LABORATORIO DI TELECOMUNICAZIONI

"PROGETTO WATERMARK"

DOCENTE:	AUTORE:
GIANMARCO ROMANO	FRANCESCO PIZZO

INTRODUZIONE

Il progetto presentato sussiste in un programma, realizzato in ambiente Matlab, che prevede la possibilità di inserire un watermark all'interno di una immagine.

Wikipedia dà la seguente definizione di watermarking: *"'inclusione di informazioni all'interno di un file multimediale, o di altro genere, che può essere successivamente rilevato o estratto";* sostanzialmente si tratta di una tecnica utilizzata in diversi ambiti (dalla protezione di contenuti -inserendo firme digitali o marchi di copyright- alla crittografia dei dati), utilizzata persino dal terrorismo (watermarking stenografico).

Fonte: Wikipedia

An example showing how terrorists may use forum avatars to send hidden messages. This avatar contains the message "Boss said that we should blow up the bridge at midnight."

Il punto di partenza è stato quello di cercare una tecnica per inserire una stringa di testo all'interno di una immagine scelta dall'utilizzatore del programma; tale stringa deve però non essere visibile all'occhio umano, che non deve percepire variazioni di aspetto o colori nell'immagine.

Successivamente l'implementazione di tale tecnica è stata affinata, permettendo all'utente di utilizzare anche una chiave numerica per crittografare il file.

Infine, visto che tale tecnica viene implementata lavorando a basso livello (operando sui singoli bit dei dati e delle immagini), si è cercato di ripetere l'operazione di watermarking, nascondendo stavolta, invece che una stringa di caratteri, una immagine (di dimensioni minori rispetto all'immagine "contenitore").

TECNICA GENERALE

La tecnica generale è molto semplice da descrivere. Supponiamo di avere un dato da voler inserire nell'immagine, senza preoccuparci per il momento di cosa questo dato rappresenti. Indichiamo allora con **W** la rappresentazione in binario di tale dato. L'immagine nella quale lo si vuole inserire invece può essere vista come una o più matrici di pixel, dove ogni elemento di tali matrici è rappresentato su 8 bit.

La tecnica consiste nel modificare il bit meno significativo (LSB) di ogni elemento della matrice, mettendo al posto di ognuno, un bit di **W**.

ESEMPIO 1:

W=**10011010**

Supponiamo per semplicità che l'immagine sia rappresentata da una sola matrice (una componente di colore) di 4x4 pixel:

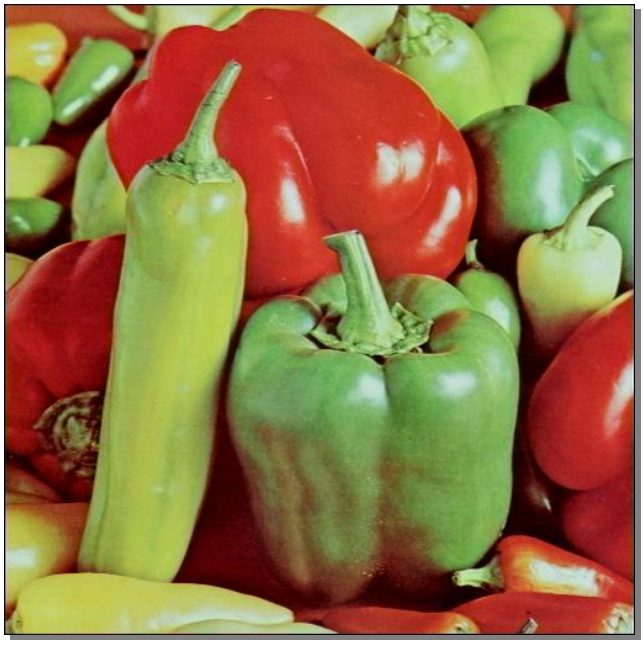
11000111	00010100	11111111	01001010
01000100	10111001	00101000	11010010
10101010	01011011	00101011	11011001
11100011	11111111	11010010	00101010

Una volta inserito il watermark **W**=**10011010** l'immagine diventerà allora:

1100011**0**	0001010**1**	11111111	01001010
0100010**1**	1011100**0**	00101000	11010010
1010101**0**	0101101**0**	00101011	11011001
1110001**1**	1111111**1**	11010010	00101010

Come si può notare, il watermark è stato inserito partendo dal suo bit meno significativo e avanzando per colonna nella matrice che rappresenta l'immagine.
Notiamo che alcuni elementi della matrice avevano già il proprio bit meno significativo settato al valore desiderato, ciò vuol dire che il numero di elementi della matrice modificati sono al più pari al numero di bit di **W** (8 in questo esempio). Inoltre la modifica effettuata corrisponde alla modifica degli elementi di un'unica componente di colore, facendo variare la sua rappresentazione al livello successivo o precedente, il che per l'occhio umano è praticamente indistinguibile.

