SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 313

Detekcija objekata naučenim značajkama histograma boje

Tomislav Babić

Zagreb, lipanj 2011.

Sadržaj

1. Uvod

U današnje vrijeme automatsko prepoznavanje prometnih znakova dobiva sve više na značenju. Sustavi za prepoznavanje prometnih znakove se mogu koristiti za povećavanje sigurnosti cesta, bolje održavanje cesta, pronalaženje oštećenih znakova ili mjesta na kojima su prometni znakovi odstranjeni. Također, ta rješenja se mogu koristiti za povećanje sigurnosti vozila. Vozač u automobilu može previdjeti prometni znak zbog umora ili nekog drugog razloga. Stoga je cilj ugraditi u vozila sustave koji bi mogli detektirati i raspoznati prometne znakove u prihvatljivom vremenskom intervalu. Takvi sustavi bi se mogli koristiti kao potpora vozaču pružajući mu informacije kojih možda nije bio svjestan. Rezultat korištenja takvih sustava bio bi povećanje sigurnosti u prometu.

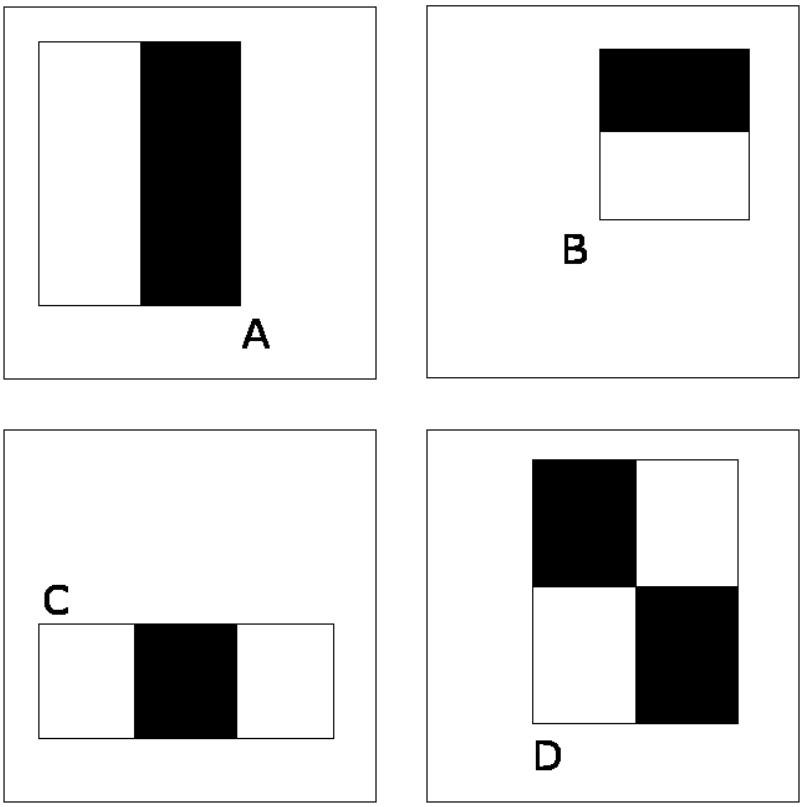
Prepoznavanje objekata je jedan od glavnih problema u području računalnog vida. Paul Viola i Michael Jones su razvili algoritam[1] koji je u stanju brzo i točno detektirati objekte na slikama. Metoda radi tako da se na temelju Haarovih značajki naprave slabi klasifikatori. Zatim se algoritmom Adaboost [2] od slabih klasifikatora napravi snažni klasifikator. Naposljetku se stvara kaskada snažnih klasifikatora kako bi se ubrzao postupak detekcije. U postupku detekcije koristi se okno koje sustavno prolazi kroz sliku te se zatim povećava. Okno na svakoj poziciji, koristeći naučenu kaskadu, provjerava postojanje traženog objekta.

Pokazalo se da algoritam daje obećavajuće rezultate kod prepoznavanja prometnih znakova[3]. Međutim, algoritam ne uzima u obzir informaciju o boji, s obzirom da se Haarove značajke računaju na temelju sive slike. Preliminarno istraživanje[4] u kojima je korišten histogram boje te su se prepoznavali prometni znakovi isključivo temeljem boje, dalo je obećavajuće rezultate.

2. Značajke

2.1. Haarova značajka i integralna slika

Haarove značajke su se pokazale jako dobrima za detekciju temeljenu na obliku. Na slici 1. možemo vidjeti četiri primjera osnovnih Haarovih značajki.



Slika 1. Osnovni tipovi značajki Viole i Jonesa. Sve ostale značajke u detekcijskom oknu dobivamo translacijom i skaliranjem osnovnih značajki.

Vrijednost značajki se računa kao razlika suma slikovnih elemenata (piksela) unutar različitih pravokutnih područja slike. Značajke sa dva pravokutnika (značajke A i B na slici 1) računaju se kao razlika između sume slikovnih elemenata u bijelom i sume slikovnih elemenata u tamnom pravokutniku. Značajka s tri pravokutnika (značajka C na slici 1) se računa kao zbroj suma slikovnih elemenata dvaju vanjskih pravokutnika umanjena za sumu slikovnih elemenata unutrašnjeg pravokutnika.

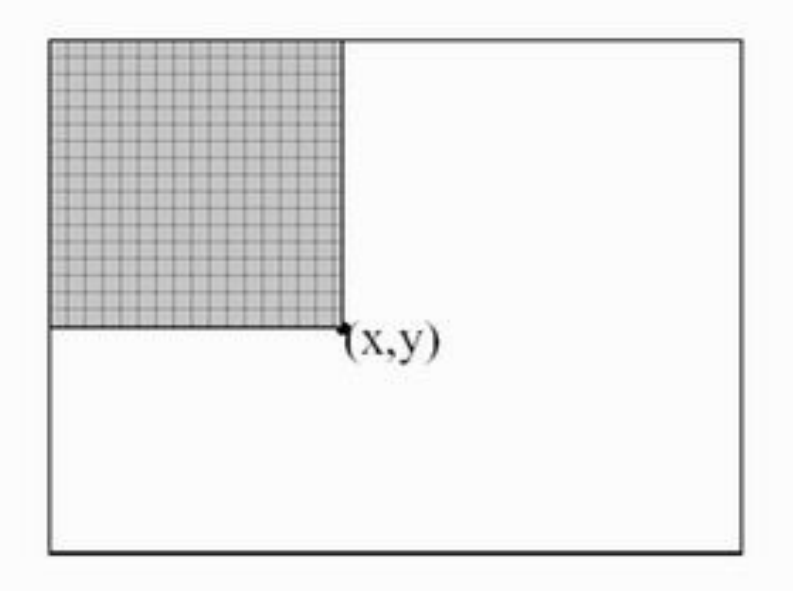
Naposljetku, značajka s četiri pravokutnika (značajka D na slici 1) se računa kao razlika između suma tamnih i bijelih dijagonalno posloženih pravokutnika. Translacijom i skaliranjem osnovnih značajki u prozoru veličine 24x24 se dobiva ukupan skup od otprilike 160,000 značajki.

Računanje suma slikovnih elemenata u pravokutnim područjima može biti procesno izuzetno zahtjevno ako se treba često ponavljati. Za ubrzanje tog postupka može se koristiti integralna slika [1]. Svaki slikovni element u integralnoj slici S_i se dobije tako da se izračuna zbroj svih slikovnih elemenata originalne slike S lijevo i iznad traženog slikovnog elementa, uključujući i traženi:

$$S_i(x,y) = \sum_{x' \le x, y' \le y} S(x',y') \tag{1}$$

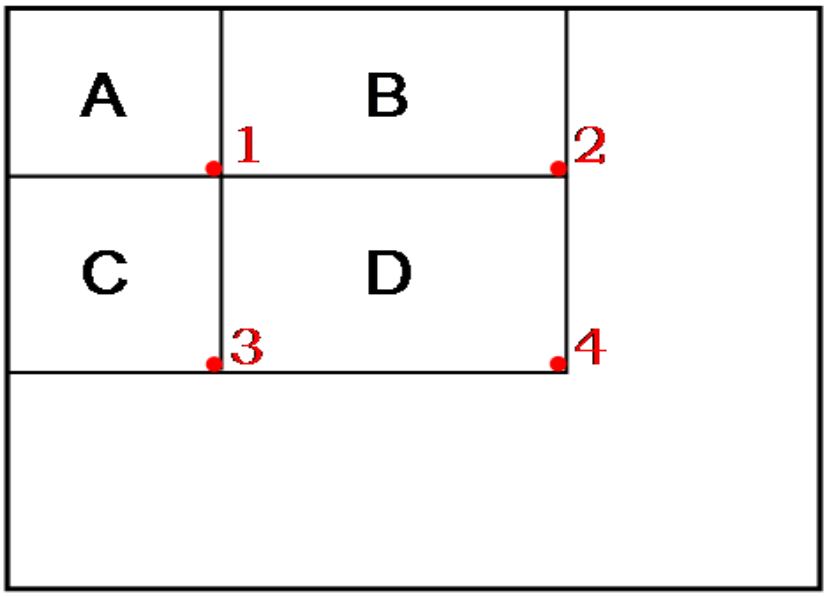
Ipak i postupak dobivanja integralne slike se može ubrzati i izračunati u samo jednom prolazu koristeći sumu S_s svih slikovnih elemenata u stupcu iznad traženog slikovnog elementa:

$$\begin{aligned} S_s(x,y) &= S_s(x,y-1) + S(x,y) \\ S_i(x,y) &= S_s(x,y) + S_i(x-1,y) \end{aligned} \tag{2}$$



Slika 2. Slikovni element na koordinatama x i y računa se kao suma slikovnih elemenata u označenom pravokutniku

Korištenjem integralne slike može se vrlo brzo i jednostavno izračunati suma slikovnih elemenata unutar bilo kojeg pravokutnika u originalnoj slici kao što je ilustrirano na slici 3. Za izračun vrijednosti Haarove značajke s dva pravokutnika, dovoljno je pristupiti integralnoj slici šest puta, za značajku sa tri pravokutnika vrijednost se može izračunati s osam pristupa integralnoj slici te za značajku s četiri pravokutnika dovoljno je pristupiti devet puta.



Slika 3. Suma slikovnih elemenata u pravokutniku D na originalnoj slici može se izračunati kao: $D = S_{i1} – S_{i2} – S_{i3} + S_{i4}$, gdje su S_{i1}, S_{i2}, S_{i3} i S_{i4} vrijednosti elemenata u integralnoj slici S_i na označenim lokacijama. To se može lako pokazati uvrštavanjem: $S_{i1} = A$, $S_{i1} = A + B$, $S_{i1} = A + C$, $S_{i1} = A + B + C + D$.

Kako bi se smanjio efekt različitih svjetlosnih uvjeta, potrebno je provesti normalizaciju varijancom. Varijanca odnosno standardna devijacija se može brzo izračunati pomoću obične integralne slike i integralne slike kvadriranih slikovnih elemenata Ako je m srednja vrijednost (engl. mean), p_{ij} vrijednost slikovnog elementa unutar okna, a N ukupan broj slikovnih elemenata u oknu, standardna devijacija σ se može odrediti kao:

$$\sigma^2 = m^2 - \frac{1}{N}\sum_{i,j} p_{ij}^2 \tag{3}$$

Ako *w* i *h* predstavljaju širinu odnosno visinu okna tada se srednja vrijednost slikovnih elemenata može lako izračunati pomoću integralne slike S_i:

$$m = \frac{S_i(x,y) - S_i(x+w,y) - S_i(x,y+h) + S_i(x,y)}{(x+w)\cdot(y+h)} \tag{4}$$

Suma kvadratnih vrijednosti slikovnih elemenata se može izračunati pomoću integralne slike kvadriranih slikovnih elemenata S_q:

$$\sum_{i,j} p_{ij}^2 = S_q(x,y) - S_q(x+w,y) - S_q(x,y+h) + S_q(x,y) \tag{5}$$

$$S_q(x,y) = \sum_{x' \le x, y' \le y} S(x',y')^2 \tag{6}$$

Prilikom detekcije efekt normalizacije se može postići dijeljenjem vrijednosti značajke s devijacijom.

