Identificación y comparación de imágenes en ambientes forenses

Bruno Constanzo, Martín Castellote, Santiago Trigo, Ana Haydée Di Iorio

Laboratorio de Investigación y Desarrollo de Tecnología en Informática Forense InFo-Lab
Universidad FASTA, Ministerio Público de la Provincia de Buenos Aires,
Municipio de General Pueyrredon
{bconstanzo, mcastellote, santiagotrigo, diana}@ufasta.edu.ar

Resumen. Las investigaciones digitales de la actualidad se enfrentan a una gran cantidad de contenido multimedia, que pone a los peritos e investigadores en la necesidad de analizar, buscar y detectar contenido específico en conjuntos de cientos de miles o millones de imágenes y videos. En particular, en las investigaciones de abuso sexual de menores esta problemática se ve ampliada por la escala de los casos, y por el fuerte impacto psicológico que tiene el mismo sobre los peritos e investigadores. Es de vital importancia implementar nuevas soluciones que tengan la capacidad de procesar el contenido en base a sus características visuales, y a partir de ellos poder hacer análisis de similitud, detección, y filtrado automático. En este trabajo se presentan 3 hashes perceptuales implementados como parte de una librería orientada al análisis forense de imágenes y video. Se explican los algoritmos, su implementación, se analiza su rendimiento, y se presentan casos de uso. Finalmente, se evalúa cómo se pueden utilizar en ámbitos de la justicia (tanto pericial como investigación), y se proponen alternativas para continuar el desarrollo.

Keywords: hashes perceptuales - procesamiento de imágenes - informática forense - investigaciones digitales

1 Introducción

La sociedad moderna ha experimentado avances tecnológicos vertiginosos y de gran impacto social en los últimos 10 años. La aparición del *smartphone* puso en los bolsillos de gran parte de la población un instrumento multimedia de enormes capacidades, tanto para generar contenido multimedia, como para compartirlo rápidamente con otras personas. Esta explosión multimedia impacta directamente sobre la tarea de los peritos informáticos y los investigadores judiciales, que en su actuación diaria deben enfrentarse ya no a miles, sino a cientos de miles, o incluso millones de imágenes y videos[1, 2].

En los casos de abuso sexual infantil, la problemática se ve agravada por el impacto psicológico que tienen las imágenes sobre los expertos que deben trabajar en estos casos[3]. Ya no es sólo un problema de escala o rapidez con la que se puedan analizar los

© Springer-Verlag Berlin Heidelberg 2011

casos, sino que se convierte en una cuestión de salud psicológica y emocional de los investigadores.

Una de las herramientas que se puede plantear para enfrentar este problema es la utilización de hashes perceptuales, especializados para el caso de imágenes y fotografías digitales. Estos tipos de hashes tienen la ventaja frente a los hashes tradicionales basados en algoritmos que se computan sobre los bytes de cada archivo: al estar basados en los patrones visuales de la imagen, son resistentes a cambios en el archivo que no modifique la misma. Es decir, si una misma imagen se guarda como JPG, PNG, BMP o TIF, los hashes perceptuales mantendrán el mismo valor (o un suficiente nivel de similitud que posibilite detectar el contenido), independientemente del formato de archivo en que se guarde, a diferencia de los hashes MD5 y SHA-1 que usualmente se utilizan en las pericias informáticas.

En este trabajo se presentan los conceptos teóricos básicos de hashes e imágenes digitales que son necesarios para comprender estas técnicas, y luego se exponen tres algoritmos de hashes perceptuales que fueron implementados por los autores. Luego se muestran resultados de las pruebas realizadas utilizando estos hashes, y ejemplos de situaciones que se pueden resolver por medio de ellos. Finalmente, se llega a conclusiones sobre su utilización, y se plantea trabajo y aplicaciones futuras que podrían aportar a mejorar esta técnica, y el trabajo de los expertos en informática forense del país.

2 Marco teórico

2.1 Hashes: funciones y digestos

Las funciones de hash son funciones matemáticas o procedimientos que transforman un mensaje de entrada m en un digesto (de tamaño fijo) d, tal que $d = f(m)$. Dependiendo de la función de hash utilizada, y su objetivo, variará la complejidad en el cálculo del digesto, y se tendrán distintas garantías sobre el resultado.