Immagine originale



Contiene la stringa "Hello, World!

INSERIMENTO STRINGA

L'inserimento di una stringa di caratteri presenta nello specifico qualche problematica in più rispetto al caso generale. Inoltre, nel caso più generale, se l'immagine in cui nascondere i dati è in formato RGB, si potrebbe dover lavorare su una struttura di tre matrici (le tre componenti di colore).

In realtà il fatto di lavorare con una o più matrici non è un grosso problema. Nello specifico di Matlab, una volta letta l'immagine, essa viene posta in un unico vettore che contiene in sequenza le colonne della singola immagine:

```
im=img(:);     %Importo l'immagine in un unico vettore colonna
```

Un problema di maggiore rilevanza invece è il come riuscire a comprendere in fase di estrazione dove termina la stringa. Per risolvere tale problema si è pensato di assicurare il corretto funzionamento di questa applicazione con tutti e soli i caratteri della tabella ASCII STANDARD. Il vantaggio sta nel fatto che i caratteri ascii standard possono essere rappresentati su 8 bit, ma i bit significativi al riconoscimento del carattere sono in realtà 7. In questo modo è stato possibile utilizzare una sorta di "carattere tappo" rappresentato su 8 bit ma che non appartiene ai caratteri standard. A tale scopo si è scelto il carattere 255 ovvero 8 bit tutti settati a 1.

Indicato con Nc il numero di caratteri componenti la stringa, dovremo allora inserire (Nc+1)*8 bit, il che vuol dire che per contenere tale stringa è necessaria una immagine di (Nc+1)*8 pixel nel caso di immagini ad una componente di colore oppure (Nc+1)*8/3 pixel nel caso di immagini RGB.

L'inserimento dei bit viene effettuato in modo lineare e per colonne, sicché sarà semplice in fase di dewatermarking estrarre tutti i caratteri, fino al carattere tappo, in quanto la decodifica avviene nel medesimo ordine dell'inserimento.

STRINGA CON KEY NUMERICA

Il metodo utilizzato finora è abbastanza efficiente, ma nel caso in cui si volessero crittografare dati sensibili, risulta molto esposto ad attacchi se si conosce bene l'algoritmo utilizzato.

Una soluzione a questo problema è stata quella di inserire i bit del watermark non in modo lineare, bensì in modalità pseudo casuale! In pratica si tratta di inserire i bit in modo "sparso" all'interno della matrice che rappresenta l'immagine. Inizialmente si potrebbe pensare di utilizzare il comando *rand* di matlab per generare l'indice nella matrice. Fissando il seme in modo arbitrario si potrebbe allora proseguire nell'inserimento, mentre per l'estrazione l'utente deve necessariamente conoscere il seme (key) utilizzato! Tuttavia utilizzare rand presenta molti problemi, in quanto potrebbe fornire 2 o più volte lo stesso indice per l'accesso alla matrice, il che comporterebbe la perdita di dati e il malfunzionamento di tutta l'applicazione!

Una alternativa a rand è allora un comando molto più utile al nostro scopo, basato sempre sul comando rand: la funzione ***randperm***! randperm fornisce una permutazione dei valori che vanno da 1 al valore passatogli in input, scelta in modo pseudo casuale grazie al comando rand.

ESEMPIO 1.1:

Supponiamo di avere lo stesso watermark e la stessa immagine dell'esempio 1:

W=**10011010**

11000111	00010100	11111111	01001010
01000100	10111001	00101000	11010010
10101010	01011011	00101011	11011001
11100011	11111111	11010010	00101010

Questa volta però invece di inserire i bit per colonna, utilizziamo il comando randperm per calcolare l'indice:

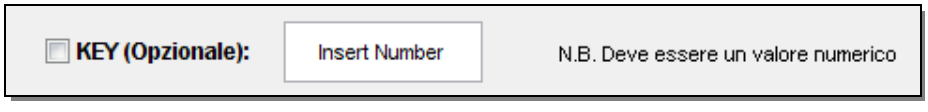
```
rand('seed',key);            %Fisso il seme della funzione rand
im=img(:);                   %Importo l'immagine in un unico vettore colonna
index=randperm(length(im)); %Creo un vettore di indici utilizzano randperm
```

index conterrà allora: [6 3 16 11 7 14 8 5 15 1 2 4 13 9 10 12]

