SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1546

Programska implementacija učenja kaskade boostanih Haarovih klasifikatora

Mateja Čuljak

Zagreb, srpanj 2010.

Sadržaj

1. Uvod

Računalni vid je područje računarstva koje se bavi ekstrakcijom korisnih informacija iz slika ili video sekvenci, te sustavima koji takvu ekstrakciju omogućavaju. Računalni vid je usko povezan s drugim područjima u računarstvu, kao npr. umjetnom inteligencijom, strojnim učenjem, robotikom, raspoznavanjem uzoraka, obradom signala, obradom slika itd. Važnije teme koje istražuje računalni vid su pronalaženje dvodimenzionalnih ili trodimenzionalnih objekata, raspoznavanje objekata, detekcija događaja, rekonstrukcija scene, restauracija slika, praćenje pokreta itd. Pronalaženje (engl. *detection*) objekata u računalnom vidu se bavi problemom nalaženja objekata koji pripadaju nekoj zadanoj klasi u slici ili video sekvenci. Dok je ljudima vrlo lako "intuitivno" detektirati objekt neovisno o njegovoj veličini, kutu gledanja, mogućoj rotaciji ili djelomičnoj zaklonjenosti, računalima to predstavlja problem, te taj problem treba predstaviti nekim algoritmom da bi on računalu bio shvatljiv. Najčešće se pronalaženje objekata primjenjuje za pronalaženje ljudskih lica, pješaka, nekih zadanih znakova itd. Postoji razlika između detekcije objekata i raspoznavanja objekata. Raspoznavanje objekata (engl. *recognition*) bavi se problemom određivanja koji iz niza zadanih objekata različitih klasa se nalazi na slici. Za razliku od toga, detekcija objekata se bavi određivanjem nalazi li se na slici objekt isključivo jedne klase. Postoji više metoda kojima je moguće ostvariti učinkovite sustave za detekciju objekata.

Zbog razvitka digitalnih kamera visoke rezolucije nastala je potreba za algoritmima za što bržu obradu sve masivnijih slika. Paul Viola i Michael Jones su predložili izuzetno brzu metodu detekcije objekata koja se temelji na kaskadi boostanih Haarovih klasifikatora. Izgradnja kaskade je određena većim brojem parametara (primjerice, tip algoritma boostanja, količina pozitivnih i negativnih primjera, korišteni tipovi Haarovih klasifikatora i sl.). Radi postizanja što veće učinkovitosti sustava, u interesu nam je ispitati različite parametre. Dostupne implementacije otvorenog koda (engl. *open source*) ovise o velikim pozadinskim

bibliotekama te je za ispitivanje i mijenjanje nekih parametara (primjerice, tipova Haarovih značajki) potrebno mijenjati i ponovo prevoditi glomazni izvorni kod. Ideja je izgraditi lako izmjenjivu i nadogradivu implementaciju neovisnu o vanjskim bibliotekama. Implementacija učenja kaskade u okviru ovog rada je izgrađena korištenjem biblioteke *OpenCV*, tj. izdvajanjem funkcionalnosti implementacije samog učenja kaskade te uklanjanjem ovisnosti o ostatku biblioteke. Poglavlje "Slična rješenja" opisuje nekoliko poznatih implementacija Viola Jones algoritma s naglaskom na učenje kaskade. Nakon toga slijedi poglavlje s opisom metode Viole i Jonesa s osvrtom na izgradnju klasifikatora i sam algoritam učenja kaskade – *AdaBoost* algoritam. Slijede poglavlja koja opisuju implementaciju napravljenu u sklopu ovog rada te evaluaciju modificiranog učenja implementiranog unutar *OpenCV* biblioteke. Rad završava sa zaključkom.

2. Slična rješenja

Najčešće korištena implementacija učenja kaskade boostanih Haarovih klasifikatora se nalazi u OpenCV programskoj biblioteci koja je opisana u 4. odjeljku. Većina drugih rješenja bazira se na izmjenama implementacije iz OpenCV biblioteke. Primjerice, Allusse et al. (2008) su razvili "*GpuCV: GPU-accelerated Computer Vision*." Radi se o implementaciji koja je napravljena po uzoru na OpenCV biblioteku, ali je prilagođena izvršavanju na grafičkim procesorima.

Slična prilagodba je opisana u (Mun, 2009a). Navedeni su rezultati prilagodbe OpenCV algoritama za izvođenje na procesorima bez jedinice za računanje s pomičnim zarezom (engl. *Floating Point Unit,* FPU). Primjer su ARM procesori koji su izuzetno često korišteni u ugradbenim računalima.

U (Mun, 2009b) opisan je postupak iskorištavanja *MPI* (*Message Passing Interface*) tehnologije za izradu raspodijeljenog postupka učenja kaskade. Navodi se konfiguracija učenja za koju je na jednom osobnom računalu bilo potrebno 6 dana (te 4 dana korištenjem *OpenMP* implementacije OpenCV-a), a raspodjelom računanja na grozd (engl. *cluster*) od 11 povezanih osobnih računala, učenje kaskade je trajalo samo 21 sat.

3. Opis metode Viole i Jonesa

Metoda koju su predložili Paul Viola i Michael Jones je jedna od metoda pronalaženja objekata koja koristi strojno učenje, te ujedno i prva koja postiže dobre rezultate u realnom vremenu (Viola i Jones, 2002). Primarna svrha joj je detekcija lica, no može biti naučena za pronalaženje bilo koje vrste objekata, budući da koristi algoritme strojnog učenja.

