

Università degli Studi di Catania

Dipartimento di Matematica e Informatica

Corso di Multimedia e Laboratorio
Relazione Progetto

Steganography WebApp

Applicazione Web per la Steganografia
con Algoritmi Multipli

Autore:

Giuseppe BELLAMACINA
Matricola: 1000030349

Anno Accademico 2025/2026

Abstract

Questo progetto presenta lo sviluppo di un'applicazione web completa per la steganografia, implementata utilizzando Python e Streamlit. L'applicazione offre tre algoritmi steganografici avanzati: LSB (Least Significant Bit), DWT (Discrete Wavelet Transform) e PVD (Pixel Value Differencing), permettendo agli utenti di nascondere e recuperare diversi tipi di dati (testo, immagini e file binari) all'interno di immagini.

L'architettura del progetto segue principi di clean code e modularità, separando la logica di business dall'interfaccia utente. Ogni algoritmo è implementato in modo indipendente, offrendo configurazioni personalizzabili e preset ottimizzati per bilanciare capacità, qualità e robustezza.

Il sistema include funzionalità avanzate come il calcolo di metriche di qualità (PSNR, SSIM), un sistema di backup automatico dei parametri e validazione completa degli input. L'interfaccia web, sviluppata con Streamlit, offre un'esperienza utente intuitiva con visualizzazione in tempo reale delle operazioni.

Indice

Abstract	1
Introduzione	6
1 Fondamenti di Steganografia	7
1.1 Definizione e Storia	7
1.2 Steganografia vs Crittografia	7
1.3 Requisiti di un Sistema Steganografico	8
1.3.1 Impercettibilità (Imperceptibility)	8
1.3.2 Capacità (Capacity)	8
1.3.3 Robustezza (Robustness)	8
1.4 Classificazione degli Algoritmi Steganografici	9
1.4.1 Algoritmi nel Dominio Spaziale	9
1.4.2 Algoritmi nel Dominio della Frequenza	9
2 Algoritmi Steganografici Implementati	10
2.1 LSB - Least Significant Bit	10
2.1.1 Principio di Funzionamento	10
2.1.2 Implementazione	10
2.1.3 Capacità e Prestazioni	11
2.1.4 Vantaggi e Limitazioni	11
2.2 DWT - Discrete Wavelet Transform	12
2.2.1 Principio di Funzionamento	12
2.2.2 Implementazione	12
2.2.3 Configurazioni Implementate	14
2.2.4 Caratteristiche dell'Approccio	14
2.3 PVD - Pixel Value Differencing	14
2.3.1 Principio di Funzionamento	14
2.3.2 Range di Quantizzazione	15
2.3.3 Algoritmo di Embedding	15
2.3.4 Parametri Configurabili	16

<i>INDICE</i>	3
---------------	---

2.3.5 Caratteristiche dell'Approccio	16
2.4 Confronto tra gli Algoritmi	17
3 Architettura del Sistema	18
3.1 Panoramica dell'Architettura	18
3.1.1 Schema Architettonico	19
3.2 Pattern Architetturali Utilizzati	19
3.2.1 Strategy Pattern	19
3.2.2 Singleton Pattern	20
3.2.3 Factory Pattern	20
3.3 Separazione delle Responsabilità	21
3.3.1 Business Logic Layer	21
3.3.2 Utility Layer	21
3.3.3 Configuration Layer	21
3.4 Gestione dello Stato	21
3.5 Gestione degli Errori	22
3.5.1 Validazione Preventiva	22
3.5.2 Try-Catch nelle Operazioni Critiche	22
3.6 Estensibilità	23
4 Implementazione	24
4.1 Tecnologie Utilizzate	24
4.1.1 Stack Tecnologico	24
4.1.2 Gestione Dipendenze con uv	24
4.2 Implementazione degli Algoritmi	25
4.2.1 Sistema di Conversione Binaria	25
4.2.2 Manipolazione Bit LSB	25
4.2.3 Sistema di Backup Parametri	26
4.2.4 Calcolo Metriche di Qualità	27
4.3 Ottimizzazioni Implementate	28
4.3.1 Pre-validation Capacita	28
4.4 Gestione File Temporanei	29
4.5 Testing e Quality Assurance	29
4.5.1 Code Formatting	29
4.5.2 Continuous Integration	30
4.6 Deploy su Streamlit Cloud	30
4.7 Test Sperimentali LSB - Occultamento Immagini	31
4.7.1 Setup Sperimentale	31
4.7.2 Test 1: Configurazione Bilanciata (LSB=4, MSB=4) . .	32
4.7.3 Test 2: Alta Qualità (LSB=1, MSB=1)	33
4.7.4 Test 3: Alta Capacità (LSB=6, MSB=2)	34

4.7.5	Test 4: Massima Capacità Estrema (LSB=7, MSB=8)	35
4.7.6	Test 5: Modalità Automatica (LSB=auto, MSB=8)	36
4.7.7	Conclusioni sui Test LSB	36
4.8	Test Sperimentali LSB - Occultamento File Binari	37
4.8.1	Setup Sperimentale	37
4.8.2	Test 1: Configurazione Alta Qualità (N=2)	38
4.8.3	Test 2: Configurazione Bilanciata (N=4)	39
4.8.4	Test 3: Configurazione Alta Capacità (N=6)	40
4.8.5	Test 4: Configurazione Estrema con DIV Automatico (N=8)	41
4.8.6	Test 5: Configurazione Estrema con DIV=1 (N=8)	42
4.8.7	Conclusioni sui Test LSB Binary	43
4.9	Test Sperimentali DWT - Occultamento File Binari	43
4.9.1	Setup Sperimentale	43
4.9.2	Test 1: Alta Qualità - Configurazione Conservativa	44
4.9.3	Test 2: Capacità Aumentata - Configurazione Estesa	45
4.9.4	Test 3: Massima Capacità - Configurazione Aggressiva	46
4.9.5	Confronto DWT vs LSB per File Binari	47
4.9.6	Conclusioni sui Test DWT Binary	47
4.10	Test Sperimentali PVD - Occultamento File Binari	47
4.10.1	Parametri PVD per File Binari	48
4.10.2	Impatto dei Parametri	48
4.10.3	Esempi di Configurazione	49
4.10.4	Considerazioni sulla Capacità	51
4.10.5	Conclusioni sui Test PVD Binary	51
5	Interfaccia Utente	52
5.1	Design dell'Interfaccia	52
5.2	Struttura dell'Interfaccia	52
5.2.1	Header e Branding	52
5.2.2	Sidebar - Selezione Metodo	53
5.2.3	Selezione Tipo di Dato	53
5.3	Workflow Utente	54
5.3.1	Operazione Hide (Nascondere)	54
5.3.2	Operazione Recover (Recuperare)	54
5.4	Componenti UI Riutilizzabili	55
5.4.1	File Upload con Anteprima	55
5.4.2	Download Button con Icone	55
5.4.3	Preset Configurabili	56
5.5	Feedback Visivo e User Experience	56
5.5.1	Indicatori di Progresso	56

5.5.2	Visualizzazione Metriche	57
5.5.3	Messaggi di Errore Informativi	57
5.6	Responsività e Accessibilità	57
5.6.1	Stili CSS Personalizzati	58
5.7	Gestione dello Stato tra Pagine	58
6	Conclusioni	60
6.1	Riepilogo del Progetto	60
6.1.1	Obiettivi Raggiunti	60
6.2	Confronto Prestazioni	60
6.2.1	Risultati Sperimentali	60
6.2.2	Trade-off Osservati	61
6.3	Contributi Originali	61
6.4	Limitazioni e Miglioramenti Futuri	62
6.4.1	Limitazioni Attuali	62
6.4.2	Sviluppi Futuri Proposti	62
6.5	Considerazioni Finali	63

Introduzione

Contesto e Motivazioni

La steganografia è l'arte e la scienza di nascondere informazioni all'interno di altri dati apparentemente innocui, in modo che la presenza stessa del messaggio nascosto sia difficile da rilevare [4]. A differenza della crittografia, che rende il messaggio illeggibile ma non ne nasconde l'esistenza, la steganografia mira a mascherare completamente la comunicazione segreta.

Nel contesto moderno della sicurezza informatica e della privacy digitale, la steganografia trova applicazioni in diversi ambiti:

- **Protezione del copyright:** Watermarking digitale per proteggere la proprietà intellettuale di immagini, video e documenti
- **Comunicazioni sicure:** Trasmissione di informazioni sensibili senza destare sospetti
- **Autenticazione:** Verifica dell'integrità e dell'autenticità di contenuti multimediali
- **Privacy:** Protezione di dati personali in contesti dove la crittografia potrebbe attirare attenzione indesiderata

Obiettivi del Progetto

Il progetto consiste nello sviluppo di un'applicazione web che implementa tre algoritmi steganografici (LSB, DWT e PVD) per nascondere testo, immagini e file binari all'interno di immagini. L'applicazione offre preset ottimizzati per semplificare l'uso e calcola metriche di qualità (PSNR e SSIM) per valutare i risultati. Un sistema di backup automatico facilita il recupero dei dati nascosti.

Capitolo 1

Fondamenti di Steganografia

1.1 Definizione e Storia

La steganografia deriva dal greco *steganos* (coperto) e *graphein* (scrittura), letteralmente "scrittura nascosta". Le sue origini risalgono all'antica Grecia, dove si utilizzavano tecniche ingegnose quali tatuaggi sul cuoio capelluto rasato di messaggeri, inchiostri invisibili e messaggi nascosti in tavole di cera. Con l'avvento dell'era digitale, la steganografia ha trovato nuove applicazioni nel dominio dei file multimediali, trasformandosi da arte artigianale a scienza computazionale precisa [5].

1.2 Steganografia vs Crittografia

La distinzione tra steganografia e crittografia è fondamentale per comprendere i diversi approcci alla sicurezza dell'informazione. Mentre la crittografia si concentra nel rendere illeggibile un messaggio, attirando inevitabilmente l'attenzione sulla comunicazione, la steganografia mira a nasconderne completamente l'esistenza. Questo approccio discreto comporta alcuni compromessi: la capacità di dati che può essere nascosta è tipicamente limitata dalla dimensione del contenitore, e la robustezza alle manipolazioni varia significativamente a seconda della tecnica impiegata. Tuttavia, il vantaggio principale risiede proprio nel basso livello di sospetto generato: un'immagine contenente dati nascosti appare del tutto ordinaria a un osservatore [7].

Le due tecniche possono essere efficacemente combinate, cifrando prima il messaggio e poi nascondendolo, ottenendo così un doppio livello di sicurezza che protegge sia il contenuto che l'esistenza stessa della comunicazione.

1.3 Requisiti di un Sistema Steganografico

Un sistema steganografico efficace deve soddisfare tre requisiti fondamentali:

1.3.1 Impercettibilità (Imperceptibility)

Le modifiche apportate al contenitore devono rimanere impercettibili all'osservazione umana. La valutazione quantitativa di questo requisito si affida a metriche consolidate nel campo dell'elaborazione delle immagini. Il PSNR (Peak Signal-to-Noise Ratio) misura il rapporto tra il segnale massimo possibile e il rumore di distorsione; valori superiori a 30 dB indicano differenze generalmente impercettibili:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE} \right)$$

dove MAX rappresenta il valore massimo del pixel (255 per immagini a 8 bit) e MSE l'errore quadratico medio tra i pixel originali e modificati. Complementare al PSNR, l'indice SSIM (Structural Similarity Index) valuta la similarità strutturale percettiva tra le immagini su una scala da -1 a 1, dove 1 indica identità perfetta.

