Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Luka Kovačič

METODE PREKRIVANJA INFORMACIJE V SLIKAH PNG

Diplomsko delo

Maribor, avgust 2021

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Luka Kovačič

METODE PREKRIVANJA INFORMACIJE V SLIKAH PNG

Diplomsko delo

Maribor, avgust 2021

METODE PREKRIVANJA INFORMACIJE V SLIKAH PNG

Diplomsko delo

Študent:	Luka Kovačič
Študijski program:	univerzitetni študijski program
	Računalništvo in informacijske tehnologije
Mentor:	red. prof. dr. Borut Žalik, univ. dipl. inž. el.
Somentor:	asist. dr. David Jesenko, mag. inž. rač. In inf. tehnol.

ZAHVALA

Zahvaljujem se mentorju, red. prof. dr. Borutu Žaliku, in somentorju, asist. dr. Davidu Jesenku, za strokovno pomoč pri izdelavi diplomskega dela. Posebna zahvala je namenjena staršem, ki so mi omogočili študij.

Metode prekrivanja informacije v slikah PNG

Ključne besede: steganografija, prostorska domena, rastrske slike, metode LSB, LSBMR, PVD, PPV, EMD

UDK: 004.353.244(043.2)

Povzetek

V diplomskem delu najprej predstavimo pomembne aspekte slikovnega formata PNG, kjer razložimo koncept prepletanja in filtre PNG. Nato se osredotočimo na metode za prekrivanje informacije. Osredotočimo se na metode LSB, LSBMR, PVD, PPV in EMD. Metode implementiramo in jih nato med seboj primerjamo.

Information hiding methods in PNG images

Keywords: steganography, spatial domain, raster images, methods LSB, LSBMR, PVD, PPV, EMD

UDC: 004.353.244(043.2)

Abstract

In this thesis first important aspects of PNG image format are explained; concept of interlacing and PNG filters. Then focus is on information hiding methods, especially on methods LSB, LSBMR, PVD, PPV and EMD. Methods are also implemented and compared.

KAZALO VSEBINE

KAZALO SLIK VI

KAZALO TABEL VII

1 UVOD 1

2 PREDSTAVITEV FORMATA PNG 2

2.1 Prepletanje 2

2.2 Stiskanje 4

2.2.1 Filtri 4

3 METODE PREKRIVANJA INFORMACIJE 7

3.1 Metoda LSB 7

3.1.1 Metoda LSBMR 8

3.2 Metoda PVD 10

3.3 Napovedna metoda PPV 14

3.4 Metoda EMD 16

3.4.1 Nadgradnja EMD 17

3.5 Metoda z zaznavo robov 18

4 REZULTATI 21

4.1 Slika Lenna 23

4.2 Slika Baboon 28

4.3 Umetno generirana slika 32

4.4 Risana slika 36

5 ZAKLJUČEK 40

VIRI IN LITERATURA 41

KAZALO SLIK

Slika 2.1: Sedem korakov prepletanja 3
Slika 2.2: Izračun napovedi 6
Slika 3.1: Razlika enega bita na kanal je neopazna 8
Slika 3.2: Piksli, sodelujoči pri napovednem modelu MED....14
Slika 3.3: Sobelova strukturna elementa....18
Slika 3.4: Zaznava navpičnih in vodoravnih robov 19
Slika 3.5: Problem pri metodi z zaznavo robov....20
Slika 4.1: Primerjava slik pri metodi LSB....23
Slika 4.2: Primerjava slik pri metodi PVD....24
Slika 4.3: Spremenjeni piksli za več kot 4 bite pri metodi PVD....24
Slika 4.4: Primerjava slik pri metodi PPV....25
Slika 4.5: Primerjava slik pri metodi LSBMR 26
Slika 4.6: Primerjava slik pri metodi LSB....28
Slika 4.7: Primerjava slik pri metodi PVD z večjo kapaciteto....29
Slika 4.8: Primerjava slik pri metodi LSBMR 29
Slika 4.9: Primerjava slik pri metodi PPV....30
Slika 4.10: Primerjava slik pri metodi LSB....32
Slika 4.11: Primerjava slik pri metodi PVD....33
Slika 4.12: Primerjava slik pri metodi LSBMR 33
Slika 4.13: Primerjava slik pri metodi PPV....34
Slika 4.14: Primerjava slik pri metodi LSB....36
Slika 4.15: Primerjava slik pri metodi PVD....37
Slika 4.16: Primerjava slik pri metodi LSBMR 37
Slika 4.17: Primerjava slik pri metodi PPV....38

KAZALO TABEL

Tabela 2.1: Vrstni red pridobivanja vrednosti pikslov bloka 3
Tabela 2.2: Pregled filtrov.....4
Tabela 2.3: Odvisnost Bpp od tipa slike.....6
Tabela 3.1: Kategorije uvrščanja blokov.....11
Tabela 4.1: Primerjava rezultatov za sliko Lenna.....27
Tabela 4.2: Primerjava rezultatov za sliko Baboon 31
Tabela 4.3: Primerjava rezultatov za umetno generirano sliko.....35
Tabela 4.4: Primerjava rezultatov za risano sliko 39

UPORABLJENI SIMBOLI IN KRATICE

EMD – (angl. Exploiting Modification Direction)

JPEG – (angl. Joint Photographic Experts Group)

LSB – (angl. Least Significant Bit)

LSBMR – (angl. Least Significant Bit Matching Revisited)

MED – (angl. Median Edge Detection)

PNG – (angl. Portable Network Graphics)

PSNR – (angl. Peak Signal-To-Noise Ratio)

PPV – (angl. Predictive Pixel Value)

PVD – (angl. Pixel Value Differencing)

RGB – barvni model, sestavljen iz rdeče (angl. red), zelene (angl. green) in modre (angl. blue) barve

SSIM – (angl. Structural Similarity Index Measure)

1 UVOD

Pošiljanje prikritih sporočil med pošiljateljem in prejemnikom ima že dolgo zgodovino. V svetovnem merilu je to vsekakor prišlo do izraza med obema svetovnima vojnama, kjer je bilo ključno, da sovražnik ni uspel razvozlati sporočila, četudi ga je na nek način prestregel. Tako imenovana šifrirana sporočila so sama po sebi vzbujala ogromno pozornosti, saj je sovražnik vedel, da se v njih (morda) nahaja pomembna informacija. Bolje je, če sovražnik sploh ne ve, da se informacija nekje nahaja. Pravimo, da je informacija prikrita oziroma steganografirana. S to tematiko se ukvarja steganografija [1], ki ima torej bistveno prednost v primerjavi s kriptografijo. Glavni namen steganografije je premagan že, če se ugotovi, da nek medij prenaša prekrite podatke. V kolikor napadalec odkrije, na kakšen način so podatki skriti, lahko dostopa do le teh. Posledično je dobro, da steganografijo in kriptografijo združimo tako, da podatke pred prekrivanjem še šifriramo.

Metode za prekrivanje informacije oziroma steganografske tehnike v grobem ločimo na fizične in digitalne. Fizične so bile aktualne predvsem pred digitalizacijo oziroma razvojem informacijsko-komunikacijskih tehnologij. Primer fizične tehnike je uporaba nevidnega črnila. Med digitalne tehnike spadajo metode prekrivanja podatkov oziroma informacij v popularne vsakdanje multimedijske tipe, to so v glavnem slike, digitalni avdio in video posnetki. S stališča steganografije imajo slednji pomembno skupno lastnost – informacijsko redundanco. Na primer, manjših sprememb na sliki naše oči ne bodo zaznale, kar je idealno za uporabo slik kot prenosni medij.

V diplomskem delu se osredotočimo na zelo popularen brezizgubni format za shranjevanje rastrskih slik PNG. Slike PNG bomo uporabili kot medij oziroma nosilec prekritih podatkov, ki jih bomo vstavili z različnimi metodami prekrivanja informacij.

Diplomsko delo sestoji iz petih poglavij. V drugem poglavju predstavimo postopek prikazovanja slike PNG s prepletanjem in filtre PNG. V tretjem poglavju obravnavamo različne metode prekrivanja informacij, ki jih lahko uporabimo nad slikami PNG. V četrtem poglavju metode testiramo nad različnimi tipi slik PNG in jih med seboj primerjamo. V poglavju 5 delo povzamemo.

2 PREDSTAVITEV FORMATA PNG

Brezizgubni format PNG za stiskanje in shranjevanje podatkov o rastrskih slikah je bil zasnovan kot odgovor na patentno zaščiten format GIF. GIF uporablja algoritem stiskanja LZW, ki pa je bil takrat še zaščiten s patentom [2]. PNG je nastal leta 1995 in spada med najbolj popularne brezizgubne formate.

PNG omogoča shranjevanje sledečih slik [3]:

- črno-bele;
- sivinske z 2^8 ali 2^{16} odtenki sivine;
- slike, ki uporabljajo 1-8 bitno paleto;
- RGB z 2^{24} ali 2^{48} barvami.

Med drugim PNG podpira tudi prosojnost z dodatnim kanalom alfa.

2.1 Prepletanje

Prepletanje (angl. interlacing) je pomembnem del formata PNG, saj je včasih slika prevelika ali pa je npr. internetna povezava prešibka, da bi se slika v trenutku prikazala na prikazovalnik. Zato je pomembno, da ima format izbiro prepletanja, saj lahko hitro pokažemo grobo sliko, kar izboljša uporabniško izkušnjo.

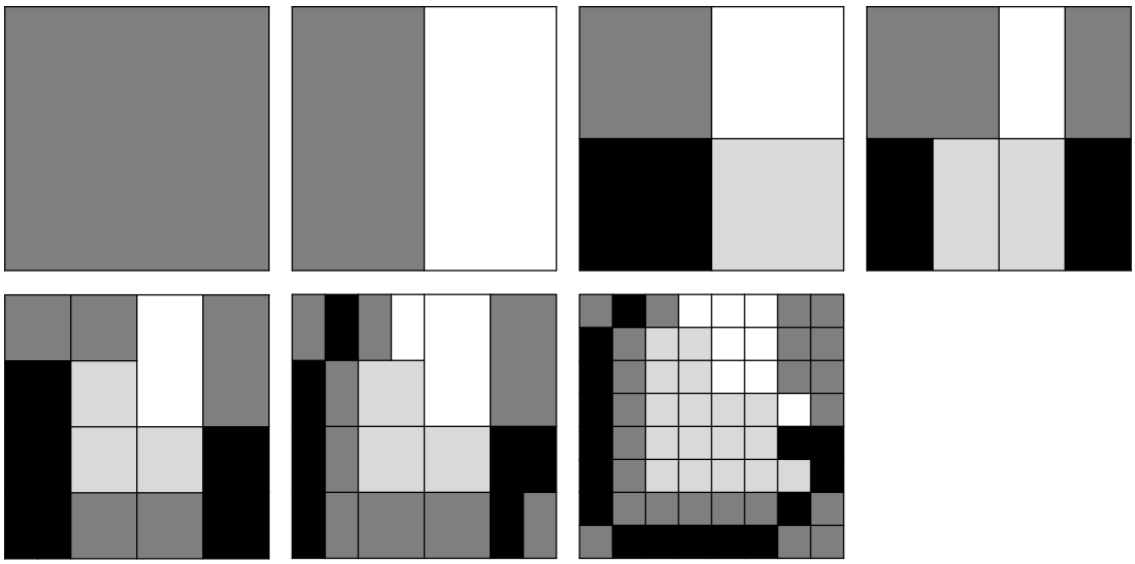
PNG najprej razdeli sliko v bloke velikosti 8x8 pikslov. Nato vsak blok pokaže v sedmih korakih. V vsakem koraku dobimo vrednosti pikslov, ki ustrezajo številki koraka, prikazane v tabeli 2.1. Te vrednosti pikslov predstavljajo levo zgornjo vrednost znotraj podblokov in jih interpoliramo po posameznem podbloku, da zapolnimo mesta pikslov, ki še niso na voljo. Toliko kot je ustreznih vrednosti, toliko podblokov nastane znotraj bloka.