Algoritmi strojnog učenja korišteni u detekciji objekata uzimaju podatke o slikama koje su već prethodno obrađene i sasvim sigurno znamo sadrže li ili ne taj objekt, te na temelju naučenog znanja uspješno detektiraju objekte na novom skupu slika. Važna prednost metoda temeljenih na strojnom učenju je mogućnost primjene na razne klase objekata, samo je potrebno prikupiti podatke za učenje. Tako isti sustav možemo primijeniti za pronalaženje lica, pronalaženje prometnih znakova ili bilo koju drugu klasu objekata. Postoji više algoritama za gradnju klasifikatora kojima klasificiramo podatke. Klasifikatorski sustavi su sustavi temeljeni na skupu pravila (klasifikatora) koja određuju reakciju sustava na uvjete dane u njegovoj okolini. Ta pravila se uče na temelju primjera koji se nalaze u skupu za učenje. Detekciju objekata tako možemo gledati kao zadaću gradnje klasifikatora, pri čemu se slika ili video sekvenca na kojoj tražimo objekte, razdvaja u više manjih podprozora, i odluka se donosi za svaki podprozor na temelju toga sadrži li on traženi objekt ili ne. Podprozori su uglavnom fiksnih dimenzija (24×24 piksela najčešće). Budući da se to pokazalo kao proces s velikom složenošću, umjesto direktnog rada s pikselima ulazne slike, ove metode rade sa određenim skupom drukčijih značajki te slike. Sustavi bazirani na tim značajkama će raditi znatno brže nego sustavi bazirani na pikselima. Vrijednosti dobivene takvim računom predstavljaju prisustvo ili odsustvo nekih karakteristika slike (kao što su rubovi, kutevi i sl.).

Metoda Viole i Jonesa vrlo brzo procesira slike i postiže visoki broj detekcija, te je brzo nakon objavljivanja postavila standarde u području detekcije lica. U usporedbi s ranijim metodama, metoda Viole i Jonesa je učinkovitija najviše zbog

brzine pronalaženja (na računalu 700MHz Intel Pentium 3 postignuta je detekcija od 15 slika u sekundi, na slikama veličine 384×288 piksela (Viola i Jones, 2002)). Tako veliku brzinu ova metoda postiže među ostalim zbog rada s informacijama koje se nalaze u samom crno–bijelom području slike. Moguće je kombinirati nekoliko različitih metoda, te postići još veću učinkovitost u pronalaženju objekata. Ulazni parametar za ovaj algoritam je slika bilo koje vrste, a izlazni parametar je lista pozicija lica na slici. U nastavku su objašnjena tri glavna dijela od kojih se metoda sastoji.

3.1. Tri glavna doprinosa metode Viole i Jonesa

Integralna slika Novi prikaz slike, koji omogućava detektoru koji koristimo da brzo izračuna značajke slike.

Gradnja klasifikatora Klasifikator je izgrađen korištenjem *AdaBoost* algoritma za strojno učenje koji izdvaja mali broj kritičnih značajki slike od vrlo velikog broja potencijalnih značajki slike.

Metoda kombinacije klasifikatora u kaskadu Omogućava da se brzo odbace suvišni predjeli slike, i detaljnije provede računanje nad predjelima slike koji sadrže mogući objekt koji tražimo, te time poboljšava performanse postupka.

3.2. Haarove značajke

Budući da se to pokazalo kao proces s velikom složenošću, metoda Viole i Jonesa, umjesto direktnog rada s pikselima ulazne slike, radi sa određenim skupom drukčijih značajki te slike. Sustavi bazirani na tim značajkama će raditi znatno brže nego sustavi bazirani na pikselima. Haarove značajke su pravokutne značajke, i računaju se kao razlika nekoliko suma piksela unutar različitih pravokutnih područja slike. Vrijednosti dobivene takvim računom predstavljaju prisustvo ili odsustvo nekih karakteristika slike (kao što su rubovi, kutevi i sl.).

Tri su tipa značajki definiranih u metodi Viole i Jonesa:

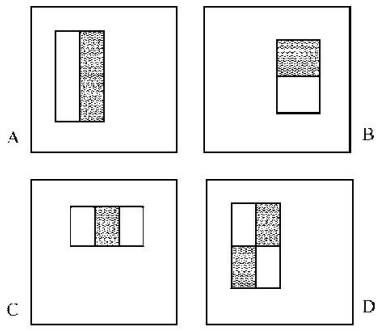
***Two-rectangle*:** Računa razliku suma piksela u 2 pravokutna predjela istog oblika i veličine koji su spojeni jednim vertikalnim ili horizontalnim bridom

***Three-rectangle*:** Uz 3 spojena pravokutnika istog oblika i veličine, računa razliku sume piksela 2 vanjska pravokutnika i sume piksela centralnog pravokutnika

***Four-rectangle*:** Uz 4 spojena pravokutnika istog oblika i veličine, računa razliku suma piksela dijagonalnih parova pravokutnika

Osnovne Haarove značajke su samo *two-rectangle* značajke, a ostale dvije vrste su u svojoj metodi dodali Viola i Jones.

Na detektoru razlučivosti 24×24 piksela, skup Haarovih značajki sadrži oko 160000 značajki, što je očigledno puno više od broja piksela slike. Tako nam Haarove značajke daju brojne informacije o slici, a zbog jednostavnog i učinkovitog načina kojim se one izračunavaju, vrlo su pogodan alat pri pronalaženju objekata.



Slika 3.1: Haarove značajke. Vrijednost značajke je razlika zbrojeva slikovnih elemenata ispod bijelih i crnih pravokutnika.

3.3. Integralna slika

Značajke predstavljene u prethodnom poglavlju možemo najbrže izračunati uvođenjem novog prikaza slike, nazvanog integralna slika. Integralna slika predstavlja transformaciju slike u matricu s jednakim brojem elemenata kao i originalna slika. Na lokaciji (x, y) integralna slika sadrži sumu piksela iznad, i s lijeve strane od x, y, tj.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

gdje je ii integralna slika, a i originalna slika.

Korištenjem sljedećih dviju jednadžbi:

$$s(x, y) = s(x, y - 1) + i(x, y)$$
$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

gdje s predstavlja ukupnu sumu jednog retka piksela, i gdje vrijedi da je $s(x, -1) = 0$ i $ii(-1, y) = 0$, integralna slika može biti izračunata u jednom prolazu kroz početnu sliku. Korištenjem integralne slike, bilo koja suma piksela pravokutnika može biti izračunata u 4 reference na polje podataka, te iz toga i činjenice da su neki bridovi 2 ili više pravokutnika zajednički, proizlazi da se *two-rectangle* značajke mogu izračunati u 6 referenci, *three-rectangle* značajke u 8 referenci, te *four-rectangle* značajke u 9 referenci.

