes is measured in terms of metrics like Hamming or Euclidean distances. The security of the image hash function is through a secret key to generate the hash. Without the knowledge of this key, the hash value should not be easy to estimated or forge.

The Singular Value Decomposition(SVD) is an important linear algebra tool, which is often used in image compression, digital watermarking and other signal processing fields. The SVD decomposes the image into three matrices, U, S and V. The matrix S, is a diagonal matrix containing the singular values of the image. The matrices U and V contain the singular vectors of the image, and by applying this decomposition we can reduce the dimensions of the data. The use of SVD decomposition in the image hashing problem was proposed by Kozat et al.l [5] where it uses SVD decomposition twice and has been shown to be robust to some small variation in rotation and scaling.

On the other hand, image normalization is an algorithm based on the invariant moments of the image. These moments are used to geometrically normalize the image in terms of scale, orientation, and flipping. This method is well-known in pattern recognition problems.

This paper introduces image normalization as a preprocessing step in the SVD decomposition hash function to increase robustness against rotation, scaling and JPEG compression. For increasing the security and introducing unpredictability in the hash value we randomly select sub-images, with a random partition algorithm [8], and apply the SVD decomposition hash function to each sub-image. After that, we rearrange the matrices U and V and again apply SVD decomposition to generate the intermediate hash value, and finally quantize and compress to generate the final binary hash value. The proposed algorithm is shown in Figure 1. As will be shown by numerical results, the addition of the normalization function substantially improves the robustness against rotation, scaling and compression.

This paper contains five sections. Section II describes the SVD decomposition, the image normalization algorithm and

Robust Image Hashing Using Image Normalization and SVD Decomposition

Ricardo Antoio Parrao Hernandez
and Mariko Nakano Miyatake
SEPI, ESIME Culhuacan
National Polytechnic Institute IPN
Av. Santa Ana Num. 1000, Mexico City, 04430 Mexico
Email: parrao744@yahoo.com.mx, nakano-m@ice.uec.ac.jp

Brian M. Kurkoski
Department of Information and Communications Engineering,
University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585 Japan
Email: kurkoski@ice.uec.ac.jp

***Abstract*— This paper introduces the use of image normalization algorithm and SVD decomposition hash functions to generate a hash value for a digital image. Image normalization is a technique that has been shown to be robust to different kinds of geometric attacks such as rotation, scaling and flipping. Using the image normalization algorithm as a preprocessing step we can increase the robustness of the hash by 50% against rotation and scaling compared to without this algorithm.**

I. Introduction

During the information era, the interchange of digital content has had tremendous growth, but in the same way tools to manipulate digital content like Photoshop, Corel Draw and Illustrator, are becoming more popular and easier to handle, which leads an increase in the use of these tools for manipulating images for illegal purposes or to discredit people. To ensure trustworthiness, multimedia authentication techniques like watermarking and image hashing have emerged to verify content integrity and prevent forgery.

Traditionally, data integrity issues are addressed by message authentication functions or cryptographic hashes, which are key-dependent and bit sensitive. The integrity of the message can be validated only when every bit of the message is unchanged [2]. This sensitivity is usually applied to text message authentication. For digital images this sensitivity is not so straightforward because it should be able to tolerate some lossy representations with graceful degradation. Therefore bit-by-bit authentication is no longer suitable, and a tool that can authenticate content is more appropriate.

An image hash is an image-based content digital signature. To generate the hash, a secret key is used to extract some features from the content of the image. These features are processed to generate the hash. In addition to content authentication, multimedia hashes are used in content-based retrieval from databases [6]. To search for multimedia content, methods such as sample-by-sample comparisons are computationally inefficient. Moreover, these methods compare the lowest level of content representation and do not offer robustness in such situations as geometric distortions or different kinds of compression. Robust image hash functions can be used to address this problem [10].

There are two important criterias in the design of the image hash functions, robustness and security [10]. By robustness, we mean that if the same key is used, perceptually similar images generate a similar hash. The similarity of hashes is measured in terms of metrics like Hamming or Euclidean distances. The security of the image hash function is through a secret key to generate the hash. Without the knowledge of this key, the hash value should not be easy to estimated or forge.

