Segmentacija slike postupkom k-srednjih vrijednosti

Haramija, Josip

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Segmentacija slike postupkom k-srednjih vrijednosti

Rijeka, rujan 2023.

Josip Haramija
0069088316

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni prijediplomski studij računarstva

Završni rad

Segmentacija slike postupkom k-srednjih vrijednosti

Mentor: prof.dr.sc. Ivo Ipšić

Rijeka, rujan 2023.

Josip Haramija
0069088316

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2023.

_____

Josip Haramija

Zahvala

Zahvaljujem mentoru prof. dr. sc. Ivi Ipšiću na pomoći tijekom pisanja ovoga rada i korisnim savjetima. Zahvaljujem obitelji, prijateljima i kolegama na podršci tijekom studiranja.

Sadržaj

Popis slika

Poglavlje 1

Uvod

Segmentacija slike je postupak rasčlanjivanja slike u segmente ili regije, odnosno grupiranje piksela po nekom predefiniranom kriteriju [1]. Postoje različite metode za segmentaciju slike koje se odabiru prema specifičnim potrebama projekta. Neke od metoda su grupiranje (engl. clustering) i metoda rasta regija (engl. regions-growing) [3].

Metoda rasta regija se koristi na način da odaberemo jedan ili više početnih piksela i postupno rastežemo regije tako da uključuju susjedne piksele koji zadovoljavaju određene uvjete. Metode grupiranja ili klasterizacije, poput metode k-srednjih vrijednosti, su općenitije i mogu se koristiti izvan konteksta segmentacije slike. Metoda rasta regija zahtjeva "sjemenke" (engl. seed points) ili početne piksele iz kojih će se regije širiti, za razliku od toga, metode grupiranja mogu krenuti od bilokoje kombinacije piksela i uče koji pikseli će biti centroidi. Kada proces segmentacije želimo precizno kontrolirati ili znamo lokacije početnih regije, korisnija će biti metoda rasta regija upravo zato što će se "sjemenke" ili početni pikseli moći precizno odrediti.

Jedna od poznatijih metoda grupiranja je metoda k-srednjih vrijednosti. Upravo je ova metoda među najraširenijima zbog svoje jednostavnosti, no ima i svoje mane. Najveća mana je što parametar K nije dinamičan, postoje metode koje isprobavaju različite vrijednosti parametra K kako bi došle do što boljeg rezultata, npr. metoda lakta (engl. elbow method), ali takve metode mogu biti vrlo vremenski zahtjevne. Još jedna mana je to što nasumičnim odabirom početnih centroida možemo utjecati

na konačni rezultat. Metodom k-srednjih vrijednosti++ (engl. k-means++) možemo minimizirati taj problem. Više o metodama lakta i "k-means++" u narednim poglavljima.

Segmentacija slike ima mnogo primjena u svakodnevnom životu, kao što su: medicinska dijagnostika, računalni vid, kartografija, video igre i mnoge druge. Iz navedenih primjena vidimo da je segmentacija slike vrlo važna tehnika koja nam omogućava dublje razumijevanje i bolje iskorištavanje vizualnih podataka i poboljšava način na koji interagiramo s tehnologijom i okolinom.

Cilj ovoga rada je prikazati algoritam k-srednjih vrijednosti te usporediti različite varijante algoritma: automatski i ručni odabir parametra K, automatska i ručna inicijalizacija centroida, kao i rezultate dobivene grupiranjem u različitim prostorima boja, konkretno RGB i HSV.

U idućem poglavlju predstaviti će se algoritam u teoriji, koraci algoritma i neke metode koje mogu poboljšati učinkovitost algoritma. Treće poglavlje biti će fokusirano na implementaciju algoritma, opisati će se svaki korak algortima i potkrijepiti primjerima iz koda. U četvrtom poglavlju biti će predstavljeni rezultati segmentacije za različite kategorije slika te će isti biti ukratko analizirani. Posljednje poglavlje biti će zaključak u kojemu će se ukratko opisati ideje za daljnje unaprjeđenje i korištenje algoritma.

Poglavlje 2

Algoritam k-srednjih vrijednosti

2.1 Teorijski temelj algoritma

Algoritam k-srednjih vrijednosti, često nazivan algoritmom k-sredina, je popularna tehnika za klasteriranje podataka. U kontekstu segmentacije i analize slike, ovaj algoritam se često koristi kako bi se grupirali pikseli slika u različite regije prema njihovim sličnostima u boji ili teksturi. Ovaj pristup omogućava automatsko razdvajanje različitih objekata ili regija unutar slike, olakšavajući daljnju analizu i obradu.

Uzmimo skup $X = \{x_i\}, i = 1, ..., n$ koji se sastoji od n točaka koje će se segmentirati u k klastera: $C = \{c_k\}, k = 1, ..., MAX\_K$. Algoritam k-srednjih vrijednosti pronalazi klastere takve da je ukupna udaljenost svih točaka unutar klastera od središta klastera što manja 2.1. Krajnji cilj algoritma je minimizirati sumu udaljenosti svih točaka od središta klastera kojemu su dodijeljene 2.2 (μ_k predstavlja srednju vrijednost klastera ili centroid). S obzirom na to da radimo sa trodimenzionalnim točkama (svaka ima parametar R, G i B), sume će se računati za svaki od tih parametara [2].

$$J(C_k) = \sum_{x_i \in C_k} (||x_{iR} - \mu_{kR}|| + ||x_{iG} - \mu_{kG}|| + ||x_{iB} - \mu_{kB}||) \tag{2.1}$$

$$J(C) = \sum_{k=1}^{K\_MAX} J(C_k) \tag{2.2}$$

Algoritam ima četiri glavna koraka: inicijalizacija centroida, dodjela podataka klasterima, ponovno računanje centroida te iterativno ponavljanje drugog i trećeg koraka do konvergencije.

Parametar k predstavlja broj klastera koji se žele dobiti. Ručno određivanje ovog parametra može dovesti do subjektivnih rezultata, gdje su neki objekti podijeljeni na više klastera nego što bi trebali biti ili obrnuto. Automatsko određivanje k može uključivati metode poput "lakta" ili analize varijance. Također, ručno određivanje parametra K može biti i korisno ako unaprijed sa sigurnošću znamo koliko bi klastera trebalo biti ili koliko ih želimo dobiti.

Također, odabir prostora boja može značajno utjecati na rezultate segmentacije [3] [4]. Upotreba RGB prostora boja fokusira se na komponente crvene, zelene i plave boje svakog piksela. S druge strane, HSV prostor boja (nijansa, zasićenost, vrijednost) može bolje modelirati ljudsko opažanje boja jer zasebno razdvaja informacije o tonu, zasićenosti i svjetlini.

2.2 Koraci algoritma

Prvi korak algoritma uključuje odabir početnih k centroida. To se obično radi nasumično, ali može se koristiti i naprednija strategija poput k-means++ inicijalizacije. K-means++ osigurava da su početni centroidi ravnomjerno raspoređeni u podacima, čime se smanjuje vjerojatnost da algoritam zapne u lokalnom minimumu.

Nakon odabira početnih centroida slijedi dodjela podataka klasterima. Svaki piksel slike dodjeljuje se klasteru čija je središnja točka (centroid) najbliža tom pikselu. Za određivanje bliskosti piksela središnjoj točki klastera najčešće se koristi Euclidova udaljenost. Za točke $A(x_1, y_1) i B(x_2, y_2)$ formula glasi:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.3}$$

Kada su svi pikseli dodijeljeni klasterima, računaju se nove središnje točke (centroidi) za svaki klaster. Ovo se radi tako da se za svaki klaster izračuna aritmetička sredina svih piksela koji su trenutno dodijeljeni tom klasteru. To jest, za svaku

komponentu boje (u RGB ili HSV prostoru) izračunava se prosječna vrijednost.