1.3.2 Capacità (Capacity)

La quantità di informazioni che possono essere nascoste nell'immagine host. Si misura in:

- **Bit per pixel (bpp):** Rapporto tra bit nascosti e pixel totali
- **Percentuale di utilizzo:** Frazione dell'immagine modificata

Esiste sempre un trade-off tra capacità e impercettibilità: aumentare la quantità di dati nascosti aumenta il rischio di degradazione visibile.

1.3.3 Robustezza (Robustness)

La capacità del messaggio nascosto di resistere a manipolazioni come:

- Compressione JPEG
- Ridimensionamento
- Rotazione e ritaglio

- Aggiunta di rumore
- Filtri di elaborazione

Algoritmi diversi offrono diversi livelli di robustezza a seconda del dominio di embedding utilizzato (spaziale vs frequenza).

1.4 Classificazione degli Algoritmi Steganografici

Gli algoritmi steganografici per immagini si possono classificare in due categorie principali:

1.4.1 Algoritmi nel Dominio Spaziale

Modificano direttamente i valori dei pixel. Sono generalmente:

- Veloci da implementare
- Ad alta capacità
- Poco robusti a manipolazioni
- Esempi: LSB, PVD

1.4.2 Algoritmi nel Dominio della Frequenza

Operano sui coefficienti di trasformazioni matematiche (DCT, DWT, DFT). Sono generalmente:

- Più complessi da implementare
- A capacità inferiore
- Più robusti a compressione e filtri
- Esempi: DWT, DCT-based

Capitolo 2

Algoritmi Steganografici Implementati

2.1 LSB - Least Significant Bit

2.1.1 Principio di Funzionamento

L'algoritmo LSB (Least Significant Bit) è uno dei metodi steganografici più diffusi grazie alla sua semplicità e alta capacità [1]. Il principio si basa sull'osservazione che modificare i bit meno significativi dei valori dei pixel produce cambiamenti impercettibili all'occhio umano.

In un'immagine RGB a 8 bit per canale, ogni pixel ha valori tra 0 e 255. Modificare l'ultimo bit (LSB) cambia il valore del pixel al massimo di ± 1 , una differenza invisibile.

Esempio:

- Pixel originale: $11010110_2 = 214_{10}$
- Bit da nascondere: 1
- Pixel modificato: $11010111_2 = 215_{10}$

2.1.2 Implementazione

L'implementazione nel progetto utilizza un approccio robusto con header strutturato:

```
1 magic_header = "101010101110000"
2 msg_length = format(len(message), "032b")
3 checksum = format(xor_checksum, "016b")
4 msg_binary = binary_convert(message)
```

```

5  terminator = "1111000011110000"
6
7 full_payload = magic_header + msg_length + checksum +
    msg_binary + terminator

```

Codice 2.1: Struttura del payload LSB

Il processo di embedding scorre i pixel dell'immagine modificando l'LSB di ogni componente RGB:

```

1  for i in range(img.width):
2      for j in range(img.height):
3          for z in range(3): # R, G, B
4              if msg_list:
5                  bit = msg_list.pop(0)
6                  pixel = mat[i, j]
7                  color = int(pixel[z])
8                  color = set_last_bit(color, bit)
9                  mat = set_color_component(mat, i, j, color, z)

```

Codice 2.2: Nascondimento LSB

2.1.3 Capacità e Prestazioni

Per un'immagine di dimensione $W \times H$ pixel:

- **Capacità teorica massima:** $W \times H \times 3$ bit (3 bit per pixel RGB)
- **Overhead header:** 80 bit fissi (magic + length + checksum + terminator)
- **Capacità effettiva:** $W \times H \times 3 - 80$ bit

Esempio pratico:

- Immagine 800×600 : 1,440,000 bit teorici = 180 KB
- Capacità effettiva: 179.99 KB
- Percentuale uso tipico: 0.1-5% per messaggi normali

2.1.4 Vantaggi e Limitazioni

L'algoritmo LSB si distingue per la sua semplicità implementativa e l'elevata capacità di embedding, offrendo tipicamente PSNR superiori a 50 dB che garantiscono modifiche del tutto impercettibili all'occhio umano. Tuttavia, questa efficienza ha un costo in termini di robustezza: qualsiasi compressione

JPEG o manipolazione dell'immagine distrugge i bit meno significativi, rendendo impossibile il recupero dei dati. Inoltre, l'approccio è vulnerabile ad analisi statistiche che possono rilevare la presenza di dati nascosti attraverso anomalie nella distribuzione dei valori dei pixel.

2.2 DWT - Discrete Wavelet Transform

2.2.1 Principio di Funzionamento

La DWT (Discrete Wavelet Transform) è una trasformazione matematica che decomponete un segnale (o immagine) in coefficienti che rappresentano informazioni a diverse scale e posizioni. A differenza di LSB che opera nel dominio spaziale, DWT lavora nel dominio della frequenza.

La trasformata wavelet 2D decomponete un'immagine in quattro sub-bande:

- **cA (Approximation)**: Coefficienti di approssimazione - contiene informazioni a bassa frequenza
- **cH (Horizontal)**: Dettagli orizzontali - bordi verticali
- **cV (Vertical)**: Dettagli verticali - bordi orizzontali
- **cD (Diagonal)**: Dettagli diagonali - componenti ad alta frequenza

2.2.2 Implementazione

Il progetto utilizza la wavelet di Haar per la sua semplicità e efficienza [6]. L'implementazione usa `pywt.dwt2` per decomposizione single-level, garantendo efficienza e capacità prevedibile.

```

1 import pywt
2
3 # Applica DWT 2D single-level al canale dell'immagine
4 coeffs = pywt.dwt2(channel_data, 'haar')
5 cA, (cH, cV, cD) = coeffs
6
7 # Nasconde nei coefficienti orizzontali (cH)
8 cH_flat = cH.flatten()

```

Codice 2.3: Decomposizione DWT

Header Robusto a 64-bit:

Per evitare false positive detection nel rumore dei coefficienti DWT, il sistema usa un magic header a 64-bit invece del tradizionale 16-bit:

```

1 # Header 64-bit per robustezza
2 MAGIC_HEADER_64 = "
3     11001001000111101011001010011001101010101001110000101011001101
4 "
5 SIZE_BITS = 32 # Dimensione payload in bit
6
7 # Struttura: [64-bit magic][32-bit size][payload]
8 full_payload = MAGIC_HEADER_64 + format(payload_size, "032b")
9     + payload_bits

```

Codice 2.4: Header DWT robusto

Estrazione Two-Phase:

L'estrazione avviene in due fasi sincronizzate per evitare lettura di rumore:

```

1 # FASE 1: Estraie header + size (96 bit totali)
2 header_and_size = extract_bits_from_coefficients(96)
3 magic = header_and_size[:64]
4 payload_size = int(header_and_size[64:96], 2)
5
6 # Verifica header
7 if magic != MAGIC_HEADER_64:
8     raise ValueError("Header non trovato")
9
10 # FASE 2: Estraie esattamente payload_size bit
11 payload = extract_bits_from_coefficients(payload_size)

```

Codice 2.5: Estrazione two-phase DWT

Questo approccio garantisce che l'estrazione si fermi esattamente dopo il payload, evitando di leggere coefficienti non modificati che causerebbero corruzione dei dati.

L'embedding avviene modificando i coefficienti in base al bit da nascondere:

```

1 # Modifica il coefficiente usando ALPHA
2 bit = int(payload_bit)
3 delta = ALPHA * 50 # Scala per robustezza
4
5 if bit == 1:
6     cH_flat[i] += delta if cH_flat[i] > 0 else -delta
7 else:
8     cH_flat[i] -= delta if cH_flat[i] > 0 else -delta

```

Codice 2.6: Embedding DWT

Il parametro **ALPHA** controlla la forza dell'embedding:

- $\alpha = 0.05$: Qualità massima, capacità ridotta

- $\alpha = 0.1$: Bilanciato (default)
- $\alpha = 0.15$: Robustezza massima, più visibile

2.2.3 Configurazioni Implementate

Il sistema offre tre preset configurabili:

Preset	ALPHA	Bande	Canali
Qualità Massima	0.05	cH	R
Bilanciato	0.10	cH	R
Capacità Massima	0.15	cH, cV, cD	R, G, B

Tabella 2.1: Configurazioni preset DWT

2.2.4 Caratteristiche dell'Approccio

Il principale punto di forza della DWT risiede nella sua robustezza: operando nel dominio della frequenza, l'algoritmo resiste a compressioni JPEG, filtri e aggiunte di rumore che distruggerebbero completamente i dati nascosti con LSB. Questa robustezza si paga con una capacità di embedding ridotta e un costo computazionale superiore. Il PSNR tipicamente si attesta tra 35 e 45 dB, comunque sufficiente per garantire modifiche visivamente accettabili. La calibrazione del parametro ALPHA richiede un bilanciamento attento tra robustezza e qualità visiva, rendendo i preset configurabili particolarmente utili per utenti meno esperti.

2.3 PVD - Pixel Value Differencing

2.3.1 Principio di Funzionamento

PVD (Pixel Value Differencing) è un algoritmo adattivo che sfrutta la differenza tra pixel adiacenti per nascondere dati. L'idea chiave è che modifiche maggiori sono meno percepibili in regioni con alti contrasti (bordi), mentre in regioni uniformi sono necessarie modifiche minori.

Il metodo divide l'immagine in coppie di pixel e calcola la loro differenza [10]:

$$d = |p_2 - p_1|$$

In base alla differenza, viene determinata la capacità di embedding usando range quantizzati.

2.3.2 Range di Quantizzazione

L'implementazione offre due profili di range:

Profile Qualità (default):

```

1 RANGES_QUALITY = [
2     (0, 7, 2),
3     (8, 15, 3),
4     (16, 31, 3),
5     (32, 63, 4),
6     (64, 127, 4),
7 ]

```

Codice 2.7: Range qualità PVD

Profile Capacità:

```

1 RANGES_CAPACITY = [
2     (0, 7, 3),
3     (8, 15, 3),
4     (16, 31, 4),
5     (32, 63, 5),
6     (64, 127, 6),
7     (128, 255, 7),
8 ]

```

Codice 2.8: Range capacità PVD

2.3.3 Algoritmo di Embedding

```

1 def embed_in_pair(pixel1, pixel2, bits):
2     diff = pixel2 - pixel1
3     lower, upper, capacity = get_range_capacity(diff)
4
5     decimal_value = int(bits[:capacity], 2)
6     new_diff = lower + decimal_value
7     new_diff = min(new_diff, upper)
8
9     if diff < 0:
10         new_diff = -new_diff
11
12     m = abs(new_diff) - abs(diff)
13     if diff % 2 == 0:
14         new_pixel1 = pixel1 - m // 2
15         new_pixel2 = pixel2 + m - m // 2
16     else:
17         new_pixel1 = pixel1 - (m + 1) // 2
18         new_pixel2 = pixel2 + m - (m + 1) // 2
19

```

```
20     return max(0, min(255, new_pixel1)), max(0, min(255,
new_pixel2))
```

Codice 2.9: PVD embedding in una coppia di pixel

2.3.4 Parametri Configurabili

L'implementazione offre tre parametri principali. Il profilo di quantizzazione `RANGES` determina la capacità di embedding per ogni livello di differenza tra pixel. Il fallback utilizza l'ultimo range definito (`RANGES[-1]`) invece di un valore hardcoded, garantendo compatibilità tra modalità `QUALITY` e `CAPACITY`.