Zagotoviti moramo, da bodo v vsakem koraku vsi podbloki enake velikosti ter da se ne bodo prekrivali. To storimo tako, da vedno izberemo podblok, kjer je število vrstic večje ali enako številu stolpcev. V prvem koraku interpoliramo celotno površino bloka, v vsakem nadaljnjem koraku pa zgolj polovico bloka, tako podvojimo ločljivost

posameznega bloka. Površino bloka, ki jo interpoliramo, enakomerno razdelimo med vse podbloke. Primer vseh sedmih korakov prikazuje slika 2.2.

Tabela 2.1: Vrstni red pridobivanja vrednosti pikslov bloka

1	**6**	**4**	**6**	**2**	**6**	**4**	**6**
7	**7**	**7**	**7**	**7**	**7**	**7**	**7**
5	**6**	**5**	**6**	**5**	**6**	**5**	**6**
7	**7**	**7**	**7**	**7**	**7**	**7**	**7**
3	**6**	**4**	**6**	**3**	**6**	**4**	**6**
7	**7**	**7**	**7**	**7**	**7**	**7**	**7**
5	**6**	**5**	**6**	**5**	**6**	**5**	**6**
7	**7**	**7**	**7**	**7**	**7**	**7**	**7**



Slika 2.1: Sedem korakov prepletanja

2.2 Stiskanje

Brezizgubnost formata je zagotovljena z dvema fazama:

- Z napovedjo pikslov, ki jo bomo opisali v naslednjem podpoglavju, in
- Z brezizgubnim algoritmom za stiskanje Deflate [4]

2.2.1 Filtri

Napoved pikslov je implementirana z različnimi napovednimi modeli, ki se v besednjaku PNG imenujejo filtri. Pregled slednjih je v tabeli 2.2. Cilj napovedovanja/filtriranja pikslov je, da lahko Deflate v drugi fazi učinkoviteje stisne podatke. Filtriranje opravimo za vsako vrstico slike posebej in deluje na nivoju zlogov in ne pikslov. Posledično filtri niso odvisni od tipa slike. Za uspešno dekodiranje vsebuje format PNG pred začetkom vsake vrstice zlog, ki določa, s katerim filtrom je bila vrstica slike filtrirana. Za posamezno vrstico izberemo tisti filter, ki bi napovedal takšne vrednosti, da bi jih Deflate lahko najučinkoviteje stisnil.

Če ima slika možnost prikazovanja s prepletanjem, se vsak blok 8x8 obravnava kot slika in tako je lahko vsaka vrstica znotraj bloka filtrirana posebej. V načinu prepletanja se podatki (v večini primerov) v drugi fazi z algoritmom Deflate ne stisnejo tako učinkovito kot sicer.

Tabela 2.2: Pregled filtrov

Oznaka	**Ime**	**Opis**
0	None	Kodiran zlog je enak trenutnemu zlogu (brez filtriranja)
1	Sub	Kodiran zlog izračunamo s pomočjo enakoležečih zlogov trenutnega in predhodnega (levega) piksla
2	Up	Kodiran zlog izračunamo s pomočjo enakoležečih zlogov trenutnega in zgornjega piksla
3	Average	Kodiran zlog izračunamo s pomočjo enakoležečih zlogov predhodnega in zgornjega piksla
4	Paeth	Kodiran zlog izračunamo s pomočjo enakoležečih zlogov predhodnega, zgornjega in levega zgornjega piksla

Pri izračunih filtrov moramo upoštevati, da nekateri enakoležni zlogi niso na voljo. Npr. levi enakoležni zlog od zloga na levem robu ne obstaja. V takšnih robnih primerih predpostavljamo, da je slednji enak 0.

Pri izračunih filtrov, podanih v enačbah 2.1 do 2.5, uporabljamo naslednje oznake:
i – položaj (zaporedna številka) zloga v prebirni[1] vrstici,
j – zaporedna številka stolpca,
B – zlog,
Bpp – število zlogov na piksel,
f_nap – vrnjena (napovedana) vrednost funkcije oz. algoritma opisanega na sliki 2.1.

- **Filter None:**

$$None(i) = B_{i,j} \tag{2.1}$$

- **Filter Sub:**

$$Sub(i) = (B_{i,j} - B_{i-Bpp,j}) \bmod 256 \tag{2.2}$$

- **Filter Up:**

$$Up(i) = (B_{i,j} - B_{i,j-1}) \bmod 256 \tag{2.3}$$

- **Filter Average:**

$$Average(i) = \lfloor (B_{i-Bpp,j} + B_{i,j-1}) / 2 \rfloor \bmod 256 \tag{2.4}$$

- **Filter Paeth:**

$$Paeth(i) = (B_{i,j} - f\_nap\ (B_{i-Bpp,j}, B_{i,j-1}, B_{i-Bpp,j-1})) \bmod 256 \tag{2.5}$$

[1] Prebiranje poteka v rastrskem prebirnem vrstnem redu

```
uint8_t f_napoved(uint8_t a, uint8_t b, uint8_t c)
{
    // a = enakoležni zlog levo glede na trenutni zlog,
    //     ki ga kodiramo. B[i-Bpp][j]
    // b = enakoležni zlog zgoraj glede na trenutni zlog,
    //     ki ga kodiramo. B[i][j-1]
    // c = enakoležni zlog zgoraj levo glede na trenutni zlog,
    //     ki ga kodiramo. B[i-Bpp][j-1]

    int p = a + b - c;
    int pa = abs(p-a);
    int pb = abs(p-b);
    int pc = abs(p-c);

    if ((pa <= pb) && (pa <= pc))
        return a;
    if (pb <= pc)
        return b;
    return c;
}
```

Slika 2.2: Izračun napovedi

V primeru 24-bitnih slik RGB so torej enakoležni zlogi elementi, ki pripadajo istemu barvnemu kanalu oz. barvni ravnini. Tako je tudi smiselno, saj so enakoležni zlogi tudi v korelaciji oz. so med seboj povezani. Odvisnost *Bpp* od tipa slike prikazuje tabela 2.3.

Tabela 2.3: Odvisnost *Bpp* od tipa slike

Tip slike	**Bitne ravnine**	*Bpp*
sivinska	1, 2, 4, 8	1
	16	2
Sivinska s prosojnostjo	8	2
	16	4
Paletna	1, 2, 4, 8	1
RGB	8	3
	16	6
RGB s prosojnostjo (RGBA)	8	4
	16	8

3 METODE PREKRIVANJA INFORMACIJE

Metode za prekrivanje informacije [5] delimo na metode, ki delujejo v frekvenčnem prostoru (angl. frequency domain) in tiste, ki delujejo v prostorski domeni (angl. spatial domain). Prve uporabljamo pri slikah z izgubami (npr. format JPEG). V kolikor bi spreminjali vrednosti pikslov slikam, ki pri stiskanju izgubijo del podatkov, ne bi mogli dekodirati prikritega sporočila, zato je potrebno sliko najprej transformirati v drugo domeno. Pri slikah brez izgub pa tega problema ni, tako lahko delujemo kar v prostorski domeni oz. neposredno nad vrednostmi pikslov.

Naivna metoda prekrivanja informacije v slikah PNG bi bila za nizom znakov »IEND«, ki v formatu PNG označuje mesto, kjer dekodirnik PNG preneha z dekodiranjem slike. Velikost slike se bo spremenila. Ampak to se zgodi tudi, če delujemo v prostorski domeni neposredno nad piksli, saj bo kasneje ob kodiranju slike prišlo do različnih rezultatov filtrov (podpoglavje 2.2.1) in posledično bo stiskanje bolj oz. manj učinkovito, kar pa bo spremenilo velikost datoteke. Naivna možnost se lahko preprosto vidi z odpiranjem slike v pregledovalniku besedila. Zato se v diplomskem delu osredotočimo zgolj na metode, ki delujejo v prostorski domeni.

3.1 Metoda LSB

Metoda LSB (angl. Least Significant Bit) [6] je najenostavnejša metoda za prekrivanje informacije in zato tudi najpogosteje uporabljena. Bite sporočila vstavljamo v najmanj pomemben bit piksla. Tako lahko npr. pri 24-bitni sliki RGB bite vstavljamo v najmanj pomemben bit vsakega barvnega kanala, kar pomeni 3 bite na piksel. Spremembe na sliki, ki jih povzročimo s spreminjanjem najmanj pomembnega bita, so za človeški vidni sistem neopazne. Primer prikazuje slika 3.1, kjer je levo enobarvna 24-bitna slika barvnega modela RGB in so vsi piksli sestavljeni iz trojice vrednosti 127, 127, 127. Desno pa je slika, kjer imajo vsi piksli vrednosti 128, 128, 128.

Slika 3.1: Razlika enega bita na kanal je neopazna

V fazi testiranja (poglavje 4) bomo preizkusili kakovost slike ob spreminjanju tudi več kot samo najmanj pomembnega bita.

3.1.1 Metoda LSBMR

Veliko metod je skušalo izboljšati metodo LSB, ena izmed njih je metoda LSBMR (angl. Least Significant Bit Matching Revisited) [7]. Slednja še vedno prikriva sporočilo v najmanj pomemben bit, prikrivanje pa deluje nad dvema piksloma naenkrat. Označimo dva bita sporočila, ki ju želimo prikrit, z b_1 in b_2 in piksla v katera bomo prikrili bita b_1 in b_2 s p_1 in p_2.

Poglejmo si postopek prikrivanja: Najprej preverimo, ali je bit b_1 enak najmanj pomembnemu bitu piksla p_1. Če je, ostane piksel p_1 nespremenjen in preverimo, ali je b_2 enak rezultatu funkcije $f(p_1, p_2)$ (3.1). Če je, potem tudi piksel p_2 ostane nespremenjen, v nasprotnem primeru spremenimo najmanj pomemben bit piksla p_2. Če pa prvi pogoj, da je bit b_1 enak najmanj pomembnemu bitu piksla p_1, ne velja, potem piksel p_2 ostane nespremenjen in spremenimo najmanj pomemben bit piksla p_1.

Po prikrivanju obeh bitov b_1 in b_2 je najmanj pomemben bit piksla p_1 enak bitu b_1, bit b_2 pa izračunamo s funkcijo $f(p_1, p_2)$.

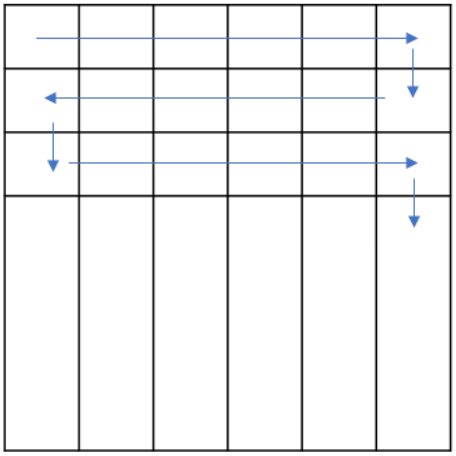
$$f(y_1, y_2) = \left(\left\lfloor \frac{y_1}{2} \right\rfloor + y_2\right) mod\ 2 \tag{3.1}$$

Zaradi tega, ker mora biti najmanj pomemben bit piksla p_1 enak bitu b_1 je verjetnost, da spremenimo najmanj pomemben bit piksla p_1 enaka kot pri navadni metodi LSB, torej $0{,}5$. Pri vstavljanju bita b_2, v kolikor ne velja prvi pogoj, piksla p_2 ne spremenimo. V nasprotnem primeru, v kolikor prvi pogoj velja, nastopi drugi pogoj, kjer bo v polovici primerov piksel p_2 prav tako ostal nespremenjen. Torej je verjetnost, da spremenimo piksel p_2 enaka 0,25, v 75 % primerov bo torej ostal nespremenjen. Posledično je verjetnost, da pri metodi LSBMR spremenimo najmanj pomemben bit enaka 0,375, kar je v primerjavi z navadno LSB metodo za 12,5 % bolje.

