SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3943

Vrednovanje postupka semantičke segmentacije temeljenog na slučajnim šumama

Ivan Fabijanić

Zagreb, lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 10. ožujka 2015.

ZAVRŠNI ZADATAK br. 3943

Pristupnik: **Ivan Fabijanić (0119007943)**
Studij: Računarstvo
Modul: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Vrednovanje postupka semantičke segmentacije temeljenog na slučajnim šumama**

Opis zadatka:

Rad razmatra segmentaciju stvarnih slika u semantičke razrede poput ceste, automobila, stabla itd. Pretpostavljamo da se semantička oznaka svakog slikovnog elementa određuje na temelju njegovog kvadratnog susjedstva. Predmet rada je analiza, eksperimentalno vrednovanje i možebitno poboljšanje nedavno razvijene metode temeljene na slučajnim stablima.

U okviru rada, potrebno je iz literature proučiti različite pristupe za ostvarivanje semantičke segmentacije. Posebnu pažnju posvetiti postupcima koji se temelje na slučajnim šumama. Uhodati postupak za označavanje semantičkih razreda na razini slikovnog elementa u slikama. Označiti skupove slika za učenje, validaciju i testiranje. Analizirati prethodno razvijenu izvedbu semantičke segmentacije u suradnji s njenim autorom. Validirati hiperparametre i vrednovati generalizacijsku točnost postupka. Prikazati i ocijeniti ostvarene rezultate.

Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:

Izv. prof. dr. sc. Siniša Šegvić

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Krešimir Fertalj

Djelovođa:

Doc. dr. sc. Ivica Botički

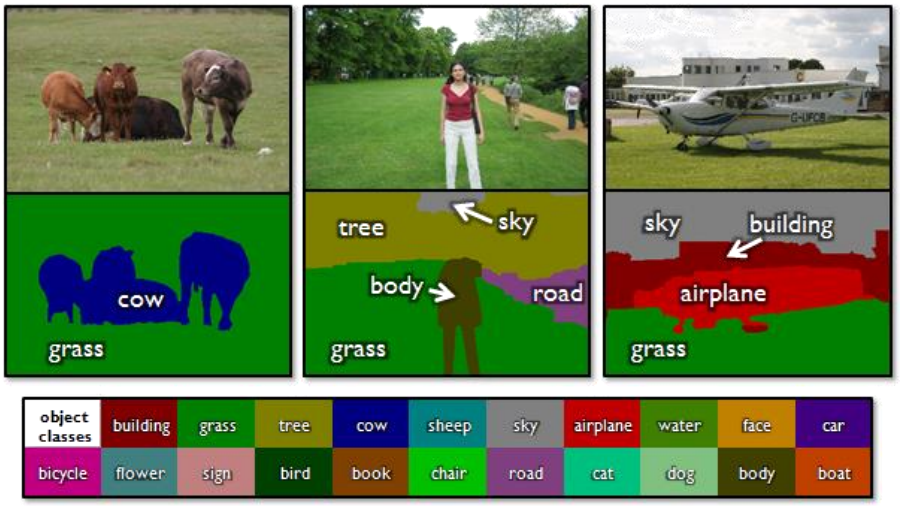
Velika zahvala mentoru prof. dr. sc. Siniši Šegviću na utrošenom vremenu, savjetima i pomoći oko izrade ovog rada

Sadržaj

1. Uvod

Računalni vid je područje umjetne inteligencije koje se bavi prikupljanjem, obradom, analiziranjem te razumjevanjem slika i općenito višedimenzionalnih podataka iz stvarnog svijeta sa svrhom izvlačenja korisnih numeričkih ili simboličkih podataka. Pojednostavljeno, računalni vid pokušava do određene mjere oponašati načine na koji ljudski mozak i pripadajuća osjetila percipiraju okolinu oko sebe. Zbog njegovih nebrojenih mogućih primjena to se područje nalazi u velikom zanimanju znanstvene zajednice i industrije koja stalno traga za poboljšanjima na postojećim algoritmima ili daljnjim istraživanjima dolazi do novih. Jedno od područja koje se intenzivno istražuje jest mogućnost učenja računala da prepoznaje objekte iz stvarnog svijeta kao što su npr. automobil, čovjek, znak ili nešto drugo kada se oni nalaze na slikama ili videu. Jedan od načina na koji se to može postići jest metoda semantičke segmentacije.

Semantička segmentacija jest metoda računalnog vida koja pokušava preko različitih karakteristika slike (vrijednosti boje po pikselu, različit kontast regija na slici, broj i usmjerenost crta na slici, itd.) izvući korisne informacije te preko njih podijeliti sliku na različite regije interesa. Te regije su najčešće neki objekti koji se žele naći na slikama.



Slika 1: Primjeri semantičke segmentacije slika [7].

Načini na koji se to može postići su mnogobrojni, a jedan od njih je metoda semantičke segmentacije temeljena na slučajnim šumama. Ta metoda uključuje

učenje više stabala odluke preko određenog broja slika sa traženim značajkama te se kasnije uz pomoć tih istreniranih stabla vrši test odluke kojim se određuje kojem razredu pripada regija slike koji se testira. Ovaj rad se prvenstveno bavi upravo tom metodom. U radu se obrađuje rad semantičke segmentacije sa slučajnim stablima, kvaliteta segmentacije te ono najbitnije, praćenje promjene u kvaliteti segmentacije ovisno o iznosima parametrima koji određuju rad algoritma , npr. broj stabala, veličina prozora za detekciju, itd. Ekperimentalni dio rada je odrađen na setu slika „KITTI road“ koji se može dohvatiti na web stranicama Karlsruhe Institute of Tehnologya[1].

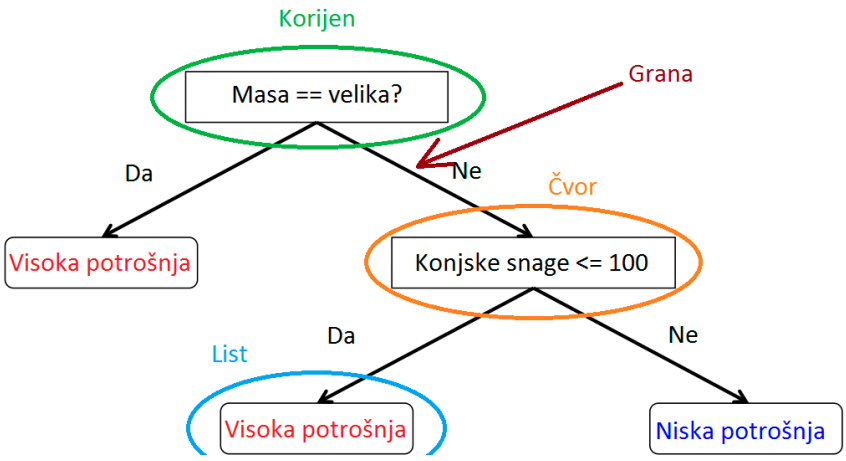
[1] http://www.cvlibs.net/datasets/kitti/eval_road.php

2. Stablo odlučivanja (decision tree)

Stablo odlučivanja (eng. decision tree) je osnovni element slučajnih šuma koje se koriste u ovome radu stoga ih treba definirati i dati primjer njihove uporabe.

2.1. Opća definicija

Stablo odlučivanja je model koji se u strojnom učenju koristi za regresiju i klasifikaciju. Ako su varijable kategorične ili diskretizirane (npr. ocjena na nekom predmetu) onda se stablo koristi za klasifikaciju, a ako su varijable kontinuirane onda za regresiju. Za semantičku segmentaciju stablo se koristi za klasifikaciju i ono je prediktivni model koji koristi binarna pravila koje primjenjuje na atribut da bi se izračunala krajnja željena vrijednost tj. klasa objekta na slici. Stablo odlučivanja sadrži čvorove (sa korijenom kao glavnim čvorom), grane i listove. Svaki čvor klasifikacijskog stabla odlučivanja sadrži test koji se primjenjuje na atributu uzorka (npr. uzorak automobil sa atributom snaga motora), svaka grana predstavlja ishod testa, a svaki list predstavlja oznaku klase. Dubina stabla je broj ukupan broj čvorova od korijena do najdaljeg lista. Put od korijena do lista predstavljaju klasifikacijska pravila.



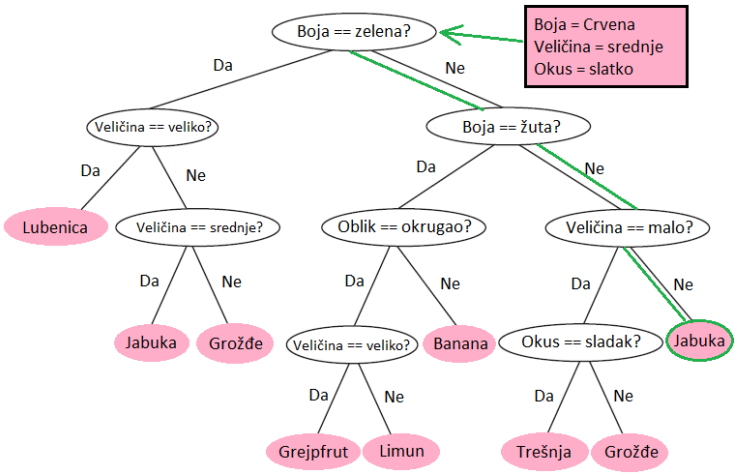
Slika 2: Primjer jednostavnog binarnog stabla odluke za potrošnju automobila.

Stablo odlučivanja u općem obliku može imati proizvoljan broj grana koje izlaze iz svakog čvora neovisno jedan o drugom. Stablo koje se najčešće koristi je sa dvije grane koje izlaze iz svakog čvora. Povećanje broja grana stabla u većini

slučajeva ne donosi zamjetna poboljšanja performansi s obzirom na kompleksnost programske implementacije. Stablo u tom obliku se zove balansirano binarno stablo i ono se jako često koristi u mnogim računalnim primjenama kao što je recimo skladištenje podataka jer daje dobar kompromis brzine traženja, ubacivanja i brisanja podataka (sve ima prosječnu složenost O($log_2(n)$)). Zbog tih karakteristika ono se često koristi i u raznim područjima umjetne inteligencije što uključuje i računalni vid.

