SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1492

DETEKCIJA OSMIJEHA U FOTOGRAFIJAMA

BRUNO KOVAČIĆ

Zagreb, srpanj 2010.

Sadržaj

1. Uvod

Izrazi lica su od velikog značaja u iskazivanju emocija, namjera i mišljenja osoba u komunikaciji s drugim ljudima. Smatra se da samo 7% emocionalnog značenja komuniciramo riječima. Oko 93% komuniciramo korištenjem tona glasa i neverbalnim znakovima kao što su izrazi lica, govor tijela, geste i sl[1]. Iz ovakvih podataka očigledan je značaj izraza lica u komunikaciji. Shvaćanjem tog značaja, prirodno je očekivati da će razvoj sustava koji prepoznaje izraze lica revolucionarno utjecati na naš svakodnevni život. Potencijalne primjene tehnologije prepoznavanja izraza lica uključuju sustave za učenje koji su osjetljivi na emocije svojih učenika, računalno potpomognuta detekcija laži i prijevare, dijagnostika i praćenje bolesti, nova sučelja za računalne igre, pametne digitalne kamere i socijalni roboti. Među izrazima lica, osmijeh je zasigurno jedan od najpotrebnijih za razumijevanje komunikacije.

Unatrag nekoliko godina, postignut je značajan napredak u razvoju sustava za detekciju emocija. Međutim, u većini njih još postoji prepreka za korištenje u stvarnom svijetu. Uglavnom je taj problem prisutan zbog nedostatka slika za učenje takvih klasifikatora, odnosno, nedostatka slika iz stvarnog svijeta. Jedni od najpoznatijih skupova takvih slika, DFAT set [3] i POFA set [4], sadrže slike slikane u laboratorijskim uvjetima, odnosno, lica su savršeno pozicionirana, slike su zadovoljavajuće kvalitete itd. Iz tog razloga, detektori učeni na tim setovima pokazali su znatno lošije rezultate u stvarnom svijetu od onih koje su njihovi autori prezentirali na temelju evaluacije detektora na slikama slikanim u laboratorijima. U stvarnom svijetu se moramo suočiti s problemima poput loše kvalitete slike, loše pozicije lica i razlikama u izgledu osmijeha među različitim skupinama ljudi. U obzir također treba uzeti da rasne razlike mogu uvelike utjecati na ispravnost rada detektora. Kako bi uspjeli riješiti navedene probleme, autori članka [1] napravili su set GENKI, koji sadrži 4000 slika osoba s pripadajućim oznakama u slučaju da se osoba na slici smije ili ne. Set GENKI se razlikuje od DFAT i POFA seta po tome što su slike sakupljene preko interneta s različitih web stranica na kojima se ljudi nalaze u prirodnim okruženjima i situacijama. Upravo iz tog razloga, autori članka [1], uspjeli su kreirati detektor koji ima značajno bolje rezultate od prethodnika u detekciji osmijeha u stvarnom svijetu.

U ovom radu, cilj mi je bio istražiti načine detekcije osmijeha, prvenstveno usmjerene za korištenje u digitalnim kamerama. Na tržištu već postoje digitalne kamere koje imaju opciju slikanja u trenutku kada detektiraju da se osobe na slici najjače smiju.

2. Set podataka

2.1. Izvor podataka

Kao izvor slika za učenje detektora osmijeha, koristio sam GENKI set slika, objavljen od strane autora [1]. GENKI set ukupno sadrži 63000 slika sakupljenih s raznih javnih repozitorija slika na internetu i privatnih web stranica. Set sadrži širok spektar mogućih slika u stvarnome svijetu, slikanih u zatvorenom i otvorenom prostoru, u svjetlijim i tamnijim uvjetima, različitih spolova, godina, rasa i kultura. Također postoje slike osoba s naočalama i bez, te s brkovima ili bradom i bez. Prednost GENKI seta nad ostalim poznatim setovima, npr DFAT i POFA, jest u raznolikosti uvjeta u kojima su nastale slike. DFAT sadrži puno više izraza lica (strah, ljutnja...) od GENKI-a pa je u za neke ostale klasifikatore izraza lica možda i bolji izbor. Međutim, kod detekcije osmijeha, GENKI je trenutno nedvojbeno najbolji izbor.

Slika 1. Primjeri slika iz GENKI seta

Slika 2. Primjeri slika iz Cohn-Kanade DFAT seta

2.2. Prilagodba podataka za učenje klasifikatora

U GENKI setu podataka sadržan je podskup podataka namijenjen upravo učenju detektora osmijeha. Za 3999 lica postoji opisna datoteka koja među ostalim sadrži i informaciju o tome da li lice na slici prikazuje osmijeh ili ne. Među tim informacijama, na žalost, nema informacija o točnoj poziciji lica u slici.

Točna pozicija lica nam je vrlo bitna kod učenja detektora. Potrebno je lice čim preciznije izrezati sa slike, kako bi okolina lica čim manje utjecala na učenje detektora. Izrezivanje lica vrlo se jednostavno postiže izradom skripte koja koristi OpenCV biblioteku. OpenCV biblioteka sadrži nekoliko već naučenih detektora lica. Eksperimentiranjem s nekoliko njih, najbolje rezultate u ovom slučaju dao je *haarcascade_frontalface_alt2.xml*. Skripta za izrezivanje lica, pokrenula je klasifikator na svakoj slici iz seta podataka i kreirala datoteku oblika:

Ime-datoteke.jpg broj-detekcija [x-koordinata-N-detekcije y-koordinata-N-detekcije širina-N-detekcije visina-N detekcije]...

```
files\file2163.jpg 1 3 10 145 145
files\file2164.jpg 1 28 36 121 121
files\file2165.jpg 1 71 51 70 70
files\file2166.jpg 1 0 8 163 163
files\file2167.jpg 1 44 43 108 108
files\file2168.jpg 1 8 53 120 120
files\file2169.jpg 1 56 48 97 97
files\file2170.jpg 1 0 26 148 148
files\file2171.jpg 1 41 70 94 94
files\file2172.jpg 1 32 36 116 116
files\file2173.jpg 1 35 40 110 110
files\file2174.jpg 1 20 27 136 136
files\file2175.jpg 1 48 59 100 100
files\file2176.jpg 1 51 55 85 85
```

Slika 3. Primjer datoteke s označenim licima

Očekivano je da klasifikator lica neće detektirati sva lica, te da će imati i ponešto pogrešnih detekcija. Iz tog razloga u skriptu sam ugradio funkcionalnost da na svim slikama iscrta okvir koji prikazuje detektirano lice. Potrebno je bilo proći kroz sve slike, te izbaciti one koje sadrže pogrešne detekcije ili ih uopće ne sadrže. Nakon obavljenog posla je preostalo, sasvim zadovoljavajućih, 3889 slika za učenje klasifikatora.

Slika 4. Označena lica pomoću klasifikatora iz OpenCV biblioteke

3. Ideja izrade klasifikatora osmijeha

Analiziranjem trenutno najuspješnijeg klasifikatora objavljenog u obliku znanstvenog članka [2], došao sam do ideje da malo modificiram njihovu ideju i napravim klasifikator

pomoću Viola-Jones algoritma. U njihovoj implementaciji, oni koriste klasifikator koji nema kaskada već koristi samo jedan stupanj s mnogo značajki. Viola-Jones algoritmom bi mogli to dosta ubrzati i time olakšati detekciju uređajima s slabom procesorskom snagom. Za početak je potrebno na slikama detektirati lice, koristeći već naučeni klasifikator iz OpenCV biblioteke - *haarcascade_frontalface_default.xml*, te na njemu onda pomoću naučenog Viola-Jones klasifikatora lica pokušati detektirati osmijeh. Detekcija osmijeha na licu se izvodi tako da izrezano lice predajemo klasifikatoru osmijeha i ako on detektira osmijeh (neovisno o položaju i veličini detekcije) smatramo da je na slici detektiran osmijeh. Iako je na taj način moguće da se pod osmijehe prikriju i neke pogrešne detekcije, u rezultatima ćemo vidjeti da se takvi slučajevi pojavljuju u vrlo malom postotku.

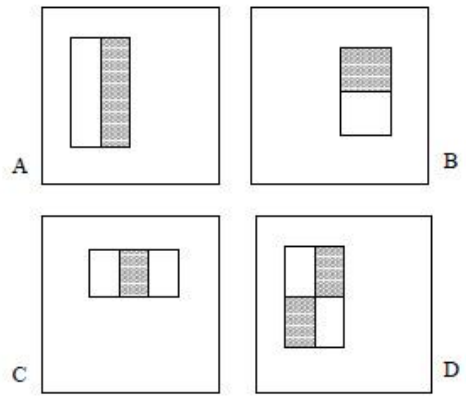
Za prikaz ovog koncepta, razvio sam dvije aplikacije koje koriste isti detektor osmijeha. Jedna je alat za korištenje u komandnoj liniji koja na zadanom skupu slika provede detekciju te zapiše u datoteku informacije o detekciji na svakoj slici. Također nudi i opciju da na slikama označi poziciju lica te indikator osmijeha.