Konačno, postupci dodjele podataka klasterima i ponovnog računanja centroida ponavljaju se iterativno. Tijekom svake iteracije, pikseli se ponovno dodjeljuju klasterima temeljem udaljenosti od novih centroida, a zatim se centroidi ponovno računaju na temelju novih grupacija. Algoritam se ponavlja sve dok se centroidi klastera više ne mijenjaju (ili mijenjaju vrlo malo) između dvije uzastopne iteracije ili dok se ne dostigne maksimalni broj iteracija definiran unaprijed. To pomaže osigurati konvergenciju algoritma.

Važno je napomenuti da algoritam k-srednjih vrijednosti može biti osjetljiv na početni izbor centroida, što može dovesti do različitih rezultata na temelju različitih početnih položaja. Jedan način za smanjenje ovog utjecaja je višestruko pokretanje algoritma s različitim početnim centroidima i odabirom najboljeg rezultata.

2.3 Metode za automatski odabir parametra k i početnih centroida

U ovom poglavlju opisati će se metoda za automatski odabir parametra k: metoda lakta, te metoda za odabir početnih centroida: k-means++. Te dva aspekta algoritma k-srednjih vrijednosti ujedno su i glavne točke na kojima se algoritam može optimizirati [5]. Odabir parametra k će utjecati na preciznost i koherentnost konačnog rezultata, dok odabir početnih centroida utječe na konvergenciju algoritma te na rezultate grupiranja.

2.3.1 Metoda lakta

Metoda lakta (engl. "elbow method") [6] je heuristička tehnika koja se koristi za određivanje optimalnog broja klastera, parametra k, u algoritmu k-srednjih vrijednosti. Ova metoda pomaže u vizualnoj identifikaciji točke na grafu gdje se promjene u varijabilnosti unutar klastera značajno smanjuju. Naziv "metoda lakta" dolazi od izgleda grafa koji vrlo često sliči na "lakat".

Do točke "lakta" može se doći i računskim putem, umjesto vizualnim, no taj

način dalje povećava mogućnost krivog odabira točke "lakta", odnosno parametra k. Kod vizualnog odabira točke "lakta", veliku ulogu igra iskustvo i poznavanje prirode podataka, dok kod računskog odabira te faktore zanemarujemo.

Suma kvadrata unutar klastera (engl. within cluster sum of squares ili WCSS) ili varijacija unutar klastera [7] je mjera koja označava ukupnu sumu kvadrata udaljenosti svih točaka unutar istog klastera od centra tog klastera. Ideja je da što je WCSS manji, to su točke unutar klastera bliže centru, što ukazuje na kompaktniju klasterizaciju.

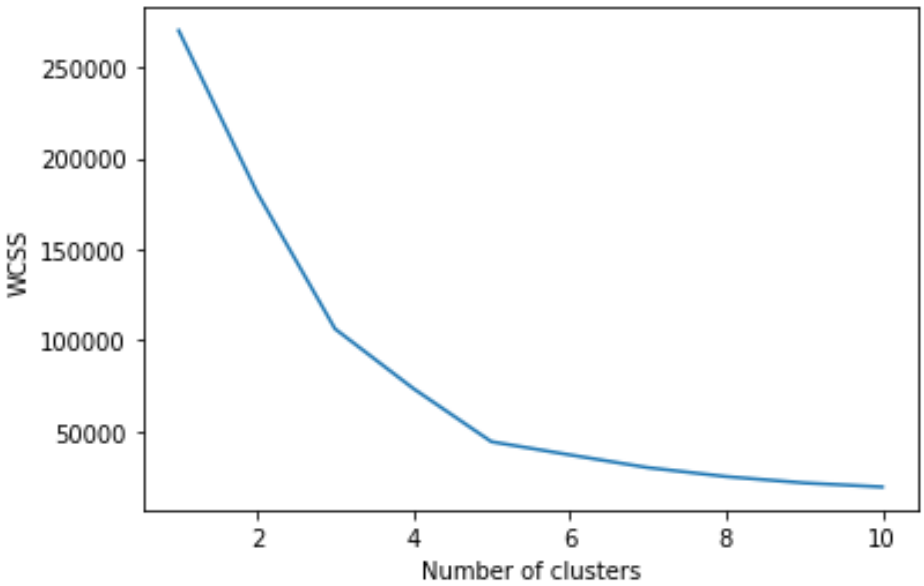
Ideja metode lakta je da se algoritam k-srednjih vrijednosti pokrene za različite vrijednosti k, obično od jedan do nekog maksimalnog broja klastera te se za svaki broj klastera izračunava suma kvadratnih udaljenosti svakog podatka od središnje točke klastera kojem pripada (WCSS). Na temelju tih vrijednosti izrađuje se graf čijom analizom tražimo točku gdje smanjenje WCSS postaje manje izraženo, tj. gdje se krivulja počinje savijati poput lakta. Ova točka na grafu smatra se optimalnim brojem klastera, jer predstavlja trenutak kada dodavanje dodatnih klastera ne donosi značajno smanjenje varijabilnosti unutar klastera te se broj klastera dodijeljen toj točki odabire kao parametar k.

Važno je napomenuti da metoda lakta nije uvijek apsolutno precizna, i može postojati situacija gdje nije jasno vidljiva točka "lakta". Također, ova metoda može biti osjetljiva na oblik podataka i nije uvijek najbolji izbor za određivanje broja klastera. U nekim složenijim slučajevima, mogu biti potrebne dodatne metode za procjenu optimalnog broja klastera.

Na Slici 2.1 [8] vidimo primjer grafa dobivenog metodom lakta. Na y-osi prikazane su vrijednosti suma kvadrata unutar klastera, dok je na x-osi prikazan broj klastera. Na ovom grafu vrlo je lako uočiti točku "lakta" kada odaberemo 5 za broj klastera.

2.3.2 K-means++

Nasumičan odabir početnih centroida za algoritam k srednjih vrijednosti može dovesti do mnogih problema. Algortiam može zapeti u lokalnom minimumu, različiti rezultati pri svakom pokretanju programa, spora konvergencija te prevelika osjetlji-



Slika 2.1 Primjer grafa dobivenog metodom lakta.

vost algoritma.

K-means++ je poboljšana strategija za odabir početnih centroida u algoritmu k-srednjih vrijednosti [9]. Ova strategija pomaže boljem rasporedu početnih centroida, čime se smanjuje vjerojatnost da algoritam zapne u lokalnom minimumu ili konvergira sporije. Prvi centroid se nasumično bira iz skupa podataka. To je početna točka od koje će algoritam početi grupirati druge podatke. Za svaki preostali podatak računa se kvadratna udaljenost između tog podatka i najbližeg centroida koji je već odabran. Pomoću dobivenih kvadratnih udaljenosti stvaraju se vjerojatnosne distribucije, gdje podaci s većom kvadratnom udaljenosti imaju veću vjerojatnost biti odabrani kao sljedeći centroid. Idući centroid odabire se nasumično iz skupa podataka, koristeći vjerojatnosti izračunate u prethodnom koraku. Podaci s većom udaljenosti od postojećih centroida imaju veću vjerojatnost biti odabrani kao centroidi, čime se postiže ravnomjernija raspodjela početnih točaka. Malo jednostavniji način je izračunati Euclidove udaljenosti (2.3) svih podataka od najbližeg centroida te za novi centroid odabrati podatak sa najvećom udaljenosti. Računanje kvadratne ili Euclidove udaljenosti te odabir novog centroida ponavljaju se dok se ne odaberu

svi potrebni centroidi.

Poglavlje 3

Implementacija algoritma u programskom jeziku C

3.1 Priprema okruženja za implementaciju algoritma

U prethodnom poglavlju istraženi su teorijski aspekti algoritma k-srednjih vrijednosti. Ovo poglavlje usmjeriti će se na praktični aspekt primjene ovog algoritma, implementirajući ga koristeći programski jezik C. Implementacija ovog algoritma omogućiti će bolje razumijevanje samog algoritma i rezultata koji će biti predstavljeni u idućem poglavlju, cjelokupan kod: [10].