Problem metode LSBMR so piksli, ki imajo vrednost 0 ali 255. Zato moramo njihove vrednosti pred prikrivanjem bitov v njih, povečati oz. zmanjšati za 1. To težavo lahko omilimo, saj imamo v postopku prikrivanja včasih možnost, da izberemo ali želimo zmanjšati oz. povečati piksel. V takšnih primerih si lahko zapomnimo kakšne vrednosti so imeli piksli, preden smo jih spreminjali in jih tako postavimo na prvotne vrednosti.

3.2 Metoda PVD

Spremembe vrednosti pikslov v gladkih (angl. smooth) delih slik je s strani človeškega očesa lažje zaznati. Metoda PVD (angl. Pixel Value Differencing) [8] razdeli sliko v ne prekrivajoče se pare zaporednih blokov v tako imenovani smeri cik-cak (slika 3.2), ki jih sestavljata dva piksla.



Slika 3.2: Smer cik-cak

Glede na absolutno razliko med piksloma so bloki razvrščeni v kategorije (tabela 3.1). Spodnjo mejo posamezne kategorije označimo z M_S, zgornjo mejo pa z M_Z. Število prikritih bitov n, ki jih skrijemo v par pikslov, pa označimo z n. Manjša kot je razlika med piksloma, bolj gladka je slika na delu, kjer se ta dva piksla nahajata. Z večanjem razlike med piksloma se torej zmanjšuje gladkost slike in posledično na tistih delih slike metoda PVD v blok skrije več bitov sporočila kot sicer, kar izračunamo po (3.2).

$$n = \log_2(M_Z - M_S + 1) \tag{3.2}$$

Tabela 3.1: Kategorije uvrščanja blokov

Številka kategorije	**Spodnja meja kategorije (M_S)**	**Zgornja meja kategorije (M_Z)**	**Število bitov, ki jih prikrivamo na par pikslov (n)**
1	0	7	3
2	8	15	3
3	16	31	4
4	32	63	5
5	64	127	6
6	128	255	7

Poglejmo postopek: Imamo sosednja piksla (blok), ki ga predstavljata levi in desni piksel. Najprej izračunamo razliko med njima in potem blok uvrstimo v ustrezno kategorijo. Nato s pomočjo kategorije in desetiške vrednosti n bitov, ki jih prikrivamo, izračunamo novo razliko (3.3):

$$razlika_{nova} = \begin{cases} M_S + n\_bitov_{10}, & razlika \geq 0 \\ -(M_S + n\_bitov_{10}), & razlika < 0 \end{cases} \quad (3.3)$$

Naj bo $m = \frac{(razlika_{nova} - razlika)}{2}$. Sedaj v (3.4) izračunamo nove vrednosti pikslov bloka:

$$levi_{novi}, desni_{novi} = \begin{cases} levi - \lfloor m \rfloor, & desni + \lceil m \rceil, \quad soda\ razlika \\ levi - \lceil m \rceil, & desni + \lfloor m \rfloor, \quad drugače \end{cases} \quad (3.4)$$

Nove vrednosti obeh pikslov bloka izračunamo tako, da vrednosti obeh pikslov približno enako zmanjšamo oz. povečamo. Večja kot je nova razlika, večji bo interval med novima vrednostima obeh pikslov in obratno. Lahko se zgodi, da katera izmed novih vrednosti

pade izven območja [0, 255]. V takšnem primeru v ta dva piksla ne bomo prekrili sporočila. Sedaj lahko razumemo, zakaj metoda PVD deluje v smeri cik-cak. Imejmo sliko z dvema barvama; leva polovica slike je črna (vrednost 0), druga polovica pa naj bo siva (vrednost 127). Na sredini slike (gledano navpično) se barvi stikata (nastane rob) in tam bi morali prikriti največ bitov. Vendar vrednosti levega piksla ne moremo zmanjšati, saj je 0. Tako bomo ta blok oz. par pikslov pustili nedotaknjen, torej ne bomo prikrivali bitov. Enak problem se posledično zgodi na sredini vsake vrstice slike. Če pa delujemo v smeri cik-cak, potem lahko v vsaki drugi vrstici uspešno prikrijemo bite, saj se zamenjata vlogi levega in desnega piksla.

Možnost preboja spodnje ali zgornje vrednosti preverimo tako, da v (3.4) vstavimo $m = M_Z - razlika$. Če katera izmed obeh dobljenih vrednosti prebije spodnjo ali zgornjo mejo, pustimo blok teh dveh pikslov nedotaknjen. Enak postopek za zaznavo blokov brez prekritega sporočila uporabimo tudi pri dekodiranju. Če ugotovimo, da blok vsebuje sporočilo, izračunamo prikrite bite z enačbo (3.5), kjer razlika v stego-sliki, to je sliki s prikritim sporočilom, dejansko predstavlja novo razliko iz faze kodiranja oz. prikrivanja sporočila.

$$biti_{10} = |razlika| - M_S \tag{3.5}$$

Poglejmo si primer: Imamo dva sosednja piksla, kjer ima levi vrednost 130 in desni 141. Razlika med njima je $141 - 130 = 11$. Blok uvrstimo v 2. kategorijo, kar pomeni, da bomo vanj prikrili 3 bite sporočila. Predpostavljamo, da so biti sporočila, ki jih moramo prikrit, $101_2 = 5_{10}$. Nova razlika je potem $8 + 5 = 13$. Po (3.4) izračunamo novo vrednost levega in desnega piksla in dobimo $levi = 129$ in $desni = 142$.

V fazi dekodiranja zopet izračunamo razliko, ki je, kot že omenjeno, enaka novi razliki iz faze kodiranja, torej 13. Vrednost prikritih bitov je torej $|13| - 8 = 5 = 101_2$.

Velja omeniti, da nam enolično dekodiranje omogoča dejstvo, da je nova razlika še vedno znotraj iste kategorije kot predhodna razlika med piksloma, torej kategorija je v fazi kodiranja in dekodiranja enaka.

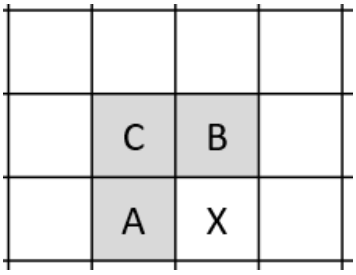
Poleg že omenjenih kategorij (tabela 3.1), so se v želji po boljši vizualni sliki uveljavile še kategorije (tabela 3.2), kjer prikrivamo manj bitov.

Tabela 3.2: Kategorije uvrščanja blokov za boljšo vizualno sliko

Številka kategorije	**Spodnja meja kategorije (M_S)**	**Zgornja meja kategorije (M_Z)**	**Število bitov, ki jih prikrivamo na par pikslov (n)**
1	0	1	1
2	2	3	1
3	4	7	2
4	8	11	2
5	12	15	2
6	16	23	3
7	24	31	3
8	32	47	4
9	48	63	4
10	64	95	5
11	96	127	5
12	128	191	6
13	192	255	6

3.3 Napovedna metoda PPV

Napovedna metoda PPV (angl. Predictive Pixel Value) [9] uporablja idejo napovedi, ki smo jo spoznali v 2. poglavju. Izberemo lahko katerikoli napovedni model, saj metoda uporablja zgolj rezultat slednjega. Tudi tukaj bi lahko, npr., za vsako vrstico slike izbrali napovedni model, ki bi dal najboljše rezultate, torej, da bi bili piksli čim manj spremenjeni. V diplomskem delu smo uporabili spremenjen napovedni model MED (angl. Median Edge Detection). Piksli, ki sodelujejo pri napovedi, so prikazani na sliki 3.2.



Slika 3.2: Piksli, sodelujoči pri napovednem modelu MED

V (3.6) je izračun napovedi piksla s spremenjenim napovednim modelom MED.

$$x_{napoved} = \begin{cases} \min(a,b), & c \geq \max(a,b) \\ \max(a,b, & c \leq \min(a,b) \\ \frac{a+b}{2}, & drugače \end{cases} \tag{3.6}$$

Poglejmo si delovanje metode: Najprej napovemo piksel in izračunamo vrednost napake $EV = OPV - PPV$, kjer je OPV originalna vrednost piksla in PPV napovedana vrednost piksla. Če je $|EV| \; mod \; 2^n = (n \; bitov \; sporočila)_{10}$, kjer n predstavlja število najmanj pomembnih prikritih bitov na piksel oz. barvni kanal, potem je spremenjena vrednost napake $MEV = EV$. MEV in EV sta v tem primeru enaka, saj se n najmanj pomembnih bitov napake EV ujema s n biti sporočila, ki jih želimo prikrit. V nasprotnem primeru izračunamo $MEV = |EV| + (n \; bitov \; sporočila)_{10} - |EV| \; mod \; 2^n)$. Če velja $|EV -$

$(-MEV)| < |EV - MEV|$, potem $MEV = -MEV$. Sedaj bo v stego-sliki vrednost piksla enaka $PPV + MEV$. Ta vrednost sodeluje pri napovedovanju vrednosti naslednjega piksla. Sprememba vrednosti napake skuša zagotoviti čim manjšo popačenje slike.

V stego-sliki nato zopet izračunamo napovedno vrednost piksla PPV, ki jo odštejemo od vrednosti piksla v stego-sliki (SPV) in preberemo n najmanj pomembnih bitov, ki so dejansko prikriti biti (3.7)

$$prikriti_{biti_{10}} = |(SPV - PPV)\ |\ mod\ 2^n \tag{3.7}$$

Iz desetiškega števila sedaj zgolj preberemo vsakega izmed n bitov s pomočjo bitnih operacij.

Velja poudariti, da v fazi dekodiranja napovedana vrednost piksla ne sodeluje pri napovedi naslednjega piksla, kar pomeni, da vrednosti pikslov stego-slike v fazi dekodiranja ne spreminjamo.

Primer: Napovedni model je za piksel vrednosti 172 vrnil napoved $PPV = 170$, torej $EV = 172 - 170 = 2$. Predpostavljajmo, da prikrivamo 2 bita sporočila na piksel oz. barvni kanal, torej $n = 2$. Bita, ki ju želimo prikrit, sta $01_2 = 1_{10}$. Izraz $|1|\ mod\ 2^2 = 2$ ne drži, zato izračunamo $MEV = |2| + (1 - |2|\ mod\ 2^2) = 1$. Izraz $|2 + 1| < |2 - 1|$ ne drži, zato vrednosti MEV ne spreminjamo. Nova vrednost piksla je torej $170 + 1 = 171$.

Pri dekodiranju bomo izračunali enako napoved $PPV = 170$. Izračunamo napako $EV = 171 - 170 = 1$. $1\ mod\ 2^2 = 1$, torej sta dekodirana dva najmanj pomembna bita enaka $01_2 = 1_{10}$.