Se las puede clasificar en base a su objetivo de la siguiente manera:

- **Funciones de índice** para tablas de hashes, que permiten ordenar información dentro de una estructura de datos de acceso aleatorio y, en base a una clave o identificador, rápidamente acceder a un contenido. Este tipo de funciones se utilizan para bases de datos, estructuras de dato de acceso rápido, o caches, entre otros.
- **Funciones de *checksum*** que tienen la capacidad de detectar cambios en un archivo que ha viajado por un medio de transmisión, de manera que se pueda verificar su integridad. Dado que están orientadas a detectar errores de transmisión, su enfoque no es tan riguroso como los hashes criptográficos.
- **Hashes criptográficos** orientados a utilización en criptografía, para asegurar comunicaciones, firmas digitales, verificación de passwords, entre otras. Las funciones de hash criptográficas idealmente cumplen con las siguientes propiedades:
  - Determinística: el cálculo repetido de d en base al mismo mensaje m siempre debe dar el mismo resultado.
  - Irreversible: no se puede inferir el mensaje m en base a d.
  - Avalancha: un cambio pequeño en m ocasiona grandes cambios en d.

  - Baja probabilidad de colisión, entendiendo como colisión al hecho que, para dos mensajes m_1 y m_2 distintos, $f(m_1) = f(m_2)$.
  - Debe ser de rápido cálculo.
- Hashes perceptuales orientados a identificar y detectar similitudes entre elementos multimedia de fuertes características sensoriales (visuales o auditivas).

Para la informática forense, son de particular interés los hashes criptográficos. Las funciones MD5 y SHA-1[4, 5] se utilizan para complementar la cadena de custodia y garantizar la validez e integridad[1] de las fuentes de evidencia digital[6 Cap. 5].

Pese a sus características para identificar y verificar archivos, tienen una problemática al enfrentarse a cambios en su estructura. Si se cambia el tipo de archivo, por ejemplo, de JPG a PNG, la imagen visualmente seguirá siendo la misma, pero los hashes criptográficos correspondientes no coincidirían entre el archivo JPG y el PNG. Lo mismo sucedería si el archivo contiene metadatos y estos fueran eliminados o modificados.

Antes de analizar en mayor detalle los hashes perceptuales que son objeto de este trabajo, es necesario introducir algunos conceptos de imágenes digitales necesarios para su presentación y análisis.

2.2 Imágenes digitales

Una imagen digital I se puede definir como una función $f(x, y)$, siendo x e y las coordenadas espaciales, y la amplitud de f en cualquier punto es la intensidad o nivel de gris de la imagen. Cuando f, x e y son todas cantidades discretas y finitas, entonces se trata de una imagen digital[7]. Es posible armar una matriz o arreglo de intensidades, de manera que:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(1,0) & \dots \\ f(0,1) & f(1,1) & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

A cada elemento de esta matriz se lo conoce como pixel, y se suelen limitar los valores que almacenan a un entero sin signo de 8 bits, que da un rango de posibles valores de entre 0 y 255[2]. Los valores menores que 0, o mayores que 255, son "saturados" y no se puede representar información por debajo o por encima de ellos, un fenómeno conocido como *clipping*.

Una imagen en escala de grises tendrá una única matriz de intensidades, que indicarán el nivel de intensidad de la imagen en cada punto. Si se trata de una imagen a color, la representación más usual consiste en utilizar tres matrices de intensidad, correspondientes a los colores rojo, verde y azul, o RGB (del inglés *red green blue*). A cada una de estas matrices de intensidad se las suele llamar "canal". Otros espacios de color

---

[1] Los posibles ataques criptográficos a estas funciones son conocidos por los autores. La discusión sobre utilizar otros tipos de hashes, cómo enfrentar la incertidumbre que plantean, y la probabilidad de encontrar un ataque real en entornos forenses, son objeto de otro trabajo.

[2] Otras representaciones utilizan enteros de más de 8 bits, o flotantes de 32 o 64 bits. Para la discusión de este trabajo, es indistinto.

representan la imagen digital en función de otras características, por ejemplo, HSL tiene tres canales dedicados al tono (*hue*), saturación (*saturation*) y nivel de brillo (*levels*).

