SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 173

DETEKCIJA OSOBA U SLIKAMA

Davor Sutić

Zagreb, lipanj 2008.

Sadržaj

1 Uvod

Prepoznavanje osoba područje je rastućeg zanimanja akademske zajednice i industrije. Neke primjene mogu biti:

- Prepoznavanje ljudskog oblika u budućnosti bi moglo zamijeniti klasične metode označavanja osoba na Web 2.0 servisima. Dovoljno napredan sustav mogao bi predlagati imena osoba na slikama temeljem sličnosti s do sada označenim slikama. Slike postavljene na ove servise obično su namijenjene dijeljenju s ostalim korisnicima i nalaze se u većoj rezoluciji u odnosu na, primjerice, kamere koje nalazimo u video nadzoru.
- Robustan algoritam detekcije osoba koristan je u video nadzoru. Napredni algoritmi pomažu u automatizaciji nadgledanja ili izdvajaju manje količine video zapisa koje se ručno pregledavaju. Pozadina je obično statična i može se učiti, što olakšava detekciju.
- Područje primjene je i analiza ponašanja. Detekcija osoba pred izlogom mogla bi odgovoriti na pitanje gdje i koliko dugo kupci zastaju, te koje proizvode gledaju. Praćenjem osobe može se utvrditi koje je sve skupine artikala kupila.
- Oko 15% smrtnih slučajeva u prometu događa se u nesrećama koje uključuju pješake (Bu, 2005). Metodama detekcije pješaka i predviđanja moguće nesreće nastoji se ovaj broj bitno smanjiti. Uz nastojanja da se vozila naprave što sigurnijima pri sudaru, detekcija pješaka djeluje preventivno utoliko da pomaže uočiti potencijalnu kriznu situaciju.

Zadaća detekcije pješaka u prometu težak je problem u odnosu na prije nabrojene. U vrijeme pisanja ovog rada u vozila se ne ugrađuju sustavi koji bi pomagali vozaču temeljem računalnog vida. Eksperimentalni sustavi spomenuti su u poglavlju o bliskim radovima. Povećanje dostupne računalne snage i novi algoritmi daju naslutiti da bi se upotrebljiv sustav mogao uskoro pronaći i za ovu uporabu. Tema je ovog rada detekcija osoba u slikama. Kao uže područje rada obrađena je detekcija *pješaka* u slikama, jer je detekcija osoba u prometu najteža i najčešće istraživana primjena, pa je uspjehe s ovog područja lakše prenijeti na druga područja, nego obratno.

1.1 Detekcija pješaka u prometu

Zahtjevi na uporabljiv algoritam detekcije pješaka su:

- Za uporabljiv sustav traži se velika pouzdanost, mali broj krivih detekcija (eng. *false positives*)
- U prometu se pješaci susreću na slikama s velikim brojem pozadina, vremenskih uvjeta i uvjeta osvjetljenja
- Različite udaljenosti i brzine kretanja pješaka za koje sustav treba raditi

Sustav koji bi zadovoljavao navedene kriterije u trenutku pisanja rada nije stavljen u uporabu, pa je ovo aktivno područje istraživanja. Pristup se ne ograničava samo na računalni vid, već se koristi čitav niz senzornih sustava:

a) Piezoelektrični senzor

Senzori se ugrađuju u pod i električni signali se generiraju u ovisnosti o pritisku koji stvara osoba koja se nalazi na podlozi. Uporabu su pronašli na pješačkim prijelazima, gdje omogućuju detekciju pješaka koji čekaju ili prolaze kolnikom. Ova vrsta senzora omogućuje veliku pouzdanost i ne zahtijeva kompleksno procesiranje signala (Bu, 2005). Pristup se navodi zbog kompletnosti izlaganja, a ima malo sličnosti s pristupom ovog rada.

b) Podzvučni senzori

Senzori emitiraju podzvučne valove koji se reflektiraju na pješacima i vraćaju u primaocu. Na temelju reflektiranog signala računa se veličina, brzina i udaljenost objekta. Moguće je detektirati pješake na udaljenosti od oko 10m. Pokazuje se da na uspjeh detekcije bitno utječu vremenske prilike, te materijal odjeće pješaka (Bu, 2005).

c) Mikrovalni radar

Oruđe detekcije ovog senzora je elektromagnetski val odašiljan s mikrovalne antene. U usporedbi s procesiranjem računalnog vida, signale je manje zahtjevno obrađivati računalom. Za razliku od podzvučnih senzora, ova vrsta detekcije manje ovisi o

vremenskim prilikama kao što su vlaga ili pritisak zraka. Radar može biti ugrađen u unutrašnjost vozila bez narušavanja vanjskog dizajna.

d) Laser

Infracrvena laserska zraka se emitira, reflektira i vraća u senzor. Tehnologija je vrlo precizna u usporedbi s drugim metodama i omogućuje precizno mjerenje udaljenosti (do u red veličine cm). Vozilo se oprema sa više senzora i procesirani podaci stvaraju višedimenzionalnu sliku predmeta. Visoka preciznost lasera omogućuje visoku rezoluciju dobivenih podataka. Budući da spada u red optičkih senzora, na ovu vrstu prikupljanja podataka utječu vremenske prilike kao što su kiša ili snijeg, te zahtijeva više procesorske snage u odnosu na prethodne metode.

e) Računalni vid

Koristeći video kameru prikuplja se vizualni zapis okoline. Dva pristupa obradi slike su algoritmi koji prepoznaju pješake temeljem specifičnosti ljudskog hoda, te temeljem obrisa ljudske siluete. Prvi pristup u nizu slika traži značajke ritmičkog kretanja karakterističnog za hod pješaka. Mane ovog pristupa su nemogućnost detekcije statičnih osoba ili osoba koje se kreću na netipičan način – primjerice skaču. Nadalje, potrebna je vidljivost nogu i stopala, budući da se detekcija najčešće vrši na temelju njihovih kretnji. Potrebno je duže vrijeme procesiranja jer se koristi niz uzastopnih slika, što odgađa prepoznavanje do obrade zadnje slike. U nastavku će više biti rečeno o pristupu koji se temelji na karakterističnom obliku pješaka.

1.2 Izazovi prepoznavanja pješaka temeljenog na obliku siluete

Privlačna prednost ovog smjera obrade je mogućnost detekcije pješaka u pokretu i statičkih pješaka. Potrebno je pronaći i kodirati ona svojstva oblika koja se pojavljuju kod svih ili velike većine pješaka, vodeći računa o računalnoj snazi potrebnoj da se prepoznavanje izvrši.

Oblici koje treba detektirati mogu se uvelike razlikovati u ovisnosti o stavu pješaka, položaju ruku, odjeći i osvjetljenju. Raznolikost oblika koje treba detektirati pridonosi

povećanju neispravno pozitivnih detekcija. Također, budući da su svojstva raspršena po velikom dijelu slike i teško ih je jednostavno opisati, ova vrsta prepoznavanja donosi velik broj značajki koje treba ispitati, što direktno uzrokuje potrebu za većom računalnom snagom.

1.2.1.1 Raznolikost oblika pješaka

Teško je pronaći i opisati jedinstvenu osobinu koju posjeduju svi pješaci. Ljudi mogu imati različite boje kože ili odjevnih predmeta. Pažnja istraživača stoga je usmjerena na oblik siluete osobe. Teškoće koje se javljaju promatrajući oblik mogu donijeti objekti kao što su torbe ili druge stvari koje osoba nosi. No i bez dodatnih objekata, ljudsko tijelo dolazi u velikom mnoštvu oblika: od pretilih do mršavih, visokih i niskih.

Slika 1. Tri različita pješaka različitih oblika

1.2.1.2 Raznolikost veličina

Uz odstupanja zbog visine osobe, osoba može biti manja jer je udaljena od kamere. Kako se pokazalo teškim kodirati promjene veličine pri treniranju prepoznavanja, većina algoritama koristi pristup da pri treniranju koristi fiksnu veličinu, a pri detekciji skalira sliku po potrebi. Pri tome je lakše sliku skalirati na manju veličinu, a problem može nastati ako je slika pješaka manja od trenirane veličine, kada je potrebno skaliranje slike naviše.

Slika 2. Različite veličine pješaka

1.2.1.3 Položaji tijela

Dodatni izazov predstavlja različitost položaja tijela, ruku i nogu. U idealnom slučaju objektu se dozvoljava samo jedan standardni položaj. Jasno je da u primjenama, osobito u prometu ili nadgledanju ovakva restrikcija nije izvediva ni moguća.

Slika 3. Pješaci s različitim položajima tijela

1.2.1.4 Osvjetljenje

Osvjetljenje predstavlja osobiti izazov pri detekciji u prometu, gdje je moguće da je detektor pokretan i time prolazi kroz različite zone osvijetljenosti. Pri tome valja voditi računa ne samo o mijenjanju osvjetljenja na cijeloj slici, već i na to da se osvjetljenje može promijeniti i na dijelu slike. Potrebna je normalizacija slike koja će uvesti što manju osjetljivost i na lokalne i na globalne (preko cijele slike) varijacije svjetla.

Slika 4. Različita osvijetljenost pješaka

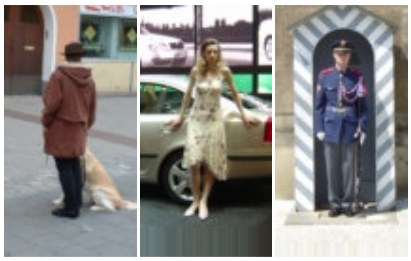
1.2.1.5 Mijenjanje točke gledišta

Za sustave koji prate pješaka bitno je imati što precizniju informaciju i kada se pješak i/ili kamera (koja se možda nalazi u automobilu) gibaju. Ovaj problem veže se na prijašnje probleme skaliranja, primjerice kada pješak dolazi prema kameri, te problema položaja tijela, ako se pješak rotira u odnosu na kameru.

Slika 5. Različite točke gledišta pješaka

1.2.1.6 Varijacije pozadine

Pozadina može biti slična pješaku i na taj način detektirana kao dio njegovog oblika, deformirajući ga. Kada je kamera stacionirana, pozadina se može učiti i tako učinkovito eliminirati. Iz već rečenog je jasno da se na navedeno u većini primjena ne može računati. U primjeni se teškim pokazuju pozadine s puno detalja.



Slika 6. Različite pozadine pješaka

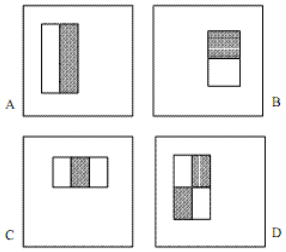
1.3 Povezani radovi

Jedan od najpoznatijih radova ovog područja je (Viola, 2001). Viola i Jones detektiraju lice koristeći algoritam učenja baziran na *AdaBoostu* kako bi izdvojili kritične značajke i ostvarili brzo procesiranje. Ovdje se ukratko predstavljaju značajke koje koriste Viola i Jones, kako bi se opisala alternativa značajkama korištenim u ovom radu.

Najjednostavniji način provjere slike vodio bi zacijelo provjeri vrijednosti pojedinih točaka slike. Ovaj pristup ima više značajnih nedostataka: čak i unutar normaliziranih slika postoje značajna odstupanja u položaju pojedinih objekata. Glava osobe može biti pomaknuta više u neku stranu, pa će tako na jednoj slici isti piksel biti dio glave osobe, a na drugoj opisivati pozadinu. Druga značajna mana odnosi se na razliku u intenzitetu. Istu osobu u različitim uvjetima osvjetljenja opisivat će slike bitno različitih intenziteta tako da

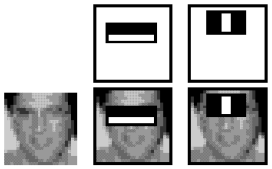
je teško postaviti odlučan uvjet na intenzitet piksela. O ovim problemima bit će više riječi u nastavku. Jedno od rješenja nude Viola i Jones u svom originalnom radu.

