Introducción de marcas de agua en imágenes mediante LSB y resistencia a ruido

## J Sardón. Carlos

**Abstract**

La naturaleza digital de la información hace que su manipulación y copia sea algo trivial e inherente a esta, apareciendo usos no deseados que pueden ser detectados o evitados mediante las marcas de agua. Estas pueden ser introducidas en cualquier soporte digital con finalidades como certificar la legitimidad de este, evitar usos no autorizados u ocultar información. Mediante MATLAB implementaremos la introducción de marcas de agua en imágenes mediante el método del bit menos significativo (LSB) y analizaremos cómo reacciona frente a distintas modificaciones.

**Palabras clave:** marca de agua, ruido, ocultación, transformación, lsb

# **Introducción**

Las marcas de agua en las imágenes es un concepto intuitivo que consiste en introducir una imagen en otra, ya sea de forma más o menos transparente para cualquier espectador. Si bien su popularización llegó con Internet, existía ya mucho antes en el medio físico para evitar la falsificación de billetes o indicar el fabricante de folios de papel, ambos aún presentes.

Las imágenes son transmitidas con gran facilidad, y normalmente, no necesitan de metadatos o algún otro archivo para tener sentido por sí mismas. A la vez que supone una gran ventaja para la distribución de estas, también hace que conceptos como la autoría o los derechos sobre las mismas no sean transmitidos.

Con las marcas de agua se brinda un método para obtener seguridad sin perder las ventajas anteriormente expuestas: el contenido de la imagen para el espectador (debe) permanece igual, no se necesita de información adicional a la que hay en la imagen y la facilidad de transmisión es similar.

El marcado puede ser visible, pero para el fin que se plantea aquí buscamos que sea lo más transparente posible y que el espectador no note diferencia alguna entre la imagen original y la que tiene el marcado. De esta forma se entra en el campo de la esteganografía, que consiste la ocultación de mensajes o información dentro de otros.

En el procesamiento de imágenes digitales, y por tanto en el marcado de agua como parte de este, se puede trabajar dentro del dominio espacial o de la frecuencia. En este documento solo trabajaremos con el primero a través del método del bit menos significativo (LSB) y experimentaremos con distintas aplicaciones y posteriores modificaciones para estudiar su resistencia.

# **Planteamiento teórico**

El marcado de agua se puede realizar para que cualquiera que vea la imagen final se percate de este o no. En el documento nos centramos en este último, y lo hacemos a través del método LSB (less siginificant bit), que consiste en introducir la marca en el bit menos significativo de una imagen.

Considerando las imágenes en blanco y negro y a color, estas pueden tener un único canal o tres, pero sea como fuere la información de los píxeles dentro de cada canal está codificada como máximo con 8 bits, lo que nos deja un rango de 0 a 255.

Con el LSB, se introduce en el último bit la información de la marca de agua, causando en la imagen final una pérdida de máximo 1 unidad por cada píxel, algo que se puede considerar despreciable.

Por un lado, hace que sea sencillo y a un coste mínimo, hablando en términos de complejidad y de la cantidad de información que hay que sacrificar, pero también supone que las marcas que se inserten sólo pueden ser binarias. Así pues, hemos extendido el método del LSB a más bits, introduciendo la información no en el bit menos significativo sino en los bits menos significativos.

Las consecuencias de esto son claras, y es que en función de si queremos un resultado a blanco y negro o a color, podemos insertar una marca con ciertos niveles de gris, pudiendo llegar a 255. Se podrían buscar otros artificios con marcas simples, por ejemplo, indexar los colores presentes en una marca y codificarlos con un número inferior de bits, algo que se antoja útil ya que típicamente las marcas no son complejas y resultaría de utilidad poder insertar una con por ejemplo 8 colores distintos, para lo que se necesitaría 3 bits, un coste más que asumible en una imagen a color. El lado negativo de esto es que necesitamos la indexación de colores realizada en la inserción a la hora de recuperar la marca, lo que haría que la facilidad que ofrece el marcado mediante LSB pierda cierto sentido.

Por último, es preciso mencionar que uno de los fines del marcado de agua que aquí se realiza es evitar el uso y distribución de imágenes sin que al menos haya información de su autoría en ellas. Al ser internet el medio, hay que ser consciente que es muy común que servicios de amplio uso como las redes sociales o las aplicaciones de mensajería apliquen algoritmos de compresión a las imágenes para reducir su tamaño y por tanto el ancho de banda necesario para transmitirlas y el espacio necesario para almacenarlas. Que nuestro método se base, en el peor de los casos respecto a esta situación, en un único bit, puede provocar que perdamos fácilmente la información de la marca de agua haciéndola irrecuperable después de aplicar cualquier algoritmo de compresión.