Druga aplikacija je detekcija osmijeha u realnom vremenu koja koristi web kameru. Aplikacija detektira lice, na njemu provodi detekciju osmijeha, te pokazuje indikator ukoliko je detektiran osmijeh. Istovremeno iscrtava i graf s gustoćom odziva klasifikatora.

4. Algoritam Viole i Jonesa

4.1. Značajke objekata

Viola-Jones detektor objekata se temelji na prepoznavanju pomoću niza jednostavnih značajki. Postoji mnogo razloga za korištenje značajki umjesto pixela direktno. Jedan od njih je golema razlika u performansama.



Slika 5. Primjer pravokutnih značajki u odnosnu na dio slike na kojemu se vrši detekcija. Suma piksela koji leže unutar bijelih pravokutnika je oduzeta od sume piksela koji leže u sivim pravokutnicima. Dvo-pravokutne značajke su prikazane na slikama (A) i (B). Slika (C) prikazuje tro-pravokutnu značajku, a slika (D) četvero-pravokutnu značajku.

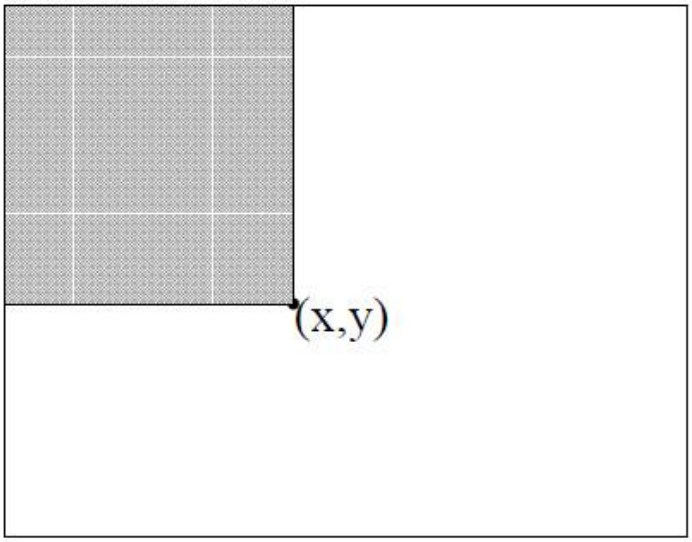
Koriste se tri različite značajke:

- Vrijednost dvo-pravokutne značajke jest razlika sume pixela između dva pravokutna dijela slike. Ta dva dijela su jednake veličine i susjedna su ili vertikalno ili horizontalno
- Vrijednost tro-pravokutne značajke jest suma vanjskih pravokutnika, oduzeta od sume centralnog pravokutnika
- Vrijednost četvero-pravokutne značajke jest razlika između dijagonalnih parova pravokutnika

Iako se prema [5] u Viola-Jones algoritmu koriste samo navedene značajke, u aplikaciju koju ćemo kasnije koristiti u učenju klasifikatora, ugrađene su još neke značajke kao npr. dvo-pravokutne značajke ali zakrenute za stupnjeve od 45°.

4.2. Integralna slika

Pravokutne značajke se mogu vrlo brzo računati koristeći prikaz slike pomoću takozvane „integralne slike“. Integralna slika na poziciji x,y sadrži sumu piksela iznad i na lijevo od x,y.



Slika 6. Vrijednost integralne slike u točki (x,y) je suma svih pixela iznad i lijevo.

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

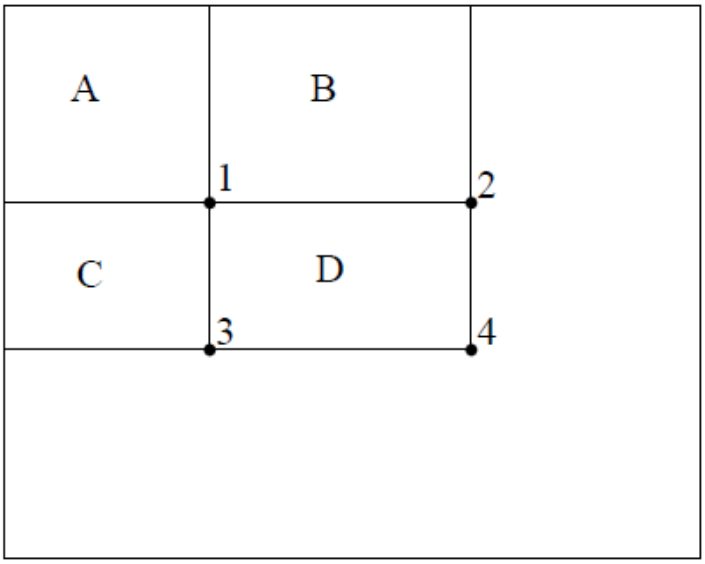
gdje je ii(x,y) integralna slika, a i(x,y) originalna slika.

Koristeći sljedeće jednakosti:

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

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

(gdje je s(x,y) kumulativna suma reda, s(x,-1) = 0, ii(-1,y)=0) integralna slika se može izračunati u jednom prolazu preko originalne slike.



Slika 7. Suma piksela unutar pravokutnika D može se izračunati pomoću četiri točke iz integralne slike. Vrijednost integralne slike na poziciji 1 je suma svih piksela pravokutnika A. Vrijednost na poziciji 2 jest vrijednost pravokutnika A+B, na poziciji 3 je A+C, a na poziciji 4 je A+B+C+D. Suma piksela unutar pravokutnika D se dakle može izračunati kao 4 + 1 – (2+3).

Koristeći integralnu sliku, bilo koja pravokutna suma može se izračunati pomoću 4 vrijednosti. Očito je onda da se razlika između dva pravokutnika može izračunati pomoću 8 vrijednosti, a pošto za naše potrebe koristimo samo susjedne pravokutnike, oni se mogu izračunati pomoću 6 vrijednosti. Osam vrijednosti je potrebno za tro-pravokutne značajke, a devet za četvero-pravokutne.

4.3. Rasprava o značajkama

Pravokutne značajke su prilično jednostavne i primitivne u usporedbi s nekim drugim alternativnim značajkama za analiziranje slika ali nam pružaju vrlo veliku brzinu izračunavanja kao kompenzaciju za nešto lošiju mogućnost analiziranja. Ipak, pravokutne značajke nam daju dovoljno dobar prikaz slike koji podržava efektivno učenje.

Da bi uočili prednost i brzinu integralnih slika, dobro je promotriti zastario način izračunavanja piramide slika. Kao i većina metoda za prepoznavanje objekata, i ova skenira ulaznu sliku na različitim veličinama. Počevši od veličine 24x24 piksela, slika se

skenira 11 puta, svaki puta s faktorom povećanja 1.25. Zastario način računanja piramida slika računa piramidu od 11 slika, svaki puta 1.25 puta manju od prethodne slike. S jednom veličinom se tada skenira slika u svakoj poziciji. Izračun takve piramide iziskuje puno vremena, te je vrlo teško izračunati piramidu u brzini od 15 slika po sekundi.

Suprotno tome, pravokutna značajka se može u bilo kojoj veličini i na bilo kojoj poziciji izračunati u svega nekoliko koraka. Efektivan detektor se može napravit već sa samo dvije pravokutne značajke. Uzimajući u obzir brzinu računanja tih značajki, kompletan proces detekcije objekta se može raditi brzinom od 15 slika po sekundi, što je brže nego samo računanje piramide slika. Time je jasno da je svaki detektor koji koristi piramidu slika nužno sporiji od Viola-Jones detektora.

4.4. Učenje klasifikatora

Za svaki dio slike (24x24 piksela) imamo 45396 različite značajke (gledajući dakle, samo ove osnovne – pravokutne). Kasnije ćemo kod učenja detektora osmijeha taj skup malo proširiti. Iako se svaka značajka vrlo brzo izračuna, računanje svih značajki iziskuje previše vremena. Eksperimentima je pokazano da se s malim brojem različitih značajki može kreirati efikasan klasifikator. Problem je naći koje su to značajke.

Ideja odabira takvih značajki je da napravimo jedan jednostavan klasifikator koji se sastoji od samo jedne značajke. Testiramo ga na podacima (slikama koje sadrže traženi objekt i onima koje ne) te promijenimo set podataka tako da povećamo težinu podataka na kojima klasifikator griješi. Na izmijenjenom setu podataka na isti način naučimo još jedan jednostavan klasifikator, kombiniramo ta dva klasifikatora, testiramo ih i opet promijenimo set podataka tako da povećamo težinu podataka na kojima klasifikator griješi. Postupak ponavljamo T(sami određujemo broj) puta kako bi dobili jaki klasifikator. Ova metoda se zove „Boosting“ i u praksi daje vrlo dobre rezultate.