Značajke koje se koriste za detekciju su razlike u površinama pravokutnika. Slika 7 ilustrira tipove značajki. Vrijednost svake značajke je razlika između zatamnjenog i svijetlog dijela pravokutnika. Tako se iznos značajke tipa A dobiva tako da se zbroje intenziteti točaka tamnog dijela, te se od tog iznosa oduzme zbroj točaka svijetlog dijela.



Slika 7. Ilustracija tipova značajki

Slika 8 prikazuje primjenu značajki. Slike 7 i 8 preuzete su iz rada (Viola, 2001). Dio pravokutnika koji pada na čelo detektiranog lica ima manje intenzitete točaka od dijela pravokutnika koji pripada očima. Ova značajka uspješno opisuje svojstvo ljudskog lica da su je područje očiju tamnije od područja čela.



Slika 8. Pronađene značajke

Odgovor na postavljene probleme pred odabrane značajke autori su riješili na sljedeći način: značajke koje oni koriste su *delokalizirane* i temelje se na *razlici zbroja intenziteta*. Time što pravokutnici pokrivaju puno veću površinu od pojedinog piksela dobiva se na fleksibilnosti. Primjerom iz detekcije osoba, čak i ako je glava osobe koju želimo detektirati pomaknuta u odnosu na standardnu poziciju, pravokutnici mogu obuhvatiti dovoljno veliko područje, tako da se ona ipak pronađe u značajki. Drugi izazov koji se tiče razlika u osvjetljenju ove značajke rješavaju time što se promatra razlika u zbrojevima intenziteta, a ne same intenzitete. Primjerom iz detekcije lica, čak i ako je lice osvijetljeno ili u sjeni, regija očiju će biti tamnija od regije lica.

Predstavljene značajke mogu se metodom integralne slike izračunati u konstantnom vremenu. Metoda će detaljnije biti objašnjena u narednim poglavljima. Nedostatak značajki je njihov fiksni pravokutni oblik zahtjevan zbog brzog izračunavanja. Teško se kodiraju kose površine ili eliptični oblici.

U prilagodbi vlastitog klasičnog rada za detekciju pješaka (Viola, 2005), Viola i Jones uz navedene značajke koriste i uzorke kretanja. Rad je namijenjen detekciji pješaka za

potrebe nadgledanja i prilagođen je niskim rezolucijama pripadajućih kamera. Ključna novost je prilagodba postojećih značajki za detekciju pokreta.

Za svaku sliku računa se pomak u odnosu na prethodnu u 4 glavna smjera: gore (G), lijevo (L), dole (D) i desno (R), što zajedno sa slikom koju dobijemo jednostavnim oduzimanjem trenutne i prethodne slike (TP) čini 5 slika. Tako se slika G dobije da se svi pikseli trenutne slike pomaknu za jedan piksel prema gore i oduzmu od prošle slike. Ovako se oponaša moguće kretanje pješaka prema gore u slici: ako se pješak kreće prema gore razlike će biti manje nego ako se, primjerice, kreće prema dole. Očekuje se da će dobivene slike biti tamne, osim na dijelovima gdje postoji pomak. Nad postojećim slikama uvode se tri nove vrste značajki:

1) Značajke koje zapisuju razliku u pravokutnicima slike TP i svake od slika G, L, D i R.
2) Značajke koje računaju iznose pravokutnika u svakoj od slika TP, G, L, D i R.
3) Značajke koje računaju razlike između dva pravokutnika na slici, gdje je slika jedna od G, L, D ili R.

Razlika značajki 2) i 3) je u tome što značajka 2) u obzir uzima zbroj intenziteta unutar pravokutnika, dok značajka 3) tom zbroju oduzima intenzitete drugog pravokutnika.

Ovom relativno jednostavnom pretragom proširila se metoda osnovnog rada na kretanje pješaka, pa tako ovaj rad obuhvaća i statičke i dinamičke informacije. Radom (Viola, 2005) postignuto je prepoznavanje od 90% pješaka uz red veličine krivo pozitivnih detekcija 10^{-5}. Prosječno vrijeme obrade slike je 0.25s za sliku veličine 360x240, na računalu brzine 2.8GHz.

Istaknut je rad (Papageorgiou, 99), gdje autori koriste računalni vid za ostvarivanje visokog stupnja detekcije uz brzinu obrade od 10Hz. I ovdje se koriste iste značajke kao i u prethodnim radovima, ali algoritam učenja nije više AdaBoost, već SVM (eng. *Support vector machine*) klasifikacija. Postiže se postotak od oko 90% detekcije pješaka uz 10^{-4} lažno pozitivnih detekcija. Sustav je integriran u testna vozila kroz projekt poznat kao DaimlerChrysler Urban Traffic Assistant (UTA).

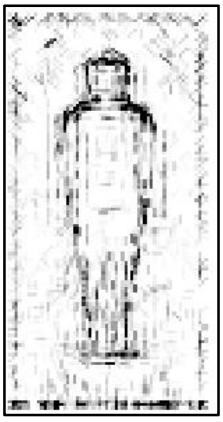
2 Izrada klasifikatora

U ovom poglavlju opisuje se teorijska podloga za treniranje klasifikatora. Dva su ključna pitanja pitanja pri dizajnu i izradi klasifikatora: kako odabrati značajke? Kako iz pojedinačnih značajki napraviti klasifikator? Tragom ovih pitanja, izrada klasifikatora opisuje se u više dijelova. Prvi dio obrađuje odabir značajki, te objašnjava zašto se koriste dvije vrste značajki. Drugi dio posvećen je algoritmu učenja.

2.1 Značajke

Kao što je do sada rečeno u uvodu, posljednjih godina najpopularnije značajke temelje se na sumama intenziteta pravokutnika. Ove značajke su poznate i kao Haarove značajke. Pristup koji se radom istražuje je prepoznavanje pješaka na temelju oblika. Nastojao se odabrati takav pristup značajkama koji će omogućiti opis oblika pješaka uz što manje računanja. Ovaj rad otklanja se od većine ostalih koristeći jednostavne značajke temeljene na intenzitetima *derivacija*, te stvarajući iz njih nove značajke. Kako se u radu koriste dvije vrste značajki, korisno ih je na početku razlaganja različito imenovati. Značajke niže razine nazvane su *pokazivači* (eng. pointers), zato što za određenu točku pokazuju iznos derivacije slike. Druga vrsta značajki nazvana je *obris* (eng. *shapelet*), jer kombinira značajke niže razine, pokazivače, kako bi opisala komadić oblika pješaka.

Objašnjenje značajki lakše je razumljivo kada je poznato što se s njima želi postići. Krajnji rezultat je klasifikator vizualiziran na slici 9. Slika je preuzeta iz (Sabzmeydani, 2007) . Tamna područja na slici opisuju karakterističan izgled pješaka, naučen od strane sustava. Testiranje dijela slike (*prozora*) koje odgovara na pitanje da li je prisutan pješak teče ovako: na zatamnjenim područjima provjeravaju se intenziteti piksela. Za svaki piksel postavlja se pitanje: da li je intenzitet piksela testirane slike veći od granice postavljene za taj piksel. Ukoliko je odgovor za *dovoljan broj* piksela potvrdan, slika se klasificira kao pješak. Ovo je pojednostavljen prikaz testiranja, koje će detaljnije biti objašnjeno u narednim poglavljima.

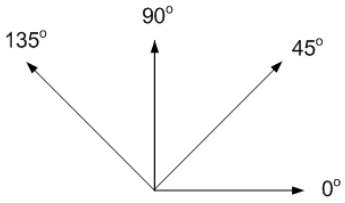


Slika 9 Ilustracija klasifikatora

2.1.1 Temeljne značajke: pokazivači

Proces računanja temeljnih značajki teče kako slijedi:

1) Prvo se slika prebaci u crno-bijeli (eng. *grayscale*) oblik, koji ćemo zapisati kao G.
2) Računa se promjena piksela u 4 smjera. Nastale slike u ostatku teks ta navode se kao derivacije slike G. Primjerice, za smjer 1 koji je pod kutem od 0^o, računa se nova slika D_1, tako da je:



Slika 10. Smjerovi u kojima se računaju derivacije slike

$$D_1(x,y) = |G(x+1,y)-G(x-1,y)| \quad (1)$$

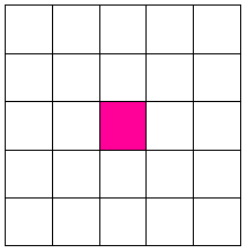
Izrazi za ostale smjerove su kako slijedi:

$$D_2(x,y) = |G(x+1,y-1)-G(x-1,y+1)| \quad (2)$$

$$D_3(x,y) = |G(x,y-1)-G(x,y+1)| \quad (3)$$

$$D_4(x,y) = |G(x-1,y-1)-G(x+1,y+1)| \quad (4)$$

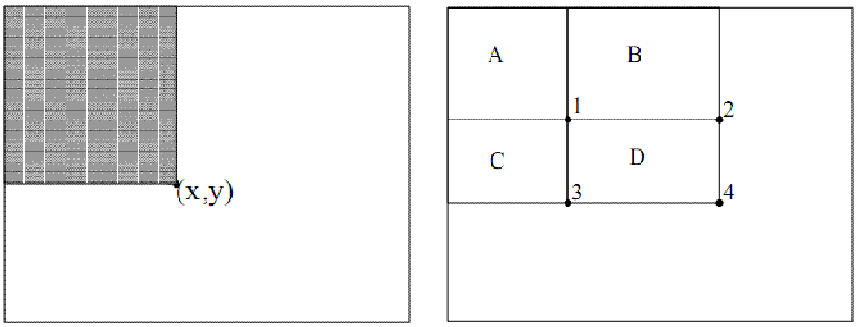
3) Za svaki piksel pojedine derivacije računa se nova vrijednost tako da se zbroji okolnih 25 piksela, i rezultat podjeli s 25. U praksi se pokazalo da ovaj postupak proširi rubove dobivene prethodnim postupkom. Kako će se kasnije pokazati, ovo je veoma važno za rezultate klasifikatora. Ovim postupkom nastoji se izbjeći prva prepreka pred značajkama koje rade s pikselima: manje varijacije u normaliziranim slikama. O ovome je više rečeno u uvodu, u dijelu o odabiru značajki za rad (Viola, 2001).


Slika 11. Označen središnji piksel i susjedi koji utječu na njegovu novu vrijednost

4) Provodi se postupak *lokalizirane* normalizacije. Intenziteti se normaliziraju s obzirom na lokalne susjede, čime se prevladava druga zamjerka značajki temeljenih na pikselima: razlike u osvjetljenju. Ako je jedan dio pješaka bolje osvijetljen, a drugi primjerice u sjeni, ovim postupkom taj će se efekt umanjiti. Kasnije je pokazano kako normalizacija utječe na rezultate algoritma. Normalizacija se provodi tako da se svi pikseli pripadnici iste regije pomnože s odgovarajućom konstantom. Točnije, prema formuli:

$$Ds(x,y) = \frac{Ds(x,y)}{\sqrt{(\sum Ds(i,j) * Ds(i,j) + 1)}} \quad (5)$$

Izbor veličine regije opisan je kasnije, pri opisu ostalih parametara algoritma. Sve regije su iste veličine i oblika kvadrata. Pri računanju zbroja kvadrata intenziteta u pikselima regije (suma u nazivniku formule 5) koristi se metoda popularizirana u (Viola, 2001) nazvana integralna slika. Ideja ove metode je da se za svaki piksel (x,y) pamti suma intenziteta u površini pravokutnika (0,0)-(x,y). Tu sumu nazivamo integralom I(x,y). Ove vrijednosti koriste se kako bi se intenziteti na površini bilo kojeg pravokutnika na slici dobili u konstantnom vremenu. Površina pravokutnika D na slici 12 je $I_4+I_1-I_2-I_3$.