De aquí en adelante en el trabajo se asumirá que las imágenes son matrices RGB con 8 bits de precisión por canal, ya que es una de las representaciones más utilizadas por los formatos de imagen, entre ellos JPG y PNG[8, 9]. Se deja de lado la cuestión de la codificación y compresión, porque esta, si bien puede traer ligeras modificaciones a la imagen en el caso de compresión con pérdida, no altera el hecho de representar la imagen como una matriz de intensidades.

Operaciones sobre imágenes.

Las representaciones como arreglo de valores o como matriz de una imagen permite aplicar operaciones matemáticas sobre la misma. Estas operaciones buscan realizar algún cambio sobre en la imagen que permita realizar un análisis o interpretación de la información que representa.

Las operaciones de imagen que se utilizarán en este trabajo son:

- Conversión RGB a escala de grises: cada pixel RGB de la imagen I se reemplaza por un único valor, $y = f(r, g, b)$, aplicando alguna función de transformación. Una de las más comunes es:

$$y = 0.299r + 0.587g + 0.114b \text{ [10]} \quad (1)$$

  Esta operación reduce la cantidad de canales de color de 3 a 1.
- Recorte (crop): el recorte de una imagen I en (x_0, y_0) con ancho w y alto h, $crop(I, x0, y0, w, h)$ genera una submatriz de imagen I' tal que su esquina superior izquierda es $I(x_0, y_0)$ y su esquina inferior derecha es $I(x_0 + w, y_0 + h)$.
- Curva de niveles: se procesa cada pixel p_{xy} de I de acuerdo a una curva que describe para cada valor [0..255] un valor correspondiente [0..255] que transforma al pixel. Esta operación permite cambiar el contraste de la imagen. Si la curva aplicada es biyectiva, la operación puede ser invertida, en caso contrario, será una operación que causará pérdida de información.
- Redimensionado (*downsample*): se transforma la matriz I de dimensiones (w, h) siendo w el ancho en pixels y h el alto en pixels, en una matriz I' de dimensiones (w', h') donde $(w' < w, h' < h)$. Cada pixel p'_{xy} tiene un valor interpolado en base a varios pixels p_{xy} originales. Esta operación reduce la resolución espacial de la imagen.
- Transformaciones geométricas: el producto matricial entre la matriz de intensidades A y una matriz M de 3x3 permite aplicar transformaciones afines u homografías a una imagen. Dependiendo de cómo se forme la matriz M, ésta puede aplicar múltiples transformaciones sobre la imagen en una sola operación, incluyendo traslaciones, rotaciones, sesgos, o cambios de perspectiva.
- Transformaciones de dominio: las operaciones que se mencionaron hasta ahora operan todas en el dominio espacial. Algunas operaciones de imagen pueden representarse mejor en un dominio transformado. En este trabajo es de interés la

Transformación Discreta del Coseno, o DCT por sus siglas en inglés, que permite obtener coeficientes que describen la estructura general de la imagen. Su fórmula es:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \quad k = 0, \dots, N-1$$

2.3 Hashes perceptuales

Un hash perceptual es aquel que se basa en características sensoriales de un archivo para realizar el cálculo. Para el caso de archivos de imagen, el objeto de interés de este trabajo, los algoritmos de hashes perceptuales se enfocan en los patrones visuales que se generan al renderizar una imagen.

En su estudio de hashes perceptuales, Zauner define los siguientes conceptos de interés[11]:

- **Modificación:** es una operación sobre el objeto de medios que **no afecta** sus características esenciales.
- **Manipulación:** es una operación sobre el objeto de medios que **sí afecta** sus características esenciales.

Los algoritmos de hashes perceptuales deben ser resistentes a las modificaciones, y no confundir las manipulaciones sobre un contenido. Es decir, si se compara la similitud de una imagen con otras versiones de la misma, que han sufrido modificaciones, debe identificarlas como la misma imagen, ya que no han cambiado sus características esenciales. Al contrario, si ha habido manipulación de la imagen, el hash perceptual debe identificarlas como imágenes distintas.