Slike učitavamo u formatu PPM (engl. Portable Pixmap Format), jednostavnom tekstualnom formatu za pohranu slika. Format je osmišljen kako bi omogućio laku razmjenu slika između različitih računalnih sustava bez potrebe za složenim kodiranjem ili dekodiranjem. PPM format podržava monokromatske (crno-bijele), sive i slike u boji. Iako je PPM format jednostavan za korištenje, nije efikasan za pohranu velikih slika zbog svog tekstualnog prikaza [11].

Za spremanje podataka o slici koristimo dvije strukture. U jednu spremamo osnovne podatke o slici, poput visine i širine, listu centroida koji će kasnije biti dodijeljeni pikselima, udaljenost od dodijeljenog centroida te podatke o pikselima. Svaki piksel sastoji se od tri vrijednosti, svaka od njih predstavlja udio jedne od boja: crvene, zelene i plave (RGB) u boji piksela. Ukoliko se koristi prostor boja HSV,

umjesto crvene, zelene i plave, svaki piksel predstavljati će nijansa ili ton boje (engl. hue), zasićenost bojom ili njen intenzitet (engl. saturation) i vrijednost, odnosno svjetlina boje (engl. value).

3.2 Inicijalizacija centroida

Centroide inicijaliziramo metodom k-means++ zbog njezinog poboljšanog pristupa odabiru početnih centroida, koji ima tendenciju raspršivanja početnih položaja tako da se postigne brža konvergencija algoritma i bolja kvaliteta konačnih klastera.

```
246     nextCentrIndex = getNextCentr(image,
    totalDistPerNumOfCentroids[k - MIN_K]);
247     centroids[i] = image->colour[nextCentrIndex];
248     totalDistPerNumOfCentroids[k - MIN_K] = 0;
249
250     for(j = 0; j < num_of_data_points; ++j){
251         image->centr[j] = assignCentr(image->colour[j],
    centroids, k);
252         image->dist[j] = calcDpDistance(image->colour[j],
    centroids[ image->centr[j] ]);
253         totalDistPerNumOfCentroids[k - MIN_K] += image->dist[j];
254     }
```

Navedeni kod predstavlja isječak funkcije za računanje centroida. Ovaj dio koda izvršava se onoliko puta koliki je parametar k, osim za prvu iteraciju petlje. U prvoj iteraciji definiramo prvi centroid koji će biti odabran nasumično ili ručnim odabirom piksela te dodijeljujemo sve piksele njegovom klasteru. Iz koda vidimo da postoji funkcija "getNextCentr()" koja računa indeks piksela koji će postati idući centroid (idući isječak). Nakon odabira novog centroida preraspodjeljujemo piksele s obzirom na bliskost novodefiniranim centroidima. Drugim riječima, računamo kojim pikselima je novi centroid najbliži u odnosu na ostale centroide te ih dodijeljujemo njegovom klasteru. Za svaku vrijednost k također računamo sumu udaljenosti unutar klastera koja će se koristiti za odabir novog centroida, kao i za potrebe metode lakta.

```
304     const int num_of_data_points = image->height * image->width;
305     int i = 0;
306     double randVal = ((double) rand() / RAND_MAX) * totalDist;
307     double partialSum = 0.0;
308
309     while(partialSum < randVal){
310         partialSum += image->dist[i];
311         ++i;
312         if(i == num_of_data_points) i = 0;
313     }
314
315     return i;
```

Drugi isječak koda, kao što je već navedeno, predstavlja funkciju za izračun novog centroida "getNextCentr()". Varijabla "randVal" predstavlja nasumičnu vrijednost između nule i sume udaljenosti unutar klastera, dok se varijabla "partialSum" koristi kao privremena varijabla u koju se pribrajaju, jedna po jedna, udaljenosti pojedinih piksela od njihovih centroida. U trenutku kada parcijalna suma dostigne ili prijeđe nasumično odabranu vrijednost, petlja se zaustavlja i piksel na čijem se indeksu zaustavila postaje novi centroid. Na taj način osiguravamo da će pikseli sa većim udaljenostima imati veće šanse da postanu novi centroidi jer pribrajanjem njihovih udaljenosti parcijalnoj sumi brže se približavamo nasumičnoj vrijednosti.

3.3 Segmentacija

```
200     while(convergence != 0){
201         for(i = 0; i < num_of_data_points; ++i){
202             image->centr[i] = assignCentr(image->colour[i],
    centroids, k);
203         }
204
```

```
205         convergence = 0;
206
207         for(i = 0; i < k; ++i){
208             sumR = 0;
209             sumG = 0;
210             sumB = 0;
211             count = 0;
212
213             for(j = 0; j < num_of_data_points; ++j){
214                 if(image->centr[j] == i){
215                     sumR += image->colour[j].r;
216                     sumG += image->colour[j].g;
217                     sumB += image->colour[j].b;
218                     count++;
219                 }
220             }
221
222             if(centroids[i].r != ( sumR / count ) ||
    centroids[i].g != ( sumG / count ) || centroids[i].b != (
    sumB / count ))
223                 convergence++;
224
225             centroids[i].r = sumR / count;
226             centroids[i].g = sumG / count;
227             centroids[i].b = sumB / count;
228         }
229     }
```

Segmentacija ili klastering, implementirana je navedenom funkcijom. Petlja se izvršava do konvergencije, odnosno dok god postoji neka promjena. Konkretno u ovom slučaju, provjerava se postoji li promjena u boji centroida, to jest njegovom položaju u 3D prostoru definiranom crvenom, zelenom i plavom bojom. U glavnoj petlji se prvo pikseli dodijeljuju klasterima najbližih centroida, ovisno o tome je li se vrijednost centroida promijenila pikseli se mogu maknuti iz jednog klastera i dodijeliti drugome. Nakon dodjele, centroidi se računaju kao aritmetička sredina svih

piksela u pripadajućim klasterima. Konkretno, računaju se aritmetičke sredine svih pojedinih komponenata, dakle: crvene, zelene i plave boje zasebno i te vrijednosti postaju nove vrijednosti centroida. Drugim rječima centroid poprima boju definiranu aritmetičkom sredinom svake od tri komponente, svih piksela unutar pripadajućeg klastera, zasebno.

3.4 Odabir parametra k metodom lakta

Ukoliko želimo koristiti metodu lakta za odabir optimalnog parametra k, morati ćemo navedene korake izvršavati u petlji za vrijednosti k od minimalne do maksimalne. Minimalnu i maksimalnu vrijednost postavljamo ručno kao raspon vrijednosti u kojemu očekujemo optimalnu vrijednost parametra k. Uz navedene korake, moramo računati i sumu kvadratnih udaljenosti za sve klastere zajedno, drugim riječima: ukupnu udaljenost za svaku vrijednost parametra k. Postoji nekoliko načina za izračun točke "lakta", no s obzirom na to da je ta metoda vizualna i glavna ideja metode je da se "lakat" prepozna na grafu, nijedan način računskog odabira nije savršen i treba naći onaj koji je zadovoljavajući za potrebe zadatka. Spomenuti graf, prikazan na Slici 2.1, prikazuje vrijednosti ukupnih udaljenosti (y-os) za svaku vrijednost parametra k (x-os). Metoda korištena u ovom zadatku je računanje najveće promjene u ukupnoj udaljenosti za dvije susjedne vrijednosti parametra k. Metoda nije polučila željene rezultate jer je davala nekonzistentne rezultate za različite raspone vrijednosti k te su rezultati uglavnom težili što manjoj vrijednosti k umjesto optimalnoj.

3.5 Ukratko o prostoru boja HSV

Kao što je već opisano u prošlom poglavlju HSV prostor boja za prikaz boje piksela umjesto kombinacije tri boje koristi nijansu, zasićenost i vrijednost. Ovaj model koristi se za opisivanje i definiranje boja na način koji je intuitivniji i ljudima prirodniji od uobičajenog RGB (crvena, zelena, plava) modela.

