Predhodne raziskave stiskanja barvnih slik s popravkom slabih napovedi

Bogdan Lipuš, Luka Lukač, Andrej Nerat

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Koroška cesta 46, 2000 Maribor
e-pošta: bogdan.lipus@um.si

Preliminary research on color image compression with correction of poor predictions

This paper investigates lossless color image compression to improve pixel value prediction and enhance overall compression efficiency. First, we identify inaccurate predictions using a bad prediction threshold. We then recalculate pixel predictions using only neighboring pixels that also had poor predictions. We apply different values of the bad prediction threshold. The results demonstrate that our method improves pixel prediction accuracy and, consequently, enhances color image compression performance. This preliminary research highlights the potential of this approach and suggests that further investigation is warranted.

1 Uvod

V članku se lotevamo problema brezizgubnega stiskanja barvnih slik [1]. Izhajamo iz naših predhodnih raziskav, v katerih smo se ukvarjali z rekonstrukcijo slik oziroma manjkajočih pikslov [2]. Ena izmed različic našega pristopa je bila že predstavljena v članku [3],v katerem smo vhodno sliko obravnavali v smislu šahovnice, kjer smo kot bela polja označili piksle, katerih vrednost je znana, ter črna, katerih vrednost smo morali izračunati iz znanih vrednosti (belih polj). V tem članku predstavljamo dve izboljšani različici tega pristopa. Pri prvi izboljšani različici smo spremenili način procesiranja na procesiranje pikslov po vrsticah in drugačno izbiro pikslov v okolici obravnavanega piksla. V drugi različici najprej izvedemo detekcijo slabih napovedi, potem pa izvedemo popravek le teh. Razlike med napovedano in dejansko vrednostjo v vseh omenjenih različicah stisnemo z aritmetičnim kodiranjem [4].

2 Postopek stiskanja barvne slike

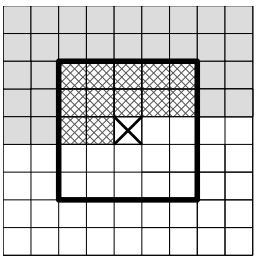
V tem razdelku bomo predstavili osnovne podrobnosti obeh izboljšanih različic našega pristopa. Barvno sliko procesiramo od zgornjega levega roba, najprej po stolpcih ter nato po vsaki vrstici in po vsaki barvni komponenti posameznega piksla. Algoritem 1 na kratko predstavlja osnovni koncept zgoraj navedenega pristopa. Izračun napovedi za trenutno obravnavan piksel opravimo iz že znanih, prej izračunanih vrednosti pikslov v okolici obravnavanega piksla. Za okolico vzamemo kvadrat določene velikosti w. V naših raziskavah smo uporabili velikost procesirnega okna $w = 5$ (slika 1). Na sliki so piksli, ki se uporabijo pri izračunu, označeni s poševno mrežo. Nahajajo se znotraj procesirnega kroga, levo od obravnavanega piksla ter zgoraj v predhodno obravnavanih vrsticah. Edina izjema je prvi obravnavan piksel; v tem primeru nimamo pikslov na levi strani obravnavanega piksla in moramo shraniti absolutno vrednosti posameznih komponent. V vseh ostalih primerih imamo vsaj eden ali več predhodno procesiranih pikslov na levi strani v isti vrstici ali v zgornjih vrsticah. Pri izračunu napovedi obravnavanega piksla uporabimo interpolacijsko funkcijo 1.

Algoritem 1 Postopek stiskanja slike

Vhod: I - vhodna barvna slika ($m \times n$), w - velikost procesirnega okna

$R \leftarrow \emptyset$
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** m **do**
    $\mathbf{p} \leftarrow I(j, i)$
    $S \leftarrow$ izbiraPikslov(j, i, w)
    $\acute{\mathbf{p}} \leftarrow$ izračunNapovediPiksla($\mathbf{p}$, S)
    $\mathbf{r} \leftarrow \mathbf{p} - \acute{\mathbf{p}}$
    $R \leftarrow R \cup \{\mathbf{r}\}$
  end for
end for
$C_R \leftarrow$ aritmetičnoKodiranje(R)
$RCC1 \leftarrow \{m, n, C_R\}$
return $RCC1$

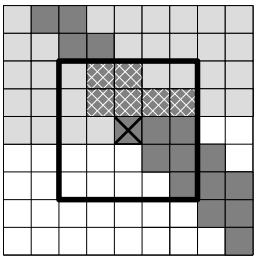
Izhod: stisnjena izhodna datoteka $RCC1$