Slika 12. Integralna slika

Slika 13 prikazuje međukorake pri stvaranju normalizirane derivacije. Za svaku sliku stvaraju se 4 takve derivacije koje odgovaraju 4 osnovna smjera navedena u slici.

Slika 13. Slike u postupku stvaranja normalizirane derivacije

Pokazivač je jednostavna značajka koja se sastoji od pozicije, smjera, iznosa (θ) i predznaka (p). Za svoju poziciju, pokazivač je značajka koja kaže da je slika vjerojatnije pješak, ako je njena normalizirana derivacija $D_d(x,y)$ na toj poziciji veća od θ. Predznak

pokazivača p određuje znak usporedbe, pa u ovisnosti o τ pokazivač može biti jednak tvrdnji da je iznos derivacije manji od θ. Može se zaključiti da je τ binarna varijabla.

$$P_{(x,y,d)} = (\tau, \theta) \quad \equiv \quad D_d(x,y) > \theta \rightarrow Slika\ je\ pješak \tag{6}$$

Ovdje se definira svojstvo da je slika *istinita* prema pokazivaču ako i samo ako je pokazivač ispravno klasificira. Pokazivač ispravno klasificira sliku ako vrijedi relacija (7). Ako slika nije istinita za pokazivač, tada je *neistinita* u odnosu na njega.

$$h(x,y,d) = \begin{cases} 1 & p_t\, D_d(x,y) > p_t\theta_t \\ 0 & inače \end{cases} \tag{7}$$

2.1.2 Značajke druge razine: obrisi

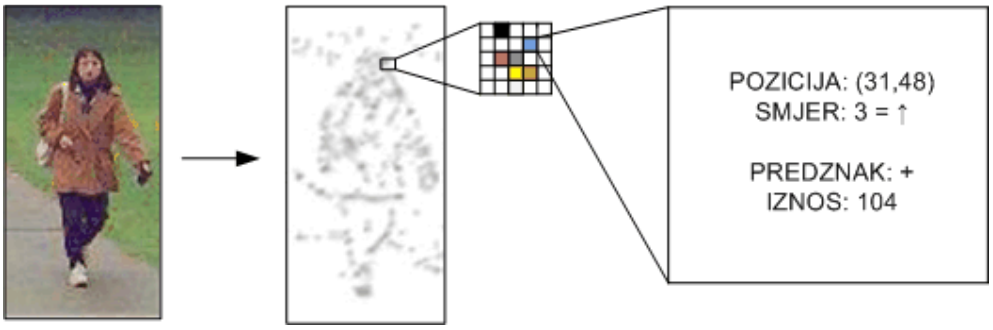
Značajke druge razine definiraju se kao niz temeljnih značajki, tako da je svakoj nižoj značajki pridružena određena *snaga*. Snaga pokazivača brojčano opisuje koliko je pokazivač dobar u razdvajanju dviju klasa: pješaka i nepješaka. Što je veća snaga pokazivača, više slika je istinito, a manje neistinito za taj pokazivač. Snagu pokazivača određujemo AdaBoost algoritmom. Budući da jedan pokazivač izdvaja tako malo informacija iz slike, ne može se očekivati da će dobro klasificirati slike. Kasnije će se, u dijelu o AdaBoost algoritmu, pokazati da dovoljnim da pokazivači klasificiraju slike tek nešto bolje od nasumične klasifikacije, primjerice 51% u odnosu na slučajnih 50%. Zato se pokazivači grupiraju u *obris*, koji iz određene podslike izdvaja pokazivače koji najbolje klasificiraju slike. Obrisi opisuju veću količinu lokalnih informacija i postižu bolje rezultate klasifikacije.

Neka je α snaga pokazivača. Klasifikator pojedinog obrisa ima formulu:

$$\mathrm{H(x)} = \begin{cases} 1 & \sum_{\mathrm{t=1}}^{\mathrm{T}} \alpha_t h_t(x) \geq 0 \\ 0 & \text{inače} \end{cases} \tag{8}$$

Koje pokazivače obrisi mogu uključiti određeno je njihovim podprozorom. Pri treniranju, slika se dijeli na podprozore veličine 5x5, 10x10 ili 15x15. Svaki obris pripada jednom podprozoru i samo iz njega uključuje pokazivače. U segmentaciji gdje je prozor veličine 5x5, mogući broj pokazivača je 5x5x4, po jedan za svaki smjer (4 smjera, po jedan za svaku derivaciju). Jasno je da svi pokazivači neće imati jednaku snagu, pa se u obris uključuju

samo oni pokazivači s najvećom snagom. Broj pokazivača parametar je algoritma, a implementacija u (Sabzmeydani, 2007) savjetuje odabir $\sqrt{5 * 5 * 4}$ najboljih pokazivača. Fleksibilnija implementacija dopušta onim obrisima koji imaju pokazivače velike snage da uključe više pokazivača.



Slika 14. Prikaz slike, jednog obrisa i jednog njegovog pokazivača

Zašto konstruirati novu vrstu značajki? Obrisi ne crpe nove informacije sa slike, već organiziraju i evaluiraju informacije pohranjene u pokazivačima. Jednostavan pristup bio bi algoritam učenja upogoniti sa svim pokazivačima, te na taj način izabrati najbolje. Kao što se navodi u (Sabzmeydani, 2007), protiv ovog pristupa postoje dvije ozbiljne zapreke. Prva se odnosi na vremensku složenost takvog rješenja. Druga i važnija zapreka odnosi se na algoritam učenja. AdaBoost iz podataka upotrebljenih za trening izvlači minimalnu količinu informacija da bi se izvršila klasifikacija. Mnoge značajke koje su na treningu prosječne ili male važnosti mogu tako biti gotovo zanemarene, a u praksi se pokazuje da na testnom skupu one mogu odigrati bitnu ulogu. AdaBoost bi se mogao koncentrirati na mali broj značajki koje dobro klasificiraju podatke za treniranje i pri tom zapostaviti velik broj drugih značajki koje su bitne za testni skup.

Ovim pristupom iz svakog podprozora se izvlači određen broj značajki, pa se nameće globalni pristup: ne može se dogoditi da konačne značajke budu usko lokalizirane na nekoliko podprozora. Kako svi podprozori ne sadrže jednako vrijedne informacije, drugim krugom AdaBoosta ocjenjuju se obrisi, kako bi se pronašla ona lokalna područja koja najbolje opisuju obrise pješaka. Pristup se može sažeti ovako: značajke se crpe iz cijele slike, a potom veća težina daje onima koje su korisnije.

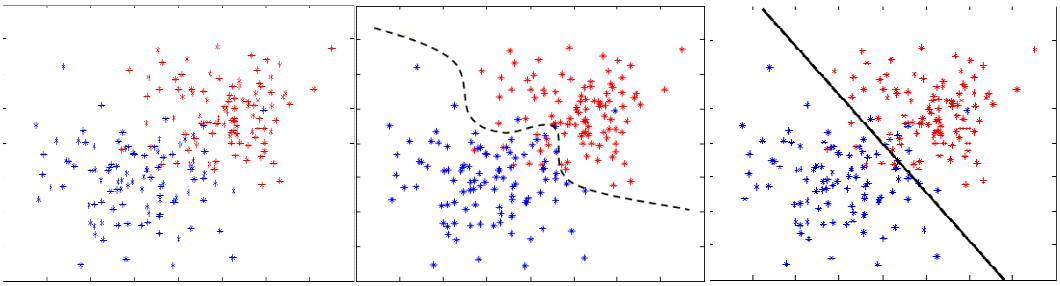
Konačni klasifikator dobije se tako da svaki obris daje brojčanu ocjenu slike – pozitivan broj znači da je slika pješak, a negativni da nije. Ocjena obrisa, osim što govori o klasifikaciji slike, nosi i ocjenu sigurnosti obrisa, budući da veći broj znači veću sigurnost da je slika pješak. Ocjena obrisa množi se njegovom težinom, pa se konačna odluka donosi temeljem težinskog „glasovanja“ obrisa.

2.2 AdaBoost

Problem detekcije pješaka svodi se na odgovor na pitanje: da li je na zadanoj slici pješak? Programi koji taj problem rješavaju su *klasifikatori* koji svaku sliku svrstavaju u jednu od dvije kategorije: pješak ili nepješak.

Klasifikator je funkcija koja ulaze $x_i \in X$ svrstava u klase, $y_i \in Y$. Iako općenito može biti više klasa kao izlaz funkcije, ovdje se razmatraju klasifikatori s dva izlaza, obično predstavljena s $Y = \{-1,\ 1\}$. Također, razmatrani klasifikatori su linearni. Dobar uvod u ovo područje predstavlja (Polikar, 2006). Ovdje je dan uvod u klasifikatore u kratkim crtama, te poseban osvrt na AdaBoost. Riječi klasifikator i hipoteza ponekad se koriste izmjenično, te u daljnjem tekstu označavaju iste pojmove.

Jednostavan primjer klasifikacije ilustrira slika 15. Potrebno je pronaći liniju koji najbolje razdvaja dva skupa koji se preklapaju. Rješenja s desne strane prikazuju neke od mogućih klasifikatora: linearne i nelinearne.

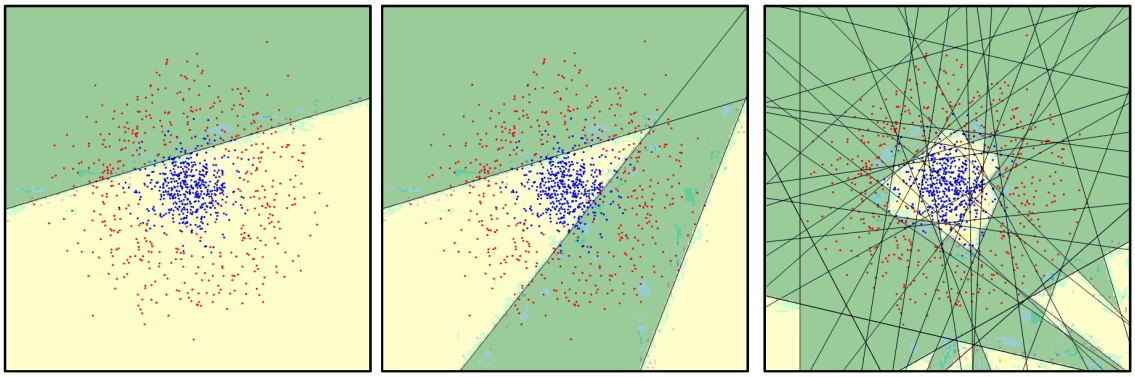


Slika 15. Testni skup i dva moguća klasifikatora – nelinearni i linearni

Moderna povijest klasifikatora počinje radom (Dasarthy, 1979), koji razmatra podjelu prostora značajki koristeći dva ili više klasifikatora. Hansen i Salamon su u (Hansen, 1990) pokazali da se izvedba neuronskih mreža može poboljšati koristeći klasifikatore slično konfiguriranih neuronskih mreža. Iste godine je Schapire radom (Schapire, 1990) razradio

teoretske osnove izrade snažnog klasifikatora iz niza slabijih, čime je postavio teoretske osnove AdaBoosta.