The Singular Value Decomposition(SVD) is an important linear algebra tool, which is often used in image compression, digital watermarking and other signal processing fields. The SVD decomposes the image into three matrices, U, S and V. The matrix S, is a diagonal matrix containing the singular values of the image. The matrices U and V contain the singular vectors of the image, and by applying this decomposition we can reduce the dimensions of the data. The use of SVD decomposition in the image hashing problem was proposed by Kozat et al.l [5] where it uses SVD decomposition twice and has been shown to be robust to some small variation in rotation and scaling.

On the other hand, image normalization is an algorithm based on the invariant moments of the image. These moments are used to geometrically normalize the image in terms of scale, orientation, and flipping. This method is well-known in pattern recognition problems.

This paper introduces image normalization as a preprocessing step in the SVD decomposition hash function to increase robustness against rotation, scaling and JPEG compression. For increasing the security and introducing unpredictability in the hash value we randomly select sub-images, with a random partition algorithm [8], and apply the SVD decomposition hash function to each sub-image. After that, we rearrange the matrices U and V and again apply SVD decomposition to generate the intermediate hash value, and finally quantize and compress to generate the final binary hash value. The proposed algorithm is shown in Figure 1. As will be shown by numerical results, the addition of the normalization function substantially improves the robustness against rotation, scaling and compression.

This paper contains five sections. Section II describes the SVD decomposition, the image normalization algorithm and

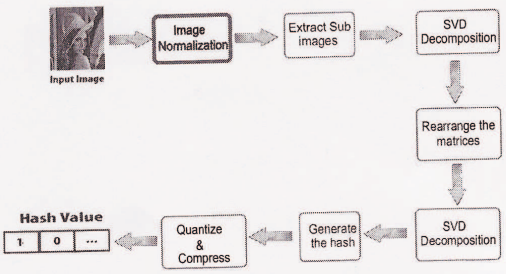


Fig. 1. Proposed algorithm for generate the hash value.

the random partition algorithm. Section III describes the proposed algorithm. Section IV present the results. Section V is the conclusion and the possible future work for this paper.

II. Definitions

In this section we introduce the theory behind the SVD decomposition, the characteristics of the image normalization algorithm and the random partition algorithm.

A. SVD Decomposition

We can see an image as a M-by-N matrix, so we can apply the SVD decomposition. We have an image A and apply the SVD decomposition as:

$$A = USV^T, \tag{1}$$

where U and V are orthogonal matrices that contain the left and right singular vectors and S is a diagonal matrix $S = \text{diag}(s_1, s_2, \ldots)$ with the singular values of A. We can express the decomposition as the following:

$$A = \sum_{i=1}^{r} U_i s_i V_i^T, \tag{2}$$

$$A = U_1 s_1 V_1^T + U_2 s_2 V_2^T + \ldots + U_r s_r V_r^T, \tag{3}$$

where r is the rank of A, U_i is a column i of U and V_i is a column of V.

The SVD decomposition for digital images has the following properties [4]:

1) The singular values change little when a small disturbance is applied.
2) The singular values represent the brightness features of the image, while the singular vectors represent the geometrical features.
3) The quality of the reconstructed image does not degenerate if we remove the small-valued singular values s_i.

B. Image normalization algorithm

Image normalization is an algorithm based on the geometric moments of the image. With this algorithm, an image F is robust to different kinds of affine transforms [3]. The algorithm is as follows:

Let F denote a digital image of size M-by-N. Its geometric moments m_{pq} and central moment μ_{pq},where $p, q = 0, 1, 2, \ldots$ are defined as:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q F(x, y). \tag{4}$$

And,

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q F(x, y), \tag{5}$$

where

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}}, \quad \bar{y} = \frac{m_{0,1}}{m_{0,0}}. \tag{6}$$

An image G is said to be an affine transform of F if there is a matrix $H = \left(\begin{smallmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{smallmatrix}\right)$ and a vector $d = (d_1 \; d_2)^T$ such that $G(x, y) = F(x_a, y_a)$, where

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = H \begin{pmatrix} x \\ y \end{pmatrix} - d. \tag{7}$$

The normalization procedure consists of the following steps for a given unnormalized image F:

1) Center the image F using the matrix $H = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and the vector $d = (d_1 \; d_2)^T$ with:

$$d_1 = \frac{m_{1,0}}{\mu_{0,0}}, \quad d_2 = \frac{m_{0,1}}{\mu_{0,0}}. \tag{8}$$

This step performs translation invariance. Let F_1 denote the resulting centered image.