Slika 1: Stiskanje brez popravka napovedi

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi(|\mathbf{x} - \mathbf{c}_i|) + ax + by + c, \qquad (1)$$

kjer predstavljajo vrednosti [λ_1,..., λ_n, a, b, c] rešitev sistema linearnih enačb. ϕ je radialna bazična funkcija [6], n je število pikslov v okolici obravnavanega piksla $\mathbf{x}$, ki mu določamo vrednost, [$\mathbf{c}_i$] so koordinate teh pikslov. Več o samem postopku izračuna najdemo v [2, 3, 5, 6]. Pri naših raziskavah smo za radialno bazično funkcijo ϕ uporabili linearno radialno bazično funkcijo ter način reševanja sistema linearnih enačb *PartialPivLU* [7].



Slika 2: Stiskanje s popravkom napovedi

Po izračunu koeficientov [λ_1,..., λ_n, a, b, c] lahko opravimo izračun vrednosti obravnavanega piksla oziroma napovemo njegovo vrednost (za vsako od barvnih komponent RGB - (rdečo, zeleno in modro)). Razlike $\mathbf{r}$ med napovedanimi in dejanskimi vrednostmi shranjujemo v seznam razlik R, ki ga na koncu stisnemo z aritmetičnim kodiranjem. Stisnjen bitni niz C_R ter velikost slike (m,n) shranimo v izhodno datoteko.

2.1 Različica s popravkom slabih napovedi

Pri analizi rezultatov algoritma 1 se je pokazalo, da se pojavljajo določena območja, predvsem robovi, in tudi druga območja, kot so osiroteli piksli, pri katerih se njihova vrednost bistveno razlikuje od vrednosti pikslov v njihovi okolici. Na podlagi teh ugotovitev smo izboljšali zgoraj opisan pristop. Glavna razlika je, da najprej preverimo vse napovedi in označimo piksle, pri katerih prihaja do slabih napovedi. V našem primeru smo kot kriterij uporabili prag slabe napovedi Δr. Razliko med napovedano in dejansko vrednostjo, višjo od praga slabe napovedi Δr, smo v maski M označili kot slabo napoved. Preverjali smo vse tri barvne komponente (RGB) piksla, in v kolikor katera koli razlika med napovedano in dejansko vrednostjo komponente preseže prag slabe napovedi Δr, je označen celoten piksel v maski M kot slaba napoved. Ta postopek je predstavljen v algoritmu 2.

Osnovni algoritem 1 tudi nekoliko spremenimo (vrstice 16 do 34 v algoritmu 2). Seveda, kjer so dobre napovedi oziroma majhne razlike med napovedano in dejansko vrednostjo pod pragom slabe napovedi, ostane izračun enak. V kolikor pa naletimo na piksel, ki je označen v maski M, da ima slabo napoved, uporabimo zgolj piksle, ki imajo znotraj procesirnega okna v maski M tudi oznako, da imajo slabo napoved. V tem primeru je večja verjetnost, da je obravnavan piksel podoben pikslom z

Algoritem 2 Postopek stiskanja slike s popravkom slabih napovedi

Vhod: I - vhodna barvna slika ($m \times n$), w - velikost procesirnega okna, Δr - prag slabe napovedi

▷ Nastavitev maske M za določitev lokacij slabih napovedi vrednosti piksla

1: **for** $j \leftarrow 1$ **to** n **do**
2: **for** $i \leftarrow 1$ **to** m **do**
3: $\mathbf{p} \leftarrow I(j, i)$
4: $S \leftarrow$ izbiraPikslov(j, i, w)
5: $\acute{\mathbf{p}} \leftarrow$ izračunNapovediPiksla(p, S)
6: $[r_r, r_g, r_b] \leftarrow |\mathbf{p} - \acute{\mathbf{p}}|$
7: **if** $|r_r > \Delta r| \vee |r_g > \Delta r| \vee |r_b > \Delta r|$ **then**
8: $M(j, i) \leftarrow 255$ ▷ slaba napoved
9: **else**
10: $M(j, i) \leftarrow 0$ ▷ dobra napoved
11: **end if**
12: **end for**
13: **end for**

▷ Izračun napovedi vrednosti pikslov

14: $R \leftarrow \emptyset$
15: $B \leftarrow \emptyset$
16: **for** $j \leftarrow 1$ **to** n **do**
17: **for** $i \leftarrow 1$ **to** m **do**
18: $\mathbf{p} \leftarrow I(j, i)$
19: **if** M(j,i) = 255 **then**