Viola i Jones su malo modificirali „Boosting“. Njihova metoda u svakom koraku testira sve značajke te izabire najbolju od njih. Najbolja je ona koja je imala najmanje grešaka. Nakon toga se izabrana značajka dodaje u klasifikator i mijenja se set podataka sukladno

rezultatima. Ponavljamo postupak T puta. T je u praksi najčešće broj oko 1000. Ova varijanta „Boosting“-a se naziva „AdaBoost“[4].

Formalni opis algoritma učenja[1]:

- Dani su primjeri slika $(x_1,y_1)....(x_n,y_n)$ gdje je $y_i = 0$ za negativne slike, a 1 za pozitivne
- Početne težine su $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ za $y_i = 0$ odnosno 1, gdje su *m* i *l* broj negativnih, odnosno pozitivnih slika.
- Za t = 1 do T
    - Normalizirajmo težine

      $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

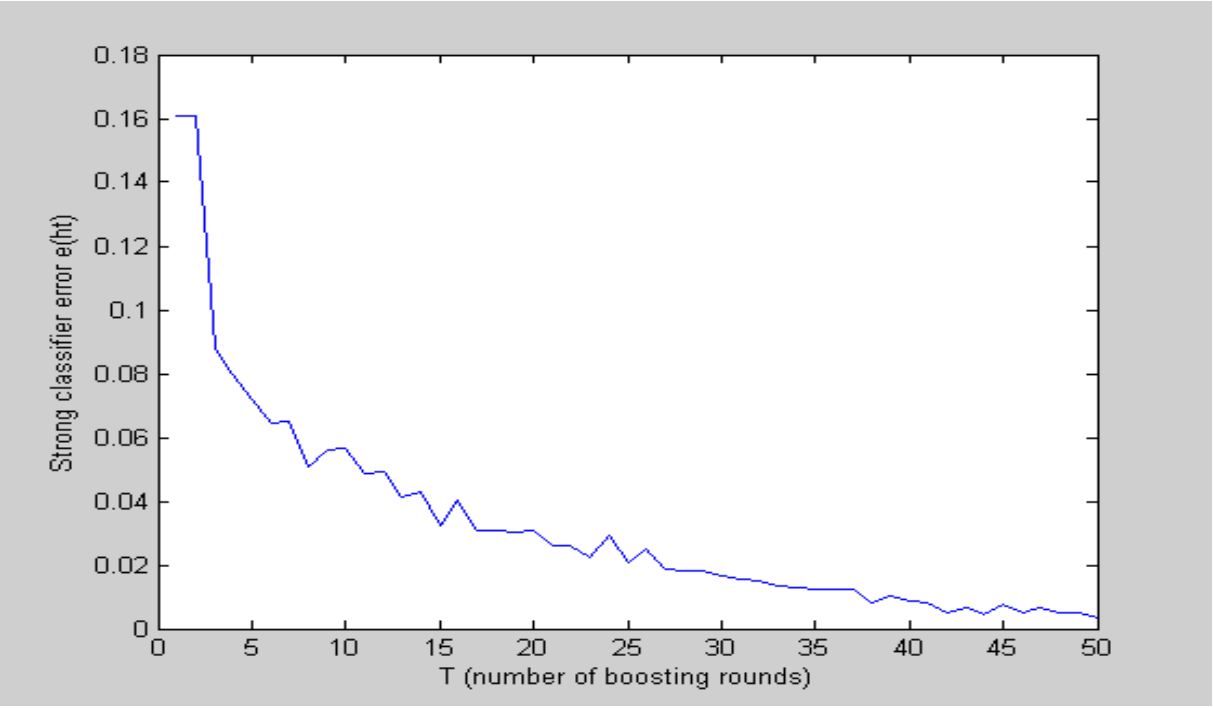
      tako da je $w_{t,i}$ distribucija vjerojatnosti.
    - Za svaku značajku j, treniramo klasifikator h_j koji je ograničen na korištenje samo jedne značajke. Greška se računa u odnosu na w_t, $\epsilon_j = \sum_i w_i \left|h_j(x_i) - y_i\right|$.
    - Izaberemo klasifikator h_t s najmanjom greškom ϵ_t
    - Ažuriramo težine:

      $$w_{t+1,i} = w_{t,i}\beta_i^{1-e_i}$$

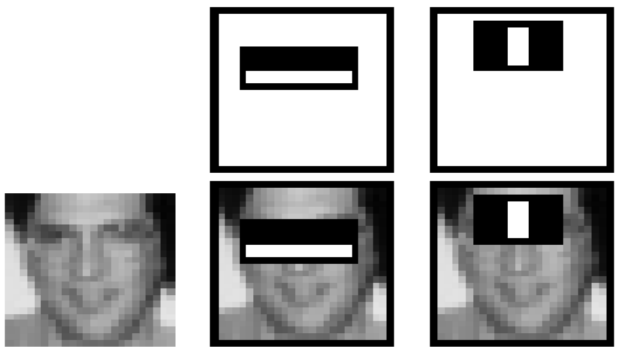
      Gdje je $e_i = 0$ ako je primjerak x_1 klasificiran točno, a 1 inače. $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- Konačni jaki klasifikator je:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & inače \end{cases}$$

gdje je $\alpha_t = \log\frac{1}{\beta_t}$



Slika 8. Pogreška klasifikatora se eksponencijalno približava nuli s povećanjem broja ponavljanja „Boosting“ postupka

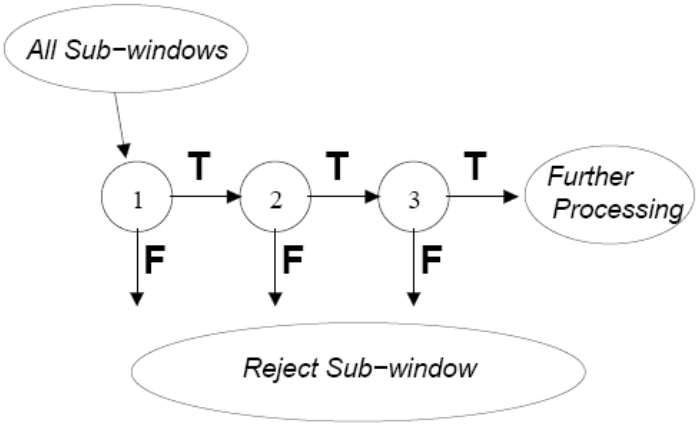


Slika 9. Prva i druga značajka klasifikatora koje je odabrao AdaBoost kod učenja na prepoznavanje lica. Značajke su prikazane u prvom redu, a u drugom redu su nacrtane preko tipične slike za testiranje. Prva značajka mjeri razliku u intenzitetu slike oko očiju i gornjeg dijela obraza. Značajka se iskazuje u tome da je najčešće područje oko očiju tamnije od gornjeg dijela obraza. Druga značajka uspoređuje intenzitet očiju i gornjeg dijela nosa.

4.5. Kaskade klasifikatora

Algoritam konstrukcije kaskade klasifikatora omogućuje nam preciznije i znatno brže prepoznavanje. Ključ je u tome što manji klasifikatori, koji su ujedno i vrlo brzi, mogu biti konstruirani tako da odbijaju mnogo negativnih dijelova slika, a prepoznaju gotovo sve pozitivne. Zato se jednostavni klasifikatori koriste za odbacivanje većine negativnih dijelova, prije nego pozovemo složenije klasifikatore koji nam smanjuju mogućnost krive detekcije.

Ideja algoritma je da složimo niz (kaskadu) klasifikatora koji će raditi na način da ako klasifikator da rezultat da u gledanom dijelu slike postoji traženi objekt, nastavljamo isti dio slike testirati sa sljedećim klasifikatorom i tako do nekog određenog broja klasifikatora. U slučaju kada neki klasifikator kaže da na određenom dijelu slike nije traženi objekt, odmah prekidamo algoritam za taj dio, te ne nastavljamo s daljnjim klasifikatorima.



Slika 10. Shematski prikaz kaskade klasifikatora. Kaskada klasifikatora se primjenjuje na svaki dio slike. Prvi klasifikator odbaci mnogo negativnih dijelova uz vrlo malo računanja. Svaki sljedeći dio odbacuje daljnje negativne dijelove al uz više računanja. Nakon samo nekoliko koraka, većina negativnih dijelova je odbačena.

Očigledno je da je većina dijelova slike negativna, te zbog tog kaskada klasifikatora teži čim veći dio njih odbaciti čim ranije kako bi bilo potrebno računati čim manji broj dijelova na složenijim klasifikatorima.