2) Apply a shearing transform to F_1 in the x direction with matrix $H = \left(\begin{smallmatrix} 1 & \beta \\ 0 & 1 \end{smallmatrix}\right)$ so the resulting image is denoted by F_2. To determine β we use:

$$\mu_{30} + 3\beta\mu_{21} + 3\beta^2\mu_{12} + \beta^3\mu_{03} = 0. \tag{9}$$

We can have up to three roots, if one of the roots is real and the others are complex, we take the real one, if two or three roots are real we pick the median of the real roots, if all roots are complex β is 0, and under some very unusual conditions β will have infinite number of solutions if all the moments are zero. This happens when the image is rotationally symmetric [9].

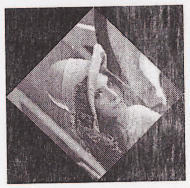
3) Apply a shearing transform to F_2 in the y direction with matrix $H = \left(\begin{smallmatrix} 1 & 0 \\ \lambda & 1 \end{smallmatrix}\right)$, the resulting image is denoted by F_3, where λ is defined with the equation:

$$\mu_{11} = \lambda\mu_{20} + \mu_{11}, \tag{10}$$

and setting $\mu_{11} = 0$ we obtain:

(a) Input: unmodified image

(b) Input: image rotate 45 degrees



(c) Output of image normalization algorithm

Fig. 2. Input and output of the image normalization algorithm

$$\lambda = \frac{\mu_{11}}{\mu_{20}}. \tag{11}$$

4) Finally, the image F_3 is resized to a specified size.

In Figure 2 we can see the result of the image normalization algorithm applied to the lena image. Figure 2(a) is the original image of Lena, Figure 2(b) is the Lena image rotated 45 degrees. Figure 2(c) is the output of the image normalization when either (a) or (b) is applied as an input. With this technique we observe that this algorithm is robust to scaling, rotation and flipping.

C. Random partition algorithm

For increasing the security of the hash value and adding unpredictability, we add some randomness by extracting sub-images. We form a number p pseudo-randomly selected sub-images with the random partition algorithm This consist of:

1) With a key k pseudo-randomly generate a set W where $W = ((x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p))$, which is the coordinate pair of the top-left corner of the sub-image $A_1, A_2, \ldots, A_p$.
2) For each coordinate pair generate a square of size m-by-m, and each square is the output that we call sub-image A_i.

III. Proposed algorithm

A robust image hash function has two inputs: an image I and a secret key k and has one output, a hash, which is a short binary vector $\vec{h} = H_k(I)$. The hash function should possess the following two perceptual properties:

1) Hash values for all perceptually identical images, that is when a person cannot find any difference in the content of the image, should be the same.
2) Perceptually different images, when two images do not have the same content in terms of visual perception, should have different hash values.

Next, we present the description of our proposed algorithm:

1) Let the M-by-N input image be $I \in [0, 255]^{M \times N}$. The number of sub-images is denoted by p and the size of this sub-image is denoted by m where $1 < m < M, N$, and the secret key for generating the hash is denoted by k. And the output $\vec{h}$ is the final hash value.
2) To I we apply the image normalization described in Section II-B to generate the normalized image I_1.
3) From I_1,with the use of a key k, pseudo-randomly form p possibly overlapping squares $A_1, A_2, \ldots, A_p$ each of size m-by-m as described in Section II-C.
4) To each rectangle A_i apply the SVD decomposition to obtain U, S, V, as is shown in equation(3).
5) From equation 3 we take the vector $U_1^{(i)}$ and $(V_1^{(i)})^T$ compose a new matrix B,

$$B = \left[U_1^{(1)}, \ldots U_1^{(p)}, (V_1^{(1)})^T \ldots (V_1^{(p)})^T\right], \tag{12}$$

with size m-by-$2p$.
6) We apply the SVD decomposition to B to get the matrices U', S' and V'.
7) The first column of U' and the first row of V'^T are the intermediate hash with length m+ $2p$. This is a vector of real numbers.
8) And finally, we quantize the resulting statistics vector and apply Gray coding to obtain the binary hash sequence. This binary sequence is then passed through the decoding stage of an order-3 Reed-Muller decoder for compression [7].
9) The output of the Reed-Muller decoder is binary hash $\vec{h}$.