La matrice dell'immagine sarà allora modificata in questo modo: (**W**=**10011010**)

11000111	00010100	11111111	01001010
01000100	1011100**0**	00101000	1101001**0**
1010101**1**	0101101**1**	0010101**1**	1101100**1**
11100011	1111111**0**	11010010	0010101**0**

Per ricostruire il dato quindi bisogna conoscere necessariamente la key utilizzata, che permette di calcolare la giusta permutazione. Questo metodo è molto efficiente in quanto le possibili permutazioni sono in numero molto elevato: già per questo semplice esempio di un'immagine 4x4 pixel ad una componente (quindi piccolissima), si hanno 16! diverse permutazioni, ovvero 20.922.789.888.000 diverse permutazioni! E' possibile quindi immaginare quante siano per un'immagine di, ad esempio, 800x600 pixel rgb!



Campo della GUI dedicato alla key

INSERIMENTO IMMAGINE

Un po' diversa è la situazione quando, invece che una stringa, si tenta di inserire un'immagine all'interno di un'altra. Il metodo è sostanzialmente lo stesso ma le problematiche da aggirare sono abbastanza diverse.

Il primo problema è che non si può utilizzare un carattere tappo, in quanto non si potrebbe più distinguere se si tratta di un dato o dell'indicazione di fine immagine. Si è pensato allora di inserire in fase di watermarking, una sorta di header di dimensione fissa, che contenga informazioni relative alla dimensione dell'immagine inserita. Sostanzialmente l'header deve contenere altezza e larghezza dell'immagine (in pixel) e il numero di matrici che la codificano. Si è deciso allora di utilizzare un'header di 4 byte, suddivisi in questo modo:

Altezza	(n° di righe della matrice):	15 bit	massimo 32.767 pixel
Larghezza	(n° di colonne della matrice):	15 bit	massimo 32.767 pixel
Componenti	(n° di matrici):	2 bit	massimo 3 matrici

Facendo così, in fase di decodifica si può ricostruire l'immagine watermark senza problemi.

Quindi l'immagine "contenitore" deve essere abbastanza grande da contenere il watermark. Cerchiamo allora di quantificare questo "abbastanza"!

Indichiamo con NW il numero di pixel dell'immagine watermark e con CW il numero di componenti di tale immagine (es. 3 nel caso RGB), la quale per essere memorizzata necessita allora di:

(NW*CW*8 + 32) bit

Visto che tale dato viene memorizzato nel bit meno significativo di ogni pixel dell'immagine "contenitore", si ha allora che tale immagine deve possedere un numero di pixel (moltiplicato per le proprie componenti) pari al numero di bit necessario a memorizzare il watermark.

ESEMPIO 2:

Supponiamo di avere un logo RGB di dimensioni 50x50 pixel. Possiamo calcolare allora l'occupazione in bit di tale logo:

W = (50*50*3*8 + 32) bit = 60.032 bit

Di conseguenza, un'immagine contenitore di tipo RGB che riesca a contenere tale dato deve avere almeno 60.032/3 pixel! Se l'immagine è quadrata dovrà allora avere 142x142 pixel.

Infatti 142*142*3 = 60.492.

IMMAGINE CON KEY

Per quanto riguarda l'utilizzo di una key invece, è stato applicato lo stesso metodo utilizzato per le stringhe, tramite il comando randperm, con la differenza che l'header, nonostante venga anch'esso inserito in modo

sparso, resta sempre confinato nei primi 32 pixel dell'immagine contenitore! Tuttavia utilizzando una key resta molto complicato risalire al watermark senza conoscerla.

SALVATAGGIO IMMAGINE WATERMARKED

Sia se si sceglie di inserire un watermark testuale, sia se si vuole inserire una immagine, si rende necessario, per un successivo utilizzo, salvare l'immagine watermarked su disco.

Bisogna allora preservare il watermark contenuto in tale immagine. Per fare ciò è necessario che il formato in cui viene salvata l'immagine non sia affetto da perdite (come lo è ad esempio jpg). Per questo motivo il salvataggio viene effettuato direttamente dall'applicazione, che permette all'utente di scegliere il nome del file e la codifica, a scelta tra bitmap, tiff e png!

Questa immagine contiene al suo interno il seguente logo ed è stata utilizzata come key la sequenza 11235813

Logo