# **Resolución Práctica**

La implementación de la solución aquí propuesta se ha realizado mediante MATLAB, y no se ha usado ninguna librería fuera de las funciones por defecto y del paquete [***Image Processing Toolbox***](https://es.mathworks.com/products/image.html).

Así pues, una vez realizada la lectura de las imágenes para tratar con ellas como matrices, las únicas funciones usadas para el procesamiento han sido aquellas destinadas a introducir ruido o comprimir imágenes. Además, también se han utilizado [*imbinarize*](https://es.mathworks.com/help/images/ref/imbinarize.html) y [*imresize*](https://es.mathworks.com/help/images/ref/imresize.html) aunque no como parte esencial del algoritmo, sino para ofrecer al usuario mayor facilidad a la hora de escoger una marca en lugar de tener que realizar una búsqueda de una imagen óptima o tener que adaptarla por otros medios.

El algoritmo se antoja sencillo, ya que solo supone sustituir los bits menos significativos de una imagen por los de otra. Antes de la ejecución se deben indicar ciertas preferencias a la hora de introducir la marca de agua para que el resultado se acerque lo máximo posible a los deseado, que se detallan a continuación:

* Sería extraño que la marca seleccionada sea del mismo tamaño que la imagen base, así que para solucionar esto, hay que indicar si se prefiere que la marca se redimensione a la resolución de la imagen base (imresize), se centre, o que se repita tantas veces como quepa.
* Al introducir la marca, esta puede ser binaria (imbinarize) o en blanco y negro. En el último caso, hay que indicar cuántos bits son los que se quieren sustituir de cada píxel de la imagen original para introducir la marca.

Una vez se han establecido las preferencias oportunas sobre las cuestiones anteriores, se ejecuta el algoritmo, que sería el siguiente:

1. Establecer a 0 los bits de la imagen base sobre los que se quiera introducir la marca mediante máscara.
2. Sumar la matriz de la marca a la imagen base.

El algoritmo se ha implementado por medio de operaciones con bits y máscaras. Lo primero que se hace es preparar la marca para ser insertada.

Con la función creaMarca:

* Si se quiere insertar una marca binaria, se usa la función imbinarize.
* Si se quieren usar entre 2 y 7 bits, se realiza un desplazamiento hacia la derecha que elimina los bits que se van a desechar-
* Si se quiere insertar la marca en blanco y negro con todos los bits, no se realiza nada.

Una vez se termina con las operaciones de bits para la marca, necesitamos una matriz del mismo tamaño que la imagen base para operar de forma más cómoda entre matrices. Se ofrecen tres opciones:

* Redimensionado: se usa la imagen imresize para que se cambie la resolución de la imagen base a la de la original, sin mantener la relación de aspecto.
* Centrado: se calcula las coordenadas del centro de la imagen, que es donde se inserta la marca
* Patrón: se replica la marca en una misma matriz tantas veces como sea pertinente hasta que se tenga un tamaño igual o superior a la imagen base, y se corta para que el tamaño final sea el mismo.

Las cuestiones relacionadas con la dimensión y el color poco tienen que ver con el marcado de agua, pero se trata de hacer más cómodo el uso del programa para que el usuario tenga pocas restricciones a la hora de buscar imágenes para usar.

Al insertar la imagen, el algoritmo es distinto en función de si la base es en blanco y negro o en color.

* Si es en blanco y negro, se realiza una operación AND con cada píxel de la matriz y una máscara que tiene a 0 sólo los bits a sustituir. Esto hace que solo tengamos que sumar al resultado la marca que procesamos con la función creaMarca.
* En color, se ha usado un bucle que realiza una iteración por cada bit a insertar. En cada iteración se elimina mediante una máscara el bit a sustituir, se suma le suma el bit de la marca desplazado para ocupar el espacio que le corresponda y se guarda el resultado en la imagen final.

Para que el bucle sea más simple, se ha usado una lista que indica en qué canal va insertado cada bit. La inserción se ha ideado para que cada canal soporte la misma carga de la marca. El hecho de que se inserte bit a bit, hace que cuando cambie la posición en la que hay que insertar los bits haya que cambiar tanto la máscara de eliminación como el número de desplazamientos que se hacen del bit que se extrae para que encaje en su posición.