2.2. Histogramska značajka i integralni histogram

Haarove značajke se računaju na sivim slikama te ne uzimaju o obzir informaciju o boji. Kako bi se ta informacija iskoristila koristi se histogramska značajka koja se bazira na histogramu boje.

Histogram boje se koristi za mjeru distribucije boja u slici. Kod digitalnih slika, histogram boje se zapravo može predstaviti kao niz odjeljaka (eng. bin) gdje svaki odjeljak sadrži broj slikovnih elemenata koji se nalaze u određenom rasponu vrijednosti. Na slici 4. možemo vidjeti histogram izgrađen u prostoru crvene i plave boje.

		red			
		0-63	64-127	128-191	192-255
blue	0-63	43	78	18	0
	64-127	45	67	33	2
	128-191	127	58	25	8
	192-255	140	47	47	13

Slika 4. Histogram u prostoru crvene (red) i plave (blue) boje. Slika preuzeta s wikipedije [5].

Umjesto da svaki slikovni element doprinosi samo jednom odjeljku, u ovom radu se koristi histogram s proporcionalnim doprinosom. U tom slučaju jedan slikovni element doprinosi dva odjeljka po boji (ukupno osam). Primjerice za histogram crvene boje sa osam odjeljaka (svaki odjeljak ima raspon iznosa 32), slikovni element s crvenom vrijednošću od 32 bi bio točno na granici prva dva odjeljka. Tada se brojač u ta dva odjeljka povećava za 0.5. Ukoliko bi element imao nižu vrijednost, onda bi proporcionalno doprinosio više prvom odjeljku, a manje drugom i obrnuto.

Histogramska značajka se sastoji od raspona vrijednosti za sve tri boje, tj. odgovarajućeg odjeljka u histogramu boja te vrijednosti tog odjeljka. Ukupan broj osnovnih značajki za histogram s osam odjeljaka po boji je 512.

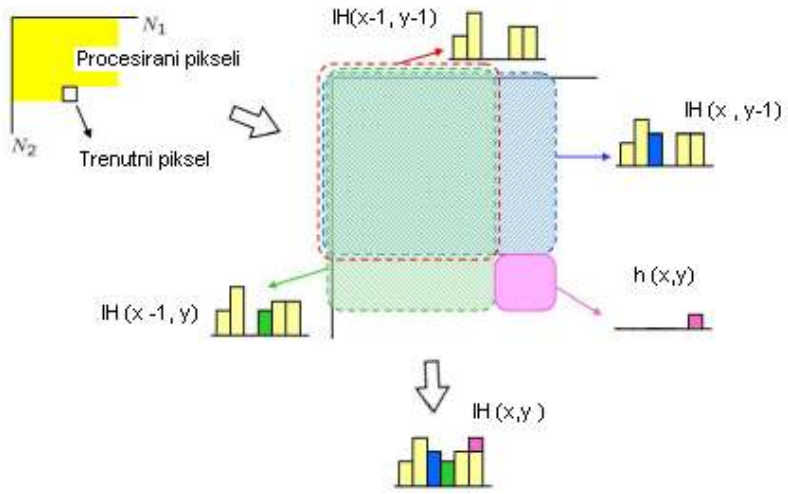
Poput Haarove značajke, računanje histograma također može biti zahtjevan proces ako se često ponavlja. U tu svrhu se, po uzoru na integralnu sliku, koristi integralni histogram[6].

Integralni histogram se može predstaviti kao dvodimenzionalna matrica dimenzija slike za koju se i računa. Tada je svaki element u toj matrici zapravo izgrađeni histogram slike od njezinog početnog elementa do pozicije elementa koji odgovara poziciji u matrici. Označimo s IH (x, y) element integralnog histograma na poziciji x i y, a H(x1, y1, x2, y2) histogram dijela slike izgrađen između dvije točke čije pozicije odgovaraju pozicijama x1 i y1, odnosno x2 i y2. Tada vrijedi:

$$IH(x, y) = H(0, 0, x, y). \quad (7)$$

Označimo s h(x,y) funkciju koja računa histogram za jedan slikovni element na poziciji x, y u slici.. Tada se histogram slike može računati i propagacijom na sljedeći način:

IH (x,y) = IH (x , y-1) + IH (x -1, y) – IH(x-1, y-1) + h(x,y). (8)



Slika 5. Na slici možemo vidjeti ilustraciju propagacije histograma. Slika preuzeta iz [6].

Kada iz integralnog histograma želimo dobiti histogram slike počevši od točke x1, y1 do točke x2 i y2, tada se to može napraviti na sljedeći način:

H(x1,y1, x2, y2) = IH (x2, y2) – IH(x1-1, y2) – IH (x2, y1-1) + IH (x1-1, y1-1). (9)

Jedan od problema kod korištenja integralnog histograma je njegova veličina. Na svakoj poziciji se nalazi histogram, zbog čega ukupan broj histograma može narasti i na nekoliko stotina tisuća. Neka su *w* i *h* dimenzije slike te neka je *n* broj odjeljaka po boji, a *c* broj boja. Također neka je *s* veličina jednog odjeljka (tj. veličina tipa podatka). Prostorna složenost, *S(IH)*, za integralni histogram je dana sljedećim izrazom:

$$S(IH) = w \cdot h \cdot n^c \cdot s \quad (10)$$

U ovom radu su korištene slike dimenzija 720x576 te histogram s osam odjeljaka po boji tj. ukupno 512 odjeljaka. Veličina integralnog histograma za odjeljke s dvostrukom preciznošću, odnosno 8 B (double), iznosi otprilike 1600 MB. Zbog toga je vrlo nepraktično koristiti više od jednog integralnog histograma istovremeno.

Postupak izgradnje integralnog histograma je također značajno sporiji od izgradnje integralne slike. Prvi razlog je u tome što dodavanje prethodnih elemenata integralnog histograma novom elementu uključuje zbrajanje svih odjeljaka, što znači n^c više operacija zbrajanje nego u slučaju integralne slike. Također dodavanje trenutnog slikovnog elementa u integralnoj slici uključuje samo jednu operaciju zbrajanja. Kod integralnog histograma korištenog u ovom radu, taj postupak uključuje i operacije dijeljenja i množenja kako bi se mogao izračunati proporcionalni doprinos.

3. Algoritam Adaboost

Ideja algoritma adaboost je stvaranje jednog snažnog klasifikatora od više slabih klasifikatora. Slabi klasifikator se sastoji od značajke (*f*) koju koristi, praga (*t*) i smjera usporedbe (*p*). Ukoliko je vrijednost značajke za neki uzorak (*x*) veća od praga uz smjer usporedbe *veće od* klasifikator će dati pozitivan odgovor. Također ako je vrijednost značajke manja od praga uz smjer usporedbe *manje* tada će klasifikator isto dati pozitivan odgovor. U suprotnim slučajevima, klasifikator daje negativan odgovor:

$$h(x, f, p, t) = \begin{cases} 1 & \text{ako } \mathrm{p} \cdot \mathrm{f}(\mathrm{x}) < \mathrm{p} \cdot \mathrm{t} \\ 0 & \end{cases} \tag{11}$$

Takav klasifikator se obično naziva i binarnim, jer ima samo dva moguća odgovora (1 i 0). Slabi klasifikator je tako nazvan jer se očekuje da će u više od 50% slučajeva točno klasificirati uzorke.

Algoritam Adaboost radi tako da se na početku svake iteracije vrši inicijalizacija težina za zadani skup uzoraka za učenje. Zatim se nad skupom svih značajki traži slabi klasifikator s najmanjom pogreškom. Težine uzoraka se osvježavaju s obzirom na rezultate klasifikacije za trenutni slabi klasifikator. Zatim se izračunava težina slabog klasifikatora ovisno o njegovoj točnosti. Postupak se ponavlja sve dok se ne pronađe zadan broj slabih klasifikatora. Na kraju se dobiva snažni klasifikator linearnom kombinacijom slabih klasifikatora i njihovih težina. Prag snažnog klasifikatora se definira kao polovica sume težina slabih klasifikatora. Precizniji opis postupka se može vidjeti u sljedećem pseudokodu:

- *N* – zadani broj slabih klasifikatora koji će uči u snažni klasifikator
- *X* – skup uzoraka za učenje
- *Y* – Skup klasifikacija uzoraka za učenje (ako je x_i pozitivan tada je $y_i = 1$ odnosno $y_i = 0$ u obrnutom slučaju)
- *K* – broj uzoraka u skupu za učenje
- *L* – broj pozitivnih uzoraka u skupu za učenje
- *M* – broj negativnih uzoraka u skupu za učenje
- Inicijalizacija težina: $w_{1,i} = \frac{1}{2L}, \frac{1}{2M}$ za $y_i = 0,1$
- Za $n = 1$ do N
  - Izračun sume težina: $s_n = \sum_{j=1}^{K} w_{n,j}$
  - Normalizacija težina: $w_{n,i} = \frac{w_{n,i}}{s_n}$
  - Pronaći i odabrati najbolji slabi klasifikator s obzirom na težinsku pogrešku: $e_n = \min_{f,p,t} \sum_{i=1}^{K} w_i \cdot \left| h(x_i, f, p, t) - yi \right|$
  - Definiraj: $h_n = h(x, f_n, p_n, t_n)$ gdje f_n, p_n i t_n minimiziraju e_n.
  - Definiraj: $\beta_n = \frac{e_n}{1 - e_n}$
  - Osvježavanje težina: $w_{n+1,i} = w_{n,i} \cdot \beta_n^{1-e_i}$ gdje je $e_i = 0$, ako je x_i klasificiran ispravno, odnosno $e_i = 1$ ako je x_i klasificiran pogrešno od strane klasifikatora h_n.
  - Definiraj $\alpha_n = \log(\frac{1}{\beta_n})$.
- Definiraj snažni klasifikator kao:

$$C(x) = \begin{cases} 1 & \text{ako} \quad \sum_{n=1}^{N} \alpha_n \cdot h_n(x) \geq 0.5 \cdot \sum_{n=1}^{N} \alpha_n \\ 0 \end{cases}$$

