

Image Forensics (Parte 2)

Nella scorsa parte abbiamo visto quindi delle tecniche per stabilire, a partire da Chromatic Aberrations, se una foto è stata fatta da un certo dispositivo di un certo modello e fare quello che si dice *Profile Linking* tra SMP profile.

Adesso, invece, tratteremo di traccia *Coding Based* dove parleremo delle tracce che possono lasciare gli algoritmi di compressione.

Infatti le **Compressioni con perdita (Lossy)** lasciano inevitabilmente impronte relative all'architettura specifica del codice di trasmissione.

In particolare se la compressione è in JPEG possiamo analizzare:

- Compressione artefatti lasciati nel dominio dello spazio.
- Compressione artefatti lasciati nel dominio della frequenza.

Compressione artefatti lasciati nel dominio dello spazio:

JPEG lavora su un'immagine divisa in 8×8 blocchi, trasformati poi individualmente e quantizzati.

I difetti (artefatti) appaiono spesso sotto forma di **linee orizzontali e verticali** lungo i bordi dei blocchi.

Questo succede perché le informazioni di frequenza vengono elaborate indipendentemente **blocco per blocco**, e quando i bordi dei blocchi non si allineano perfettamente, si creano queste **discontinuità visive nette**.

Una manipolazione può perturbare questi artefatti "bloccosi".

Domanda: Come si può vedere se un'immagine è stata precedentemente compressa?

Idea: vedere se ci sono discontinuità tra i blocchi

- se non si ha compressione, la differenza tra i blocchi dovrebbe essere piccola, quindi sono simili.
- se si ha avuto la compressione, la differenza tra i blocchi dovrebbero essere accentuate.

Assumiamo che la griglia dei blocchi è conosciuta.

➡ Per ogni 8×8 blocchi, si selezionano 4 pixel interni e 4 pixel sui bordi dove i pixel interni partono da:

$$i = 4, 12, 20, 28, \dots$$

$$j = 4, 12, 20, 28, \dots$$

mentre quelli esterni si calcolano partendo da il primo pixel centrale e gli si aggiunge (4,4). I valori dei pixel vanno da A a H.

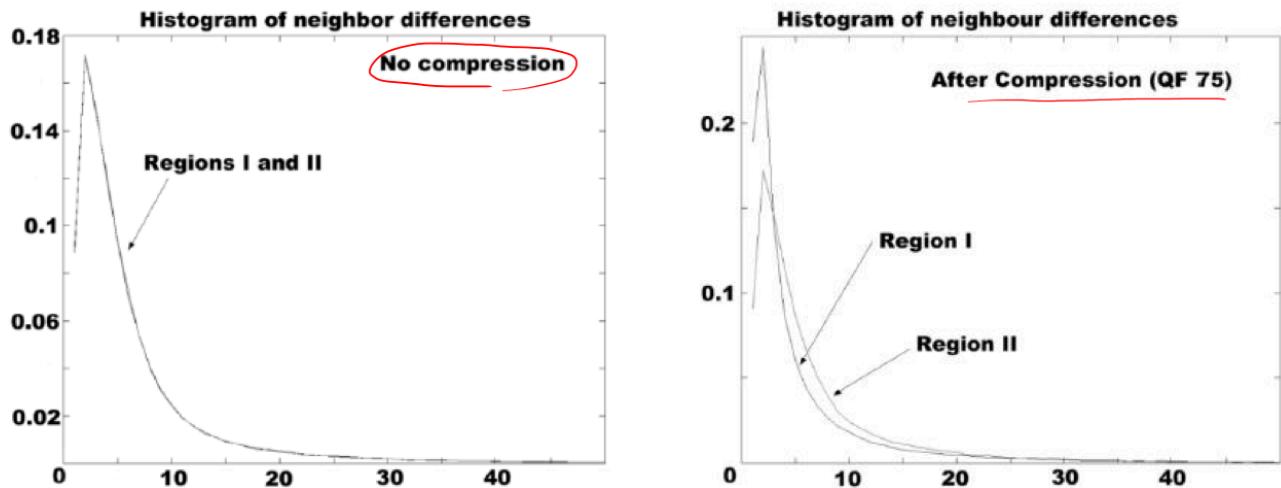


Si calcola poi la differenza diagonale ovvero:

$$Z' = |A + D - B - C|$$

$$Z'' = |E + H - F - G|$$

Si ottengono gli istogrammi normalizzati(divide ogni frequenza per il **numero totale dei campioni**, trasformando i conteggi in **probabilità**).



Spieghiamo questo grafico:

- **Asse x:** c'è il valore della differenza tra pixel adiacenti
- **Asse y:** frequenza relativa delle differenze tra pixel adiacenti.

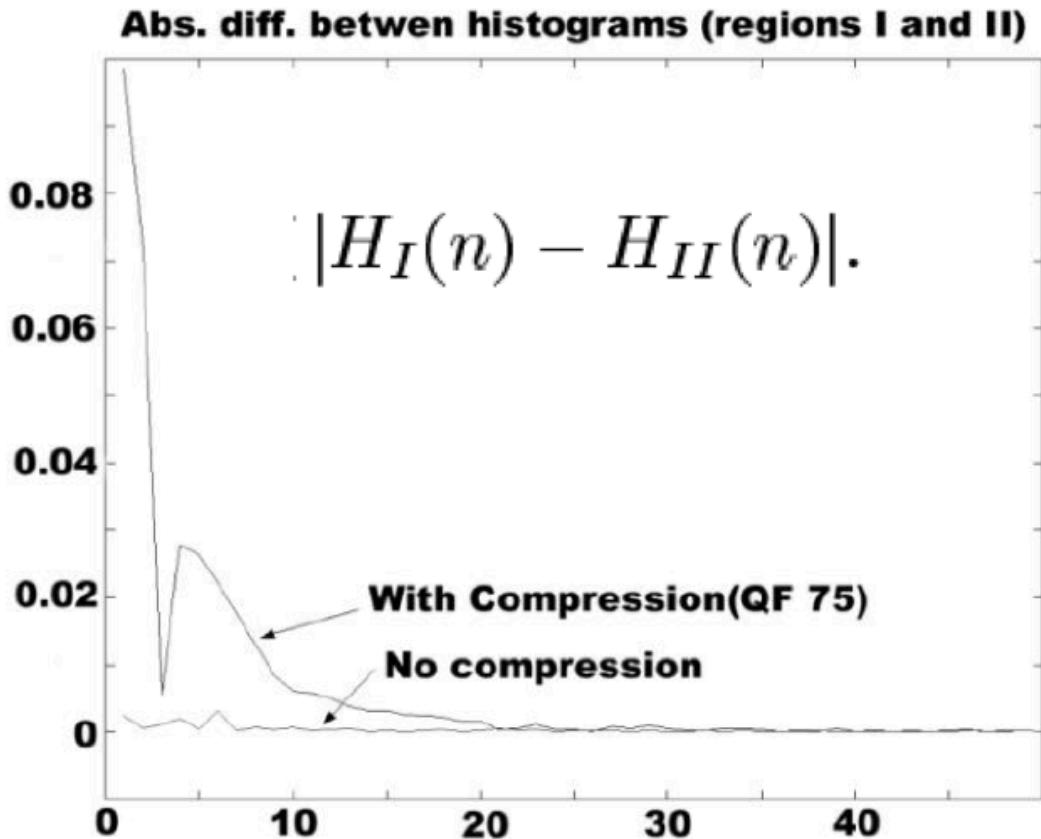
Nel primo grafico si vede che sia nei bordi che all'interno si hanno piccole differenze con più alta frequenza "alta", non si ha una netta distinzione tra regione I e II e via via che si aumenta con le differenze diminuisce la probabilità.