3.4 Metoda EMD

Glavna ideja metode EMD (angl. Exploiting Modification Direction) [10] je, da najprej združujemo po n bitov sporočila in jih pretvarjamo v $(2k + 1)$ notacijski sistem, kjer sta $n\ in\ k$ parametra, ki ju izberemo. Pretvorjenih n bitov sporočila prenaša skupina k pikslov, kjer je največ eden izmed k pikslov povečan oz. zmanjšan za 1, tako je vseh možnosti $2k$ oz. $2k + 1$, če upoštevamo, da lahko ostane vrednost vseh k pikslov nespremenjena. Za predstavitev n bitov, v $2k + 1$ notacijskem sistemu, potrebujemo m števk. V (3.8) vidimo povezavo med n in m.

$$n = \lfloor m * \log_2(2n + 1) \rfloor \tag{3.8}$$

Primer: Če $n = 4, k = 2$ in je niz štirih naslednjih bitov sporočila $1010_2 = 10_{10}$, potem je ta vrednost v ($2 * 2 + 1$) notacijskem sistemu zapisana kot 20_5. Sedaj vsako števko prikrijemo v skupino po k pikslov. Torej v skupini imamo dva piksla p_1 in p_2, ki naj imata vrednosti 137 in 138. Sedaj je potrebno izračunati rezultat funkcije f (3.9)

$$f(p_1, p_2, \dots, p_c) = (\sum_{i=1}^{c} p_i * i\ )\ mod\ (2k + 1) \tag{3.9}$$

V kolikor je skrivna števka d enaka rezultatu funkcije f, potrem ne spreminjamo nobenega izmed k pikslov. V nasprotnem primeru izračunamo $s = (d - f)\ mod\ (2k + 1)$. Če $s \leq k$, potem povečamo vrednost piksla p_s za 1, drugače pa zmanjšamo vrednost piksla p_{2k+1-s} za 1. Rezultat funkcije je v našem primeru 3. Vstavljamo prvo števko $d = 2$. Vrednosti f in d nista enaki, zato izračunamo $s = (2 - 3)\ mod\ 5 = 4$. Ker $4 \leq 2$ ne drži, dekrementiramo vrednost piksla p_1. V stego-sliki ima skupina pikslov v našem primeru vrednosti 136 in 138. Pri dekodiranju dobimo prikrito števko z računanjem funkcije f skupine pikslov iz stego-slike. Dobimo 2, kar je števka, ki smo jo kodirali.

Problem nastane pri robnih pikslih, saj npr. vrednosti 255 ne moremo povečati. Takrat metoda EMD vrednost piskla dekrementira in ponovno začne z računanjem funkcijske vrednosti, saj bitov še nismo prikrili. V takšnih robnih primerih se torej lahko spremeni več kot samo en piksel v skupini n elementov. Še vedno pa bo vsak piksel na sliki v najslabšem primeru dekrementiran ali inkrementiran za 1.

V želji, da bi lahko povečali število prikritih bitov sporočila na piksel, saj npr. pri $k = 2$, EMD prikrije zgolj eno števko d na dva piksla, so se razvile številne nadgradnje EMD, ena izmed njih (opisana v podpoglavju 3.4.1) podvoji število bitov, ki jih lahko prikrijemo na piksel.

3.4.1 Nadgradnja EMD

Pri tej različici lahko v vsak piksel na sliki shranimo eno števko d. Postopek kako pridemo do števk d, je enak kot v prvotni EMD metodi. Za vsak piksel oz. barvni kanal izračunamo vrednost funkcije f (3.10) za vse vrednosti s, za katere velja $|s| \leq k$. V kolikor je vrednost piksla p med vključno 0 in 1, potem f izračunamo za vse s, za katere velja $0 \leq s < 2k + 1$. Če pa je vrednost med vključno 254 in 255, potem f izračunamo za vse s, za katere velja $-(2k + 1) < x \leq 0$.

$$f = (p + s)\ mod\ (2k + 1) \tag{3.10}$$

Vrednost novega piksla, torej piksla stego-slike, izračunamo po (3.11), kjer je izbran takšen s, da velja $f = d$.

$$p_{novi} = p + s \tag{3.11}$$

Pri dekodiranju se skrivna števka d izračuna po (3.12):

$$d = piksel_{stego}\ mod\ (2k + 1) \tag{3.12}$$

3.5 Metoda z zaznavo robov

Metoda PVD (podpoglavje 3.2), ko primerja zaporedne piksle, na nek način zaznava, če je prišlo do navpičnega roba, ker bo metoda takrat našla veliko razliko med sosednjima piksloma. Dobro bi bilo, če bi zaznavala tudi vodoravne robove, saj bi tako lahko prikrili več bitov sporočila. Rešitev so zaznavalniki robov. Pri tej metodi z zaznavalnikom robov lahko po (3.13) povežemo dva Sobelova operatorja [11] in združimo zaznavo vodoravnih in navpičnih robov s pomočjo dveh strukturnih elementov (slika 3.3), s katerimi izvajamo konvolucijo s piksli slike. Levi strukturni element omogoča zaznavo navpičnih, desni pa vodoravnih robov. Tako lahko zaznamo vodoravne in navpične robove hkrati (slika 3.4).

$$hv = \sqrt{h^2 + v^2}, \tag{3.13}$$

kjer so:

h = rezultat vodoravnega operatorja,

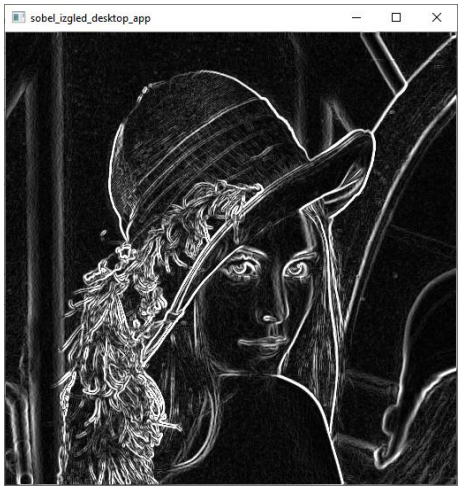
v = rezultat navpičnega operatorja,

hv = združeni vrednosti.

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

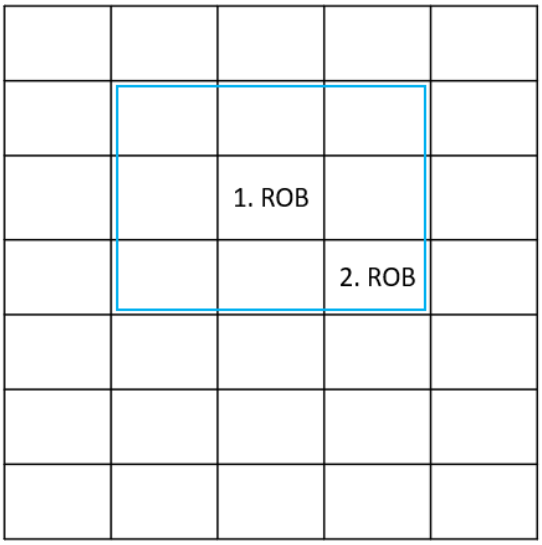
Slika 3.3: Sobelova strukturna elementa



Slika 3.4: Zaznava navpičnih in vodoravnih robov

Problem operatorjev je, da se morajo v celoti prekrivati s sliko. Posledično operatorjev vedno ne moremo izvajati nad robnimi deli slik. Ta problem se v 2D filtriranju pogosto rešuje tako, da se podaljša rob na sliki, kar pa pri steganografiji ne pride v poštev. Pri prikrivanju informacije v robove pride še do enega problema (slika 3.5). Ko zaznamo prvi rob, tam ležečemu pikslu prikrijemo bite. Nato naletimo na drugi rob in prikrijemo bite. Piksel, kjer se nahaja drugi rob, smo pred prikrivanjem bitov uporabili pri izračunu, kjer smo prvi rob zaznali kot rob. Vendar ob prikritju bitov v piksel drugega roba tvegamo, da v fazi dekodiranja prvega roba več ne bomo zaznali kot rob. To pomeni, da bitov, prikritih v prvem robu, ne bomo dekodirali. Posledično je popolna rekonstrukcija prikritega sporočila nemogoča.

Rešitev je, da sliko razdelimo na neprekrivajoče se bloke velikosti 3x3, kjer računamo vrednosti Sobelovih operatorjev nad centralnim pikslom znotraj blokov.



Slika 3.5: Problem pri metodi z zaznavo robov

4 REZULTATI

V okolju Qt [12] smo v jeziku C++ implementirali metode za prekrivanje informacije, opisane v 3. poglavju. Kodirali smo slike v 24-bitnem barvnem modelu RGB. Najprej v sliko skrijemo 32 bitov, ki vsebujejo informacijo o dolžini sporočila, ki ga zatem prikrijemo v sliko. V diplomskem delu smo metode primerjali na različnih slikah PNG velikosti 512x512 pikslov.

Pri primerjanju slik smo si pomagali z metriko PSNR (angl. Peak Signal-to-Noise-Ratio) [13], ki jo izračunamo z enačbo 4.2. V obeh enačbah MSE predstavlja povprečno kvadratno napako (angl. Mean Square Error), u označuje število pikslov slike, MAX_i je največja možna vrednost piksla, P_i in S_i pa predstavljata i-ti piksel izvorne slike oz. stego-slike.

$$MSE = \frac{1}{u}\sum_{i=1}^{u}(P_i - S_i)^2 \tag{4.1}$$

$$PSNR = 20\log MAX_i - 10\log MSE \tag{4.2}$$

Vidimo, da ima MSE pomembno vlogo pri izračunu metrike PSNR. Dve sliki, vizualno zelo drugačni, lahko imata enak MSE, zato smo poleg metrike PSNR uporabili še metriko SSIM (angl. Structural Similarity Index Measure) [14].

Po enačbi 4.3 lahko izračunamo SSIM za poljubno okno slike, kjer x in y predstavljata istoležni okni slik, ki ju primerjamo. SSIM se poskuša prilagoditi vidnemu zaznavnemu sistemu. Bolj kot sta si sliki podobni, večji sta vrednosti metrik PSNR in SSIM. Kot parameter primerjave smo vzeli tudi kapaciteto, ki predstavlja koliko bitov lahko prikrijemo v sliko z uporabo določene metode za prekrivanje informacije.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}, \tag{4.3}$$

kjer so:

μ_x = povprečje vrednosti v oknu x,

μ_y = povprečje vrednosti v oknu y,

σ_x = varianca vrednosti v oknu x,

σ_y = varianca vrednosti v oknu y,

σ_{xy} = kovarianca x in y.

4.1 Slika Lenna

a) Originalna slika b) Sprememba 1 bit

c) Sprememba 2 bita d) Sprememba 3 biti

e) Sprememba 4 biti f) Sprememba 5 bitov

Slika 4.1: Primerjava slik pri metodi LSB

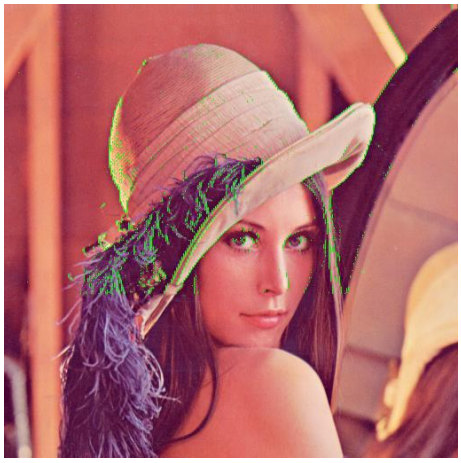
Na sliki 4.1e že opazimo rahle spremembe med slikama, na sliki 4.1f pa je slika vizualno zelo očitno različna od originalne slike.