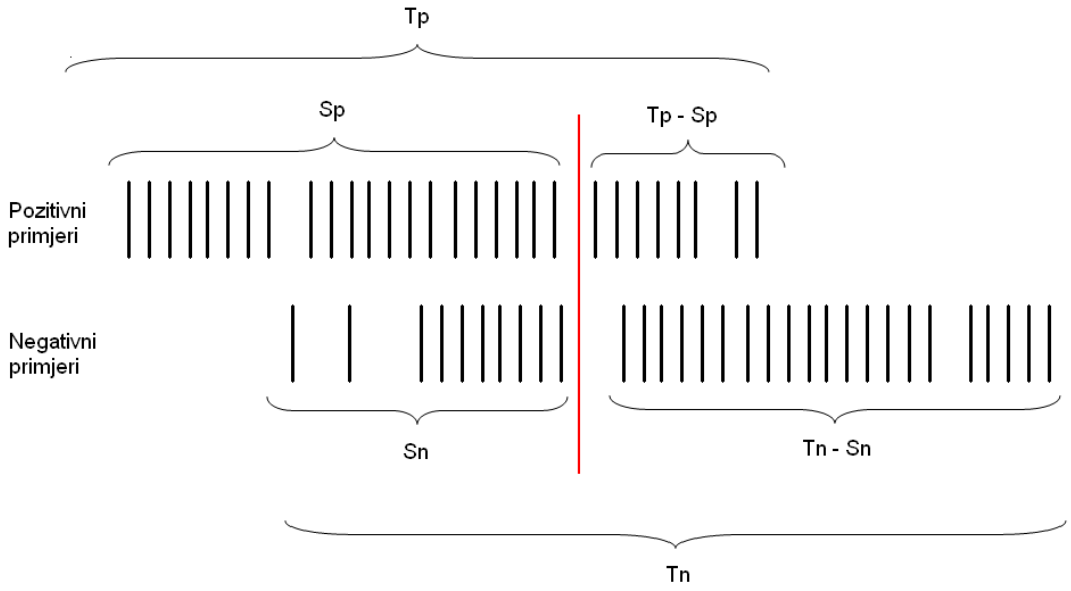
Pronalaženje najboljeg slabog klasifikator u iteraciji se može efikasno izračunati na sljedeći način[1]. Za svaku značajku uzorci za učenje se uzlazno sortiraju po vrijednosti te značajke. Tada se pogreška može izračunati osvježavanjem četiri sume jednim prolaskom kroz sortiranu listu uzoraka (slika 6.). Neka je Tp suma težina svih

pozitivnih uzoraka, neka je Tn suma težina svih negativnih uzoraka. Neka je Sp suma težina svih pozitivnih uzoraka prije trenutnog uzorka u listi. Naposljetku, neka je Sn suma težina svih negativnih uzoraka prije trenutnog uzorka u listi. Definirajmo trenutni prag kao aritmetičku sredinu između trenutnog i prethodnog uzorka. Tada se pogreška za taj prag može izračunati na sljedeći način:

$$e = \min(Sp + (Tn - Sn), Sn + (Tp - Sp)) \tag{12}$$

Vrijednost *p* koja definira smjer usporedbe je dana sljedećim izrazom:

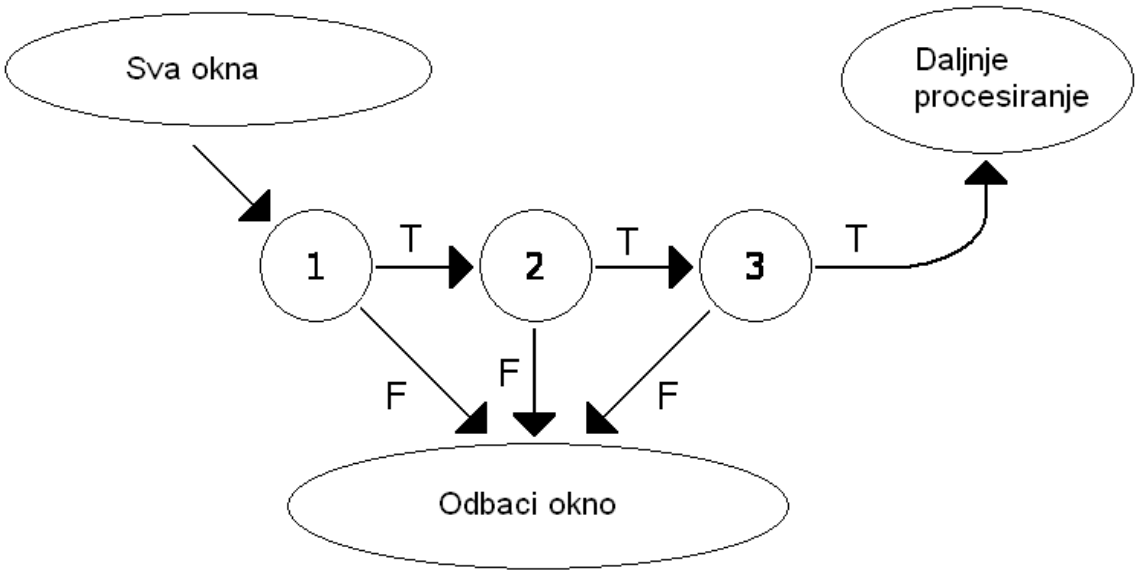
$$p = \begin{cases} 1 & \text{ako } Sp + (Tn - Sn) > Sn + (Tp - Sp) \\ 0 & \end{cases} \tag{13}$$



Slika 6. Ilustracija traženja optimalnog praga i pogreške za značajku.

4. Stvaranje kaskade

Snažni klasifikator koji bi bio prikladan za postupak detekcije trebao bi se sastojati od iznimno velikog broja značajki (reda veličine nekoliko stotina). Takav klasifikator bi u postupku detekcije bio iznimno spor. Umjesto toga koristi se više snažnih klasifikatora s manjim brojem značajki. Ako boostani klasifikator u i-toj razini zadovoljava prag to znači da je traženi objekt možda u oknu, te se nastavlja s evaluacijom u segmentu i+1. Ukoliko prag nije zadovoljen, ne pozivaju se daljnji klasifikatori, već se ustanovljuje da traženi objekt nije u oknu te se ono pomiče na novu poziciju. Takav "serijski" spoj klasifikatora se naziva kaskada (slika 7).



Slika 7. Arhitektura i ilustracija izvođenja kaskade.

Kaskada se oblikuje tako da na početku budu klasifikatori koji odbacuju najviše negativnih primjera (jer u slikama na većini pozicija detekcijskog okna neće biti traženog objekta). Takva kaskada može postići rezultate jednako dobre kao veliki klasifikator, uz brže performanse.

Neka je F učestalost lažno-pozitivnih detekcija (eng. false-positive rate) cijele kaskade, a f_i učestalost lažno-pozitivnih detekcija i-tog snažnog klasifikatora u kaskadi te neka je K broj klasifikatora u kaskadi. Tada vrijedi sljedeći izraz:

$$F = \prod_{i=1}^{K} f_i \tag{14}$$

Neka je D učestalost ispravnih detekcija (eng. detection rate) cijele kaskade, a d_i učestalost ispravnih detekcija i-tog snažnog klasifikatora u kaskadi te neka je K broj klasifikatora u kaskadi. Tada vrijedi sljedeći izraz:

$$D = \prod_{i=1}^{K} d_i \tag{15}$$

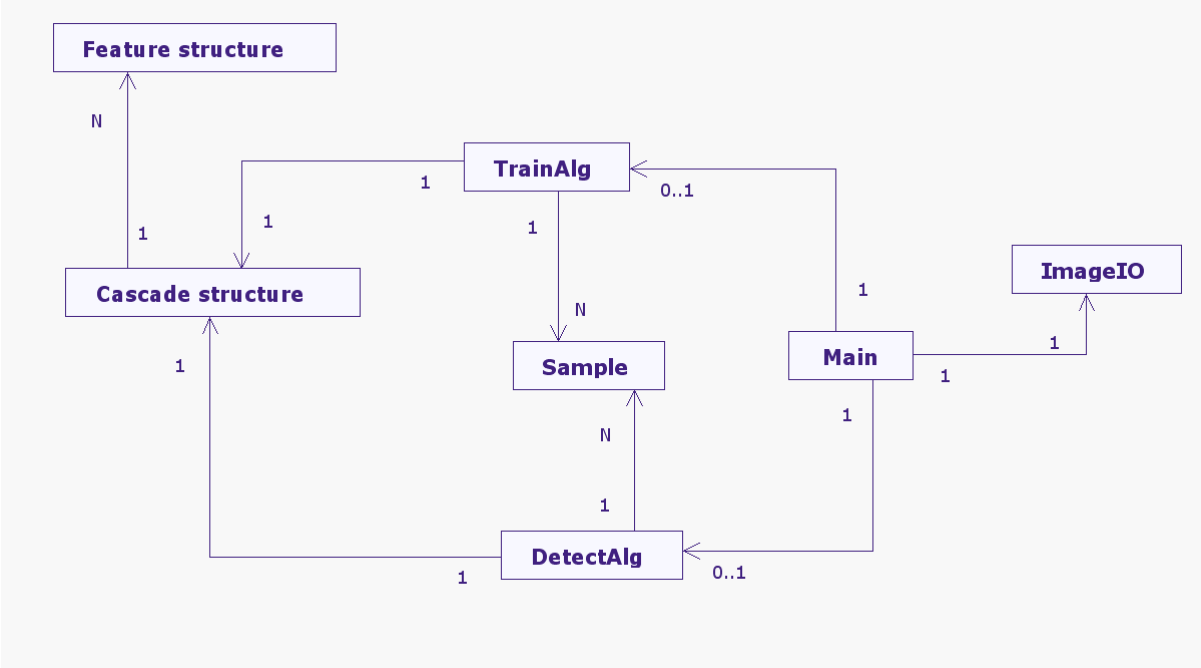
Postizanje željenih ciljeva za učestalost ispravnih i lažno-pozitivnih detekcija moguće je određivanjem željenih učestalosti za snažni svaki klasifikator u kaskadi. Primjerice za kaskadu s 10 snažnih klasifikatora i sa željenom učestalošću ispravnih detekcija od 0.9, svaki klasifikator bi trebao imati učestalost ispravnih detekcija od 0.99. Taj zadatak je olakšan činjenicom da svaki klasifikator treba imati učestalost lažno-pozitivnih detekcija od 0.3. Tada će cijela kaskada imati učestalost lažno pozitivnih detekcija u iznosu od otprilike $6 \cdot 10^{-6}$.

Za stvaranje kaskade korisnik mora zadati maksimalnu prihvatljivu učestalost lažno-pozitivnih detekcija i minimalnu prihvatljivu učestalost ispravnih detekcija. Korisnik također mora zadati željenu učestalost lažno-pozitivnih detekcija za cijelu kaskadu. Za svaki sloj kaskade se pokreće algoritam adaboost. Zatim se dobivenom klasifikatoru spušta prag sve dok ne zadovolji željenu učestalost ispravnih detekcija za taj sloj. Ako željena učestalost lažno-pozitivnih detekcija nije zadovoljena, tada se postupak ponavlja s uvećanim brojem slabih klasifikatora u snažnom klasifikatoru. Nakon što su željena učestalost ispravnih detekcija i učestalost lažno-pozitivnih detekcija za taj sloj zadovoljeni, postupak se ponavlja za sljedeći sloj. Postupak završava nakon što je postignuta željena učestalost lažno-pozitivnih detekcija za cijelu kaskadu. Opis postupka se može vidjeti i u sljedećem pseudokodu:

- f – maksimalna prihvatljiva učestalost lažno-pozitivnih detekcija za svaki sloj
- d – minimalna prihvatljiva učestalost ispravnih detekcija za svaki sloj
- F_{cilj} – željena učestalost lažno-pozitivnih detekcija za cijelu kaskadu
- P – skup pozitivnih uzoraka u skupu za učenje
- N – skup negativnih uzoraka u skupu za učenje
- $F_0 = D_0 = 1.0$
- $i = 0$
- dok $F_i > F_{cilj}$
    - $i=i+1$
    - $n_i = 0$
    - $F_i = F_{i-1}$
    - dok $Fi > f \cdot F_{i-1}$
        - $ni = ni+1$
        - pokretanje adaboost algoritma koristeći P i N kao skup uzoraka za učenje te sa n_i kao željenim brojem slabih klasifikatora u snažnom klasifikatoru.
        - evaluacija dosad naučene kaskadom i dobivenim snažnim klasifikatorom nad skupom za validaciju
        - Smanjivati prag dobivenog snažnog klasifikatora sve dok učestalost ispravne detekcije nije barem $d \cdot D_{i-1}$ na skupu za validaciju (ovaj postupak također utječe i na F_i)
    - dodaj zadnji dobiveni snažni klasifikator u kaskadu.
    - $N = \{\emptyset\}$
    - ako $F_i > F_{cilj}$ tada evaluiraj trenutnu kaskadu i iz skupa lažno-pozitivnih uzoraka nasumično odaberi određen broj uzorak i dodaj u skup N