Mentre nel secondo grafico, le due regioni si distinguono:

- **Regione I:** si ha cambiamenti di pixel piccoli con più alta probabilità
- **Regione II:** il grafico si schiaccia evidenziando cambiamenti più grandi con più probabilità.

Da qui si può notare, quindi, che i pixel sui bordi mostrano più discontinuità rispetto ai pixel interni.

Se si fa poi la differenza tra i due istogrammi, si evidenzia ancora meglio quella con compressione e quella senza.



Ora, per affermare che c'è stato un cambiamento si utilizza sempre la soglia e la si confronta con K che è definita in questo modo:

$$K = \sum_n |H_I(n) - H_{II}(n)|$$

se $K \geq \tau$ allora I è stata compressa.

Compressione artefatti lasciati nel dominio della frequenza:

8×8 DCT Quantizzazione $F_Q(u, v)$

Dove $F_Q(u, v)$ sono i coefficienti sono i coefficienti DCT quantizzati attraverso la tabella, dove $F_Q(u, v)$ sono interi.

$$F_Q(u, v) = \text{Round}\left(\frac{F(u, v)}{Q(u, v)}\right)$$

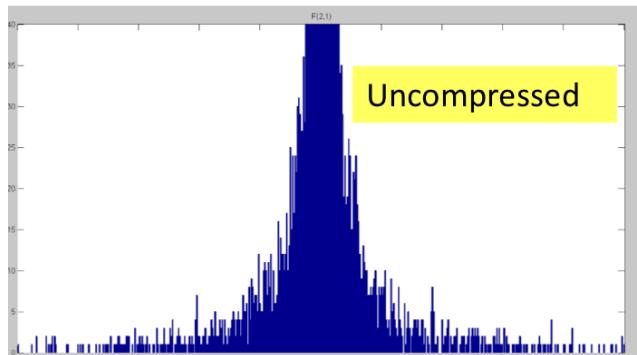
Di conseguenza, il valore di ogni DCT **dequantizzato** sarà un multiplo intero di $Q(u, v)$. E rimarrà un multiplo anche se viene ritrasformato in DCT.

Compressione Singola

Domanda: l'immagine è stata precedentemente compressa?

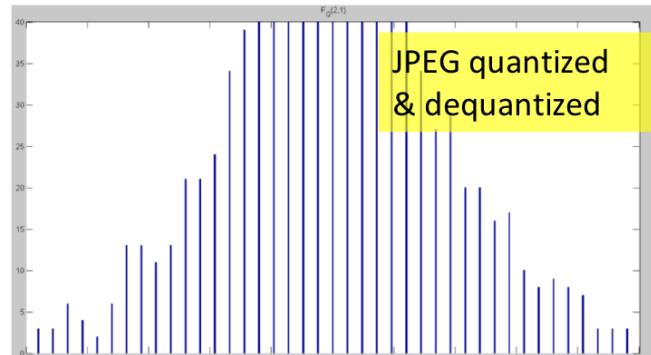
Idea: cercare tracce lasciate nell'istogramma dei coefficienti DCT.

Si compie DCT 8×8 e collezioniamo i coefficienti DCT della stessa frequenza (u,v) .



Histogram for a typical image without compression

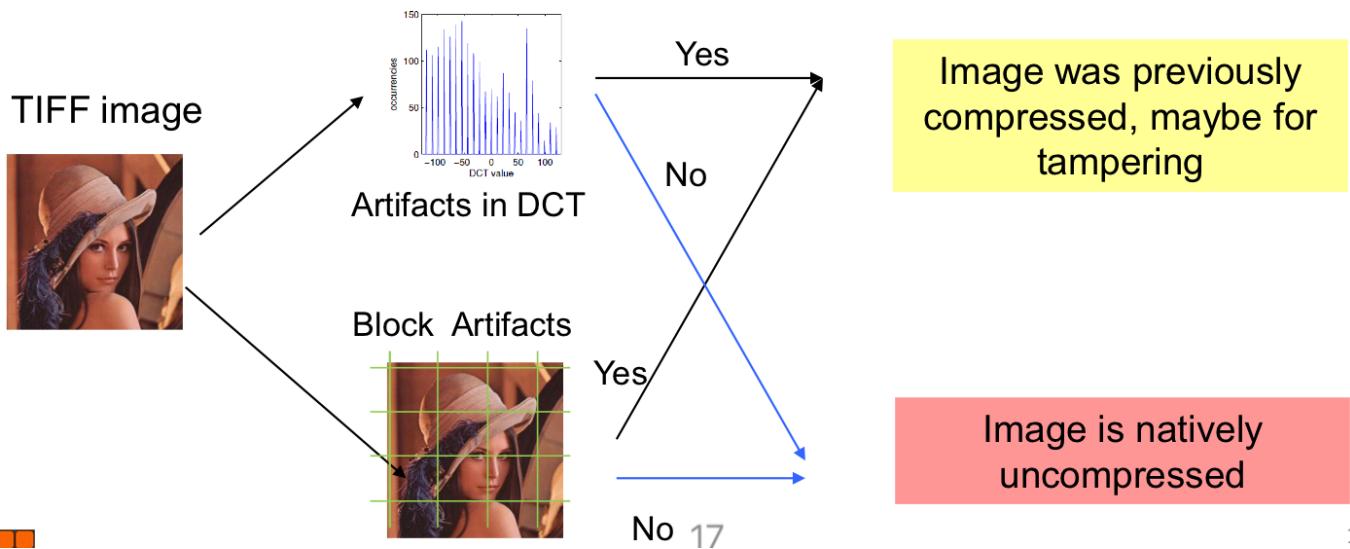
$F(2,1)$: Real value



Histogram if the image previously underwent compression where the coefficient $(2,1)$ was quantized with value 12

$$F(2,1) \approx F_{DEQ}(2,1) = n * Q(2,1) = n * 12$$

Quindi l'immagine che è sotto analisi è in un formato non compresso o nativo se si fanno le seguenti analisi:

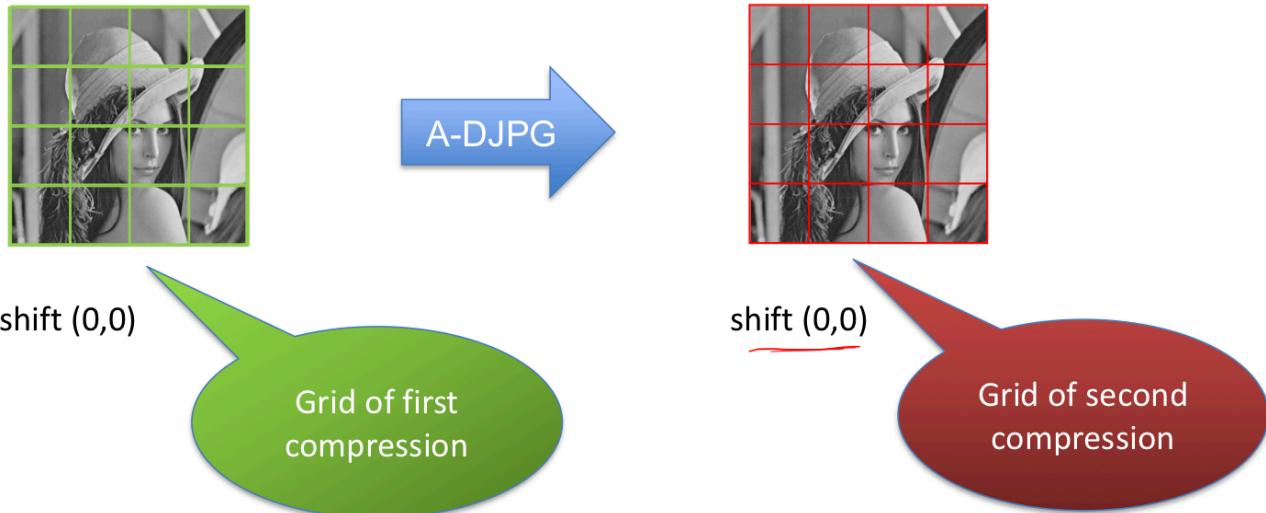


Manomissione (Tampering) delle immagini JPEG:

