Elektrotehnički fakultet

OE4DOS

Treći domaći zadatak iz digitalne obrade slike

Autor:
Vuk Vukomanović 18-2014

Contents

1 Uvod

Izveštaj je podeljen u logičke celine, koje se odnose na taskove koji su zadati trećim domaćim zadatkom. U izveštaju se nalaze slike i rezultati koji su generisani mimo poslatih .m fajlova radi manje robosnusti. Shodno tome ceo dokument je podeljen u sledeće 2 celine:

1. Huffmanovo kodiranje
2. Custom kompresija

2 Hafmanovo kodiranje

Napomena: Kreiranje Hafmanove tabele, kodiranje, dekodirenje je implementirano u c++-u ali zbog nemogućnosti mex toolboxa da radi sa std::vector objektima matlab funkcije ih ne koriste. Kodovi (koji se mogu pokrenuti) se nalaze u folderu Cpp.

Prve tri tačke domaćeg zadatka su vezani za hafmanovo kodiranje.

Hafmanov kod je prefiks kod koji se koristi za kompersiju podataka bez gubitaka. Razlog zašto se hufmanov kod često koristi je imzeđu toga što on ima najmanjih broj kodnih simbola po simbolu izvora. Tj. po Šenonovoj prvoj teoremi rezulutujući kod je optimalan za fiksnu vrednost N, sa ograničenjem da se simboli izvora koduju jedan po jedan.

Sam postupak kodiranja simbola je sledeći.

1. Računanje verovatnoća pojavljivanja simbola
2. Sortiranje niza verovatnoća
3. Kombinovanje(sabiranje) 2 najmanje verovatnoće u novi simbol
4. Ukoliko su ostale samo 2 vrednosti stajemo, u suportonom se vraćamo na prvi korak

Drugi deo kodiranja je kodiranje svakog redukovanog izvora. To se radi tako što se počne od najmanjeg izvora i ,,ide,, se natrag do originalnog. Tako što simbol sa manjom verovatnoćom dobija kao deo svog koda nulu a veći jedinicu. U nastavku su date slike iz knjige Digital Image Processing koje predstavljaju objašnjen proces.