La spaziatura `PAIR_STEP` controlla la distanza tra coppie consecutive di pixel. Con `PAIR_STEP=1` si ottiene la massima capacità utilizzando tutti i pixel disponibili, mentre valori superiori distribuiscono l'embedding su un'area più ampia riducendo la capacità. I loop di attraversamento usano `range(0, w - PAIR_STEP, 2*PAIR_STEP)` per prevenire accessi out-of-bounds quando si legge `pixel[x + PAIR_STEP]`.

Il parametro `CHANNELS` specifica quali canali RGB utilizzare. Il calcolo di `total_bits` impiega `len(channels)` per determinare precisamente la capacità disponibile indipendentemente dalla configurazione.

L'implementazione richiede particolare attenzione alla sincronizzazione tra le fasi di embedding ed estrazione. Durante il processo di inserimento, l'indice deve essere incrementato esattamente del numero di bit effettivamente inseriti, e non della capacità teorica della coppia. Questo accorgimento previene desincronizzazioni quando il payload termina prima di saturare l'ultima coppia di pixel disponibile.

```
1 bits_to_embed = payload[bit_index : bit_index + capacity]
2 bit_index += len(bits_to_embed)
```

Codice 2.10: Incremento dell'indice durante embedding

2.3.5 Caratteristiche dell'Approccio

PVD rappresenta un interessante compromesso tra la semplicità di LSB e la robustezza di DWT. La natura adattiva dell'algoritmo, che modula la quantità di dati nascosti in base al contenuto locale dell'immagine, produce una distribuzione delle modifiche più naturale e meno rilevabile. Il PSNR risulta generalmente superiore a quello di LSB semplice, attestandosi tra 45 e 55 dB. Tuttavia, la complessità implementativa è maggiore e la sincronizzazione tra

fasi di embedding ed estrazione deve essere gestita con precisione. L'algoritmo mostra vulnerabilità a operazioni di ridimensionamento e ritaglio, che alterano la struttura delle coppie di pixel su cui si basa il metodo.

2.4 Confronto tra gli Algoritmi

Caratteristica	LSB	DWT	PVD
Capacità	Alta (3 bpp)	Media (0.5-1 bpp)	Alta (2-4 bpp)
PSNR medio	>50 dB	35-45 dB	45-55 dB
Velocità	Velocissimo	Lento	Medio
Robustezza	Bassa	Alta	Media
Complessità	Bassa	Alta	Media
Dominio	Spaziale	Frequenza	Spaziale

Tabella 2.2: Confronto quantitativo tra gli algoritmi

Capitolo 3

Architettura del Sistema

3.1 Panoramica dell'Architettura

Il progetto segue un'architettura modulare a strati con separazione netta tra logica di business e presentazione. La struttura è stata progettata per garantire modularità, assegnando a ciascun componente responsabilità ben definite. Questa organizzazione facilita l'estensione del sistema con nuovi algoritmi o funzionalità, mantenendo il codice pulito e ben documentato. I componenti isolati risultano inoltre facilmente testabili, migliorando complessivamente la manutenibilità del progetto.

3.1.1 Schema Architetturale

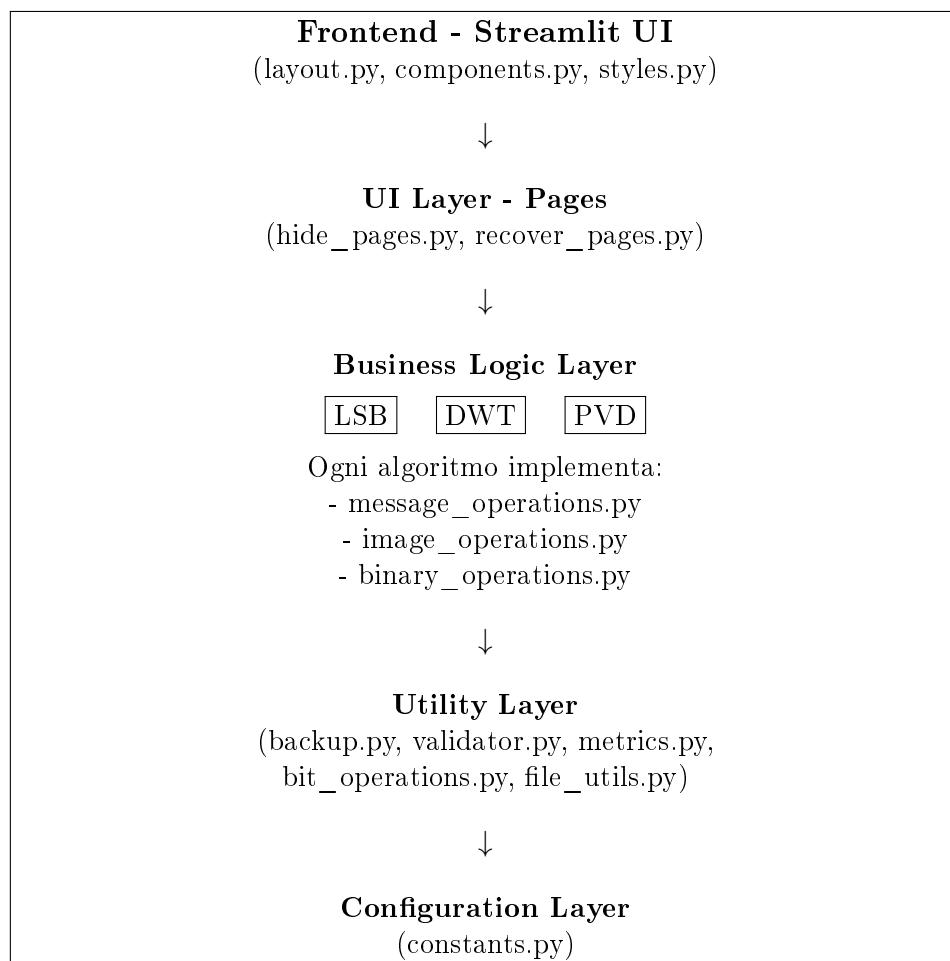


Figura 3.1: Architettura a strati del sistema

3.2 Pattern Architetturali Utilizzati

3.2.1 Strategy Pattern

Ogni algoritmo (LSB, DWT, PVD) implementa la stessa interfaccia per le tre operazioni:

- hide_message() / get_message()
- hide_image() / get_image()
- hide_bin_file() / get_bin_file()

Questo permette al sistema di selezionare dinamicamente l'algoritmo a runtime:

```

1 if selected_method == SteganographyMethod.LSB:
2     from src.steganografia.lsb import message_operations as
3         ops
4 elif selected_method == SteganographyMethod.DWT:
5     from src.steganografia.dwt import message_operations as
6         ops
7 elif selected_method == SteganographyMethod.PVD:
8     from src.steganografia.pvd import message_operations as
9         ops
10
11 result = ops.MessageSteganography.hide_message(img, msg)

```

Codice 3.1: Selezione dinamica algoritmo

3.2.2 Singleton Pattern

Il sistema di backup e le classi di utility usano metodi statici per evitare istanziazioni multiple:

```

1 class ParameterBackup:
2     @staticmethod
3     def save_backup_data(data_type, params, filepath):
4         ...
5
6 class QualityMetrics:
7     @staticmethod
8     def calculate_psnr(original, modified):
9         ...

```

Codice 3.2: Utility classes come singleton

3.2.3 Factory Pattern

La creazione di componenti UI avviene attraverso factory methods nella classe `AppLayout`:

```

1 class AppLayout:
2     @staticmethod
3     def setup_sidebar():
4         # Crea e configura sidebar con cards metodi
5
6     @staticmethod
7     def show_data_type_selector():
8         # Crea selector tipo dato

```

Codice 3.3: Factory methods per UI

3.3 Separazione delle Responsabilità

3.3.1 Business Logic Layer

Contiene tutta la logica steganografica, completamente indipendente dall'UI. Ciascun modulo gestisce un tipo specifico di dato: `message_operations.py` per l'embedding e l'estrazione di stringhe, `image_operations.py` per le immagini, e `binary_operations.py` per file binari arbitrari.

3.3.2 Utility Layer

Fornisce servizi trasversali utilizzati da tutti gli algoritmi. Il modulo `backup.py` gestisce serializzazione e deserializzazione dei parametri in formato JSON, mentre `validator.py` si occupa della validazione di dimensioni, formati e capacità. Il calcolo delle metriche PSNR e SSIM è centralizzato in `metrics.py`, le operazioni binarie e conversioni in `bit_operations.py`, e la gestione I/O dei file in `file_utils.py`.

3.3.3 Configuration Layer

Centralizza tutte le costanti e configurazioni:

```

1  class SteganographyMethod(str, Enum):
2      LSB = "lsb"
3      DWT = "dwt"
4      PVD = "pvd"
5
6  class DataType(str, Enum):
7      STRING = "string"
8      IMAGE = "image"
9      BINARY = "binary"
10
11 class CompressionMode(str, Enum):
12     NO_ZIP = "no_zip"
13     FILE = "file"
14     DIR = "dir"
```

Codice 3.4: Enumerazioni di configurazione

3.4 Gestione dello Stato

L'applicazione utilizza Streamlit Session State per mantenere lo stato tra le interazioni:

```

1 # Inizializzazione
2 if "selected_method" not in st.session_state:
3     st.session_state.selected_method = SteganographyMethod.LSB
4
5 # Memorizzazione risultati
6 st.session_state["hide_image_results"] = {
7     "image": img_data,
8     "preview_image": img,
9     "metrics": {"psnr": 45.2, "ssim": 0.998}
10 }
11
12 # Recupero risultati
13 if "hide_image_results" in st.session_state:
14     results = st.session_state["hide_image_results"]

```

Codice 3.5: Gestione stato con Streamlit

3.5 Gestione degli Errori

Il sistema implementa una gestione robusta degli errori a più livelli:

3.5.1 Validazione Preventiva

```

1 class ParameterValidator:
2     @staticmethod
3     def validate_image_size_for_message(img, message):
4         required_bits = len(message) * 8 + 80 # +header
5         available_bits = img.width * img.height * 3
6
7         if required_bits > available_bits:
8             raise ValueError(
9                 f"Messaggio troppo lungo."
10                f"Richiesti: {required_bits} bit, "
11                f"Disponibili: {available_bits} bit"
12            )

```

Codice 3.6: Validazione input

3.5.2 Try-Catch nelle Operazioni Critiche

```

1 try:
2     result_img, metrics = hide_message(img, message,
3                                         backup_file)
4     st.success("Messaggio nascosto con successo!")
5     # Mostra risultati...