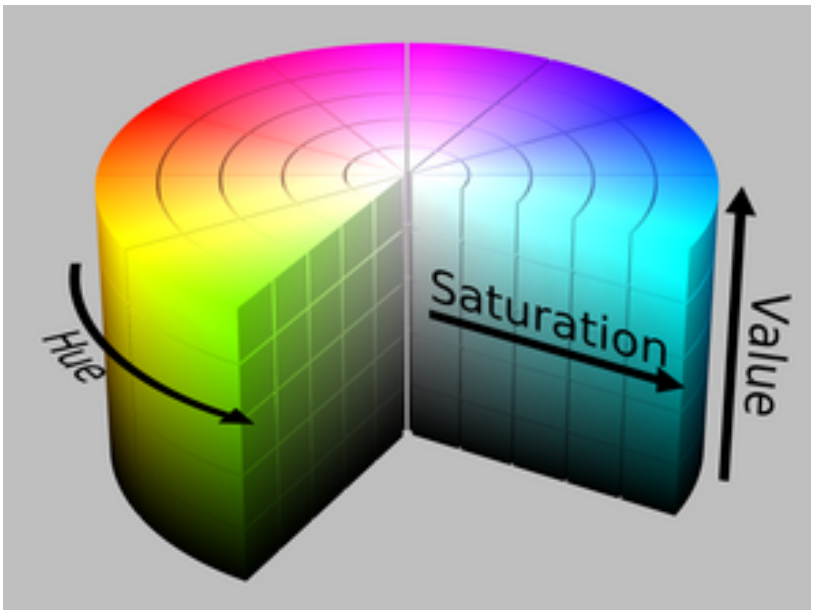
Nijansa predstavlja osnovni spektralni ton boje. Ovaj parametar pokriva cijeli spektar boja i određuje percepciju boje, poput crvene, zelene, plave itd. Vrijednosti

se izražavaju kutem u krugu boja, gdje 0° odgovara crvenoj, 120° zelenoj, 240° plavoj boji te drugim bojama za preostale vrijednosti do punog kruga odnosno 360°.

Zasićenost označava intenzitet boje ili čistoću. Veća zasićenost znači življe boje, dok niža zasićenost rezultira bljeđim ili pastelnim nijansama. Ukoliko je zasićenost jednaka nuli piksel će biti nijansa sive.

Konačno, vrijednost predstavlja svjetlinu boje. Ukoliko je vrijednost ovog parametra jednaka nuli, piksel će biti crn, dok će za vrijednost 1 ili 100% biti bijel.

Vizualni prikaz HSV prostora boja [12] možemo vidjeti na slici 3.1:



Slika 3.1 Vizualni prikaz HSV prostora boja u obliku cilindra

3.6 Implementacija u prostoru boja HSV

Implementacija segmentacije u prostoru boja HSV je ista kao za prostor boja RGB, uz iznimku da se u slučaju modela HSV vrijednosti boja piksela moraju pretvoriti iz RGB u HSV. Pretvorbe se vrše nakon učitavanja slike, u smjeru RGB u HSV te na kraju, prije ispisa nove slike u smjeru HSV u RGB.

3.6.1 Konverzija iz RGB u HSV

Konverzija iz RGB formata u HSV jednostavnja je [13]. Normaliziramo vrijednosti R, G i B tako što ih podijelimo sa 255 (u kodu vrijednosti "rNorm", "gNorm", "bNorm", zatim odredimo koja od tih vrijednosti je najveća - u kodu "cmax", a koja najmanja - u kodu "cmin" te izračunamo njihovu razliku - varijabla "diff". Parametar V se ne treba ni računati jer je jednak varijabli "cmax". Zasićenost će biti jednaka nuli ukoliko je "cmax" jednako nuli, a inače ćemo ju dobiti tako da "diff" podijelimo sa "cmax". Nijansu je malo kompliciranije izračunati, ona ovisi o tome koja komponenta RGB modela je najveća te se računa na sljedeći način:

$$H = \begin{cases} 0, & \text{za } diff = 0, \\ (60 \times (gNorm - bNorm) \div diff + 360) \mod 360, & \text{za } cmax = rNorm \\ (60 \times (bNorm - rNorm) \div diff + 120) \mod 360, & \text{za } cmax = gNorm \\ (60 \times (rNorm - gNorm) \div diff + 2400) \mod 360, & \text{za } cmax = bNorm. \end{cases}$$

U kodu to izgleda ovako:

```
142     for(i = 0; i < num_of_data_points; ++i){
143
144         rr = 0;
145         gg = 0;
146         bb = 0;
147         rr += image->colour[i].r;
148         gg += image->colour[i].g;
149         bb += image->colour[i].b;
150         rNorm = rr / 255;
151         gNorm = gg / 255;
152         bNorm = bb / 255;
153
154         cmax = maxVal(rNorm, gNorm, bNorm);
155         cmin = minVal(rNorm, gNorm, bNorm);
156         diff = cmax - cmin;
```

```
157
158         image->colourHSV[i].v = cmax * 100;
159
160         if(cmax == 0){
161             image->colourHSV[i].s = 0;
162         } else{
163             image->colourHSV[i].s = (diff / cmax) * 100;
164         }
165
166         if(diff == 0.0){
167             image->colourHSV[i].h = 0;
168         } else if(cmax == rNorm){
169             image->colourHSV[i].h = fmod(60 * ( (gNorm - bNorm)
    / diff ) + 360, 360.0);
170         } else if(cmax == gNorm){
171             image->colourHSV[i].h = fmod(60 * ( (bNorm - rNorm)
    / diff ) + 120, 360.0);
172         } else{
173             image->colourHSV[i].h = fmod(60 * ( (rNorm - gNorm)
    / diff ) + 240, 360.0);
174         }
175     }
```

3.6.2 Konverzija iz HSV u RGB

Konverzija iz HSV formata u RGB malo je kompleksnija. Prije početka konverzije moramo se uvjeriti da su vrijednosti varijabli S (zasićenost) i V (vrijednost/svjetlina) normalizirane, odnosno između vrijednosti nula i jedan. Ukoliko je vrijednost tona ili vrijednosti/svjetline nekog piksela jednaka nuli, R, G i B vrijednosti računamo tako da V pomnožimo sa 255. U tom slučaju dobiti ćemo nijansu sive.

```
208     hh = centroids[i].h;
209         sNorm = centroids[i].s / 100;
```

```
210         vNorm = centroids[i].v / 100;
211
212         if(sNorm == 0 || hh == 0){
213             rr = (int) vNorm * 255;
214             gg = (int) vNorm * 255;
215             bb = (int) vNorm * 255;
216
217         }
218         ...
```

Ako vrijednosti H i V nisu jednake nuli moramo računati pomoćne varijable: C (kroma/intenzitet, engl. chroma), X (međuvrijednost, engl. intermediate value) te koeficijent m. Navedene pomoćne varijable računaju se na sljedeći način:

$$C = sNorm * vNorm \tag{3.1}$$

$$X = C * (1 - |((hh \div 60) \mod 2) - 1|) \tag{3.2}$$

$$m = vNorm - C \tag{3.3}$$

Gdje su "sNorm" i "vNorm" normalizirane vrijednosti S i V, a "hh" je vrijednost H trenutnog piksela. Za potrebe računanja pomoćne varijable napisane su funkcije "decAbs" i "decMod" koje omogućavaju računanje apsolutne vrijednosti i primjenu modulo operatora nad decimanlim vrijednostima. Kada smo izračunali pomoćne vrijednosti možemo računati privremene vrijednosti R, G i B, u kodu zapisane kao "rr", "gg" i "bb".

```
218         ...
219         else{
220             C = sNorm * vNorm;
221             X = C * (1 - decAbs( decMod(hh / 60.0, 2.0) - 1 ) );
222             m = vNorm - C;
```

```
223
224             if(hh < 60){
225                 rr = C;
226                 gg = X;
227                 bb = 0;
228             } else if(hh >= 60 && hh < 120){
229                 rr = X;
230                 gg = C;
231                 bb = 0;
232             } else if(hh >= 120 && hh < 180){
233                 rr = 0;
234                 gg = C;
235                 bb = X;
236             } else if(hh >= 180 && hh < 240){
237                 rr = 0;
238                 gg = X;
239                 bb = C;
240             } else if(hh >= 240 && hh < 300){
241                 rr = X;
242                 gg = 0;
243                 bb = C;
244             } else if(hh >= 300){
245                 rr = C;
246                 gg = 0;
247                 bb = X;
248             }
249
250             centroidsRGB[i].r = round( (rr + m) * 255 );
251             centroidsRGB[i].g = round( (gg + m) * 255 );
252             centroidsRGB[i].b = round( (bb + m) * 255 );
253         }
254     }
```