5. Učenje klasifikatora alatima OpenCV-a

U biblioteci OpenCV postoji nekoliko alata koji nam vrlo učinkovito mogu pomoći u izradi detektora osmijeha. Implementacija AdaBoost algoritma sadržana je u aplikaciji *haartraining*, pomoću koje sam i izučio klasifikatore osmijeha. Prije korištenja samog haartraininga, potrebno je pripremiti podatke u odgovarajućem obliku za učenje. I u tom pogledu nam OpenCV pomaže aplikacijom *createsamples*.

5.1. Kreiranje malih sličica za učenje

Aplikacija createsamples nam omogućuje kreiranje vec datoteka iz niza slika. To nam je potrebno kako bi pozitivne slike skalirali na željenu veličinu za učenje (24 x 24 je uobičajena vrijednost), normalizirali svjetlinu slika, te ih pretvorili u tonove sive boje. Na taj način slike su spremne za koristiti u haartrainingu. Pored toga, createsamples sve te umanjene slike spremi zajedno u jednu datoteku s ekstenzijom vec, koja je upravo tražena kao ulazna datoteka u haartraining.

Createsamples nudi mnoštvo opcija, ali u ovom primjeru, potrebne su samo neke:

-vec <vec_file_name>	Naziv izlazne datoteke
-img <image_file_name>	Naziv datoteke s popisom slika (primjer na slici 3)
-num	Broj sličica koje treba kreirati, odgovara ukupnom broju slika
-w	Širina generiranih sličica
-h	Visina generiranih sličica
-show	Korisna opcija za provjeravanje ispravnost generiranja sličica. Program nam prikazuje sličice koje generira. Prikazivanje možemo prekinuti tipkom 'Esc'.

Ako uzmemo kao primjer strukturu direktorija :

/slike

slika1.jpg

slika2.jpg

slike.idx

Tada bi datoteka slike.idx mogla izgledati ovako:

slike/slika1.jpg 1 10 10 20 20

slike/slika2.jpg 1 15 15 20 20

gdje su brojevi pisani u obliku:

<broj_lica_na_slici> [<x-koordinata-Ntog-lica> <y-koordinata-Ntog-lica> <širina-Ntog-lica> <visina-Ntog-lica>]...

Ovu datoteku treba ručno kreirati. Za potrebe ovog rada, napravljena je skripta koja pomoću detektora lica iz biblioteke OpenCV, detektira lica na svim datotekama iz GENKI set i generira kao izlaz ovakvu idx datoteku.

Kreiranu idx datoteku predajemo programu createsamples na način:

createsamples -vec slike.vec -img slike.idx -num 2 -w 32 -h 32

u slučaju da želimo kreirati sličice veličine 32 x 32 piksela.

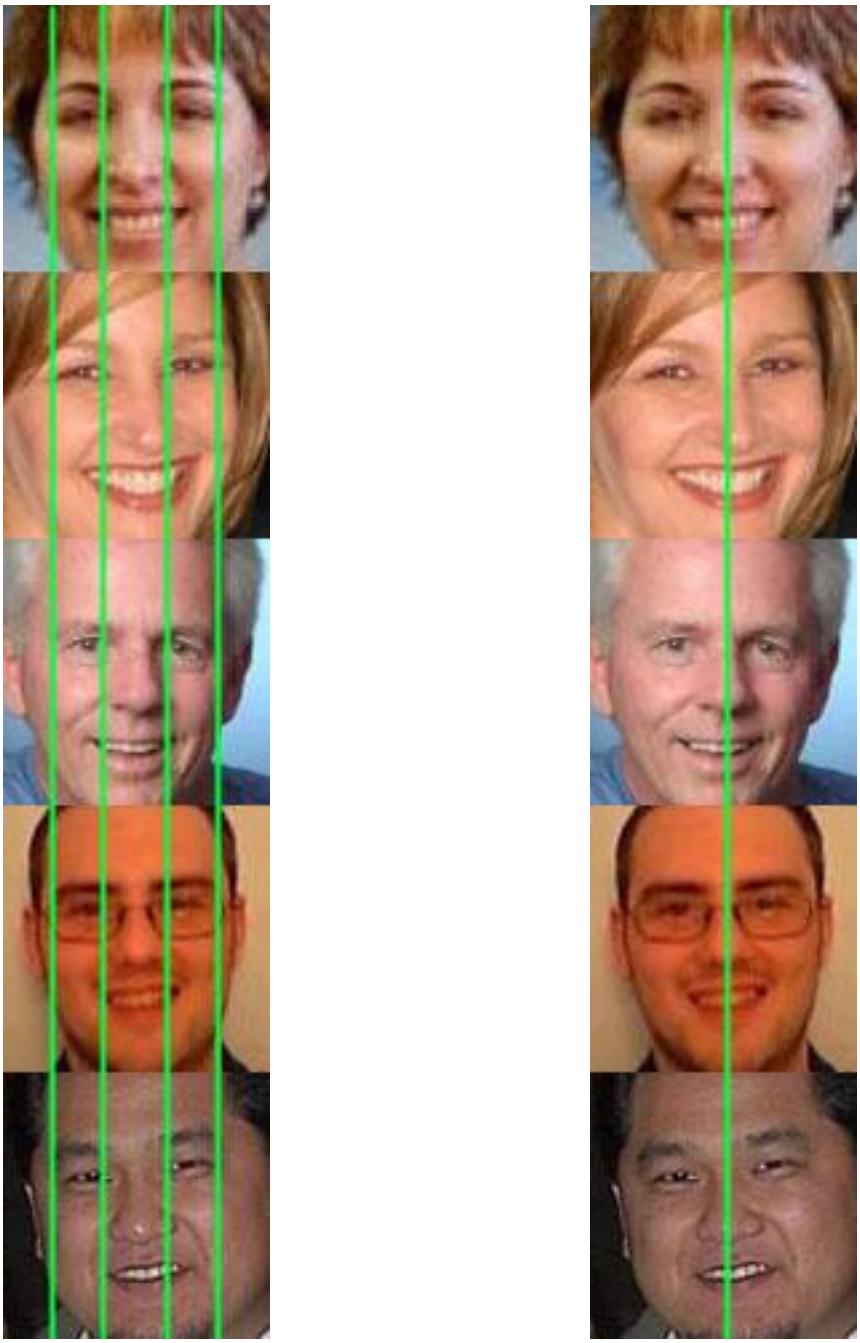
Istim ovim postupkom možemo generirati i vec datoteku s negativnim slikama. Iako je uobičajeno da se negativne slike rade na drugačiji način, u nekim slučajevima ovo može biti praktičnije. Više o tome u poglavlju o kreiranju pozadinskih(negativnih) slika.

Createsamples je inače pogodan alat ukoliko nemamo dovoljno pozitivnih slika za učenje klasifikatora. On se može iskoristiti za generiranje slika primjenjujući na slike razne distorzije, varijante svjetlosti, transformacije, šumove, promjene boja itd. Tako generirane slike tada može lijepiti po raznim pozadinskim slikama kako bi dobili i slike za testiranje našeg klasifikatora. Takav proces često se primjenjuje kod izrade npr. klasifikatora za

prepoznavanje loga kompanije. Createsamples-u se da jedan logo, te on iz njega generira mnoštvo slika s primjerima u kakvima bi se taj logo mogao naći u stvarnom svijetu.

Na uspješnost učenja klasifikatora osim količine pozitivnih slika utječe i njihova kvaliteta. Kvalitetna slika znači da osim što je slikana bez povećih šumova, mora biti normalizirana što se kontrasta i boje tiče. Poželjno je da sve slike budu pozicionirane isto, npr. kod učenja ovog detektora, savršeno bi bilo kada bi sve slike imale oči, nos i usta na približno jednakim pozicijama. Sam detektor lica je to stavio u neku ravnotežu, ali ona nije savršena. Idealno bi bilo ručno ih postaviti, no to iziskuje jako puno posla na setu od 4000 slika. Drugi način je detekcija očiju pomoću nekog klasifikatora pa skaliranje i translatiranje slika prema dobivenim rezultatima tako da se oči uspiju čim preciznije pozicionirati. Na raspolaganju sam imao naučene detektore očiju iz OpenCV biblioteke, međutim, pokrenuvši sve detektore na setu za učenje, nisam dobio zadovoljavajuće rezultate. Većina očiju uopće nije bila detektirana, dok one koje jesu, nisu bile dovoljno precizne. Odnosno, pozicioniranjem slika prema tim detekcijama, dobivao sam preciznost vrlo sličnu kao onu što mi je dao detektor lica, a pri tome sam izgubio otprilike pola seta za učenje zbog neuspjelih detekcija. Iz tog razloga, zaključio sam da je za ovaj rad dovoljna preciznost pozicioniranja koju nam pruža detektor lica.