▷ Izberemo piksle znotraj procesirnega okna w, ki imajo tudi slabe napovedi

20: $S \leftarrow$ izbiraSlabihPikslov(j, i, M, w)
21: **if** $S = \emptyset$ **then**
22: $S \leftarrow$ izbiraPikslov(j, i, w)
23: **end if**
24: $\acute{\mathbf{p}} \leftarrow$ izračunNapovediPiksla($\mathbf{p}$, S)
25: $\mathbf{r} \leftarrow \mathbf{p} - \acute{\mathbf{p}}$
26: $B \leftarrow B \cup \{\mathbf{r}\}$
27: **else**

▷ Uporabimo enak postopek, kot pri algoritmu 1

28: $S \leftarrow$ izbiraPikslov(j, i, w)
29: $\acute{\mathbf{p}} \leftarrow$ izračunNapovediPiksla($\mathbf{p}$, S)
30: $\mathbf{r} \leftarrow \mathbf{p} - \acute{\mathbf{p}}$
31: $R \leftarrow R \cup \{\mathbf{r}\}$
32: **end if**
33: **end for**
34: **end for**
35: $C_R \leftarrow$ aritmetičnoKodiranje(R)
36: $C_B \leftarrow$ aritmetičnoKodiranje(B)
37: $M_{j2} \leftarrow$ shraniKotJBIG2(M)
38: $RCC2 \leftarrow \{m, n, M_{j2}, C_R, C_B\}$
39: **return** $RCC2$

Izhod: stisnjena izhodna datoteka $RCC2$

enakimi težavami pri določanju napovedi oziroma, da morda predstavlja rob ali drugo težavno območje. Na sliki 2 so, na primer, uporabljeni piksli (temnejše območje) znotraj procesirnega okna označeni s poševno mrežo. Tako na primer uporabimo za izračun robnih pikslov zgolj robne točke. V kolikor v okolici obravnavanega piksla slabe napovedi ni pikslov s slabo napovedjo, izračunamo napoved na enak način kot v algoritmu 1.

Ostanke oziroma razlike med napovedano in dejansko vrednostjo shranimo v dva seznama R in B. Oba stisnemo z aritmetičnim kodirnikom in dobimo dva bitna niza C_R in C_B. V izhodno datoteko moramo shraniti še masko M, ki označuje piksle slabih napovedi. Gre za monokromatsko sliko, ki jo lahko zelo učinkovito shranimo v formatu JBIG2 [1]. Izhodni niz M_{j2} prav tako shranimo v izhodno datoteko.

3 Rezultati

Na sliki 3 so prikazane slike, ki smo jih uporabili v naših eksperimentih. Vse uporabljene slike so barvne slike RGB v velikosti 512×512.

Naš pristop pri napovedovanju pikslov smo primerjali s popularnim formatom za stiskanje slik PNG (Portable Network Graphics), ki uporablja drugačne modele napovedi (več o tem v [8, 9]). Preglednica 1 je primerjava med različnimi formati shranjevanja slik. Z oznako RCC so označeni formati na podlagi naših različic, in sicer:

- RCC0 – je naš prvotni format, predstavljen v [3];
- RCC1 – je format, kjer smo piksle procesirali po vrsticah, kot je opisano v algoritmu 1;
- RCC2 – je format, kjer smo uporabili popravek napovedi, kot je opisan v algoritmu 2;

Rezultati kažejo, da smo uspeli izboljšati napovedi pikslov oziroma izboljšali stiskanje slik glede na format PNG (v preglednici so izboljšave označeno z odebeljeno pisavo), pa tudi glede na pristop, ki smo ga uporabili v preteklosti. Iz rezultatov je tudi razvidno, da pri višjih vrednostih praga slabe napovede Δr dobimo nekoliko boljše rezultate.

V preglednici 2 so prikazani podatki velikost posameznih podatkov je v izhodni datoteki RCC2. Opazimo, da se pri povečanju velikosti shranjevanja maske (M_{j2}) v formatu JBIG2 hkrati nekoliko poveča velikost shranjevanja razlik med napovedano in dejansko vrednostjo pikslov in obratno. Na sliki 4 so prikazane monokromatske slike, na katerih so z belo barvo označena območja slabih prvotnih napovedi. Uporabili smo prag slabe napovedi $\Delta r = 10$. Te monokromatske slike (masko M) zapišemo v formatu JBIG2, in so nujna informacija pri dekompresiji podatkov.