Iz koda vidimo da se privremene RGB komponente postavljaju na različite vrijednosti na temelju tona boje "hh". Kada je ton boje manji od 60 stupnjeva privremena

crvena komponenta posatviti će se na pomoćnu vrijednost C, zelena na vrijednost X, a plava na nulu. Ako je ton veći ili jednak 60, a manji od 120 privremene RGB komponente poprimiti će druge vrijednosti i tako za svaki sektor od 60 stupnjeva. Naposljetku, privremenim vrijednostima se pribraja pomoćna vrijednost m te se skaliraju na vrijednosti [0, 255] i zaokružuju na najbliži cijeli broj. Kako je prikazano u formulama:

$$\begin{aligned} centroid_{xR} &= \lfloor (rr + m) * 255 \rceil, \\ centroid_{xG} &= \lfloor (gg + m) * 255 \rceil, \\ centroid_{xB} &= \lfloor (bb + m) * 255 \rceil. \end{aligned} \tag{3.4}$$

Poglavlje 4

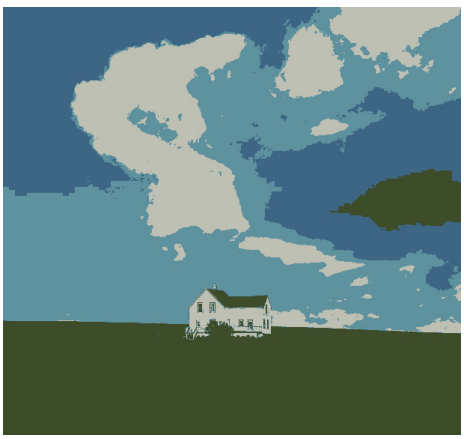
Primjeri i rezultati

4.1 Testirane varijacije

Za potebe ovog rada testirano je nekoliko slika. Biti će prikazane originalne slike, slike dobivene segmentacijom u RGB prostoru boja i slike dobivene segmentacijom u HSV prostoru boja. Također prikazati će se slike segmentirane za različite vrijednosti k: k = 4, k = 10 te za detaljnije slike sa više boja, k = 20. Konačno, prikazati će se nekoliko slika na kojima će se moći vidjeti proces klasterizacije, odnosno dio algoritma koji ponavlja drugi i treći korak kako je navedeno u potpoglavlju 2.2.