2.2. Način rada

Stablo odlučivanja radi na način da se na njegov vršni čvor (korijen) dovedu podaci koji se želi ispitati. U svakom čvoru se vrši test nad podacima i taj test daje binarnu odluku. Ukoliko podatak zadovoljava test on se šalje na lijevu ili desnu granu (proizvoljno se određuje prilikom izrade čvora i testa) do sljedećeg čvora. Svaki čvor može sadržavati isti test samo sa različitim (ili istim) pragom ili posve novi test na nekoj drugoj značajki podataka, recimo test na prvom čvoru može dijeliti značajke po boji piksela, a na drugom po odzivu HOG filtera. Podatak tako ide niz stablo dok ne dođe do listova gdje se nalazi klasa koju stablo daje kao izlaz. Način rada stabla odlučivanja je prikazan na primjeru voća na sljedećoj slici:

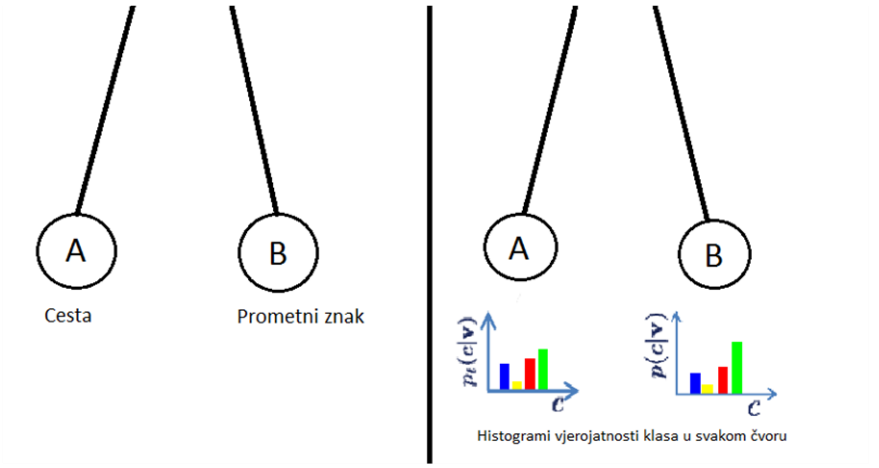


Slika 3: Način rada stabla odluke.

Treba napomenuti da u listovima stabla ne mora nužno biti donesena čvrsta odluka već je moguće da se tamo nalaze iznosi vjerojatnosti za skupove kojima može pripadati testirani skup. Takav oblik listova sadrži iznose uvjetnih vjerojatnosti *p(c|v)* , gdje je *c* vjerojatnost događanja uz uvjet da se dogodio *v*, koji se mogu prikazati preko histograma i za koje vrijedi ovaj izraz:

$$\sum_{i}^{n} p(c|v)_i = 1 \tag{2.1}$$

Taj oblik se najčešće nalazi u stablima koja se koriste za semantičku segmentaciju i on se također nalazi u programskoj implementaciji ovog rada. Razlika između listova sa čvrstom odlukom i histogramom uvjetnih vjerojatnosti prikazana je na sljedećoj slici:



Slika 4: Primjer listova sa čvrstom odlukom i histogramom vjerojatnosti. Histogrami vjerojatnosti za razliku od čvrstih odluka u sebi sadrže iznose koji određuju kolika je vjerojatnost da podatak koji je došao do tog lista pripada određenim klasama.

2.3. Mjere kvalitete podjele u pojedinačnom čvoru

Vjerojatno najbitniji dio svakog stabla odlučivanja je način na koji se značajke binarno klasificiraju po mjeri kvalitete podjele u svakom čvoru. To je ključan dio svakog postupka izgradnje stabla i on određuje koliko će kvalitetno stablo klasificirati objekte. Cilj testa podjele u svakom čvoru je da nađe najboljeg kandidata po kojem bi se klasificirali preostali podaci. Podaci koji se šalju u stablo obično u obliku višedimenzionalnog vektora test funkcija može biti izvršena po bilo kojoj dimenziji (značajki) tog vektora te je stoga ogranak odluke određen upravo po kojoj dimenziji se radi podjela u čvoru.

2.3.1. Gini nečistoća

Gini nečistoća je mjera koliko često bi nasumično odabrana značajka iz nekog skupa bila krivo klasificirana ako bi ju se nasumično klasificiralo s obzirom na to kakva je razdioba značajka po razredima u nekom podskupu od svih značajki koje se koriste u testu. Gini nečistoća se računa tako da se zbroje vjerojatnosti odabira svakog elementa pomnožene sa vjerojatnošću krive klasifikacije tog elementa. Ona doseže minimum (nula) kada svi elementi u čvoru spadaju u jedan razred. Gini nečistoća se računa preko sljedećeg izraza [6]:

$$I(t) = 1 - \sum_i p_i \cdot (1 - p_i) \tag{2.2}$$

Ovdje je ***i = 1, 2, 3,..., r*** broj razreda, ***$p_i(t)$*** su vjerojatnosti klasifikacije značajke u čvoru, a ***t*** je odabrana značajka. Jasno se vidi iz formule da je u slučaju samo jednog razreda cijeli izraz jednak nuli, a u slučaju jednolike distribucije razreda izraz teži u jedan ako se povećava broj razreda.

2.3.2. Srednji uzajamni sadržaj informacije

Veoma često se koristi za generiranje stabla i za korištenje kao test funkcija u čvorovima. Koristi se u algoritmima za generiranje stabla (ID3[2], C4.5[3] i C5.0[4]). Srednji uzajamni sadržaj informacije (transinformacija) je pojam iz teorije informacije i koristi koncept entropije. Entropija je u teoriji informacije definirana kao količina informacije koju nosi neka poruka i računa se preko sljedećeg izraza [6]:

$$H(X) = -\sum_{i=1}^{n} p(x_i) * log_2 p(x_i) \tag{2.3}$$

[2] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106
[3] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993
[4] http://rulequest.com/see5-info.html (algoritam je komercijalan)

U ovom slučaju ***i = 1, 2, 3,..., n*** je broj razreda, ***X*** je skup svih razreda, a ***p(xᵢ)*** je razdioba elemenata. Nakon što je definiran pojam entropije sada se može dati formula za mjeru uzajamne informacije, a ona je zadana preko ove definicije:

$$I(X;Y) = H(X) - H(X|Y) \tag{2.4}$$

Iz ove definicije se vidi da je mjera uzajamne informacije očekivana promjena u entropiji informacije ako iz početnog stanja prijeđe u trenutno stanje. Ovo je jako korisna mjera koja za većinu podataka može dati dobru razdiobu podataka za daljnju izgradnju stabla. Način na koji se to radi je da se prvo izračuna entropija svih podataka u čvoru te se zatim izračuna srednja vrijednost entropije za svaku značajku koju posjeduje podatak, npr. ako su podaci ljudi tada se računa entropija ako se oni prvo podijele po visini, pa po težini, pa po boji kose, itd. Nakon toga se računa mjera uzajmne informacije za svaku od tih podjela te ona koja daje najveći dobitak informacije se uzima kao ona po kojoj će se podaci djeliti u tom čvoru.

2.3.3. Smanjenje varijance

Ova metoda se najčešće koristi ako su varijable kontinuirane i stablo regresijsko jer bi ostale tehnike zahtjevale diskretizaciju značajki prije nego bi mogle upotrijebiti. Smanjenje varijance čvora ***N*** se definira kao konačno smanjenje varijance od tražene varijable ***x*** prema funkciji koja radi podjelu značajki u tom čvoru prema sljedećem izrazu [6]:

$$I_V(N) = \frac{1}{|S|}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right) \tag{2.5}$$

Ovdje su ***S***,***Sₜ*** i ***S_f*** set već podjeljenih indeksa, set odabranih indeksa koji za koje test funkcija daje istinu kao izlaz te set indeksa za koju test funkcija daje laž kao izlaz. Ova metoda nije od prevelikog značaja za ovaj rad jer se u njemu radi sa klasifikacijskim stablima.

2.4. Algoritam za treniranje stabla odluke

Potrebno: *D* skup instanci za treniranje

Potrebno: *F* broj karakteristika (podataka) za generiranje stabla

Potrebno: *P* broj parametara karakteristika

Potrebno: *T* broj kandidata za izbor granice

Potrebno: kriterij zaustavljanja (npr. maksimalna dubina stabla ili količina informacije)

1: **funkcija** GenerirajStablo(*D*)
2: **ako** vrijedi kriterij zaustavljanja **onda vrati**
3: (D_{lijevo};D_{desno}) ← OdrediMjestoNajboljePodjeleSkupa(*D*)
4: GenerirajStablo(D_{lijevo}) ▷ rekurzivno generiraj lijevo dijete
5: GenerirajStablo(D_{desno}) ▷ rekurzivno generiraj desno dijete
6: **funkcija** OdrediMjestoNajboljePodjeleSkupa(*D*)
7: $\mathbf{F} \in \mathbf{R}^{F \times P}$ ← stvori nasumične kandidate za karakteristike
8: $\mathbf{T} \in \mathbf{R}^{F \times T}$ ← stvori nasumične kandidate za mjesto podjele za svaku karakteristiku
9: $I^* \leftarrow \infty$ ▷ inicijaliziraj optimalnu nečistoću
10: **za svaki** $f \in 1..F$ **radi**
11: **za svaki** $\theta \in \mathbf{T}_f$ **radi**
12: $D_{lijevo} \leftarrow \{d \mid d \in D, \text{OdzivKarakteristike}(\mathbf{F}_f, d) < \theta\}$ ▷ lijeva strana
13: $D_{desno} \leftarrow D \setminus D_{lijevo}$ ▷ desna strana
14: I ← OcjenaNečistoće(D,D_{lijevo}, D_{desno})
15: **ako je** $I < I^*$ **onda** ▷ nadogradi najbolje parametre f^*, θ^*
16: $I^* \leftarrow I$; $f^* \leftarrow f$; $\theta^* \leftarrow \theta$
17: $D_{lijevo} \leftarrow \{d \mid d \in D, \text{OdzivKarakteristike}(\mathbf{F}_{f^*}, d) < \theta^*\}$ ▷ lijeva djeca
18: $D_{desno} \leftarrow D \setminus D_{lijevo}$
19: **vrati** (D_{lijevo};D_{desno}) [1]

2.5. Prednosti i nedostaci stabla odlučivanja

2.5.1. Prednosti

1. Lagano ih je razumjeti i shvatiti

2. Zahtjevaju veoma malo obrađivanja podataka prije unosa

3. Mogu se koristiti za numeričke i kategoričke podatke

4. Koriste model „bijele kutije“ (lagano je razumjeti zašto su rezultati koji su dobiveni takvi kakvi jesu)

5. Moguće je napraviti validaciju kvalitete preko statističkih testova

6. Robusnost

7. Dobro rade i sa velikim količinama podataka

2.5.2. Nedostaci

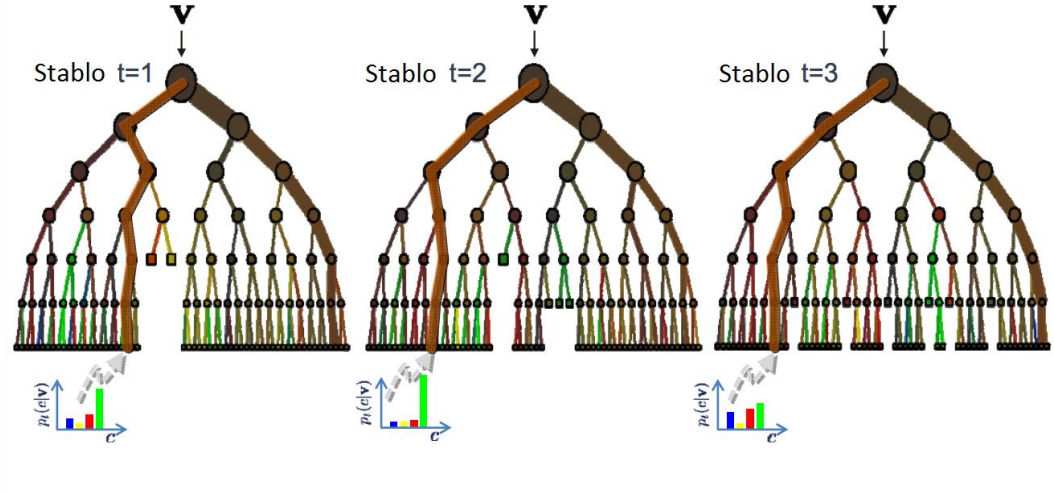
1. Problem dobivanja optimalnog stabla je NP-kompletan (radi toga se najčešće koriste pohlepni algortimi kao npr. već spomenuti ID3).