Slika 11. Primjer ravnine očiju i usta u setu podataka za učenje



Slika 12. Primjer ravnine očiju i usta u setu podataka za učenje

5.2. Priprema pozadinskih(negativnih) slika

Viola-Jones detektor je binarni klasifikator: on jednostavno odlučuje da li (dakle, isključivo „da“ ili „ne“) je neki objekt u slici sličan onima iz seta za učenje. Opisan je već postupak kako prikupiti i pripremiti pozitivne slike za učenje koje sadrže objekt kojeg želimo prepoznavati. Sada je potrebno opisati i objasniti postupak nabavke i pripreme negativnih

primjera, kako bi klasifikator mogao naučiti kako ne izgleda traženi objekt, u ovom slučaju ostale emocije osim osmijeha. Svaka slika koja ne sadrži traženi objekt može biti negativna slika, međutim, u ovom primjeru pošto će se detektor osmijeha pokretati isključivo na slikama lica, najbolje je da negativne slike budu lica bez osmijeha. Iako je najbolje da negativne slike budu iz okoliša traženog objekta, čak i uz one koje to nisu,već su neke slučajne slike, mogu se dobiti zadovoljavajući rezultati. Neki od naučenih detektora, učeni su tako da je za negativne dijelove haartraining nasumce izrezivao dijelove lica (dakle, ne cijelo lice već samo dio) i to je dalo vrlo dobre rezultate!

Uobičajeno je negativne slike staviti u jedan zaseban direktorij te kreirati datoteku s ekstenzijom txt, u kojoj ćemo pobrojati sve slike s putanjom do njih.

Pretpostavimo da imamo strukturu direktorija:

/neg_slike
    slika1.jpg
    slika2.jpg
bg.txt

Datoteka bg.txt:
neg_slike/slika1.jpg
neg_slike/slika2.jpg

Slika 13. Primjeri negativnih(pozadinskih) slika korištenih u učenju detektora osmijeha

Kada haartrainingu predamo slike u ovom obliku, on kod učenja svake kaskade sam generira sličice veličine pozitivnih slika (npr 24x24 ili 32x32 piksela) slučajnim odabirom iz slika koje mu predamo kao pozadinske. Iz tog razloga kod učenja detektora on u nekim od naučenih detektora uspoređuje neku značajku s pozitivnim licem s osmijehom, ali kao negativ uzima samo dio lica koje se ne smiješi (to npr. može biti samo oko, nos itd.). Vidjet ćemo kasnije u rezultatima da je i to dalo sasvim dobre rezultate.

Haartraining možemo natjerati da kao negative uzima isključivo cijela lica, na način da sve negativne slike prije učenja skaliramo na veličinu pozitivnih slika. U tom slučaju haartraining neće moći izrezivati nikakve dijelove slika, već će morati uzeti cijelu sliku.

Isto to možemo napraviti i na jednostavniji način od kad je izašla verzija 2.1 OpenCV biblioteke. U haartraining je dodan argument –bg-vecfile pomoću kojeg mu specificiramo da se pozadinske nalaze u vec formatu. Vec format kreiramo isto kao i kod pozitivnih slika. Pri tome moramo paziti da pozitivne i negativne slike budu istog formata. Ovaj način nam uvelike smanjuje broj negativnih slika kod učenja. U prvom načinu haartraining iz jedne slike može izrezati hrpu manjih sličica koje koristi u učenju, dok u drugom načinu iz jedne slike haartraining dobiva ravno jednu sličicu. Nedostatak negativnih slika kod učenja može stvarati probleme kod pokušaja da se ostvari što niži postotak pogrešnih detekcija (onih u kojima detektor detektira osmijeh, a on zapravo ne postoji). Više o tome kod raspravljanja rezultata naučenih detektora.

5.3. Učenje klasifikatora

Za učenje kaskade klasifikatora koristimo haartraining alat iz biblioteke OpenCV. Haartraining nudi učenje kaskade klasifikatora potrebne za algoritam Viole i Jonesa, te uz to nudi još mnoštvo opcija za parametriziranje učenja.

Neke od bitnih opcija za učenje detektora osmijeha su:

-data <dir_name>

Lokacija naučenog klasifikatora. Haartraining kreira direktorij <dir_name> te u njega sprema svaku kaskadu u zaseban poddirektorij. Kada završi s učenjem, kreira i datoteku <dir_name>.xml koji pomoću OpenCV funkcija možemo koristiti za evaluaciju i pokretanje samog detektora.

-vec <vec_file_name>

Datoteka s pozitivnim slikama. Kreirana pomoću createsamples alata.

-bg <background_file_name>

Datoteka s opisima pozadina. Idx datoteka s pobrojanim slikama ili vec datoteka kao i kod pozitivnih slika.

-npos <number>

Broj pozitivnih slika koje se koriste kod učenja pojedinog stupnja kaskade.

-nneg <number>

Broj negativnih slika koje se koriste kod učenja pojedinog stupnja kaskade. Negativne slike koje neka kaskada uspješno odbaci se ne koriste u daljnjem učenju. Stoga, ukoliko imamo ograničen broj negativnih primjera, treba biti oprezan s ovim parametrom.

-nstages <number_of_stages>

Broj stupnjeva kaskade koje treba izgraditi. To je uz *maxfalsealarm* drugi uvjet zasutavljanja učenja.

-mem <MB>

Dopuštena količina memorije za kalkulacije unutar haartraininga. Teoretski, što više memorije imamo, učenje bi trebalo ići brže. Međutim, kod učenja detektora osmijeha, haartrainingu je bilo dopušteno koristiti 2GB memorije, no on niti u jednom trenutku nije koristio više od 200 mb.

-sym

Time kažemo haartrainingu ukoliko su objekti za detekciju simetrični. Simetričnost objekata može ubrzati učenje. Ovo je pretpostavljena vrijednost.

-nonsym

U slučaju da objekti za detekciju nisu simetrični, na ovaj način isključujemo optimizacije brzine koje su uvedene zbog simetričnosti objekata.

-minhitrate <min_hit_rate>

Minimalni željeni postotak uspješnost svakog stupnja kaskade. Ukupni postotak uspješnosti može se aproksimirati kao $min\_hit\_rate^{number\_of\_stages}$.

-maxfalsealarm <max_false_alarm_rate>

Maksimalni željeni postotak krive detekcije osmijeha u pojedinom stupnju kaskade. Ukupni postotak krive detekcije može se aproksimirati kao $max\_false\_alarm\_rate^{number\_of\_stages}$.

-mode <BASIC (pretpostavljena vrijednost) | CORE | ALL>

Pomoću ovog argumenta odabiremo koje sve značajke želimo istraživati u procesu učenja. BASIC označuje samo uspravne pravokutne značajke (kao što se prema Violi i Jonesu[5] koristi u algoritmu), dok ALL uključuje i značajke rotirane za 45 stupnjeva.

-w <sample_width>

Širina pozitivnih slika, mora biti identična onoj koju smo koristili kod kreiranja vec datoteke putem createsamples alata.

-h <sample_height>

Visina pozitivnih slika, mora biti identična onoj koju smo koristili kod kreiranja vec datoteke putem createsamples alata.

-bg-vecfile

Ovaj parametar nam omogućuje da pozadinske (negativne) slike predamo kao vec datoteku kreiranu s createsamples alatom. Na taj način dajemo konkretne slike, umjesto da haartraining izrezuje iz priloženih.

-bt (DAB | RAB | LB | GAB (pretpostavljena vrijednost))

Odabiremo koji Boosting algoritam se koristi. U izboru su nam: Discrete AdaBoost, Real AdaBoost, LogitBoost i Gentle AdaBoost. Prema autorima trenutno najuspješnijeg detektora osmijeha [2], najbolje rezultate daje Gentle AdaBoost pa sam njega i ja koristio.

Jedan on načina poziva haartraininga za učenje detektora osmijeha izgledao bi:

haartraining –data smile_detector –vec smiles.vec –bg nonsmiles.vec –bg-vecfile –minhitrate 0.998 –maxfalsealarm 0.4 –w 32 –h 32 –mem 2048 –sym –mode ALL –npos 1000 –nneg 500

Vrijeme trajanja jednog učenja po prilici traje od nekoliko dana do tjedan dana za kvalitetne rezultate. To zna biti nezgodno kod prvih pokušaja jer je ne praktično čekati na rezultate tjedan dana a nismo niti sigurni ako smo dobro zadali slike i parametre. No i za to postoji rješenje.

