

Steganography and Watermarks Report

W tej analizie sprawdziłem, jak ukrywanie informacji oraz nakładanie znaku wodnego wpływa na jakość obrazu nośnika i możliwość poprawnego odzyskania danych. Testowałem klasyczną steganografię tekstu i obrazu, znak wodny oraz dekonstrukcję obrazu zapisując losowe wartości do różnych bitów, oceniając jakość za pomocą PSNR i SSIM. Wnioski opisane są w każdym ze scenariuszy testowych.

Steganography Hidden Text

Blue Channel



Blue w/ Text



Channel

PSNR

SSIM

Blue

51.16

0.9975

```
 1  rec_text = rec_bytes.decode('utf-8', errors='ignore')
 2  print(f'Original text: {text}')
 1  print(f'Recovered text: {rec_text}')
92  print(f'Match? {text == rec_text}')
 1  psnr_val = psnr(B, B_encoded, data_range=255)

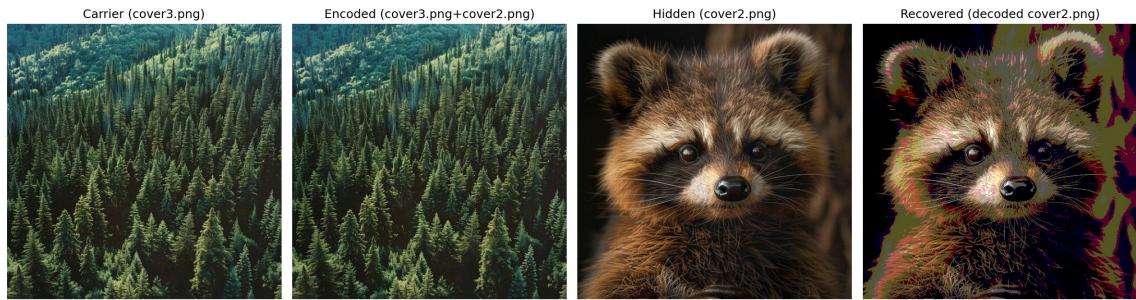
1: vladislav@svstk:~/Documents/college-CS/semester-6/multimedia-systems/lab11
!
[A: ~] ~/Documents/college-CS/semester-6/lab11 on master ! 2105 py main.py      took 4s ✘ multimedia-systems 2.43 MiB 186 ↗ at 06:20:53 ⚡ 79% 87.8 Mbps
Original text: The Constitution is not an instrument for the government to restrain the people; it is an instrument for the people to restrain the government. Liberty cannot be preserved without a general knowledge among the people.

Recovered text: The Constitution is not an instrument for the government to restrain the people; it is an instrument for the people to restrain the government. Liberty cannot be preserved without a general knowledge among the people.

Match? True
```

Tu tekst został zakodowany w najmłodszym bicie kanału niebieskiego obrazu. Operacja nie miała żadnego zauważalnego wpływu na jakość obrazu, co potwierdzają metryki. Tekst został w pełni poprawnie odzyskany. Wniosek: metoda jest skuteczna zarówno pod względem bezpieczeństwa wizualnego, jak i poprawności przesyłu danych tekstowych.

Steganography Hidden Image



Kanał	PSNR	SSIM
R	44.62	0.9959
G	44.19	0.9962
B	38.28	0.9818

Tu obraz cover2.png został ukryty w cover3.png przez zapisanie kolejno 3, 2 i 2 bitów do kanałów niebieskiego, czerwonego i zielonego nośnika. Przed zakodowaniem ukryty obraz został poddany kwantyzacji. Jak widać, nośnik praktycznie nie został uszkodzony, co potwierdzają metryki. Odzyskany obraz jest nieco zdegradowany przez kwantyzację, ale nadal czytelny. Wniosek: nawet przy takiej liczbie użytych bitów metoda nie pogarsza jakości nośnika i pozwala skutecznie ukryć obraz.