2. Problem prenaučenosti (eng. overfitting problem) [4]. Neka je dan prostor hipoteza H. Hipoteza $h \in H$ je prenaučena ako postoji hipoteza $h' \in H$ takva da h ima manju pogrešku nego h’ na na primjerima za učenje, ali h’ ima manju pogrešku nego h na cijelom prostoru primjera. Ovaj problem se rješava podrezivanjem stabla (ograničavanje maksimalne dubine).

3. Postoje koncepti za koje je teško naučiti stablo jer ih ono ne izražava na lagan način (npr. XOR, paritetni i multipleksirajući problemi). Ovaj problem se rješava tako da se pokuša promjeniti domena problema tj. pokušava se riješiti problem po nekom drugom parametru.

3. Slučajne šume

Slučajne šume su klasifikator koji se sastoji od kolekcije nezavisnih stabala odlučivanja gdje svako stablo predstavlja jedan glas u većinskom donošenju odluke [3]. Vizualni primjer slučajne šume prikazan je na sljedećoj slici:



Slika 5: Primjer slučajne šume sa 3 stabla. Na slici je vidljivo da svako stablo daje različitu odluku. One se na kraju sve zbrajaju i onaj razred sa najvećom vjerojatnosti se uzima kao krajnja odluka.

Kao što je vidljivo, slučajna šuma je logična nadogradnja na model stabla odlučivanja koja rješava neke njegove nedostatke. Kao što se da zaključiti iz imena klasifikatora bitna stvar prilikom treninga stabla je izvor slučajnosti. Izvori slučajnosti prilikom treninga mogu biti različiti kao recimo nasumičan odabir podataka za treniranje, nasumično odabran podskup od odabranih podataka za treniranje (bilo da su oni odabrani nasumično ili ne) pa čak i nasumičan odabir značajki po kojem će se raditi test funkcija u svakom čvoru. Ti izvori slučajnosti rješavaju problem velike varijance pojedinačnog stabla odlučivanja jer je svako stablo trenirano na drugačijem podskupu podataka. Krajnja odluka se dobiva tako da se uzme histogram odluka svakog stabla i ona odluka sa najvećim brojem ponavljanja se uzima kao konačna.

3.1. Algoritam izrade slučajne šume

1. Uzorkuje se ***T*** nezavisnih podskupova skupa podataka za učenje. Svako nezavisno sampliranje je osnova za učenje jednog stabla odlučivanja [3].

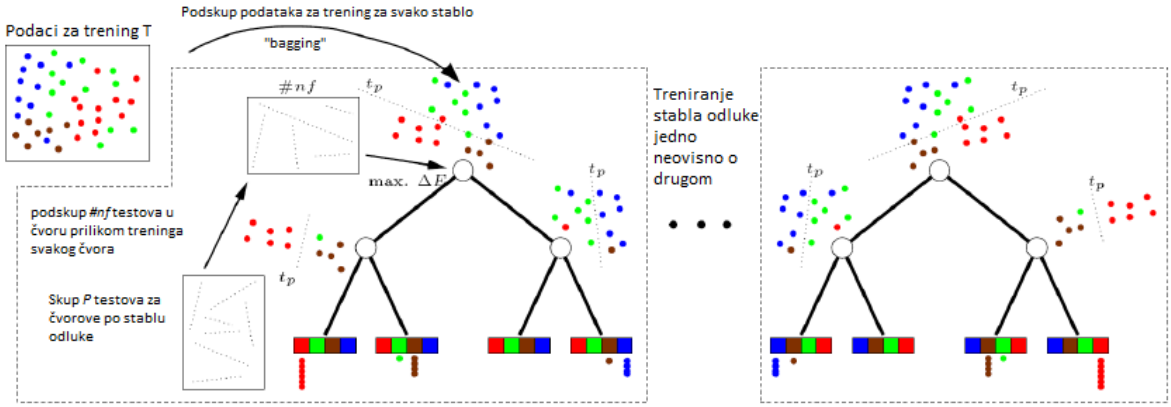
2. Izgradi se maksimalno duboka stabla (bez ograničenja dubine stabla). U izgradnji stabala za svaki čvor slučajno se odabere mali podskup značajki. Unutar toga podskupa odredi se najbolja značajka uobičajenom metodom (entropija ili Gini nečistoća) [3].

3. Podskup stabala testira se na svim preostalim primjerima ili se svako stablo testira sa svojim preostalim primjerima. Pogreška se usrednjava preko svih stabala [3].

4. Za nepoznati primjer X prikupljaju se glasovi svih stabala i određuje većinski razred klasifikacije [3].

Objašnjenja [3]:

- Broj stabala (***T***) se povećava dok se točnost klasifikacije na skupovima za evaluaciju ne ustali.
- Veličina podskupa slučajnih značajki za svaki čvor je $\sqrt{n}$ gdje je n veličina cijelog skupa. Povećanje slučajnog podskupa povećava točnost klasifikacije svakog pojedinog stabla, ali i međusobnu korelaciju između stabala što smanjuje točnost klasifikacije cijele šume.



Slika 6: Trening slučajne šume.

3.2. Prednosti slučajnih šuma

1. Nije potrebna međuvalidacija. Razlog za to je sama nasumičnost odabira podataka.
2. Slučajne šume se ne prilagođuju podacima za učenje ("overfitting") zbog toga jer je svako stablo naučeno na drugom nasumičnom podskupu podataka tako da daje drugačije odluke.
3. Potvrđeno bolja točnost klasifikacije u usporedbi s popularnim algoritmima [3].
4. Efikasna metoda za velike baze podataka i podatke sa nepoznatim vrijednostima.

3.3. Primjer upotrebe slučajnih šuma

Jedan od najpoznatijih i komercijalno najuspješnijih proizvoda koji koristi slučajne šume u realnoj aplikaciji je Microsoftov sustav za detektiranje položaja osobe Kinect. On koristi slučajne šume kao klasifikator kojim detektira dijelove tijela osobe u realnom vremenu za što koristi jednu procesorsku jezgru Xenon CPU-a u Xbox-u 360 (prva generacija Kinect-a). Trening za slučajne šume se izvršio na superračunalu sa preko 1000 procesorskih jezgri. Prva generacija ima frekvenciju osvježavanja od 30 Hz i radi sa VGA rezolucijom (640 × 480 piksela). Najčešće se koristi kao dodatak za igranje u podržanim igrama, ali svoju primjenu je pronašao i u području računalnog vida gdje se koristi radi dubinske kamere kojom je opremljen.

Slika 7: MIcrosoft Kinect.

4. Semantička segmentacija slika

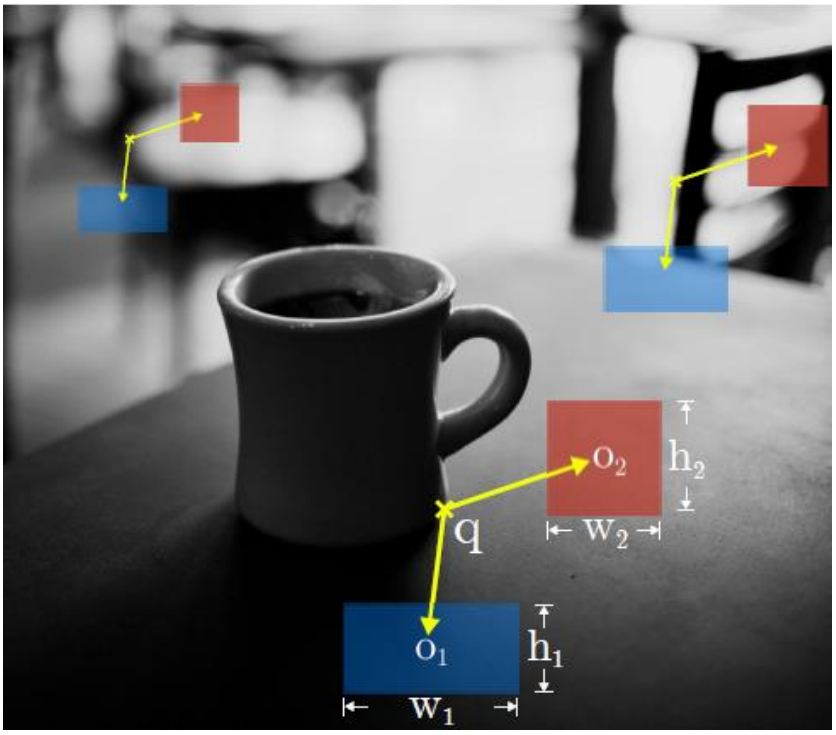
Kao što je već spomenuto u uvodu semantička segmentacija slika se može opisati kao zadatak pronalaženja grupa piksela koji „idu zajedno“. U statistici ovaj problem je poznat kao analiza grupa i to je naširoko proučavano područje sa stotinama različitih algoritama. U računalnom vidu semantička segmentacija slika je jedan od najstarijih i najčešće proučavanih problema. Postoji mnogo različitih pristupa rješavanju tog problema i za opisivanje svakog od njih bi trebalo mnogo više stranica nego što to ima ovaj rad. Stoga će se u ovom radu opisati samo metoda najbližeg susjedstva koja se koristi u njemu.