IV. Results

To measure the performance of the proposed algorithm, we choose the Hamming distance between the binary hashes, normalized with respect to the length (L) of the binary hash as a performance metric. The normalized Hamming distance is defined as:

$$d_h(h_1, h_2) = \frac{1}{L} \sum_{k=1}^{L} \|h_1(k) - h_2(k)\|. \tag{13}$$

For doing the numerical evaluation we used different standard test images as an input and considered different kinds of attacks such as rotation, compression and scaling using the stirmark benchmark tool 4.0 [1]. These images are a 512-by-512 grayscale and also this is the size we use for the output in the image normalization algorithm. The number of sub-images is $p = 15$ and the subimage size is $m = 100$. We compare the algorithm with and without image normalization [5].

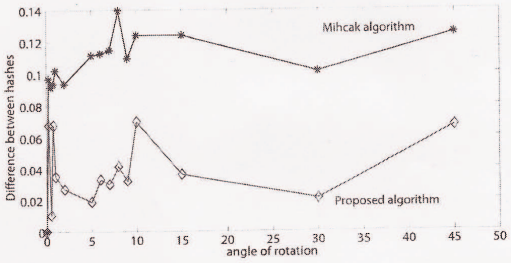


Fig. 3. Rotation attack, different angle of rotation vs distance between hashes.

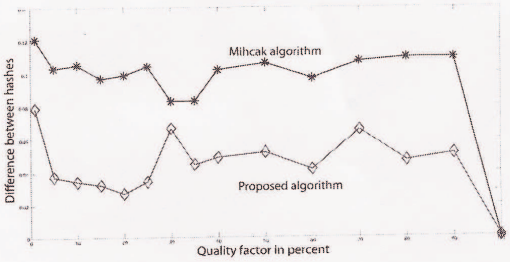


Fig. 5. JPEG compression attack, different quality factor in percent vs distance between hashes, where 100 percent is without compression.

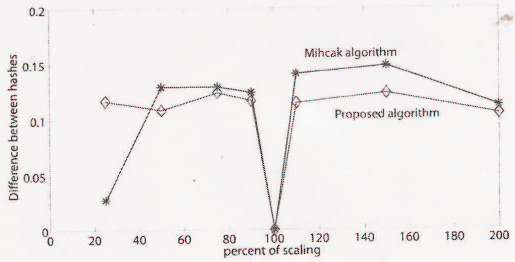


Fig. 4. Scaling attack, different percent of scaling vs distance between hashes, where 100 percent is the image in its original size.

For rotatation attack in Figure 3 we can observe after apply image normalization we can improve the performance in terms of Hamming distance we pass from a 0.14 to 0.07 that is an improvement of 50%.

For a scaling attack in Figure 4 we can observe the reduction of the Hamming distance is lower after applying image normalization, while Mihcak looks lower when we reduces the image, actually this effect occur when the image is reduce to 128-by-128 and all subimages are almost the same.

In the same way, for JPEG compression in Figure 5 we get better performance after applying the image normalization,and we can reduce the Hamming distance to under 0.1, and for doing authentication we need a threshold and we can say its under 0.1.

V. Conclusion

We present the use of image normalization as a preprocessing step to increase the robustness of the image hashing function based on SVD decomposition. As a result, there was significant improvement in terms of Hamming distance against geometric attacks such as rotation and scaling. For applications to a databases this characteristic is very useful because we can search based in the content of the image without taking care of the orientation or the size of the image.

We apply SVD decomposition twice because if we apply the SVD decomposition only once, then the hash is not robust or secure enough. But, if it is performed more than twice, experiments have shown that it would fail [8]. A good SVD-based image authentication method should use the SVD decomposition exactly twice.

As a future work. We need to explore different parameters for generating the sub-images to obtain a better content preservation, and also to try different codes for compressing the final hash value.

Acknowledgment

This research was sponsored by the UEC Japan through JUSST program, and CONACyT of Mexico.

References

[1] http://www.petitcolas.net/fabien/watermarking/stirmark/.

[2] A. Menezes, V. O., and Vanstone, S. *Handbook of Applied Cryptography*. CRC Pressr, 1998.

[3] Alghoniemy, M., and Tewfik, A. Geometric invariance in image watermarking. *Image Processing, IEEE Transactions on 13*, 2 (2004), 145 –153.