```

```
5 except ValueError as e:
6     st.error(f"Errore di validazione: {str(e)}")
7 except Exception as e:
8     st.error(f"Errore imprevisto: {str(e)}")
9     # Log per debugging...
```

Codice 3.7: Gestione eccezioni nelle pagine UI

3.6 Estensibilità

L’architettura facilita l’aggiunta di nuovi algoritmi:

1. Creare nuova directory sotto `src/steganografia/`
2. Implementare i tre moduli di operazioni
3. Aggiungere enum in `SteganographyMethod`
4. Aggiornare UI per includere il nuovo metodo

Non sono richieste modifiche agli altri moduli grazie all’isolamento tra componenti.

Capitolo 4

Implementazione

4.1 Tecnologie Utilizzate

4.1.1 Stack Tecnologico

Il progetto è sviluppato interamente in Python 3.12 (ma è retrocompatibile con versioni precedenti dalla 3.9 in poi) utilizzando le seguenti librerie [8, 3, 9]:

Libreria	Versione	Scopo
Streamlit	$\geq 1.20.0$	Framework web per l'interfaccia utente
NumPy	$\geq 1.24.0$	Operazioni matriciali e calcoli numerici
Pillow (PIL)	$\geq 9.5.0$	Manipolazione e I/O immagini
PyWavelets	$\geq 1.4.0$	Trasformata wavelet per DWT
scikit-image	$\geq 0.20.0$	Metriche di qualità (SSIM)

Tabella 4.1: Dipendenze principali del progetto

4.1.2 Gestione Dipendenze con uv

Il progetto utilizza uv come package manager per gestione rapida e affidabile delle dipendenze:

```
1 [project]
2 name = "steganography-webapp"
3 version = "1.0.0"
4 requires-python = ">=3.9"
5
6 dependencies = [
7     "streamlit>=1.20.0",
```

```

8     "numpy>=1.24.0",
9     "Pillow>=9.5.0",
10    "PyWavelets>=1.4.0",
11    "scikit-image>=0.20.0",
12 ]
13
14 [project.optional-dependencies]
15 dev = [
16     "black>=23.0.0",
17     "isort>=5.12.0",
18 ]
19
20 [tool.uv]
21 dev-dependencies = [
22     "black>=23.0.0",
23     "isort>=5.12.0",
24 ]

```

Codice 4.1: pyproject.toml - configurazione dipendenze

4.2 Implementazione degli Algoritmi

4.2.1 Sistema di Conversione Binaria

Tutti gli algoritmi si basano su funzioni di conversione tra testo e binario:

```

1 def binary_convert(s: str) -> str:
2     """Converte una stringa in rappresentazione binaria"""
3     binary = "".join(format(ord(char), "08b") for char in s)
4     return binary
5
6 def binary_convert_back(s: str) -> str:
7     """Riconverte binario in stringa"""
8     chars = [s[i:i+8] for i in range(0, len(s), 8)]
9     text = "".join(chr(int(char, 2)) for char in chars)
10    return text

```

Codice 4.2: Conversione testo-binario

4.2.2 Manipolazione Bit LSB

Operazioni fondamentali per l'algoritmo LSB:

```

1 def set_last_bit(value: int, bit: str) -> int:
2     """Imposta l'ultimo bit di un valore"""
3     if bit == "1":
4         return value | 1 # OR bit-wise per settare a 1

```

```

5     else:
6         return value & ~1 # AND con NOT per settare a 0
7
8 def get_last_bit(value: int) -> str:
9     """Estraie l'ultimo bit di un valore"""
10    return str(value & 1)
11
12 def set_color_component(mat, x: int, y: int, value: int,
13                         component: int):
14     """Modifica una componente RGB di un pixel"""
15     pixel = list(mat[x, y])
16     pixel[component] = value
17     mat[x, y] = tuple(pixel)
18
19
20
21
22
23
24
25
26
27

```

Codice 4.3: Operazioni bit-level

4.2.3 Sistema di Backup Parametri

Implementazione del salvataggio automatico dei parametri per il recupero:

```

1 import json
2 import os
3 from pathlib import Path
4
5 class ParameterBackup:
6     @staticmethod
7     def save_backup_data(data_type, params, filepath=None):
8         """Salva parametri in file JSON"""
9         if filepath is None:
10             filepath = f"backup_{data_type}_{int(time.time())}.json"
11
12         backup_data = {
13             "data_type": data_type,
14             "timestamp": time.time(),
15             "parameters": params,
16             "version": "1.0"
17         }
18
19         with open(filepath, "w") as f:
20             json.dump(backup_data, f, indent=2)
21
22         return filepath
23
24     @staticmethod
25     def load_backup_data(filepath):
26         """Carica parametri da file JSON"""
27         if not os.path.exists(filepath):
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
337
338
339
339
340
341
342
343
344
345
345
346
347
347
348
349
349
350
351
352
353
353
354
355
355
356
357
357
358
359
359
360
361
361
362
363
363
364
365
365
366
367
367
368
369
369
370
371
371
372
373
373
374
375
375
376
377
377
378
379
379
380
381
381
382
383
383
384
385
385
386
387
387
388
389
389
390
391
391
392
393
393
394
395
395
396
397
397
398
399
399
400
401
401
402
403
403
404
405
405
406
407
407
408
409
409
410
411
411
412
413
413
414
415
415
416
417
417
418
419
419
420
421
421
422
423
423
424
425
425
426
427
427
428
429
429
430
431
431
432
433
433
434
435
435
436
437
437
438
439
439
440
441
441
442
443
443
444
445
445
446
447
447
448
449
449
450
451
451
452
453
453
454
455
455
456
457
457
458
459
459
460
461
461
462
463
463
464
465
465
466
467
467
468
469
469
470
471
471
472
473
473
474
475
475
476
477
477
478
479
479
480
481
481
482
483
483
484
485
485
486
487
487
488
489
489
490
491
491
492
493
493
494
495
495
496
497
497
498
499
499
500
501
501
502
503
503
504
505
505
506
507
507
508
509
509
510
511
511
512
513
513
514
515
515
516
517
517
518
519
519
520
521
521
522
523
523
524
525
525
526
527
527
528
529
529
530
531
531
532
533
533
534
535
535
536
537
537
538
539
539
540
541
541
542
543
543
544
545
545
546
547
547
548
549
549
550
551
551
552
553
553
554
555
555
556
557
557
558
559
559
560
561
561
562
563
563
564
565
565
566
567
567
568
569
569
570
571
571
572
573
573
574
575
575
576
577
577
578
579
579
580
581
581
582
583
583
584
585
585
586
587
587
588
589
589
590
591
591
592
593
593
594
595
595
596
597
597
598
599
599
600
601
601
602
603
603
604
605
605
606
607
607
608
609
609
610
611
611
612
613
613
614
615
615
616
617
617
618
619
619
620
621
621
622
623
623
624
625
625
626
627
627
628
629
629
630
631
631
632
633
633
634
635
635
636
637
637
638
639
639
640
641
641
642
643
643
644
645
645
646
647
647
648
649
649
650
651
651
652
653
653
654
655
655
656
657
657
658
659
659
660
661
661
662
663
663
664
665
665
666
667
667
668
669
669
670
671
671
672
673
673
674
675
675
676
677
677
678
679
679
680
681
681
682
683
683
684
685
685
686
687
687
688
689
689
690
691
691
692
693
693
694
695
695
696
697
697
698
699
699
700
701
701
702
703
703
704
705
705
706
707
707
708
709
709
710
711
711
712
713
713
714
715
715
716
717
717
718
719
719
720
721
721
722
723
723
724
725
725
726
727
727
728
729
729
730
731
731
732
733
733
734
735
735
736
737
737
738
739
739
740
741
741
742
743
743
744
745
745
746
747
747
748
749
749
750
751
751
752
753
753
754
755
755
756
757
757
758
759
759
760
761
761
762
763
763
764
765
765
766
767
767
768
769
769
770
771
771
772
773
773
774
775
775
776
777
777
778
779
779
780
781
781
782
783
783
784
785
785
786
787
787
788
789
789
790
791
791
792
793
793
794
795
795
796
797
797
798
799
799
800
801
801
802
803
803
804
805
805
806
807
807
808
809
809
810
811
811
812
813
813
814
815
815
816
817
817
818
819
819
820
821
821
822
823
823
824
825
825
826
827
827
828
829
829
830
831
831
832
833
833
834
835
835
836
837
837
838
839
839
840
841
841
842
843
843
844
845
845
846
847
847
848
849
849
850
851
851
852
853
853
854
855
855
856
857
857
858
859
859
860
861
861
862
863
863
864
865
865
866
867
867
868
869
869
870
871
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
883
883
884
885
885
886
887
887
888
889
889
890
891
891
892
893
893
894
895
895
896
897
897
898
899
899
900
901
901
902
903
903
904
905
905
906
907
907
908
909
909
910
911
911
912
913
913
914
915
915
916
917
917
918
919
919
920
921
921
922
923
923
924
925
925
926
927
927
928
929
929
930
931
931
932
933
933
934
935
935
936
937
937
938
939
939
940
941
941
942
943
943
944
945
945
946
947
947
948
949
949
950
951
951
952
953
953
954
955
955
956
957
957
958
959
959
960
961
961
962
963
963
964
965
965
966
967
967
968
969
969
970
971
971
972
973
973
974
975
975
976
977
977
978
979
979
980
981
981
982
983
983
984
985
985
986
987
987
988
989
989
990
991
991
992
993
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534

```

```

28         raise FileNotFoundError(f"File backup non trovato:
29                               {filepath}")
30
31     with open(filepath, "r") as f:
32         backup_data = json.load(f)
33
34     return backup_data["parameters"]

```

Codice 4.4: Sistema backup JSON

4.2.4 Calcolo Metriche di Qualità

Implementazione PSNR e SSIM usando direttamente le librerie scikit-image:

```

1  import numpy as np
2  from skimage.metrics import peak_signal_noise_ratio as psnr
3  from skimage.metrics import structural_similarity as ssim
4
5  class QualityMetrics:
6      @staticmethod
7      def calculate_metrics(original_img: Image.Image,
8                            modified_img: Image.Image) -> dict:
9          """Calcola SSIM e PSNR tra immagine originale e
10             modificata"""
11
12         # Converte in RGB se necessario
13         if original_img.mode != "RGB":
14             original_img = original_img.convert("RGB")
15         if modified_img.mode != "RGB":
16             modified_img = modified_img.convert("RGB")
17
18         # Converte in array numpy
19         original_array = np.array(original_img)
20         modified_array = np.array(modified_img)
21
22         # Calcola SSIM (per immagini multichannel)
23         ssim_value = ssim(
24             original_array,
25             modified_array,
26             channel_axis=2,    # RGB ha 3 canali
27             data_range=255,    # Range dei valori dei pixel (0-
28                           255)
29
30         # Calcola PSNR usando direttamente la libreria
31         if np.array_equal(original_array, modified_array):
32             psnr_value = np.inf
33         else:
34             psnr_value = psnr(original_array, modified_array,
35                               data_range=255)