1	1	1
1	1	1
1	1	1

(a) Ulazna slika.

1	2	3
2	4	6
3	6	9

(b) Integralna slika.

Slika 3.2: Integralna slika.

3.4. Gradnja klasifikatora strojnim učenjem

Svaku od izračunatih značajki možemo predstaviti kao jedan slabi klasifikator. Matematički, slabi klasifikator je opisan kao:

$$h_j(x) = \begin{cases} 1 & \text{ako } p_j f_j(x) < p_j \theta_j \\ 0 & \text{inače.} \end{cases}$$

gdje je x podprozor nad kojim radimo, veličine 24×24 piksela, f je korištena značajka, p je polaritet, a θ prag na temelju kojeg odlučujemo treba li x biti klasificiran kao pozitivan (sadrži objekt) ili negativan (ne sadrži objekt).

Kako je provođenje operacija nad cijelim skupom značajki također iznimno sporo (podsjetimo se, broj značajki znatno nadmašuje broj piksela), a eksperimentalno je utvrđeno da vrlo mali broj značajki može formirati klasifikator, potrebno je pronaći taj mali podskup kritičnih značajki. Metoda Viole i Jonesa koristi *boostanje* tj. varijantu *AdaBoost* algoritma strojnog učenja za odabir značajki.

Boostanje je meta-algoritam strojnog učenja, koji iz skupa slabih klasifikatora gradi jedan jaki klasifikator. Naime, slabi klasifikator nedovoljno uspješno klasificira podatke, te je zbog toga potrebno izgraditi jaki klasifikator koji bi postizao veći broj pogodaka.

3.4.1. AdaBoost algoritam

AdaBoost algoritam u originalnoj formi obavlja funkciju *boostanja* performansi jednostavnog algoritma strojnog učenja. To čini kombiniranjem skupa slabih klasifikatora tako da tvore jaki klasifikator, tj. gradi jaki klasifikator kao linearnu kombinaciju $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ slabih klasifikatora $h_t(x)$. Algoritam izgleda ovako :

- Uzmemo uzorke za učenje $(x_1, y_1), \ldots, (x_n, y_n)$ pri čemu je $y_1 = 0$ za negativne, odnosno $y_1 = 1$ za pozitivne uzorke.
- Inicijaliziramo težine $w_{1,i} = \frac{1}{2m} za y_1 = 0$ i $1_{\overline{2l}}$ za $y_1 = 1$ pri čemu su m i l broj pozitivnih, odnosno negativnih uzoraka.
- za $t = 1, \ldots, T$:
  1. Normaliziraj težine $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  2. Za svaku značajku j treniraj klasifikator h_j, te odaberi klasifikator s najmanjim postotkom grešaka ϵ_t:
  $$\epsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|$$
  3. Odredi $h_t(x) = h(x, f_t, p_t, \theta_t)$ pri čemu su f_t, p_t i θ_t minimalne vrijednosti za koje se postiže ϵ_t.
  4. Ponovo izračunaj težine:
  $$w_{t+1,i} = w_{t,i} \cdot \beta^{1-e_i}$$
  pri čemu je $e_i = 0$ ako je uzorak x_i proglašen ispravnim, a $e_i = 1$ u suprotnom; $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
- Završni jaki klasifikator je:
$$C(x) = \begin{cases} 1 & \text{ako } \sum_{t=1}^{T} \alpha_t H_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{inače.} \end{cases}$$
pri čemu je $\alpha_t = \log \frac{1}{\beta_t}$

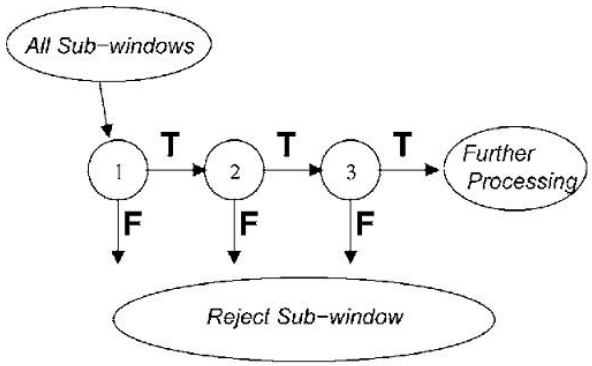
Iz gornjeg algoritma možemo iščitati kako zapravo *boostamo* slabi klasifikator. Svaki put naglašavamo one primjere koji su bili neispravno klasificirani, tako da im održavamo konstantne težine (koje se u sljedećem koraku renormaliziraju), dok primjerima koji su ispravno klasificirani smanjujemo težine. Svaki klasifikator pokušava bolje klasificirati one primjere koje je njegov prethodnik loše klasificirao. Tako tvorimo jaki klasifikator. Broj grešaka takvog klasifikatora se eksponencijalno približava nuli s povećanjem broja primjera za učenje koje smo mu zadali.

U jednom dijelu gornjeg algoritma je potrebno odrediti najbolju značajku, polaritet i prag. Postavlja se pitanje kako to uspješno i brzo riješiti. Viola i Jones su predložili jednostavnu *brute-force* metodu pretraživanja. To znači da za određivanje jednog slabog klasifikatora moramo evaluirati svaku značajku na svakom primjerku za učenje, da bi našli značajku s najboljim performansama. Ovaj postupak je vremenski najzahtjevniji dio cijele metode.

3.5. Metoda za kombiniranje klasifikatora u strukturu kaskade

Nakon što je završena gradnja klasifikatora, performanse se mogu poboljšati kombiniranjem više klasifikatora. Naime, jedan klasifikator dobiven prethodnim algoritmom učenja ne može davati rezultate u realnom vremenu, pa se klasifikatori kombiniraju u *kaskadu* poredani po kompleksnosti, tako da je svaki klasifikator kompleksniji od svog prethodnika. Prvi klasifikator u kaskadi dobiva na evaluaciju sve podprozore slike, a svaki sljedeći klasifikator evaluira samo one primjere koje prethodni klasifikator nije odbacio kao negativne (ne sadrže zadani objekt). Ako je neki podprozor u tom procesu odbačen od strane nekog klasifikatora, on se potpuno odbacuje i na njega se više ne primjenjuju nikakve operacije, i pretraživanje kreće do idućeg podprozora. Kaskada tako ima oblik degeneriranog stabla odlučivanja, kao što je prikazano na slici 3.3.