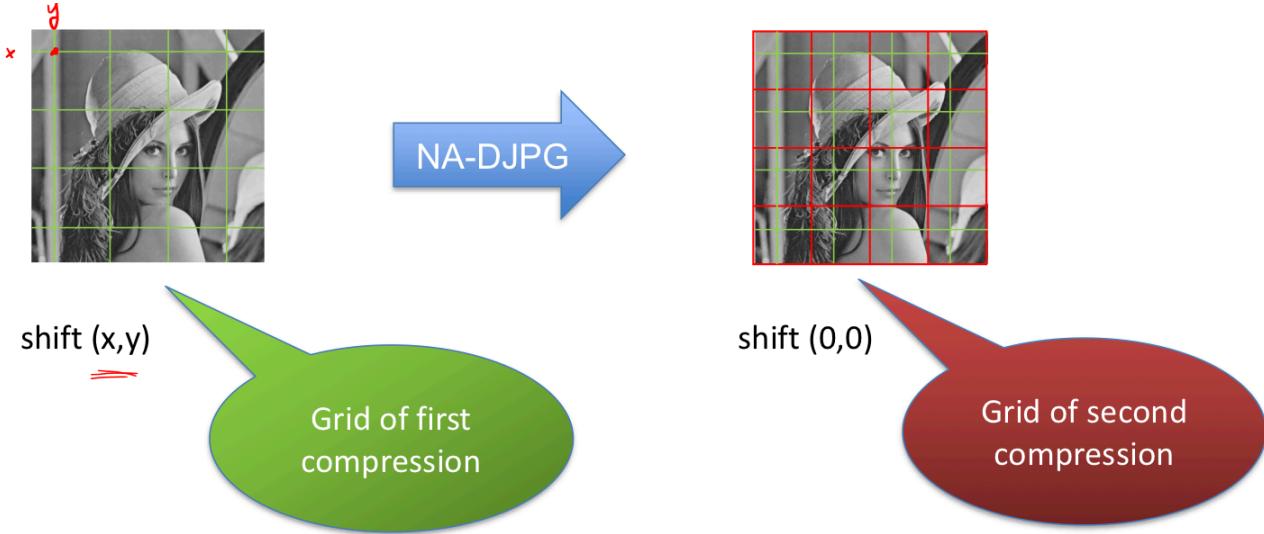
Qualsiasi manomissione digitale ha bisogno che l'immagine sia caricata in un software di foto editing e poi verrà risalvato. Poiché molte immagini sono salvati in JPEG è possibile che entrambi, sia l'originale che quella manipolata, siano salvati in questo formato.

L'immagine manipolata è quindi compressa due volte, si hanno due differenti casi:

- **Aligned double JPG compression(A-DJPEG)**

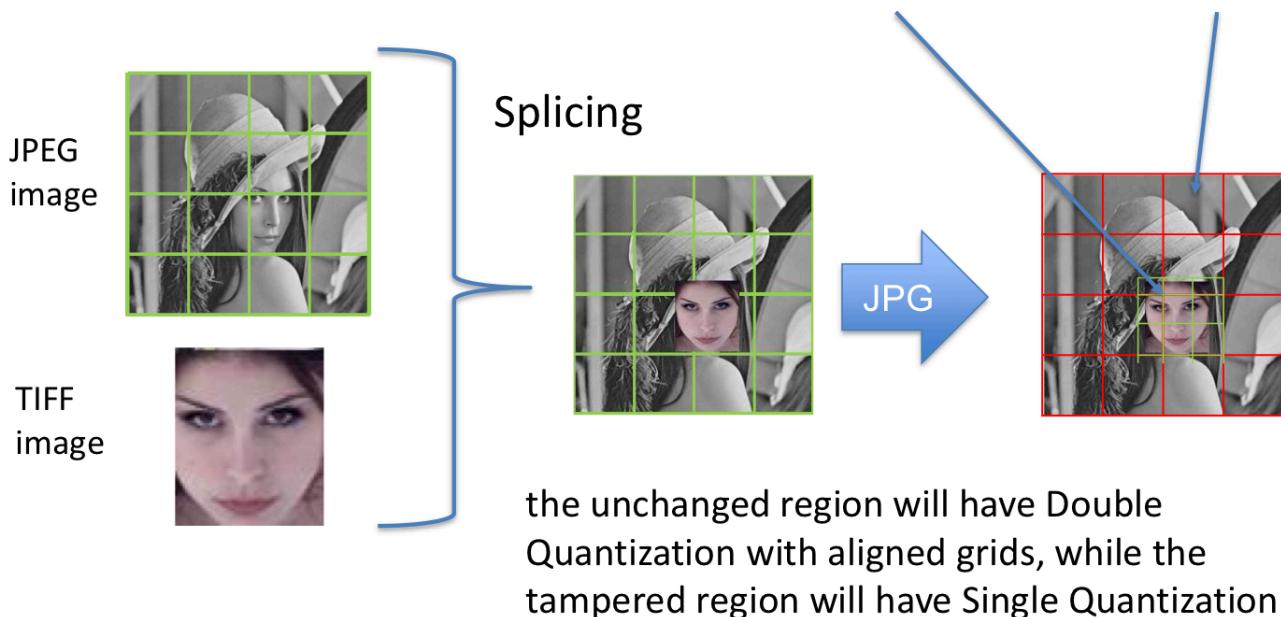


- **Not Aligned double JPG compression(NA-DJPEG)**

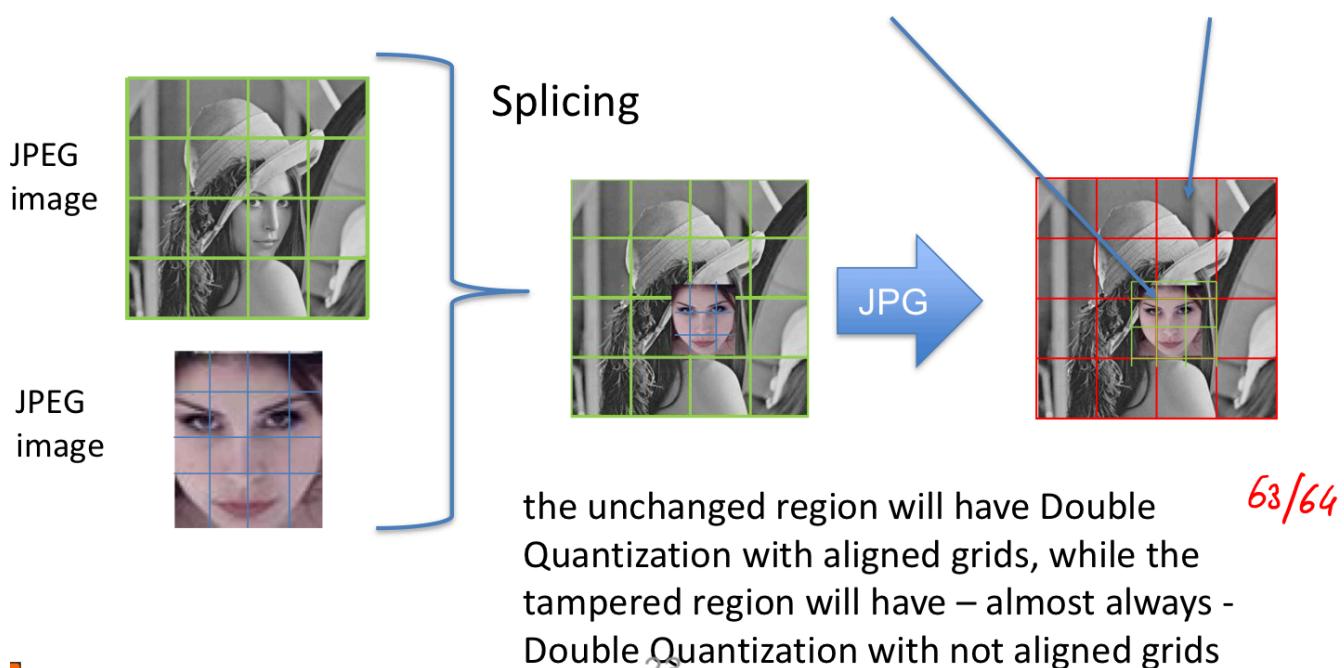


Dato da un possibile **splicing**, alcune aree nell'immagine manipolata potranno esibire A-DJPEG o NA-DJPEG a seconda del tipo dell' immagine.

Nella pagina sotto vediamo che alcuni blocchi sono **single JPEG**, gli altri A-DJPEG



Infatti il tipo TIFF non è compresso ma è un formato contenitore. Di conseguenza nella figura sopra la **doppia quantizzazione** si ha nella parte non cambiata mentre la singola quantizzazione si ha in quella manomessa.

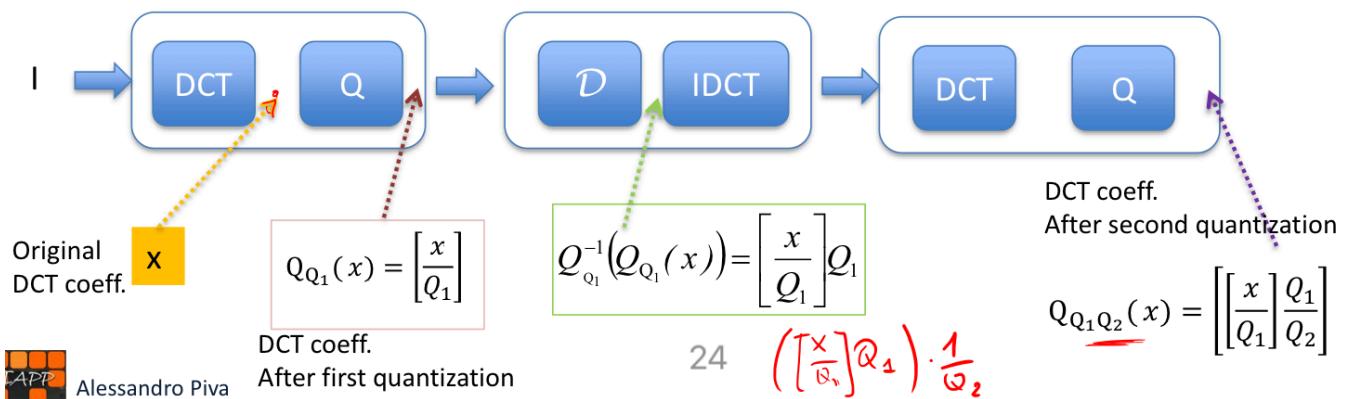


In questo caso, però, le regioni non cambiate sono in A-DJPEG mentre la zona manomessa avrà NA-DJPEG.

A-DJPEG: DQ (Double Quantization) effect:



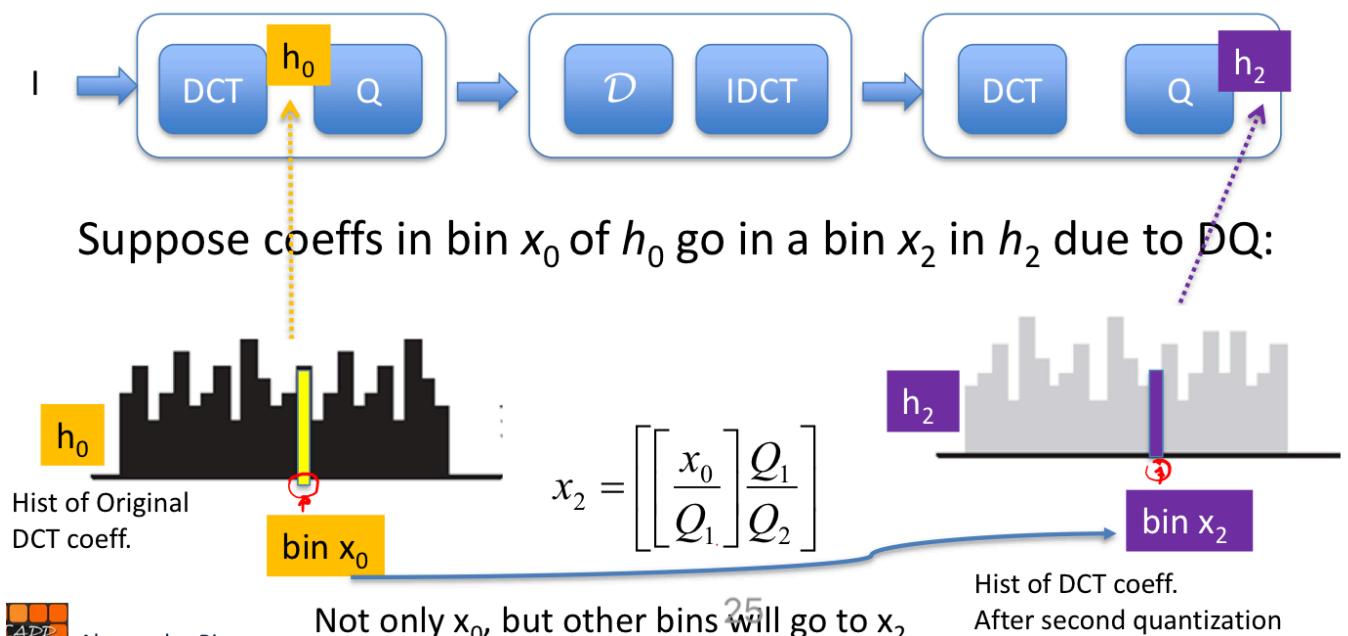
What happens to DCT coefficients:



Spieghiamo la figura sopra, l'immagine viene trasformata nel dominio di frequenza con coefficienti DCT che vengono quindi **quantizzati**, la matrice di quantizzazione viene salvato nell' header JPEG e quindi viene compresso.