4 Zaključek

Rezultati so pokazali, da je možno izboljšati napovedovanje pikslov in s tem stiskanje barvnih slik. V nadaljevanju raziskav se bomo osredotočili na izboljšanje predstavljenega pristopa, morda z bolj optimalnim in ustreznejšim zaznavanjem slabih napovedi, tudi v smislu lažjega shranjevanja podatkov (maske) glede lokacije slabih napovedi pikslov ter ustreznejšega in boljšega shranjevanja teh podatkov v formatu JBIG2.

5 Zahvala

Projekt (Paradigma stiskanja podatkov z odstranjevanjem obnovljivih informacij, št. J2-4458) in raziskovalni program P2-0041 sta financirali Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije ter Czech Science Fundation (Project No. 23-04622L) iz državnega proračuna.

Literatura

[1] D. Salomon, G. Motta: Data Compression: The Complete Reference, 5th Edition. Springer, 2010.

[2] B. Lipuš, B. Žalik (2012). Efficient reconstruction of images with deliberately corrupted pixels, *Informatica*, 23(1), 47–63.

[3] B. Lipuš, S. Pečnik, A. Nerat, M. Šmolík , D. Strnad. Uporaba radialnih bazičnih funkcij za napovedovanje vrednosti pikslov pri brezizgubnem stiskanju rastrskih slik RGB. *Zbornik dvaintridesete mednarodne Elektrotehniške in računalniške konference*: 149-152

[4] I. H. Witten, M. N. Radford, J. G. Cleary (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6): 520-540.

[5] K. Uhlíř, V. Skala (2006). Radial basis function use for the restoration of damaged images, V: K. Wojciechowski, B. Smolka, H.Palus, R. Kozera, W. Skarbek, L. Noakes (uredniki), *Computer Vision and Graphics International Conference*, Springer, Vol. 32, 839–844.

[6] M. D. Buhmann (2003). *Radial Basis Functions*, Cambridge University Press.

[7] G. Guennebaud, B. Jacob in ostali (2023). Eigen v3, https://eigen.tuxfamily.org.

[8] G. Roelofs: PNG: The Definitive Guide. O'Reilly & Associates, Inc., 1999

[9] L. Lukač, A. Jeromel, L. Váša. A Short Overview of Prediction Methods for Image Compression. *Zbornik dvaintridesete mednarodne Elektrotehniške in računalniške konference*: 145-148

(a)

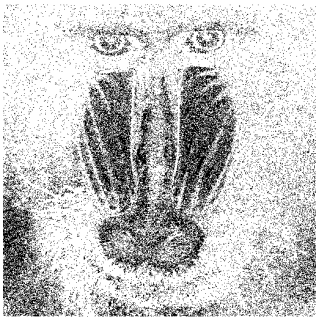
(b)

(c)

Slika 3: Testne slike: (a) Airplane. (b) Pepper. (c) Baboon.

(a)

(b)



(c)

Slika 4: Monokromatski prikaz območij, kjer absolutne razlike med napovedano in dejansko vrednostjo presežejo nastavljen prag slabe napovedi (v primeru na sliki je nastavljen $\Delta r = 10$) (z belo so označena območja slabih napovedi): (a) Airplane. (b) Pepper. (c) Baboon.

Preglednica 1: Primerjava stiskanja barvnih slik RGB (podano s povprečnim številom bitov na pixel - BPP): RCC0 – pristop, uporabljen v [3], RCC1 – procesiranje pikslov brez popravka napovedi, RCC2 – s popravkom napovedi in različnimi vrednosti praga slabih napovedi Δr

Slika	**BMP**	**PNG**	**RCC0**	**RCC1**	Δr	**RCC2**
Airplane	24	12,94	13,54	13,52	5	**12,53**
					10	**12,59**
					20	**12,90**
Pepper	24	15,44	15,44	**15,13**	5	**15,37**
					10	**15,34**
					20	**15,04**
Baboon	24	19,11	19,44	19,33	5	19,28
					10	19,16
					20	**18,97**

Preglednica 2: Prikaz velikosti podatkov pri pristopu s popravkom napovedi glede na različen prag slabe napovedi Δr (z oznako C_{sum} je označena skupna velikost C_R in C_B)

Slika	Δr	C_{sum} (kB)	M_{j2} (kB)	RCC2
Airplane	5	392,349	18,121	410,470
	10	400,738	11,711	412,449
	20	416,592	6,140	422,732
Pepper	5	476,740	27,012	503,752
	10	472,511	30,133	502,644
	20	482,932	9,965	492,897
Baboon	5	619,649	12,008	631,657
	10	605,258	22,731	627,989
	20	594,198	27,350	621,548