Takvim postupkom kaskada povećava brzinu cijelog detektora, jer fokusira pažnju na predjele slika koji "obećavaju", a kompleksnije procesiranje se čuva samo za te predjele (podprozore). Kada se radi o detekciji lica, prvi klasifikator u kaskadi koristi samo 2 značajke, i postiže postotak greške od 0% kada postupak nalazi objekte koje nismo tražili (false positive rate) i 40% kada postupak ne nalazi tražene objekte (false negative rate). Dakle, prvi klasifikator reducira broj



Slika 3.3: Kaskada klasifikatora.

značajki koje prolaze kroz kaskadu za gotovo jednu polovinu. Kako učinak bilo kojeg klasifikatora u kaskadi ovisi o ponašanju njegovog prethodnika, ukupan broj grešaka kada postupak nalazi objekte koji nisu traženi (false positive rate) jest:

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

Analogno tome, broj ispravnih detekcija jest:

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

Kako to poboljšava performanse, vidljivo je na primjeru: Ako kaskada koja sadrži 32 klasifikatora postiže postotak greške veličine $F = 10^{-6}$, svaki pojedini klasifikator unutar kaskade može postizati postotak greške $F = 65\%$. Međutim, svaki klasifikator mora imati vrlo visok postotak ispravnih detekcija, npr. $D = 99.7\%$ da bi cijela kaskada imala postotak detekcija veličine $D = 90\%$.

4. OpenCV biblioteka

Open Computer Vision library je *open source* biblioteka pod *BSD* licencom[1] koja sadrži preko 500 algoritama iz područja računalnog vida.

Algoritmi rade na obradi slika u stvarnom vremenu. Dostupna je na Linuxu, Windowsima i MacOS X operacijskim sustavima, a pisana u programskim jezicima *C* i *C++* te postoje programska sučelja prema *Pythonu*, *Rubyu*, *Matlabu*, i drugim jezicima.

Sadrži module za rad sa matričnim strukturama podataka, rad sa video sadržajem iz datoteka ili kamera, procesiranje i transformaciju slika, izgradnju histograma i kontura, alate za praćenje pokretnih objekata, kalibriranje kamera, izgradnju projekcije i 3D pogleda, algoritme za strojno učenje, te modul za izgradnju grafičkog sučelja (Bradski i Kaehler, 2008).

Službeni OpenCV resursi na Internetu:

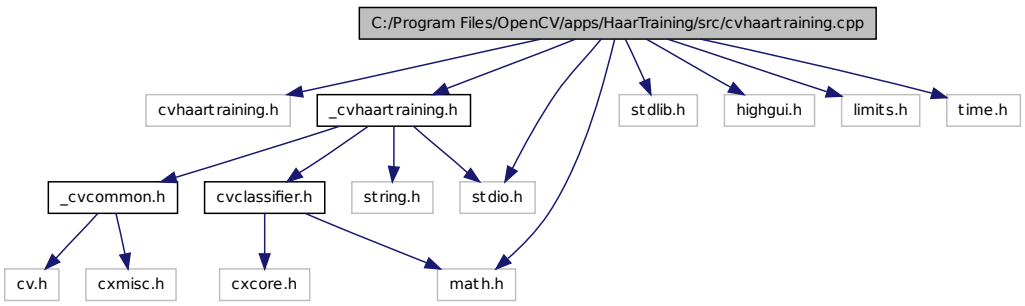
- *http://sourceforge.net/projects/opencvlibrary/*
  Izvorni kod Open Computer Vision biblioteke.
- *http://tech.groups.yahoo.com/group/OpenCV/*
  Mailing lista i grupa OpenCV zajednice.
- *http://opencv.willowgarage.com/wiki/*
  Wiki stranice OpenCV zajednice.

[1] URL: http://www.opensource.org/licenses/bsd-license.php

5. Implementacija

Implementacija je izvedena na temelju biblioteke OpenCV, odnosno izdvajanjem funkcionalnosti učenja kaskade te uklanjanjem ovisnosti o ostatku biblioteke OpenCV. Korištena je biblioteka OpenCV verzije 1.1.

Učenje kaskade unutar biblioteke OpenCV pokreće se pozivom funkcije `cvCreateTreeCascadeClassifier`. Navedena funkcija se nalazi u aplikaciji `haartraining` OpenCV-a te se njena implementacija nalazi u datoteci `cvhaartraining.cpp`. Ovisnost datoteke `cvhaartraining.cpp` o ostatku OpenCV-a može se vidjeti na slici 5.1.



Slika 5.1: Ovisnosti `cvhaartraining.cpp` datoteke.

Funkcija `cvCreateTreeCascadeClassifier` gradi stablastu kaskadu. Stablasta kaskada je Lienhartova izmjena originalne ideje Viola Jones algoritma za detekciju objekata. Rezultati evaluacije pokazuju da korištenje stablaste kaskade nema značajan utjecaj na uspješnost detekcije. Rezultati se mogu vidjeti u tablici 6.1.

Izgradnja linearnog klasifikatora također je implementirana u biblioteci OpenCV, ali je njeno korištenje napušteno. Radi značajne razlike u jednostavnosti implementacije u ovom radu je iskorištena implementacija izgradnje linearnog klasi-

fikatora. Ta izgradnja je implementirana funkcijom `cvCreateCascadeClassifier`. Razlika u složenosti izvedbi se može prikazati grafovima poziva funkcija (engl. *call graph*) koji su prikazani na slikama 5.2 i 5.3 te broju linija koda (205 linija koda za izgradnju linearnog klasifikatora te 487 linija koda za izgradnju stablastog klasifikatora). Grafovi poziva funkcija dobiveni su programom *Doxygen* (vidi: dodatak A).