Il file jpeg viene quindi messo nel software di editing, dove viene quindi decompresso e viene fatto il DCT Inverso, viene manomesso il contenuto e poi risalvato in JPEG passando quindi da una seconda quantizzazione.

Cerchiamo di notare gli istogrammi dei coefficienti DCT



Vediamo che **bin** sono le colonne dell'istogramma.

La relazione tra i bin di \mathbf{h}_0 e \mathbf{h}_2 non è necessariamente uno-a-uno.

Quindi $n(x_2)$ ovvero il numero di bin in \mathbf{h}_0 che contribuisce al bin x_2 nell'istogramma DQ \mathbf{h}_2 dipende dal valore x_2 e Q_1 , Q_2 :

$$n(x_2) = R(x_2) - L(x_2)$$

dove:

where
$$\begin{cases} R(x) = Q_1 \left(\left\lfloor \frac{Q_2}{Q_1} \left(x + \frac{1}{2} \right) \right\rfloor + \frac{1}{2} \right) \\ L(x) = Q_1 \left(\left\lceil \frac{Q_2}{Q_1} \left(x - \frac{1}{2} \right) \right\rceil - \frac{1}{2} \right) \end{cases}$$

Vediamo quindi che questa funzione è periodica e introduce pattern (picchi e valli) negli istogrammi dei coefficienti quantizzati doppiamente.

Questo è il DQ effect.

Se $Q_2 > Q_1$: la seconda compressione è di peggior qualità rispetto alla prima, si svilupperà un pattern periodici nell'istogramma con picchi e valli.

Se $Q_2 < Q_1$: la seconda compressione sarà di miglior qualità rispetto alla prima. L'istogramma ha periodicamente dei valori mancanti.

Questi artefatti possono essere visti nel dominio di Fourier con forti picchi a medie e grandi frequenze.

Si può stimare Quantization Function che è stato utilizzato. Q_2 può essere trovato nella tabella di quantizzazione salvata nel file JPEG. Mentre Q_1 può essere dedotto dalla locazione dei picchi di frequenza nella trasformata di Fourier dell'istogramma dei coefficienti DCT.

NA-DJPEG Detection & Estimation:

Si ha quindi un immagine in input, dove sappiamo la QF data dall'header della foto, il blocco non è shiftato.

Assumiamo quindi che l'immagine I_2 è un immagine decompressa con una QF_2 dell'immagine I_1 , con griglia allineata all'angolo in alto a sinistra(quindi non shiftata).

Quindi, decomprimendo il JPEG di I_1 si ottiene I_2 .

I_2 può essere modellato in questo modo:

$$I_2 = D_{00}^{-1}Q_2(D_{00}I_1) + E_2 = I_1 + R_2$$

dove:

- I_1 : immagine originale
- D_{00} : 8×8 DCT con griglia allineata con l'angoli in alto a sinistra.
- D_{00}^{-1} : 8×8 IDCT con solita griglia allineata con angolo in alto a sinistra.
- Q_2 : è la quantizzazione.
- E_2 : errore dovuto al troncamento dei valori.
- R_2 è l'errore totale

Supponiamo a questo punto che I_1 fosse precedentemente compressa con una funzione di quantizzazione QF_1 e che la griglia sia shiftata di (x,y) partendo quindi da una immagine I_0 .

Quindi I_2 è stato doppiamente compresso e alla fine decompresso, di conseguenza I_2 si può modellare così:

$$I_2 = I_1 + R_2 = (D_{xy}^{-1}Q_1(D_{xy}I_0) + E_1) + R_2$$

Vogliamo quindi *analizzare* l'immagine I_2 affinchè si possa capire se è avvenuta una sola compressione.

Ora immaginiamo di dover di nuovo comprimere l'immagine **shiftando** di un certo (i,j) si possono avere quindi queste casistiche:

Case a)

$$(i,j) = (0,0)$$

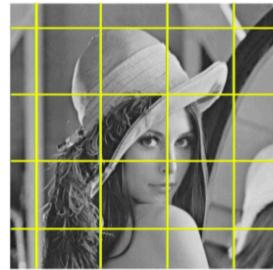
Blocks aligned
with last
compression



Case b)

$$(i,j) = (x,y)$$

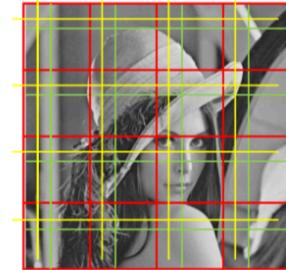
Blocks aligned
with *possible*
first compression



Case c)

$$\begin{aligned} (i,j) &\neq (x,y) \\ (i,j) &\neq (0,0) \end{aligned}$$

grid is aligned to
neither of the two
JPEG compressions



dove se fosse $(i,j)=(0,0)$ allora sarebbe uguale alla prima compressione.

$$D_{i,j}I_2 = D_{0,0}I_2 = D_{0,0}(D_{0,0}^{-1}Q_2(D_{0,0}I_1 + E_2)) = Q_2(D_{0,0}I_1) + D_{0,0}E_2$$

Da qui si nota che l'errore non fa che aumentare...

Se invece $(i,j)=(x,y)$ allora:

$$\begin{aligned} D_{i,j}I_2 &= D_{x,y}I_2 = D_{x,y}(I_1 + R_2) = D_{x,y}((D_{x,y}^{-1}Q_1(D_{x,y}I_0) + E_1) + R_2) \\ &= Q_1(D_{x,y}I_0) + D_{x,y}(E_1 + R_2) \end{aligned}$$

Se invece siamo nel **caso c** allora:

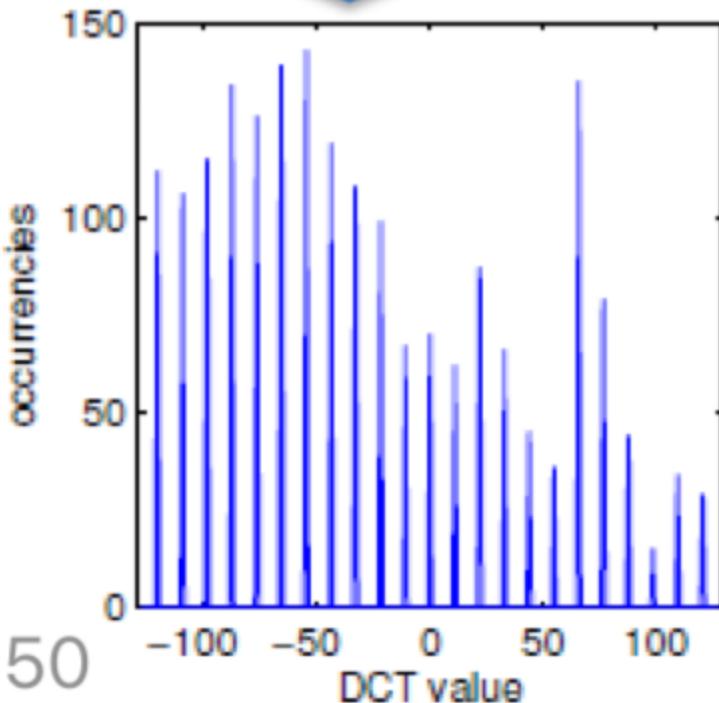
$$D_{i,j}I_2 = D_{i,j}D_{0,0}^{-1}Q_2(D_{0,0}I_0) + D_{i,j}E_2$$

