UNIVERSITÀ
degli STUDI
di CATANIA

Risoluzione della blocchettizzazione in immagini JPEG

Giuseppe Napoli

A.A. 2023/24

Indice

1 Sommario

Il presente progetto si propone di affrontare la problematica della blocchettizzazione insita nella compressione JPEG di immagini, mediante l'implementa- zione di filtri bilaterali, Non-Local Means (NLM) e trasformate wavelet. L'obiettivo principale è migliorare la qualità visiva di immagini soggette a compressione su diversi livelli di qualità.

La relazione si apre con una introduzione al contesto della blocchettizzazione nelle immagini JPEG, seguita da un'esposizione dettagliata dei principi teorici della compressione e della risoluzione mediante l'applicazione di filtri. Vengono poi analizzati approfonditamente i filtri bilaterali, NLM e wavelet, con particolare attenzione alla loro applicabilità nel contesto proposto. Le metriche di qualità adottate, PSNR, SSIM e MSE, sono presentate e spiegate in dettaglio. Il metodo di lavoro è delineato in maniera chiara, comprendendo la selezione delle immagini di test, la compressione su diverse gradazioni di qualità e l'applicazione sequenziale dei filtri prescelti.

I risultati ottenuti riportati evidenziando le variazioni nelle metriche di qualità a seguito dell'applicazione dei filtri. La sezione conclusiva riassume le principali acquisizioni, offre una riflessione sull'efficacia dei filtri e propone possibili sviluppi futuri della ricerca.

2 Introduzione

La compressione JPEG, ampiamente utilizzata per ridurre la dimensione dei file delle immagini, è spesso associata a un fenomeno noto come blocchettizzazione, che può compromettere la qualità visiva dell'immagine risultante. Tale fenomeno si manifesta attraverso la suddivisione dell'immagine in blocchi rettangolari, comportando artefatti visivi evidenti, soprattutto ad alti livelli di compressione. Questo progetto si propone di affrontare concretamente la problematica della blocchettizzazione, ricorrendo all'applicazione sequenziale di filtri bilaterali, Non-Local Means (NLM) e trasformate wavelet. L'obiettivo principale è il miglioramento della qualità visiva di immagini sottoposte a compressione su diversi livelli di qualità.

La comprensione approfondita della compressione JPEG, della blocchettizzazione e delle metodologie di risoluzione mediante l'impiego di specifici filtri riveste un ruolo cruciale nell'ambito della manipolazione delle immagini digitali. Attraverso l'applicazione di filtri bilaterali, NLM e wavelet, si mira a preservare e migliorare i dettagli delle immagini compresse, riducendo significativamente gli artefatti indesiderati.

La presente relazione si articola in sezioni successive, fornendo una panoramica teorica sulla compressione JPEG, la blocchettizzazione e l'applicazione dei filtri selezionati. Il metodo di lavoro è dettagliatamente descritto, delineando il processo di selezione delle immagini di test, la compressione su diversi livelli di qualità e l'applicazione dei filtri proposti. I risultati ottenuti sono esaminati attraverso l'utilizzo di metriche di qualità, quali PSNR, SSIM e MSE, al fine di valutare quantitativamente l'efficacia delle soluzioni implementate.

Il presente progetto, pertanto, si propone di fornire un contributo sostanziale alla comprensione e alla risoluzione della blocchettizzazione in immagini JPEG, offrendo un approccio pratico basato su filtri avanzati e metriche di valutazione della qualità.

3 Cenni teorici

3.1 Compressione JPEG

La compressione JPEG (Joint Photographic Experts Group) è uno dei formati più diffusi per la compressione di immagini digitali. Questo standard è stato sviluppato dalla Joint Photographic Experts Group, un comitato di esperti che ha lavorato su una serie di standard per la codifica di immagini fisse e in movimento.

La compressione JPEG è comunemente utilizzata per ridurre la dimensione dei file delle immagini senza compromettere eccessivamente la qualità visiva. Questo formato è particolarmente adatto per immagini fotografiche e altre immagini complesse con una vasta gamma di colori e dettagli.

Ecco alcuni aspetti chiave della compressione JPEG:

- Compressione Lossy: JPEG è un algoritmo di compressione lossy, il che significa che riduce le dimensioni del file sacrificando alcune informazioni dell'immagine. Questa perdita di informazioni può causare la generazione di artefatti visivi, soprattutto a livelli di compressione più elevati.
- Blocchettizzazione: Durante la compressione JPEG, l'immagine è divisa in blocchi rettangolari di 8x8 pixel. Questa suddivisione in blocchi è responsabile della blocchettizzazione, uno degli effetti collaterali visivi della compressione JPEG.
- Coefficienti di Frequenza: JPEG utilizza la trasformata discreta del coseno (DCT) per rappresentare le frequenze dell'immagine. Questo processo consente di concentrare la maggior parte dell'energia dell'immagine nei coefficienti di bassa frequenza, consentendo una maggiore compressione.
- Livelli di Qualità: Gli algoritmi JPEG consentono di regolare il livello di qualità della compressione. Maggiore è il livello di qualità, minore è la compressione e minore è la perdita di dettagli, ma si avrà anche un file di dimensioni più grande.
- Utilizzo Diffuso: JPEG è ampiamente utilizzato per la condivisione di immagini su Internet, in fotografia digitale, e in molte applicazioni che richiedono la visualizzazione e la condivisione di immagini.

Sebbene la compressione JPEG sia efficace per ridurre le dimensioni dei file, è importante bilanciare la compressione con la qualità, specialmente quando la fedeltà dell'immagine è fondamentale. Inoltre, in contesti in cui è necessaria una compressione senza perdita di informazioni, altri formati come PNG o TIFF possono essere preferibili rispetto a JPEG.

3.2 Risoluzione mediante filtri

La risoluzione mediante filtri applicati dopo la compressione rappresenta un approccio avanzato finalizzato a mitigare gli effetti indesiderati derivanti dalla compressione di immagini, in particolare dall'utilizzo di algoritmi lossy come JPEG. L'obiettivo principale di questa fase di risoluzione è migliorare la qualità visiva dell'immagine compressa, riducendo o eliminando artefatti e blocchettizzazione.