4.1. Metoda temeljena na klasifikacijskim oknima

Metoda temeljena na klasifikacijskim oknima [1] je jedna od mnogobrojnih metoda za rješavanje problema semantičke segmentacije. Ovdje se koristi ponajviše radi toga jer je jako pogodna za rad sa slučajnim šumama tj. za izvlačenje vizualnih karakteristika iz slike. Ova metoda koristi pomični prozor proizvoljne veličine koji se pomiče po slici piksel po piksel. Za svaki piksel $\boldsymbol{q}$, slikovna značajka $\boldsymbol{f_\theta}$ se najčešće računa kao razlika, iako se može koristiti i suma, produkt ili neka druga kompliciranija operacija, prosječnog iznosa kanala slike $\boldsymbol{\phi_i}$ u dvije pravokutne regije $\boldsymbol{R_1}$, $\boldsymbol{R_2}$ u susjedstvu oko $\boldsymbol{q}$ prema sljedećem izrazu:

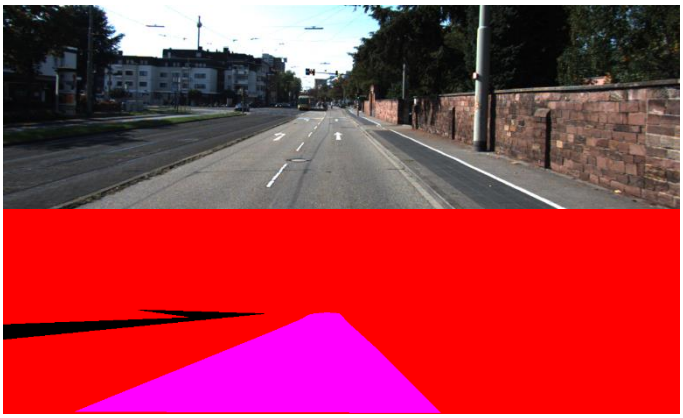
$$f(q|R_1, R_2) \coloneqq \frac{1}{|R_1(q)|}\sum_{p\in R_1}\phi_1(q) - \frac{1}{|R_2(q)|}\sum_{p\in R_2}\phi_2(q) \qquad (4.1)$$

Također, računanje slikovnih karakteristika se može računati i sa proizvoljnim pomakom pravokutnih regija $\boldsymbol{R_1}$ i $\boldsymbol{R_2}$ u odnosu na piksel na kojem se nalazi prozor. To najčešće donosi poboljšanje performansi detekcije u odnosu na računanje samog susjedstva piksela. Vizualizacija rada metoda temeljena na klasifikacijskim oknima je prikazana na sljedećoj slici:



Slika 8: Rad metode temeljene na klasifikacijkim oknima sa tri različite veličine prozora. Ovdje se vidi da okna mogu biti različite veličine, ali i različitog pomaka u odnosu na piksel gdje se računa značajka. Različite dimenzije i pomaci daju različitu preciznost i ona se mora eksperimentalno izmjeriti.

Računanje razlike se može računati na više načina, bilo preko razlike u standardnim kanalima boje (RGB), CIE Lab prostoru boja, dubinskoj karakteristici slike, itd. Dobivene karakteristike se tada asociraju sa klasom objekta koji je predstavljen slikom sa označenim klasama objekata na slici od kojih je svaki predstavljen različitom bojom (eng. „ground truth“ slika) te se šalju za trening slučajne šume. Jedan od načina dobivanja „ground truth“ slike je naveden u poglavlju 6.1. Primjer „ground truth“ slike je prikazan na sljedećoj slici:

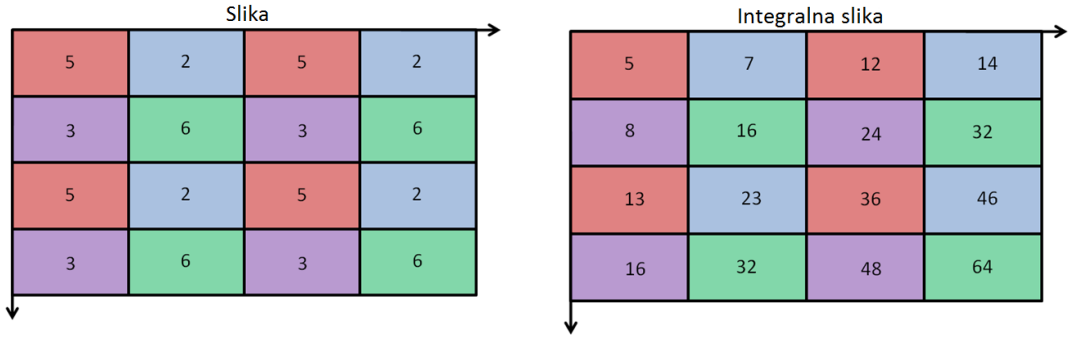


Slika 9: Primjer "ground truth" slike za cestu.

Problem koji se javlja prilikom računanja prosječnog iznosa kanala na slici je taj što je potrebno n × m operacija po pikselu pri čemu je n je visina pravokutnika, a m širina pravokutnika da se dobije rezultat. Pošto je potrebno izračunati prosječne iznose za svaki piksel na slici ovaj način nije daje prihvatljive performanse. Način rješavanja ovog problema su integralne slike.

4.1.1. Integralna slika

Integralna slika je brz i efikasan način računanja suma vrijednosti (iznosa piksela na slikama) na slikama ili pravokutnom podskupu slika [5]. Najčešće se koristi za računanje prosječnog intenziteta piksela na nekom području slike i može se koristiti bilo za slike sa jednim ili više kanala. Način dobivanja integralne slike jest da se ide stupac po stupac piksela na slici te se računa suma svih iznad onog na kojem se trenutno nalazimo uključujući i njega samog te se na to pribroji i vrijednost već izračunatog sa lijeve strane onog na kojem se nalazimo. Kad se izračuna cijela integralna slika računanje prosječnog iznosa kanala na bilo kojem pravokutniku na slici postaje složenosti O(1). Primjer integralne slike je prikazan na sljedećoj slici:

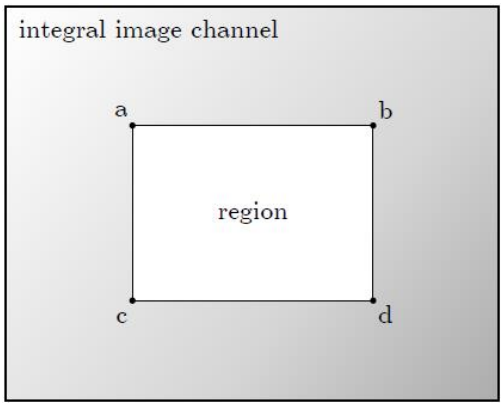


Slika 10: Primjer slike i njezine integralne slike.

Računanje prosječnog iznosa kanala na pravokutniku na integralnoj slici se tada izvodi preko izraza (4.2) ako su ***a***, ***b***, ***c*** i ***d*** na pozicijama kao na slici 10:

$$S = a + d - b - c \tag{4.2}$$

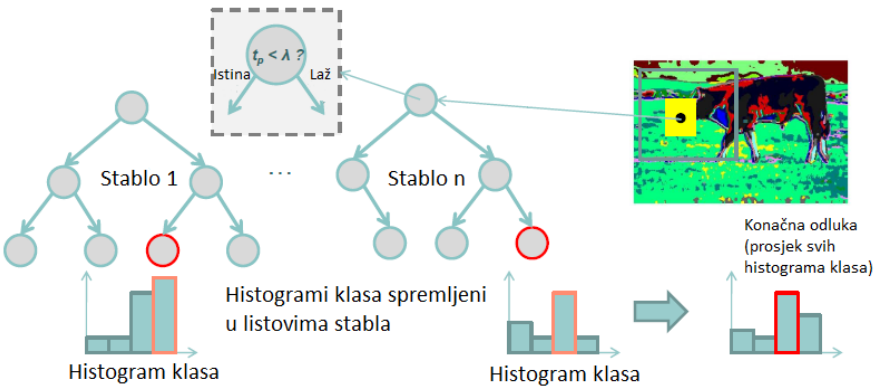
Kao što je vidljivo iz formule, umjesto n · m operacija po pikselu sada se računanje izvršava sa ukupno četiri pristupa memoriji [1].



Slika 11: Računanje prosječnog iznosa kanala na pravokutniku na integralnoj slici.

4.2. Segmentacija slike pomoću slučajne šume

Kao što je već prije nevedeno segmentaciju slika je moguće provesti preko klasifikatora slučajne šume. Navedeni postupak je efikasan za treniranje, samu segmentaciju i daje dobre rezultate te je stoga dosta popularan. Segmentacija slike pomoću slučajne šume koja se koristi u ovom radu koristi sve metode navedene u prijašnjim poglavljima. Prvo što treba odrediti je broj stabala odlučivanja koji će biti u slučajnoj šumi te naći skup slika za trening stabala. Nakon treninga svakog pojedinačnog stabla na zasebnom podskupu slika može se krenuti sa postupkom segmentacije slike. Za svaku sliku koja se semantički segmentira prvo se naprave njezine integralne slike po svakom kanalu te se preko metode najbližeg susjedstva računa odziv svakog piksela. Izračunati odziv tada se šalje na evaluaciju u svako stablo odlučivanja te se računa prosjek histograma rezultata svih stabala. Klasa koja ima najveću vrijednost u histogramu se uzima kao rezultat i dodjeljuje se pikselu koji se evaluira. Cijeli postupak je prikazan na sljedećoj slici [2]:



Slika 12: Semantička segmentacije preko slučajne šume. Za svaki piksel se računa značajka u oknu te se ta značajka šalje u slučajnu šumu. U rezultatu, razred sa najvećom vjerojatnosti je krajnja odluka.

5. Programska implementacija

U ovom poglavlju se nalaze programske implementacije postupaka navedenih u prethodnim poglavljima zajedno sa programskom podrškom u kojoj se izvela implementacija.

5.1. Programska podrška

C++ je programski jezik opće namjene. Podržava programiranje u proceduralnoj, objektnoj, funkcijskoj i generičkoj paradigmi. On je nadskup programskog jezika C sa kojim ima gotovo 100% kompatibilnost. Popularan je radi svoje brzine, fleksibilnosti i sintakse koju dijeli sa jezikom C te je siguran odabir za svaku namjenu gdje je brzina izvođenja ključna. Glavni nedostatak mu je kompliciranost.

OpenCV[5] (*Open Source Computer Vision Library)* je biblioteka programskih funkcija primarno namjenjena za računalni vid koji se izvodi u realnom vremenu. Razvio ju je Intel i besplatna je za korištenje pod open source BSD[6] licencom. Biblioteka je multiplatformska sa podrškom za sve najpopularnije jezike (npr. C, C++, Java i Python). Ima široku primjenu i kvalitetnu dokumentaciju radi velike zajednice koja radi na njoj.

Visual Studio 2013 (Community Edition)[7] je programsko okruženje koje je razvio Microsoft koje je u verziji Community besplatno. Ima podršku za velik broj jezika (npr. C++, C, C#, Visual Basic, Python, itd.). Glavne karakteristike su mu stabilnost, mogućnost nadogradnje raznim dodacima, kvalitetan sustav za otklanjanje greški, hrpa dodatnih alata ugrađenih u njega te pomoć pri pisanju koda.