El resultado de los hashes perceptuales que se presentan en este trabajo es una secuencia de 64 bits, organizados en memoria como un arreglo de 8x8. La comparación entre dos hashes se realiza aplicando la distancia de Hamming[12] entre ellos, y definiendo un umbral de similitud. Este umbral es dependiente del tipo de hash que se haya aplicado. Un umbral de 0 exige al algoritmo que el hash de las imágenes debe ser idéntico. Un umbral más alto, permitirá detectar modificaciones sobre una imagen, pero descartar manipulaciones.

Como parte de este trabajo, se han implementado tres algoritmos de hashes perceptuales de imagen: Average Hash (*a_hash*), Difference Hash (*d_hash*) y DCT-Hash (*p_hash*)[3]. Se presentarán a continuación, con una breve discusión sobre sus ventajas y desventajas.

Average Hash.

[3] Para los nombres de los hashes se ha adoptado la convención de Zauner, pero también se brinda entre paréntesis el nombre popular con el que algunas librerías implementan estos hashes (ver [13-15])

Este hash perceptual es conceptualmente muy simple y fácil de calcular. Como modelo conceptual para entender otros hashes es interesante, pero no es tan robusto como las otras alternativas. El algoritmo para calcularlo es el siguiente:

- Convertir la imagen a escala de grises
- Redimensionar la imagen a 8x8. Se llamará a esta imagen, I'.
- Calcular el valor promedio de los 64 pixels, avg.
- Para cada pixel p'_{xy} de I', el valor de hash H en la posición (x, y) será $h_{xy} = 1$ si $p_{xy} > avg$, o $h_{xy} = 0$ en caso contrario.

De esta manera, el hash resultante describe la estructura macro de la imagen. Cualquier alteración sutil, o que afecte detalles finos, no generará cambios lo suficientemente grandes en el pixel p_{xy} afectado para impactar en el valor h_{xy} final. Si en cambio fueran alteraciones mayores, entonces se modificará tanto el valor de p_{xy}, como el promedio de la imagen. Un cambio de esa magnitud afectaría globalmente al hash H, modificando varios de sus elementos h_{xy}.

La mayor problemática de este algoritmo es que, al basarse en el valor promedio de los pixels, las modificaciones sobre el contraste de la imagen afectan al promedio de intensidad y generan grandes alteraciones en el hash, sin que haya cambiado la esencia de la imagen. Esto es particularmente notable si se realizan cambios sobre el canal verde, que como indica la ecuación (1), tiene una fuerte influencia sobre el nivel de gris calculado. Por estas razones, se concluye que este no es un hash robusto.

Difference Hash.

Este hash se basa en los gradientes horizontales de la imagen. Su cálculo es relativamente simple, pero el concepto en el que se basa es un mejor descriptor de la estructura macroscópica de una imagen que el utilizado por el Average Hash. El algoritmo para calcular este hash es el siguiente:

- Convertir la imagen a escala de grises.
- Redimensionar la imagen a 9x8. Se llamará a esta imagen, I'.
- Se toman dos subregiones de I' de tamaño 8x8, α y β, siendo α la subregión de más a la izquierda de I', y β la subregión de más a la derecha.
- El hash final H se construye como un arreglo de 8x8, donde valor de hash en la posición (x, y) se construye de la siguiente manera: $h_{xy} = 1$ si $\alpha_{xy} < \beta_{xy}$, $h_{xy} = 0$ en caso contrario.

Esta construcción aproxima el gradiente horizontal de la imagen escalada I'. Esto permite que el hash se construya en base a los bordes de imagen, que son más resistentes a las alteraciones que resultan problemáticas para el Average Hash. Otra ventaja de este algoritmo es que es muy simple de implementar, y su velocidad de cálculo es elevada ya que sólo hay que hacer comparaciones entre valores adyacentes en memoria.

No tiene ninguna desventaja en sí, pero tiene debilidades comunes a los tres algoritmos que se mencionan más adelante.

DCT-Hash.