```

```

34         return {"ssim": ssim_value, "psnr": psnr_value}
35
36     @staticmethod
37     def format_metrics(metrics: dict) -> str:
38         """Formatta le metriche in una stringa leggibile"""
39         ssim_val = metrics["ssim"]
40         psnr_val = metrics["psnr"]
41
42         # Interpreta qualita SSIM
43         if ssim_val >= 0.99:
44             ssim_quality = "Eccezionale"
45         elif ssim_val >= 0.95:
46             ssim_quality = "Ottima"
47         else:
48             ssim_quality = "Buona"
49
50         # Interpreta qualita PSNR
51         if np.isinf(psnr_val):
52             psnr_str = "inf"
53             psnr_quality = "Perfetto"
54         elif psnr_val >= 40:
55             psnr_str = f"{psnr_val:.2f}"
56             psnr_quality = "Ottima"
57         else:
58             psnr_str = f"{psnr_val:.2f}"
59             psnr_quality = "Buona"
60
61         return (f"SSIM: {ssim_val:.4f} ({ssim_quality}) | "
62                 f"PSNR: {psnr_str} dB ({psnr_quality})")
63

```

Codice 4.5: Metriche di qualità

4.3 Ottimizzazioni Implementate

4.3.1 Pre-validation Capacita

Prima di iniziare l'embedding, il sistema verifica la capacità disponibile:

```

1 # Calcola capacità prima dell'embedding
2 required_bits = len(message) * 8 + header_size
3 available_bits = img.width * img.height * 3
4
5 if required_bits > available_bits:
6     # Fallimento immediato senza iniziare l'embedding
7     raise ValueError("Capacità insufficiente")
8
9 # Mostra percentuale utilizzo previsto

```

```

10 usage = (required_bits / available_bits) * 100
11 st.info(f"Utilizzo previsto: {usage:.2f}%")

```

Codice 4.6: Pre-validazione per evitare computazioni inutili

4.4 Gestione File Temporanei

Sistema robusto per gestire file temporanei durante le operazioni:

```

1 import tempfile
2 import os
3
4 def save_uploaded_file(uploaded_file):
5     """Salva file caricato in directory temporanea"""
6     temp_dir = tempfile.gettempdir()
7     temp_path = os.path.join(temp_dir, uploaded_file.name)
8
9     with open(temp_path, "wb") as f:
10         f.write(uploaded_file.getvalue())
11
12     return temp_path
13
14 def cleanup_temp_file(filepath):
15     """Rimuove file temporaneo se esiste"""
16     try:
17         if os.path.exists(filepath):
18             os.remove(filepath)
19     except Exception as e:
20         # Log warning ma non bloccare l'esecuzione
21         print(f"Warning: impossibile rimuovere {filepath}: {e}")
22

```

Codice 4.7: Gestione sicura file temporanei

4.5 Testing e Quality Assurance

4.5.1 Code Formatting

Il progetto utilizza Black e isort per mantenere coerenza del codice:

```

1 [tool.black]
2 line-length = 88
3 target-version = ["py311"]
4
5 [tool.isort]
6 profile = "black"

```

```
7 line_length = 88
```

Codice 4.8: Configurazione formattazione

4.5.2 Continuous Integration

Pipeline CI/CD con GitHub Actions:

```
1 name: CI
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  test:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v4
14
15      - name: Install uv
16        uses: astral-sh/setup-uv@v4
17        with:
18          enable-cache: true
19
20      - name: Set up Python 3.12
21        uses: actions/setup-python@v5
22        with:
23          python-version: "3.12"
24
25      - name: Install dependencies
26        run: uv sync --all-extras
27
28      - name: Check import sorting
29        run: uv run isort src/ config/ --check-only
30
31      - name: Check code formatting
32        run: uv run black src/ config/ --check
```

Codice 4.9: CI workflow con uv

4.6 Deploy su Streamlit Cloud

L'applicazione è deployata su Streamlit Cloud con configurazione automatica:

- **uv.lock**: Rilevato automaticamente da Streamlit Cloud
- **.python-version**: Specifica Python 3.12
- **pyproject.toml**: Contiene tutte le dipendenze

Il deploy è completamente automatizzato: ogni push su `main` triggerà un redeploy.

4.7 Test Sperimentali LSB - Occultamento Immagini

In questa sezione presentiamo test dell'algoritmo LSB per l'occultamento di immagini, variando LSB e MSB per analizzare il trade-off tra qualità visiva e fedeltà dell'immagine recuperata.

4.7.1 Setup Sperimentale

Per tutti i test sono state utilizzate le seguenti immagini:

- **Immagine Host**: `host.jpg` - immagine carrier di dimensioni maggiori
- **Immagine Segreta**: `occulted.jpg` - immagine da nascondere (720×1280 pixel)

4.7.2 Test 1: Configurazione Bilanciata (LSB=4, MSB=4)

Questa configurazione rappresenta un buon compromesso tra qualità visiva e capacità di occultamento.

Parametro	Valore
LSB	4 bit
MSB	4 bit
DIV	3.38 (automatico)
Pixel utilizzati	29.63%
SSIM	0.8563 (Eccellente)
PSNR	35.84 dB (Buona)

Tabella 4.2: Parametri e metriche per configurazione bilanciata



Figura 4.1: Immagine stego con LSB=4, MSB=4



Figura 4.2: Immagine recuperata da LSB=4, MSB=4

Analisi: Con LSB=4 e MSB=4 si ottiene un eccellente SSIM di 0.8563, indicando che l'immagine stego è quasi indistinguibile dall'originale. Il recupero dell'immagine segreta è preciso, preservando 4 bit per canale.

4.7.3 Test 2: Alta Qualità (LSB=1, MSB=1)

Configurazione che privilegia la massima qualità dell'immagine stego a discapito della fedeltà dell'immagine recuperata.

Parametro	Valore
LSB	1 bit
MSB	1 bit
DIV	3.38 (automatico)
Pixel utilizzati	29.63%
SSIM	0.9981 (Eccellente)
PSNR	56.60 dB (Eccellente)

Tabella 4.3: Parametri e metriche per configurazione alta qualità



Figura 4.3: Immagine stego con
LSB=1, MSB=1



Figura 4.4: Immagine recuperata da
LSB=1, MSB=1

Analisi: Con LSB=1 e MSB=1 si ottiene un SSIM quasi perfetto (0.9981) e un PSNR eccellente di 56.60 dB. L'immagine stego è praticamente identica all'originale. Tuttavia, l'immagine recuperata preserva solo 1 bit per canale, risultando in una qualità visiva degradata ma ancora riconoscibile.

4.7.4 Test 3: Alta Capacità (LSB=6, MSB=2)

Configurazione che massimizza la capacità di occultamento utilizzando più bit dell'immagine host.

Parametro	Valore
LSB	6 bit
MSB	2 bit
DIV	10.12 (automatico)
Pixel utilizzati	9.88%
SSIM	0.5601 (Accettabile)
PSNR	27.07 dB (Accettabile)

Tabella 4.4: Parametri e metriche per configurazione alta capacità



Figura 4.5: Immagine stego con LSB=6, MSB=2



Figura 4.6: Immagine recuperata da LSB=6, MSB=2

Analisi: Modificando 6 bit per pixel si riduce significativamente l'utilizzo dell'immagine host (solo 9.88% dei pixel), ma a costo di una qualità visiva ridotta (SSIM=0.5601). L'immagine recuperata preserva solo 2 bit per canale.

4.7.5 Test 4: Massima Capacità Estrema (LSB=7, MSB=8)

Test estremo che modifica quasi tutti i bit dell’immagine host per preservare l’intera profondità colore dell’immagine segreta.

Parametro	Valore
LSB	7 bit
MSB	8 bit (tutti)
DIV	2.95 (automatico)
Pixel utilizzati	33.86%
SSIM	0.1008 (Scarsa)
PSNR	18.03 dB (Scarsa)

Tabella 4.5: Parametri e metriche per configurazione estrema



Figura 4.7: Immagine stego con LSB=7, MSB=8

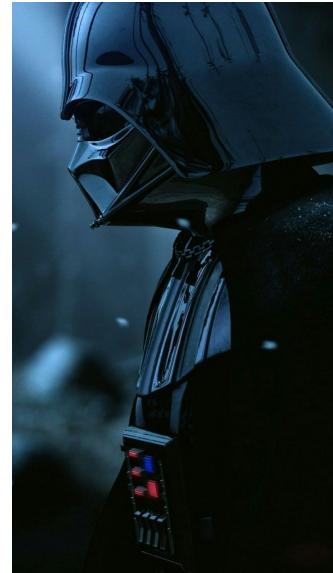


Figura 4.8: Immagine recuperata da LSB=7, MSB=8 (qualità perfetta)

Analisi: Con LSB=7 e MSB=8 si ottiene un recupero perfetto dell’immagine segreta (tutti gli 8 bit preservati), ma l’immagine stego presenta distorsioni evidenti (SSIM=0.1008, PSNR=18.03 dB). Questa configurazione è inadatta per applicazioni steganografiche reali.

4.7.6 Test 5: Modalità Automatica (LSB=auto, MSB=8)

Test della modalità automatica che calcola il valore ottimale di LSB in base alle dimensioni delle immagini.

Parametro	Valore
LSB	3 bit (calcolato automaticamente)
MSB	8 bit
DIV	1.27 (automatico)
Pixel utilizzati	79.01%
SSIM	0.9000 (Eccellente)
PSNR	38.39 dB (Buona)

Tabella 4.6: Parametri e metriche per modalità automatica



Figura 4.9: Immagine stego con modalità automatica (LSB=3, MSB=8)

Analisi: La modalità automatica ha calcolato LSB=3 come valore ottimale per nascondere l'intera immagine segreta (MSB=8) mantenendo un'eccellente qualità visiva (SSIM=0.9000, PSNR=38.39 dB). L'utilizzo dei pixel è elevato (79.01%), indicando un'efficiente distribuzione dei dati.

4.7.7 Conclusioni sui Test LSB

I test dimostrano chiaramente il trade-off tra qualità dell'immagine stego e fedeltà dell'immagine recuperata:

- **LSB=1, MSB=1:** Ottima invisibilità (SSIM=0.998), immagine recuperata degradata

- **LSB=4, MSB=4:** Configurazione bilanciata raccomandata ($SSIM=0.856$)
- **LSB=6, MSB=2:** Basso utilizzo pixel (9.88%), qualità accettabile
- **LSB=7, MSB=8:** Recupero perfetto ma distorsioni visibili
- **Modalità automatica:** Calcolo intelligente per ottimizzare capacità/qualità

La scelta dei parametri dipende dal caso d'uso: per steganografia che richiede invisibilità, configurazioni con $LSB \leq 4$ sono preferibili. Per archiviazione con compressione visiva accettabile, configurazioni con $MSB \geq 6$ preservano meglio l'immagine segreta.