5. Programsko rješenje

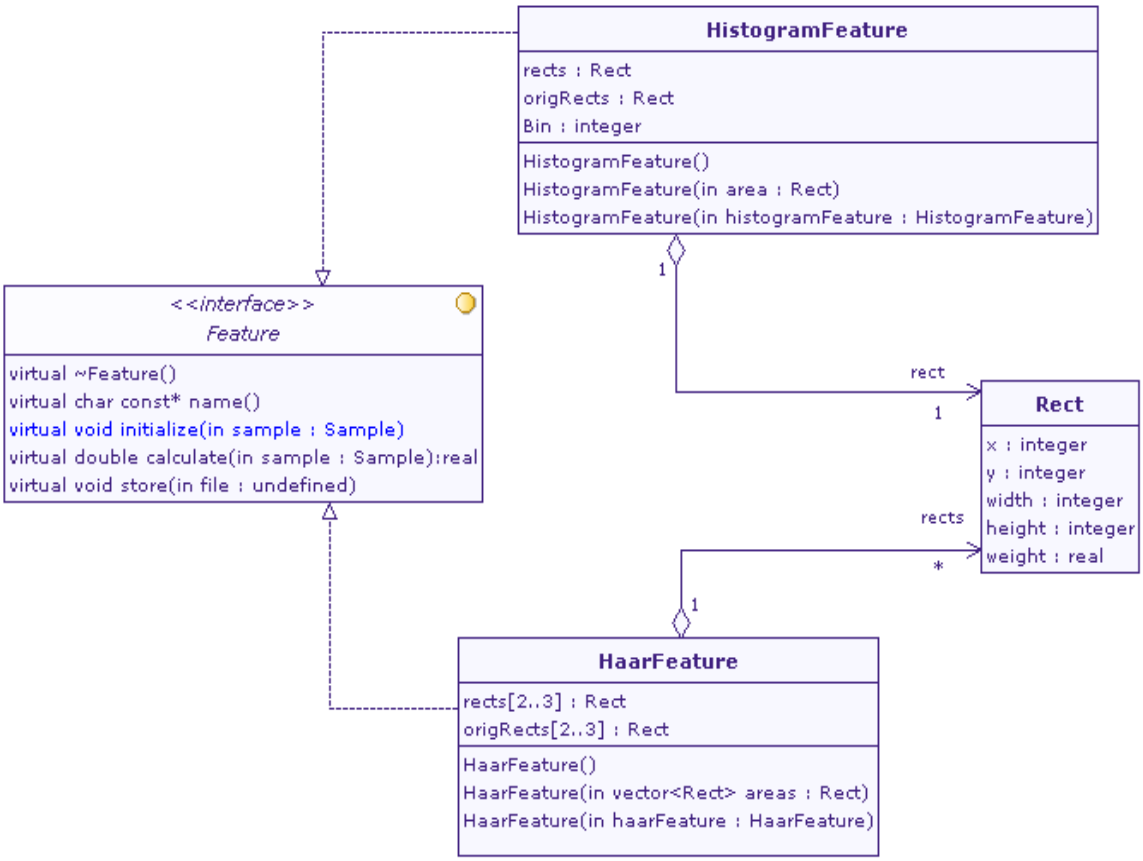
Implementacija ovog rada podijeljena je u nekoliko segmenata koji se odnose na značajke, strukturu kaskade, uzorke, komponente za učitavanje i spremanje slika, algoritam za učenje, algoritam za detekciju te pomoćne i testne funkcije. Osnovna struktura implementacije se može vidjeti na slici 8.



Slika 8. Osnovna struktura implementacije.

U sklopu dijela izvornog koda koji se odnosi na značajke, implementirani su razredi za histogramsku (`HistogramFeature`) i Haarovu (`HaarFeature`) značajku. Oba razreda su nasljeđivanjem izvedana iz apstraktne klase `Feature`, koja predstavlja sučelje značajki prema ostalim dijelovima implementacije. Dijagram klasa se može vidjeti na slici 9.

```
class Feature
{
public:
	virtual ~Feature(){}
	virtual char const* name() =0;
	virtual void initialize(Sample &sample) = 0;
	virtual double calculate(Sample& sample) = 0;
	virtual void store(ofstream &file) = 0;
};
```



Slika 9. Dijagram klasa koji prikazuje strukturu značajki.

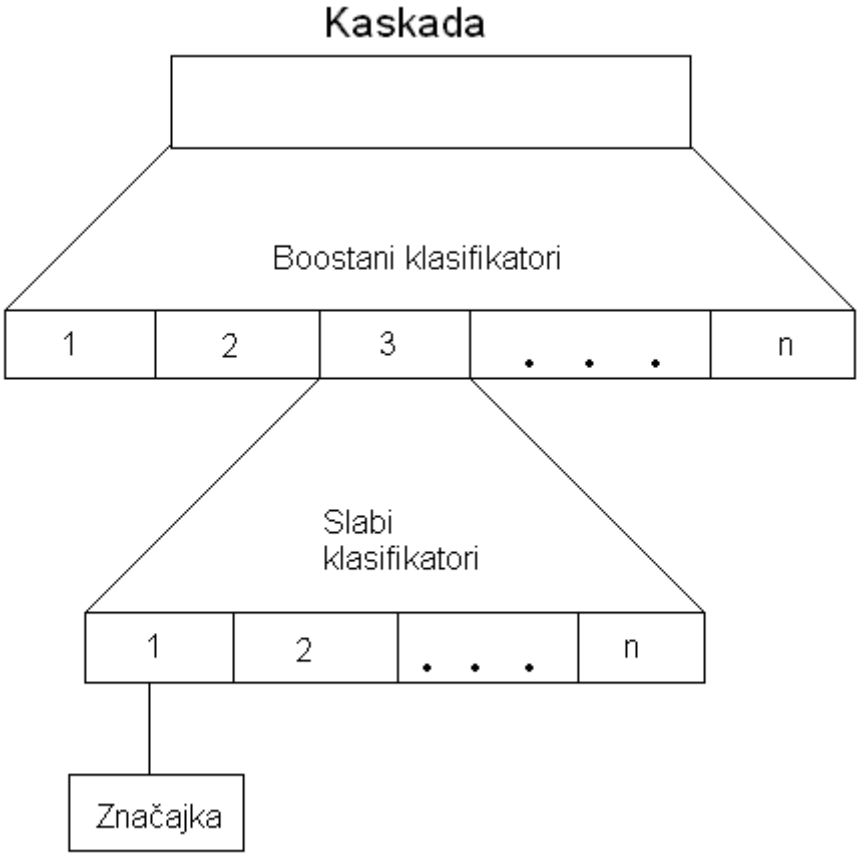
Funkcijom `initialize(…)` značajke se inicijaliziraju i skaliraju za veličinu primljenog uzorka. Ta funkcija omogućava da se prilikom detekcije značajke ne moraju skalirati pri svakoj poziciji detekcijskog okna, već samo onda kada se detekcijsko okno povećava.

Funkcija `calculate(…)` izračunava vrijednost značajke, kao što je opisano u 2. poglavlju. Funkcija `store(…)` sprema značajku u primljenu datoteku. Funkcija `name()` vraća ime značajke (*histogram_feature* odnosno *Haar_feature*).

U komponenti featureUtil.cpp implementirane su funkcije koje stvaraju ukupan skup značajki.

Struktura kaskade je implementirana pomoću tri klase (slika 10). Klasa `cascade` sadrži vektor boostanih klasifikatora. Klasa `BoostClassifier` sadrži vektor slabih

klasifikatora i vektor njihovih težina. Klasa `WeakClassifier` sadrži pokazivač na apstraktnu klasu `Feature`.



Slika 10. Prikaz strukture kaskade i boostanih klasifikatora odnosno slabih klasifikatora.

Sve tri klase sadrže `initialize(…)`, `run(…)`, `store(…)`, pri čemu se inicijalna funkcija poziva na razini kaskade. Funkcija kaskade će zatim pozvati istoimenu funkciju boostanog klasifikatora, koja će zatim pozvati funkciju slabog klasifikatora. Slabi klasifikator će pozvati odgovarajuću funkciju značajke. Funkcija `initialize(…)` omogućuje inicijalizaciju svih značajki u trenutnoj razini kaskade tj. klasifikatora. Izvršavanje kaskade tj. klasifikatora se odvija u funkciji `run(…)`. Funkcija `store(…)` izvršava spremanje kaskade/klasifikatora u datoteku.

Sve tri klase također sadrže i funkcije za dodavanje, dohvaćanje i brisanje klasifikatora tj. značajki.

Struktura uzoraka je implementirana u klasi Sample koja sadrži dimenzije uzorka, pokazivače na sliku, integralnu sliku (i integralnu sliku kvadrata) i integralni histogram. S obzirom da je zbog veličine korištenje integralnog histograma neprikladno, klasa također sadrži i pokazivač na histogram cijelog uzorka. Tada se memorija koju zauzima integralni histogram može osloboditi. U konstruktoru klase potrebno je predati originalnu sliku u boji te poziciju i veličinu uzorka u toj istoj slici. Ostali elementi u uzorku se mogu predati pomoću odgovarajuće `set…(…)` funkcije. (primjerice `void setHistogram(ColorHist &hist)` za postavljanje histograma). S obzirom da je za izračun Haarove značajke potrebna i varijanca sive slike, ona se može izračunati funkcijom `calculateGrayVariance()`. Ta funkcija je također implementirana u klasi `Sample`.

Komponente za učitavanje i spremanje slika su preuzete su iz cvsh ljuske, koja je razvijena na zavodu. Također u sklopu tih komponenti preuzeta je i klasa `img_wrap` koja predstavlja sliku.

U komponenti util.cpp su implementirane pomoćne funkcije za zaokruživanje broja, odabir nasumičnog broja iz intervala, spremanje i učitavanje uzoraka na disk, čitanje skupa slika, učitavanje kaskade s diska te skaliranje pravokutnika za neki faktor.

5.1. Implementacija učenja

Postupak učenja se sastoji od algoritma Adaboost (koji je opisan u poglavlju 3.) i algoritma za stvaranje kaskade boostanih klasifikatora (poglavlje 4.). Prvo se poziva funkcija `trainAlgorithm(…)` implementirana u komponenti TrainAlgMain.cpp. Funkcija prima sve potrebne parametre te započinje postupak učenja.

Algoritam za boostanje klasifikatora je implementiran u klasi `AdaBoost`. Algoritam se poziva funkcijom `train(…)` koja prima skup pozitivnih uzoraka za učenje, skup

negativnih uzoraka, skup svih značajki, broj slabih klasifikatora koji ulaze u boostani klasifikator, te referencu boostanog klasifikatora gdje će se spremiti dodatni slabi klasifikatori. Klasa također sadrži kao privatne članove funkcije za normalizaciju i osvježavanje težina, te pronalazak optimalnog praga i izračun pogreške neke značajke.

Postupak za stvaranje kaskade klasifikatora je implementiran u klasi `TrainCascade`. U konstruktoru klase se predaju slike prometnih znakova (pozitiva) i slike pozadina iz kojih se stvaraju uzorci za skup za učenje. U konstruktoru se također predaje i popis tipova struktura koji ulaze u uzorak (integralna slika, integralni histogram i/ili histogram) te popis tipova značajki koje će se stvoriti (Haarova i/ili histogramska značajka).