AdaBoost je općenit algoritam strojnog učenja. Cilj je algoritma napraviti takav klasifikator koji će osigurati najbolje razdvajanje dvije klasa, uz minimalan račun. Od 1997, kada su ga Freund i Schapire objavili, ovaj algoritam privukao je osobito veliku pažnju istraživača.



Slika 16. Prikaz klasifikacije točaka AdaBoost algoritmom za 1, 2 i 40 pravaca

Princip rada jednostavno ilustrira primjer autora algoritma, naveden u (Freund, 1999). Kladioničar na utrke konja nada se postići što veći prihod, pa odluči napraviti program koji će točno predviđati pobjednika utrke na temelju dostupnih informacija: broj utrka u kojima je konj prije pobjeđivao, koeficijenti drugih kladionica itd. Pri stvaranju programa, kladioničar pita stručnjake koji se uspješno klade da mu objasne svoju strategiju. Stručnjaci ne mogu ponuditi jednu strategiju koja bi uvijek bila točna, ali opremljeni s dovoljnim brojem podataka mogu odlučiti na kojeg konja će se kladiti, te odgovoriti zašto baš na njega: „Kladi se na konja koji je u posljednje vrijeme osvojio najviše utrka“ ili „Kladi se na konja koji ima najmanji koeficijent“. Iako je nevjerojatno da će takav odgovor poslužiti predviđanju svih utrka, razumno je pretpostaviti da će takav naputak poslužiti bolje od nasumičnog odabira. Velikim brojem podataka kladioničar može prikupiti proizvoljan broj savjeta. Da bi iskoristio ove naputke pri stvaranju svog programa, kladioničar želi odgovoriti na dva pitanja:

- Koji savjeti su dobri, a koji ne? Kladioničar za svoju bazu utrka ima i konačne ishode, te može provjeriti koliko je pojedini savjet dobar uzimajući u obzir ukupno sve utrke.

- Kako kombinirati savjete u jedan sustav odlučivanja visoke preciznosti? Kladioničar se odlučuje koristiti AdaBoost.

AdaBoost pripada generičkoj skupini algoritama koji stvaraju precizno pravilo predviđanja (klasifikacije) koristeći više hipoteza malog stupnja preciznosti. U navedenom primjeru, AdaBoost će ispitati kvalitetu svakog savjeta koji je kladioničar prikupio, te svakom savjetu odrediti težinu.

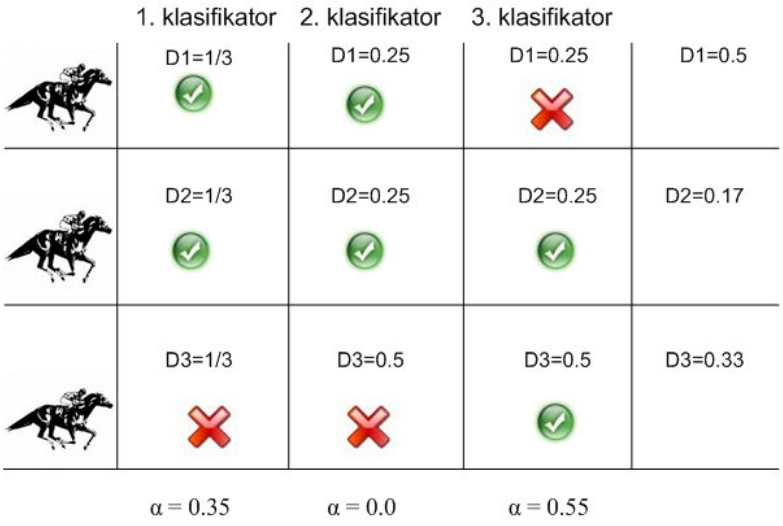
$$\alpha = \frac{1}{2} * \ln\left(\frac{1-\epsilon}{\epsilon}\right) \qquad (9)$$

Težina svakog savjeta određuje se u skladu s formulom (9), gdje je ϵ greška slabe hipoteze (savjeta). Konačno pravilo odlučivanja uzet će u obzir sve savjete u skladu s njihovom težinom. Detalji i numerički odnosi se preskaču kako bi se algoritam prikazao što je jednostavnije moguće. Pseudokod algoritma ja kako slijedi:Inicijaliziraj skup greški $D_1(i) = 1/m$, gdje je m broj testnih podataka
Za t od 1 do T:

- Napravi slabi klasifikator nad testnim podacima, uzimajući u obzir greške D_t
- Izračunaj grešku koju klasifikator radi
- Pridijeli klasifikatoru težinu α u ovisnosti o grešci – za manju grešku pridijeli veću težinu, prema formuli (9)
- Izračunaj novu distribuciju grešaka D_{t+1}, tako da
    - Testnim primjerima gdje trenutni klasifikator griješi povećaš težinu za faktor e^{α}
    - Testnim primjerima koje klasifikator dobro klasificira smanjiš težinu za faktor $e^{-\alpha}$

Konačna hipoteza je suma ishoda svih slabih klasifikatora pomnoženih s vlastitom težinom

Ključno svojstvo algoritma je održavanje distribucije pogrešaka, te stvaranje novih klasifikatora u skladu s ovom distribucijom. Primjerom se ovo može pokazati ilustrirati na sljedeći način: one utrke za koje savjeti stručnjaka točno predviđaju ishod u idućem koraku će dobiti manju težinu, a one gdje je ishod netočno prognoziran veću težinu. To će prisiliti idući klasifikator da se usmjeri prema onim testnim primjerima koje je prethodnik loše klasificirao.



Slika 17. Ilustracija AdaBoost algoritma primjerom

Slika 17 ilustrira detalje algoritma. Kladioničar je prikupio 3 savjeta: prvi savjet točno predviđa ishode prve i druge trke, a netočno ishod treće trke. Drugi savjet je identičan prvom, a treći točno savjetuje za posljednje dvije trke. Pretpostavimo za svrhe primjera da se radi binarnoj klasifikaciji, pa se odluka donosi samo između dva konja. U tom slučaju, svi klasifikatori imaju točnost od 2/3. Budući da su samo dva ishoda, ako su oba približno jednako vjerojatna točnost nasumičnog klasifikatora je 1/2.

Algoritam teče kako slijedi:

1. Klasifikator ima jednoliku distribuciju grešaka, $D_i = 1/3$. Njegove sposobnosti predviđanja su bolje od nasumičnog pogađanja i pridjeljuje mu se težina $\alpha=0.35$. U skladu s predviđanjima podešava se distribucija grešaka: utrke gdje

je točno prognoziran ishod dobile su manju težinu, a utrka u kojoj je ishod krivo prognoziran veću težinu.

2. Klasifikator više nema jednoliku distribuciju grešaka. Savjet koji je kladioničar dobio dobar je za prve dvije trke, ali greška koju AdaBoost računa je D_3=0.5, jer je to koeficijent treće trke. Greška 0.5 govori algoritmu da se ništa novo nije saznalo (savjet je beskoristan), pa mu se pridjeljuje težina α =0.0. Klasifikator se zanemaruje, pa se tako ni greške ne mijenjaju.
3. Točno se prognozira ishod posljednje dvije trke. Rezultat je težina od α =0.55 i nova distribucija grešaka.

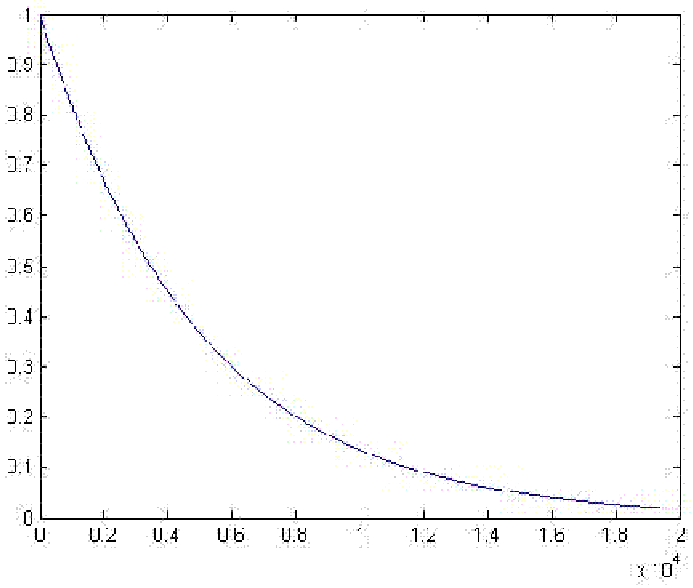
Iz primjera je vidljivo da su u koraku *t* AdaBoosta sve informacije o prethodnim hipotezama zapisane u distribuciji grešaka D.

2.2.1 Teoretska podloga AdaBoosta

Freund i Schapire u radu (Freund, 1997) pokazali su da je greška nad treniranim podacima ograničena sljedećim izrazom:

$$\prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4{\gamma_t}^2} \leq e^{\left(-2\sum_t {\gamma_t}^2\right)} \qquad (10)$$

Pri tome je $\gamma_t = \frac{1}{2} - \epsilon_t$, pa je γ_t mjera za to koliko je slabi klasifikator bolji od nasumičnog pogađanja. Iz izraza (9) slijedi da greška nad treniranim podacima pada eksponencijalno ako su generirane hipoteze bolje od nasumičnog pogađanja. Greška nad testnim podacima ovisi o tome koliko su testni podaci slični podacima korištenim pri treniranju.

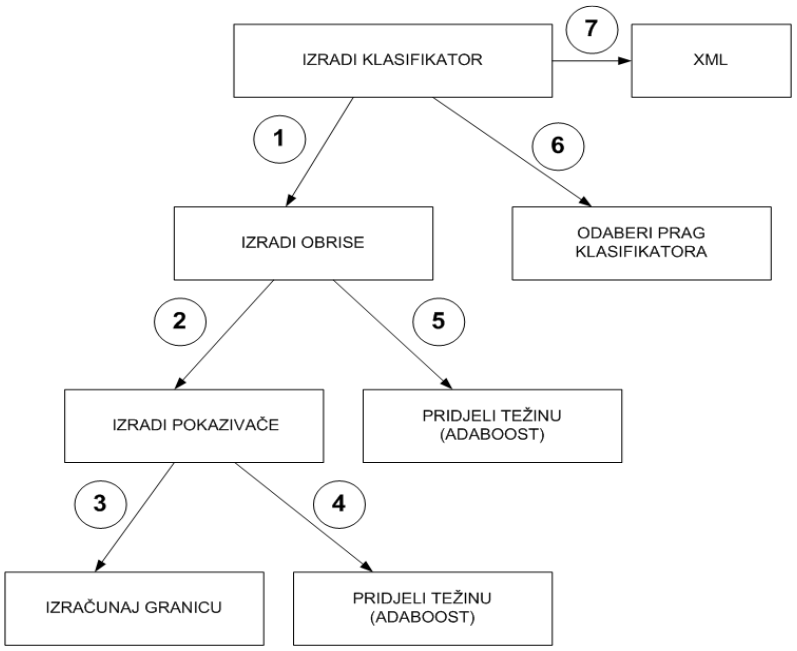


Slika 18. Pad granice AdaBoost greske brojem hipoteza za γ_t konstantno 0.01

Složenost AdaBoost algoritma sadržana je u stvaranju slabih klasifikatora na temelju prošlih grešaka. Ako se pri stvaranju klasifikatora svaki testni podatak razmatra konstantno vrijeme O(1), tada je vremenska složenost algoritma O(TM), gdje je T broj slabih klasifikatora, a M broj primjera nad kojima se vrši treniranje. Prostorna složenost samog algoritma je O(M), budući da sam algoritam zahtijeva tek niz duljine M za distribuciju grešaka.