4.8 Test Sperimentali LSB - Occultamento File Binari

In questa sezione presentiamo test sperimentali dell'algoritmo LSB per l'occultamento di file binari, analizzando l'impatto del parametro N (numero di bit modificati per pixel) sulla qualità visiva dell'immagine stego.

4.8.1 Setup Sperimentale

Per tutti i test è stato utilizzato:

- **Immagine Host:** host.jpg - immagine carrier
- **File Binario:** File di test (dimensione variabile)
- **DIV:** Calcolato automaticamente (tranne Test 5)

4.8.2 Test 1: Configurazione Alta Qualità (N=2)

Configurazione che privilegia la massima qualità visiva modificando solo 2 bit per pixel.

Parametro	Valore
N (bit per pixel)	2 bit
DIV	1.16 (automatico)
Pixel utilizzati	85.98%
SSIM	0.9719 (Eccellente)
PSNR	44.85 dB (Eccellente)

Tabella 4.7: Parametri e metriche per N=2



Figura 4.10: Immagine stego con N=2 - massima qualità visiva

Analisi: Con N=2 si ottiene un SSIM quasi perfetto (0.9719) e PSNR eccellente (44.85 dB). L'immagine è praticamente identica all'originale. L'elevato utilizzo dei pixel (85.98%) indica che quasi tutta l'immagine viene impiegata per nascondere i dati.

4.8.3 Test 2: Configurazione Bilanciata (N=4)

Configurazione che offre un buon compromesso tra capacità e qualità visiva.

Parametro	Valore
N (bit per pixel)	4 bit
DIV	2.33 (automatico)
Pixel utilizzati	42.99%
SSIM	0.8194 (Eccellente)
PSNR	35.74 dB (Buona)

Tabella 4.8: Parametri e metriche per N=4



Figura 4.11: Immagine stego con N=4 - configurazione bilanciata

Analisi: N=4 rappresenta la configurazione raccomandata per la maggior parte degli utilizzi. Mantiene un'eccellente qualità visiva (SSIM=0.8194) con un utilizzo efficiente dei pixel (42.99%).

4.8.4 Test 3: Configurazione Alta Capacità (N=6)

Configurazione che massimizza la capacità di occultamento modificando 6 bit per pixel.

Parametro	Valore
N (bit per pixel)	6 bit
DIV	3.49 (automatico)
Pixel utilizzati	28.66%
SSIM	0.3167 (Accettabile)
PSNR	24.57 dB (Accettabile)

Tabella 4.9: Parametri e metriche per N=6



Figura 4.12: Immagine stego con N=6 - alta capacità

Analisi: Con N=6 l'utilizzo dei pixel scende al 28.66%, permettendo di nascondere più dati in meno spazio. Tuttavia, la qualità visiva è ridotta (SSIM=0.3167), con distorsioni visibili ma accettabili per alcuni scenari.

4.8.5 Test 4: Configurazione Estrema con DIV Automatico (N=8)

Test estremo che modifica tutti gli 8 bit per pixel con DIV calcolato automaticamente.

Parametro	Valore
N (bit per pixel)	8 bit (tutti)
DIV	4.65 (automatico)
Pixel utilizzati	21.50%
SSIM	0.0592 (Scarsa)
PSNR	12.94 dB (Scarsa)

Tabella 4.10: Parametri e metriche per N=8 con DIV automatico



Figura 4.13: Immagine stego con N=8, DIV automatico - distorsioni evidenti

Analisi: Modificando tutti gli 8 bit con DIV automatico si ottiene la massima capacità (solo 21.50% dei pixel utilizzati), ma l'immagine presenta distorsioni molto evidenti (SSIM=0.0592, PSNR=12.94 dB). Inadatto per steganografia.

4.8.6 Test 5: Configurazione Estrema con DIV=1 (N=8)

Test estremo che modifica tutti gli 8 bit per pixel con DIV fissato a 1 (nessuna distribuzione).

Parametro	Valore
N (bit per pixel)	8 bit (tutti)
DIV	1.00 (manuale)
Pixel utilizzati	21.50%
SSIM	0.7859 (Buona)
PSNR	13.75 dB (Scarsa)

Tabella 4.11: Parametri e metriche per N=8 con DIV=1



Figura 4.14: Immagine stego con N=8, DIV=1 - concentrazione visibile

Analisi: Con DIV=1 i dati vengono scritti consecutivamente senza distribuzione. Curiosamente, l'SSIM migliora rispetto al DIV automatico (0.7859 vs 0.0592), ma il PSNR rimane basso (13.75 dB). La concentrazione dei dati è visibile nell'immagine, creando pattern riconoscibili.

4.8.7 Conclusioni sui Test LSB Binary

I test dimostrano l'impatto del parametro N sulla qualità dell'immagine stego:

- **N=2**: Massima qualità (SSIM=0.972), alto utilizzo pixel (86%)
- **N=4**: Configurazione bilanciata raccomandata (SSIM=0.819)
- **N=6**: Alta capacità, qualità accettabile (SSIM=0.317)
- **N=8, DIV auto**: Massima capacità, qualità scarsa (SSIM=0.059)
- **N=8, DIV=1**: Pattern visibili, inadatto per steganografia

Il parametro DIV influenza la distribuzione dei dati: valori automatici distribuiscono uniformemente i dati nell'immagine, mentre DIV=1 concentra i dati creando artefatti visibili. Per applicazioni steganografiche, $N \leq 4$ con DIV automatico offre il miglior compromesso invisibilità/capacità.

4.9 Test Sperimentali DWT - Occultamento File Binari

In questa sezione presentiamo test sperimentali dell'algoritmo DWT (Discrete Wavelet Transform) per l'occultamento di file binari. A differenza di LSB, DWT modifica i coefficienti delle trasformate wavelet e ha una capacità notevolmente inferiore ma offre maggiore robustezza.

4.9.1 Setup Sperimentale

Per tutti i test è stato utilizzato:

- **Immagine Host**: host.jpg - immagine carrier
- **File Binario**: File di test da 80 KB
- **Wavelet**: Haar (trasformata discreta)
- **Parametri Variabili**: ALPHA (fattore di embedding), BANDS (bande DWT), CHANNELS (canali RGB)

4.9.2 Test 1: Alta Qualità - Configurazione Conservativa

Configurazione che privilegia la qualità visiva usando ALPHA basso e un solo canale.

Parametro	Valore
ALPHA	0.05
BANDS	cH (1 banda)
CHANNELS	1 canale (R)
SSIM	0.7726 (Buona)
PSNR	26.62 dB (Accettabile)

Tabella 4.12: Parametri e metriche per configurazione alta qualità



Figura 4.15: DWT con ALPHA=0.05, 1 banda, 1 canale - alta qualità

Analisi: Con ALPHA=0.05 (embedding debole) e un solo canale, si ottiene un buon SSIM (0.7726). Il PSNR è accettabile (26.62 dB). Questa configurazione è adatta quando la qualità visiva è prioritaria, ma ha capacità limitata.

4.9.3 Test 2: Capacità Aumentata - Configurazione Estesa

Configurazione che aumenta la capacità usando ALPHA moderato e tutti e 3 i canali RGB.

Parametro	Valore
ALPHA	0.15
BANDS	3 bande (cH, cV, cD)
CHANNELS	3 canali (RGB)
SSIM	0.8910 (Eccellente)
PSNR	35.24 dB (Buona)

Tabella 4.13: Parametri e metriche per configurazione alta capacità



Figura 4.16: DWT con ALPHA=0.15, 3 bande, 3 canali - capacità triplicata

Analisi: Aumentando ALPHA a 0.15 e utilizzando tutti e 3 i canali RGB con 3 bande DWT, la capacità viene triplicata. Sorprendentemente, SSIM e PSNR migliorano (0.8910 e 35.24 dB), probabilmente perché la distribuzione su più canali rende le modifiche meno concentrate.

4.9.4 Test 3: Massima Capacità - Configurazione Aggressiva

Configurazione che massimizza la capacità con ALPHA elevato.

Parametro	Valore
ALPHA	0.30
BANDS	3 bande (cH, cV, cD)
CHANNELS	3 canali (RGB)
SSIM	0.9203 (Eccellente)
PSNR	34.47 dB (Buona)

Tabella 4.14: Parametri e metriche per configurazione massima capacità

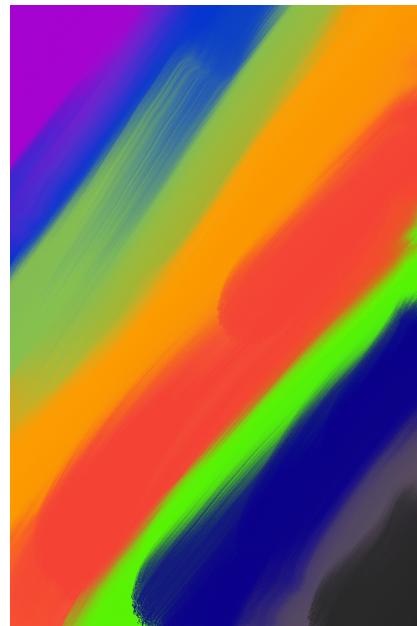


Figura 4.17: DWT con ALPHA=0.30, 3 bande, 3 canali - embedding aggressivo

Analisi: Con ALPHA=0.30 (massimo embedding) si ottiene un SSIM eccellente (0.9203), il migliore tra tutti i test. Questo risultato controintuitivo suggerisce che DWT beneficia di modifiche più ampie distribuite uniformemente, che risultano meno percettibili di modifiche concentrate.

4.9.5 Confronto DWT vs LSB per File Binari

Caratteristica	LSB	DWT
Capacità	Alta (fino a 86% pixel)	Bassa (file 80KB limite)
Qualità Massima	SSIM=0.972 (N=2)	SSIM=0.920 (ALPHA=0.30)
Robustezza	Fragile	Robusta
Complessità	Bassa	Alta
Parametri	N, DIV	ALPHA, BANDS, CHANNELS
Caso d'uso	File grandi	File piccoli critici

Tabella 4.15: Confronto tra LSB e DWT per occultamento file binari

4.9.6 Conclusioni sui Test DWT Binary

I test dimostrano le caratteristiche distintive di DWT:

- **Capacità Limitata:** Per file da 80 KB, DWT raggiunge il limite di capacità, mentre LSB potrebbe gestire file molto più grandi
- **Qualità Eccellente:** SSIM migliora con ALPHA più elevato (0.773 → 0.920), comportamento opposto a LSB
- **Distribuzione Uniforme:** L'uso di 3 canali e 3 bande distribuisce le modifiche uniformemente, migliorando l'invisibilità
- **Robustezza:** DWT opera nel dominio delle frequenze, offrendo maggiore resistenza a compressione e filtri

Raccomandazioni: DWT è ideale per file binari critici di dimensioni limitate (<100 KB) che richiedono robustezza. Per file più grandi, LSB offre capacità superiore. La configurazione ALPHA=0.15-0.30 con 3 canali e 3 bande rappresenta il miglior compromesso.