Dopo la compressione di un'immagine, specialmente in presenza di elevati livelli di compressione, possono emergere artefatti visivi, perdita di dettagli e altri difetti che influiscono negativamente sulla qualità dell'immagine. L'approccio della risoluzione mediante filtri prevede l'applicazione di filtri specifici per attenuare tali imperfezioni e ripristinare la fedeltà visiva dell'immagine originale.

3.2.1 Introduzione ai filtri Bilaterali

Il filtro bilaterale è un tipo di filtro non lineare che tiene conto sia della differenza di intensità tra i pixel, sia della loro distanza spaziale. Questo significa che, durante l'applicazione del filtro, pixel simili in termini di intensità e vicini spazialmente saranno trattati in modo simile. Ciò consente di preservare i dettagli dell'immagine mentre si riducono rumore ed artefatti. L'utilizzo di filtri bilaterali può essere particolarmente efficace nella correzione di distorsioni introdotte durante la compressione JPEG.

3.2.2 Introduzione al filtro NLM

Il filtro Non-Local Means (NLM) è un approccio basato sulla similarità non locale tra regioni dell'immagine. A differenza dei filtri tradizionali, NLM considera non solo la vicinanza spaziale, ma anche la similarità di pixel in diverse regioni dell'immagine. Questo metodo è efficace nel mantenere dettagli sottili e preservare le caratteristiche dell'immagine originale, rendendolo un filtro adatto per il denoising e la correzione di artefatti.

3.2.3 Introduzione alle trasformate Wavelet

Le trasformate wavelet sono una classe di filtri utilizzati per rappresentare un'immagine in termini di onde di diverse frequenze e posizioni. L'applicazione delle trasformate wavelet consente di separare l'immagine in componenti di bassa e alta frequenza, consentendo una modifica mirata e la gestione delle informazioni in modo più efficiente. Questo può essere utile nella compensazione della perdita di dettagli dovuta alla compressione.

In sintesi, l'utilizzo combinato di filtri bilaterali, NLM e trasformate wavelet può contribuire significativamente alla risoluzione dei problemi introdotti dalla compressione JPEG, migliorando la qualità visiva dell'immagine compressa e restituendo dettagli importanti precedentemente persi.

3.3 Filtri

Dopo averli introdotti nel paragrafo precedente, vi è di seguito un'analisi più approfondita dei filtri in questione

3.3.1 Filtri bilaterali

I filtri bilaterali operano sulla base di due criteri principali: la differenza di intensità e la distanza spaziale tra i pixel. L'idea fondamentale è preservare i dettagli dell'immagine, evitando al contempo la fusione di regioni con intensità simili ma appartenenti a diverse strutture. Quando il filtro bilaterale viene applicato a un'immagine, esso calcola una media ponderata dei valori dei pixel, attribuendo pesi diversi in base alla distanza spaziale e alla differenza di intensità. Questo permette di ottenere un effetto di smoothing efficace mantenendo i dettagli significativi dell'immagine.
La funzione di filtro bilaterale può essere rappresentata come:

$$\text{I'}(x,y) = \frac{1}{W_p} \sum_i \sum_j I(i,j) \cdot G_s(||x-i||) \cdot G_r(|I(x,y) - I(i,j)|) \tag{1}$$

Dove:

- $I'(x,y)$ è il valore del pixel filtrato
- $I(x,y)$ è il valore del pixel originale
- $G_s(||x-i||)$ è la funzione di Gauss che valuta la distanza spaziale tra i pixel

- $G_r(|I(x,y) - I(i,j)|)$ è la funzione di Gauss che valuta la differenza di intensità tra i pixel
- W_p è il peso normalizzato

Approfondendo i parametri utilizzati dal filtro abbiamo:

- Raggio Spaziale (σ_s): Indica quanto lontano il filtro considera i pixel come "vicini" in base alla loro posizione spaziale.
- Raggio Intensità (σ_r): Indica quanto due pixel possono differire in intensità senza essere considerati "diversi". Un valore più alto consente di considerare un intervallo più ampio di intensità.
- Funzione di Gauss per la Distanza Spaziale ($G_s(||x - i||)$): Questa funzione valuta quanto due pixel sono distanti nello spazio. È una funzione di Gauss centrata sulla distanza tra i pixel x e i.
- Funzione di Gauss per la Differenza di Intensità $G_r(|I(x,y) - I(i,j)|)$: Questa funzione valuta la differenza di intensità tra i pixel x,y e i,j. È una funzione di Gauss centrata sulla differenza di intensità.

L'applicazione del filtro bilaterale permette di ottenere un'immagine filtrata che preserva i dettagli significativi mentre riduce il rumore, fornendo un contributo rilevante alla risoluzione della blocchettizzazione in immagini compresse in JPEG .

3.3.2 NLM (Non-Local Means)

Il filtro Non-Local Means (NLM) è un metodo avanzato per il denoising delle immagini che considera la similarità non locale tra regioni dell'immagine, andando oltre il concetto di vicinanza spaziale. Il filtro NLM è particolarmente utile per la correzione di immagini affette da rumore, inclusi gli artefatti introdotti durante la compressione JPEG.

Concetto di Similarità Non Locale: contrariamente ai filtri tradizionali che si basano principalmente sulla vicinanza spaziale, il NLM valuta la somiglianza tra le regioni dell'immagine, consentendo di preservare dettagli significativi e di effettuare una correzione più accurata del rumore. L'idea principale è che pixel simili in differenti regioni dell'immagine contribuiranno in modo più rilevante alla ricostruzione

dell'immagine originale. La formula generale per il calcolo di un pixel nell'immagine filtrata è la seguente:

$$\text{I'}(x, y) = \frac{1}{W_p} \sum_i \sum_j I(i, j) \cdot w(i, j, x, y) \tag{2}$$

Dove:

- $I'(x, y)$ è il valore del pixel filtrato
- $I(x, y)$ è il valore del pixel originale
- W_p è il peso normalizzato

Il peso w(i, j, x, y) è calcolato come segue:

$$\text{w}(i, j, x, y) = \frac{1}{Z(x, y)} \cdot e^{-\frac{1}{h^2} \sum_k \sum_l (I(k,l) - I(x+k-i, y+l-j))^2} \tag{3}$$

Dove:

- Z(x, y) è il termine di normazillzazione
- h è un parametro che regola la sensibilità della differenza di intensità

Inoltre, il filtro NLM fa uso dei seguenti parametri:

- Raggio di ricerca (r): Specifica la distanza massima da esplorare per trovare regioni simili. Questo parametro regola l'ampiezza dell'area considerata per la somiglianza non locale.
- Raggio di Similarità (h): Regola la sensibilità alla differenza di intensità tra i pixel. Un valore più alto consente di includere regioni più dissimili nella somiglianza.