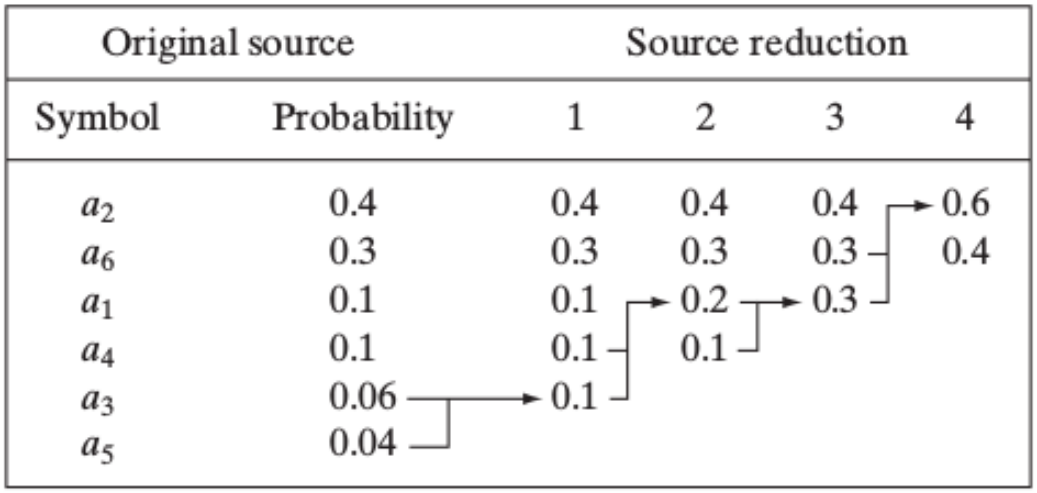

Figure 1: Prvi deo kodiranja

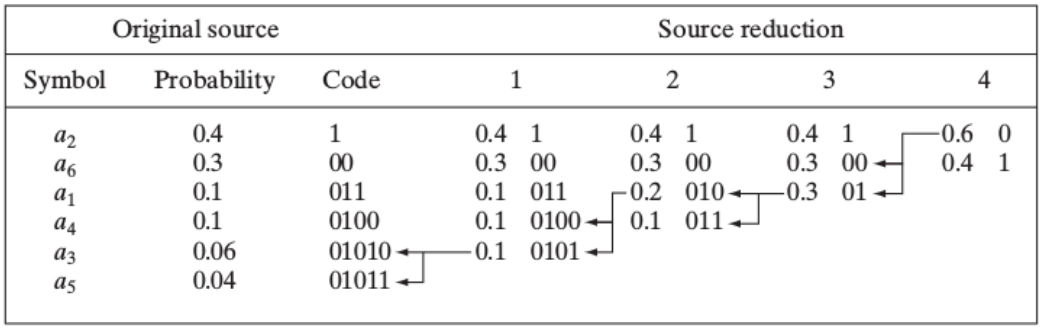

Figure 2: Drugi deo kodiranja

2.1 generate_huffman_code

Funkcija generate_huffman_code generiše Hafmanovu kodnu tabelu na osnovu ulaznog histograma slike, tj. na osnovu niza simbola i verovatnoća njihovih pojavljivanja.

Funkcija radi tako što od niza verovatnoća pravi binarno stablo. Počev od originalnog niza, u svakoj iteraciji, od 2 najmanje verovatnoće napravi novi roditelj nod čija je verovatnoća jednaka zbiru dece nodova. Ovaj postupak se ponavlja sve dok originalna struktura nema 2(ili 1 član zavisi od imple-

mentacije). Generisanje kodova se sada svodi na prolazak kroz binarno stablo i dodavanje 0 ukoliko idemo u levo podstablo i 1 ukoliko idemo u desno podstablo. Nakon generisanja kodova oni se po zahtevu domaćeg konvertuju u niz 8bitnih decimalnih brojeva.

2.2 encode_huffman

Funkcija encode_huffman kompresuje ulaznu sliku hufmanovim kodom. Funkcija prvo računa jedinstvene simbole(vrednosti) ulazne slike i njihove verovatnoće. Nakon toga koristeći generate_huffman_code generiše Hafmanovu kodnu tabelu koju koristi kao look up tabelu. Tj. za svaki element ulazne slike se koristi luk up tabela kako bi se dobio njen kod i dodao na izlaznu poruku. Tada je na izlazu niz jedinica i nula koji predstavlja kodovane vrednosti ulazne slike, taj kod se grupise u 8bitne brojeve.

2.3 decode_huffman

Funkcija decode_huffman dekoduje sliku koja je kodova hafmanovim kodiranje. Realizacija ove funkcije se razlikuje od one sa vezbi zato što ne koristi niz link već heš mapu. Na ovaj način je kompleksnost traženja simbola/koda smanjena sa **O(nlogn)** na **O(n)** zato što se pristup heš mapi odvija u **O(1)** za razliku od pretrage binarnog stabla(niza link) koja se odvija u **O(nlogn)**.

2.4 Rezultati

U nastavku su dati rezultati testiranja prethodne 3 funkcije na matricama **F** i **K**.

Rezultati za matricu **F**, koja je data u nastavku.

$$\begin{bmatrix} 26 & 85 & 198 & 26 \\ 85 & 26 & 198 & 198 \\ 26 & 26 & 42 & 26 \\ 42 & 42 & 26 & 26 \end{bmatrix}$$

Hafmanova tabela

symbol	length	word
26	1	128
42	3	96
85	3	64
198	2	0

Kompresovani podaci :

Compressed : 171, 91, 7, 152

Nakon dekodovana se dobija polazna matrica F, kao i što je očekivano. Rezultati za matricu **K**, koja je data u nastavku.

$$\begin{bmatrix} -85 & 0 & -12485 & 0 \\ 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ -85 & -85 & 0 & 2 \end{bmatrix}$$

Hafmanova tabela

symbol	length	word
-12485	3	64
-85	3	96
0	1	128
2	2	0

Kompresovani podaci :

Compressed : 113, 203 , 95, 128

Nakon dekodovana se dobija polazna matrica K, kao i što je očekivano.