Posebno zanimljiv segment implementacije unutar OpenCV-a je izgradnja Haarovih značajki. Haarove značajke se grade funkcijom `icvCreateIntHaarFeatures` s prototipom:

```
static
CvIntHaarFeatures* icvCreateIntHaarFeatures(CvSize winsize,
                                            int mode,
                                            int symmetric)
```

`winsize` – veličina podprozora (najčešće 24×24),

`mode` – tip značajki koje se grade:

- 0 (`BASIC`) – značajke prikazane na slici 5.4a,
- 1 (`CORE`) – značajke prikazane na slikama 5.4a i 5.4b,
- 2 (`ALL`) – značajke prikazane na slikama 5.4a, 5.4b i 5.4c.

`symmetric` – 1 ako su pozitivni primjeri za učenje simetrični (primjerice, lica i prometni znakovi), inače 0.

Funkcija generira skup Haarovih značajki svih veličina iz skupa $\{h \times w \mid 1 \leq h \leq H, 1 \leq w \leq W\}$, pri čemu su su H i W visina i širina podprozora, tj. veličine definirane argumentom `winsize`.

Segment koda koji generira značajku prikazanu na slici 5.5:

```
// haar_y3
if ((x+dx*3 <= winsize.height) && (y+dy <= winsize.width)){
  if (dx*3*dy < s0) continue;
  if (!symmetric || (y+y+dy <= winsize.width)) {
    haarFeature = cvHaarFeature( "haar_y3",
          y, x,    dy, dx*3, -1,
          y, x+dx, dy, dx,   +3 );
    CV_WRITE_SEQ_ELEM( haarFeature, writer );
  }
}
```

Značajka se interno reprezentira `CvTHaarFeature` strukturom:

```
typedef struct CvTHaarFeature {
  char desc[CV_HAAR_FEATURE_DESC_MAX];
  int  tilted;
  struct {
    CvRect r;
    float weight;
  } rect[CV_HAAR_FEATURE_MAX];
} CvTHaarFeature;
```

te se gradi `cvHaarFeature` funkcijom koja ima prototip:

```
CV_INLINE CvTHaarFeature cvHaarFeature(const char* desc,
           int x0, int y0, int w0, int h0, float wt0,
           int x1, int y1, int w1, int h1, float wt1,
           int x2, int y2, int w2, int h2, float wt2 )
```

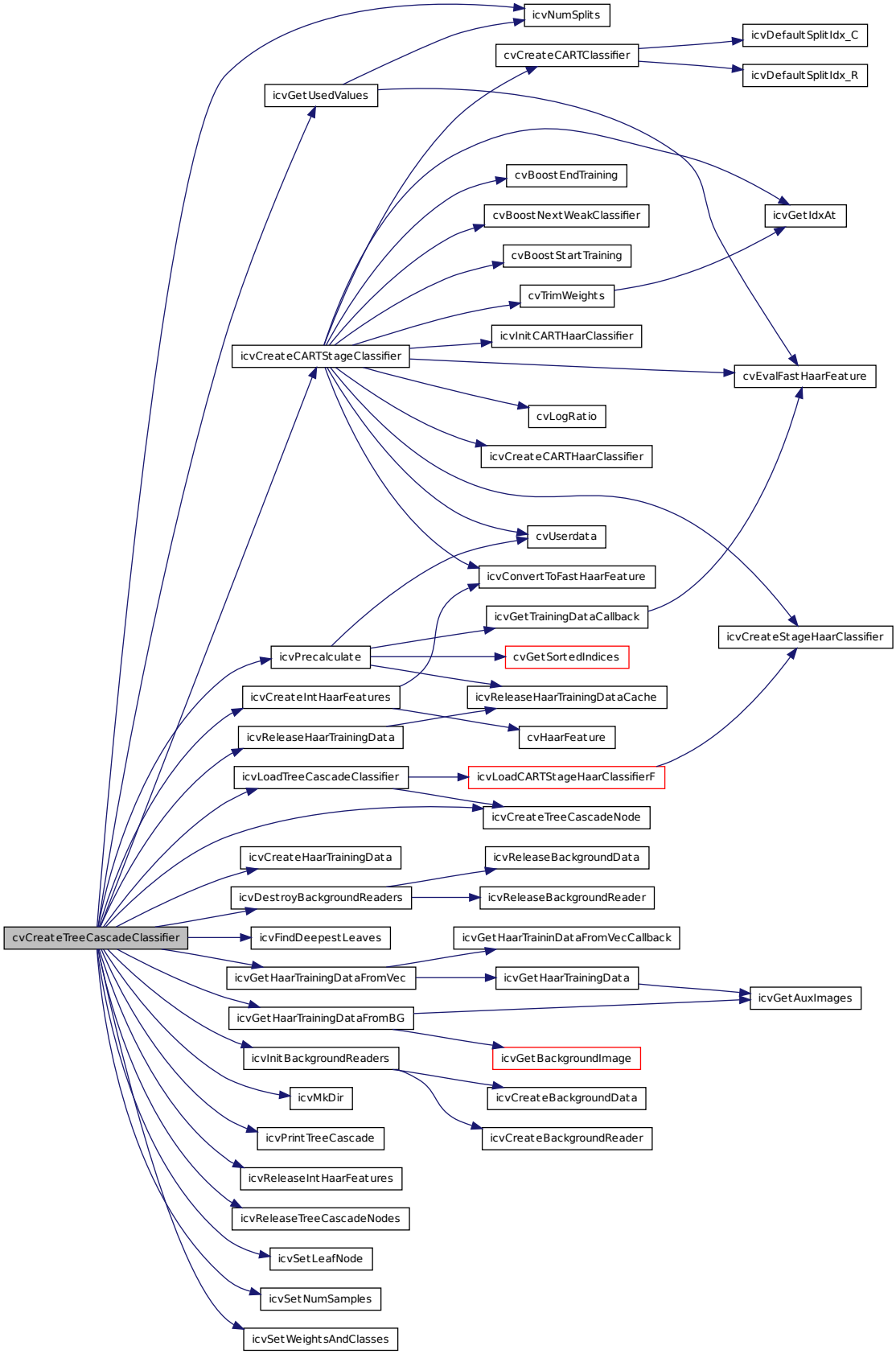
te dodatni prototip s predefiniranim vrijednostima trećeg pravokutnika postavljenim na nulu (što znači da se tada treći pravokutnik ne uzima u obzir).