L'applicazione del filtro NLM offre una maggiore adattabilità rispetto ai filtri tradizionali, preservando dettagli sottili e mitigando efficacemente il rumore, inclusi gli artefatti introdotti dalla compressione JPEG.

3.3.3 Wavelet

Le trasformate wavelet sono una classe di tecniche matematiche utilizzate per decomporre un'immagine in componenti di diverse frequenze e risoluzioni, consentendo una rappresentazione più efficace degli elementi dell'immagine. Nella risoluzione della blocchettizzazione post-compressione JPEG, l'applica- zione delle trasformate wavelet è essenziale per identificare e correggere dettagli specifici a diverse scale di frequenza.

La trasformata wavelet è solitamente applicata attraverso una serie di iterazioni di filtraggio e decimazione. In termini matematici, la trasformata wavelet 1D di una funzione f(x) può essere espressa come:

$$\mathrm{W}f(a,b) = \int_{-\infty}^{\infty} f(x) \cdot \psi_{a,b}(x) dx \tag{4}$$

Dove:

- a rappresenta il fattore di scala o la dilatazione.
- b indica la traslazione.
- $\psi_{a,b}(x)$ è la funzione madre (wavelet) trasformata, modulata per la dilatazione e la traslazione.

Decomposizione 2D: La trasformata wavelet 2D applicata a un'immagine I(x,y) coinvolge operazioni simili su entrambe le dimensioni. La decomposizione 2D può essere espressa come:

$$\mathrm{W}f(a,b,c,d) = \int\int I(x,y) \cdot \psi_{a,b}(x) \cdot \phi_{c,d}(y) dx dy \tag{5}$$

Dove $\psi_{a,b}(x)$ e $\phi_{c,d}(y)$ sono le funzioni wavelet nelle direzioni x e y rispettivamente. Esistono diverse tipologie di wavelet, come ad esempio la wavelet di Haar, la wavelet di Daubechies, la wavelet di Morlet, o la wavelet biortogonale. Ogni tipo di wavelet ha caratteristiche uniche e può essere selezionato in base alle specifiche esigenze dell'applicazione.

Le trasformate wavelet sono ampiamente utilizzate nella risoluzione della blocchettizzazione in immagini JPEG. Poiché le onde ad alta frequenza possono rappresentare bordi e dettagli fini, la decomposizione wavelet consente di isolare questi elementi.

Inoltre, la capacità di separare le componenti di bassa e alta frequenza rende possibile l'applicazione di correzioni mirate, contribuendo così a migliorare la qualità dell'immagine compressa.

In sintesi, l'applicazione delle trasformate wavelet nella risoluzione della blocchettizzazione offre un approccio efficace per analizzare e correggere dettagli a diverse scale di frequenza, consentendo una migliore preservazione dei dettagli e un miglioramento complessivo della qualità dell'immagine.

L'applicazione di filtri bilaterali, Non-Local Means e trasformate wavelet rappresenta un approccio multidimensionale per affrontare i problemi derivanti dalla compressione JPEG, mitigando la blocchettizzazione, preservando dettagli importanti e migliorando la qualità globale dell'immagine compressa.

3.4 Metriche di qualità

Le metriche di qualità delle immagini sono strumenti indispensabili per valutare oggettivamente la fedeltà e l'accuratezza di un'immagine rispetto al suo originale o a uno standard di riferimento. Esse forniscono misure quantitative che consentono di valutare le prestazioni di algoritmi di compressione, miglioramento, filtraggio o qualsiasi processo di manipolazione di immagini.

Le metriche di qualità risultano fondamentali poichè dotate di:

- Valutazione Obiettiva: Permettono una valutazione oggettiva delle prestazioni di algoritmi e tecniche di elaborazione delle immagini. Ciò è cruciale, specialmente in contesti in cui è necessario garantire la fedeltà dell'immagine, come nell'ambito medico, industriale o nelle applicazioni di visione artificiale.

- Ottimizzazione degli Algoritmi: Forniscono un mezzo per ottimizzare algoritmi, stabilire parametri e confrontare diverse implementazioni. L'utilizzo di metriche di qualità consente di raffinare e perfezionare algoritmi per ottenere risultati visivi migliori.

- Standardizzazione: Nell'industria e nella ricerca, l'utilizzo di metriche di qualità standardizzate permette una comunicazione più chiara e universale sulle prestazioni di algoritmi e sistemi, agevolando la riproducibilità delle ricerche e delle sperimentazioni.

Per effettuare valutazioni sul lavoro svolto, si è deciso di far uso di tre diverse metriche di qualità:

- Mean Squared Error (MSE): Valuta la differenza quadratica media tra i valori effettivi e quelli predetti o stimati. Viene spesso utilizzato per confrontare immagini pixel per pixel.
- Structural Similarity Index (SSI o SSIM): Misura la similarità strutturale tra due immagini, tenendo conto di luminanza, contrasto e struttura. È più adatto a riflettere la percezione umana.
- Peak Signal-to-Noise Ratio (PSNR): Misura il rapporto tra il massimo valore possibile di un segnale (senza errore) e la potenza del rumore presente nel segnale. È spesso utilizzato in compressione e trasmissione di immagini.

3.4.1 Mean Squared Error (MSE)

Mean Squared Error (Errore Quadratico Medio, MSE) è una metrica utilizzata per valutare la discrepanza tra due insiemi di dati, fornendo una misura quantitativa della differenza quadratica media tra i valori effettivi e quelli predetti o stimati. La formula generale del MSE tra due sequenze di valori X e Y, ciascuna con N elementi, è espressa come segue:

$$\mathrm{M}SE(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(X_i, Y_i)^2 \tag{6}$$

Dove:

- X_i è l'i-esimo elemento della sequenza X
- Y_i è l'i-esimo elemento della sequenza Y
- N rappresenta la lunghezza delle sequenze

Il MSE calcola la somma dei quadrati delle differenze tra ciascun elemento corrispondente delle due sequenze. Un valore più basso indica una maggiore concordanza tra le sequenze. Gli errori sono elevati al quadrato, accentuando pertanto le discrepanze più grandi. Ciò implica che il MSE è particolarmente sensibile agli errori più significativi, rendendolo utile nel penalizzare le differenze maggiori. L'obiettivo è quindi quello di ridurre il MSE il più possibile. Un MSE pari a zero indicerebbe una perfetta corrispondenza tra le sequenze, mentre valori più alti indicano una maggiore discrepanza.