3 Custom kompresija

U nastavku je dato objašnjeni koraci zadate kompresije, tj. zbog čega je koji korak primenjen i na kraju su dati rezultati kompresije.

3.1 Kompresija

Koraci im2dos kompresije su prikazani na slici ispod.

Figure 3: Proces kompresije

Nulti korak kompresije jeste proširivanje ulazne slike tako da se dobiju dimenzije koje su umnožak broja 8. Ovo se radi kako bi slika mogla biti podeljena u nepreklapajuće blokove veličine 8x8.

Prvi korak kompresije je block transform coding, tj. primena odredjene funkcije na svaki od 8x8 blokova slike. Cilj i razlog podele slike pa primene funkcije na svaki blok je da se dekorelišu pikseli svake podslike i da se što više infromacija spakuje u što više koeficijenata. Funkcija za transfromaciju ima dosta, razloga zašto na primer nije korišćena Furijeova transformacija umesto kosinusne ima više. Prvi je to što su nakon transformacije

koeficijente DCT-a realni brojevi, takođe FFT ima oštre diskonuitete (zbog pretpostavke periodičnosti), i što će se posle i rezultati prikazati koeficijenti DCT su više grupisani ka prvom, tj. u manjem broju koeficijenata možemo da sačuvamo više informacija. DCT nije optimalna transformacija za razliku od KL transfromacije, ali KL za svaki blok određuje optimalnu ,,funkciju,, što je komputaciono skoro nemoguće, DCT za svaki blok ima iste transformacione koeficijente što komputaciono mnogo olakšava stvari.

Nakon DCT-a se odstranjuje zadat broj najmanjih koeficijenata, na ovaj način je smanjen broj podataka a slika nije izgubila mnogo informacija.

Drugi korak kompresije je kvantizacija. U prethodnom koraku odbačeni su najmanji koeficijenti. Ideja kvantizacije je da finije od odbacivanja smanji količinu podataka koji su suvišni. Na primer ukoliko imamo neku vrednost koeficijenta 2.7 možemo da predpostavimo da nećemo dobiti mnogo lošiji rezultat ako zaokruzimo na 3 i na taj način smanjimo broj bita potreban za taj koeficijent. Odatle proizilazi ideja kvantizacije, tj. koeficijente koji su blizi prvom (oni koji nose najviše informacija) ćemo kvantizovati sa većom rezolucijom, koeficijente koji su dalji sa manjom rezolucijom. Ovo se realizuje kvatizacionom matricom, koja je unapred određena za svaki algoritam kompresije. Praktično na ovaj način veliki deo koeficijenata postavljamo na nulu.

Na primer prvi član zadate kvantizacione matrice je 16, to znači da će svi brojevi [0, 16] biti 0, svi od [17, 32] biti 2 itd. Za član matrice koji iznosi 101, znači da će svi brojevi manji od 100 biti zaokruženi na 0, što predstavlja agresivniju kvantizaciju. Ovaj korak se može unaprediti tako što se ne koristi uniformna kvantizacija.

Treći korak kompresije je prediktivno kodovanje DC koeficijenata. Cilj prediktivnog kodovanja je da se iskoristi sličnost koeficijenata i da se salje samo njihova razlika a ne i oni celi.

Četvrti korak kompresije je Hafmanovo kodovanje koje je objašnjeno u prvom delu izveštaja.

3.2 Dekompresija

Koraci dos2im dekompresije su prikazani na slici ispod.