[5] http://opencv.org
[6] http://opensource.org/licenses/BSD-3-Clause
[7] https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx

5.2. Programska implementacija

Ovdje su navedene programske implementacije najbitnijih stvari koje su opisane u prijašnjim poglavljima. Svi navedeni kodovi su izvorno napisani u programskom jeziku C++. U ovom poglavlju sve funkcije su prikazane kao pseudokod jer bi prikazivanje cijelog koda bilo preopširno. Sav izvorni kod se nalazi na optičkom mediju priloženim uz rad.

5.2.1. Metoda *Dodaj* u razredu *Stablo*

```
void Stablo::Dodaj(cvor **cv, vector<Piksel> pikseli, int dubina) {

        stvori histograme;

        sadrzajInformacije = Split(pikseli, histogrami, prag, brojRazreda);

        stvori novi cvor;

        ako (sadrzajInformacije < Prag && dubinaStabla < PragDubine) {
                podijeli piksele po pragu za lijevu i desnu granu;
                dubina++;
                Dodaj(lijevo, lijeviPikseli, dubina);
                Dodaj(desno, desniPikseli, dubina);
        }
        inače {
                trenutniČvor je list;
        }
};
```

Metoda *Dodaj* je najbitnija metoda cijelog razreda Stablo. Ona rekurzivno gradi i trenira stablo odluke po algoritmu koji je naveden u poglavlju 2. Kao parametre prima pokazivač na strukturu *cvor* koja je definirana u razredu Stablo, vektor sa razredom Piksel koji sadrži koordinatu piksela, njegovu značajku i razred objekata kojoj pripada te cjelobrojni broj koji govori na kojoj se dubini rekurzije nalazi trenutni poziv. Kada taj broj prijeđe maksimalnu dopuštenu dubinu (poslanu kao argument) daljnja izgradnja stabla staje. Najbitnija funkcija koju koristi metoda *Dodaj* je Split koja traži odziv po kojem se najbolje razdjeljuju pikseli na lijevu i desnu stranu gdje bi u lijevoj trebalo biti čim više piksela razreda *cesta*, a desno piksela razreda *ostalo*.

5.2.2. Funkcija *Split*

```
double Split(vector<Piksel> Pikseli, int &Prag, vector<int> &Histogram, vector<int>
&HistogramLijevi, vector<int> &HistogramDesni, const int brojKlasa) {

        ispremjesajPiksele;

        velicina = sqrt(BrojPiksela);

        izracunaj entropiju pisksela prema razredima;

        za svaki piksel od 0 do velicina {

                prag = značajka Piksela;

                podijeli preostale piksele prema toj značajki;

                izracunaj entropiju na lijevoj i desnoj strani praga;

                izracunaj kolicinuInformacije;

                ako (kolicinaInfomacija > Maksimum) {

                        Prag = prag;

                        Maksimum = kolicinaInformacije;

                }

        }

        Vrati Maksimum;

}
```

Funkcija kao argumente prihvaća vektor sa pikselima, referencu na cijelobrojni broj u koji se sprema iznos praga, tri reference na vektore sa cjelobrojnim brojevima u koje se spremaju lijevi i desni histogram, histogram koji ostaje u čvoru gdje je pozvana funkcija te broj razreda. Zadnji argument je broj mogućih razreda na slikama. Funkcija prvo unosi nasumičnost tako da premiješa vektor sa pikselima te onda uzme prvih n piksela, gdje je n jednak korijenu od ukupnog broja piksela zaokružen na cijeli broj. Funkcija tada računa za svaki od piksela srednji uzajamni sadržaj informacije te značajku piksela sa najvećim sadržajem informacije vraća preko reference. Također funkcija vraća i srednji uzajamni sadržaj informacije jer to treba za provjeru u metodi *Dodaj* da se vidi da li je se stablo nastavlja dalje graditi ili se staje.

5.2.3. Funkcija *TreningStabla*

```
list<Stablo> TreningStabla(const int BrojStabla, const int BrojPikselaPoSlici, const int
VisinaProzora, const int SirinaProzora, vector<string> Slike, int Dubina) {

        stvori vektor piksela;
        za svaku sliku poslanu u vektoru Slike {
                otvori datoteku sa slikom;
                otvori datoteku sa oznakama razreda;
                izračunaj integralne slike;
                od 0 do brojaPikselaPoSlici {
                        uzmi nasumični piksel i njegovu okolinu na slici;
                        prepoznaj klasu kojoj pripada preko slike sa oznakama razreda;
                        izračunaj značajku piksela preko integralnih slika;
                        spremi piksel u vektor piksela;
                }
        }
        stvori praznu listu stabala odlučivanja;
        na svakoj procesorskoj jezgri paralelno treniraj stabla;
        vrati listu stabla;
}
```

Funkcija *TreningStabla* kao argumente prima broj stabla za trening, broj piksela koji se uzima po slici, visinu i širinu prozora te vektor sa imenima slika. Glavna zadaća funkcije je uzeti nasumične piksele sa svake od slika. Za svaku sliku poslanu u vektoru se prvo izračuna integralna slika po svakom kanalu boje te se za svaki piksel izračuna značajka i gleda se da li je taj piksel označen kao dio ceste. Nakon toga kreće trening stabla. Broj stabla je određen brojem proslijeđenim u argumentu. *TreningStabla* je paralelizirana i može iskoristiti svaku jezgru u računalu. Paralelizam je izveden preko std::thread klase koja je standard jezika C++ od verzije C++11 i radi na svakom operacijskom sustavu.

5.2.4. Funkcija *SemantičkaSegmentacija*

```
vector<int> SemantickaSegmentacija(Mat &Slika, string imeSlike, list<Stablo> &Stabla,
const int VisinaProzora, const int SirinaProzora, const int BrojRazreda) {

        otvori sliku naziva imeSlike;
        izracunaj integralne slike;
        otvori sliku sa oznakama razreda;
        stvori privremenu sliku;
        za svaki piksel slike {
                izracunaj značajku piksela preko integralnih slika;
                posalji dobivenu značajku piksela na evaluaciju u istreniranu slucajnu
                sumu
                poslanu preko argumenta;
                prema rezultatu evaluacije odredi razred piksela na slici;
                tocnost evaluacije u odnosu na sliku sa oznakama razreda spremi u
                povratni vektor;
        }
        Izracunaj integralnu sliku privremene slike;
        Pomici prozor velicine 10x10 piksela po slici {
                ako je u prozoru preko 75% piksela razreda cesta {
                        oznaci te piksele na originalnoj slici kao cestu;
                }
                inace {
                        ostavi izvorne vrijednosti piksela na slici;
                }
        }
        vrati vektor rezultata tocnosti evaluacije;
```

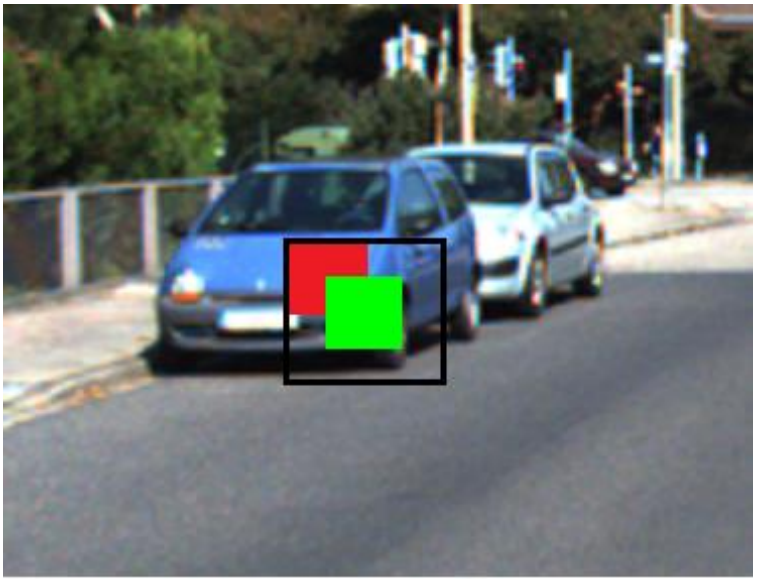
SemantičkaSegmentacija je funkcija koja koristi slučajnu šumu za detekciju karakteristika na slici, u ovom slučaju ceste. Argumenti funkcije su matrica sa slikom na koju se označavaju pikseli koji pripadaju cesti, ime slike koju spremamo u tu matricu, lista stabala tj. slučajna šuma, visina i širina prozora te broj različitih razreda na slikama. Funkcija učitava sliku i za svaki piksel provjerava značajku na isti način kao u treningu šume. Nakon toga šalje tu značajku u svako stablo slučajne šume te se donosi odluka od svakog stabla i na kraju krajnja odluka ovisno o vjerojatnostima iskazanim u listu svakog stabla. Svaki piksel ceste se sprema u privremenu sliku nad kojom se kasnije vrši jednostavan filter. Taj filter je prozor veličine 10 × 10

piksela koji se miče po slici i ako je 75% piksela u njemu klase cesta onda ih ostavlja. Inače se to računa kao šum i ne prikazuje se na krajnjoj slici. Funkcija na kraju vraća povratne vrijednosti koliko piksela ima slika, koliko je točno i krivo detektiranih piksela te koliko ima piksela ceste na „ground truth“ slici.

6. Eksperimentalni rezultati

Eksperimentalna evaluacija je provedena na setu „KITTI road“ koji se može dohvatiti na web stranici Karlsruhe Institute of Tehnologya: http://www.cvlibs.net/datasets/kitti/eval_road.php . Skup slika korišten za trening stabla i evaluaciju nalazi se u datoteci „training“ gdje su slike raznih vrsta cesta na različitim lokacijama te odvojeni skup slika gdje su lokacije cesta na slikama označene u rasterskom („ground truth“) obliku. Broj slika koji je korišten u evaluaciji je reduciran sa 289 na 200 radi lakše podjele u skupove sa jednakim brojem slika. Skup slika je prilično zahtjevan za segmentaciju jer se na slikama traži cesta, a slike su pune raznih asfaltiranih površina koje sliče na cestu poput pločnika, zgrada, zidova itd.

Nešto što ima veoma velik utjecaj na performanse detekcije je način dobivanja odziva piksela. On se u ovom radu dobiva preko prozora proizvoljne veličine na način prikazan na sljedećoj slici:



Slika 13: Način dobivanja odziva piksela. Uzimaju prosječne vrijednosti piksela na zelenom i crvenom kanalu slike unutar okna. Kvadrati se preklapaju radi boljih rezultata.