Nel contesto della valutazione di immagini, il MSE può essere utilizzato per confrontare pixel per pixel le immagini originali e quelle ripristinate o manipolate. Ad esempio, nell'ambito della risoluzione della blocchettizzazione in immagini JPEG, il MSE può essere calcolato tra l'immagine originale e quella processata con tecniche di correzione.

Nonostante la sua utilità, il MSE presenta alcune limitazioni:

- Sensibilità agli Outlier: Essendo basato su errori quadratici, il MSE può essere fortemente influenzato da outlier.
- Assenza di Contesto Visivo: Non tiene conto delle caratteristiche visive percepite dagli esseri umani, focalizzandosi esclusivamente sulla differenza numerica.

In conclusione, il Mean Squared Error è una metrica fondamentale per la valutazione quantitativa della discrepanza tra insiemi di dati, ma è importante considerare le sue limitazioni, soprattutto quando si valutano immagini in cui il contesto visivo è essenziale.

3.4.2 Structural Similarity Index (SSIM)

Il Structural Similarity Index (Indice di Similarità Strutturale, SSIM) è una metrica di qualità delle immagini progettata per valutare la somiglianza strutturale tra due immagini. A differenza di metriche più semplici come il Mean Squared Error (MSE), l'SSIM tiene conto di aspetti più complessi della percezione visiva umana, includendo elementi come la luminanza, il contrasto e la struttura.

L'SSIM è calcolato utilizzando tre componenti principali: luminanza (l), contrasto (c), e struttura (s). La formula generale è espressa come segue:

$$\mathrm{S}SIM(X,Y) = [l(X,Y)]^{\alpha} \cdot [c(X,Y)]^{\beta} \cdot [s(X,Y)]^{\gamma} \tag{7}$$

Dove:

- X e Y sono le due immagini confrontate
- α, β, γ sono parametri che controllano l'importanza relativa delle componenti
- $l(X,Y)$, $c(X,Y)$ e $s(X,Y)$ sono le componenti di luminanza, contrasto e struttura rispettivamente

Le componenti sono definite come:

$$\mathrm{l}(X,Y) = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1} \tag{8}$$

$$\mathrm{c}(X,Y) = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2} \tag{9}$$

$$\mathrm{s}(X,Y) = \frac{\sigma_{XY} + C_3}{\sigma_X + \sigma_Y + C_3} \tag{10}$$

Dove:

- μ_X, μ_Y sono le medie di X e Y
- $\sigma_X e \sigma_Y$ sono le deviazioni standard di X e Y
- σ_{XY} è la covarianza tra X e Y
- C_1, C_2, C_3 sono costanti che stabilizzano il denominatore per evitare divisioni per zero

Calcolando le singole componenti otteniamo:

- Luminanza (l): Riflette la differenza di luminanza media tra le due immagini. Valori più alti indicano maggiore somiglianza.
- Contrasto (c): Misura la differenza tra le deviazioni standard delle due immagini. Valori più alti indicano maggiore contrasto.
- Struttura (s): Riflette la covarianza normalizzata tra le due immagini. Valori più alti indicano una struttura più simile.

Il valore dell'SSIM è compreso tra -1 e 1, dove 1 indica una corrispondenza perfetta tra le immagini. Un valore di 0 indica completa dissimilarità, mentre valori negativi indicano una discrepanza ancora maggiore.

L'SSIM è ampiamente utilizzato nella valutazione di algoritmi di compressione, riduzione del rumore, e qualsiasi processo che coinvolge la manipolazione delle immagini. La sua capacità di incorporare aspetti visivi più complessi fa sì che l'SSIM sia una metrica più accurata rispetto a misure più semplici come il MSE quando si tratta di riflettere la percezione umana della qualità delle immagini.

3.4.3 Peak Signal-to-Noise Ratio (PSNR)

Il Peak Signal-to-Noise Ratio (Rapporto Picco Segnale-Rumore, PSNR) è una metrica utilizzata per valutare la qualità di un'immagine confrontando il segnale dell'immagine originale con il rumore introdotto da processi di compressione o manipolazione. Il PSNR è spesso utilizzato nell'ambito della compressione delle immagini per valutare quanto il processo di compressione ha alterato l'immagine rispetto all'originale.

La formula generale del PSNR è espressa come segue:

$$\mathrm{P}SNR = 10 \cdot \log_{10}(\frac{S^2}{MSE}) \tag{11}$$

Dove:

- S è il valore massimo possibile del segnale (solitamente $2^{bit-depth} - 1$ per immagini a scala di grigi o colori).
- MSE è l'Errore Quadratico Medio (Mean Squared Error), calcolato tra l'immagine originale e quella compressa o manipolata.

Il PSNR rappresenta il rapporto tra il massimo possibile segnale (S) e l'errore quadratico medio (MSE). Un PSNR più alto indica un minore livello di distorsione. La scala logaritmica è utilizzata per riflettere la percezione umana della qualità dell'immagine. Un incremento di 10 unità nel PSNR corrisponde a una percezione di miglioramento della qualità dell'immagine.

Il PSNR presenta due limitazioni:

- Sensibilità a Piccole Variazioni: Il PSNR è sensibile alle più piccole variazioni tra l'immagine originale e quella compressa. Questa sensibilità potrebbe portare a valutazioni meno accurate quando la distorsione non è visibile all'occhio umano.
- Assunzioni Gaussiane: La metrica assume che il rumore dell'immagine segua una distribuzione gaussiana. In contesti in cui questa assunzione non è valida, il PSNR potrebbe non fornire una rappresentazione accurata della qualità dell'immagine.