Por último, para la extracción se tiene un bucle similar al de inserción: tiene el mismo número de iteraciones y también debe ajustar la máscara para extraer bits y el offset para encajarlos en la matriz de la imagen recuperada cada vez que cambia la posición del bit que se extrae.

En el desarrollo del algoritmo, en fases tempranas se implementó la inserción con un bucle for que tenía una iteración por cada píxel. Esto a priori puede suponer más sencillo de idear, pero supone un tiempo de ejecución mucho mayor y resulta más tedioso para depurar.

# **Experimentación**

Con lo anteriormente expuesto, poca experimentación cabe sobre el script más allá de mostrar cómo afecta a la imagen original el hecho introducir una marca, y cómo afecta a la marca introducida el hecho de aplicar distintos ruidos a la imagen que se encuentra ya marcada, para comprobar como afecta tanto a la imagen base como a la marca.

# **Manual de usuario**

El código desarrollado a ejecutar consiste en un script de Matlab, en el cual sólo hay que configurar 7 parámetros, a saber:

* saveImages: si se establece a 1, tanto las imágenes resultado como las intermedias se guardan en un archivo png como los siguientes nombres
  + baseGS.png: la imagen en la que se inserta la marca. Sólo se exporta si se establece el resultado en escala de grises.
  + toMark.png: la marca que se inserta.
  + final.png: la imagen resultado.
  + recovered.png: la marca recuperada.
* Histograms: si se establece a 1 se muestran por pantalla los histogramas de las imágenes de entrada y resultado.
* insertionType:
  + 1 se inserta la marca redimensionada al tamaño de la base
  + 2 se inserta la marca a su tamaño original centrada en la base
  + 3 se repite la marca como un patrón tantas veces como sea necesario hasta que se alcance el mismo tamaño o superior de la imagen base y se recorta para que sea el mismo.
* markDepth: número de bits a insertar de la marca en la base, mínimo 1 máximo 8.
* colouredBase: 1 para que la imagen resultado sea a color, 0 para escala de grises
* noiseType
  + 1 para ruido gausiano
  + 2 para poisson
  + 3 para sal y pimienta
  + 4 para speckle
* noiseIntensity: valor propio de cada ruido, que en el caso de sal y pimienta establece la densidad de ruido (1 céntesima = 1%). Para el resto media y varianza.

Las imágenes de entrada se seleccionan en una ventana de diálogo que aparece al ejecutar el programa.

Hay que tener en cuenta que el programa no implementa ninguna comprobación sobre si los parámetros de configuración son correctos.

# **Conclusiones**

Introducir una sección de conclusiones que incluya propuestas claras de mejora o extensión del trabajo (por ejemplo, si no se han podido alcanzar todos los objetivos iniciales). También con- clusiones sobre los resultados obtenidos, en qué medida difieren de los esperados. También son apropiadas conclusiones sobre las desviaciones en cuanto a la planificación inicial y conclusiones sobre la experiencia de la realización del trabajo.

# **Referencias**

Las referencias se citan así: bla, bla [[2](#_bookmark1)], bla, bla [[1](#_bookmark0)]. La bibliografía debe seguir el estilo de este documento. A continuación aparecen dos ejemplos de referencias bibliográficas: un artículo en una revista y un libro.

\begin{thebibliography}{9}

\bibitem{clave:revista}

Y. O. Mismo, ‘‘Título del artículo", \emph{Revista Publicación Periódica}, Vol. 17, pp. 1-100, 1997.

\bibitem{clave:libro}

U. N. Experto, \emph{Un libro que escribí}, Editorial, 1996.

\end{thebibliography}

El resultado de estos ejemplos puede verse a continuación, con las referencias ordenadas al- fabéticamente por autores.

# **References**

1. U. N. Experto, *Un libro que escribí*, Editorial, 1996.
2. Y. O. Mismo, “Título del artículo”, *Revista Publicación Periódica*, Vol. 17, pp. 1-100, 1997.

**Anexo I: Tabla de tiempos**

Se debe justificar el trabajo realizado por cada componente del grupo, indicando el **tiempo total que cada miembro del grupo ha dedicado** al trabajo (lo que puede implicar diferencia de notas obtenidas por los distintos miembros del grupo). El trabajo realizado debe ser de **70 horas por alumno**. Además, debe haber un plan de trabajo detallado con las tareas realizadas por cada miembro del grupo. Para esto último, se puede usar la tabla siguiente o bien documentos generados por la herramienta de gestión de proyectos que se use, como Projetsii o Microsoft Project, por ejemplo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha de la actividad | Inicio | Fin | Tiempo | Miembros | Actividad realizada |