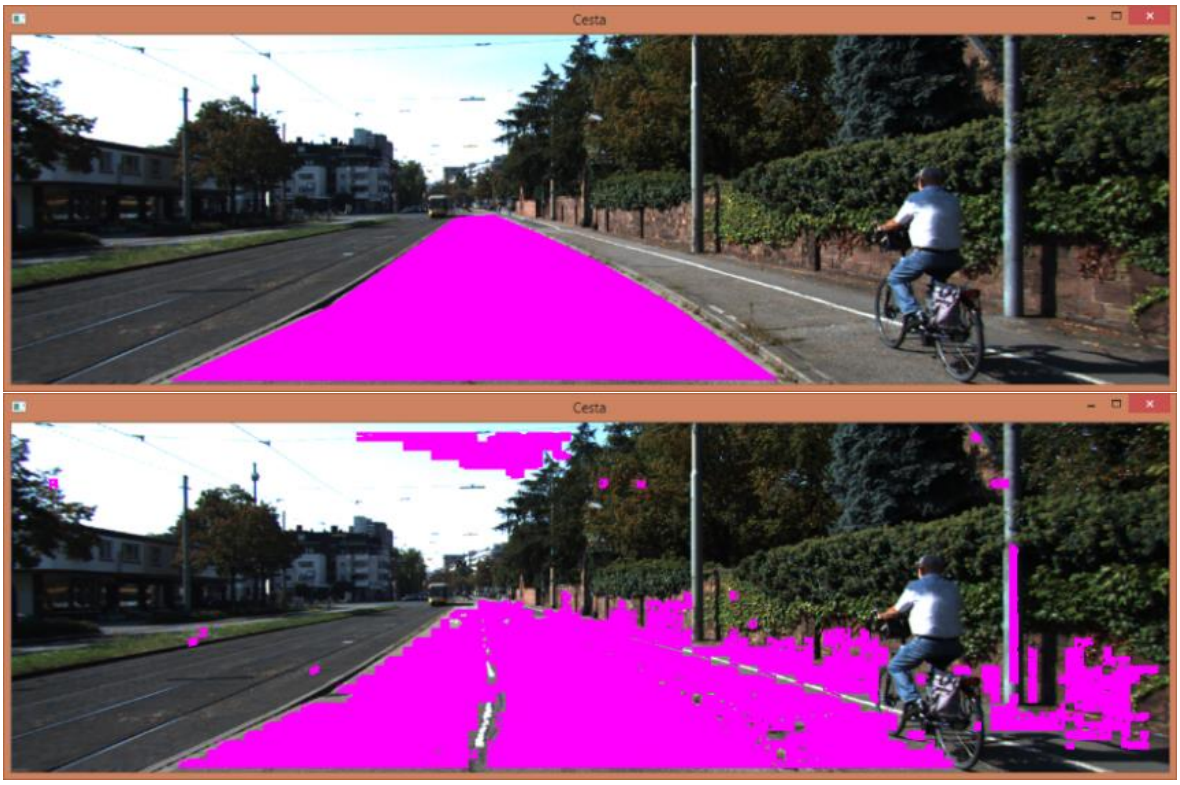
Ono što se vidi iz slike je da se uzimaju vrijednosti iz dva kanala boje slike (crveni i zeleni) na način da se uzme pravokutnik veličine *visinaProzora/2* * *širinaProzora/2* u oba slučaja, ali da ih se rasporedi na različita mjesta i da imaju

djelomično preklapanje. Pravokutnik za crveni kanal se nalazi gornjem kutu prozora, a za zeleni kanal u sredini prozora. Odziv se dalje dobiva preko izraza (6.1):

$$Odziv = |VrijednostR - VrijednostG| \quad (6.1)$$

Ovdje su *VrijednostR* odziv sa crvenog kanala slike, a *VrijednostG* je odziv zelenog kanala.

Primjer semantičke segmentacije dobivene na taj način zajedno sa točnim rezultatom je prikazan na sljedećoj slici:

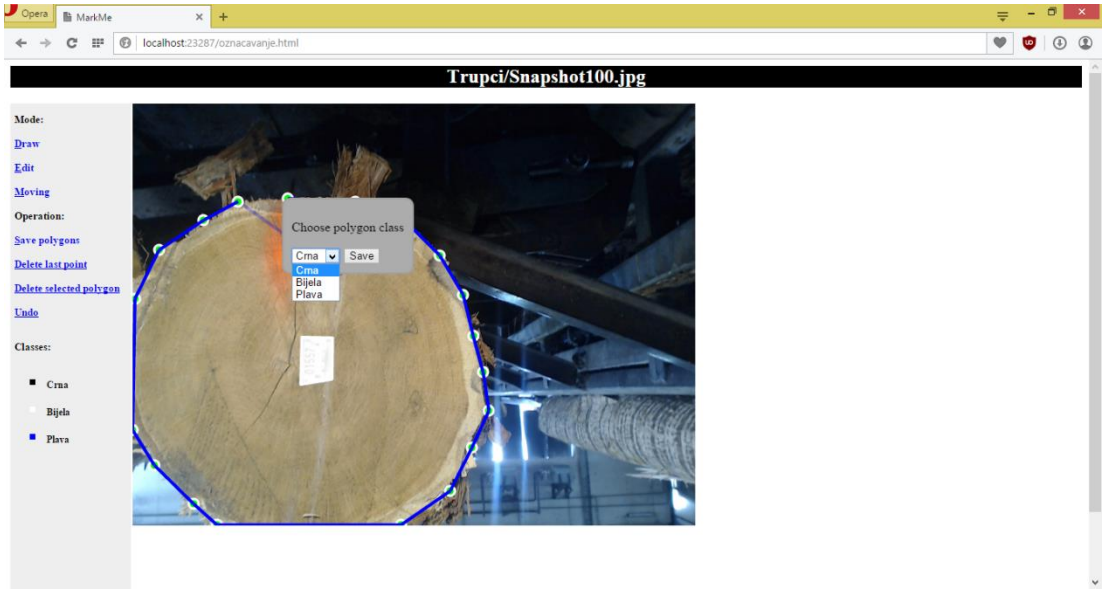


Slika 14: Točna segmentacija (gore) i segmentacija dobivena preko algoritama navedenih u radu (dolje)

Mjera točnosti koju ćemo mjeriti u ovom radu je pikselna i klasna preciznost. Pikselna preciznost je definirana kao broj svih točno klasificiranih piksela u odnosu broj svih piksela na slici. Klasna preciznost je definirana kao postotak točno klasificiranih piksela za svaku klasu. Ona se računa kao dijagonalni prosjek u matrici zbunjenosti (eng. confusion matrix) koja je prije toga normalizirana.

6.1. Program za označavanje slika

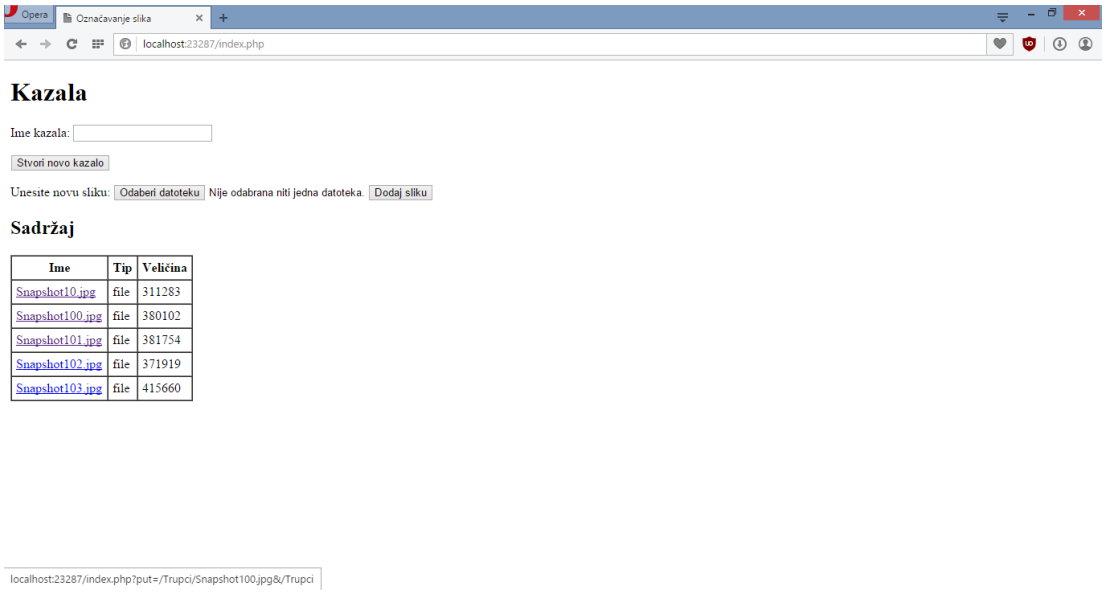
Jedan od problema koji se javlja u procesu segmentacije slika jest stvaranje slika sa označenim razredima (eng. „ground truth“ slika) što je često mukotrpan posao. Stoga je u sklopu ovog rada razvijen prototip programa za ubrzavanje postupka označavanja slika. Program je baziran na kodu postojećeg programa DrawMe[8], ali sa jako proširenim setom mogućnosti. Program je pisan u JavaScriptu, HTML-u 5 i PHP-u bez ikakvih dodatnih biblioteka što mu omogućuje maksimalnu kompatibilnost sa svim internetskim preglednicima i operacijskim sustavima.



Slika 15: Primjer rada na slici

Program je primarno namjenjen za rad na serveru i način rada je taj da netko tko želi raditi na označavanju slika se spoji preko browsera na server i krene sa radom. Lista razreda se dodaje na serveru tako da korisnici ne mogu raditi neželjene promjene na njoj. Program podržava međusobno isključivanje korisnika da se ne dogodi da dva ili više korisnika rade na istoj slici u isto vrijeme. Također omogućuje rad sa kazalom što uključuje i dodavanje novih slika na server. Primjer toga je prikazan na slici 16:

[8] http://vision.princeton.edu/pvt/DrawMe/



Slika 16: Rad sa kazalom

Program omogućuje jednostavno označavanje dijelova slike u obliku proizvoljnog poligona ili pravokutnika i dodavanje razreda tom dijelu slike. Prilikom otvaranja slika svaki prije označeni poligon ako postoji se automatski učitava. Program također omogućuje i brisanje svakog poligona ili pravokutnika, brisanje i proizvoljno pomicanje pojedinačne točke po poligonu te undo funkciju. Ima i mogućnost povećanja i pomicanja slike mišom, ali one još nisu dovršene do kraja i nalaze se u testnoj fazi. Označene regije slike se spremaju u vektorskom obliku i nalaze se na serveru gdje se nalaze i slike.