2.2.2 Implementacija AdaBoosta

AdaBoost je središnji dio ovog rada. Pri treniranju završnog klasifikatora koristi se dva puta: prvi put kako bi se ocijenili pokazivači unutar jednog podprozora i stvorio efikasan obris, a idući put kako bi se usporedili obrisi i dobio završni klasifikator. Sam je tok algoritma generički, pa stoga jednak u oba prolaza. Detaljnije će biti objašnjeno stvaranje slabih hipoteza.



Slika 19. Postupci izrade klasifikatora

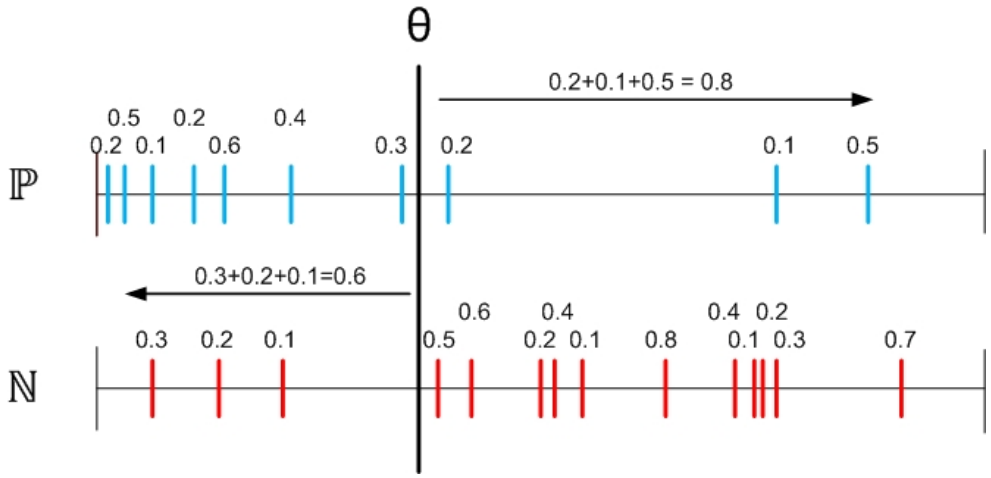
Označimo veličinu podprozora za stvaranje obrisa sa L. Već je rečeno da se u jednom podprozoru nalazi L*L*4 pokazivača. Primjenjuje se algoritam učenja kako bi se ocijenili i izdvojili oni pokazivači koji najbolje razdvajaju dva skupa pješaka i nepješaka. Stvaranje slabih hipoteza jednako je odabiru smjera i iznosa pokazivača. Iznos pokazivača je vrijednost intenziteta slike, pa poprima vrijednosti iz skupa [0,255]. Ukupan broj pokazivača koje treba obraditi je 128*64*4 = 32 768, gdje su 128 i 64 visina i širina slike. Prva implementacija mogla bi biti: za moguću vrijednost pokazivača provjeri koliko slika je krivo klasificirano, što bi značilo da je složenost O(I*M). I je u ovom slučaju 256, a M broj test primjera. Broj test primjera kojima se trenira klasifikator bit će reda veličine 1000, pa je tada ukupan broj operacija 32 768*256*1000 ≈ 10^{10}. Ipak, primjenom odgovarajućeg postupka za odabir iznosa pokazivača broj operacija značajno se skraćuje.

Problem se može definirati kako slijedi: definirajmo dva niza duljina M_p i M_N, $\mathbb{P}$ i $\mathbb{N}$, koji oba sadrže elemente skupa $I = [0,255]$. $\mathbb{P}$ je duljine M_p , a $\mathbb{N}$ duljine M_N . Ako je slika *m* pješak i na lokaciji $D_d(x,y)$ pokazivača ima piksel intenziteta *z,* tada niz $\mathbb{P}$ ima na lokaciji *m* zapisano *z*. Ekvivalentno se definira i niz $\mathbb{N}$ za negativne slike. Definirajmo dva dodatna

niza E_p i E_N koji sadrže težinske koeficijente za svaki element nizova $\mathbb{P}$ i $\mathbb{N}$. Cilj je pronaći takvu granicu θ za koju vrijedi da minimizira grešku zadanu izrazom:

$$\sum_i^{\mathrm{Mn}} \begin{cases} E_N[i] & ako\ \mathbb{N}[\mathrm{i}] > \theta \\ 0 & inače \end{cases} + \sum_i^{\mathrm{Mp}} \begin{cases} E_P[i] & ako\ \mathbb{P}[\mathrm{i}] \leq \theta \\ 0 & inače \end{cases} \tag{11}$$

Grafički se rečeno može predstaviti slikom 20. Niz $\mathbb{P}$ sadrži informaciju o pozicijama pozitivnih elemenata, a niz E_p sadrži koeficijente. Treba pronaći granicu θ tako da zbroj pozitivnih koeficijenata iznad granice i zbroj negativnih koeficijenata ispod granice budu što manji. U primjeru na slici ukupna greška iznosi 0.8+0.6=1.4, i to je najmanja moguća greška.



Slika 20. Odabir praga razdvajanja dvaju skupova

Granica se pronalazi tako da za svaki interval zbroji pozitivne elemente iznad granice i negativne elemente ispod granice. Ključno je da se ovi zbrojevi mogu unaprijed izračunati bez povećanja složenosti, pa se svaka moguća pozicija granice ispituje u konstantnom vremenu. Ukupna složenost općenito je jednaka broju mogućih intervala za granicu, što je jednako broju slika s kojima se trenira algoritam, M. Algoritam i složenost jednaki su kod stvaranja pokazivača i kod stvaranja obrisa.

2.3 Programska implementacija

Prioriteti dizajna aplikacije bili su olakšati testiranje i ispravljanje grešaka, te napraviti skalabilan program koji jednako lako radi s velikim i malim brojem slika. Kao arhitektura odabrana je objektno-orijentirano programiranje. Za izbor programskog jezika razmatrane su tri mogućnosti: C, C++ i Java. Iako nije moguće mjeriti brzinu ili memorijsko zauzeće samog programskog jezika, već samo brzinu i memorijske potrebe neke od njegovih implementacija, u (Fulgham, 2008) se može pronaći niz testova koji sugeriraju sljedeće: C je neznatno brži od C++-a (do faktora 1.5) te zahtijeva nešto manje memorije. C++ je brži od Jave (za faktor 1.0-2.0), ali *Java zahtijeva i do red veličine više memorije.* Kako je memorija značajan čimbenik u treniranju aplikacije, budući da se sve trenirane slike zajedno sa svojim derivacijama nalaze u memoriji, odabran je programski jezik C++. Da bi se ostvarila skalabilnost i prilagodljivost programa koriste se generičke C++ klase: STL (eng. Standard Template Library). Od generičkih klasa STL-a izdvaja se klasa *vector.* Klasa *vector* djeluje kao omotač oko polja i omogućuje jednostavno alociranje prostora tokom izvođenja, pa se njegova veličina može dinamički prilagođavati broju elemenata. *Vector* se često koristi, a kao primjer generalizacije koja je s njim moguća izdvaja se primjer parametriranja smjerova derivacije, o čemu je već bilo riječi. Standardna implementacija koristi 4 smjera derivacija opisanih na slici 10. Za dodavanje novog smjera derivacije u program dovoljno je dodati tek jedan redak u odgovarajući *vector* koji opisuje novi smjer derivacije.

Za rad sa slikama koristi se biblioteka OpenCV. Njen kod napisan je u C-u, pa se lako može uključiti u C++ projekt. OpenCV omogućuje jednostavno učitavanje slika, pretvaranje u crno-bijelu sliku, te očitavanje piksela. Druge naprednije mogućnosti se ne koriste, već se implementiraju samostalno.

Nakon treniranja klasifikator se zapisuje u XML dokument. Ovaj XML dokument dovoljno je pri svakom idućem testiranju učitati, bez potrebe za novim treniranjem klasifikatora. Osim što je na taj način lako učitati klasifikator za potrebe testiranja, XML dokument se pokazao korisnim u pronalaženju grešaka pri kodiranju.

```
<classifier>
. . .
<shapelet begx="80" begy="60" endx="85" endy="65" size="10" area="5" threshold="-
0.305287" power="1.46782" error="0.0504199" >

        <pointer x="83" y="60" direction="2" threshold="6" positive="0" error="0.400639"
                power="0.201402" />
        <pointer x="82" y="60" direction="1" threshold="0" positive="1" error="0.40627"
                power="0.189703" />
        <pointer x="82" y="61" direction="3" threshold="10" positive="1" error="0.406575"
                power="0.189071" />
. . .
</shapelet>
. . .
</classifier>
```

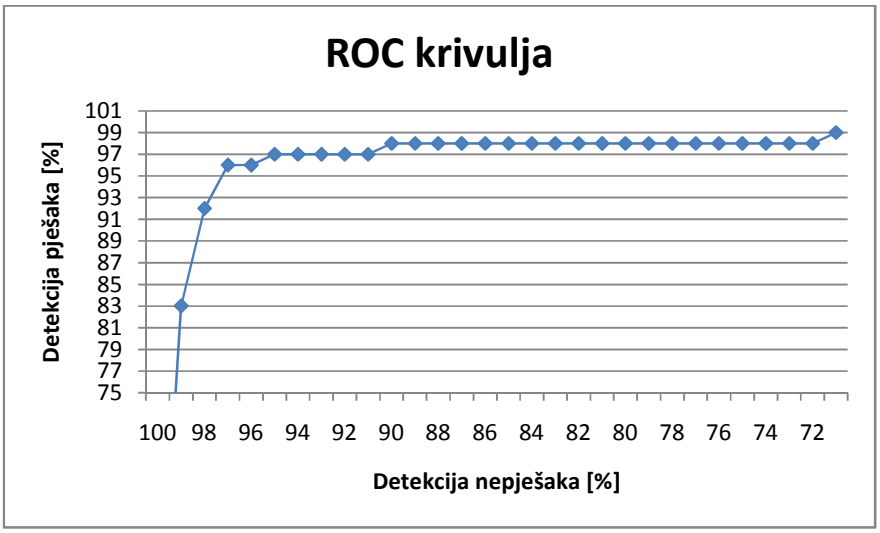
Slika 21. XML sadržaj klasifikatora

3 Testiranje i rezultati

Kao skup podataka za treniranje klasifikatora odabrana je baza podataka pješaka MITpedestrian. Baza sadrži 924 slike pješaka u ppm (eng. *portable pixmap format*) formatu. Kao negativni primjeri poslužile su slike baze INRIA.

Klasifikator sliku ocjenjuje tako da "pozove" na glasanje svaki obris i njegov glas uzme s određenom težinom. Pri tom obris ne ocjenjuje samo binarno, kao pješak ili nepješak, već brojčano: što je veći broj ocjene to je obris sigurniji da je na slici pješak. Taj broj množi se s težinom (snagom) koju obris ima. Konačna ocjena klasifikatora je broj dobiven zbrajanjem ocjena obrisa. Parametar koji se može mijenjati je prag detekcije: ako se prag postavi visoko tada je potrebna visoka ocjena za sliku da bi ona bila pješak. Suprotno, ako je prag nizak tada će većina pješaka sakupiti dovoljnu ocjenu klasifikatora, ali će dovoljnu ocjenu postići i određen broj nepješaka.