Pozivom funkcije `startTraining(…)` započinje postupak stvaranja kaskade, kao što je opisan u poglavlju 4. Prije poziva funkcije se može pozvati i funkcija `setValidationSet(…)` koja prima pozitivne i negativne slike iz skupa za validaciju te nad njima stvara uzorke. Kao što je opisano u poglavlju 4. skup za validaciju se koristi za evaluaciju trenutnog boostanog klasifikatora. Ukoliko se dotična funkcija ne pozove, tada će se evaluacija vršiti na skupu za učenje.

Klasa `TrainCascade` putem funkcije `setCascade(…)` ima opciju za primanje već postojeće kaskade. U tom slučaju se neće stvarati nova kaskada, već će nadograditi postojeća s dodatnim boostanim klasifikatorima.

Za stvaranje uzoraka za skup za učenje je zaslužna komponenta sampleUtil.cpp. U njoj je implementirana funkcija `createBaseSamples(…)` koja stvara osnovne uzorke i to jedan uzorak koji sadrži cijelu sliku. Osnovni uzorci dobiveni nad skupom slika prometnih znakova se mogu koristiti i kao pozitivni uzorci.

Funkcija `createNegSamples(…)` stvara negativne uzorke iz osnovnih negativnih uzoraka (koji su stvoreni nad slikama pozadina) i to na jedan od tri načina.

Funkcija `searchSamplesRandom()` prima trenutni uzorak pozadine i kaskadu klasifikatora. Funkcija također prima željeni broj uzoraka koji se trebaju stvoriti iz slike

pozadine. Nasumično se odabire pozicija i veličina negativnog uzorka. Ako kaskada daje pozitivan odgovor (prazna kaskada uvijek daje pozitivan odgovor), tada se negativni uzorak sprema u skup uzoraka za učenje. Funkcija prestaje s radom kada pronađe dovoljan broj uzoraka. Ako funkcija ne može stvoriti dovoljan broj uzoraka, tada se koristi drugi način.

Funkcija `searchSamplesCombo(…)` detekcijskim oknom sustavno pretražuje sliku pozadine (slično kao u postupku detekcije opisanom u poglavlju 5.2) i na svakoj poziciji provjerava odgovor kaskade. Ukoliko je odgovor kaskade pozitivan stvara se novi negativni uzorak s pozicijom i veličinom okna. Ako je broj pronađenih uzoraka veći od broja potrebnog uzoraka, tada se iz skupa pronađenih uzoraka nasumično odabiru uzorci koji ulaze u skup za učenje. Ako je broj uzoraka manji tada se koristi funkcija `searchSamplesAll(…)` koja u skup uzoraka za učenje dodaje sve pronađene uzorke.

5.2. Implementacija detekcije

Postupak detekcije je implementiran u komponenti detAlgMain.cpp. Postupak započinje u funkciji `detectionAlgorithm(…)` koja za svaku sliku poziva funkciju `detectImage(…)`. Nakon obrade pojedinačne slike ispituje se točnost na temelju poznatih podataka o traženim objektima na slici.

U funkciji `detectImage(…)` se slika sustavno pretražuje detekcijskom oknom, koje se definira klasom `Sample`. Inicijalna veličina detekcijskog okna je 24x24 slikovnih elemenata. Integralni histogram i integralna slika se mogu postaviti odgovarajućim funkcijama klase `Sample`. Nakon što okno prođe jednom kroz cijelu sliku, njegove dimenzije se povećavaju za određeni faktor. Iznos tog faktora se predaje funkciji kao parametar. Za inicijalnu veličinu okna te nakon svakog povećavanja potrebno je inicijalizirati i skalirati sve značajke u kaskadi za trenutnu veličinu okna. To se čini pokretanjem funkcije `initialize(…)` primljene kaskade. Na svakoj poziciji okna pokreće se ispitivanje kaskade funkcijom `run(…)`. Ako funkcija daje pozitivan odgovor to znači da je kaskada prepoznala trenutno okno kao traženi objekt. U tom slučaju se

pozicija i veličina trenutnog okna, spremaju u listu pravokutnika koja predstavlja pronađene detekcije.

Ukoliko je u pozivu funkcije `detectImage(…)` postavljena vrijednost varijable `group`, tada će se izvršiti grupiranje pronađenih detekcija. Komponenta ext_vj_grouping.cpp u kojoj je implementirano grupiranje je preuzeta iz cvsh ljuske. U postupku grupiranja se stvara lista pravokutnika. Za svaku listu se izračunava prosječan pravokutnik svih pravokutnika u toj listi. Ako se još nesvrstana detekcija preklapa u zadanim parametrima s prosječnim pravokutnikom liste, tada se ona dodaje u nju. Nakon svakog dodavanja, izračunava se novi prosječni pravokutnik liste. Ako se detekcija ne može svrstati u niti jednu već postojeću listu, tada se stvara nova prazna lista u koju se dodaje detekcija.

5.3. Upute za korištenje

Prilikom pokretanja programa učitava se datoteka config.ini koja sadrži sve konfiguracijske parametre. Ukoliko program ne može učitati takvu datoteku, tada će se koristiti pretpostavljeni parametri. Također ako neki parametri nisu postavljeni u datoteci, onda će se za njih koristiti pretpostavljene vrijednosti. Pretpostavljene vrijednosti su postavljaju pozivom funkcije `initConfig(…)`. Dotična funkcija zatim poziva funkcije `initDetectConfig(…)` i `initTrainConfig(…)` za postavljanje parametara za postupak detekcijske odnosno učenja.

Datoteka config.ini je podijeljena u tri glavna dijela. Prvi dio započinje s linijom `[Main]`. Iza toga se mogu postaviti dva parametra: `useTrain` i `useDetect`, za pokretanje postupka učenja odnosno postupka detekcije. Ako su oba parametra postavljena na vrijednost 1, tada će se prvo izvoditi učenje, a zatim detekcija.

Parametri za detekciju započinju nakon linije `[Detection]`. Parametrom `cascadePath` se zadaje putanja do direktorij u kojem se nalazi naučena kaskada. Pomoću parametra `imgDir` se postavlja putanja do direktorija u kojem se nalaze slike za obradu, datoteka s popisom slika (parametar `imgFileName`) i datoteka s popisom pozicija traženih objekata na slikama (parametar `imgRects`). Obrađene slike s

označenim detekcijama će se spremiti u direktorij koji se može postaviti parametrom `resultDir`. U datoteku koja se postavlja parametrom `detectionRectAddress` se mogu spremiti popis pozicija svih prijavljenih detekcija po slikama.

Postavljanjem parametara `grouping`, `useIntegralHist` i `measureTime`, korisnik može specificirati želi li u postupku detekcije koristi grupiranje, izračun integralnog histograma odnosno osnovni ispis o vremenskog trajanju postupka detekcije. Parametrom `scaleFactor` se može postaviti iznos za koji se detekcijsko okno u svakom prolazu kroz sliku povećava.

Parametri za postupak učenja se mogu postaviti nakon linije `[Training]`. Parametrom `cascadePath` se zadaje putanja do direktorij u kojem postojeća kaskada. Ako je postavljen parametar `loadCascade` na vrijednost 1, tada će se u postupku učenja nadograđivati postojeća kaskada. Pomoću parametara `posDir`, `negDir`, `posFileName` i `negFileName` se potrebne putanje za učitavanje slika znakova i pozadina.

Pomoću parametra `sampleTypes` može se specificirati koji će se tip (integralna slika ili histogram itd.) uzorka koristiti. Za korištenje dodatnih tipova potrebno je ponoviti naredbu s novim tipom uzorka. Pomoću parametra `listOfFeatures` mogu se, slično kao i kod postavljanja tipova uzoraka, postaviti značajke koje će se koristiti u postupku učenja. Također, pomoću parametra `negsPerLayer` može se za svaku razinu kaskade specificirati broj negativnih uzoraka koji će se izvaditi iz slika pozadina prilikom učenja te razine. Prvi broj u liniji specificira broj razina, dok drugi broj specificira broj negativnih uzoraka za te razine. Između brojeva ne smiju biti nikakvi znakovi osim razmaka. Slično kao i kod prethodna dva parametra, ponavljanjem naredbi se može specificirati neki drugi broj negativnih uzoraka za dodatne slojeve.

Parametri za maksimalnu prihvatljivu učestalost lažno-pozitivnih detekcija i minimalnu prihvatljivu učestalost ispravnih detekcija po sloju se mogu postaviti pomoću parametar `maxFPR` odnosno `minDR`. Željena učestalost lažno-pozitivnih detekcija za cijelu kaskadu se može postaviti pomoću parametra `targetFPR`.

Primjer konfiguracijske datoteke se može vidjeti u nastavku:

```
[Main]
useTrain=0
useDetect=1

[Detection]
cascadePath = cascade
imgDir = ../sluzbeniTest
imgFileName = test.txt
imgRects = testRects.txt
resultDir = ../sluzbenaRjesenja
detectionRectAddres = detRects.txt
grouping= 1
scaleFactor = 1.2

[Training]
cascadePath = cascade
loadCascade = 1
posFileName = ucenje.txt
negFileName = pozadine.txt
posDir = ../sluzbeniPoz
negDir = ../200_pozadine
sampleTypes = integral_image
sampleTypes = histogram
listOfFeatures = haar_feature
listOfFeatures = histogram_feature
negsPerLayer= 5 2000
negsPerLayer= 5 2000
typeOfSelection=0
maxFPR = 0.5
minDR = 0.995
targetFPR = 0.000006
```

6. Rezultati i analiza

6.1. Analiza naučenih kaskada

Za evaluaciju postupka je korišteno računalo s Intelovim procesorom Q6600 (četiri jezgre takta 2.4 Ghz) i 4 GB radne memorije. U postupku učenja napravljene su četiri kaskade za detektiranje trokutastih znakova.

Prva kaskada (Haar1) napravljena je isključivo od Haarovih značajki. Pri tome se skup za učenje sastojao od 898 isječenih slika trokutastih znakova i 97 pozadina koje nisu sadržavale znakove.

Druga kaskada (hetero1) je napravljena nad identičnim skupom za učenje, pri čemu su u prvih 20 razina kaskade korišteni heterogeni klasifikatori (klasifikatori izgrađeni sa Haarovim i histogramskim značajkama). Pokazalo se da takva kaskada i dalje ima relativno veliki broj lažno-pozitivnih detekcija čija raspodjela boja ne odgovara prometnim znakovima. Zbog toga je nastavljeno učenje isključivo s histogramskim značajkama. Učenje se izvodilo sve dok nisu iscrpljeni svi primjeri u skupu pozadina, pri čemu je stvoreno pet dodatnih slojeva kaskade.