I campi di applicazione del PSNR sono i seguenti:

- Compressione delle Immagini: È comunemente utilizzato per valutare gli effetti della compressione delle immagini, ad esempio in formati come JPEG.

- Video Compression: Nell'ambito della compressione video, il PSNR viene utilizzato per valutare la qualità dei video compressi rispetto agli originali.
- Misurazione del Rumore: Può essere utilizzato anche per misurare il rumore in segnali o immagini.

In conclusione, il PSNR è una metrica comune e utile per valutare la qualità delle immagini, ma è importante interpretare i suoi risultati con consapevolezza delle sue limitazioni e considerare l'utilizzo di metriche più avanzate, come l'Indice di Similarità Strutturale (SSIM), per ottenere una valutazione più completa della percezione umana della qualità dell'immagine.

3.4.4 Considerazioni Finali

Le metriche di qualità delle immagini forniscono una base cruciale per valutare le prestazioni di algoritmi e modelli. Tuttavia, è essenziale considerare la natura complessa della percezione umana e utilizzare un insieme diversificato di metriche per ottenere una valutazione completa. L'accurata comprensione e applicazione di queste metriche sono fondamentali per il successo e l'affidabilità di progetti che coinvolgono il trattamento delle immagini digitali.

4 Metodo di lavoro

Il progetto ha come obiettivo principale la valutazione delle prestazioni di diverse tecniche di restauro applicate a un campione eterogeneo di immagini, caratterizzato da diversità sia nei soggetti rappresentati (primo piano, pianta, montagna, stanza, ecc.) che nella colorazione (colori caldi, freddi, scala di grigi). Per analizzare nel dettaglio il lavoro svolto, si è deciso di suddividere il progetto in quattro fasi, di seguito elencate.

4.1 Fasi del Progetto:

1. Selezione del campione: È stata effettuata a priori una raccolta di immagini in grado di analizzare la bontà degli algoritmi su elementi diversi fra loro. Questo campione include immagini con varie caratteristiche di soggetto e colorazione.

2. Compressione in JPEG: Ogni immagine del campione è stata compressa utilizzando il formato JPEG. L'operazione di compressione è stata eseguita con diversi livelli di compressione, regolati dal parametro "quality" che varia da 0 a 100.

3. Restauro mediante filtri: Le immagini compresse sono state sottoposte a un processo di restauro mediante l'applicazione di tre filtri distinti: il filtro bilaterale, il filtro Non-Local Means (NLM) e le trasformate wavelet.

4. Valutazione tramite metriche di qualità: Al fine di valutare oggettivamente i risultati del restauro, sono state applicate tre metriche di qualità:

    - MSE (Mean Squared Error): Misura la discrepanza pixel per pixel tra l'immagine originale e quella restaurata.
    - SSIM (Structural Similarity Index): Valuta la somiglianza strutturale tra le immagini considerando luminanza, contrasto e struttura.
    - PSNR (Peak Signal-to-Noise Ratio): Calcola il rapporto tra il massimo segnale possibile e l'errore quadratico medio.

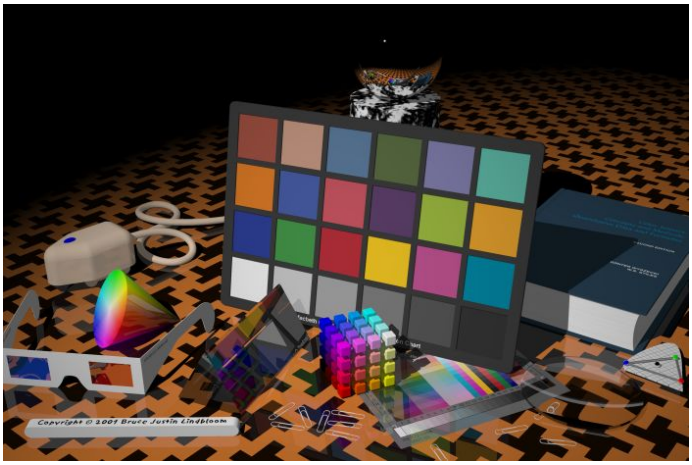
Per ogni fase del progetto, sono stati utilizzati script specifici per eseguire le operazioni descritte, ognuno di essi sarà presentato nella relativa sezione.

4.1.1 Selezione del campione

Fase di approviggionamento delle risorse da fornire in input. Si è scelto di effettuare questa analisi su di un pool di immagini abbastanza eterogneneo da mettere in risalto le capacità dei singoli algoritmi. Nello specifico le immagini prese in considerazione sono:

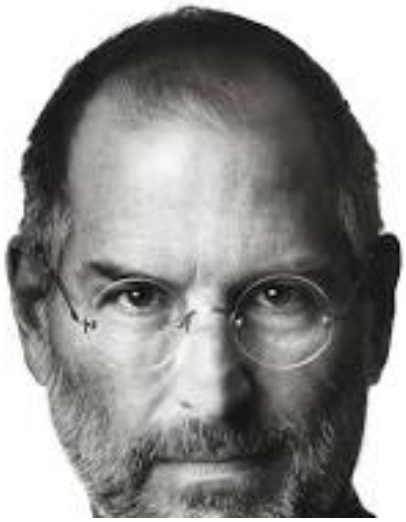
- *bedroom*: Stanza da letto con prevalenza di colori caldi e vari dettagli (piccoli oggetti e pattern sui cuscini);

- *colors*: Palette di colori, caratterizzata da un'ampia varietà cromatica ma con molti elementi quadrati/rettangolari e porzioni abbastanza significative a tinta unita;



- *everest*: Soggetto imponente (montagna innevata) e prevalenza di colori freddi. Pochi dettagli;

- *jobs*: Fotografia di un primo piano di Steve Jobs. In scala di grigi e con pochi dettagli, apparentemente più semplice da restaurare;



- *living*: Soggiorno caratterizzato dalla presenza di ombre dovute alla presenza di una finestra (ad ovest, non presente nello scatto) e raffigurante, in lontananza, un punto luce molto luminoso, rappresentato da una finestra;

- *mountain*: Altra montagna ma, a differenza della precedente, caratterizzata da una luce calda probabilmente dovuta al tramonto del sole;

- *rose*: Scatto ad una rosa, con il soggetto in primo piano di colorazione calda ed uno sfondo poco dettagliato ma con prevalenza del colore verde (freddo);