Complessivamente si può vedere così:

$$D_{ij}I_2 = \begin{cases} Q_2(D_{00}I_1) + D_{00}E_2 & i = 0, j = 0 \\ Q_1(D_{xy}I_0) + D_{xy}(E_1 + R_2) & i = x, j = y \\ D_{ij}D_{00}^{-1}Q_2(D_{00}I_1) + D_{ij}E_2 & \text{elsewhere} \end{cases}$$

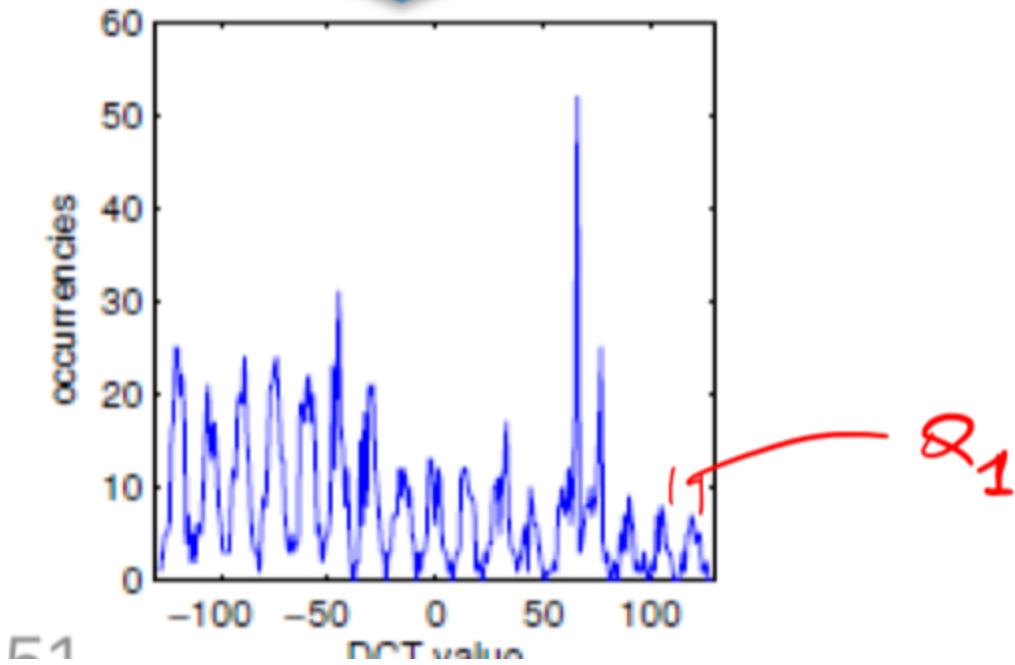
Notiamo una cosa importante:

- $i = 0, j = 0$: i coefficienti tendono a avvicinarsi attorno ai reticoli definiti da $Q_2()$ con una dispersione dovuta alla presenza di errori. In questo caso la dispersione è limitata con una distribuzione Gaussiana a media nulla e una variazione di 1/12.

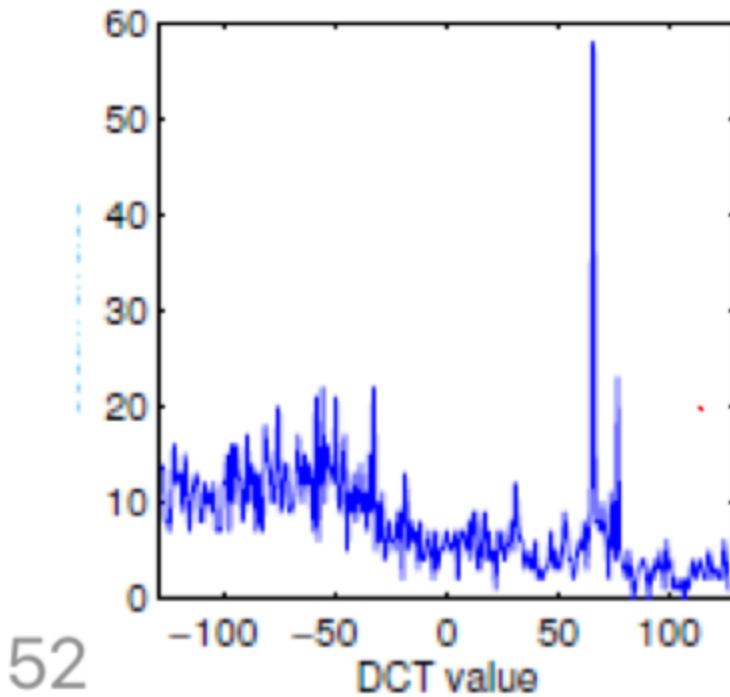


- $i = x, j = y$: i coefficienti tendono ad avvicinarsi attorno ai reticoli definiti da $Q_1()$ e con una dispersione che dipende da R_2 . Quindi è una distribuzione gaussiana con media nulla e

varianza $((Q_2^2 + 1)/12)$.



- *altrimenti* : i coefficienti non si avvicinano ad un pilone.



Idea: misurando come i coefficienti DCT sono ammucchiati attorno ad un pilone per ogni possibile shift della griglia.

L'effetto è più evidente nei coefficienti DCT e rendono la individuazione più semplice, poichè si ha un componente DC per ogni blocco.

Il raggrupparsi attorno ad un pilone si può misurare guardando la periodicità dell'istogramma . Si valuta considerando la sua trasformata di Fourier alle frequenze il quale sono il reciproco del passo Q:

$$f_{i,j}(Q) = \sum_k h_{i,j}(k) e^{-j \frac{2\pi k}{Q}}$$

con $Q \in \mathbb{Z} \setminus \{0\}$

- $h_{i,j}(k)$ è il coefficiente DC dell'istogramma per uno shift della griglia (i,j)
- Q: è il possibile passo di quantizzazione di un coefficiente DC

La Trasformata di Fourier avrà quindi la massima ampiezza se lo shift(i,j) e Q sono **gli stessi parametri della compressione precedente**.

Questo perchè:

- La quantizzazione introduce una periodicità nei coefficienti DCT.
- La Trasformata di Fourier cattura questa periodicità.
- Se la seconda compressione condivide gli stessi parametri della prima (shift e fattore di quantizzazione), questa periodicità viene rafforzata, causando un **massimo nell'ampiezza** della trasformata.