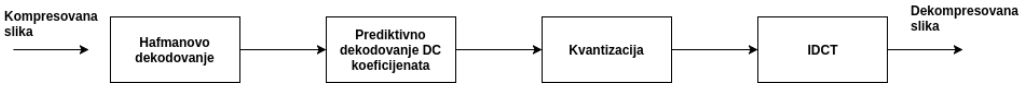

Figure 4: Proces kompresije

Dekompresije predstavlja inverzan proces kompresiji kao što se vidi na blok dijagramu.

3.3 Rezultati

Svi prikazani rezultati su dobijeni primenom kompresije na sliku lena.tiff koja je data u nastavku.

Figure 5: Ulazna slika

Kompresija je testirana za različite parametre kvaliteta kompresije, tj. broj DCT koeficijenata koji nisu postavljeni na nulu.

Za $quality = 64$ dobijeni su sledeći rezultati

$$PSNR = 36.4672$$

$$Stepen \quad kompresije = 13.5$$

Figure 6: Kompresovana slika, quality = 64

Vreme izvršavanja programa je $76,2$ sekunde.

Za $quality = 33$ dobijeni su sledeći rezultati

$$PSNR = 36.4665$$

$$Stepen \quad kompresije = 13.58$$

Figure 7: Kompresovana slika, quality = 33

Vreme izvršavanja programa je $75,6$ sekunde.

Za *quality* $= 15$ dobijeni su sledeći rezultati

$$PSNR = 36.1759$$

$$Stepen \quad kompresije = 13.7$$

Figure 8: Kompresovana slika, quality = 33

Vreme izvršavanja programa je $71,2$ sekunde.

Za $quality = 7$ dobijeni su sledeći rezultati

$$PSNR = 33,6428$$

$$Stepen \quad kompresije = 16$$

Figure 9: Kompresovana slika, quality = 7

Vreme izvršavanja programa je $52,3$ sekunde.

Za $quality = 1$ dobijeni su sledeći rezultati

$$PSNR = 24,09578$$

$$Stepen \quad kompresije = 67$$

Figure 10: Kompresovana slika, quality = 1

Vreme izvršavanja programa je $16,6$ sekunde.

Posmatrajući PSNR-ove za različite stepene kompresije, vidi se da je kao što je rečeno većina koeficijenata DCT-a koji su važni za kasniji prikaz slike blizu prvog koeficijenta. Tj. da prvi koeficijenti nose najveći broj infromacija. Za parametre 64,33 i 15 vizuelno se ne vide promene u odnosu na original, za parematar 7 se vidi blaga razlika, dok je za parametar 1 razlika i više nego očigledna.

Korak koji kada se izbaci dovodi do najmanjih gubitaka je kvantizacija. Razlog tome je opisan u procesu kompresije. Data kvantizaciona matrica veoma agresivno vrši kvantizaciju za koeficijente koji su udaljeni od prvog. Prethodni rezultati nam pokazuju kako će slika izgledati kada se određen broj koeficijenata postavi na 0 i na osnovu toga možemo da pretpostavimo

kvalitet rezultata kada kvantizacijom veliki broj postavimo na nulu ili na veoma male vrednosti.

U nastavku su prikazani rezultati koji potvrdjuju prethodnu pretpostavku Kada se izbaci kvantizacija dobijeni su sledeći rezultati

$$PSNR = 47.37$$

$$Stepen \quad kompresije = 1.7$$

Figure 11: Kompresovana slika, izbačena kvantizacija

Stepen kompresije je dosta manji jer sada izmedju ostalog postoji mnogo veći broj različitih simbola pri hafmanovom kodovanju.
Ulazna i izlazna slika nisu identične, jedan od razloga je što kada se izbaci kvantizacija koeficijenti i dalje moraju da se zaukruže na najbliži ceo broj kako pri Hafmanovom kodovanju ne bi bilo previše različih vrednosti. Stepen

kompresije se smanjio na 2, što je i logično da se toliko smanji jer kvantizacija predstavlja veliki deo kompresije.