Postignuto je prepoznavanje s greškom reda veličine 1% za pješake i nepješake. Nad testiranim podacima klasifikacije prikazane na slici, pješaci se prepoznaju u 97% posto slučajeva, kada se nepješaci prepoznaju u 96% slučajeva. ROC (eng. Receiver Operating Characteristic) krivulja pokazuje odnos detektiranih pješaka i nepješaka u postotcima. Krivulja je dobivena učenjem nad 200 slika pješaka i 600 slika nepješaka, veličine 128x64. Testni podaci su 200 pješaka i 800 nepješaka. Rezultati su usporedivi s rezultatima rada (Sabzmeydani, 2007), koji koristi jednake značajke.



Slika 22. Konačna ROC krivulja treniranog klasifikatora, ovisno o pragu klasifikacije

Važno je napraviti razliku između prepoznavanja pješaka u *prozoru* i slici. Razliku ilustriraju slike 23 i 24. Prepoznati pješaka u prozoru znači odgovoriti na pitanje: za podsliku (prozor) veličine 128x64, da li je u njoj pješak? Prepoznavanjem pješaka na slici ovdje podrazumijevamo njihovu detekciju kao što je označeno na slici 24. Takva detekcija koristi detekciju pješaka u prozorima, kako je objašnjeno kasnije.

Slika 24. Detekcija pješaka na slici

Slika 23. Prozor

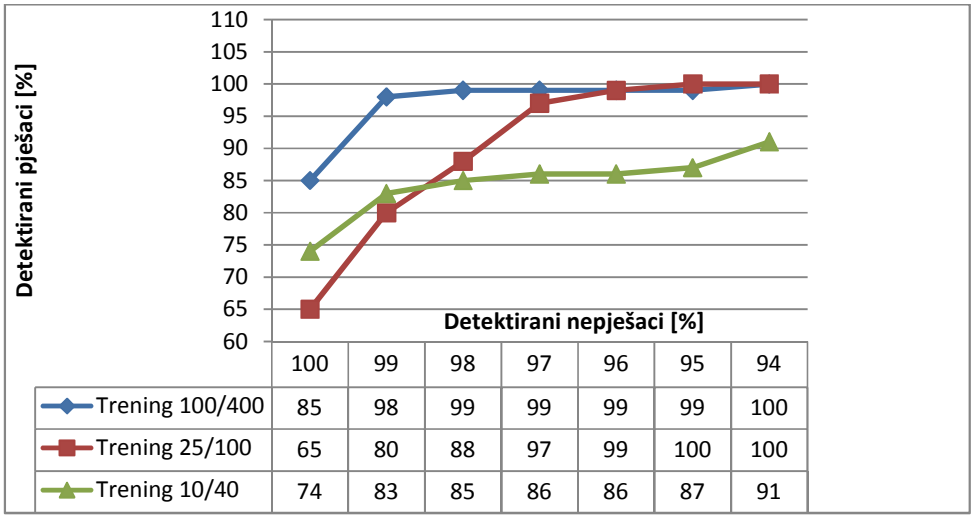
Na rezultate klasificiranja utječu parametri algoritma. Njihovim mijenjanjem podešavaju se rezultati unutar istog reda veličine. Pri podešavanju algoritma koristi se uvijek isti skup testnih slika koji se sastoji od 100 slika pješaka i 200 slika nepješaka. Ovaj broj slika nije dovoljan za konačno i precizno opisivanje rezultata klasifikatora, već se koristi kao pokazatelj pri mijenjanju parametara algoritma. Razumno je pretpostaviti da će klasifikator koji bolje klasificira ovaj testni skup bolje klasificirati i veći skup.

Izdvojeno je 7 najvažnijih parametara kojima se algoritam fino podešava. Potpuna pretraga za njihovom najboljom kombinacijom računalno je neizvediva. Naprednji pristupi vodili bi genetskom algoritmu ili sličnoj metodi optimizacije. Budući da se podešavanjem parametara ne postiže poboljšanje točnosti za red veličine, ovakva potraga ocijenjena je neisplativom. Parametrizacija se vrši pohlepnim algoritmom (eng. greedy algorithm): prvo se odredi najbolja vrijednost za jedan parametar, dok se drugi ne

mijenjaju. Uzima se najbolja vrijednost od testiranih i prelazi na idući parametar, te se postupak ponavlja.

3.1 Parametri

Najvažniji parametar za brzinu i preciznost treniranja je *broj pozitivnih (pješaka) i negativnih slika* nad kojima se klasifikator trenira. Rezultati su prikazani na slici 25. Klasifikator već na uzorku od 10 pozitivnih i 40 negativnih slika postiže razinu prepoznavanja nepješaka od 94%, uz 91% detektiranih pješaka.



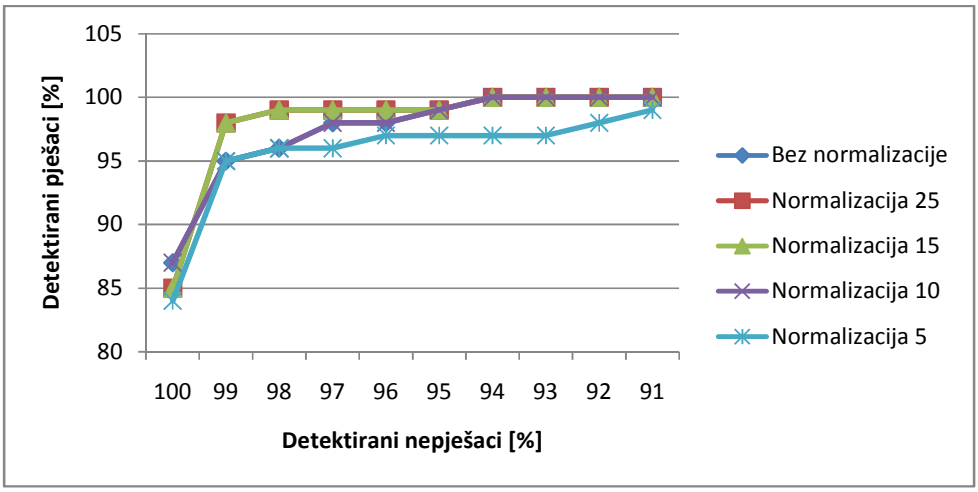
	100	99	98	97	96	95	94
Trening 100/400	85	98	99	99	99	99	100
Trening 25/100	65	80	88	97	99	100	100
Trening 10/40	74	83	85	86	86	87	91

Slika 25. ROC krivulje u ovisnosti o broju slika za treniranje klasifikatora

Prag derivacije je prag koji piksel mora imati da bi ušao u normaliziranu derivaciju, kao na slici 13. Pikseli nižeg intenziteta se zanemaruju jer se inače procesom normalizacije šum povećava. Ispitane su vrijednosti praga derivacije od 10, 20, 30 i 40. Najbolji rezultati postižu se za prag derivacije 30.

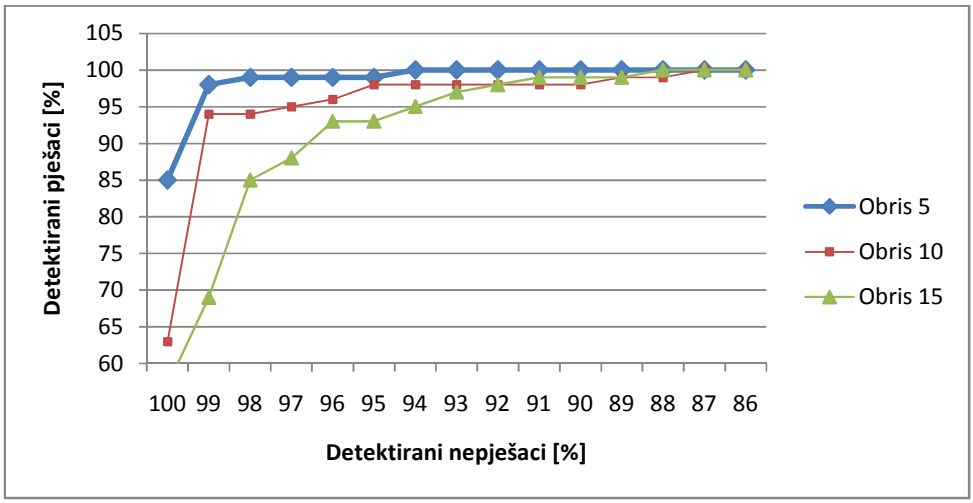
Pri stvaranju derivacija slika se lokalno normalizira po formuli (5). Normalizacija se odvija tako da se svi pikseli prozora pomnože s nekim faktorom, tako da piksel najvećeg intenziteta dobije intenzitet 255. *Prozor normalizacije* zadaje veličinu prozora. Manjom veličinom prozora postiže se lokalizirana normalizacija i uvodi veća razina šuma. Slika 26 uspoređuje rezultate normalizacije za različite veličine prozora normalizacije. Odabrana

veličina prozora je 25x25, koja ima jednake rezultate kao i veličina 15x15. Normalizacija smanjuje grešku sa 5% na 2% uz grešku na klasifikaciji nepješaka od 1%.



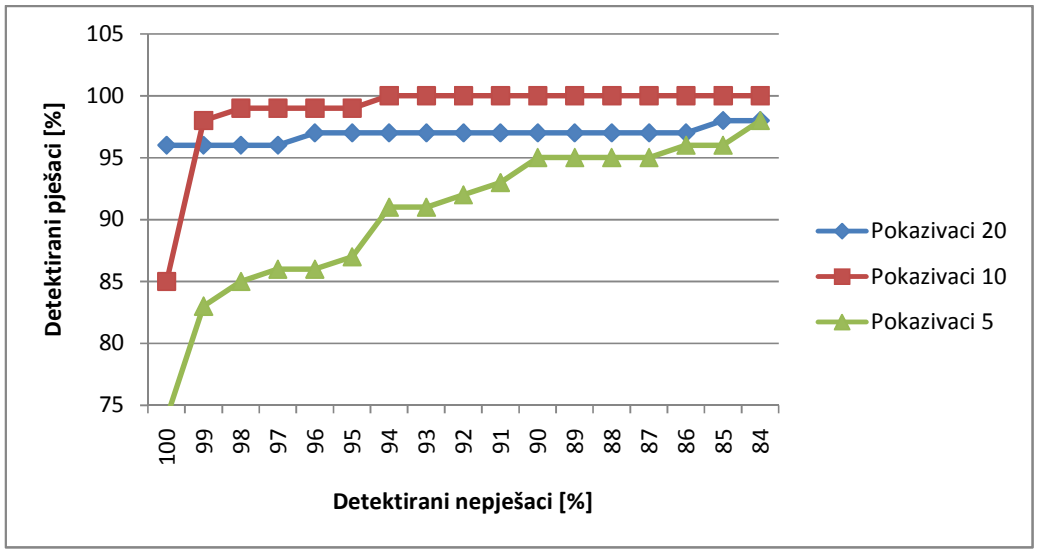
Slika 26. ROC krivulja za različite vrste normalizacije

Prozor obrisa dijeli sliku na podprozore iz kojih se stvaraju obrisi. Primjer prozora obrisa prikazan je slikom 14. Ovaj parametar određuje veličinu prozora za svaki obris. Među testiranim veličinama od 5,10 i 15, najbolja se pokazala veličina 5. Manja veličina prozora prisiljava klasifikator da uključi više obrisa, a time i više pokazivača iz različitih dijelova slike. Slika 27 prikazuje rezultate klasifikatora s različitim veličinama prozora obrisa.