Problema: Non sappiamo quale spostamento è stato applicato

Abbiamo da provare tutte i possibili spostamenti delle griglie. E per ogni spostamento (i,j):

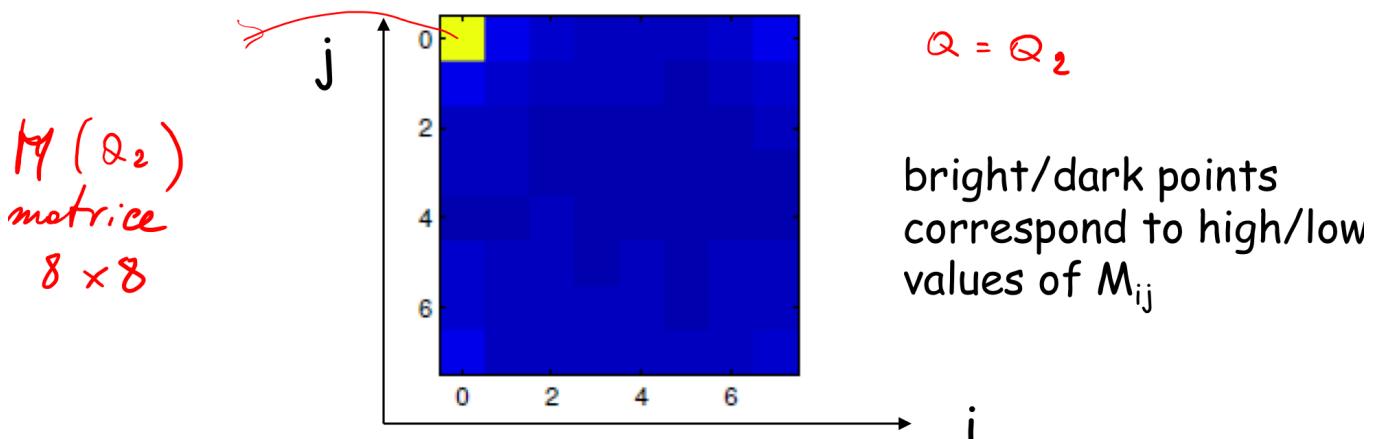
- l'applichiamo ad un blocco DCT
- Prendiamo tutte le componenti DC e le quantizziamo
- calcoliamo il magnitudo delle trasformate di Fourier

Ma come analizziamo tutti gli spostamenti insieme?

Si introduce **Integer Periodicity Map** (IPM) al passo di quantizzazione Q che ci permette di visualizzare i valori della FT per ogni possibile spostamento della griglia.

$$M_{i,j}(Q) = \frac{|f_{i,j}(Q)|}{\sum_{i',j'} |f_{i',j'}(Q)|}$$

con $0 \leq i \leq 7$ e $0 \leq j \leq 7$



Si nota che il picco alla locazione (0,0) si ha quando $Q = Q_2$ e lo shift è di (0,0).

Inoltre si nota che Q_2 è stato estratto dalla matrice di quantizzazione salvata nel file JPEG.

Però non sappiamo ancora Q_1 .

Quindi possiamo utilizzare IPM per ogni Q in un insieme di possibili valori ma diversi da Q_2 e notare:

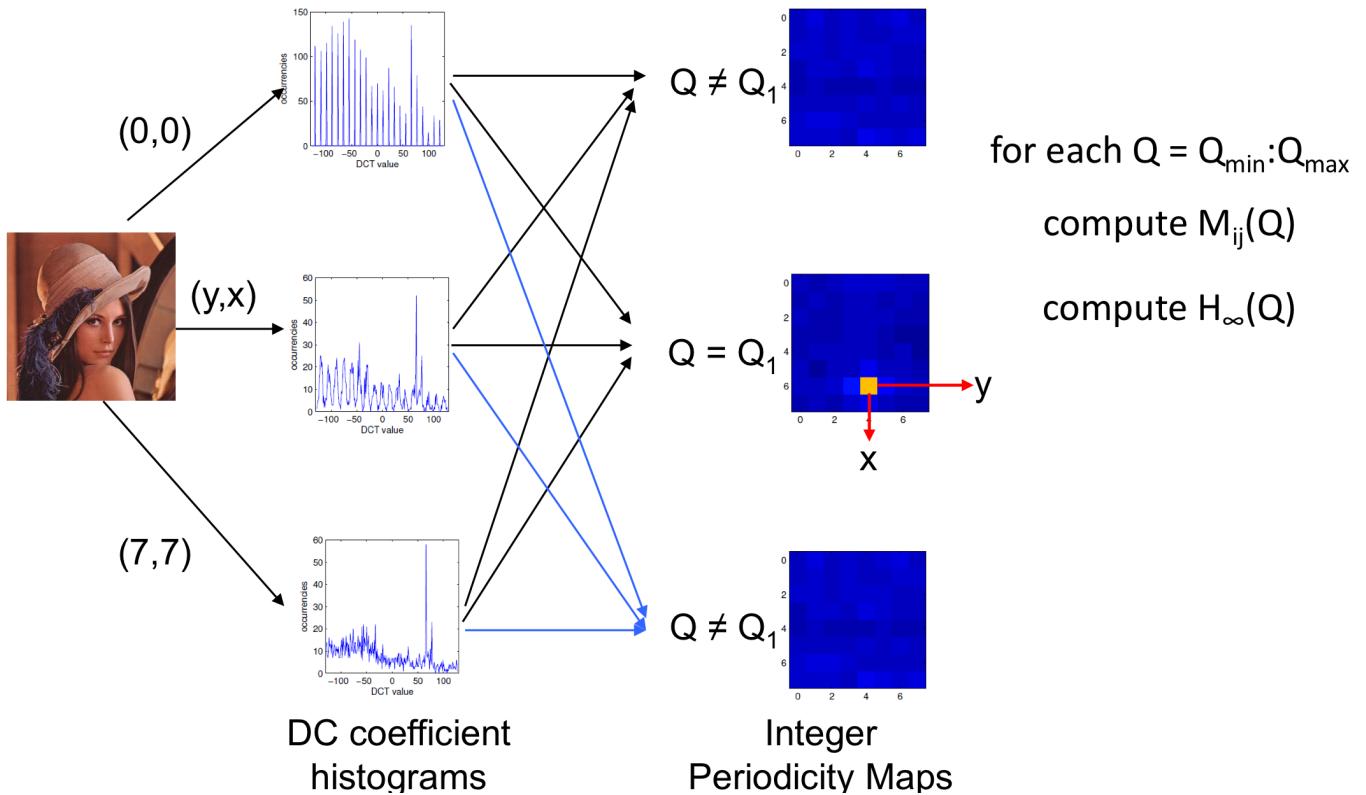
- **Presenza di NA-DJPEG:** troviamo un valore Q_1 tale che $M(Q_1)$ sia molto più grande degli altri alla prima compressione (x,y).
- **Assenza di NA-DJPEG:** $M(Q)$ è uniforme per ogni $Q \neq Q_2$ poichè c'è stata solo una compressione.

Di conseguenza l'algoritmo è:

```

per ogni possibile spostamento (i,j) do:
    esegui D_ij*I2
    esegui hij
    per ogni Q= Qmin : Qmax
        esegui fi,j(Q)
    end for
end for

```



Allora I_2 è classificato NA-DJPEG se:

$$\exists Q \neq Q_2 : H_\infty(Q) < T$$

dove H è il *min-entropy* calcolato in questo modo:

$$H_\infty(Q) = \min_{i,j}(-\log(M_{i,j}(Q)))$$

e (x,y) sono gli spostamenti tali che favoriscono questo risultato.

Si può usare CNN per capire la doppia compressione dei JPEG

JPEG Dimples (fossette)

Esse sono introdotte da molti modelli di fotocamere digitali durante la compressione di un'immagine.

Queste servono per:

- localizzare regioni contraffatte (forged) nell'immagine.
- valutare la compatibilità di un'immagine con una presunta fotocamera modello

Per capire i Dimples bisogna tornare indietro al processo di codifica di un JPEG.



focalizziamoci adesso sulla quantizzazione:

$$\hat{c} = \text{round}(c/q)$$

ma questo non è imposto come standard...

Infatti si può avere anche:

- **floor**: example $\text{floor}(6.7)=6$
- **ceil**: example $\text{ceil}(6.1)=7$

La domanda, quindi, è cosa cambierà nel dominio DCT?