Evaluacija Haarovih značajki se vrši funkcijom `cvEvalFastHaarFeature` čiji je prototip:

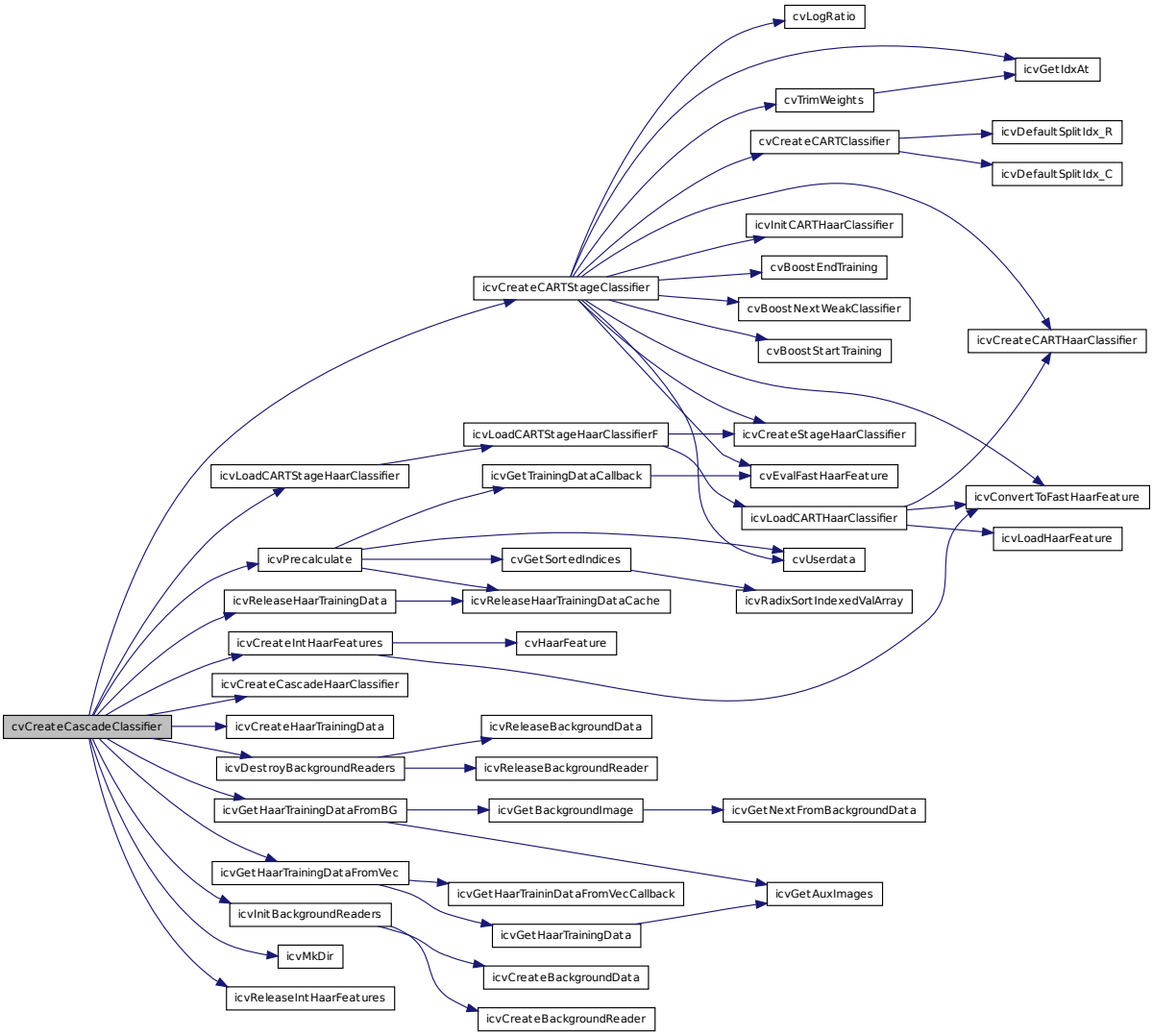
```
CV_INLINE
float cvEvalFastHaarFeature(const CvFastHaarFeature* feature,
                            const sum_type* sum,
                            const sum_type* tilted)
```

Pri evaluaciji se koristi struktura `CvFastHaarFeature`, malo optimirana verzija strukture `CvTHaarFeature` (nema opis te su vrijednosti spremljene direktno u strukturu tj. bez korištenja strukture `CvRect`).

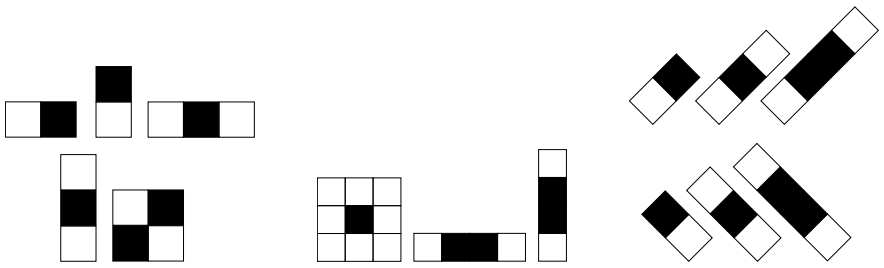
Sama evaluacija je funkcionalno vrlo jednostavna. Sastoji se od zbrajanja i oduzimanja segmenata integralne slike te množenja s pripadnim težinama značajke. S obzirom na jednostavnost reprezentacije značajki te njihove evaluacije, relativno je jednostavno modificirati implementaciju da koristi drugačije tipove značajki ili da značajke koriste dodatne informacije sa slike (primjerice, boju).



Slika 5.2: Graf poziva funkcije `cvCreateTreeCascadeClassifier`

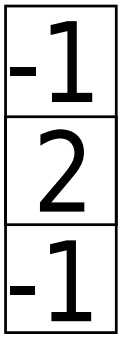


Slika 5.3: Graf poziva funkcije `cvCreateCascadeClassifier`



(a) Osnovni skup **(b)** "`CORE`" proširenje **(c)** "`ALL`" proširenje

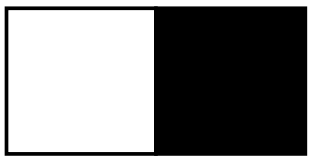
Slika 5.4: Skupovi OpenCV-ovih Haarovih značajki



Slika 5.5: OpenCV Haarova značajka "`haar_y3`" s pripadnim težinama.