4.1.2 Compressione in JPEG

Successivamente all'accurata scelta di immagini da sottoporre ai test, è stato necessario effettuare la compressione in JPEG per creare gli artefatti che ci si è posti il problema di rimuovere. Per effettuare tale compressione è stato utilizzato il seguente script:

```python
from PIL import Image


def compress_jpeg(input_image_path, output_image_path, quality):
    # Apri l'immagine
    original_image = Image.open(input_image_path)

    # Salva l'immagine compressa in formato JPEG
    original_image.save(output_image_path, "JPEG", quality=quality)
```

Come si può evincere, è risultato necessario far uso di un parametro *quality*, che rientra in un range da 0 a 100 e che indica la percentuale qualità che si desidera preservare. Naturalmente, questo valore risulterà inversamente proporzionale al peso dell'immagine risultante ma, non essendo la bontà della compressione effettuata il cardine dello studio svolto, esso risulta poco rilevante. Per ciò che risulta invece fondamentale ai fini del progetto, va ricordato che minore è il valore di *quantity* maggiore sarà la quantità di artefatti introdotti, rendendo più arduo il compito di restaurare l'immagine. Di seguito alcuni esempi della compressione eseguita:

Immagine *bedroom* compressa con $quality = 10$

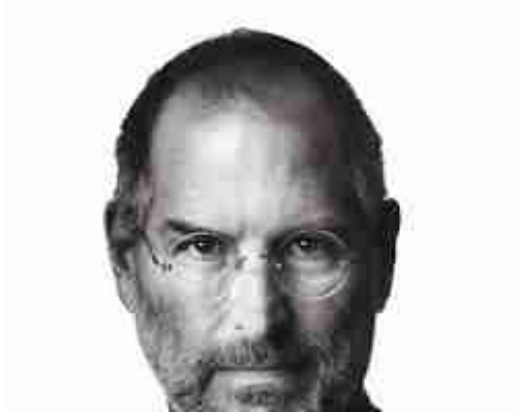


Immagine *jobs* compressa con $quality = 20$

Immagine *living* compressa con $quality = 20$

Immagine *colors* compressa con $quality = 10$

4.1.3 Restauro mediante filtri

Come più volte anticipato, si è scelto di valutare la bontà di tre diverse tecniche di restauro per rimuovere il problema della blocchettizzazione causato in fase di compressione. Di seguito i dettagli pratici dell'applicazione dei filtri:

- Filtro bilaterale: In questo caso si è scelto di utilizzare la funzione $bilateralFilter$ presente nella libreria OpenCV di python. Nello specifico, i parametri d, $sigma\_color$ e $sigma\_space$ sono stati impostati di default per proporre un'applicazione univoca del filtro. Si lascia a sviluppi futuri la possibilià di confrontare l'applicazione del filtro con diverse combinazioni

```python
import cv2


def bilateral_filter(image_path, output_path, d=5, sigma_color=15, sigma_space=15):
    # Leggi l'immagine
    img = cv2.imread(image_path)

    # Applica il filtro bilaterale
    filtered_img = cv2.bilateralFilter(img, d, sigma_color, sigma_space)

    # Salva l'immagine filtrata
    cv2.imwrite(output_path, filtered_img)
```

- NLM: Nel caso del filtro NLM, sono state prese in considerazione due diverse implementazioni del filtro. Alla fine però, grazie ai risultati ottenuti, si è optato per utilizzare il primo (nlm_filter) e scartare il secondo. La differenza sostanziale dei due sta nell'utilizzo del $GaussianBlur$, che si è rivelato fondamnetale per ottenere i risultati migliori dell'intero progetto. Anche in questo caso i parametri h_color $sigma\_color$ e $sigma\_space$ sono stati impostati a priori e ciò apre la possibilità di ampliare le casistiche di studio.

- Wavelet: Come citato nella sezione teorica dell'elaborato, esistono vari tipi di trasformate wavelet. In questo caso si è scelto di prendere in considerazione la trasformata wavelet di Haar, implementata così come si può evincere dallo script sottostante.

```python
import cv2


def nlm_filter(image_path, save_path, h_value=10, sigma_color=1, sigma_space=1):
    # Leggi l'immagine
    img = cv2.imread(image_path)

    # Applica il filtro Non-Local Means (NLM) separatamente su ciascun canale di colore
    b, g, r = cv2.split(img)

    b_denoised = cv2.fastNlMeansDenoising(b, None, h=h_value)
    g_denoised = cv2.fastNlMeansDenoising(g, None, h=h_value)
    r_denoised = cv2.fastNlMeansDenoising(r, None, h=h_value)

    # Combina i canali denoised
    denoised_img = cv2.merge([b_denoised, g_denoised, r_denoised])

    # Applica l'operazione di smoothing per rimuovere la blocchettizzazione
    smoothed_img = cv2.GaussianBlur(denoised_img, (5, 5), sigma_color, sigma_space)

    # Salva l'immagine risultante
    cv2.imwrite(save_path, smoothed_img)

def nlm_filter2(image_path, save_path, h_value=10):
    # Leggi l'immagine
    img = cv2.imread(image_path)

    # Applica il filtro Non-Local Means (NLM) all'intera immagine
    denoised_img = cv2.fastNlMeansDenoisingColored(img, None, h=h_value)

    # Salva l'immagine risultante
    cv2.imwrite(save_path, denoised_img)
```

```python
import cv2
import pywt
import numpy as np


def wavelet_filter(image_path, output_path):
    # Carica l'immagine JPEG a colori
    img = cv2.imread(image_path)

    # Split dei canali RGB
    b, g, r = cv2.split(img)

    # Applica la trasformata wavelet di Haar a ciascun canale
    coeffs_b = pywt.dwt2(b, "haar")
    coeffs_g = pywt.dwt2(g, "haar")
    coeffs_r = pywt.dwt2(r, "haar")

    # Modifica i coefficienti a tua discrezione, ad esempio, quantizzando in modo più aggressivo
    coeffs_b_modified = [np.round(c) for c in coeffs_b]
    coeffs_g_modified = [np.round(c) for c in coeffs_g]
    coeffs_r_modified = [np.round(c) for c in coeffs_r]

    # Ricostruisci i canali dalla trasformata wavelet modificata
    b_wavelet = pywt.idwt2(coeffs_b_modified, "haar")
    g_wavelet = pywt.idwt2(coeffs_g_modified, "haar")
    r_wavelet = pywt.idwt2(coeffs_r_modified, "haar")

    # Unisci i canali per ottenere l'immagine finale
    img_wavelet = cv2.merge(
        (
            b_wavelet.astype(np.uint8),
            g_wavelet.astype(np.uint8),
            r_wavelet.astype(np.uint8),
        )
    )

    # Assicurati che l'immagine risultante abbia le stesse dimensioni di quella di input
    img_wavelet = img_wavelet[: img.shape[0], : img.shape[1], :]

    # Salva l'immagine risultante
    cv2.imwrite(output_path, img_wavelet)
```