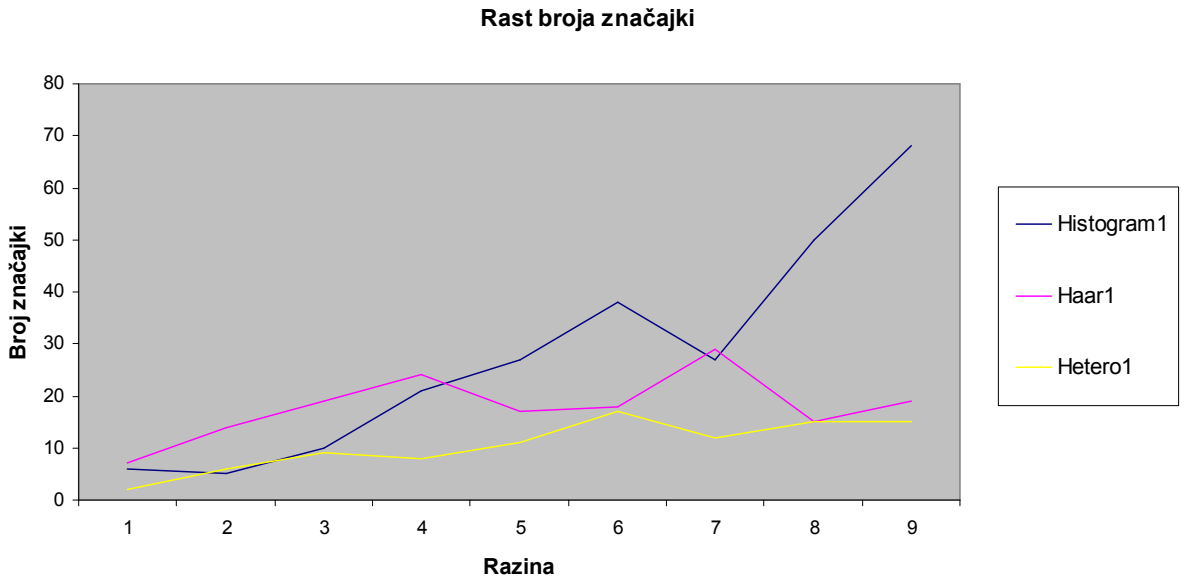
Treća kaskada (hetero2) je naučena nad skupom za učenje koji se sastoji od 2151 znakova i 178 pozadina. U prvih 20 razina kaskade koriste se heterogeni klasifikatori, nakon čega su naučene još četiri razine isključivo s histogramskim značajkama.

Po uzoru na prethodne dvije kaskade, kaskada s Haarovim značajkama je također nadograđena i tako je dobiven nova kaskada (Haar2). Ukupno je dodano šest razina koje se sastoje isključivo od histogramskih značajki. U tablici 1. možemo vidjeti raspodjelu tipa značajki u sve četiri kaskade.

Kaskada	O_Haar	O_hist	D_hist	U5	T	A
Haar1	444	0	0	81	444	22.2
Haar2	444	0	185	81	629	24.19
hetero1	237	76	281	36	594	23.76
hetero2	247	88	287	35	622	25.92

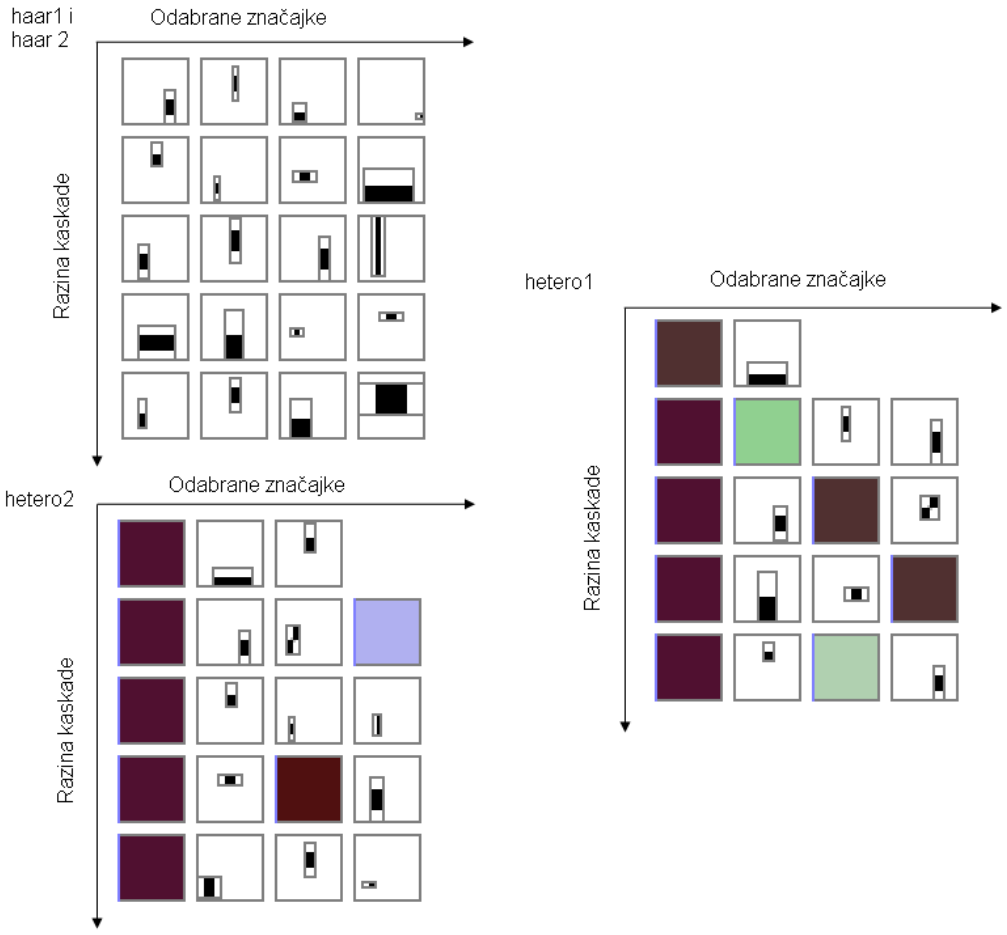
Tablica 1. O_Haar – broj Haarovih značajki bez dodatnih razina. O_hist – broj histogramskih značajki bez dodatnih razina. D_hist – histogramske značajke u dodatnim razinama. U5 – ukupan broj značajki u prvih pet razna kaskade. T – ukupan broj značajki u cijeloj kaskadi. A – prosječan broj značajki po razini.

Iz tablice 1. može se vidjeti kako se Haarove puno češće koriste nego histogramske. Također se može vidjeti kako heterogene kaskade prije dodavanja dodatnih razina imaju puno manji broj značajki. Iz toga se može zaključiti kako histogramske značajke u početku drastično olakšavaju učenje. U početnim razinama, broj histogramskih i Haarovih značajki je otprilike jednak, dok kasnijim razinama Haarove značajke postaju dominantne. Iz toga se može zaključiti kako razdvajanje uzoraka po boji brzo prestaje biti učinkovito. Kako bi se ta pretpostavka potvrdila, naučena je dodatna kaskada s isključivo histogramskim značajkama (histogram1).



Slika 11. Graf prikazuje rast broja značajki po razinama za tri kaskade (histogramsku, Haarovu i heterogenu).

Na slici 11. može se vidjeti kako broj značajki kod histogramske kaskade puno brže raste nego kod Haarove ili heterogene kaskade. Također se može vidjeti da je broj značajki u prve tri razine značajno manji nego broj značajki kod Haarove kaskade. To pokazuje kako se bojom može razdvojiti većina broj prometnih znakova od uzoraka koji to nisu. Međutim, čak i ako kaskada ispravno razvrstava 80% uzoraka, preostalih 20% na jednoj slici predstavlja nekoliko desetaka tisuća detekcija. Kako bi se razdvojili i ti uzorci, stvaraju se dodatne razine kaskade, pri čemu razdvajanje uzoraka koristeći boju postaje sve manje učinkovito. Taj problem postaje nešto lakši pri stvaranju završnih razina kaskade, jer broj negativnih uzoraka koji se mogu dobiti iz slika pozadina, postaje značajno manji.

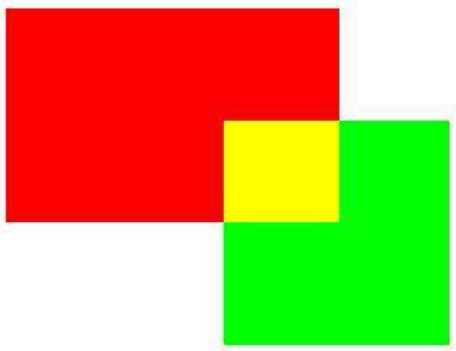


Slika 12. Na slici se mogu vidjeti najbolje četiri značajke u prvih pet razina za sve korištene kaskade. Razine kaskade su prikazane na okomitoj osi. Značajke su prikazane na horizontalnoj osi.

Na slici 12. se može vidjeti struktura značajki za sve četiri kaskade u početnim razinama (Haar1 i Haar2 imaju identične početne razine). Određen broj Haarovih značajki su odabrane tako da predstavljaju donji tj. ravni rub prometnog znaka. Preostale značajke su uglavnom dijagonalno posložene te tako mogu predstavljati bočne tj. kose rubove. Veliki broj histogramskih značajki odgovara tamno crvenoj boji koja je lako upadljiva na prometnim znakovima. Preostale značajke odgovaraju bojama (poput plave i zelene) koje se javljaju u negativnim uzorcima. Uzorci koji zadovolje takve značajke se klasificiraju kao negativni.

6.2. Evaluacija postupka detekcije za naučene kaskade

U postupku detekcije korišten je skup od 1037 slika na kojima se nalaze trokutasti znakovi. Slike su izvađene iz video sekvence koja je snimljena s kamerom ugrađenom na prometnom vozilu. Većina znakova se nalazi na četiri slike. Na prvoj slici je znak snimljen u daljini, na drugoj je snimljen na manjoj udaljenosti. Na trećoj slici je ta udaljenost još više smanjena, dok se na četvrtoj slici znak nalazi neposredno ispred vozila. Podaci o položaju i veličini znakova su zapisani u odgovarajućoj datoteci. Vrlo rijetko se prijavljena detekcija savršeno preklapa s podacima o znaku. Stoga je potrebno izračunati površinu preklapanja znaka i detekcije. Na temelju te površine se detekcija klasificira kao ispravna ili lažno-pozitivna detekcija.



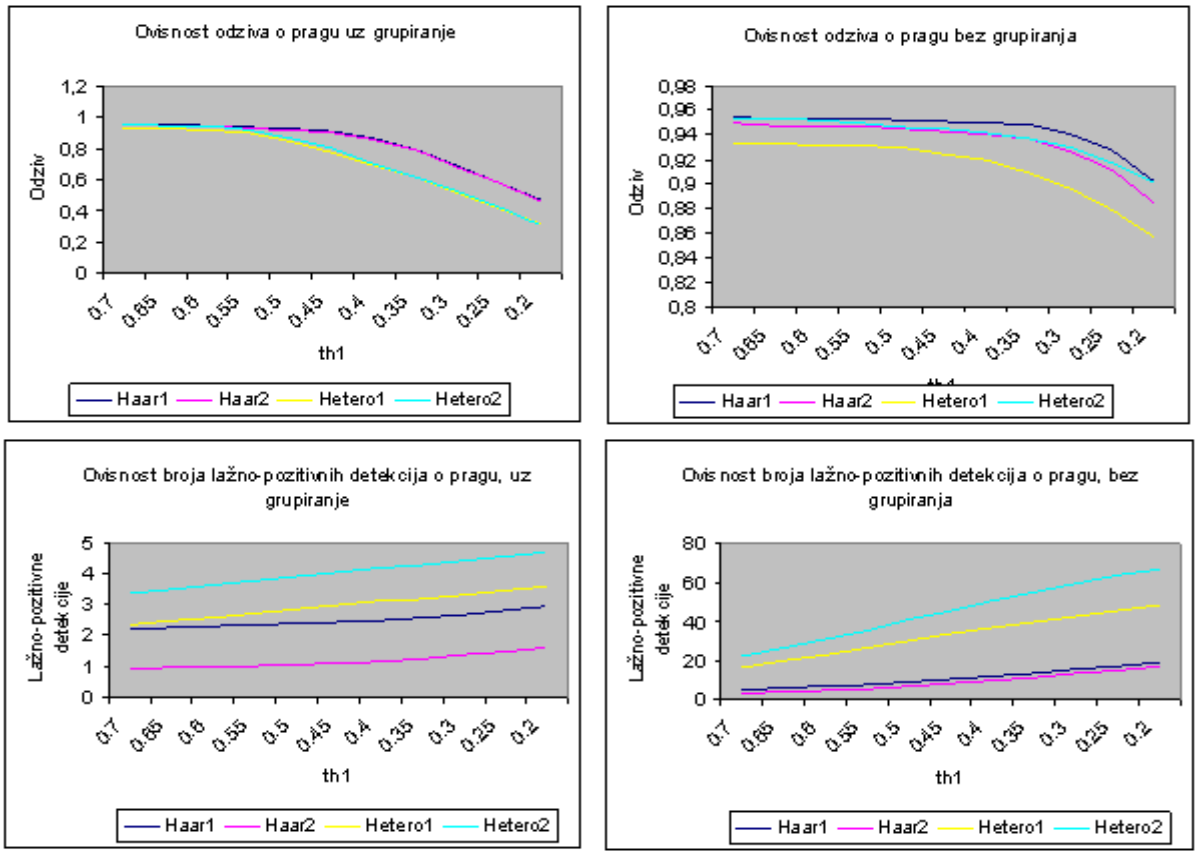
Slika 13. Na slici je prikazana mogući odnos između traženog objekta na slici i prijavljene detekcije. Crveni pravokutnik predstavlja traženi objekt, a zeleni pravokutnik detekciju. Tada žuti pravokutnik predstavlja površinu preklapanja.