[4] Andrews, H., and Patterson, C. Singular value decompositions and digital image processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on 24*, 1 (Feb. 1976), 26 – 53.

[5] Kozat, S., Venkatesan, R., and Mihcak, M. Robust perceptual image hashing via matrix invariants. In *Image Processing, 2004. ICIP '04. 2004 International Conference on* (2004), vol. 5, pp. 3443 – 3446 Vol. 5.

[6] Lin, S., Özsu, M. T., Oria, V., and Ng, R. T. An extendible hash for multi-precision similarity querying of image databases. In *Proceedings of the 27th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 2001), VLDB '01, Morgan Kaufmann Publishers Inc., pp. 221–230.

[7] Mhak, M. K., and Venkatesan, R. A tool for robust audio information hiding: A perceptual audio hashing algorithm. In *in Proceedings of 4th Information Hiding Workshop* (2001), pp. 51–65.

[8] Monga, V., and Evans, B. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. *Image Processing, IEEE Transactions on 15*, 11 (Nov. 2006), 3452 –3465.

[9] Shen, D., and Ip, H. Generalized affine invariant image normalization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*, 5 (May 1997), 431 –440.

[10] Venkatesan, R., Koon, S.-M., Jakubowski, M., and Moulin, P. Robust image hashing. In *Image Processing, 2000. Proceedings. 2000 International Conference on* (Feb. 2000), vol. 3, pp. 664 –666 vol.3.

to increase robustness against geometric attacks. For increasing the security and introduce unpredictability in the hash value we randomlyselect sub-images, with a random partitioning algorithm [8], and apply the SVD decomposition hash function to each sub-image, after that we rearrange the matrices U and V and again apply SVD decomposition for generate the final hash value, in this case the hash value is a sequence of real numbers. The proposed algorithm is shown in Figure 1.

This paper contains five sections. Section1 introduces a brief description of why is important to have a hash based on the content of a digital image. Section 2 describes how to use SVD decomposition, the image normalization algorithm and the random partition algorithm, Section 3 describes the proposed algorithm, Section 4 present the results, Section 5 is the conclusion and the posible future work for this paper.

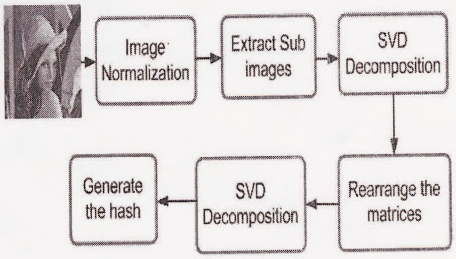


Figure 1: Proposed algorithm for generate the hash value

2 Definitions

In this section we introduce the theory behind the singular value decomposition (SVD), the characteristics of the image normalization algorithm and the random partition algorithm.

2.1 SVD Decomposition

We can see an image as a M-by-N matrix, so we can apply the SVD decomposition. We have an image A and apply the SVD decomposition as:

$$A = USV^T, \tag{1}$$

where U and V are orthogonal matrices that contain the left and right singular vectors and S is a diagonal matrix $S = \text{diag}(s_1, s_2, \ldots)$ with the singular values of A. We can express the decomposition as the following:

$$A = \sum_{i=0}^{r} U_i s_i V_i^T, \tag{2}$$

$$A = U_1 s_1 V_1^T + U_2 s_2 V_2^T + \ldots + U_r s_r V_r^T, \tag{3}$$

where r is the rank of A.

The SVD decomposition for digital images has the following properties [4]:

1. The singular values change little when a small disturbance is applied.

2. The singular values represent the brightness features of the image, while the singular vectors represent the geometrical features.

3. The quality of the reconstructed image does not degenerate if we remove the small-valued singular values s_i. This property we can be seen it in Figure 2 were we do not need all the singular values to generate an image with a good quality.