A conclusione del paragrafo ecco alcuni risultati comparati:

Immagine *jobs* compressa con $quality = 10$

Immagine *jobs* compressa con $quality = 10$ e restaurata con nlm

Immagine *jobs* compressa con $quality = 10$ e restaurata con filtro bilaterale

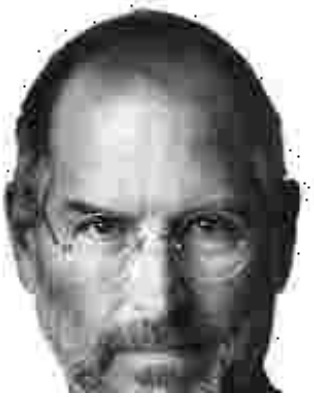


Immagine *jobs* compressa con $quality = 10$ e restaurata con trasformata di wavelet

4.1.4 Valutazione tramite metriche di qualità

Come ultima fase del lavoro svolto, si è deciso di implementare tre diverse metriche di qualità per valutare i risultati ottenuti dalla fase di restauro a seguito dell'applicazione dei filtri:

- MSE (Mean Squared Error)

```python
def calculate_mse(img1_path, img2_path):
    img1 = np.array(Image.open(img1_path))
    img2 = np.array(Image.open(img2_path))

    if img1 is None or img2 is None:
        print("Impossibile leggere una delle immagini.")
        return None

    if img1.shape != img2.shape:
        raise ValueError("Le dimensioni delle immagini non corrispondono.")

    # Calcola il Mean Squared Error (MSE)
    mse = np.sum((img1 - img2) ** 2) / float(img1.shape[0] * img1.shape[1])

    return mse
```

- SSIM (Structural Similarity Index)

```python
def calculate_ssim(img1_path, img2_path):
    img1 = cv2.imread(img1_path)
    img2 = cv2.imread(img2_path)

    if img1 is None or img2 is None:
        print("Impossibile leggere una delle immagini.")
        return None

    if img1.shape != img2.shape:
        print("Le dimensioni delle immagini non corrispondono.")
        return None

    # Converti le immagini in scala di grigi
    img1_gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    img2_gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

    # Calcola l'indice SSIM
    indice_ssim, _ = ssim(img1_gray, img2_gray, full=True)

    return indice_ssim
```

- PSNR (Peak Signal-to-Noise Ratio)

```python
def calculate_psnr(img1_path, img2_path):
    img1 = cv2.imread(img1_path)
    img2 = cv2.imread(img2_path)

    if img1 is None or img2 is None:
        print("Impossibile leggere una delle immagini.")
        return None

    if img1.shape != img2.shape:
        print("Le dimensioni delle immagini non corrispondono.")
        return None

    # Calcola il PSNR
    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return float("inf")
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))

    return psnr
```

5 Risultati ottenuti

Table 1: Analisi delle immagini con diverse percentuali di compressione - MSE

Immagine	Compressione	MSE			
		Senza Filtro	Bilaterale	NLM	Wavelet
mountain	10%	114.12	109.21	100.65	110.68
	20%	33.06	30.82	32.72	32.49
	30%	19.06	17.38	23.88	18.73
bedroom	10%	190.59	186.66	179.14	189.17
	20%	153.42	147.53	153.82	152.46
	30%	133.55	127.34	147.44	132.81
jobs	10%	81.72	80.45	77.33	81.87
	20%	72.88	71.06	79.58	72.66
	30%	55.41	53.43	68.78	55.56
rose	10%	142.16	136.19	116.38	139.63
	20%	84.09	77.82	75.55	83.13
	30%	68.11	62.07	66.52	67.59
living	10%	145.6	141.97	135.5	144.84
	20%	96.51	91.47	105.15	96.06
	30%	76.38	71.88	98.8	76.28
colors	10%	142.29	138.8	136.65	141.32
	20%	111.13	105.11	114.24	110.22
	30%	96.57	90.19	108.82	95.92
everest	10%	156.5	152.01	137.97	154.63
	20%	108.53	103.69	105.69	107.41
	30%	91.54	86.76	98.68	90.93

Table 2: Analisi delle immagini con diverse percentuali di compressione - SSIM

Immagine	Compressione	SSIM			
		Senza Filtro	Bilaterale	NLM	Wavelet
mountain	10%	0.935	0.94	0.954	0.935
	20%	0.964	0.968	0.958	0.964
	30%	0.976	0.979	0.96	0.975
bedroom	10%	0.717	0.723	0.693	0.716
	20%	0.799	0.807	0.715	0.797
	30%	0.847	0.854	0.72	0.845
jobs	10%	0.897	0.9	0.898	0.892
	20%	0.929	0.932	0.906	0.923
	30%	0.946	0.949	0.911	0.939
rose	10%	0.854	0.866	0.902	0.851
	20%	0.918	0.929	0.916	0.909
	30%	0.941	0.95	0.92	0.928
living	10%	0.788	0.795	0.782	0.787
	20%	0.861	0.868	0.8	0.86
	30%	0.894	0.899	0.804	0.892
colors	10%	0.864	0.872	0.864	0.864
	20%	0.912	0.923	0.88	0.912
	30%	0.934	0.944	0.887	0.933
everest	10%	0.805	0.812	0.806	0.804
	20%	0.869	0.877	0.821	0.868
	30%	0.898	0.905	0.825	0.897