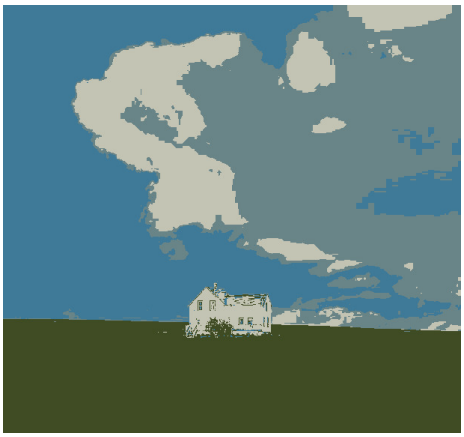
4.1.1 Slika s malo detalja

Slika 4.1 Originalna slika kuće



Slika 4.2 k = 4, RGB

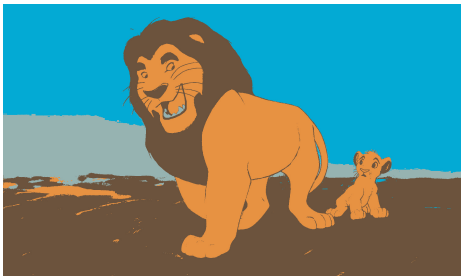
Slika 4.3 k = 10, RGB



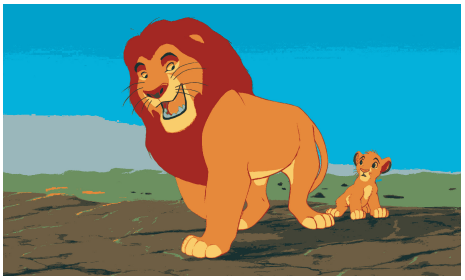
Slika 4.4 k = 4, HSV

Slika 4.5 k = 10, HSV

Slika 4.6 Originalna slika iz crtića s malo detalja

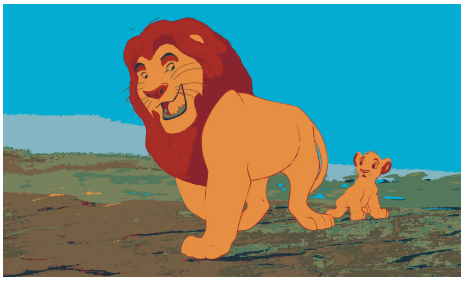


Slika 4.7 k = 4, RGB



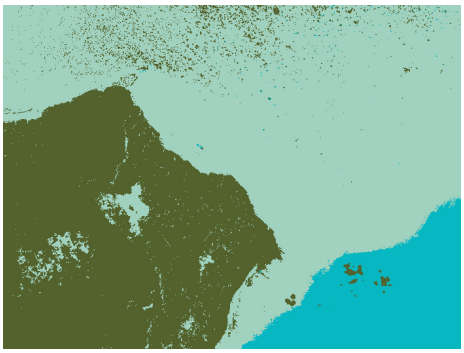
Slika 4.8 k = 10, RGB

Slika 4.9 k = 4, HSV

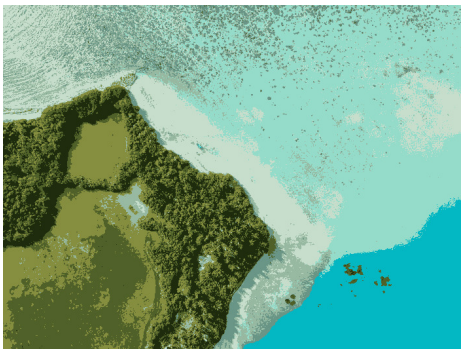


Slika 4.10 k = 10, HSV

Slika 4.11 Originalna slika plaže iz zraka

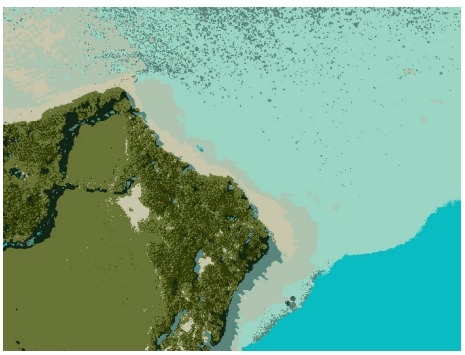


Slika 4.12 k = 3, RGB



Slika 4.13 k = 8, RGB

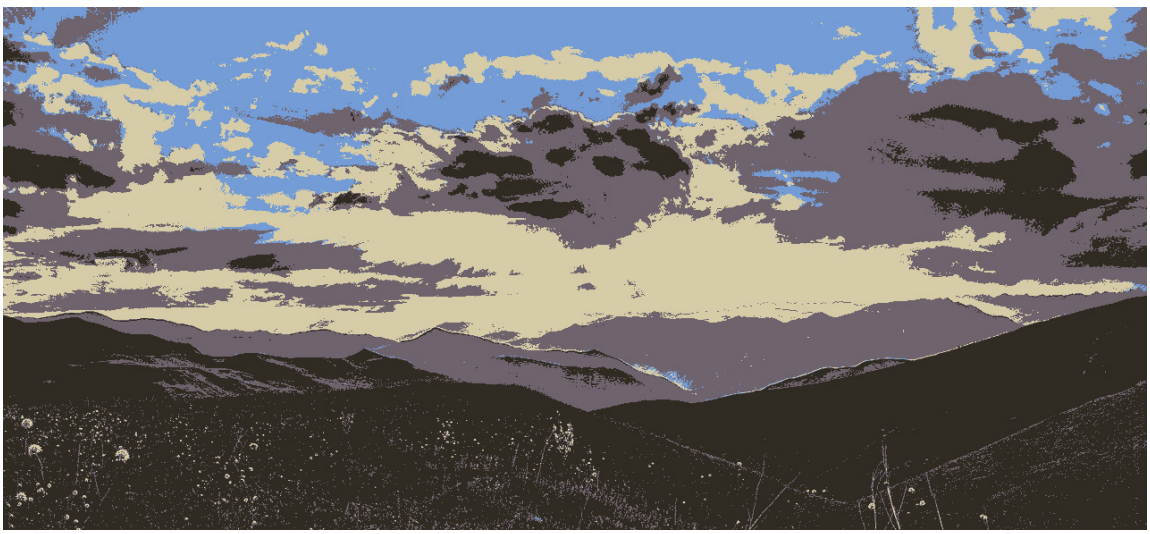
Slika 4.14 k = 3, HSV



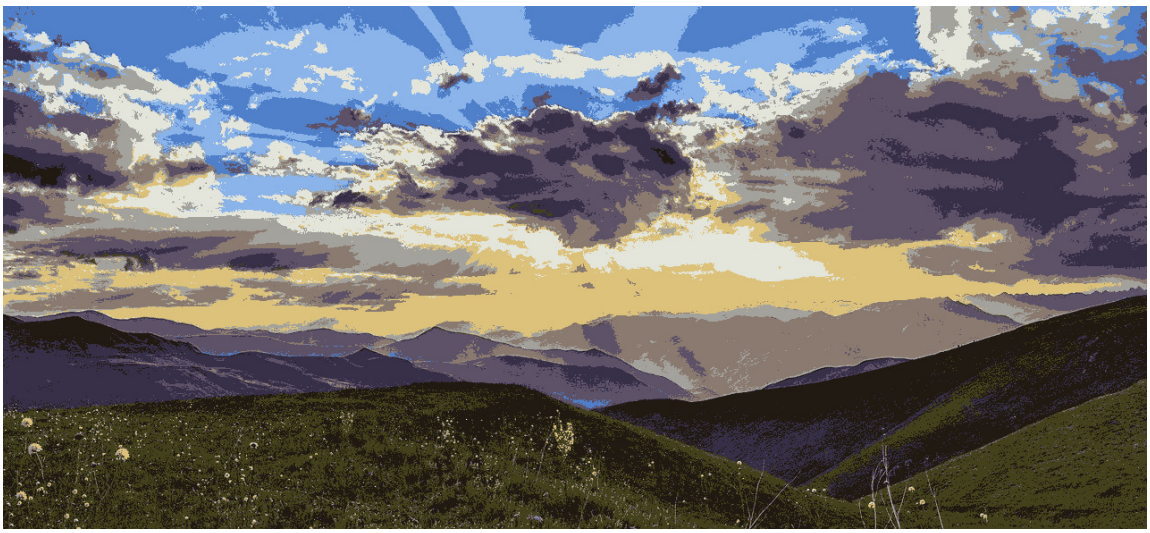
Slika 4.15 k = 8, HSV

4.1.2 Slika s puno detalja

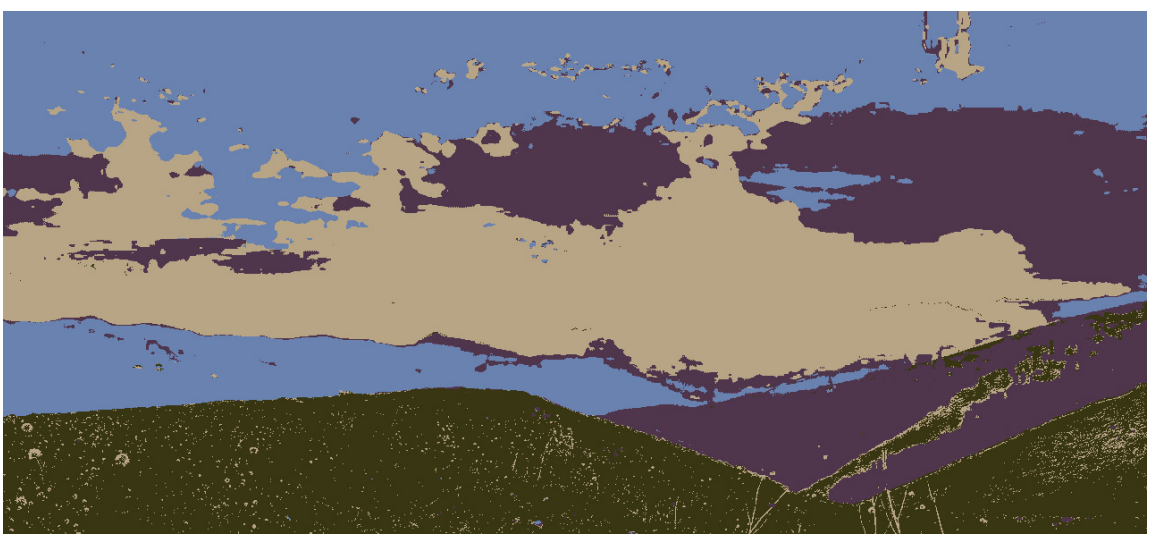
Slika 4.16 Originalna slika prirode s puno različitih boja