Na sliki 4.2 vidimo stego-sliko, dobljeno z metodo PVD.

a) Originalna slika b) stego-slika

Slika 4.2: Primerjava slik pri metodi PVD

Na sliki 4.3 smo z zeleno označili, kje je metoda PVD spreminjala več kot 4 bite, torej tam, kjer je bila razlika med obema piksloma bloka večja od 31.



Slika 4.3: Spremenjeni piksli za več kot 4 bite pri metodi PVD

Na sliki 4.4 vidimo slike, dobljene z metodo PPV.

a) Originalna slika b) Sprememba 1 bit

c) Sprememba 2 bita d) Sprememba 3 biti

e) Sprememba 4 biti f) Sprememba 5 bitov

Slika 4.4: Primerjava slik pri metodi PPV

Če primerjamo metodo LSB in metodo PPV lahko vidimo občutno razliko, če primerjamo kakšna je bila stego-slika pri metodi LSB ob prikrivanju 5 bitov (slika 4.1f) in kakšna je stego-slika pri metodi PPV (slika 4.4f). Vidimo, da se je metoda PPV v tem primeru vizualno izkazala bolje, kljub temu, da je sicer opazna razlika, a ne tako očitna kot pri metodi LSB.

Na sliki 4.5b imamo stego-sliko, kjer smo z metodo LSBMR malce izboljšali navadno metodo LSB. Razlike, tako kot pri metodi LSB, prostemu očesu niso vidne, saj spreminjamo (največ) en bit, a redkeje kot pri metodi LSB.

a) Originalna slika b) stego-slika

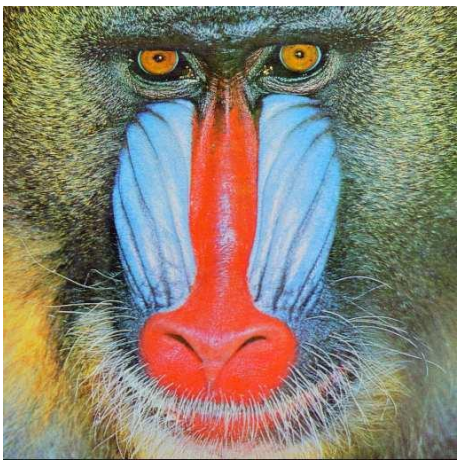
Slika 4.5: Primerjava slik pri metodi LSBMR

Tabela 4.1: Primerjava rezultatov za sliko Lenna

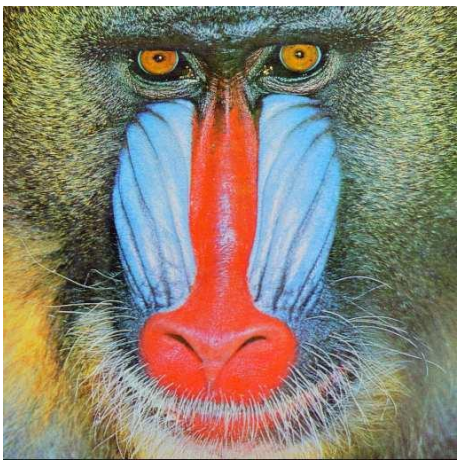
Metoda	Število prikritih bitov (pri metodi PVD na par kanala pikslov)	Kapaciteta na 24-bitni piksel v bitih	PSNR (dB)	SSIM
LSB	1	3	51,14	0,997
	2	6	44,14	0,990
	3	9	37,89	0,965
	4	12	31,77	0,884
	5	15	25,74	0,708
LSBMR	1	3	52,39	0,998
PVD	3, 4, 5, 6 ali 7	4,7	40,98	0,988
	1, 2, 3, 4, 5 ali 6	2,6	47,55	0,997
PPV	1	3	51,15	0,997
	2	6	44,30	0,991
	3	3	38,13	0,967
	4	12	32,00	0,884
	5	15	25,84	0,680

V testiranje smo vključili tudi rezultate metode PVD s kategorijami, kjer so razponi med spodnjo in zgornjo vrednostjo manjši (prikrivamo manj bitov). Pričakovano se je najbolje izkazala metoda LSBMR, ampak za vizualno boljšo sliko prikrivamo malo bitov. Že prej smo s primerjavo slik ugotovili, da je metoda PPV učinkovitejša v primerjavi z metodo LSB, kar dobljeni rezultati tudi potrjujejo.

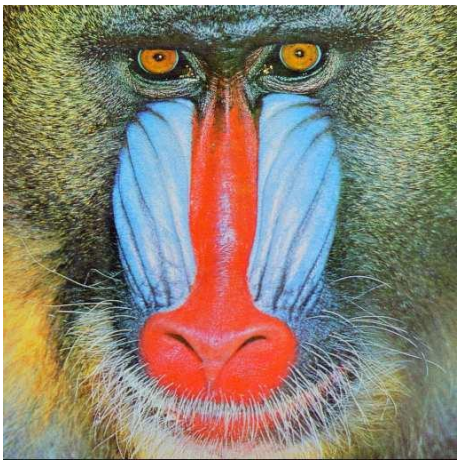
4.2 Slika Baboon



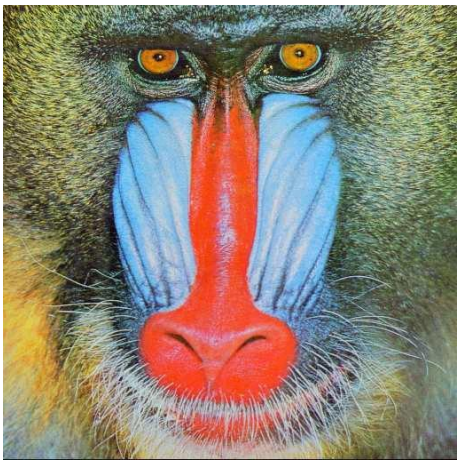
a) Originalna slika



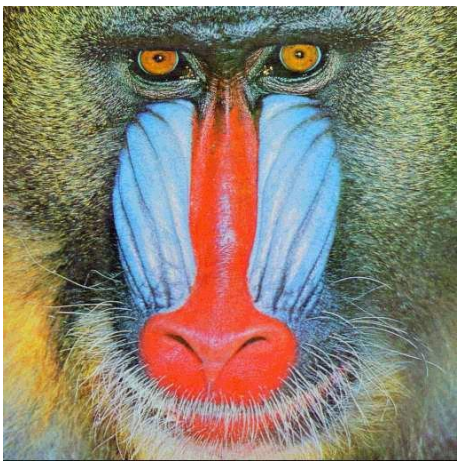
b) Sprememba 1 bit



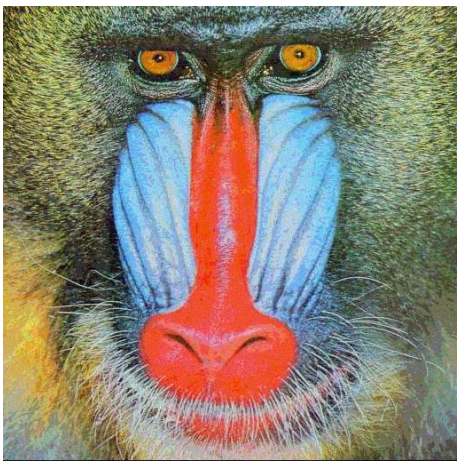
c) Sprememba 2 bita



d) Sprememba 3 biti



e) Sprememba 4 biti

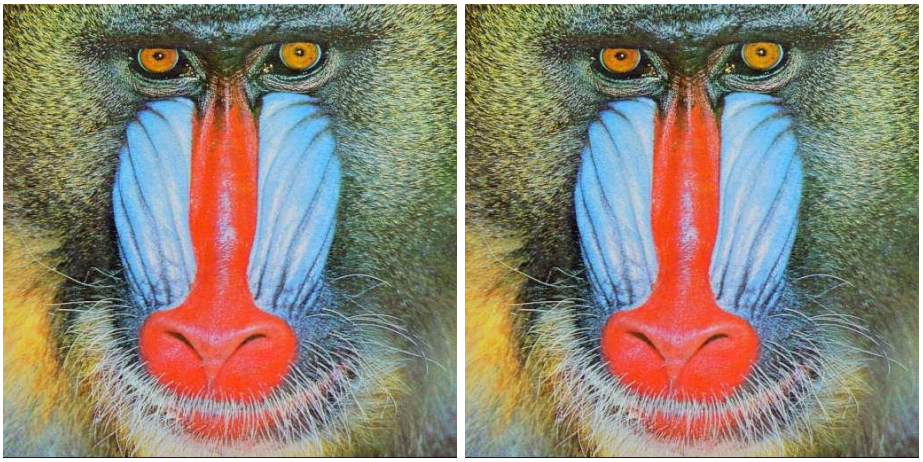


f) Sprememba 5 bitov

Slika 4.6: Primerjava slik pri metodi LSB

Spremembe na slikah (4.6) pri metodi LSB niso tako očitne v primerjavi s sliko Lenne, se pa hitreje opazijo razlike na tistih delih slike , kjer so si vrednosti pikslov zelo podobne.

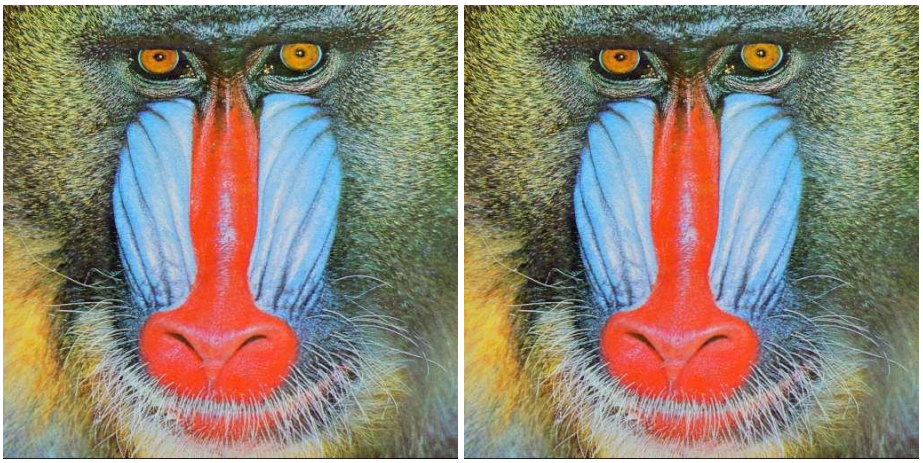
Na sliki 4.7 vidimo stego-sliko, dobljeno z metodo PVD, kjer razlik med slikama ne opazimo.



a) Originalna slika b) stego-slika

Slika 4.7: Primerjava slik pri metodi PVD z večjo kapaciteto

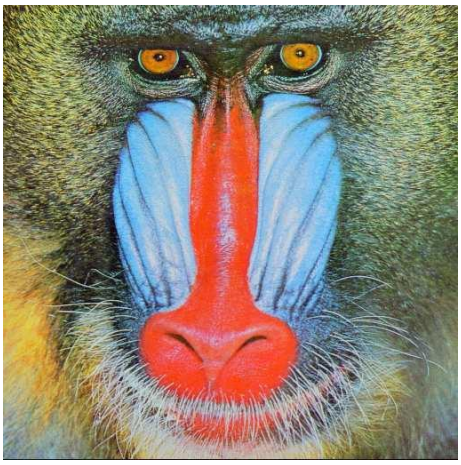
Na sliki 4.8b je stego-sliko dobljena z metodo LSBMR.