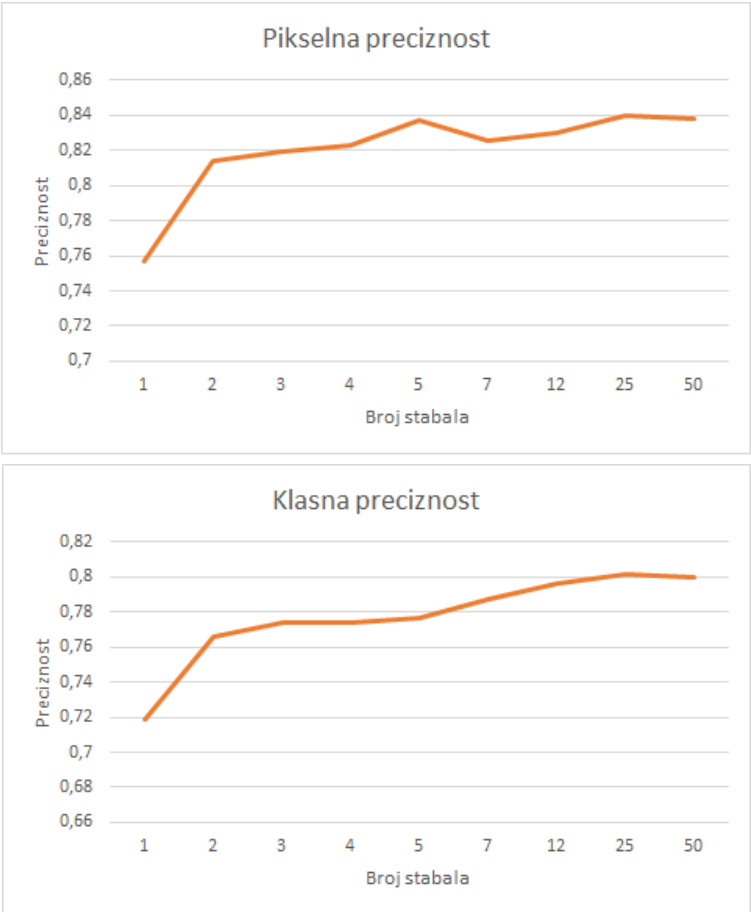
Iako se ovaj program nije koristio u ovom radu on se spominje zbog toga jer će se koristiti u budućnosti za buduća istraživanja u području segmentacije slika.

6.2. Eksperimentalna evaluacija utjecaja broja stabla na kvalitetu semantičke segmentacije

Prvi skup eksperimenata koji je proveden imao je za cilj odrediti kako broj stabala utječe na kvalitetu semantičke segmentacije tj. na broj točno detektiranih piksela koji pripadaju cesti te broj krivo detektiranih piksela koji pripadaju nekom drugom objektu na cesti, ali su detektirani kao cesta. Test je proveden na 200 slika podjeljenih u 5 podskupova. Svaki skup se sastoji od 40 različitih slika. U testu je korištena metoda unakrsne validacije što znači da se tijekom testa 4/5 skupa koristi za trening stabala, a evaluacija se vrši na preostalom skupu od 40 slika. To se

ponavlja još 4 puta tako da se svaki put koriste druga 4 skupa za treniranje te jedan preostali za evaluaciju. Svaki eksperiment je također proveden 5 puta te je napravljen prosjek rezultata. Razlog za to je usrednjavanje rezultata zbog samog načina treninga slučajne šume koji uključuje uzimanje nasumičnih piksela u slici za trening te uzimanje njihovog nasumičnog podskupa za funkciju „split“. Sve to daje poprilično dozu nasumičnosti pa je stoga bilo potrebno raditi više testova radi dobivanja vjerodostojnijih razultata.

Za očekivati je bilo da će povećanje broja stabala donijeti sve bolje i bolje rezultate sa nekom granicom kada će daljnje povećanje donijeti jako malo poboljšanje. Rezultati evaluacije su prikazani na grafovima:

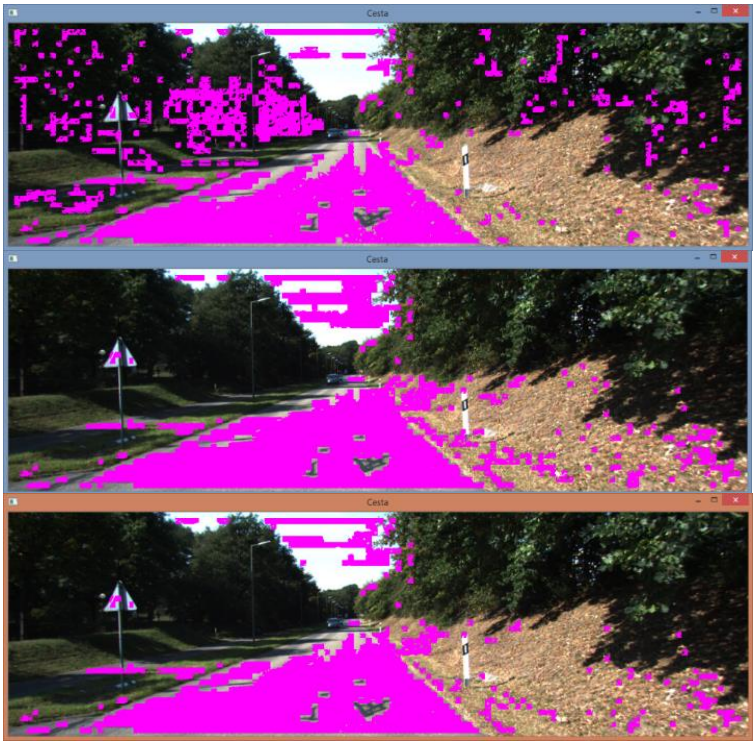


Slika 17: Ovisnost pikselne i klasne preciznosti o broju stabla

Evaluacija je izvršena sa dubinom stabla 10 te širinom i visinom prozora od 10 piksela.

Kao što se može vidjeti rezultati se slažu sa očekivanjima što znači da veći broj stabala znači i bolju detekciju. Rezultati također pokazuju da se zadovoljavajući

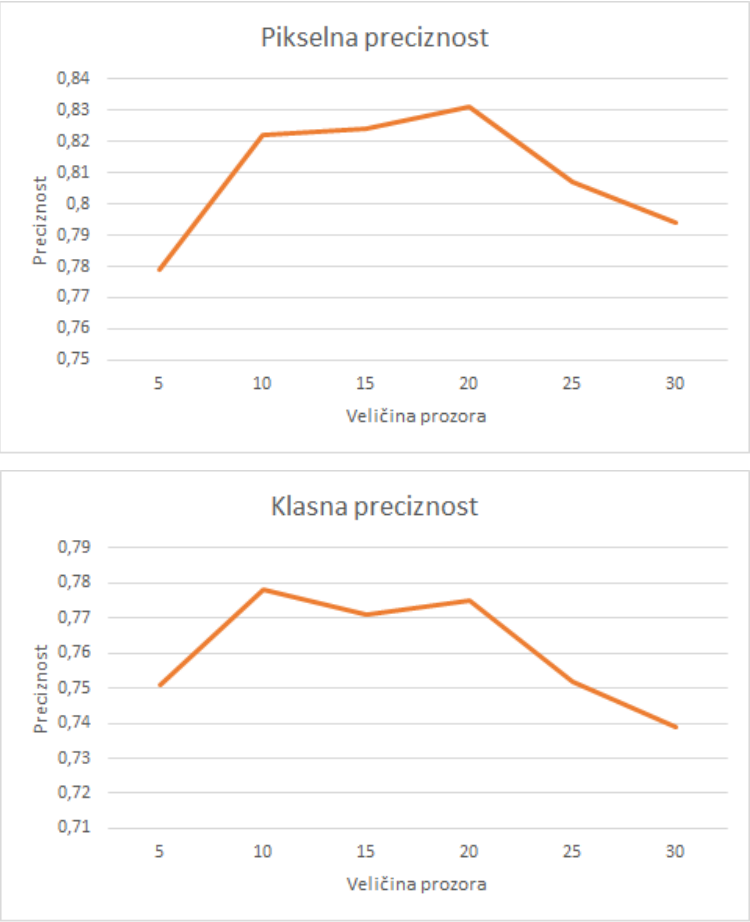
rezultati dobivaju već sa 3 stabla, a da nakon 25 više nema nikakve zamjetne razlike. Rezultati segmentacije sa različitim brojem stabala su prikazani na sljedećoj slici:



Slika 18: Primjeri segmentacije sa 1, 5 i 25 stabala

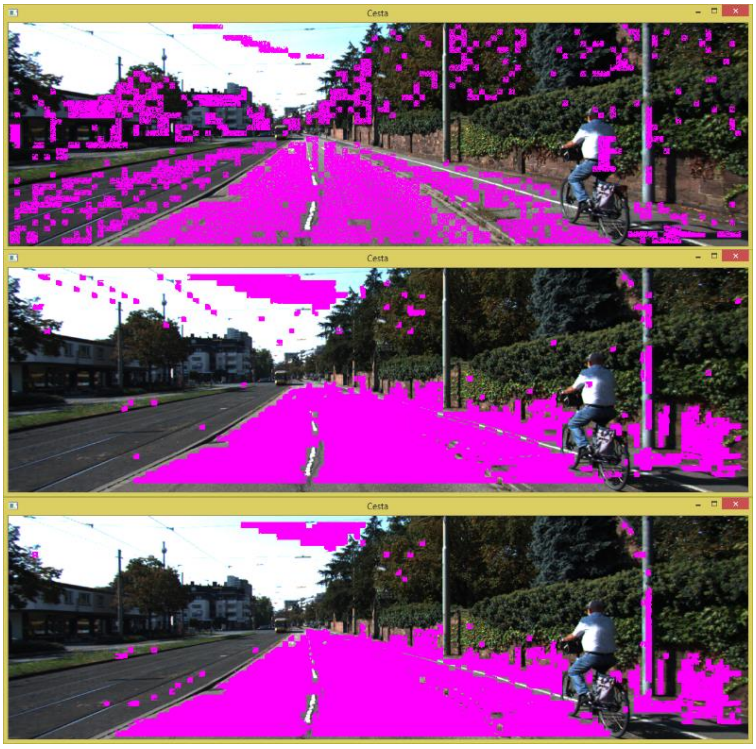
6.3. Eksperimentalna evaluacija utjecaja veličine prozora na kvalitetu semantičke segmentacije

Način testiranja je isti kao i u prošlom slučaju samo što se ovdje mijenja veličina prozora umjesto broja stabla. Broj stabla se drži konstantnim i ovdje iznosi 5, jer kao što je viđeno od ranije već su 3 stabla dovoljna za dobivanje dobrih rezultata. Dubina stabla također je konstanta i iznosi 10 kao i u prošlom primjeru. Rezultati evaluacije su prikazani na grafovima:



Slika 19: Ovisnosti pikselne i klasne preciznosti o veličini prozora

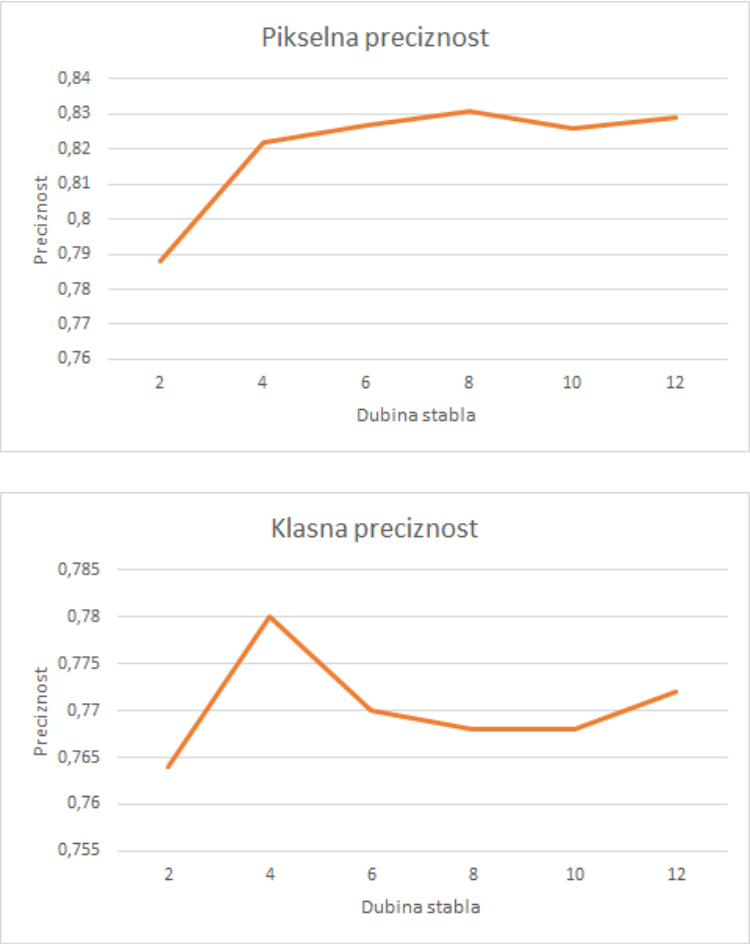
Rezultati pokazuju rast preciznosti segmentacije sa povećanjem veličine prozora, ali do neke granice, koja je u ovom slučaju oko 20 piksela visine i širine. Nakon toga preciznost počinje padati. Razlog za to je vjerojatno taj što se sa povećanjem okoline piksela može razaznati više informacija o okolini i donijeti bolji zaključak o tome kojoj klasi pripada piksel, ali do neke granice. Nakon što se ona prođe okolina postane prevelika i postaje ponovno sve sličnija drugima kao i u slučaju malog prozora. Rezultati segmentacije za različite veličine prozora prikazani su na sljedećoj slici:



Slika 20: Primjeri segmentacije sa veličinom prozora od 5x5, 10x10 i 20x20

6.4. Eksperimentalna evaluacija utjecaja dubine stabla na kvalitetu semantičke segmentacije

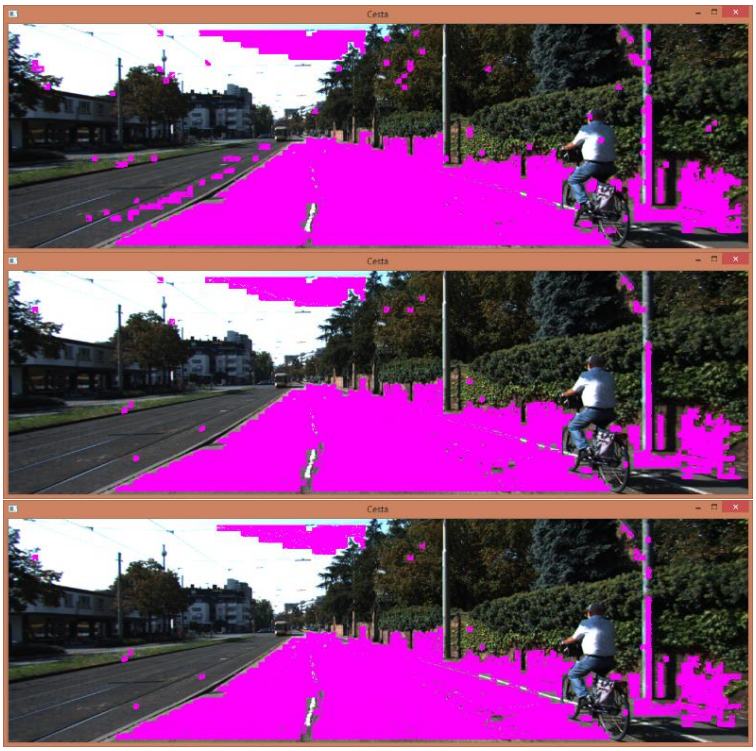
Način testiranja je isti kao i za prethodne dvije evaluacije. U ovom slučaju broj stabla se drži konstantnim i ima ih 10. Također se i veličina prozora drži konstantnom i veličine je 10 × 10 piksela. U ovoj evaluaciji se može očekivati da će povećanje dubine povećanje dubine ponešto povećati i preciznost detekcije, ali ne puno. Razlog za to je taj što imamo samo dvije klase: cesta i ostalo pa bi i plitko stablo trebalo dati dobre rezultate. Rezultati evaluacije su prikazani na grafovima:



Slika 21: Ovisnost pikselne i klasne preciznosti o dubini stabala odlučivanja

Kao što je iz rezultata vidljivo dubina stabala ne utječe previše na performanse segmentacije, ali se može primjetiti rast performansi sve do dubine od 8 čvorova. Nakon stoga performanse su manje-više konstantne. Najveći razlog za to je što postoje samo dvije klase jer bi u slučaju sa više njih sigurno bilo većih razlika.

Rezultati segmentacije za različite dubine stabala su prikazani na sljedećoj slici:



Slika 22: Primjeri segmentacije sa dubinama stabla od 4, 8 i 12 čvorova

6.5. Analiza rezultata

Iz dobivenih rezultata se može vidjeti da preciznost segmentacije ponajviše ovisi o veličini prozora i broju stabala. Dobri rezultati se mogu dobiti već sa 5 ili 10 stabala, a više od 25 nema smisla dodavati u ovom slučaju jer sa svakim dodanim stablom duže traje proces segmentacije. Veličinu prozora je najbolje staviti negdje oko 10 do 15 piksela jer tada se dobivaju najbolji rezultati. Što se tiče dubine stabla, vidi se da ona ne utječe previše na performanse detekcije, ali to vrijedi samo ako ima malo klasa. U slučaju više klasa preporučljivo je ili izgraditi svako stablo do kraja bez ograničenja dubine ili dubinu ograničiti barem na 10 čvorova.

7. Zaključak

Ovaj rad se primarno bavio slučajnim šumama i njihovim ulogama u procesu segmentacije slika. U skladu s tim svaki parametar koji se mjenjao u evaluaciji bio je vezan uz slučajne šume. Dobiveni rezultati su poprilično dobri ako se uzme u obzir da je ovo praktički čista metoda bez ikakvih poboljšanja, a testni skup slika je jako zahtjevan za segmentaciju. Ono što se može vidjeti da kvaliteti segmentacije najviše pridonosi broj stabala koji se koristi te nešto manje veličina prozora za dobivanje karakteristike piksela. Dubina stabla nije previše bitna, ali to se može promjeniti ukoliko postoji više razreda objekata koji se segmentiraju na slici.

U budućem radu svakako bi se trebalo pozabaviti problemom odabira idealne značajke, tj. načina dobivanja iznosa značajke korištenjem drugih metoda osim klasifikacijskog okna koje je korišteno u ovom radu, za podjelu prilikom generiranja stabla. Također, način dobivanja značajke piksela ima jako velik utjecaj na preciznost segmentacije te bi svako poboljšanje metoda koje se bave tim bilo od velike koristi. Testovi sa više klasa objekata na slici bi također jako pomogli u još boljoj evaluaciji preciznosti ovog načina segmentacije slike te bi pokazali da li onda dubina stabla ima veću ulogu u tome.

8. Literatura

[1] Benedikt Waldvogel. Accelerating Random Forests on CPUs and GPUs for Object-Class Image Segmentation (Master's Thesis). Rheinische Friedrich – Wilhelms - Universität Bonn, Njemačka

[2] D.Phil Thesis. Semantic Image Segmentation and Web-Supervised Visual Learning. University Of Oxford, Engleska

[3] Nikola Bogunović. Algoritmi strojnog učenja – 1, Strojno učenje (predavanja, URL http://www.zemris.fer.hr/predmeti/kdisc/Algoritmi_1.ppt). Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Hrvatska

[4] Bojana Dalbelo Bašić. Stabla odluke, Strojno učenje (predavanja, URL http://www.fer.unizg.hr/_download/repository/SU-4-Stabla-odluke.pdf). Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Hrvatska

[5] Computer Vision – The Integral Image. URL https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

[6] Bauhaus-Universität Weimar. Machine learning (predavanja, URL http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/machine-learning/unit-en-decision-trees-impurity.pdf)

[7] Jamie Shotton. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. University of Cambridge. Engleska

Vrednovanje postupka semantičke segmentacije temeljenog na slučajnim šumama

Sažetak

Rad razmatra i evaluira postupak semantičke segmentacije slika temeljene na slučajnim šumama. Objašnjeni su koncepti stabla odlučivanja i slučajne šume te način rada semantičke segmentacije slika. Evaluirana je metoda segmentacije temeljena na slučajnim šumama te je unakrsnom evaluacijom određena preciznost detekcije u ovisnosti o promjeni parametara poput broja stabala, veličine prozora i dubine stabala. Za evaluaciju metode korišten je skup slika „KITTI road“ koji je objavio Karlsruhe Institute of Tehnology i koji se koristi kao referenca za kvalitetu algoritama namijenjenih za detekciju ceste. Također je paralelno sa radom razvijen i softver za označavanje razreda slika koji se može koristiti za buduća istraživanja povezana sa semantičkom segmentacijom.

Ključne riječi: semantička segmentacija; slučajne šume; stablo odluke; integralna slika; vrednovanje postupka semantičke segmentacije; OpenCV

Experimental evaluation of semantic segmentation by random forests

Abstract

This paper examines and evaluates algorithm for semantic image segmentation by random forests. Concepts of decision trees and random forest have been explained along with the semantic image segmentation algorithm. Semantic segmentation by random forests precision has been evaluated and cross-validated with changing parameters such as number of decision trees, window size and tree depth. Picture set that has been used for evaluation is „KITTI road“ which was published by Karlsruhe Institute of Tehnology and which is used as a reference for detecting quality of road detecting algorithms. New software for image labeling has also been developed alongside with this paper which can be used for future semantic segmentation scientific papers.

Keywords: semantic segmentation; random forests; decision tree; integral image; experimental evaluation of semantic segmentation; OpenCV