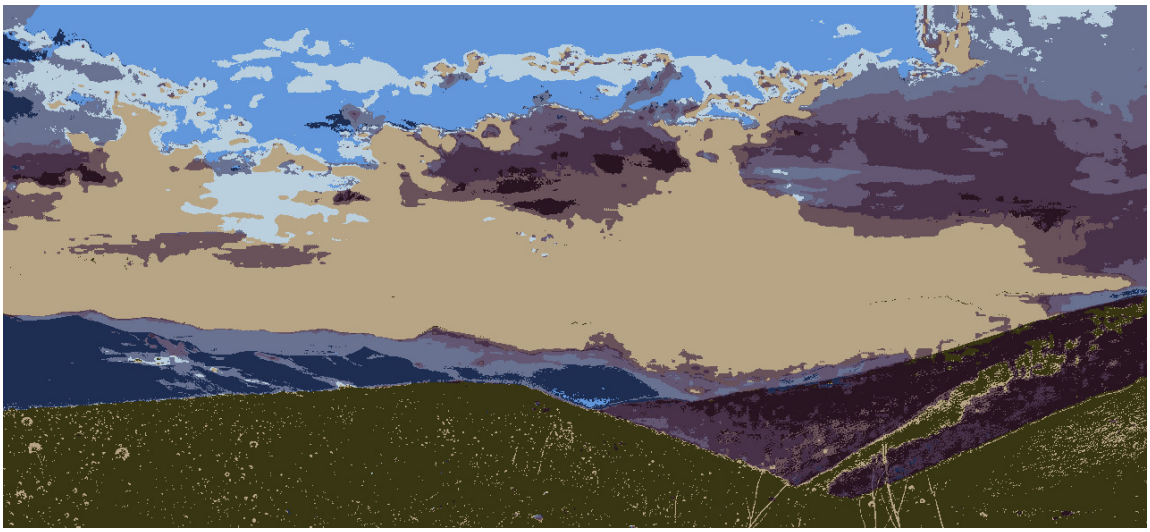
Slika 4.17 k = 4, RGB



Slika 4.18 k = 10, RGB



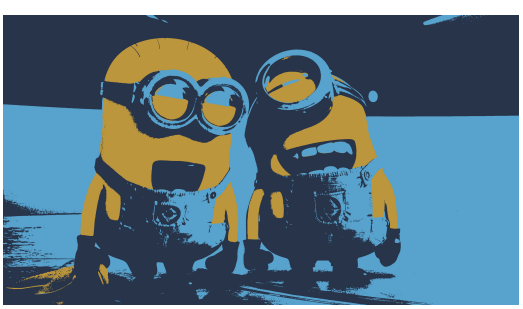
Slika 4.19 k = 4, HSV



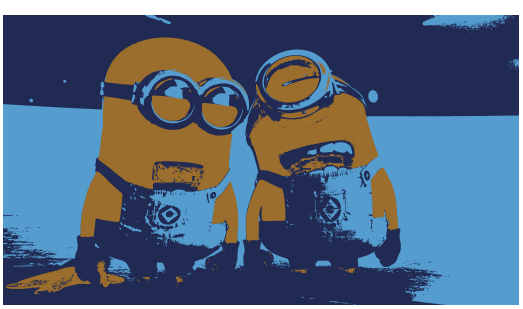
Slika 4.20 k = 10, HSV

4.1.3 Računalno generirana slika

Slika 4.21 Originalna računalno generirana slika

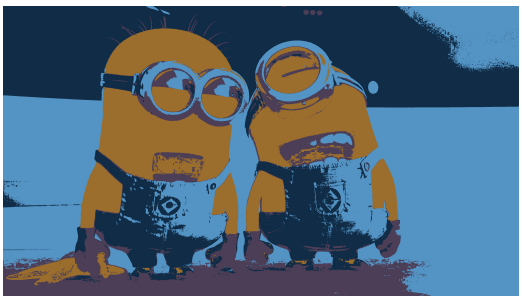


Slika 4.22 k = 3, RGB



Slika 4.23 k = 3, HSV

Slika 4.24 k = 4, RGB



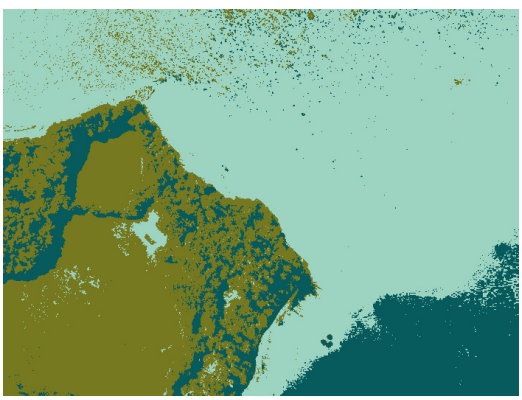
Slika 4.25 k = 4, HSV

Slika 4.26 k = 10, RGB

Slika 4.27 k = 10, HSV

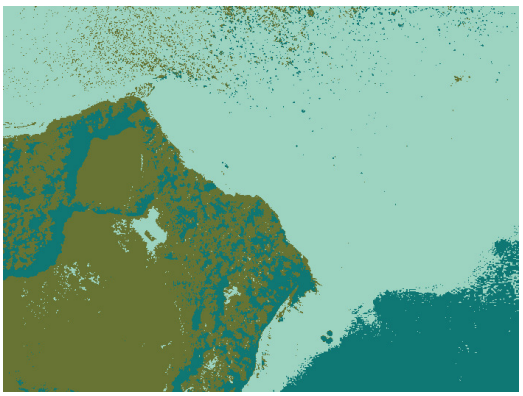
4.2 Proces klasterizacije

Slika 4.28 Originalna slika

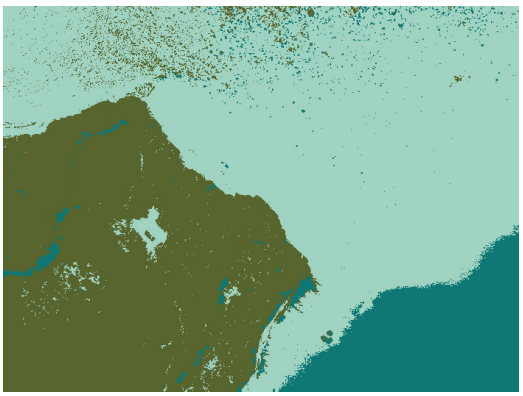


Slika 4.29 Prva iteracija

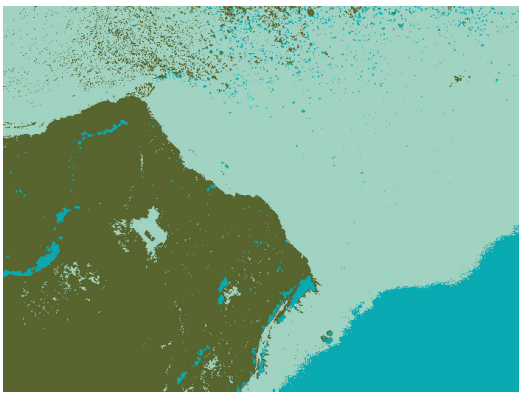
Iz prikazanih slika vidimo da su centroidi klastera već u prvoj iteraciji 4.29 vrlo dobro raspodijeljeni zahvaljujući metodi k-means++. Na ostalim slikama vidi se proces klasterizacije u kojemu se sve točke dodijele najbližem centroidu te se pomoću tih točaka računa nova vrijednost za svaki centroid 2.2.



Slika 4.30 Treća iteracija



Slika 4.31 Peta iteracija



Slika 4.32 Šesta iteracija

Slika 4.33 Osamnaesta iteracija

4.3 Analiza rezultata

Kao što se može vidjeti, slike su podijeljene u tri potpoglavlja. Prvo potpoglavlje prikazuje slike sa malo detalja, drugo sa puno detalja te treće je računalno generirana slika. U opisu svake slike piše koliki je parametar k korišten za segmentaciju i je li segmentacija provedena u RGB ili HSV prostoru boja.

Prvi set slika prikazuje kuću na brežuljku 4.1 - 4.5. Slika je vrlo jednostavna, nema puno detalja niti različitih boja. Ako usporedimo slike dobivene segmentacijom u RGB sa onima u HSV prostoru boja, vidjeti ćemo da HSV prostor boja puno bolje razdvaja različite boje ili tonove boja, dok na RGB prostor jak utjecaj ima i intenzitet boje. To se vrlo dobro vidi u trećem potpoglavlju na računalno generiranim slikama. Na slikama 4.24 - 4.27 vidmo da RGB prostor boja sjenu na licu tretira kao zasebnu

boju, dok HSV prostor istu sjenu prikazuje u istoj boji kao i ostatak lica. Time je pokazano kako sjene i intenzitet boje imaju puno manji utjecaj na segmentaciju u HSV prostoru boja nego na RGB prostor boja.

U nekim slučajevima rezultat HSV segmentacije ima puno više šuma nego rezultat RGB kao što se može vidjeti na drugom setu slika 4.7 - 4.10). Treći set slika 4.12 - 4.15 daje rezultat suprotan prethodnom setu i vidimo da u ovom slučaju RGB segmentacija daje više šuma. Možemo zaključiti da će količina šuma biti različita od slike do slike te će u nekim slučajevima biti bolje koristiti RGB prostor boja, dok će u drugim slučajevima bolji rezultat polučiti HSV prostor. Jedno od riješenja za veliku količinu šuma u slici može biti i filter šuma ali više o tome u idućem poglavlju.