Este hash se basa en coeficientes DCT calculados sobre la imagen. Si bien el cálculo es más exigente que las alternativas vistas hasta ahora, la implementación del algoritmo lograda no presenta grandes diferencias de velocidad con respecto a los anteriores. El algoritmo para calcular este hash es el siguiente:

- Convertir la imagen a escala de grises.
- Redimensionar la imagen a 32x32. Se llamará a esta imagen I'.
- Aplicar la transformada DCT sobre I'.
- De $DCT(I')$, quedarse con el segmento superior izquierdo de tamaño 8x8. Se llamará a esta imagen I''.
- Calcular la mediana de los valores en I'', avg.
- Para cada pixel ${p''}_{xy}$ de I'', el valor de hash final H en la posición (x, y) se construye de la siguiente manera: $h_{xy} = 1$ si ${p''}_{xy} > avg$, $h_{xy} = 0$ en caso contrario.

La DCT es una transformación relacionada con la transformada de Fourier que permite expresar señales de manera compacta. Al igual que la transformada de Fourier, transforma la imagen del dominio espacial al dominio de la frecuencia. La DCT tiene una gran capacidad de almacenar información en relación al bajo costo computacional que exige su cálculo (comparado con otras transformaciones al dominio de la frecuencia). Por esta razón, se utiliza para la compresión con pérdida en JPG. Esta misma capacidad se aprovecha en el cálculo de este hash para representar la estructura general de la imagen, con pocos coeficientes que en última instancia son transformados a los bits del hash final.

Al igual que el Difference Hash, no tiene ninguna desventaja propia, más allá de las debilidades comunes que tienen estos tres algoritmos. Durante algún tiempo se discutía que el cálculo de la DCT resultaba en un impacto notable en la velocidad de cálculo, sin embargo, se verá que en la implementación realizada por los autores todos los algoritmos son equivalentes en velocidad de ejecución.

3 Implementación y resultados

La implementación realizada de los algoritmos se encuentra disponible en un repositorio de GitHub de los autores[16]. Para la misma, se utilizó el lenguaje de programación Python, utilizando las librerías Numpy[17], SciPy[18] y OpenCV[19]. A modo ilustrativo, se incluye el código final de la implementación del Difference Hash:

```
def d_hash(source):
    y = cv2.cvtColor(source, cv2.COLOR_BGR2GRAY)
    y = cv2.resize(y, (9, 8))
    hash_ = y[:, 0:8] < y[:, 1: 9]
    return hash_.astype(np.uint8)
```

Con la forma en que se ha elegido representar los hashes, el cálculo de la distancia de Hamming se simplifica a una función que calcula el bitwise-XOR entre dos hashes

perceptuales, y luego la sumatoria de elementos sobre el arreglo resultante. Dicha función puede expresarse en Python de la siguiente manera:

```
def hamming(h1, h2):
    return np.sum(h1 ^ h2)
```

De esta forma, si los elementos de hash coinciden, contribuyen con 0 a la sumatoria, y si son distintos contribuyen con 1, siendo esta la definición de la distancia de Hamming.

Como experimento, se realizó una búsqueda sobre un volcado de WhatsApp compuesto por 8628 imágenes. La búsqueda estaba orientada a encontrar imágenes visualmente similares a alguna de las presentes en una base de 40 imágenes de interés. En repetidas pruebas, se determinó de manera experimental los umbrales adecuados que permiten detectar modificaciones sobre las imágenes base para cada algoritmo de hash perceptual utilizado.

Hash	Umbral	Dectadas (Falsos positivos)	Sin detectar
Average	3	6 (2)	7
Difference	6	10 (1)	3
DCT	12	7 (0)	4

Tabla 1. Resultados de búsqueda con hashes perceptuales en el conjunto de prueba.

Algunas consideraciones para tener en cuenta:

- Un total de 11 imágenes que deberían ser encontradas. Estas imágenes son modificaciones de 6 imágenes base del conjunto de interés.
- Tanto Average Hash como Difference Hash generaron falsos positivos. De haber bajado más su umbral de detección, se habrían disminuído los falsos positivos generados, pero se habrían dejado de detectar imágenes de interés.
- En el caso de DCT-Hash el umbral parece relativamente alto (en comparación con los otros), también está en un delicado equilibrio. De aumentarlo, se comienzan a generar falsos positivos sin lograr encontrar las 3 imágenes faltantes del conjunto de interés. En caso de disminuirlo, se dejarían de detectar algunas de las imágenes que generan un match.
- Una de las imágenes de interés no es detectada a menos que los umbrales de detección se eleven por encima del nivel que en generan una alta tasa de falsos positivos.