5.4. Evaluacija klasifikatora prije završetka učenja

Kako ne bi potratili vrijeme čekajući po tjedan dana za učenje jednog pogrešnog klasifikatora, postoji i način da provjerimo njegov rad i prije nego završi cijelo učenje. Haartraining za vrijeme učenja, sve podatke sprema unutar zadanog direktorija, te unutar

njega kreira poddirektorije u koje sprema informacije o svakom pojedinom stupnju kaskade. Za vrijeme dok se klasifikator još uči, trenutne podatke možemo iskopirati na neku drugu lokaciju, te iz njih kreirati .xml datoteku pomoću OpenCV alata *convert_cascade*.

Convert_cascade nudi parametre:

convert_cascade --size="<width>x<height>" input_cascade_path output_cascade_filename

Pri čemu je:

--size="<width>x<height>"

Veličina pozitivnih primjerka slika korištenih u učenju kaskade

input_cascade_path

Put do direktorija gdje se nalaze opisi stupnjeva kaskade kreirani pomoću haartraining alata

output_cascade_filename

Datoteka u koju da se spremi kreirani klasifikator.

Na ovaj način možemo već nakon nekoliko izučenih stupnjeva provjeriti ima li smisla učenje klasifikatora. Ukoliko je postotak točnih detekcija već onda premalen, nema smisla nastaviti učenje, jer se daljnjim učenjem taj postotak može jedino smanjiti. Isto tako, ovaj alat možemo iskoristiti u slučaju da nam gotova kaskada ima vrlo dobar *maxfalsealarm*, ali prenizak *hitrate*, pa možemo probati kakvi bi rezultati bili bez zadnjih nekoliko stupnjeva u kaskadi. Na taj način bi *hitrate* trebao porasti, ali se postotak *falsealarma* povećati. To bi smo napravili tako da direktorij s opisima iskopiramo na novu lokaciju, pobrišemo poddirektorije zadnjih nekoliko stupnjeva kaskade, te pokrenemo convert_cascade alat kako bi dobili novu kaskadu.

6. Evaluacija klasifikatora

Svaku naučenu kaskadu potrebno je evaluirati na nekom skupu podataka te na neki način izmjeriti uspješnost detekcije. Evaluaciju ću provoditi jednom od dvije razvijene aplikacije u ovome radu: alatom koji se poziva iz komande linije i na zadanom setu podataka detektira osmijehe, te rezultat, zajedno s statističkim opisom zapisuje u datoteku.

Kao mjerila uspješnosti, odlučio sam koristiti:

- Postotak uspješnih detekcija

  Mjeri se na način da za skup uzmemo sve slike na kojima jest osmijeh, te iz njih izbacimo slike na kojima nije uspješno detektirano lice. Neuspjela detekcija lica nije pogreška detektora osmijeha već lica. Kako detekcija lica nije predmet ovog rada, to nećemo uzimati u razmatranje. Na tako određenom skupu, gledamo omjer slika na kojima je detektor uspješno detektirao osmijeh, naspram broja slika u skupu.

  Želja nam je maksimizirati ovaj postotak.

- Postotak krivih detekcija

  Iz skupa slika bez osmijeha, izbacimo slike s neuspjelom detekcijom lica, te gledamo omjer broja slika na kojima je detektiran osmijeh naspram broj slika u skupu.

  Ovaj postotak je potrebno minimizirati.

- Ispravnost detekcija

  Iako je ovaj parametar jednoznačno određen s prethodna dva, on nam dosta informativno može opisati koliko je uspješan naš klasifikator. Na skupu slika s ispravnom detekcijom lica, brojimo na koliko slika je detektor dao točan rezultat. Taj broj stavljamo u omjer s brojem slika u skupu.

Za evaluaciju, najbolje je imati skup slika koji nije sudjelovao u učenju, kako bi izmjereni rezultati bili što bliži onima koje bi detektor trebao imati u stvarnom svijetu. Iz tog razloga, iz GENKI seta sam prije učenja izlučio 50 slika s osmjesima i 50 lica bez osmijeha i ostavio ih samo za evaluaciju. Ukupno dakle imamo 100 slika za evaluaciju i mislim da se rezultati dobiveni na tom skupu, mogu smatrati vjerodostojnima. Slike su izabrane slučajnim odabirom te sadrže ljude slikane u raznim okolinama, položajima i uz različite uvijete slikanja.

6.1. Aplikacija za evaluaciju

Aplikacija za evaluaciju je, dakle, jedna od dvije razvijene u okviru ovog rada. Napisana je u jeziku C++, koristeći OpenCV biblioteku verzije 2.1. Kao parametre prima datoteku s popisom slika za evaluaciju, te uz svaku sliku oznaku da li je na njoj osmijeh. Moguće je pomoću parametra aplikaciji specificirati da na svim slikama označi i indicira ukoliko postoji osmijeh, te sliku spremi u novu datoteku, dodajući joj nastavak „_d“. Kao izlaz aplikacija kreira datoteku s popisom svih slika na kojima je načinjena detekcija, lokacijom lica na slici te oznakom da li to lice sadrži osmijeh ili ne. Na kraju datoteke dobivamo i ispisane parametre koje koristimo za usporedbu uspješnosti klasifikatora nabrojane u prethodnom poglavlju. Uz to, dobijemo i podatak o broju neuspjelih detekcija lica. Aplikacija detekciju osmijeha smatra uspješnom ukoliko detektira najmanje dvije detekcije u približno istom području. Viola-Jones detektor uvijek napravi nekoliko detekcija oko istog objekta, pa na taj način eliminiramo lažne detekcije.

7. Rezultati evaluacije

Ukupno sam izučio sedam različitih klasifikatora, koji variraju u pogledu seta podataka na kojem su učeni i parametrima učenja. Neki od njih su bili grupe pogreške zbog krivog postavljanja učenja, skupa podataka i sl.

7.1. Klasifikator broj 1

Klasifikator je učen uz parametre haartraininga:

Minhitrate 0.995

Maxfalsealarm 0.5

Nstages 14

Slike su kod učenja skalirane na veličinu 32 x 32 piksela, a za negativne slike se uzimaju dijelovi lica, nasumce izrezani. Učenje je trajalo oko šest sati i s obzirom na to dalo vrlo zadovoljavajuće rezultate. Evaluator je kod evaluacije bio postavljen da detekciju smatra uspješnom samo ukoliko postoje barem tri slične detekcije na licu.

Uz te parametre, dobio sam rezultate:

Postotak uspješnih detekcija – **65.3%**

Postotak pogrešnih detekcija – **43.8%**

Ispravnost detekcija – **60.8%**

7.2. Klasifikator broj 2

Kako je u prethodnom klasifikatoru bio izražen postotak pogrešnih detekcija, u ovome sam ih želio smanjiti, te sam postavio dosta strože parametre, dok je sve ostalo ostalo jednako:

Minhitrate 0.995

Maxfalsealarm 0.35

Nstages 20

Nakon 5 dana učenja (14 izučenih stupnjeva kaskade), odlučio sam do tada prikupljene rezultate spojiti u jedan klasifikator pomoću *convert_cascade* alata. Rezultati pokazuju da smo uspjeli gotovo ukloniti broj pogrešnih detekcija, međutim, uz veliku žrtvu ispravnih detekcija. Ovakav klasifikator ne bi bio od velike koristi u stvarnome svijetu.

Postotak uspješnih detekcija – **23.9%**

Postotak pogrešnih detekcija – **2.3%**

Ispravnost detekcija – **68.2%**

Uspješnost je visoka u ovom klasifikatoru iz razloga što je 97.7% ne osmijeha ispravno klasificirao, međutim, to nije baš najbolji pokazatelj željenog rada našeg klasifikatora.

7.3. Klasifikator broj 3

Kod ovog klasifikatora, poučen prethodnom pogreškom nisam želio postaviti previše niski prag pogrešnih detekcija, već pokušati što više povećati broj ispravnih detekcija. U ovom slučaju koristim slike dimenzija 24x24 piksela.

Minhitrate 0.999

Maxfalsealarm 0.50

Nstages 14

Iako sam očekivao veći postotak uspješne detekcije nego u prvom klasifikatoru zbog višeg postavljenog praga, izgleda da se smanjivanjem slike na 24x24 piksela izgubio dio korisne informacije, te su rezultati degradirani:

Postotak uspješnih detekcija – **52.2%**

Postotak pogrešnih detekcija – **19.05%**

Ispravnost detekcija – **65.9%**

7.4. Klasifikator broj 4

Poučen pogreškom s smanjivanjem slika kod učenja, kod ovog klasifikatora sam ih vratio na 32x32 piksela. S istim uvjetima učenja, dobivam poprilično bolje rezultate:

Postotak uspješnih detekcija – **62.5 %**

Postotak pogrešnih detekcija – **16.37%**

Ispravnost detekcija – **72.92%**

Iako je uspješnost porasla tek nekoliko posto, smatram ovaj klasifikator uspješnijim jer ima za 10% veći postotak uspješnih detekcija. 16.7% smatram kao zadovoljavajući postotak pogrešnih detekcija za ovaj rad.