Table 3: Analisi delle immagini con diverse percentuali di compressione - PSNR

Immagine	Compressione	PSNR			
		Senza Filtro	Bilaterale	NLM	Wavelet
mountain	10%	32.33	32.52	32.87	32.46
	20%	37.71	38.01	37.75	37.78
	30%	40.1	40.5	39.12	40.18
bedroom	10%	30.1	30.19	30.37	30.13
	20%	31.04	31.21	31.03	31.07
	30%	31.65	31.85	31.22	31.67
jobs	10%	33.78	33.85	34.02	33.77
	20%	34.28	34.39	33.89	34.29
	30%	35.47	35.62	34.53	35.45
rose	10%	31.37	31.56	32.24	31.45
	20%	33.65	33.99	34.12	33.7
	30%	34.57	34.97	34.67	34.6
living	10%	31.27	31.38	31.58	31.29
	20%	33.06	33.29	32.68	33.08
	30%	34.07	34.34	32.95	34.08
colors	10%	31.37	31.48	31.55	31.4
	20%	32.44	32.69	32.32	32.48
	30%	33.05	33.35	32.53	33.08
everest	10%	30.96	31.08	31.5	31.01
	20%	32.55	32.74	32.66	32.59
	30%	33.29	33.52	32.96	33.31

6 Conclusioni

Analizzando i risultati ottenuti, si giunge alle seguenti conclusioni:

- In tutte le casistiche analizzate, applicare la trasformata wavelet di haar non si è dimostrata la scelta migliore. Dai risultati ottenuti da tutte le metriche si può evincere che il filtro bilaterale e il filtro NLM si comportano meglio, a prescindere da colore, soggetto e qualità preservata in compressione.

- Analizzando i risultati di MSE e PSNR, ci si rende conto di come il filtro NLM risulti essere sempre il più performante nel caso di immagini molto blocchettizzate. E' possibile notarlo verificando i valori ottenuti in tutte le casistiche con qualità 10%. Come accennato nella sezione inerente al lavoro svolto, il merito di questo risultato è da attribuire alla sfocatura gaussiana applicata poichè, in sua assenza, i valori risultano abbastanza peggiorati.

- Il filtro bilaterale risulta essere il più performante nella maggior parte dei casi. Ottiene valori migliori di wavelet in ogni caso analizzato e, fatta eccezione per le immagini compresse al 10%, è quasi sempre migliore anche di NLM.

- Dai risultati ottenuti da SSIM, si nota come il filtro bilaterale risulta quasi sempre migliore di NLM anche su immagini molto blocchettizzate se si effettua un confronto dal punto di vista strutturale.

- Sono emerse alcune casistiche di peggioramento dell'immagine dopo l'applicazione di un filtro, nello specifico, tutti i rari casi riscontrati fanno riferimento al filtro NLM e mettono d'accordo tutte le metriche. In particolare si tratta di:

  - mountain 30% con NLM
  - bedroom 20 e 30% con NLM
  - jobs 20 e 30% con NLM
  - living 20 e 30% con NLM
  - colors 20 e 30% con NLM
  - everest 30% con NLM

  Analizzando questo elenco, si può dedurre come il problema sia derivato dall'aggiunta di altri artefatti a causa dello sfocamento. Si attenzioni come in nessun caso l'immagine è stata prima compressa con il 10% di qualità. Ciò porta a supporre che immagini non troppo blocchettizzate a volte subiscano un peggioramento

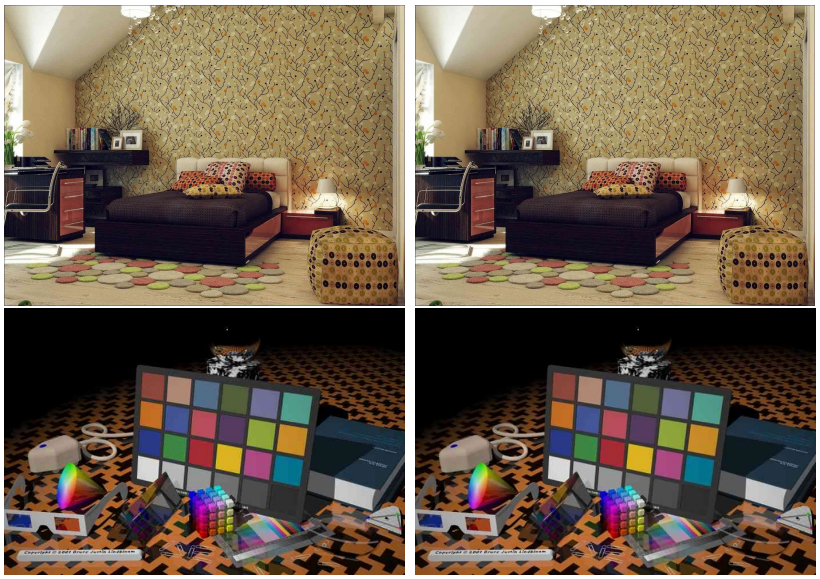
a causa di questo filtro, che risulta invece ottimale se l'immagine era stata compressa mantenendo solo il 10% di qualità.

A seguire i risultati che maggiormente evidenziano i risultati dello studio compiuto:

Risultato ottimale ottenuto dal filtro NLM applicato sull'immagine *rose* compressa con il 10% di qualità mantenuta. A sinistra l'immagine risultante dalla compressione, a destra l'immagine retaurata.

Altro ottimo risultato di restauro con NLM su immagine *everest* con il 10% di qualità preservata. Come possibile osservare da alcuni dettagli, ad esempio le nuvole, lo sfocamento gaussiano ha permesso di attenuare la blocchettizzazione. A sinistra immagine da restaurare, a destra immagine restaurata.

Un esempio di peggioramento è disponibile osservando l'immagine *living* prima e dopo l'applicazione del filtro NLM. In questo caso l'immagine era stata compressa con il 30% di qualità mantenuta. E' possibile notare una perdita di dettaglio (vedasi tabella di riferimento per dati precisi) a causa della sfocatura. Casistica riscontrata solo in immagini non eccessivamente blocchettizzate. A sinistra l'immagine compressa, a destra l'immagine dopo l'applicazione del filtro.



Qui invece due esempi dei migliori risultati ottenuti dal filtro bliaterale. In alto immagine *bedroom*, in basso immagine *colors*. A sinistra le immagini blocchettizzate, a destra quelle restaurate.