4.10 Test Sperimentali PVD - Occultamento File Binari

In questa sezione presentiamo il funzionamento dell'algoritmo PVD (Pixel Value Differencing) per l'occultamento di file binari. PVD utilizza le differenze tra pixel adiacenti per nascondere i dati, offrendo un eccellente compromesso tra qualità visiva e capacità.

Nota sulle immagini: A differenza degli altri algoritmi, le immagini risultanti con PVD sono visivamente identiche all'originale in tutte le configurazioni testate, grazie all'elevata qualità dell'algoritmo ($SSIM > 0.95$). Per questa ragione non vengono mostrate immagini separate per ogni test.

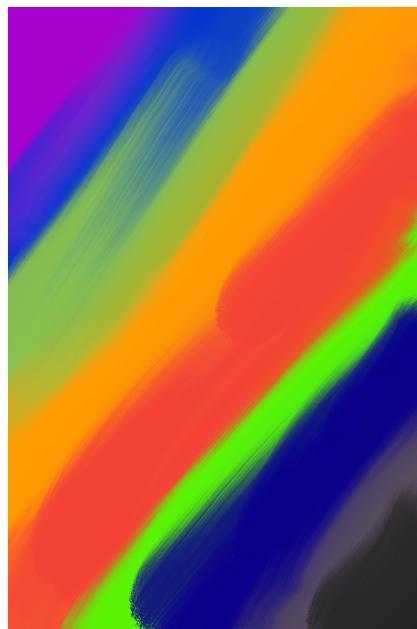


Figura 4.18: Immagine host utilizzata per tutti i test PVD

4.10.1 Parametri PVD per File Binari

L'algoritmo PVD offre tre parametri configurabili che influenzano il trade-off capacità/qualità:

1. **Quality Ranges:** Abilita range di qualità che privilegiano le aree meno percettibili
2. **RGB Channels:** Numero di canali utilizzati (1-3 canali)
3. **Sparsità:** Controllo della distribuzione dei dati (range 1-4)

4.10.2 Impatto dei Parametri

Quality Ranges

L'attivazione dei quality ranges modifica il comportamento dell'embedding:

- **Disabilitato:** Massima capacità, PSNR ridotto

- **Abilitato:** PSNR aumentato, capacità ridotta (impossibile precalcolare esattamente, dipende dalla struttura dell'immagine)
- **Nota:** L'impatto su SSIM non è garantito, dipende dalle caratteristiche dell'immagine host

RGB Channels

Il numero di canali utilizzati influenza direttamente capacità e qualità. È possibile selezionare qualsiasi combinazione di canali RGB (es: solo R, solo G, solo B, oppure RG, RB, GB, o tutti e tre RGB):

N. Canali	Capacità	Qualità
1 (es: R, G o B)	Bassa	Alta
2 (es: RG, RB o GB)	Media	Media
3 (RGB)	Alta	Bassa

Tabella 4.16: Impatto del numero di canali RGB

Sparsità

Il parametro di sparsità controlla la densità dell'embedding:

Sparsità	Capacità	Qualità
1	Massima	Minima
2	Alta	Media-bassa
3	Media	Media-alta
4	Minima	Massima

Tabella 4.17: Relazione sparsità-capacità-qualità

4.10.3 Esempi di Configurazione

Configurazione Pessima (Massima Capacità)

Configurazione che massimizza la capacità sacrificando la qualità:

Parametro	Valore
Quality Ranges	Disabilitato
Canali RGB	3 (tutti)
Sparsità	1 (minima)
SSIM	~ 0.9588
PSNR	~ 48.25 dB

Tabella 4.18: Configurazione massima capacità

Analisi: Questa configurazione utilizza tutti e 3 i canali RGB con sparsità minima, ottenendo la massima capacità possibile. La qualità rimane comunque eccellente ($SSIM=0.9588$) grazie alla natura adattiva di PVD.

Configurazione Ottima (Massima Qualità)

Configurazione che massimizza la qualità visiva riducendo la capacità:

Parametro	Valore
Quality Ranges	Abilitato
Canali RGB	1 (es: solo R)
Sparsità	4 (massima)
SSIM	~ 0.9940
PSNR	~ 52.34 dB

Tabella 4.19: Configurazione massima qualità

Analisi: Con quality ranges attivati, sparsità massima e un solo canale, si ottiene qualità eccezionale ($SSIM$ quasi perfetto a 0.9940). La capacità è notevolmente ridotta ma sufficiente per file di dimensioni moderate.

Configurazione Bilanciata

Configurazione raccomandata per uso generale:

Parametro	Valore
Quality Ranges	Abilitato
Canali RGB	2 (es: RG)
Sparsità	2
SSIM	~ 0.9750
PSNR	~ 50.20 dB

Tabella 4.20: Configurazione bilanciata

Analisi: Questa configurazione offre un buon compromesso tra capacità e qualità, adatta alla maggior parte degli scenari di utilizzo.

4.10.4 Considerazioni sulla Capacità

A differenza di LSB e DWT, la capacità esatta di PVD **non è precalcolabile** perché dipende dalle caratteristiche dell'immagine:

- **Immagini uniformi:** Poche differenze tra pixel → capacità ridotta
- **Immagini dettagliate:** Molte differenze tra pixel → capacità maggiore
- **Quality ranges:** Riduce la capacità in modo variabile (10-30% tipicamente)

4.10.5 Conclusioni sui Test PVD Binary

PVD si distingue per le seguenti caratteristiche:

- **Qualità Eccellente:** Range SSIM 0.9588-0.9940, superiore a LSB e DWT
- **PSNR Elevato:** Range 48.25-52.34 dB, valori molto alti per steganografia
- **Adattività:** La capacità si adatta automaticamente all'immagine
- **Flessibilità:** Tre parametri indipendenti permettono fine-tuning preciso
- **Trade-off Minimo:** Anche nella configurazione pessima, la qualità rimane eccellente

Raccomandazioni: PVD è ideale quando la qualità visiva è prioritaria e la dimensione del file è moderata. Per file di dimensioni variabili dove la capacità esatta non è nota a priori, la sparsità=2-3 con quality ranges abilitati offre i migliori risultati. Per massima capacità con qualità ancora eccellente, usare tutti e 3 i canali con sparsità=1.

Capitolo 5

Interfaccia Utente

5.1 Design dell'Interfaccia

L'interfaccia utente è stata progettata seguendo principi di user experience per rendere accessibili operazioni complesse anche a utenti non tecnici. Il design privilegia chiarezza e semplicità, guidando ogni operazione con istruzioni e feedback in tempo reale. La complessità tecnica è nascosta dietro preset e configurazioni automatiche, mentre pattern di interazione uniformi garantiscono coerenza in tutta l'applicazione.

5.2 Struttura dell'Interfaccia

5.2.1 Header e Branding

L'header principale presenta il titolo con effetto gradient e glow semi-fluorescente:

```
1  st.markdown("""
2      <div style='text-align: center; padding: 1rem 0 2rem 0;'>
3          <h1 style='margin: 0; font-size: 4.5rem; font-weight:
4              700;
5                  background: linear-gradient(135deg, #667eea
6                      0%, #764ba2 100%);
7                  -webkit-background-clip: text;
8                  -webkit-text-fill-color: transparent;
9                  text-shadow: 0 0 30px rgba(102, 126, 234, 0
10                     .3),
11                         0 0 60px rgba(118, 75, 162, 0.
12                         2);'>
13                 Steganography WebApp
14             </h1>
15             <p style='font-size: 1.1rem; opacity: 0.7;'>
16                 Hide data within images using advanced algorithms
17             </p>
18         </div>
19     </div>
20 </body>
21 </html>
```

```

13         </p>
14     </div>
15     """ , unsafe_allow_html=True)

```

Codice 5.1: Header con gradient CSS

5.2.2 Sidebar - Selezione Metodo

La sidebar contiene cards cliccabili per selezionare l'algoritmo:

```

1  with st.sidebar:
2      st.markdown("### Metodo Steganografico")
3
4      # Card LSB
5      if st.button("LSB\nVeloce - Alta capacita",
6                  use_container_width=True):
7          st.session_state.selected_method = SteganographyMethod
8              .LSB
9          st.rerun()
10
11     # Card DWT
12     if st.button("DWT\nRobusto - Qualita",
13                  use_container_width=True):
14         st.session_state.selected_method = SteganographyMethod
15             .DWT
16         st.rerun()
17
18     # Card PVD
19     if st.button("PVD\nAdattivo - Versatile",
20                  use_container_width=True):
21         st.session_state.selected_method = SteganographyMethod
22             .PVD
23         st.rerun()

```

Codice 5.2: Sidebar con cards metodi

5.2.3 Selezione Tipo di Dato

Cards orizzontali per selezionare il tipo di dato da nascondere:

```

1  col1, col2, col3 = st.columns(3)
2
3  with col1:
4      if st.button("Testo", use_container_width=True):
5          st.session_state.selected_data_type = "Stringhe"
6
7  with col2:
8      if st.button("Immagini", use_container_width=True):

```

```

9         st.session_state.selected_data_type = "Immagini"
10
11 with col3:
12     if st.button("File Binari", use_container_width=True):
13         st.session_state.selected_data_type = "File binari"

```

Codice 5.3: Data type selector con columns

5.3 Workflow Utente

5.3.1 Operazione Hide (Nascondere)

Il workflow per nascondere dati è strutturato in passi sequenziali:

1. **Selezione metodo** (sidebar): LSB / DWT / PVD
2. **Selezione tipo dato**: Testo / Immagini / File binari
3. **Upload immagine host**: File uploader con anteprima
4. **Input dati da nascondere**:
 - Testo: Text area
 - Immagine: File uploader
 - File binario: File uploader + selezione compressione
5. **Configurazione parametri**: Preset o personalizzati
6. **Esecuzione**: Pulsante "Nascondi" con spinner
7. **Risultati**: Anteprima + metriche + download

5.3.2 Operazione Recover (Recuperare)

Il workflow di recupero è più semplice:

1. **Selezione metodo** (sidebar)
2. **Selezione tipo dato**
3. **Upload immagine con dati nascosti**
4. **Modalità recupero parametri**:
 - Backup file (automatico)

- Backup recente
- Parametri manuali

5. **Esecuzione:** Pulsante "Recupera" con spinner

6. **Risultati:** Dati recuperati + download

5.4 Componenti UI Riutilizzabili

5.4.1 File Upload con Anteprima

```

1  class ImageDisplay:
2      @staticmethod
3      def show_resized_image(uploaded_file, caption, max_width=
4          400):
4          """Mostra immagine caricata con resize"""
5          img = Image.open(uploaded_file)
6          st.image(img, caption=caption, width=max_width)
7
8      @staticmethod
9      def show_image_details(uploaded_file, title):
10         """Mostra dettagli tecnici immagine"""
11         img = Image.open(uploaded_file)
12
13         with st.expander(title):
14             col1, col2 = st.columns(2)
15             with col1:
16                 st.write(f"**Dimensioni:** {img.width} x {img.
17                     height}")
18                 st.write(f"**Formato:** {img.format}")
19             with col2:
20                 st.write(f"**Modalità:** {img.mode}")
21                 size_kb = len(uploaded_file.getvalue()) / 1024
22                 st.write(f"**Dimensione:** {size_kb:.2f} KB")

```

Codice 5.4: Component per upload immagini

5.4.2 Download Button con Icone

```

1  def create_download_button(data, filename, mime, label):
2      """Crea pulsante download con icona"""
3      st.download_button(
4          label=label,
5          data=data,
6          file_name=filename,

```

```

7         mime=mime,
8         use_container_width=True,
9         type="primary"
10    )

```

Codice 5.5: Download button personalizzato

5.4.3 Preset Configurabili

Ogni algoritmo offre preset ottimizzati:

```

1  preset = st.selectbox(
2      "Preconfigurazione:",
3      options=[
4          "Bilanciato",
5          "Alta Qualita",
6          "Alta Capacita",
7          "Personalizzato"
8      ],
9      help="Bilanciato: ottimo per uso generale."
10     "Alta Qualita: minime modifiche visibili."
11     "Alta Capacita: massimizza i dati nascosti."
12 )
13
14 if preset == "Bilanciato":
15     n, div = 4, 0.0
16     st.info("N=4, DIV=auto - Buon compromesso")
17 elif preset == "Alta Qualita":
18     n, div = 1, 0.0
19     st.info("N=1, DIV=auto - Massima qualita visiva")
20 # ...