7.5. Klasifikator broj 5

Kako su svi klasifikatori do sada učeni tako da su za pozadinske slike izrezivali nasumce dijelove lica bez osmijeha, smatrao sam da bi rezultati trebali biti bolji ako učenje prisilim da koristi isključivo cijela lica. Međutim, pokretanjem učenja s takvim skupom podataka i uvjetima:

Minhitrate 0.999

Maxfalsealarm 0.50

Nstages 14

dobio sam nakon otprilike tjedan dana, vrlo razočaravajuće rezultate:

Postotak uspješnih detekcija – **97.92 %**

Postotak pogrešnih detekcija – **95.83 %**

Ispravnost detekcija – **51.04 %**

Iako je postotak uspješnih detekcija vrlo visok, uz postotak pogrešnih detekcija je očigledno da je klasifikator gotovo uvijek odlučivao da je na slici osmijeh. Konkretnije, samo na jednoj slici na kojoj je bilo osmijeh je detektor vratio da nema osmijeha. Dok je kod slika bez osmijeha, samo na jednoj slici uspješno odredio da osmijeha nema. Nisam očekivao ovako loše rezultate niti u kojem slučaju, međutim, mislim da je opravdanje u tome što je ovako učenje imalo daleko manji broj negativnih slika, očito nedostatan za učenje uspješnog detektora.

7.6. Zaključak o rezultatima detekcije slika

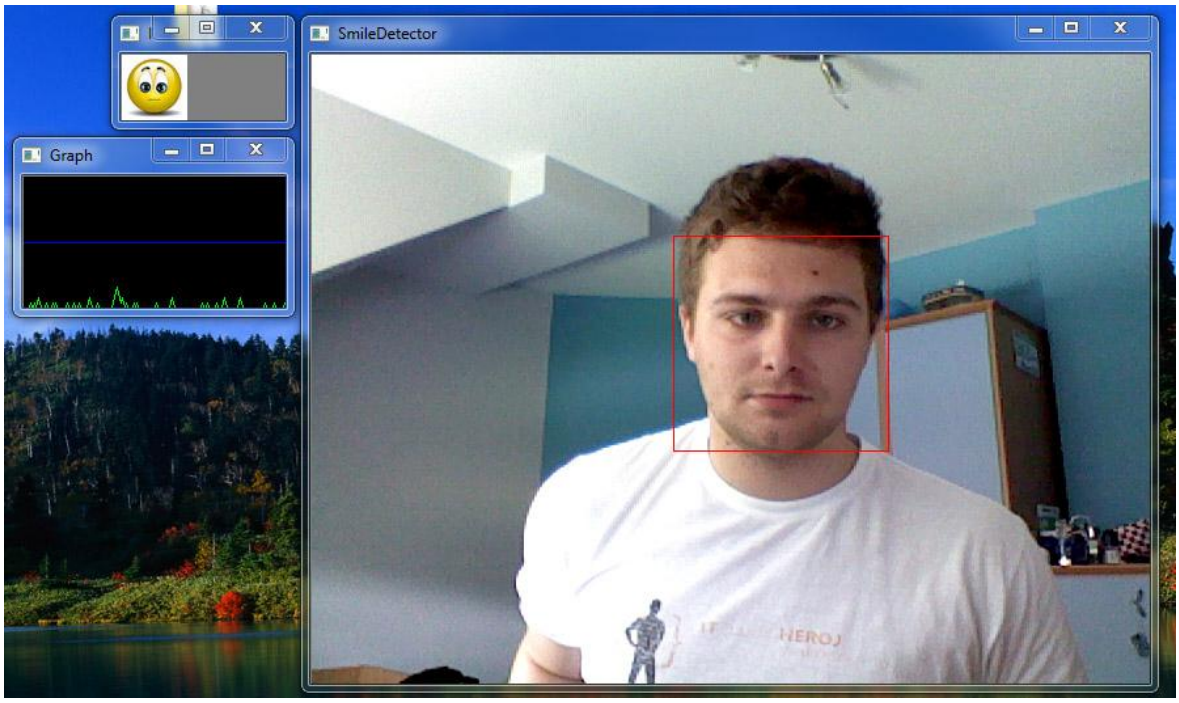
Kod učenja svakog novog klasifikatora, uočio sam po jednu bitnu činjenicu i tako mislim da je detektor pod brojem 4 najviše iskoristiv. Međutim, niti jedan od ovih rezultata nije dovoljno dobar da bi se usudio reći da je klasifikator spreman za korištenje u stvarnome svijetu. Odgovor na taj problem leži u metodi koju sam koristio za izradu detektora. Viola-Jones sam odabrao kako bih napravio eksperiment i vidio ukoliko pomoću tog algoritma možemo detektirati osmijeh. U situaciji da su rezultati bili bolji, imali bi detektor koji bi bio znatno brži od do sada predstavljenih. Razlog zašto algoritam Viole i Jonesa ne daje bolje rezultate je taj što je algoritam dizajniran s idejom da treba češće odbacivati nego detektirati, pošto je npr. kod detekcije lica više dijelova slike bez lica, nego ih s licem. Upravo iz tog razloga Viola-Jones ima kaskade kako bi u čim ranijem koraku, i uz čim manje utrošenog vremena, odbacio negativne dijelove slike. Prema tome, samo učenje ne tretira jednako pozitivne i negativne primjere. Natjerati haartraining da ih jednako tretira bi značilo da postavimo npr. minhitrate na 0.9 i maxfalsealarm na 0.1, međutim, u praksi haartraining ne uspijeva izučiti kaskade s tako malim maxfalsealarmom. Ovime zaključujem da korištenje Viola-Jones algoritma nije najbolje rješenje ovog problema. Rješenje bi vrijedilo potražiti u nekim algoritmima za raspoznavanje koji jednako tretiraju sve primjere za učenje, na primjer SVM – Support Vector Machines (mehanizam potpornih vektora). Nadalje, ručnim pregledavanjem testnog seta podataka, primijetio sam da se pogrešne detekcije učestalo pojavljuju na licima ukošenima u desno (iz perspektive osobe na slici), dok je kod ravno postavljenih lica puno veća uspješnost. Kako je GENKI set vrlo raznolik i sadrži slike koje su gotovo iz profila i neke vrlo jako nakošene, a ne garantira nam uravnoteženost takvih slika, moguće je da je to produkt ne savršenog seta podataka za učenje. Autori [2] su imali dosta bolje rezultate, osim što su koristili drugi algoritam za detekciju, iz razloga što su imali set podataka od 63000 slika, što je više od 16 puta veći broj od meni raspoloživog seta. Sigurno je da se takva razlika mora osjetiti. Da je riječ u nekim laboratorijskim slikama (DFAT primjerice), znatno manji broj slika bi bio potreban za učenje, ali takav detektor bi naravno i prepoznavao samo slike slikane u takvim laboratorijskim uvjetima. Pošto između pozitivnih slika u takvom setu nema velikih razlika, puno se lakše izluče bitne značajke.

8. Detekcija osmijeha u videu

Za demonstraciju detekcije osmijeha u videu, koristeći izučeni klasifikator, razvio sam aplikaciju koja u realnom vremenu prikazuje video s web kamere računala, te na videu detektira osmijeh.