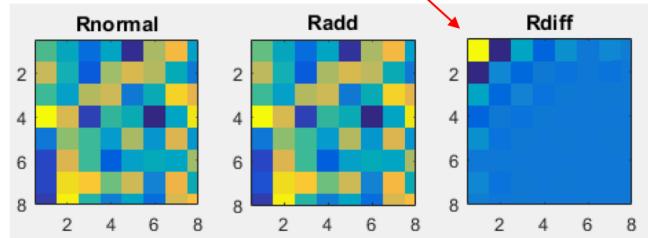
Possiamo simularlo aggiungendo e sottraendo 1 al coefficiente DCT.

```
%Random 8-by-8 integer block
block = randi(255,8,8);
%2D Forward DCT
blockDCT = dct2(block);
%Inverse DCT
Rnormal = idct2(blockDCT);

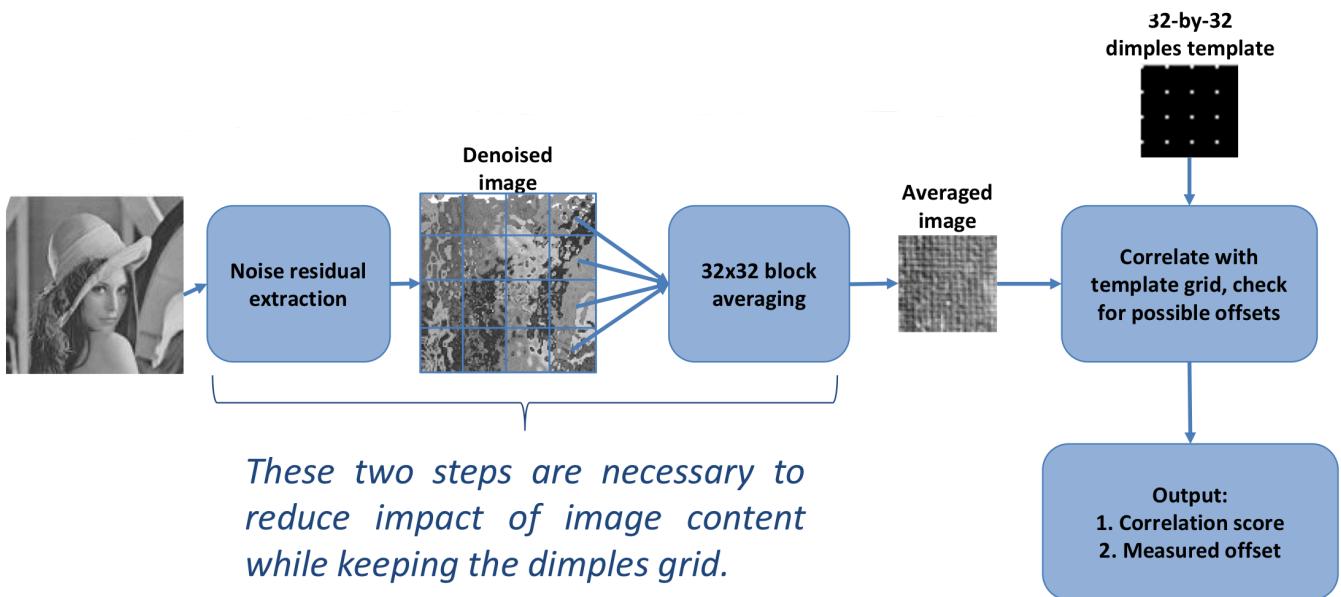
%Now let's add 1 in the DCT domain
Badd = blockDCT + 1;
%Inverse DCT
Radd = idct2(Badd)

%Difference
Rdiff = Radd - Rnormal;
```

The difference concentrates on the upper-left element of the block!



La differenza si concentra in alto a sinistra del blocco.



80

Descriviamo il diagramma sopra:

1. In primo luogo abbiamo l'immagine in input.
2. Viene fatto l'estrazione del rumore sulla foto, facendo un immagine denoised, intesa come una foto dove i dettagli ad alta frequenza(contenuto immagine) vengono scartati.
3. La foto denoised viene poi divisa in una griglia 32x32
4. Viene poi fatta la media dei pixel di ogni blocco di questa griglia.
5. Viene confrontata con il dimples template (i dimples che di solito ci si aspetta).
6. Viene ottenuto un punteggio di correlazione e un offset.

Notiamo che i JPEG Dimples della griglia alcune volte sono shiftate rispetto al pixel in alto a sinistra e questo lo chiamiamo **offset**.

Molte fotocamere non presentano uno shift [0,0] ma alcuni producono offset del tipo [0,7], [7,0] o altri valori. Ed questo è dovuto al processo di generazione interna dell'immagine.

Quando l'immagine digitale è affetta da artefatti di questo tipo *rotando* di 90 gradi non cambierà la forza dei dimples ma **cambierà l'offset**.



Proprietà dei Dimples:

1. Non sono introdotte da software ma sono introdotte dai dispositivi di acquisizione.
2. Se un'immagine viene ricompressa attraverso un software i dimples sopravvivono.
3. La conversione del formato non rimuove i dimples
4. Se l'immagine è scalata o rotata, i dimples diventano inosservabili poiché la sincronizzazione con la griglia del modello viene persa.

Potenzialità dei Dimples:

- Se si ha un'immagine in un formato non compresso è un chiaro indizio di un eventuale compressione JPEG fatto dalla fotocamera (perché il software stesso non introduce i dimples).
- Con un offset strano può essere un indizio per cropping o rotazione.
- Essere coerenti/ incoerenti con il marchio/modello dichiarato nei metadati della foto.

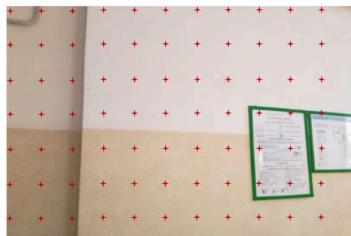
Idea: Quando gli artefatti JPEG Dimples sono globalmente presenti, possiamo usare la griglia dei dimples 8×8 come un segno di autenticità (watermarks) relativo alla sua autenticità:

1. si analizza l'immagine per trovare presenze di Dimples JPEG globali e i suoi possibili offset.
2. Poi, trovare i Dimples Locali (analisi della sliding window) e compara la presenza e l'offset con quelli ottenuti dallo step 1

-se i Dimples sono presenti e l'offset corrisponde allora la regione è autentica.

-altrimenti è stata manipolata.

Host (JPEG having dimples)



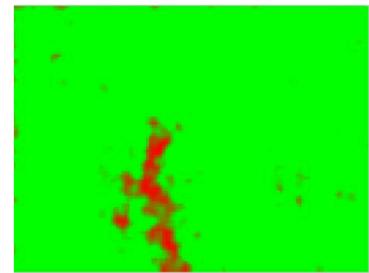
Source (JPEG or not)



Final recompression at decent quality does not destroy dimples

Untouched pixels keep showing the dimples grid

Forged pixels do not (or, they do but with an inconsistent offset!)



JPEG Dimples Map: pixels in red regions do not show dimples and are considered manipulated

93

Limitazione dei Dimples:

- Se l'immagine è globalmente scalata, i dimples sono inosservabili(non funziona sui social media platform).
- I dimples diventano più deboli se si ha una compressione JPEG grossa(<80)(si possono avere falsi allarmi)
- funziona solo su immagini che mostrano artefatti JPEG Dimples.
- su saturati o regioni con texture , dimples sono più difficili da trovare (possibili falsi allarmi)