Slika 27. ROC krivulje za različite veličine obrisa

Slika 28 prikazuje detekciju u ovisnosti o broju pokazivača uključenih u obris. Intuitivno, više uključenih pokazivača znači i više informacija o slici, pa će detekcija za više pokazivača biti bolja. Ipak, pokazuje se da je 10 uključenih pokazivača bolje od 20. Područje koje pokriva obris pri tome je konstantnih 5x5.



Slika 28. ROC krivulje za različiti broj pokazivača po obrisu

Pri usrednjavanju svaki piksel postaje srednja vrijednost svojih susjeda. Veličinu susjedstva definira ovaj parametar. Od ispitanih vrijednosti, najboljom se pokazala 3, pa su susjedi svakog piksela 8 okolnih piksela.

Može se dogoditi da neki obris potpuno točno razdvaja sve trenirane slike pješaka i nepješaka. Tada je njegova pogreška 0, što u skladu s formulom (9) daje beskonačnu snagu obrisu. Zbog toga se minimalna greška u tom slučaju postavlja na vrijednost *AdaBoost epsilona*. Vrijednost 1^{-20} odabrana je među razmatranim 1^{-10}, 1^{-15}, 1^{-20}, 1^{-30}.

4 Detekcija pješaka u slikama

Nakon što je objašnjeno podešavanje klasifikatora, u narednom dijelu objašnjava se korištenje treniranog klasifikatora za pronalaženje pješaka na slikama, kao u primjeru slike 23. Treba imati na umu da klasifikator za podsliku veličine 128x64 odgovara na pitanje: da li je na isječku pješak? Dio slike za koji se pitanje postavlja nazivamo prozorom. Pješak će biti detektiran samo ako je u sredini i dominantan u prozoru. Pri traženju pješaka na slici, lokacija prozora pomiče se po slici i za svaki prozor postavlja upit pred klasifikator. Korak kojim se pomiče prozor je 5 piksela. Primjerom, za neku poziciju (x,y) klasifikator odgovara na pitanje je li u pravokutnoj podslici (x,y)-(x+128,y+64) pješak. Nakon toga se prozor pomiče za 5 piksela desno i upit ponavlja za podsliku (x+5,y+5)-(x+5+128,y+5+64).

Broj prozora koje treba ispitati jednak je $H*W/S^2$, gdje su H i W dimenzije slike, a S veličina koraka. Za sliku veličine 320x64 i S=5 treba ispitati 3072 prozora. Za sliku veličine 640x480 broj prozora penje se na 12 288.

Pomicanjem prozora po slici nisu riješeni svi izazovi detekcije. Već je rečeno da pješak mora zauzimati dominantan dio slike, kao u slikama nad kojima je klasifikator treniran. Postavlja se pitanje, što ako je pješak manjih ili većih dimenzija od treniranih 128x64? Klasifikator nije moguće prilagoditi, budući da je on treniran na velikom broju slika, već treba prilagoditi veličinu pješaka. Zato se nakon prolaza prozora kroz sliku ona treba skalirati za faktor 1.2 i postupak ponoviti. Ako je slika bila dimenzija 640x480, skaliranjem za faktor 1.2 nova će slika biti veličine 533x400.

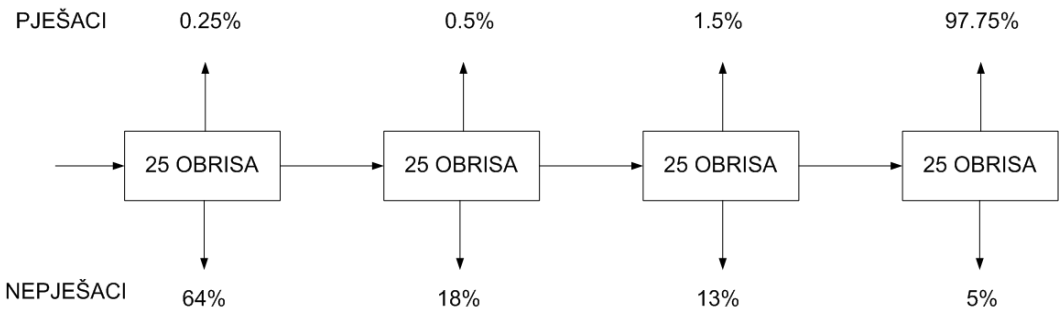
4.1 Ubrzanje klasificiranja prozora

Da bi se klasifikacija provela, potrebno je izraditi slike derivacija. Složenost ove operacije je $O(H*W*S^2*D)$, gdje su H i W dimenzije prozora, ovdje 128 i 64. S^2 je prozor kojim se izvodi usrednjavanje, u primjeni S=3. D je broj smjerova po kojima se izvode derivacije, D=4. Slike derivacije ne rade se za svaki podprozor, već za cijelu sliku na početku pretrage. Skaliranjem se i derivacije skaliraju za isti koeficijent.

Samo klasificiranje prozora teče tako da sliku ocjenjuju obrisi i računa se suma njihovih težinskih glasova. Ukupan broj obrisa je oko 330, a svaki sadrži 10 pokazivača, kako je objašnjeno u dijelu o parametrima. Ubrzanjem klasificiranja nastojao se ukupan broj ispitanih pokazivača po prozoru smanjiti.

Ubrzanje se vrši kroz kaskadno ocjenjivanje. Cilj je kaskade podijeliti ocjenu prozora u više dijelova, te nakon svakog dijela eliminirati postotak najlošijih kandidata za pješaka. Kaskada je usmjerena na eliminiranje nepješaka, budući da će u detekciji pješaka na slici puno veći postotak prozora činiti nepješaci.

Obrisi se svrstavaju po snazi, tj. po težini svojih glasova, te se najprije vrši ispitivanje 25 najboljih obrisa. Ako je ocjena 25 najboljih obrisa manja od granice θ_1, prozor se odbacuje kao nepješak. Ako je ocjena veća od $\theta_1$sustav nastavlja klasifikaciju u drugi dio kaskade. Ispituje se daljnjih 25 obrisa s novom granicom θ_2i ponovno odbacuju najlošiji kandidati. Kaskada ukupno ima 4 dijela i 100 obrisa. Slika 29 prikazuje postotak odbačenih prozora za svaki stupanj kaskade. Testni primjer je 200 slika pješaka i 800 slika nepješaka. Prvi test u kaskadi odbacuje 64% nepješaka uz 0.25% pješaka. Idealan test odbacivao bi što više nepješaka i nijednog pješaka.

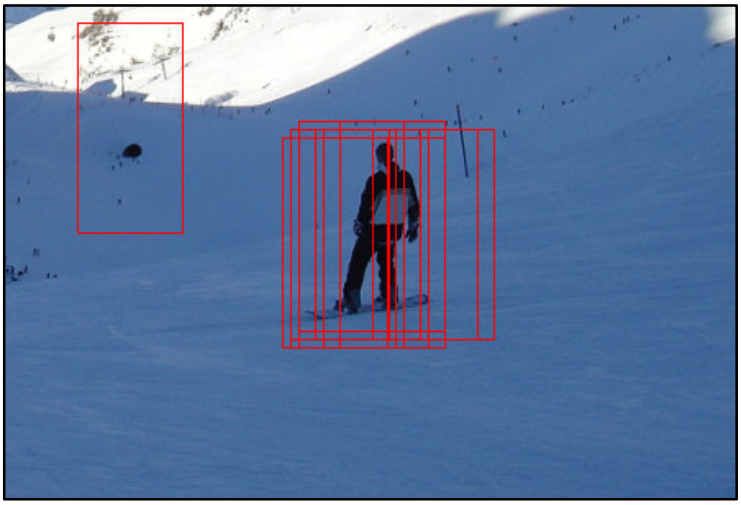


Slika 29. Prikaz kaskade klasifikatora

Već je rečeno da postoji ukupno oko 330 obrisa, svaki sa 10 pokazivača, što čini 3330 pokazivača ukupno. Kaskadno testiranje pokazuje da se za prozor nepješaka testira prosječno 40 obrisa s po 10 pokazivača, što je ubrzanje od oko 8 puta.

4.2 Detekcija pješaka klasifikacijom prozora

Pomicanjem prozora po slici, kao što je opisano u uvodu 4. poglavlja, isti pješak detektira se više puta. Pri tome se i neki prozori koji pripadaju pozadini detektiraju kao pješak, ali za takve detekcije najčešće vrijedi da ne dolaze grupirane, već su više nasumično raspoređene. Slika 30 prikazuje višestruke detekcije karakteristične za osobu i izdvojenu pogrešnu detekciju.



Slika 30. Višestruke detekcije osobe na slici

Da bi se višestruke bliske detekcije spojile u jednu detekciju osobe, a izvojene detekcije odbacile koristi se sljedeći algoritam:

Napravi pomoćnu sliku tako da svakoj poziciji prozora pripada jedna točka
Idi po svim lokacijama prozora, korakom od 5 piksela:

- Ako je u određenom prozoru detektiran pješak, označi tu točku crnom bojom u pomoćnoj slici, u suprotnom je točka bijela.

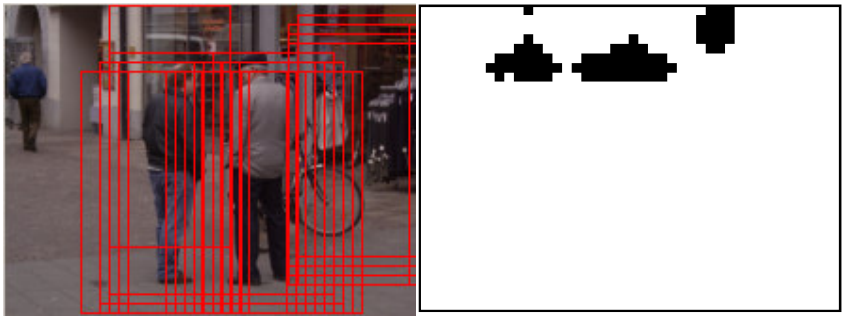
Nakon što su pregledani svi prozori:

- Proširi sve crne komponente za jedan piksel
- Za svaku komponentu površine veće od 20, izračunaj njeno središte
- Središte svake komponente veće od 20 označi kao pješaka

Detalji pomoćne slike pokazuju se primjerom. Ako je slika u kojoj se detektiraju pješaci velika 320x240, tada postoji 320x240/25 prozora koji će se klasificirati kao pješak ili nepješak. Redom, lokacije tih podprozora su: (0,0), (0,5), … (0,235),(1,0),…,(315,235). Svakom od tih lokacija pridjeljuje se točka u pomoćnoj slici, tako da lokaciji (0,0) pripada točka (0,0), prozoru na (0,5) točka (0,1) i tako redom. Formula preslikavanja je:

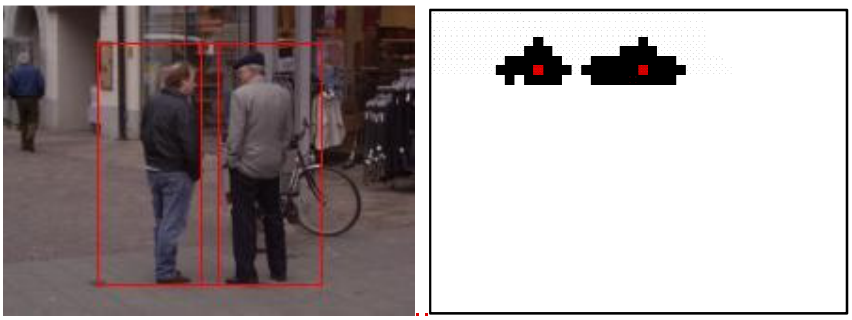
$$(x, y) \rightarrow (\frac{x}{5}, \frac{y}{5}) \qquad (12)$$

Pomoćna slika koja se koristi u algoritmu prikazana je na slici 31, zajedno sa slikom u kojoj su prikazane višestruke detekcije. Može se primijetiti da se osim pješaka na slici lijevo pojavljuju i dvije skupine lažnih detekcija: jedna pri sredini slike iznad lijevog pješaka i druga nešto iznad i s desne strane drugog pješaka. Lažne detekcije u pomoćnoj slici desno predstavljene su svojim komponentama.