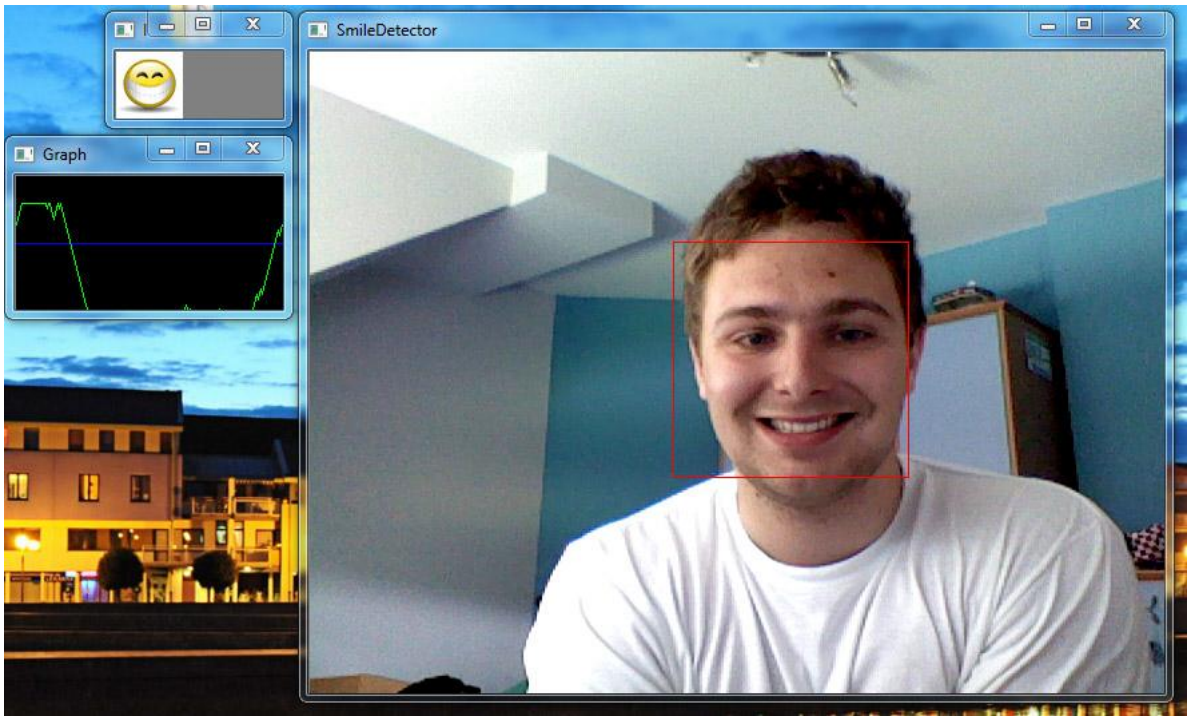
Detekcija se ostvaruje pomoću klasifikatora broj 4, uz malu modifikaciju načina detekcije. Kako se u videu obrađuje oko 15 slika u sekundi, a naš detektor nije savršen, u ponekoj slici se dogodi pogrešna detekcija. I obratno, kod osmijeha se neka slika ne detektira. Takvo ponašanje možemo smatrati šumom detektora. Da bi izbjegli to, potrebno je gledati gustoću detekcija, odnosno da li je detekcija osmijeha kontinuirana kroz nekoliko slika ili je to izolirana detekcija. Izolirane detekcije uglavnom znače pogrešnu detekciju. Isto tako, ako u nizu uspješnih detekcija dobijemo negativnu detekciju na jednoj slici, to ne znači automatski da nema osmijeha, već je vjerojatnije riječ o pogrešci.

Aplikacija ima mjerač koji se kreće u rasponu od 0 do 20, te za se za svaku detekciju poveća za jedan, a kod slike bez detekcije smanji za jedan. Eksperimentalno sam odlučio da je optimalan prag za detekciju osmijeha kada mjerač pređe 12. Ovakvim načinom detekcije smo dobili određeno kašnjenje detekcije jer je potrebno nekoliko uzastopnih slika osmijeha kako bi detektor odlučio o osmijehu. Zauzvrat smo dobili vrlo uspješnu eliminaciju pogrešnih detekcija.



Slika 14. Aplikacija u trenutku kad nema osmijeha. Na grafu se vidi šum, odnosno pogrešne detekcije.

Da ne koristim opisanu metodu, svaki lagani skok na grafu bio bi promjena odluke detektora. Očigledno je da bi takvih odluka previše i detektor ne bi bio uporabljiv.



Slika 15. Uspješna detekcija osmijeha, porast odziva je jasno vidljiva u grafu

Unatoč slabijim rezultatima detekcije slika, na videu klasifikator radi mnogo bolje. Izmjeriti uspješnost na ovaj način je vrlo teško. Uspješnost jako ovisi o uvjetima u kojima je video sniman i poziciji osobe. Detektor sam imao prilike isprobati na svega nekoliko osoba i kod svih je radio zadovoljavajuće. Nažalost, premalena je to brojka za provesti neke statističke metode. Kao i kod detekcije u slikama, ovdje se javlja problem detekcije kada je lice nakošeno u desnu stranu. Takav položaj najčešće rezultira pogrešnom detekcijom. Primijetio sam da i svjetlina prostora u kojem je video nastao jako utječe na detekciju. Konkretno, ukoliko je vrlo svijetla slika, izgubi se kontrast a time i informacija bitna za značajke, te je detekcija otežana.

9. Zaključak

Detektori osmijeha već postoje u raznim komercijalnim proizvodima, međutim vrlo je malo objavljenih radova na tom području. Posljedica je to želja kompanija da zaštite svoje proizvode od konkurencije na tržištu. Iz akademskih krugova, pronašao sam desetak radova koji su se bavili ovom tematikom, ali samo rad autora Whitehill i ostalih [2] dalje rezultate primjenjive u praksi. Ostali radovi su, poput ovog, izabrali smjer istraživanja koji nije polučio najbolje rezultate. Razlozi loših rezultata uglavnom su bili odabir pogrešnog algoritma detekcije i nedovoljan broj slika za učenje.

Kako niti jedan od tih radova nije eksperimentirao s Viola-Jones algoritmom, smatram da je ovaj rad koristan jer dobro pokazuje i argumentira rezultate dobivene tim načinom. Činjenice na koje nas rad upućuje su potreba za velikim i raznovrsnim skupom podataka za treniranje i korištenje algoritma koji jednoliko tretira pozitivne i negativne slike. Za daljnje istraživanje ovog područja preporučio bi algoritam SVM (mehanizam potpornih vektora) jer on ravnopravnije prihvaća pozitivne i negativne slike.

Ispravnost rada detektora od 72,92% smatram vrlo dobrom s obzirom na raznolikost testnih slika. Detektor autora Whitehill i ostalih[2] je na nekim podskupovima GENKI seta imao uspješnost od samo 70-80%, što je u rangu ovog detektora. Međutim, na nekim drugim podskupovima uspješnost je dosezala i preko 95% što je ipak daleko preciznije.

Smatram da je u budućnosti od presudne važnosti izgraditi što veću i kvalitetniju bazu za učenje s označenim slikama osmijeha, jer je to zbog raznolikosti osmijeha među ljudima od ključne važnosti.

10. Literatura

[1] Dr. Sc. Ana Kandare Šoljaga, *Neverbalna komunikacija*, 1.6.2010.

http://www.psiholog-rijeka.com/odnosi/neverbalna-komunikacija/381-1289.aspx

[2] Jacob Whitehill, Gwen Littlewort, Ian Fasel, Marian Bartlett i Javier Movellan, *Developing a Practical Smile Detector,* 18.3.2010.

http://mplab.ucsd.edu/~jake/pami_paper.pdf

[3] Takeo Kanade, Jeffrey Cohn i Ying Li Tian, *Comprehensive database for facial expression analysis*, 18.3.2010.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.8048&rep=rep1&type=pdf

[4] P. Ekman i W. Friesen, *Pictures of facial affect*, Slike, 1976, Dostupno od Human Interaction Laboratory, UCSF, HIL-0984, San Francisco, CA 94143.

[5] P. Viola i M. Jones, *Robusst Real-time Object Detection,* 18.3.2010.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.4987&rep=rep1&type=pdf

[6] Bradski, G. i Kaehler, A., *Learning OpenCV,* prvo izdanje, O'Reilly Media Inc., rujan 2008.

[7] Y. Shinohara, *Facial Expression Recognition using Fisher Weight Maps,* 18.3.2010.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.1442&rep=rep1&type=pdf

[8] Jacob Whitehill, Gwen Littlewort, Ian Fasel, Marian Bartlett i Javier Movellan, T*oward Practical Smile Detection,* 18.3.2010.

http://www.cs.arizona.edu/~ianfasel/homepage/publications_files/WhitehillSmileDetection.pdf

11. Sažetak

DETEKCIJA OSMIJEHA U FOTOGRAFIJAMA

U ovom radu opisana je ideja i postupak izrade detektora osmijeha u slikama i video zapisu. Objašnjena je primjena i potreba za takvim detektorom u današnje vrijeme. Predložen je algoritam Viole i Jonesa za detekciju i opisan je princip njegova rada.

U sklopu rada razvijene su dvije aplikacije za detekciju osmijeha, jedna za rad nad slikama, a druga na videu koji dolazi s web kamere računala. Izrada aplikacija je potpomognuta bibliotekom OpenCV, te su navedeni primjeri u kojima nam OpenCV može pomoći.

Na kraju su prezentirani rezultati uspješnosti aplikacije i dane smjernice za daljnje istraživanje ovog područja.

Ključne riječi: računalni vid, detekcija osmijeha, Viola-Jones, OpenCV

12. Abstract

SMILE DETECTION IN PHOTOGRAPHS

This paper describes an idea and process of development of a practical smile detector, that works on images and video sequences. Also there is an information why is smile detection important nowadays. It's been given a description of suggested Viola Jones algorithm.

As a result of this study, two applications for the smile detection were developed, one for detection on images, and the other for detection on video stream from a webcam. Creating applications was assisted by OpenCV library, and some examples of used OpenCV features are described.

Finally the efficiency results of applications are presented and given guidance for further research in this area.

Keywords: computer vision, smile detection, Viola-Jones, OpenCV