Neka je a_y površina područja preklapanja znaka i detekcije. Neka je a_r površina područja prijavljene detekcije te neka je a_z površina područja označenog znaka. Prijavljena detekcija se klasificira kao pogodak ako vrijede sljedeća dva uvjeta:

$$1 - \frac{a_y}{\max(a_r, a_g)} < th1 \tag{16}$$

$$1 - \frac{a_y}{\min(a_r, a_g)} < th2 \tag{17}$$

Na slici 14. možemo vidjeti performanse kaskada za različite veličine praga *th1*. Prag *th2* je postavljen tako da izraz (17) uvijek bude zadovoljen (*th2 > 1*).



Slika 14. Prikazana su četiri grafa. Gornja dva grafa prikazuju ovisnost odziva o pragu. Donja dva grafa prikazuju kako ovisi prosječan broj lažno-pozitivnih detekcija po slici o pragu. Grafovi u lijevom stupcu su dobiveni uz korištenje algoritma za grupiranje preklapajućih detekcija. Grafovi u desnom stupcu su dobiveni bez korištenja tog algoritma.

Može se vidjeti kako smanjivanjem vrijednosti praga performanse svih kaskada opadaju. Također se može vidjeti kako algoritam grupiranja drastično smanjuje preciznost prijavljenih detekcija u odnosu na označeni znak. Varijanta u kojoj se ne koristi grupiranje ima značajno bolji odziv za najstrože uvjete, ali i drastično veći broj prijavljenih i lažno-pozitivnih detekcija. Veliki broj lažno-pozitivnih detekcija se djelomično preklapaju sa označenim znakom. Upravo prilikom grupiranja takvih detekcija dolazi do pogreške kod idealne detekcije.

Pokazalo se da se za vrijednosti pragova *th1 = 0.7* i *th2 = 0.2*, u izrazima (16) i (17), statistički podaci slažu s ljudskim opažanjima pri klasifikaciji prijavljenih detekcija. Svi statistički rezultati u nastavku rada su dobiveni uz korištenje ta dva praga. U tablici 2. se mogu vidjeti statističke podatke o broju pogodaka, promašaja i lažno-pozitivnih detekcija za sve tri kaskade.

Kaskada	R	O	T	M	P	F	TF
Haar1	3644	1261	1199	62	497	1833	2330
Haar2	2264	1261	1194	67	417	536	953
Hetero1	4080	1261	1172	89	1589	1066	2655
Hetero2	5218	1261	1199	62	2067	1659	3726

Tablica 2. Rezultati za tri kaskade na skupu za detekciju. R – sve prijavljene detekcije. O – ukupan broj označenih trokutastih znakova u skupu. T – ispravne detekcije. P – lažno pozitivne detekcije koje se djelomično preklapaju sa znakom. F – lažno-pozitivne detekcije koje se ne preklapaju sa znakom. TF – sve lažno-pozitivne detekcije. M – promašeni znakovi.

Iz podataka se može vidjeti kako Haarova kaskada prijavljuje manji broj detekcija od heterogenih kaskada. Iako histogramske kaskade imaju veći broj lažno-pozitivnih detekcija, velik broj tih pogrešnih detekcija se preklapa s označenim znakom. Iz toga možemo zaključiti kako postupak lokalizacije odnosno grupiranja prijavljenih detekcija nije idealan. S idealnim postupkom bi performanse heterogene kaskade bile jednake ili nešto bolje Haarovoj kaskadi.

Prednost heterogenog klasifikatora je što može klasificirati uzorak i na temelju boje (histogram) i oblika (Haarova značajka). Međutim, upravo to je i njegov najveći nedostatak. Naime kod heterogenih klasifikatora se nije potrebno u potpunosti zadovoljiti kriterije o boji ili obliku, već se može napraviti kombinacija. Ponekad ta kombinacija ne odgovara prometnom znaku. Na slici 15. možemo vidjeti jedan takav slučaj.

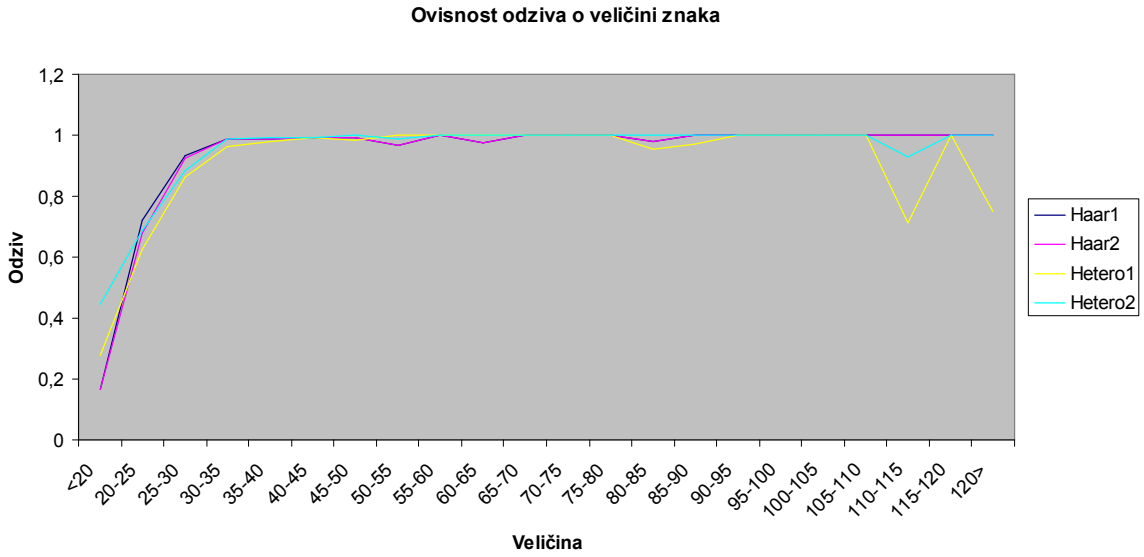
Slika 15. Primjer lažno-pozitivnih detekcija dobivenih korištenjem heterogene kaskade (hetero1) bez dodatnih razina.

Slika 16. Primjer lažno-pozitivnih detekcija dobivenih korištenjem heterogene kaskade (hetero1) s dodatnim razinama.

Lažno-pozitivne detekcije u krošnji drveta ne odgovaraju niti oblikom niti raspodjelom boja traženom prometnom znaku. Na slici 16. možemo vidjeti da se korištenjem dodatnih homogenih (histogramskih) razina klasifikatora, taj problem može riješiti. Upravo zato najbolje performanse postiže Haarova kaskada s dodatnim razinama histogramskih značajka. Takva kaskada koristi informaciju o obliku i o boji, a da pritom ne dolazi do „razvodnjavanja“ istih.

Na slici 17. može se vidjeti ovisnost odziva o veličini znaka za sve četiri kaskade. Može se primijetiti kako se mali znakovi puno rjeđe detektiraju. To je i očekivano jer velika većina znakova u skupu za učenje imaju veličinu u rasponu od 30 do 90. Znakovi s veličinom u tom rasponu se gotovo savršeno detektiraju. Oscilacije u odzivu kod velikih znakova se javljaju zato što je broj takvih znakova jako mali u skupu za evaluaciju, pa svaki promašaj značajno utječe na performanse odziva u

tom rasponu veličine znakova. Veličina znakova je izračunata kao prosjek visine i širine označenog znaka. S obzirom da se znakovi nalazi na četiri slike snimljene s različite udaljenosti, može se zaključiti da će u velikoj većini slučajeva bar na jednoj slici biti ispravno detektirani.



Slika 17. Graf koji prikazuje ovisnost odziva o veličini znaka. Veličina je izračunata kao prosjek visine i širine znaka.

U tablici 3. možemo vidjeti prosječno vrijeme izvođenja za svaku kaskadu. Kako bi se vrijeme izvođenja ubrzalo, postupak detekcije je paraleliziran s openMP alatom koji je dostupan u razvojnom okruženju Visual Studio.

Ime kaskade	**Prosječno vrijeme izvođenja (s)**	**Izračun integralne slike (ms)**	**Izračun integralnog histograma (s)**
Haar1	0.54	9.55	0.0
Haar2	0.67	10.08	0.0
Hetero1	3.92	9.62	3.28
Hetero2	3.94	9.67	3.29

Tablica 3. Prikazano je prosječno vrijeme izvođenja po slici. U prvom stupcu je ukupno vrijeme potrebno da se obradi jedna slika. U druga dva stupca se može vidjeti koliko od tog vremena otpada na vrijeme potrebno za izračun integralne slike odnosno integralnog histograma.

Iz podataka možemo vidjeti da su kaskade s heterogenim klasifikatorima puno sporije od druge dvije kaskade, ponajviše zbog sporog izračuna integralnog histogram. Za kaskadu Haar2 nije potrebno računati integralni histogram zbog toga što je ona dobivena nadograđivanjem kaskade Haar1. Iz tablice 2. možemo vidjeti kako kaskada Haar1 prosječno prijavljuje otprilike 3.5 detekcija po slici. Upravo je to i brojka koliko puta se mora izračunati histogram okna. Iako se histogram računa mali broj puta svejedno usporenje kaskade Haar2 za iznosi 24% u odnosu na Haar1.

6.3. Primjeri ispravnog pronalaženja

Svi primjeri u nastavku rada su dobiveni korištenjem kaskade Haar2. Na slici 18. može se vidjeti kako su oba znaka uspješno detektirana. U oba slučaja detekcije se skoro savršeno preklapaju sa područjem znaka

Slika 18. Primjer uspješne detekcije znaka

Na slici 19. možemo vidjeti primjer dvije uspješne detekcije. U slučaju gornjeg znaka, preklapanje detekcije i znaka nije idealno, ali ipak dovoljno da se detekcija klasificira kao ispravna.

Slika 19. Primjer uspješne detekcije znaka

Na slici 20. mogu se vidjeti isti znakovi kao i na slici 19. Ova slika je obrađena bez korištenja algoritma grupiranja. Ogroman broj prijavljenih detekcija se nalazi na gornjem znaku, od čega ih je većina fokusirana u unutrašnjosti znaka. Zbog toga, prilikom grupiranja dolazi do pomaka detekcije, koja više ne odgovara idealno traženom znaku, kao što se može vidjeti na prethodnoj slici. Do pomaka idealne detekcije najčešće dolazi kod velikih znakova. Razlog tome je što je takve znakove lakše detektirati, stoga kaskada prijavljuje velik broj djelomičnih detekcija.