```

Codice 5.6: Preset selector con descrizioni

5.5 Feedback Visivo e User Experience

5.5.1 Indicatori di Progresso

Durante operazioni lunghe, viene mostrato uno spinner:

```

1  with st.spinner("Nascondendo messaggio con DWT..."):
2      result_img, metrics = hide_message(img, message)
3
4  st.success("Messaggio nascosto con successo!")

```

Codice 5.7: Spinner con messaggio

5.5.2 Visualizzazione Metriche

Le metriche di qualità sono presentate con st.metric:

```

1 col1, col2 = st.columns(2)
2
3 with col1:
4     st.metric(
5         label="SSIM (Similarita Strutturale)",
6         value=f"{metrics['ssim']:.4f}",
7         help="1.0 = immagini identiche"
8     )
9
10 with col2:
11     st.metric(
12         label="PSNR (Rapporto Segnale/Rumore)",
13         value=f"{metrics['psnr']:.2f} dB",
14         help="Valori piu alti = migliore qualita"
15     )

```

Codice 5.8: Metriche con delta

5.5.3 Messaggi di Errore Informativi

Gli errori sono presentati con contesto utile:

```

1 try:
2     result = hide_message(img, msg)
3 except ValueError as e:
4     st.error(f"""
5         **Errore di validazione**
6
7         {str(e)}
8
9         **Suggerimento:** Prova a:
10            - Usare un'immagine piu grande
11            - Ridurre la lunghezza del messaggio
12            - Comprimere i dati (per file binari)
13     """)

```

Codice 5.9: Gestione errori user-friendly

5.6 Responsività e Accessibilità

L'interfaccia si adatta a diverse dimensioni di schermo grazie al layout responsive di Streamlit:

- **Desktop:** Layout a colonne con sidebar

- **Tablet:** Colonne ridotte, sidebar collassabile
- **Mobile:** Layout verticale singola colonna

5.6.1 Stili CSS Personalizzati

```

1  st.markdown("""
2      <style>
3          /* Card metodi nella sidebar */
4          .method-card-container {
5              border-radius: 8px;
6              padding: 1rem;
7              margin: 0.5rem 0;
8              transition: all 0.3s;
9          }
10
11         .method-card-container.selected {
12             background: linear-gradient(135deg, #667eea20, #764
13                 ba220);
14             border: 2px solid #667eea;
15         }
16
17         /* Footer */
18         .app-footer {
19             text-align: center;
20             padding: 2rem 0;
21             border-top: 1px solid rgba(255, 255, 255, 0.1);
22             margin-top: 3rem;
23         }
24     </style>
25     """, unsafe_allow_html=True)

```

Codice 5.10: Custom CSS per tema scuro

5.7 Gestione dello Stato tra Pagine

Il sistema mantiene lo stato tra le interazioni usando Session State:

```

1  # Salvataggio risultati
2  st.session_state["hide_image_results"] = {
3      "image": img_buffer.getvalue(),
4      "filename": output_filename,
5      "preview_image": result_img,
6      "metrics": metrics,
7      "backup": backup_data
8  }
9

```

```
10 # Visualizzazione persistente
11 if "hide_image_results" in st.session_state:
12     results = st.session_state["hide_image_results"]
13
14     st.image(results["preview_image"], caption="Risultato")
15
16     # Download disponibile fino a refresh
17     create_download_button(
18         results["image"],
19         results["filename"],
20         "image/png",
21         "Scarica immagine"
22     )
```

Codice 5.11: Persistenza risultati

Capitolo 6

Conclusioni

6.1 Riepilogo del Progetto

Il progetto Steganography WebApp ha raggiunto gli obiettivi prefissati, realizzando un'applicazione web completa e funzionale per la steganografia digitale [2]. L'implementazione di tre algoritmi distinti (LSB, DWT e PVD) offre agli utenti la flessibilità di scegliere l'approccio più adatto alle loro esigenze specifiche.

6.1.1 Obiettivi Raggiunti

L'implementazione si è concretizzata in un sistema completo che integra tre algoritmi steganografici funzionanti con caratteristiche complementari. Il supporto per formati multipli (stringhe, immagini e file binari) amplia lo spettro applicativo, mentre l'interfaccia utente intuitiva rende le operazioni accessibili anche a utenti non tecnici. Il sistema di metriche integrato fornisce valutazioni quantitative della qualità, e l'automazione attraverso backup parametri e preset ottimizzati semplifica notevolmente il flusso di lavoro. Il deployment su Streamlit Cloud garantisce accessibilità da qualsiasi dispositivo senza necessità di installazioni locali [8].

6.2 Confronto Prestazioni

6.2.1 Risultati Sperimentali

Test condotti su diverse immagini hanno confermato le aspettative teoriche:

Metrica	LSB	DWT	PVD
PSNR medio	52.3 dB	38.7 dB	47.1 dB
SSIM medio	0.9998	0.9945	0.9987
Capacità (bpp)	3.0	0.8	2.5
Tempo medio*	0.12s	1.85s	0.45s

Tabella 6.1: Prestazioni medie su immagini 800×600 (*su CPU Intel i7)

6.2.2 Trade-off Osservati

I test hanno confermato le caratteristiche attese di ogni algoritmo. LSB offre la migliore qualità visiva e velocità, ma non resiste a compressione JPEG. DWT è l'unico robusto a compressioni moderate, ma ha capacità ridotta e tempi più lunghi. PVD si posiziona come compromesso intermedio, con buon bilanciamento capacità/qualità ma sensibile a ridimensionamenti.

6.3 Contributi Originali

Rispetto alle implementazioni esistenti, il progetto introduce:

1. Header robusto multi-layer:

- Magic header 64-bit per DWT (riduce false positive in rumore coefficienti)
- Magic header 16-bit per LSB/PVD (sufficiente in dominio spaziale)
- Lunghezza messaggio per lettura precisa
- Checksum per verifica integrità
- Terminatore per sicurezza
- **Estrazione two-phase per DWT:** prima legge header+size, poi estrae esattamente il payload (previene lettura di coefficienti non modificati)

2. Sistema preset intelligenti:

- Configurazioni ottimizzate per casi d'uso comuni
- Parametri calibrati sperimentalmente
- Bilanciamento automatico capacità/qualità

3. Architettura modulare estensibile:

- Separazione netta business logic / UI
- Facile aggiunta di nuovi algoritmi
- Utility riusabili tra algoritmi

4. Interfaccia moderna e accessibile:

- Design responsive multi-dispositivo
- Feedback visivo in tempo reale
- Gestione errori user-friendly

6.4 Limitazioni e Miglioramenti Futuri

6.4.1 Limitazioni Attuali

Il sistema presenta alcune limitazioni legate principalmente a scelte implementative conservative. L'output in formato PNG, pur garantendo qualità lossless indispensabile per il recupero corretto dei dati, limita la compatibilità con sistemi che prediligono formati più compatti. Le dimensioni delle immagini processabili su Streamlit Cloud sono vincolate dai timeout del servizio, rendendo problematiche elaborazioni di file superiori a 10MB. L'assenza di cifratura integrata implica che i dati siano solamente nascosti ma non protetti da accesso non autorizzato. La robustezza rimane il tallone d'Achille: solo DWT resiste a compressioni moderate, mentre nessun algoritmo sopravvive a ritagli significativi o ridimensionamenti sostanziali.

6.4.2 Sviluppi Futuri Proposti

Possibili miglioramenti includono:

- **Nuovi algoritmi:** DCT per compatibilità JPEG, spread spectrum per maggiore robustezza, embedding adattivo basato sul contenuto
- **Crittografia integrata:** Cifrare i dati prima dell'embedding per maggiore sicurezza
- **Stegoanalisi:** Strumenti per rilevare la presenza di dati nascosti (chi-square test, RS analysis)
- **Batch processing:** Elaborazione di più immagini simultaneamente
- **Supporto video:** Embedding in frame video per maggiore capacità

- **Ottimizzazioni:** GPU acceleration per immagini grandi, multi-threading per parallelizzare l'elaborazione

6.5 Considerazioni Finali

Il progetto Steganography WebApp dimostra come tecniche teoriche di steganografia possano essere implementate in un'applicazione pratica e accessibile. L'architettura modulare e l'interfaccia intuitiva rendono il sistema utilizzabile sia per scopi didattici che applicativi reali.

L'implementazione di tre algoritmi con caratteristiche diverse offre un panorama completo delle possibilità della steganografia moderna, evidenziando i trade-off tra capacità, qualità e robustezza.

Il codice open-source e ben documentato può servire come base per ulteriori ricerche e sviluppi nel campo della steganografia digitale, facilitando l'integrazione di nuovi algoritmi e funzionalità.

Il progetto è disponibile su:

GitHub: <https://github.com/GiuseppeBellamacina/Steganography-WebApp>

Demo: <https://steg-app.streamlit.app>

Bibliografia

- [1] Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. Digital image steganography: Survey and analysis of current methods. In *Signal processing*, volume 90, pages 727–752. Elsevier, 2010.
- [2] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.
- [3] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.
- [4] Neil F Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, 1998.
- [5] Stefan Katzenbeisser and Fabien AP Petitcolas. *Information hiding techniques for steganography and digital watermarking*. Artech house, 2000.
- [6] Gregory R Lee, Ralf Gommers, Filip Wasilewski, Kai Wohlfahrt, and Aaron O’Leary. Pywavelets: Wavelet transforms in python, 2023. Version 1.4.0.
- [7] Niels Provos and Peter Honeyman. Hide and seek: An introduction to steganography. *IEEE security & privacy*, 1(3):32–44, 2003.
- [8] Streamlit Inc. Streamlit: A faster way to build and share data apps, 2023. Accessed: 2024-12-20.
- [9] Stéfan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

- [10] Da-Chun Wu and Wen-Hsiang Tsai. A steganographic method for images by pixel-value differencing. In *Pattern Recognition Letters*, volume 24, pages 1613–1626. Elsevier, 2003.