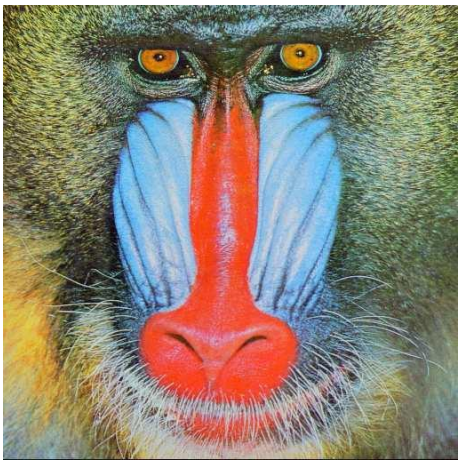
a) Originalna slika b) stego-slika

Slika 4.8: Primerjava slik pri metodi LSBMR

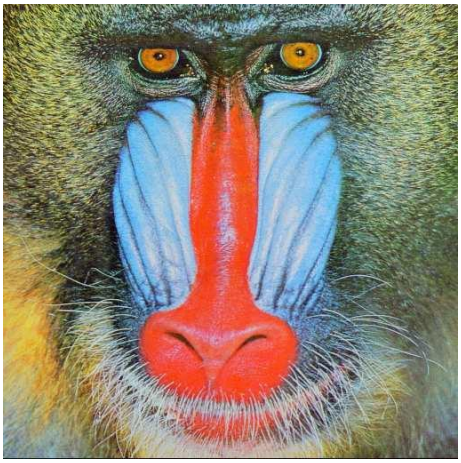
Slike (4.9), dobljene z metodo PPV, so manj popačene kot pri metodi LSB.



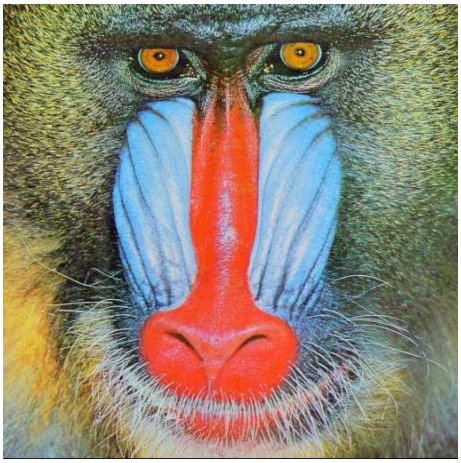
a) Originalna slika



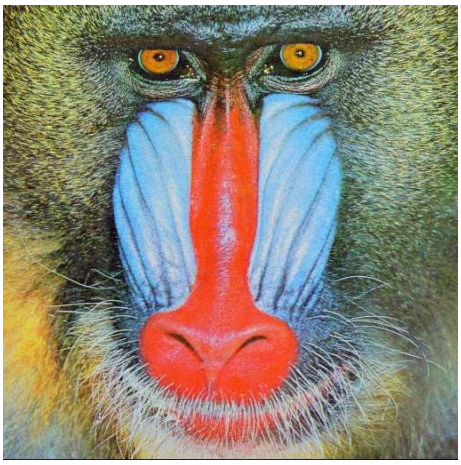
b) Sprememba 1 bit



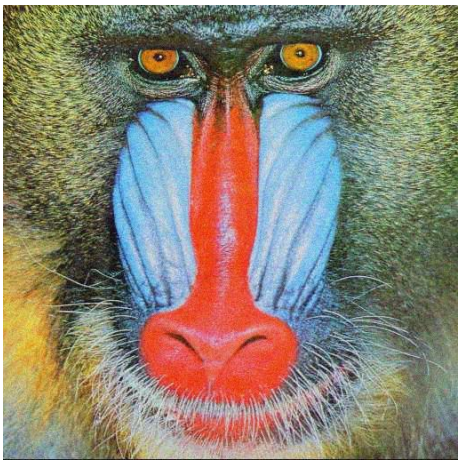
c) Sprememba 2 bita



d) Sprememba 3 biti



e) Sprememba 4 biti



f) Sprememba 5 bitov

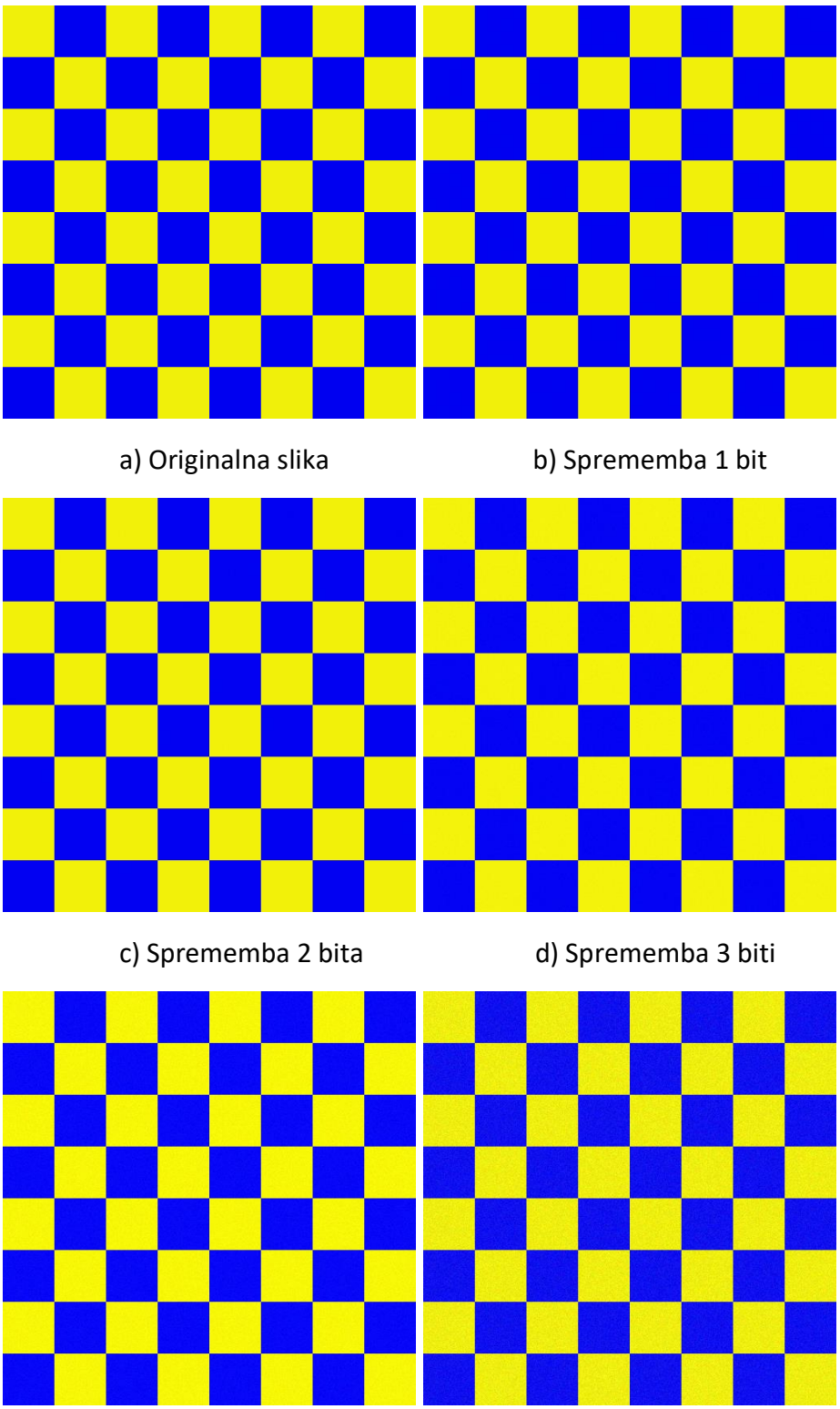
Slika 4.9: Primerjava slik pri metodi PPV

Tabela 4.2: Primerjava rezultatov za sliko Baboon

Metoda	Število prikritih bitov (pri metodi PVD na par kanala pikslov)	Kapaciteta na 24-bitni piksel v bitih	PSNR (dB)	SSIM
LSB	1	3	51,14	0,999
	2	6	44,14	0,997
	3	9	37,91	0,988
	4	12	31,83	0,956
	5	15	25,81	0,866
LSBMR	1	3	52,36	0,999
PVD	3, 4, 5, 6 ali 7	5,1	37,43	0,993
	1, 2, 3, 4, 5 ali 6	3,2	43,65	0,998
PPV	1	3	51,16	0,999
	2	6	44,24	0,997
	3	3	38,04	0,988
	4	12	31,93	0,957
	5	15	25,78	0,862

Slika Baboon se je izkazala kot zelo dobra za prekrivanje informacije, saj so vrednosti metrike PSNR in SSIM višje kot pri sliki Lenne.

4.3 Umetno generirana slika



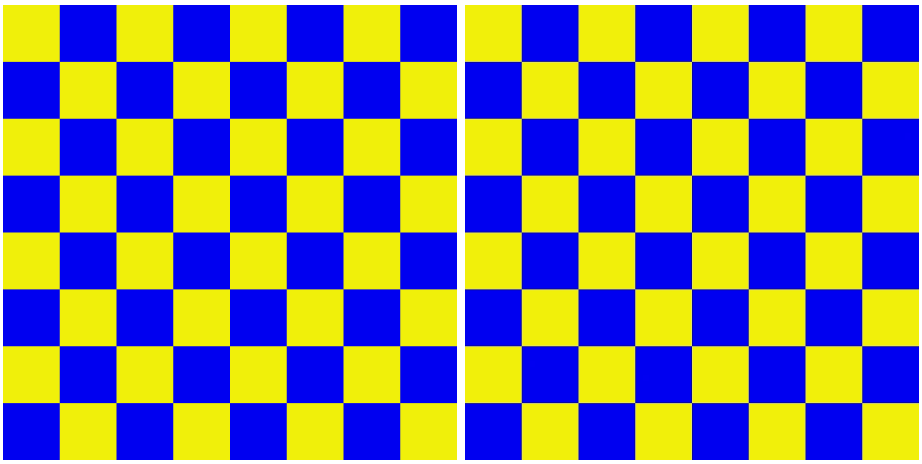
a) Originalna slika b) Sprememba 1 bit

c) Sprememba 2 bita d) Sprememba 3 biti

e) Sprememba 4 biti f) Sprememba 5 bitov

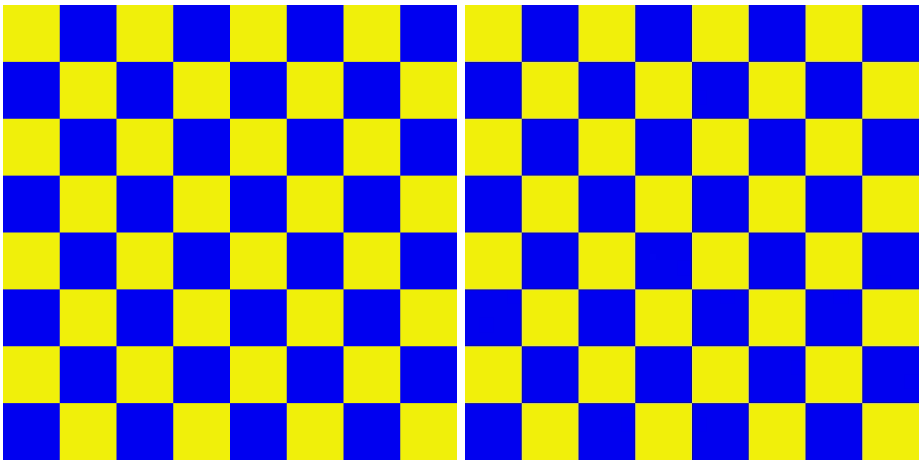
Slika 4.10: Primerjava slik pri metodi LSB