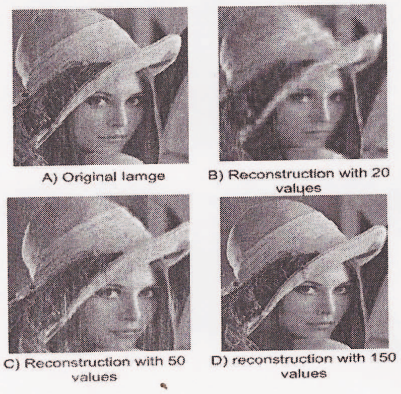


Figure 2: Use of different numbers of singulars values and vectors for reconstructing the original image

2.2 Image normalization algorithm

Image normalization is an algorithm based on the geometric moments of the image. With this algorithm, an image F is robust to different kinds of affine transforms [3]. The algorithm is as follows:

Let F denote a digital image of size M-by-N. Its geometric moments m_{pq} and central moment μ_{pq},where $p, q = 0, 1, 2, \ldots$ are defined as:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q F(x, y). \quad (4)$$

And,

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q F(x, y), \quad (5)$$

where

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}}, \quad \bar{y} = \frac{m_{0,1}}{m_{0,0}}. \quad (6)$$

An image G is said to be an affine transform of F if there is a matrix $H = \left(\begin{smallmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{smallmatrix}\right)$ and a vector $d = (d_1\ d_2)^T$ such that $G(x, y) = F(x_a, y_a)$, where

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = H \begin{pmatrix} x \\ y \end{pmatrix} - d. \quad (7)$$

The normalization procedure consists of the following steps for a given unnormalized image F:

1. Center the image F using the matrix $H = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ and the vector $d = (d_1\ d_2)^T$ with:

$$d_1 = \frac{m_{1,0}}{\mu_{0,0}}, \quad d_2 = \frac{m_{0,1}}{\mu_{0,0}}. \quad (8)$$

   This step performs translation invariance. Let F_1 denote the resulting centered image.

2. Apply a shearing transform to F_1 in the x direction with matrix $A = \left(\begin{smallmatrix} 1 & \beta \\ 0 & 1 \end{smallmatrix}\right)$ so the resulting image is denoted by F_2 To determine β we use:

$$\mu_{30} + 3\beta\mu_{21} + 3\beta^2\mu_{12} + \beta^3\mu_{03} = 0. \quad (9)$$

   We can have up to three roots, if one of the roots is real and the others are complex, we take the real one, if two or three roots are real we pick the median of the real roots, if all roots are complex β is 0, on some very unusual conditions β will have infinite number of solutions if all the moments are zero. This happens when the image is rotationally symetric [9].

3. Apply a shearing transform to F_2 in the y direction with matrix $H = \left(\begin{smallmatrix} 1 & 0 \\ \lambda & 1 \end{smallmatrix}\right)$, the resulting image is denoted by F_3, where λ is defined with the equation:

$$\mu_{11} = \lambda\mu_{20} + \mu_{11}, \quad (10)$$

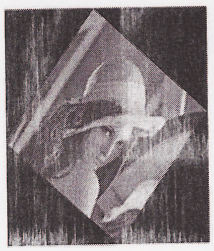
   and making $\mu_{11} = 0$ we obtain:

$$\lambda = \frac{\mu_{11}}{\mu_{20}}. \quad (11)$$

4. Finally, the image F_3 is resized to a specified size.

In Figure 3 we can see the result of the image normalization algorithm applied to the lena image. Figure 3(a) is the original image of lena, Figure 3(b) is the lena image rotated 45 degrees. Figure 3(c) is the output of the image normalization when either (a) or (b) is applied as an input. With this techniq we observe that this algorithm is robust to scaling, rotation and flipping.

A) Original Image B) Image Rotate 45 degrees



C) Output of the Image Normalization Algorithm

Figure 3: Input and output of the image normalization algorithm

2.3 Random partition algorithm

For increasing the security of the hash value and adding unpredictability, we add some randomless by generating sub-images. We form a number p pseudo-randomly selected sub-images with the random partition algorithm as shown in Figure 4. This consist of:

1. With a key k pseudo-randomly generate a set W where $W = ((x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p))$, which is the coordinate pair of the top-left corner of the sub-image $A_1, A_2, \ldots, A_p$.

2. For each coordinate pair generate a square of size m-by-m, and each square is the output that we call sub-image A_i.

3 The proposed algorithm

Robust image hash function has two inputs: an image I and a secret key k and has one output, a hash, which is a short vector $\vec{h} = H_k(I)$. The hash function should possess perceptual properties:

1. Hash values for all perceptually identical images, this means when a person cannot find any difference in the content of the image should have the same hash value.

2. Perceptually different images, when two images do not havet the same content in terms of visual perception, should produce different hash values.