Con respecto a la velocidad de cálculo de las funciones, no se encontraron mayores diferencias entre una y otra. Tomando como ejemplos una imagen pequeña de 160x160, una imagen mediana de 2400x1400 y una imagen grande de 4608x3456, del cálculo repetido de cada función de hash se obtuvieron los siguientes tiempos:

Hash	Pequeña	Mediana	Grande
Average	72.8 µs ± 4.41 µs	4.68 ms ± 106 µs	25.9 ms ± 2.35 ms
Difference	103 µs ± 2.8 µs	11.7 ms ± 274 µs	25.6 ms ± 3.22 ms
DCT	214 µs ± 42.9 µs	12.9 ms ± 1.89 ms	24 ms ± 2.08 ms

Tabla 2. Tiempos de cálculo de las funciones de hash perceptual sobre una imagen pequeña, mediana y grande. Obtenido con el comando %timeit de IPython.

Las mediciones se realizaron en Python 3.6.3 de 64 bits, bajo Windows 10, y las librerías Numpy, SciPy y OpenCV compiladas con la Intel Math Kernel Library, en un equipo de pruebas con un procesador Intel Core i5 7200U y 8GB de memoria RAM, utilizando el comando %timeit de IPython[20].

Analizando el rendimiento en cada caso, se concluye que la operación que más tiempo lleva en cada una de las funciones de hash es el redimensionado.

Es notable que, pese a lo que indica la literatura previa[13, 14], no se ha observado una diferencia notable en el rendimiento del DCT-Hash respecto al Average Hash o Difference Hash una vez superado un cierto umbral de tamaño (ver columna "Grande" en la Tabla 2). Para imágenes de tamaño medio, el Average Hash presenta una cierta ventaja con respecto a los otros dos, pero su rendimiento es pobre para detectar variaciones de una imagen base.

4 Conclusiones y trabajo futuro

La experiencia de implementación de los algoritmos y de las pruebas realizadas se ha llegado a las siguientes conclusiones:

- La utilización de hashes perceptuales permite realizar un análisis de alta velocidad en busca de imágenes específicas. Esta búsqueda es resistente a modificaciones de imagen, con distinto nivel de robustez dependiendo tanto del algoritmo utilizado, como del umbral de distancia que se decida utilizar.
- Los umbrales de detección deben elegirse teniendo en cuenta el delicado equilibrio que se debe alcanzar si se pretende buscar variantes de las imágenes que hayan sufrido modificaciones. Al contrario, si solamente se buscan matches totales, puede utilizarse un umbral de 0.
- Este trabajo se enfocó en las imágenes en RGB con 8 bits de color por canal. Sería útil ampliar la problemática para considerar las imágenes almacenadas en RGBA, y analizar las posibilidades y problemáticas que se abren en este caso.
- ImageHash tiene una variante del Difference Hash que calcula los gradientes verticales. Es posible implementar esta variante en el desarrollo realizado, para agregar una herramienta más que permita identificar de manera unívoca una imagen en base a sus hashes perceptuales.
- Los hashes se han planteado con un tamaño fijo en las dimensiones de imagen que, en todos los casos, resulta en 64 bits, que pueden almacenarse de manera compacta en un entero largo. Es posible extender los hashes para que las imágenes tengan otros tamaños, que resulten en hashes de más bits, por ejemplo, imágenes intermedias de 16x16 resultarían en hashes de 256 bits.
- La utilización de hashes perceptuales abre la puerta a generar bases de datos propias de las instituciones nacionales, que permitan reconocer contenido multimedia ya identificado en investigaciones realizadas por las fuerzas de seguridad y los investigadores judiciales. Para esto, es necesario continuar el trabajo de estudio de estas

técnicas, definir los alcances y las posibilidades que brindan, y también las estrategias de mitigación para los falsos positivos. Ese trabajo permitirá implementar bases de datos de contenido sensible basadas en los hashes perceptuales, complementaria de las bases de datos de hashes criptográficos que hoy se utilizan.

Analizando el rendimiento en cada caso, se concluye que la operación que más tiempo lleva en cada una de las funciones de hash es el redimensionado.