6. Evaluacija

Cilj evaluacije je pokazati utjecaj tipa kaskade (linearna ili stablasta) i skupa značajki na uspješnost izgrađenog klasifikatora. Učenje je vršeno OpenCV implementacijom uz modifikaciju koda u konfiguracijama koje koriste linearnu kaskadu te skup samo "*dvostrukih*" značajki. *Dvostruke* značajke su one Haarove značajke koje se sastoje od samo dva pravokutnika. Primjer takve značajke može se vidjeti na slici 6.1.



Slika 6.1: Primjer dvostruke Haarove značajke.

Evaluacija je vršena OpenCV-ovim alatom *performance*. Primjer pokretanja:

```
opencv_performance.exe -data dvostruke \
                       -info trokuti.dat
                       -ni
```

Performance koristi XML zapis naučene kaskade (npr. `data-tree-dvostruke-.xml`) i skup slika opisan `dat` datotekom. Parametar "`ni`" isključuje označavanje dobivenih odziva na slikama za testiranje. Rezultat evaluacije je broj ispravno pronađenih objekata (*Hits*, *True positive* – TP), broj traženih objekata koji nisu prepoznati kao traženi (*Missed*, *False negative* – FN) te broj odziva koji ne obuhvaćaju traženi objekt (*False*, *False positive* – FP).

Kao mjera uspješnosti korištena je F_1 mjera. F_1 mjera se definira preko *preciznosti* (engl. *precision,* P) i *odziva* (engl. *recall,* R) (Van Rijsbergen, 1979). Preciznost se računa po formuli:

$$P = \frac{TP}{TP + FP}, \tag{6.1}$$

a odziv po:

$$R = \frac{TP}{TP + FN}. \tag{6.2}$$

F_1 mjera se definira na temelju preciznosti i odziva kao:

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}. \tag{6.3}$$

Mjera F_1 je poseban slučaj F_β mjere za $\beta = 1$. Ona daje jednak značaj preciznosti i odzivu.

$$F_\beta = \left(1 + \beta^2\right) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \tag{6.4}$$

Za učenje je korišteno 2100 pozitivnih primjera i 711 negativnih. Za testiranje je korišteno 918 slika koje se nisu koristile u postupku učenja. Rezultati evaluacije mogu se vidjeti u tablici 6.1.

Tablica 6.1: Rezultati evaluacije modifikacije učenja OpenCV Viola Jones algoritma.

Tip kaskade	Skup značajki	Preciznost [%]	Odziv [%]	F_1 mjera [%]
Linearna	samo *dvostruke*	14.43	86.91	24.75
Linearna	`BASIC`	14.43	86.91	24.75
Linearna	`CORE`	8.99	93.55	16.40
Linearna	`ALL`	31.99	82.36	46.08
Stablasta	samo *dvostruke*	16.80	90.64	28.34
Stablasta	`BASIC`	16.80	90.64	28.34
Stablasta	`CORE`	9.41	91.45	17.07
Stablasta	`ALL`	25.59	82.91	39.11

7. Zaključak

U okviru ovog rada je opisano pronalaženje objekata kaskadom boostanih Haarovih klasifikatora, implementiran je postupak učenja kaskade, te su rezultati uspoređeni s rezultatima originalnog Lienhartovog programa iz biblioteke OpenCV. Uspješnost postupka detekcije objekata kaskadom boostanih Haarovih klasifikatora u velikoj mjeri ovisi o parametrima koje zadajemo prije učenja kaskade. Postoje poboljšanja postupka koji mijenjaju učenje tako da se ono izvodi brže, te su neka od njih opisana u okviru ovog rada. Implementacija dana u ovom radu, nasuprot tome, pridonosi konfigurabilnosti postupka, kako bi isprobavanje parametara bilo što jednostavnije, te kako bi se olakšalo dobivanje boljeg klasifikatora. U daljnjem radu implementaciju je moguće ubrzati, izvorni kod učiniti što više nadogradivim, te ispitati rad programa s različitim parametrima učenja.

Literatura

Y. Allusse, P. Horain, A. Agarwal, i C. Saipriyadarshan. GpuCV: an open-source GPU-accelerated framework forimage processing and computer vision. U *Proceeding of the 16th ACM international conference on Multimedia*, stranice 1089–1092. ACM, 2008.

Gary Bradski i Adrian Kaehler. *Learning OpenCV*. O'Reilly Media, Sebastopol, 1 izdanju, 2008.

Vasiliy Mun. Fast & furious face detection with opencv. Blog, 06 2009a. `http://www.computer-vision-software.com/blog/2009/06/fast-furious-face-detection-with-opencv/`.

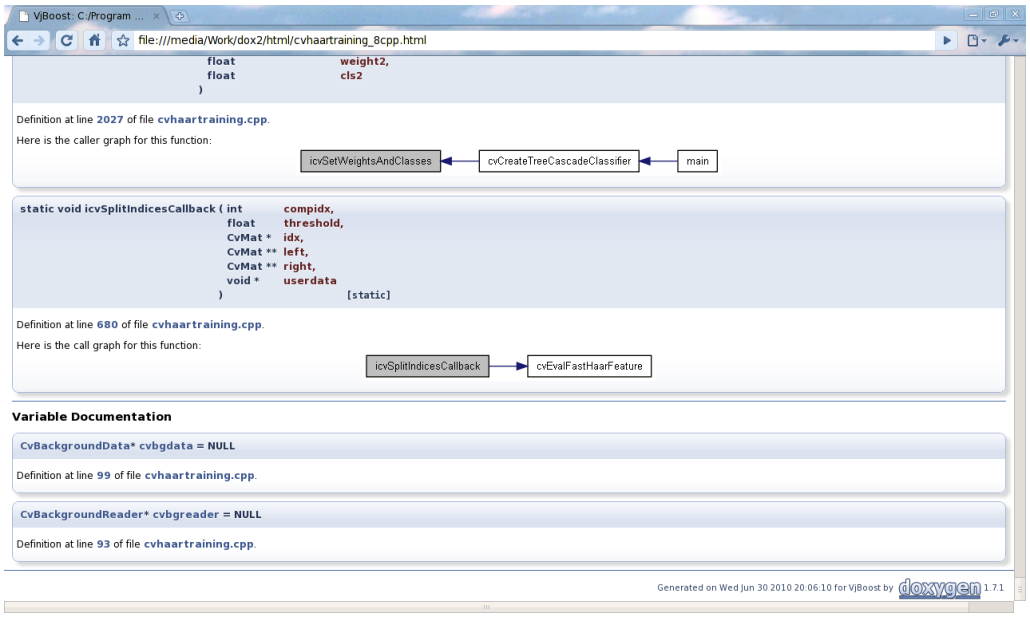
Vasiliy Mun. Parallel world of opencv (haartraining). Blog, 06 2009b. `http://www.computer-vision-software.com/blog/2009/06/parallel--world-of-opencv/`.