Next, we present the description of our proposed algorithm:

1. Let the M-by-N input image be $I \in [0, 255]^{M \times N}$. The number of sub-images is denoted by p and the size of this sub-image is denoted by m where $1 < m < M, N$, and the secret key for generate the hash is denoted by k. And the output $\vec{h}$ is the final hash value.

2. To I we apply the image normalization described in Section 2.2 to generate the normalized image I_1.

3. From I_1,with the use of a key k, pseudo-randomly form p possibly overlapping squares $A_1, A_2, \ldots, A_p$ each of size m-by-m as described in Section 2.3.

4. To each rectangle A_i apply the SVD decomposition to get $\{U_1S_1V_1, U_2S_2V_2, \ldots, U_pS_pV_p\}$ where matrices U_i and V_i are the singular vectors that have the content information of each sub-image and S_i are the singular values.

5. The first column of each U_i and the first row of each V_i^T compose a new matrix B.,

$$B = \left[U_1, \ldots U_p, V_1^T \ldots V_i^T\right], \tag{12}$$

with size m-by-$2p$.

6. We apply the SVD decomposition to B to get the matrices U', S' and V'.

7. And finally, the first column of U' and the first row of V'^T are the image hash with length $m+ 2p$

8. output hash $\vec{h}$, $\vec{h} \in \mathbb{R}^{m+2p}$.

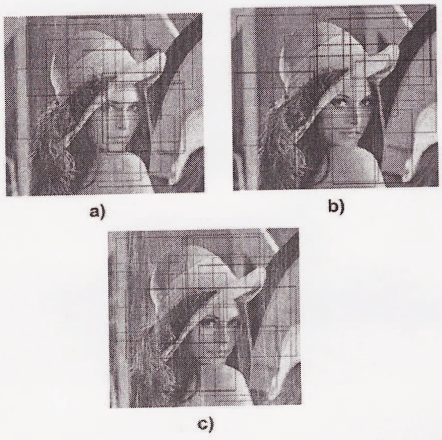


Figure 4: Examples of generating 15 sub-images of Lena, each image is partitioned with different key (a) key 1, (b) key 2 and (c) key 3

4 Results

To evaluate the performance of the algorithm we compare it with an algorithm without image normalization [6]. We apply different standard test images as an input and consider different kind of attacks such as rotation, compression, scaling and salt and pepper noise generating by the stirmark benchmark tool 4.0 [1]. These images are a 512-by-512 grayscale and also this is the size we use for the output in the image normalization algorithm. The parameters for the number of sub-images $p = 100$ and size $m = 50$.

For measure the difference d between two hashes $\vec{h}_1$ and $\vec{h}_2$ in the L_2 norm sense we use:

$$d = \left\| \frac{\vec{h}_1 - \vec{h}_2}{2 * \sqrt{\left\|\vec{h}_1\right\| \left\|\vec{h}_2\right\|}} \right\|. \tag{13}$$

We show in Figure 5 different hashes of length 160 generate with the proposed algorithm and with the one without image normalization. We can observe adding the image normalization algorithm does not change the characteristics of the hash.

In Figure 6 we can see the difference of the lena image attack with rotation, the stars represent the difference of the algorithm without image normalization and the diamond with it. As we can observe, the image normalization increases the robustness against rotation attacks, if we increase the angle of rotation

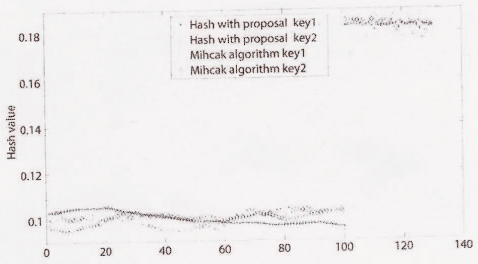


Figure 5: Hash values generated with and without image normalization and different keys.

the algorithm without image normalization increases the distance between hashes. Instead of the algorithm with image normalization, where we can observe, if we increase the angle of rotation the distance is very close to 0.

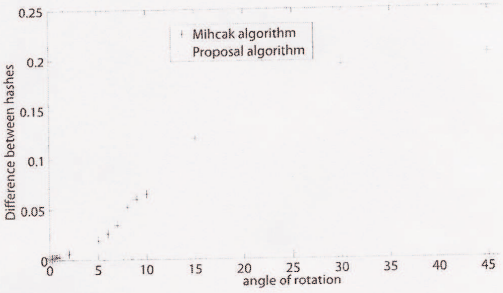


Figure 6: Rotation attack, different angle of rotation vs distance between hashes.