Četvrti set slika i posljednji kojega se još nismo dotakli prikazuje sliku prirode sa puno detalja i različitih nijansi, Slike 4.17 - 4.20. Iz ovih primjera vidimo da HSV prostor bolje razlikuje dijelove slika i može donekle raspoznati oblake, planine u daljini i blizini kamere. Dok RGB segmentacija, pogotovo za veći parametar k (Slika 4.18), bolje prikazuje detalje i opet, kao što je navedeno u prethodnom primjeru ima manje šuma nego HSV segmentacija.

Poglavlje 5

Zaključak

U ovom radu detaljno su objašnjene teorijske osnove algoritma k-srednjih vrijednosti i njegova primjena u segmentaciji slike te implementacija istoga u programskom jeziku C. Također je istraženo kako se algoritam ponaša u HSV i RGB prostoru boja. Rezultati su pokazali da je algoritam k-srednjih vrijednosti učinkovit alat za segmentaciju slika, ali da njegova uspješnost ovisi o mnogim faktorima, uključujući izbor inicijalnih centroida, broj klastera te prikladnost korištenog prostora boja.

Segmentacija slike u HSV prostoru boja često se pokazala korisnom za segmentaciju temeljem nijansi boje bez obzira na svjetlinu i zasićenost, dok RGB prostor često grupira boje koje nisu slične po tonu ali jesu po svjetlini (npr. grupirati će svjetlocrvenu sa svjetlonarančastom ali neće uključiti tamnocrvenu). Ovisno o konkretnoj primjeni, korisno bi bilo eksperimentirati s različitim prostorima boja ili kombinacijama istih kako bi se postigli najbolji rezultati.

Važno je napomenuti da algoritam k-srednjih vrijednosti ima neka ograničenja. Na primjer, osjetljiv je na početne pozicije centroida i zahtijeva unaprijed zadan broj klastera. Što se tiče odabira početnih centroida, u ovom je radu uspješno prikazana metoda k-means++. Za odabir parametra k spomenuta je metoda lakta, no ta metoda je vizualna i nije ju lako automatizirati na način da bude primjenjiva u većini slučajeva već se mora prilagođavati specifičnom zadatku. Još jedno od mogućih poboljšanja segmentacije bi bila kombinacija algoritma k-srednjih vrijednosti sa detektorom rubova poput Cannyjevog detektora. Kombinacijom rezultata dobivenih

tim metodama mogli bi eliminirati dio šuma i bolje detektirati rubave, a samim time i objekte na slici.

Segmentacija slike pomoću algoritma k-srednjih vrijednosti predstavlja koristan alat za različite primjene, ali zahtijeva pažljiv odabir parametara i prostora boja ovisno o konkretnoj situaciji. Daljnja istraživanja i eksperimentiranje s ovom metodom mogu dovesti do poboljšanja i optimizacije za različite scenarije.

Bibliografija

[1] Singh Chauhan, N.: "Introduction to Image Segmentation with K-Means clustering", s Interneta, https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html, 04. rujna 2023.

[2] Jain, A. K.: "Data clustering: 50 years beyond K-means", Pattern Recognition Letters 31, 2010, pp. 651-666.

[3] Chen, W.; Shi, Y. Q.; Xuan, G.: "Identifying computer grahics using HSV color model and statistical moments of characteristic functions", Proceedings of IEEE International Conference on Multimedia and Expo, srpanj 2007, pp. 1123–1126.

[4] Chen, T. W.; Chen, Y. L.; Chien, S. Y.: "Fast image segmentation based on K-Means clustering with histograms in HSV color space", 2008 IEEE 10th Workshop on Multimedia Signal Processing, studeni 2008, pp. 322-325.

[5] Mayo, M.: "Centroid Initialization Methods for k-means Clustering", s Interneta, https://www.kdnuggets.com/2020/06/centroid-initialization-k-means-clustering.html, 04. rujna 2023.

[6] Wikipedia: "Elbow method (clustering)", s Interneta, https://en.wikipedia.org/wiki/Elbow_method_(clustering), 04. rujna 2023.

[7] Gupta, A.: "Elbow Method for optimal value of k in KMeans", s Interneta, https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/, 04. rujna 2023.

[8] Saji, B.: "Elbow Method for Finding the Optimal Number of Clusters in K-Means", s Interneta, https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/, 04. rujna 2023.

[9] Arthur, D.; Vassilvitskii, S.: "k-means++: The Advantages of Careful Seeding", SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, siječanj 2007, pp. 1027-1035.

[10] Haramija, J: "k-means_segmentation", s Interneta, https://github.com/jharamija/k-means_segmentation, 04. rujna 2023.

[11] Netpbm: "ppm", s Interneta, https://netpbm.sourceforge.net/doc/ppm.html, 04. rujna 2023.

[12] Wikipedia: "HSL and HSV", s Interneta, https://en.wikipedia.org/wiki/HSL_and_HSV, 04. rujna 2023.

[13] Bansal, S.: "C Program to Change RGB color model to HSV color model", s Interneta, https://www.tutorialspoint.com/c-program-to-change-rgb-color-model-to-hsv-color-model, 04. rujna 2023.

Sažetak

Ovaj rad istražuje primjenu algoritma k-srednjih vrijednosti u kontekstu segmentacije slika u HSV i RGB prostoru boja. Prvo poglavlje pruža teorijski pregled algoritma k-srednjih vrijednosti, objašnjavajući njegovo osnovno načelo i način rada. Nakon toga, rad se fokusira na praktičnu implementaciju algoritma u programskom jeziku C, što omogućava čitatelju dublje razumijevanje njegove primjene.

Sljedeća dva poglavlja istražuju segmentaciju slika u HSV i RGB prostoru boja. HSV prostor boja često je bolji izbor za segmentaciju temeljenu na nijansama boja i svjetlini, dok je RGB prostor bolji za razdvajanje objekata sličnih boja (ne nužno istih), ali različitih svjetlina. Analizirane su prednosti i ograničenja ovih dvaju pristupa, uz naglasak na dobivene rezultate.

Rezultati segmentacije pokazuju da je algoritam k-srednjih vrijednosti učinkovit alat za izdvajanje objekata na slikama, ali njegova uspješnost ovisi o različitim faktorima kao što su inicijalni centroidi i broj klastera. Također se ukazuje na potrebu za prilagodbom izbora prostora boja ovisno o konkretnoj primjeni.

Konačno, zaključak rada sumira glavne nalaze i ističe važnost pažljivog odabira parametara i prostora boja prilikom primjene algoritma k-srednjih vrijednosti za segmentaciju slika. Naglašava se potreba za daljnjim istraživanjem i eksperimentiranjem kako bi se postigla optimizacija i poboljšanje ove metode za različite scenarije i primjene u budućnosti.

Ključne riječi — **segmentacija, klastering, k-srednje vrijednosti, metoda lakta**

Abstract

This paper explores the usage of k-means algorithm for image segmentation in RGB and HSV colour space. In the first chapter, a theoretical overview of the algorithm is provided, explaining its basic principle and method of operation. Subsequently, the paper focuses on the practical implementation of the algorithm in C programming

language. Practical implementation enables the reader to gain a deeper understanding of its real-world application.

The following two chapters explore image segmentation in RGB and HSV colour spaces. HSV colour space is often the better choice for segmentation based on colour hues and brightness, while RGB space is a better choice for separating objects of similar colour (not necessarily of same colour) and different brightness. Advantages and limitations of the two approaches were analyzed, with an emphasis on the obtained results.

The segmentation results demonstrate that the k-means clustering algorithm is an effective tool for extracting objects in images, but its success is greatly impacted by various factors such as initial centroids and the number of clusters. Additionally, there is an indication of the need to adapt the choice of color space depending on the specific application.

In conclusion, the paper summarizes the main findings and highlights the importance of careful parameter and color space selection when applying the k-means clustering algorithm for image segmentation. There is an emphasis on the need for further research and experimentation to achieve optimization and improvement of this method for various scenarios and applications in the future.

Keywords **— segmentation, clustering, k-means, elbow method**