Slika 20. Primjer uspješne detekcije znaka bez grupiranja

Na slici 21. se može vidjeti prometni znak u udaljenosti koji je skoro savršeno detektiran. Zbog toga što je znak male veličine, nema dovoljno djelomičnih detekcija koje bi negativno utjecale na grupiranje.

Slika 21. Primjer uspješne detekcije znaka

6.4. Lažno-pozitivne detekcije

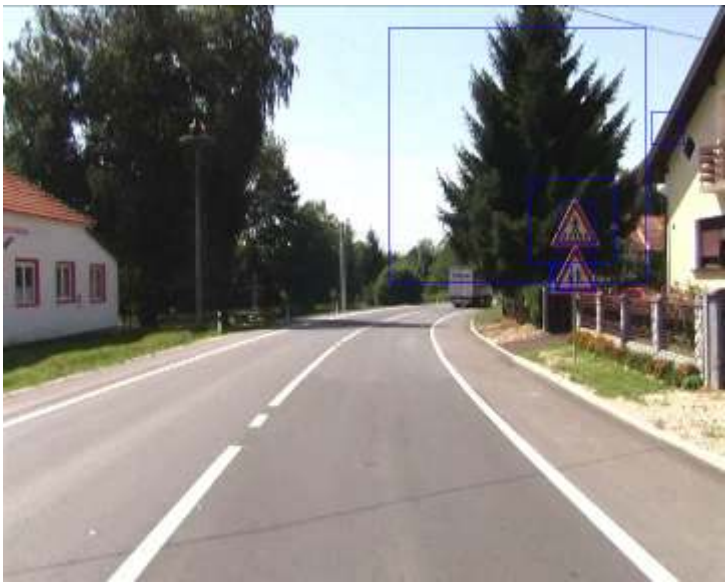
Na slici 22. može se vidjeti ispravno detektiran prometni znak. Također u donjem lijevom i donjem desnom uglu znaka, prijavljene su dvije male djelomične detekcije. Iako se sve tri detekcije preklapaju, algoritam grupiranja ih nije spojio u jednu zajedničku.

Slika 22. Primjer pogrešne detekcije

Na slici 23. mogu se vidjeti pogrešne detekcije u području krovova kuća. Takve detekcije su najčešće jer krovovi kuća bojom i oblikom odgovaraju bočnim bridovima prometnog znaka. Vanjski zidovi na kućama su obično svjetlije boje što odgovara bijeloj ili žutoj unutrašnjosti prometnog znaka. Prozori na kućama mogu oblikom odgovarati samoj oznaci na znaku.

Slika 23. Primjer pogrešne detekcije

Na slici 24. može se vidjeti pogrešna detekcija na stablu pored ceste. Detekcija također obuhvaća prometne znakove i krov kuće u pozadini. Upravo zbog toga distribucija boje može odgovarati prometnom znaku. Sam oblik stabla donekle nalikuje uskom trokutu, zbog čega dolazi do pogrešne detekcije.



Slika 24. Primjer pogrešne detekcije

6.5. Primjeri propuštenih detekcija

Na slici 25. mogu se vidjeti dva prometna znaka. Donji od ta dva znaka je uspješno detektiran dok je gornji promašen. Male znakove je značajno teže uspješno detektirati, kao što je pokazano s grafom na slici 17.

Slika 25. Primjer propuštene detekcije znaka

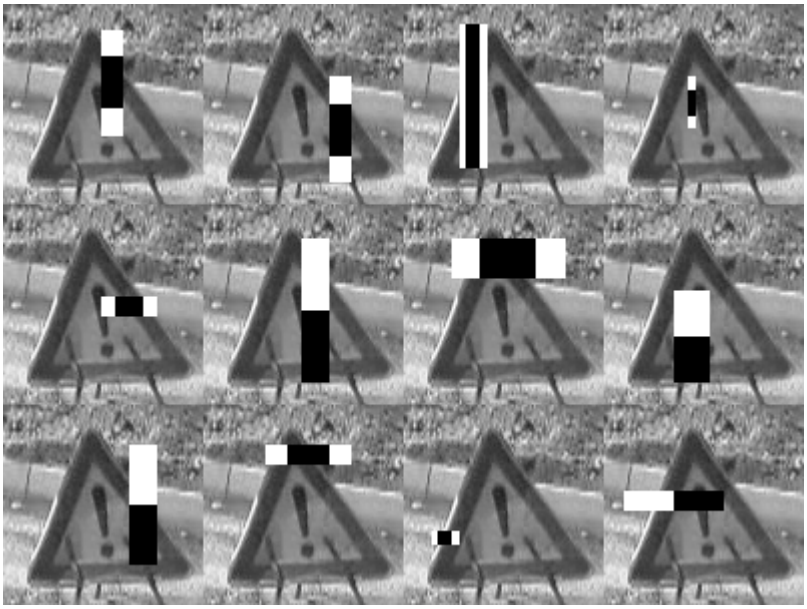
Na slici 26. može se vidjeti prometni znak koji nije uspješno detektiran. Znak ima žutu unutrašnjost, te su takvi znakovi jako rijetki u skupu za učenje. Također zbog žute unutrašnjosti razlika u kontrastu između ruba i unutrašnjosti je jako mala.

Slika 26. Primjer propuštene detekcije znaka

Na slici 27. može se vidjeti primjer propuštene detekcije prometnog znaka. Znak je snimljen u blizini kamere, u relativno idealnim uvjetima osvjetljenja.

Slika 27. Primjer propuštene detekcije znaka

Znak je odbačen u trećoj razini kaskade, pri čemu su negativan odgovor dala 12 od 19 slabih klasifikatora. Na slici 28. se mogu vidjeti značajke za koje se dobiva negativan odgovor. Određen broj značajki daje krivi odgovor jer im ne odgovara središnja oznaka na znaku. Također treba primijetiti kako je znak blago zarotiran. Zbog te rotacije također bočne stranice znaka ne odgovaraju savršeno značajkama, stoga dolazi do pogreške.



Slika 28. Značajke na propuštenom znaku sa slike 27.

7. Zaključak

Cilj ovog rada je bio iskoristiti informaciju o boji u sklopu metode koju su razvili P. Viola i M. Jones. Originalna metoda koristi Haarove značajke koje se mogu jako brzo izračunati pomoću integralne slike, pri čemu se koristi siva slika. Kako bi se informacija o boji iskoristila dodana je nova histogramska značajka, temeljena na histogramu boje. Također je implementiran integralni histogram koji omogućuje brzo izračunavanje histograma unutar slike.

U sklopu rada su naučene četiri kaskade. Prva kaskada je izgrađena koristeći isključivo Haarove značajke. Primjetno je da informacija o boji može značajno olakšati učenje, zbog čega kaskade s heterogenim klasifikatorima u početnim razinama imaju puno manji broj značajki od Haarove kaskade. Međutim, performanse Haarove kaskade su vidljivo bolje od heterogenih kaskada, čak i uz manju veličinu skupa za učenje. Također Haarova kaskada ima značajno brže vrijeme izvođenje jer se ne treba računati integralni histogram.

Najbolje performanse su dobivene s Haarovom kaskadom koja je nadograđena s šest dodatnih razina histogramskih klasifikatora. Takva kaskada ima najmanji broj pogrešnih detekcija, a odziv je tek nešto manji od obične Haarove kaskade. Još jedna prednost takve kaskade je što se može dobiti značajno brže vrijeme izvođenja od heterogene kaskade. Naime, histogramski klasifikatori se nalaze u posljednjim razinama i iskoristit će se vrlo mali broj puta (otprilike 3.5 puta po slici). Puno je brže izračunati histograme za te pozicije detekcijskog okna, nego izgraditi cijeli integralni histogram.

Velik broj prijavljenih detekcija za sve kaskade, nema idealno preklapanje s znakom na slici. Razlog tome je što algoritam za grupiranje bliskih detekcija stvara novu detekciju koja odgovara njihovom prosjeku. Korištenjem naprednijih metoda strojnog učenja, može se dobiti značajno poboljšanje [7].

8. Literatura

[1] Viola P., Jones M., Robust Real-Time Face Detection, International Journal of Computer Vision 57(2), 137–154, http://lear.inrialpes.fr/people/triggs/student/vj/viola-ijcv04.pdf, 2004.

[2] Freund Y., Schapire R., A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, no. 55., 1997.

[3] Babić T., Programska implementacija traženja objekata kaskadom boostanih Haarovih klasifikatora, Završni rad, Fakultet elektrotehnike i računarstva, Zagreb, http://www.zemris.fer.hr/~ssegvic/project/pubs/babic09bs.pdf, 2009.

[4] Babić T., Primjena histograma boje u računalnom vidu, Diplomski seminar, Fakultetet elektrotehnike i računarstva, Zagreb, http://www.zemris.fer.hr/~ssegvic/project/pubs/babic10sem2.pdf, 2010.

[5] Color Histogram – Wikipedia, http://en.wikipedia.org/wiki/Color_histogram, 2007.

[6] Porikli F., Integral histogram: a fast way to extract histograms in cartesian space, s Interneta, http://www.merl.com/papers/docs/TR2005-057.pdf, 2005.

[7] Bonači I. et al., Addressing false alarms and localization inaccuracy in traffic sign detection and recognition, CVWW, Mitterberg, Austria, http://www.zemris.fer.hr/~ssegvic/pubs/bonaci11cvww.pdf, Siječanj 2011

[8] Brkić K.. Adaboost, tehnički izvještaj, http://www.zemris.fer.hr/~ssegvic/mastif/pubs/brkic08AdaBoost.pdf, 17.12.2008.

[9] Laptev, I., Improving object detection with boosted histograms, Image and Vision Computing 27 pp. 535–544, http://www.irisa.fr/vista/Papers/2009_ivc_laptev.pdf, 2009

Detekcija objekata naučenim značajkama histograma boje

Sažetak

U sklopu ovog rada razmatra se unaprjeđenje algoritma Viole i Jonesa dodavanjem histograma boje kao značajke. U radu su opisane Haarove i histogramske značajke, kao i integralna slika i integralni histogram koji omogućuju brz izračun tih značajki. Opisan je i implementiran algoritam AdaBoost te način stvaranja kaskade boostanih klasifikatora.

U radu je napravljena analiza postupka detekcije za kaskadu s isključivo Haarovim značajkama, kao i za kaskadu koja koristi homogene klasifikatore bazirane na Haarovim ili histogramkim značajkama. Također su analizirane dvije kaskade s heterogenim klasifikatorima (koji su izgrađeni kao kombinacija Haarovih i histogramskih značajki).

Ključne riječi:

Računalni vid, prometni znakovi, detekcija objekata, Haarove značajke, histogramske značajke, histogram boje, integralna slika, integralni histogram, AdaBoost, boostani klasifikatori, slabi klasifkatori, kaskada klasifikatora, strojno učenje, algoritam Viole i Jonesa.

Object detection using trained color histogram features

Abstract

This paper considers using color histogram features as means of improving algorithm by Viola and Jones. Haar and histogram features are described, as well as concepts of integral image and integral histogram as means of fast calculations of those features. As a part of this paper, AdaBoost algorithm and the process of the creation of a cascade of boosted classifiers were described and implemented.

This paper analyzes the detection process for a cascade based exclusively on Haar features and a cascade that uses homogeneous classifiers base either on Haar or Histogram features. Additionally, two cascades that use heterogeneous classifiers (based on both Haar and Histogram features) were trained and analyzed.

Key words:

Computer vision, traffic signs, object detection, Haar features, histogram features, color histogram, integral image, integral histogram, AdaBoost, boosted classifier, weak classifier, cascade of classifiers, machine learning, Viola and Jones algorithm.

Privitak

Uz rad je priložen DVD-ROM s izvornim kodom te izvršnom datotekom. Uz to, priloženi su skupovi slika za učenje i testiranje.