CJ Van Rijsbergen. *Information retrieval*, svezak 2. Butterworths, London, 1979.

P. Viola i M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2002.

Dodatak A
Doxygen

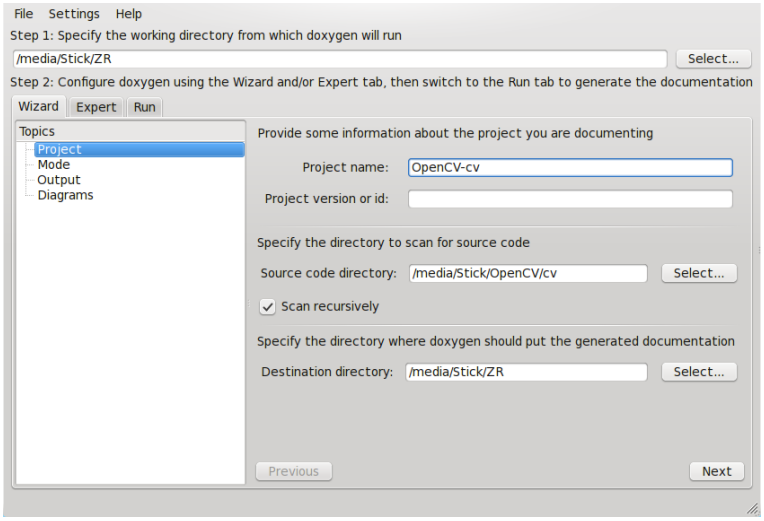
Doxygen je sustav za dokumentiranje programskog koda. Generira HTML (*Hypertext markup language*) ili LaTeX dokumentaciju na temelju programskog koda i komentara koji se u njemu nalaze. Primjer izgrađene HTML dokumentacije može se vidjeti na slici A1. Podržava programske jezike: C++, C, Javu, Objective-C, Python, IDL, Fortran, VHDL, PHP, C#, i djelomično D. Doxygen je otvorenog koda te je izdan pod GPL licencom i dostupan na većem broju platformi (Linux, MacOS X, Microsoft Windows).



Slika A1: Primjer generirane HTML dokumentacije.

Pri analizi programskog rješenja, dokumentacija nudi širi pregled od samog programskog koda. Doxygen, među ostalim, omogućava generiranje grafova poziva (engl. *call graph*) i pozivatelja (engl. *caller graph*) koji su izuzetno velika pomoć

pri analizi velikih programskih rješenja. Za iscrtavanje grafova, Doxygen koristi alat *GraphViz*, najpoznatiji alat otvorenog koda za crtanje raznih grafova i dijagrama. Podešavanje Doxygen alata moguće je napraviti kroz grafičko sučelje, Doxygen wizarda. Primjer konfiguracije Doxygen projekta može se vidjeti na slici A2. Doxygen se nalazi na adresi`http://www.stack.nl/~dimitri/doxygen/index.html`, a GraphViz na `http://www.graphviz.org/`.



Slika A2: Konfiguracija Doxygen projekta preko grafičkog sučelja.

Programska implementacija učenja kaskade boostanih Haarovih klasifikatora

Sažetak

Uspješnost detekcije objekata algoritmom Viole i Jonesa ovisi o konfiguraciji učenja kaskade boostanih Haarovih klasifikatora. S obzirom da je svaki problem detekcije specifičan radi geometrijskih svojstava objekta, potrebno je evaluirati veći broj različitih konfiguracija. Da bi ispitivanje bilo što jednostavnije i lakše za automatizaciju, potrebno je razviti konfigurabilnu implementaciju izgradnje kaskade. Trenutno najkorištenija javno dostupna implementacija učenja je ona iz biblioteke OpenCV. Nedovoljno je konfigurabilna (primjerice, nije moguće definirati nove tipove Haarovih značajki bez ponovnog prevođenja) te ovisi o velikom dijelu ostatka biblioteke (stoga je teško ugradiva u druge projekte). U radu su opisane i evaluirane pojedinosti implementacije učenja kaskade u sklopu biblioteke OpenCV te predstavljena nova implementacija učenja kaskade izgrađena korištenjem biblioteke OpenCV.

Ključne riječi: Viola Jones, računalni vid, strojno učenje, AdaBoost, OpenCV.

Software implementation of the training procedure for boosted Haar cascade classifiers

Abstract

The success of object detection algorithm of Viola and Jones depends on the configuration of cascade training. Every detection problem is unique due to the geometric properties of an object. Therefore, it is necessary to evaluate a number of different configurations. To make testing easy and to automate process, it is neccesary to develop a configurable implementation of cascade construction. Currently most used publicly available implementation of cascade learning is a part of the OpenCV library. It is poorly configurable (for example, it is not possible to define new types of Haar features without recompiling). Also, because of the dependence on the rest of OpenCV, it is difficult to embed it into the other projects. The paper describes and evaluates the details of OpenCV's implementation of cascade construction and presents new implementation based on OpenCV's implementation.

Keywords: Viola Jones, computer vision, machine learning, AdaBoost, OpenCV.