Na sliki 4.10f (ob spremembi petih bitov) je sprememba v primerjavi z originalno sliko takoj opazna.



a) Originalna slika b) stego-slika

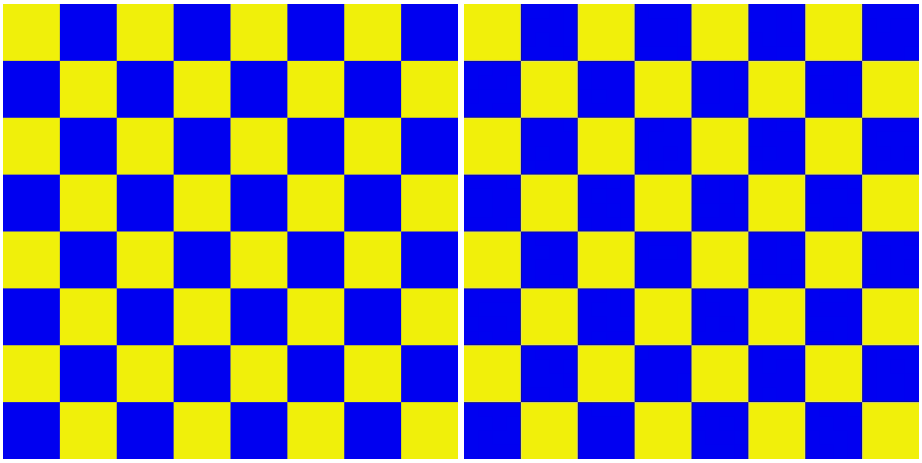
Slika 4.11: Primerjava slik pri metodi PVD



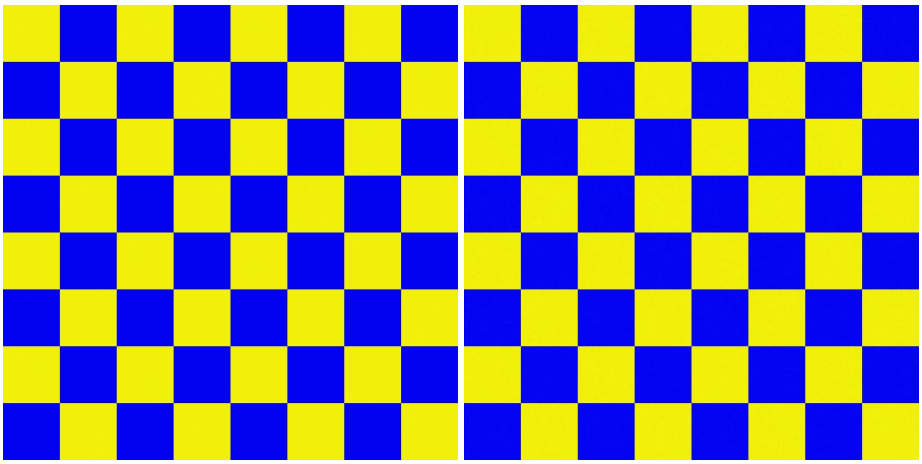
a) Originalna slika b) stego-slika

Slika 4.12: Primerjava slik pri metodi LSBMR

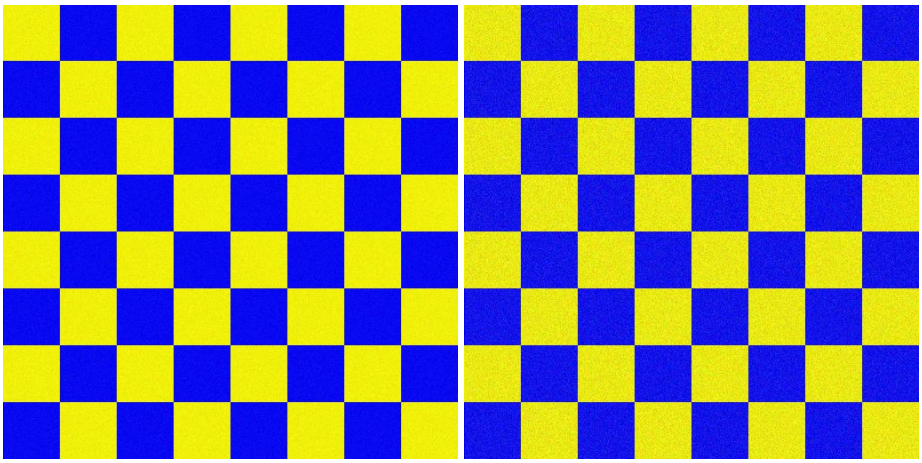
Na slikah 4.11 in 4.12 pričakovano ni vidnih razlik med originalno in stego-sliko.



a) Originalna slika b) Sprememba 1 bit



c) Sprememba 2 bita d) Sprememba 3 biti



e) Sprememba 4 biti f) Sprememba 5 bitov

Slika 4.13: Primerjava slik pri metodi PPV

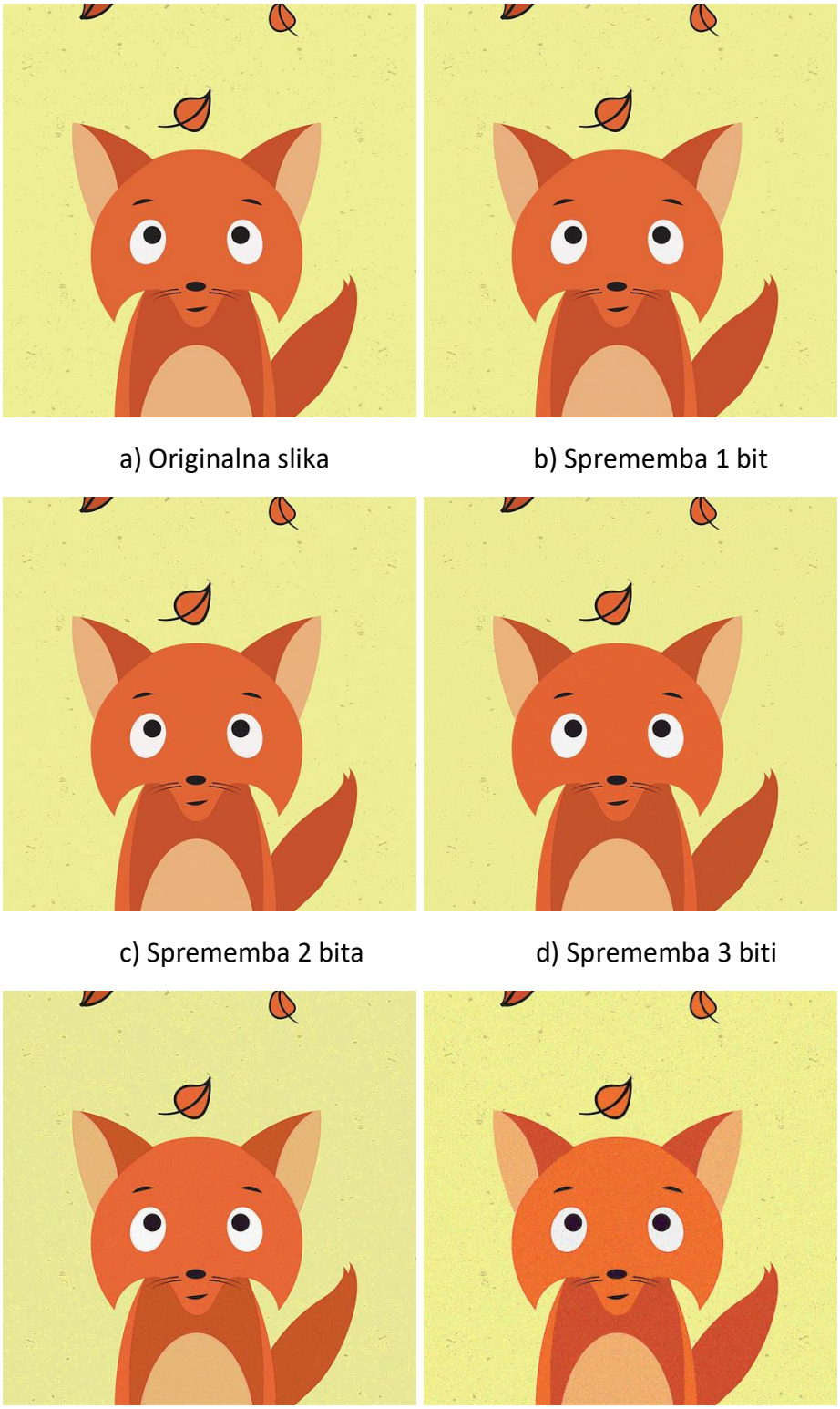
V primeru šahovnice se je metoda LSB (slika 4.10) bolje izkazala v primerjavi z metodo PPV (slika 4.13). Pri metodi PPV so polja šahovnice bolj popačena.

Tabela 4.3: Primerjava rezultatov za umetno generirano sliko

Metoda	Število prikritih bitov (pri metodi PVD na par kanala pikslov)	Kapaciteta na 24-bitni piksel v bitih	PSNR (dB)	SSIM
LSB	1	3	51,14	0,997
	2	6	43,13	0,990
	3	9	36,14	0,963
	4	12	29,73	0,870
	5	15	25,83	0,648
LSBMR	1	3	49,15	0,996
PVD	3, 4, 5, 6 ali 7	3,0	43,37	0,973
	1, 2, 3, 4, 5 ali 6	1,5	54,15	0,998
PPV	1	3	51,16	0,997
	2	6	43,19	0,985
	3	3	36,16	0,936
	4	12	29,61	0,780
	5	15	22,73	0,487

Metoda PVD ima v obeh primerih malo kapaciteto, saj v blokih nastopajo zgolj elementi istih vrednosti. Tam, kjer se barvi šahovnice stikata, pa nastane nov blok. Posledično različne barve nikoli niso znotraj istega bloka, kar pomeni, da vedno prikrivamo zgolj minimalno možno število bitov, saj je razlika med elementoma bloka zmeraj enaka 0. Zato ima metoda PVD z manjšim razponom bitov v tem primeru vizualno najboljšo sliko, vendar najmanjšo kapaciteto, enkrat slabšo kot navadna metoda LSB. Prejšnje ugotovitve, da se je metoda LSB izkazala bolje kot metoda PPV, dokazujejo tudi vrednosti PSNR in SSIM.