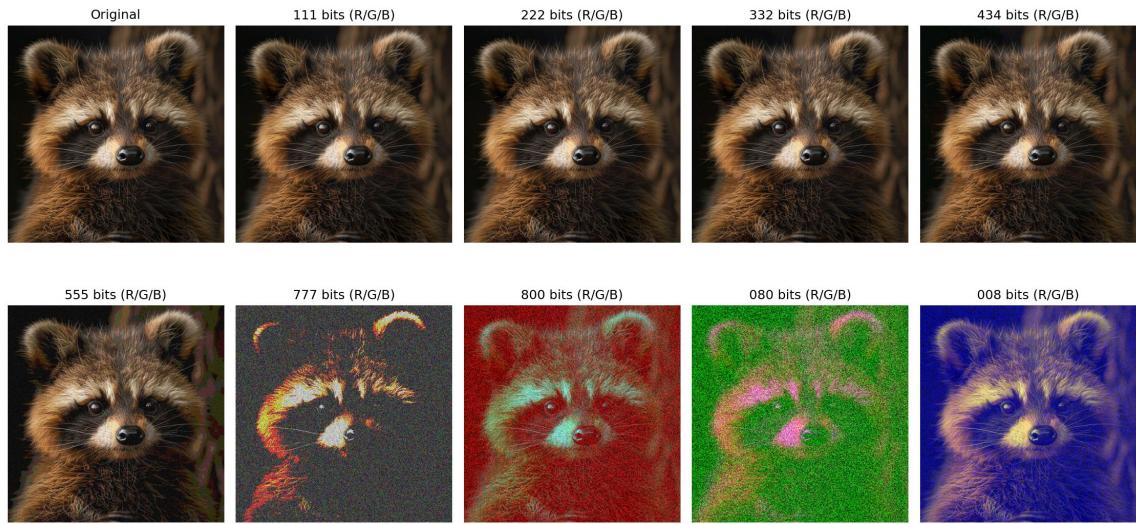
Watermark



Alpha	PSNR	SSIM
0.10	22.62	0.9275
0.25	14.88	0.8349
0.50	9.82	0.7144

Tu nakładałem znak wodny – logo GitHub – na obraz dla różnych wartości alpha. Dla alpha = 0.1 logo jest słabo widoczne, zaś dla większych wartości alpha już jest wyraźnie widoczne, a obraz staje się prześwietlony, co odzwierciedlażą także metryki (zwłaszcza warto zauważyć, że nawet przy alpha = 0.1 PSNR był już relatywnie niski)

Image Deconstruction



Bits (R/G/B)	PSNR	SSIM
1/1/1	51.14	0.9974
2/2/2	44.15	0.9873
3/3/2	39.19	0.9646
4/3/4	33.09	0.8838
5/5/5	25.99	0.6405
7/7/7	13.88	0.0897
8/0/0	12.36	0.6693
0/8/0	11.94	0.6686
0/0/8	11.37	0.6687

Tutaj przeprowadziłem badanie wpływu modyfikacji różnej liczby bitów (również LSB) na obraz, by określić optymalny budżet bitowy. Pierwsze cztery warianty, czyli użycie odpowiednio 3, 6 i 8 bitów w zrównoważonych proporcjach pomiędzy kanałami miały dość niski wpływ na jakość obrazu ($SSIM > 0.96$). Nawet zestaw 4/3/4 – czyli 10 bitów – nie miał poważnego wpływu, i również nie widać tego wizualnie. Dopiero 5/5/5 ma już niskie metryki i widać kwantyzacje kolorów. 7/7/7 całkowicie zniekształcił obraz, co potwierdza bardzo niska wartość SSIM - 0.09. Ciekawie, że pełna zamiana bitów na jednym z kanałów miała taki sam wskaźnik SSIM co 5/5/5, a jednak dość widać, że cały obraz jest zaszumiony jednym kolorem – dlatego też różni się tu PSNR: 26 vs ~12. Warto zauważyć, że tutaj użyte

były losowe liczby, więc ukrycie rzeczywistego obrazu w jednym kanale mogłoby mieć inny efekt.