Slika 31. Višestruke detekcije opisane u pomoćnoj slici

Komponente koje u se pomoćnoj slici stvaraju za lažne detekcije manje su površine od komponenti za detekcije pješaka. Prag za eliminaciju je određen kao površina od 20 piksela. Konačan rezultat nakon eliminacije vidljiv je na slici 32. Crvenom točkom na pomoćnoj slici je označena sredina svake preostale komponente, nakon eliminacije.



Slika 32. Konačna detekcija pješaka

Detekcije na odabranim slikama baze INRIA priložene su u dodatku.

Završne primjedbe za daljnji razvoj sustava detekcije su:

- Sustav ima veći broj krivo pozitivnih detekcija u slučajevima kada je pozadina raznolika. Moguće je da bi uključivanjem većeg broja takvih slika u treniranje taj broj smanjio.

- Postotak detekcije povećao bi se poboljšanjem parametara. Aplikacija je pisana tako da su svi parametri izdvojeni u posebnu klasu i jednostavno se mijenjaju. Idući korak mogao bi biti izrada testnog programa koji bi te parametre mijenjao, pokretao program za klasifikaciju i automatski računao postotak točnih detekcija.

- Detekciju bi poboljšao algoritam koji će konačnu odluku o osobama na slici donositi na temelju detektiranih prozora u različitim veličinama slike, tj. algoritam koji bi spajao rezultate prije i poslije skaliranja slike. Za ostvarivanje takvog algoritma predlaže se sljedeći pristup: za svaku veličinu slike napraviti ranije opisan postupak temeljen na pomoćnoj slici. Nakon toga udruživanje provesti tako da se prednost daje detekcijama koje su veće površinom, tako da sve detekcije na slici manje veličine koje padaju unutar veće detekcije budu zanemarene.

5 Zaključak

Prepoznavanje osoba u slikama važan je problem s mnogim primjenama, među kojima se ističe prepoznavanje pješaka kao osobito izazovno i traženo područje. Do sada su postignuti rezultati detekcije od 90% prepoznatih pješaka u stvarnom vremenu. Ovaj rad na tragu je takvih rezultata.

Opisana su tri dijela razvoja sustava za detekciju pješaka: najvažniji među njima je izbor i priprema značajki. O izboru značajki uvelike ovise brzina i rezultati prepoznavanja. Značajke obrađene u ovom radu opisuju oblik siluete osobe, općenite su i stoga primjenjive na drugim područjima, kao što je prepoznavanje lica. Priprema značajki vremenski je najzahtjevniji dio učenja, pa se posebna pozornost posvećuje ubrzanju njihove obrade.

Drugi dio sustava za detekciju pješaka je implementacija algoritma strojnog učenja. AdaBoost se pokazao kao učinkovit, jednostavan i memorijski skroman kandidat. Budući da je moguće da se ovaj algoritam usmjeri na mali broj značajki koje dobro opisuju trenirani skup slika, ali ne i testni skup, značajke se izrađuju u dvije razine. Značajke niže razine opisuju pojedine piksele slike, a značajke više razine organiziraju bliske niže značajke u cjeline. Kako se u klasifikator trenira nad svim značajkama više razine, algoritam učenja prisiljava se na upotrebu nižih značajki sa svih dijelova slike.

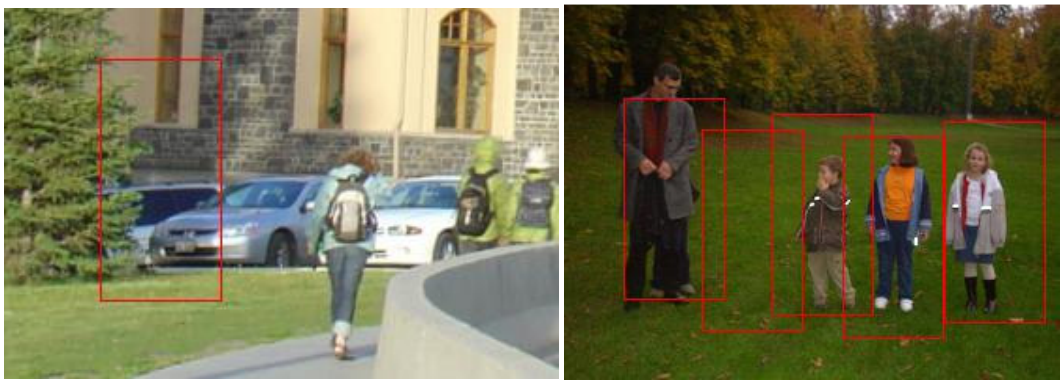
Posljednji dio izrade sustava za detekciju je izrada kaskade i uporaba klasifikatora za detekciju. Ovaj dio zahtjeva opsežna testiranja za podešavanje parametara, pa do izražaja dolazi sam dizajn aplikacije i jednostavnost uporabe. U sklopu ovog dijela implementiran je algoritam koji rezultate postignute u detekciji prozora prenosi na detekciju pješaka u slikama.

Iako su sva tri navedena koraka potrebna za stvaranje sustava implementirana, težište rada je stavljano na odabir značajki i strojno učenje. Postignuti rezultati mogu se usporediti s poznatim radovima ovog područja: postotak detekcije *prozora* penje se do 98% za pješake i nepješake. Mijenjanjem parametara rezultati se poboljšavaju unutar istog reda veličine.

Kako je odabir *vrste* značajki ključan za uspjeh prepoznavanja, predlaže se novi pristup dizajnu značajki: osmisliti algoritam koji će samostalno odabrati najbolje *vrste* značajki i te značajke međusobno kombinirati. Takav algoritam trebao bi kombinirati najbolja svojstva Haarovih značajki i značajki temeljenih na obliku, ili možda tek otkriti novu vrstu uspješnih značajki. Izazov stvaranja takvog algoritma bio bi dvostruk: kao prvo, potrebno je na općenit način opisati značajku, tako da u taj opis spadaju postojeće uspješne značajke. Kao drugo, potrebno je prostor mogućnosti na inteligentan način pretražiti, budući da će on svakako biti prevelik za jednostavnu pretragu.

6 Dodatak

Slike uspješnih i neuspješnih detekcija.



7 Literatura

[1] Sabzmeydani, P., Mori, G., "Detecting Pedestrians by Learning Shapelet Features", IEEE Conference on Computer Vision and Pattern Recognition, 2007., stranice: 1-8, 17-22

[2] Viola, P., Jones, M., "Robust Real-time Object Detection", Eighth IEEE International Conference on Computer Vision, 2001., vol.: 2, stranice: 747-757

[3] Papageorgiou, C., Poggio, T., "Trainable pedestrian detection", International Conference on Image Processing, Kobe, Japan, 1999., vol.: 4, stranice: 35-39

[4] Freund, Y., Schapire R., "A Short Introduction to Boosting", Journal of Japanese Society for Artificial Inteligence, 1999., stranice: 771-780

[5] Viola, P., Jones., M., Snow, D., "Detecting pedestrians Using Patterns of Motion and Appearance", International Journal of Computer Vision, 2005., izdanje 63, stranice 153-161

[6] Polikar R., "Esemble based systems in decision making," IEEE Circuits and Systems Mag., 2006., vol. 6, no. 3, stranice: 21-45

[7] "Gentoo : Intel® Pentium® 4 Computer Language Benchmarks Game", Datum nastanka: 02. siječnja 2008., "*The Computer Language Benchmarks Game",* URL: http://shootout.alioth.debian.org/gp4/, Datum pristupa: 10. svibnja 2008.

[8] Bu, F., Chan., C., "Pedestrian Detection in Transit Bus Application: Sensing Technologies and Safety Solutions", Intelligent Vehicles Symposium, 2005., Vol, 6-8 2005, stranice: 100 – 105

[9] Monteiro, G., Peixoto, P., Nunes, U., " Vision-based Pedestrian Detection using Haar-Like features", Robotica 2006 - Scientific meeting of the 6th Robotics Portuguese Festival, Portugal, 2006

8 Sažetak i ključne riječi

8.1 Detekcija osoba u slikama

Detekcija osoba u slikama zahtjevno je područje koje se sve više istražuje u akademske i komercijalne svrhe. Među mnogim uporabama kao jedna od težih primjena ističe se prepoznavanje pješaka. Ovaj rad opisuje razvoj sustava koji detektira pješake u slikama opisom značajki ljudske siluete. Pristup značajkama razlikuje se od većine radova ovog područja: dvije vrste značajki izrađuju se u dva sloja. Pokazuje se da ovakav pristup poboljšava rezultate detekcije. Značajke se obrađuju AdaBoost algoritmom i implementiraju u konačni klasifikator koji ima oblik kaskade. Konačna odluka o detekciji pješaka donosi se na temelju učestalosti i gustoće detektiranih pješaka po podslikama. Priloženi su rezultati testiranja sustava. Na kraju se temeljem ovog rada predlaže nov sustav odabira značajki.

Ključne riječi: detekcija osoba, detekcija pješaka, značajke oblika, dva sloja značajki, AdaBoost, kaskada, odabir značajki

8.2 People detection in images

People detection in images is rapidly becoming more interesting to both academic and commercial researchers. Among many uses, pedestrian detection seems to be one of the most challenging and rewarding. This paper aims to develop a system that will detect pedestrians using shape features. Features are not, as usually, of one kind. Instead, it is shown that system performance is improved by introducing a two layered approach to features. Features are then trained with AdaBoost, and implemented in a final classifier that has cascade form. Final decisions regarding pedestrian detection are made based on frequency and density of positively classified sub images. System is tested and the results are presented. A new approach to feature selection is suggested based on this work.

Keywords: people detection, pedestrian detection, shape features, two layered features, AdaBoost, cascade, feature selection

9 Prilozi

Radu se prilažu sljedeći sadržaji:

- Izvorni kôd programa koji se sastoji od glavne .cpp datoteke *Hello.cpp* s pratećim .h datotekama: *AdaBoost.h*, *Classifier.h*, *ErrorReport.h*, *Image.h*, *Pointer.h*, *Shapelet.h*, *Params.h*
- Izvorni kôd pomoćnog programa *thresholder.cpp* koji nakon treniranja kaskade stvara ROC krivulju mijenjajući prag klasifikacije
- XML datoteka *Classifier99.xml* treniranog klasifikatora
- Baza slika MITpedestiran koja se koristi za treniranje kaskade
- Baza slika INRIA čiji se prvi dio koristi za treniranje kaskade slikama nepješaka, a drugi za detekciju pješaka na slikama