4.4 Risana slika



a) Originalna slika b) Sprememba 1 bit

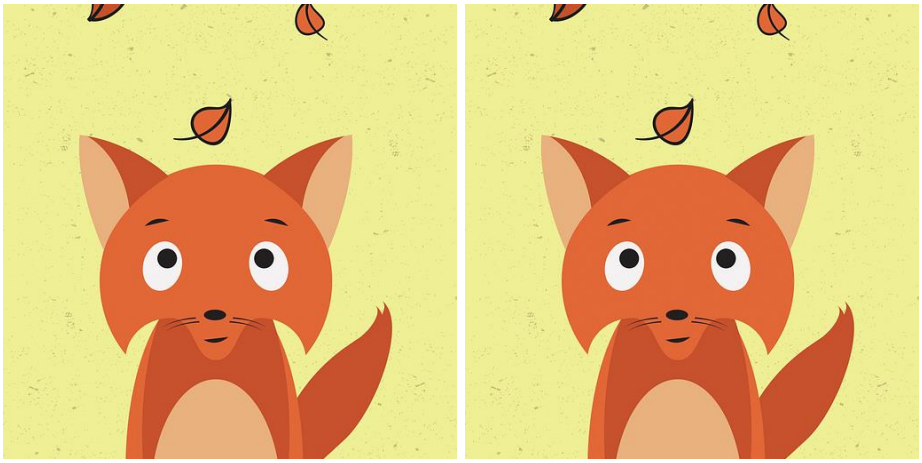
c) Sprememba 2 bita d) Sprememba 3 biti

e) Sprememba 4 biti f) Sprememba 5 bitov

Slika 4.14: Primerjava slik pri metodi LSB

Slika 4.14f na prvi pogled izstopa glede na barvo, saj je ta zelo očitno (glede na slike 4.14a-e) predvsem v obrazu lisice, prešla iz rjavega odtenka na svetlejši, oranžni odtenek.

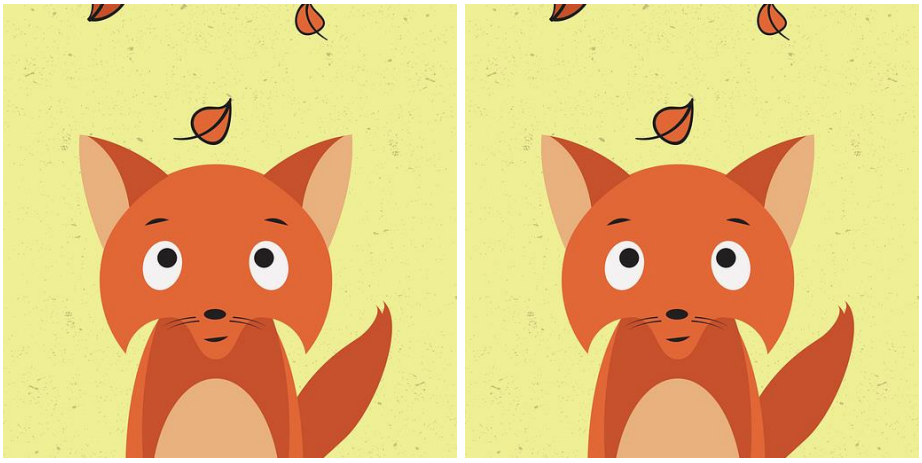
Na sliki 4.15 vidimo rezultat metode PVD.



a) Originalna slika b) stego-slika

Slika 4.15: Primerjava slik pri metodi PVD

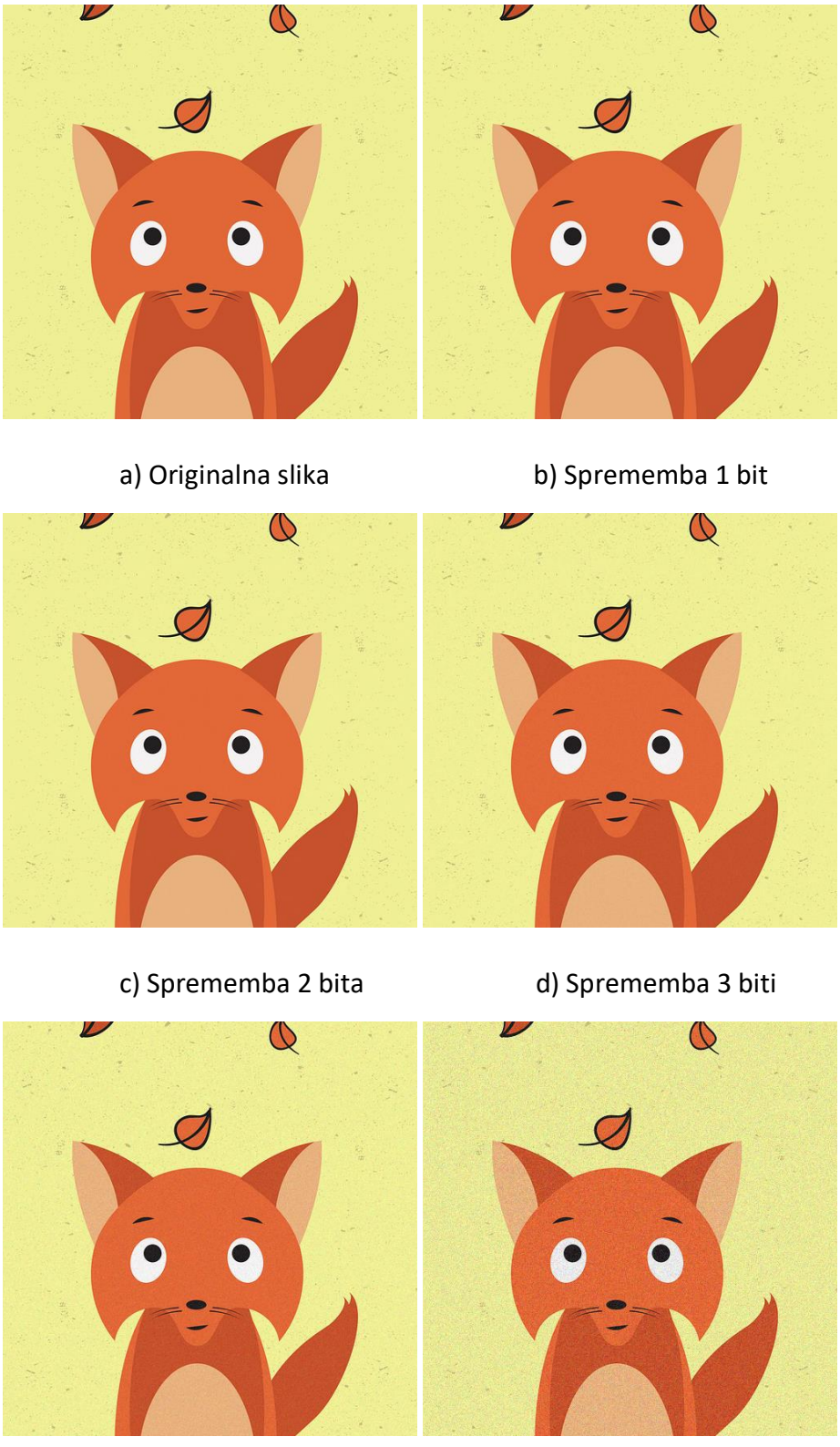
Rezultat metode LSBMR (4.16):



a) Originalna slika b) stego-slika

Slika 4.16: Primerjava slik pri metodi LSBMR

Na sliki 4.17 vidimo slike, dobljene z metodo PPV.



a) Originalna slika b) Sprememba 1 bit

c) Sprememba 2 bita d) Sprememba 3 biti

e) Sprememba 4 biti f) Sprememba 5 bitov

Slika 4.17: Primerjava slik pri metodi PPV

Opazimo, da, npr. na sliki 4.17f, sedaj ni tako očiten oranžni odtenek, a slika izgleda, v primerjavi z metodo LSB (slika 4.14f), manj gladka.

Tabela 4.4: Primerjava rezultatov za risano sliko

Metoda	**Število prikritih bitov (pri metodi PVD na par kanala pikslov)**	**Kapaciteta na 24-bitni piksel v bitih**	**PSNR (dB)**	**SSIM**
LSB	1	3	51,13	0,997
	2	6	43,87	0,989
	3	9	36,94	0,955
	4	12	30,97	0,841
	5	15	27,21	0,673
LSBMR	1	3	52,36	0,997
PVD	3, 4, 5, 6 ali 7	4,5	41,10	0,956
	1, 2, 3, 4, 5 ali 6	1,7	49,82	0,998
PPV	1	3	51,16	0,997
	2	6	44,63	0,988
	3	3	38,29	0,948
	4	12	32,04	0,818
	5	15	25,40	0,523

Zelo slabo se je izkazala metoda PVD, saj ima glede na nizko kapaciteto vizualno zelo slabo sliko. Tudi pri risanih slikah se je metoda PPV izkazala slabše kot metoda LSB. Stego-slika metode LSBMR je, pričakovano, vizualno najbolj podobna originalni sliki.

5 ZAKLJUČEK

V diplomskem delu smo predstavili dva pomembna aspekta formata PNG; prepletanje in napovedovanje pikslov s pomočjo filtrov.

V tretjem poglavju smo nato predstavili različne metode za prekrivanje informacije, primerne za slike PNG; LSB, LSBMR, PVD, PPV, EMD in metodo z zaznavo robov. Nekaj tehnik smo implementirali in jih med seboj v fazi testiranja tudi primerjali.

V nasprotju s pričakovanji, se metoda PVD ni izkazala kot učinkovita. Na fotografijah se je dobro izkazala metoda PPV, pri umetno generirani in risani sliki pa je bila (razširjena) metoda LSB vizualno bolj učinkovita. Izkazalo se je, da so slike z več podrobnostmi zelo primerne za prikrivanje informacije, saj vizualno težje zaznamo razlike med izvorno sliko in stego-sliko.

VIRI IN LITERATURA

[1] Wikipedia The Free Encyclopedia. Steganography. Dostopno na: https://en.wikipedia.org/wiki/Steganography [16. 8. 2021]

[2] Welch, A. T. 20. 6. 1983. *High speed data compression and decompression apparatus and method.* U.S. Patent and Trademark Office, US 4558302 (A).

[3] Žalik, B., *MULTIMEDIA Zapiski predavanj*. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, 2020.

[4] Wikipedia The Free Encyclopedia. Deflate. Dostopno na: https://en.wikipedia.org/wiki/Deflate [16. 8. 2021]

[5] Yahya, A. *Steganography Techniques for Digital Images*. New York: Springer, 2019.

[6] Chan, K. C., Cheng, M. L. Hiding data in images by simple lsb substitution. *Pattern Recognition*, 37, (2004), str. 469-474.

[7] Mielikainen, J. LSB matching revisited. *IEEE Signal Processing Letters*, 5, (2006), str. 285-287.

[8] Wu, C. D., Tsai, H. W. A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters*, 24, (2003), str. 1613–1626.

[9] Yu, -H. Y., Chang, -C. C., Hub, -C. Y., Hiding secret data in images via predictive coding. *Pattern recognition*, 38, (2005), str. 691–705.

[10] Zhang, X., Wang, S. Efficient Steganographic Embedding by Exploiting Modification Direction. *IEEE Communication Letters*, 11, (2006), str. 781-783.

[11] Wikipedia The Free Encyclopedia. Sobel operator. Dostopno na: https://en.wikipedia.org/wiki/Sobel_operator [16. 8. 2021]

[12] Qt. Development Framework. Dostopno na https://www.qt.io/ [16. 8. 2021]

[13] Wikipedia The Free Encyclopedia. Peak signal-to-noise ratio. Dostopno na: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio [16. 8. 2021]

[14] Wikipedia The Free Encyclopedia. Structural similarity. Dostopno na: https://en.wikipedia.org/wiki/Structural_similarity [17. 8. 2021]

[15] Salomon, D., Motta, G. *Handbook of Data Compression*, 5th Edition. London: Springer, 2010.

[16] Hussain, M., Wahab, A. W. A., Idris, B. I. Y., Ho, S. T. A., Jung -H. K., Image steganography in spatial domain: A survey. *Signal Processing: Image Communication*, 65, (2018), str. 46-66.

[17] Wikipedia The Free Encyclopedia. Portable Network Graphics. Dostopno na: https://en.wikipedia.org/wiki/Portable_Network_Graphics [16. 8. 2021]