Es notable que, pese a lo que indica la literatura previa[13, 14], no se ha observado una diferencia notable en el rendimiento del DCT-Hash respecto al Average Hash o Difference Hash una vez superado un cierto umbral de tamaño (ver columna "Grande" en la Tabla 2). Para imágenes de tamaño medio, el Average Hash presenta una cierta ventaja con respecto a los otros dos, pero su rendimiento es pobre para detectar variaciones de una imagen base.

5 Agradecimientos

Los autores desean agradecer a la Universidad FASTA, al Ministerio Público de la Provincia de Buenos Aires y al Municipio de General Pueyrredon por brindar un espacio único de trabajo como es el InFo-Lab.

En especial, desean agradecer a Ayrton Betti y Axel Ziegler, estudiantes que realizaron sus práctica profesionales en el Laboratorio participando de este proyecto, a Mónica Pascual, secretaria de investigación de la Facultad de Ingeniería de la Universidad FASTA, y a Roberto Giordano Lerena, decano de la Facultad de Ingeniería.

Referencias

1. Ministerio Público Fiscal de la Ciudad Autónoma de Buenos Aires: Operación Luz de Infancia: golpe internacional contra la pornografía infantil. (Noticia). Disponible online en: https://www.fiscalias.gob.ar/project/operacion-luz-de-infancia-golpe-internacional-contra-la-pornografia-infantil/.
2. Diario Clarín: Mega operativo contra la pornografía infantil: secuestran casi un millón de archivos con videos y fotos. (Noticia). Disponible online en: https://www.clarin.com/sociedad/operativo-luz-infancia-secuestran-millon-archivos-pornografia-infantil_0_b2QxGiDz_.html.
3. Whelpton, J.: The Psychological Effects Experienced by Computer Forensic Experts Working with Child Pornography. Masters Thesis. University of South Africa. Febrero 2012. Disponible online en: https://core.ac.uk/download/pdf/43169033.pdf.
4. Rivest, R.: The MD5 Message-Digest Algorithm. RFC 1321. Abril 1992.
5. Federal Information Processing Standards Publication: Secure Hash Standard. FIPS PUB 180-1. Abril 1995.
6. Di Iorio, A. H. et al: El Rastro Digital del Delito. Universidad FASTA Ediciones. Marzo 2017.

7. Gonzalez, R. C., Woods, R. E: Digital Image Processing. 3ra Edición. Pearson Prentice Hall. 2008.
8. CCITT International Telegraph and Telephone Consultative Committee: INFORMATION TECHNOLOGY – DIGITAL COMPRESSION AND CODING OF COTINUOUS-TONE STILL IMAGES – REQUIREMENTS AND GUIDELINES. 1993. Disponible online en: https://www.w3.org/Graphics/JPEG/itu-t81.pdf.
9. Boutell, T., Lane, T., et al: Portable Network Graphics (PNG) Specification and Extensions. Disponible online en: http://www.libpng.org/pub/png/spec/.
10. Documentación de OpenCV 4.1: Color conversions. Disponible online en: https://docs.opencv.org/4.1.0/de/d25/imgproc_color_conversions.html.
11. Zauner, C.: Implementation and Benchmarking of Perceptual Image Hash Functions. Master's thesis, Upper Austria University of Applied Sciences, Hagenberg Campus. 2010.
12. Hamming, R.: Error detecting and error correcting codes. The Bell System Technical Journal. Volúmen 29, Número 2. Abril 1950.
13. Krawetz, N.: Looks like it. (Blog personal). Disponible online en: http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html.
14. Krawetz, N.: Kind of Like That. (Blog personal). Disponible online en: http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html.
15. Buchner, J.: ImageHash. Software libre, disponible online en: https://github.com/JohannesBuchner/imagehash.
16. Constanzo, B.: phantom. Software libre, disponible online en: https://github.com/bconstanzo/phantom.
17. Oliphant, T.: Python for Scientific Computing. Computing Science and Engineering. Volúmen 9, Número 3. Junio 2007.
18. Jones, E., Oliphant, T., Peterson, P.: SciPy: Open source scientific tools for Python. http://www.scipy.org/. 2001.
19. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools. Noviembre 2000.
20. Pérez, F., Granger, B. E.: IPython: A System for Interactive Scientific Computing. Computing in Science and Engineering. Volúmen 9, Número 3. Mayo-Junio 2007.