The same way in Figure 7, we can see the distance between different scales of the image, so if we modify size of the image, the distance is very close to zero with image normalization algorithm as compared to algorithm without image normalization where if the image change its size the distance increase .

In the case of JPEG compression we can see in Figure 8 that even the algorithm without normalization has a good performance because the distance between hashes is lower in image with visually without degradation but the use of the image normalization algorithm reduce the distance between hashes.

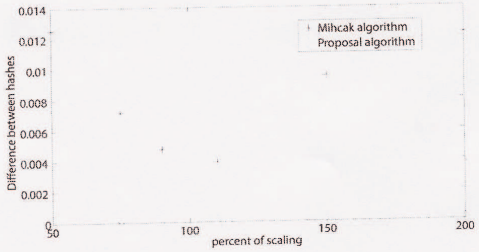


Figure 7: Scaling attack, different percent of scaling vs distance between hashes, where 100 percetn is the image in its original size.

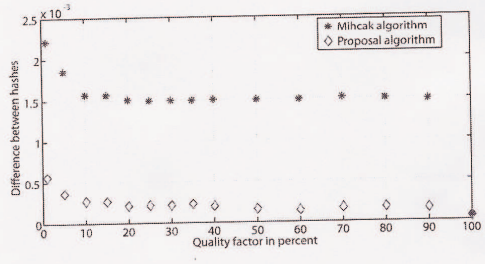


Figure 8: JPEG compression attack, different quality factor in percent vs distance between hashes, where 100 percent is without compression.

5 Conclusions

We present the use of image normalization to increase the robustness of the image hashing function based in Singular Value Decomposition. In the algorithm we apply SVD decomposition twice because if we apply the SVD decomposition only once, then the hash is not robust and secure enough. But, if it is performed more than twice, experiments have shown that it would fail [8]. A good SVD-based image authentication method should use the SVD decomposition exactly twice.

Using Image normalization also we can achieve better performance to different attacks such as rotation, scaling and compression. For applications to a databases this characteristic is very useful because we can search based in the content of the image without taking care of the orientation or the size of the image.

We see the image normalization algorithm as a solution for geometric attacks, and this technique can be applied to different content-based image hashing methods

to improve the performance, using it as a preprocessing step in the problem of the image hashing.

Acknowledgements: This research was sponsored by the UEC Japan through the JUSST program, and the CONACyT of Mexico.

References

[1] stirmark benchmark, Jannuary 1999.

[2] A. Menezes, V. O., and Vanstone, S. *Handbook of Applied Cryptography*. CRC Pressr, 1998.

[3] Alghoniemy, M., and Tewfik, A. Geometric invariance in image watermarking. *Image Processing, IEEE Transactions on 13*, 2 (2004), 145 –153.

[4] Andrews, H., and Patterson, C. Singular value decompositions and digital image processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on 24*, 1 (Feb. 1976), 26 – 53.

[5] Cannons, J., and Moulin, P. Design and statistical analysis of a hash-aided image watermarking system. *Image Processing, IEEE Transactions on 13*, 10 (2004), 1393 –1408.

[6] Kozat, S., Venkatesan, R., and Mihcak, M. Robust perceptual image hashing via matrix invariants. In *Image Processing, 2004. ICIP '04. 2004 International Conference on* (2004), vol. 5, pp. 3443 – 3446 Vol. 5.

[7] Lin, S., Özsu, M. T., Oria, V., and Ng, R. T. An extendible hash for multi-precision similarity querying of image databases. In *Proceedings of the 27th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 2001), VLDB '01, Morgan Kaufmann Publishers Inc., pp. 221–230.

[8] Monga, V., and Evans, B. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. *Image Processing, IEEE Transactions on 15*, 11 (Nov. 2006), 3452 –3465.

[9] Shen, D., and Ip, H. Generalized affine invariant image normalization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*, 5 (May 1997), 431 –440.

[10] Venkatesan, R., Koon, S.-M., Jakubowski, M., and Moulin, P. Robust image hashing. In *Image Processing, 2000. Proceedings. 2000 International Conference on* (Feb. 2000), vol. 3, pp. 664 –666 vol.3.