SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 745

PROGRAMSKA OKOLINA ZA EVALUACIJU POSTUPAKA DETEKCIJE OBJEKATA U RAČUNALNOM VIDU

Stipe Grbić

Zagreb, lipanj 2009.

Sadržaj

1. Uvod

Cilj ovog rada je napraviti programsku okolinu za evaluaciju postojećih algoritama detekcije objekata u računalnom vidu. Budući da je te algoritme potrebno testirati na velikom broju slika, korisno je napraviti program koji će to obaviti automatizirano i koji radi za više algoritama. Takav program treba omogućiti odabir algoritma koji se želi testirati, jednostavnu izmjenu parametara ispitivanih algoritama i odabir vrste objekata s kojima se želi testirati algoritam. Glavna prednost automatizirane evaluacije je mogućnost objektivne karakterizacije performansi ispitivanog postupka. U kontekstu razvoja pouzdanih sustava za industrijsku primjenu, ta prednost opravdava napor uložen u ručno označavanje objekata u slici.

Programu za evaluaciju potrebno je zadati skup slika na kojima se želi evaluirati algoritam i datoteku u kojoj je za svaku sliku opisana pozicija, veličina i vrsta objekta koji se na njoj nalazi. Program za svaku sliku iz zadanog skupa pokrene implementaciju zadanog algoritma detekcije, pročita rezultat obrade i usporedi da li je taj rezultat u skladu sa zapisom iz datoteke s točnim pozicijama objekata. Usporedba dobivenog i točnog pravokutnika se radi tako da se odredi udio preklapanja tih pravokutnika. Ako je taj udio veći ili jednak udjelu zadanom u parametrima evaluacije, objekt je ispravno detektiran. Mogući rezultati evaluacije su da je objekt ispravno detektiran ili da je detekcija netočna. Ako nijedna detekcija za određeni objekt nije točna, onda objekt nije detektiran.

Automatska evaluacija rada algoritma, osim za ocjenu performansi samog algoritma, može poslužiti i za ispitivanje ispravnosti prometne signalizacije. Moguće je zadati skup slika uz određenu prometnicu i datoteku s opisom gdje bi se koji prometni znak trebao nalaziti. Program će prijaviti koliko znakova je točno označeno tj. vidljivo, a koliko ih je nedetektirano pa se može provjeriti što je s tim znakovima.

2. Postupci pronalaženja prometnih znakova

Razvijeni program za automatiziranu evaluaciju postupaka detekcije objekata biti će primijenjen na dva prethodno razvijena postupka:

- Pronalaženje prometnih znakova Houghovom transformacijom za kružnice – Završni rad 2009, Maja Šverko [1]
- Prepoznavanje prometnih znakova na temelju boje – Projekt 2007/2008, Tomislav Babić, Tomislav Lukinić, Damir Kovač, Kristina Popović, Dominik Rajković, Maja Šverko [2]

Ovi postupci razvijeni su u programskom okruženju (ljusci) cvsh (*computer vision shell*). Ljuska je razvijena na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave (ZEMRIS) fakulteta elektrotehnike i računarstva (FER) u Zagrebu.

Oba postupka bilo je potrebno spojiti u jedno programsko rješenje tako da programska okolina može mijenjati postupak koji će koristiti samo preko parametra za konfiguraciju. Budući da oba postupka koriste istoimene funkcije s različitom implementacijom trebalo je promijeniti njihova imena tako da svaki postupak koristi svoju funkciju. Zbog toga je sljedećim funkcijama u postupku prepoznavanje prometnih znakova na temelju boje dodan sufiks "_signs":

```
int *narastanje_signs
void nadjiBoju_signs
void flood_fill_signs
```

2.1. Prepoznavanje prometnih znakova na temelju boje

Postupak prepoznavanje prometnih znakova na temelju boje stvara binarnu sliku u ovisnosti o primljenim parametrima koji određuju boju. Binarna slika ima dvije moguće vrijednosti intenziteta za svaki piksel – 0 i 255 koje predstavljaju crnu i bijelu boju. Ako se HSI vrijednost piksela izvorne slike nalazi unutar boje koja je zadana parametrima, tada se intenzitet tog piksela postavlja na 255, a inače na 0. Tako se dobije crna slika s bijelim područjima koja bi mogla predstavljati prometni

znak. Postupak zatim za ta područja određuje mogu li ona biti prometni znak tako da za svaki bijeli piksel provjerava jesu li okolni pikseli također bijeli. Ako je pronađena dovoljno velika skupina bijelih piksela, tada ona predstavlja prometni znak. Za takvu skupinu se odrede krajnje točke (lijeva, desna, gornja i donja), a iz njih se onda izračunaju koordinate točaka pravokutnika koji opisuje taj znak.

Implementacija ovog postupka nije omogućavala konfiguraciju parametara preko komandne linije pa su učinjene sljedeće promjene:

1. U razredu `alg_signsHSI` dodani su parametri:

   ```
   double satLo_, double satHi_, double intLo_ i double intHi_
   ```

2. U funkciji `config` razreda `alg_signsHSI` odsječak koda:

   ```
   if (strlen(buf)>0){
         std::istringstream iss(buf);
         iss >>hueLo_ >>hueHi_;
         oss <<std::endl;
   }
   ```

   zamijenjen je sa sljedećim:

   ```
   if (strlen(buf)>0){
         std::istringstream iss(buf);
         iss >>hueLo_ >>hueHi_ >>satLo_ >>satHi_ >>intLo_ >>intHi_;
         oss <<std::endl;
   }
   ```

3. Naredba funkcije `dumpParams` razreda `alg_signsHSI` promijenjena je iz:

   ```
   os <<intro <<"hueLo, HueHi: " <<"(" <<hueLo_ <<", " <<hueHi_
   <<")\n" <<std::endl;
   ```

   u sljedeću:

   ```
   os <<intro <<"hueLo, HueHi, satLo, satHi, intLo, intHi : " <<"("
   <<hueLo_ <<", " <<hueHi_ <<", " <<satLo_ <<", "<< satHi_ <<", " <<
   intLo_ <<", "<< intHi_ <<")\n" <<std::endl;
   ```

4. Funkciji `nadji_boju_signs` dodani su parametri za zasićenje i intenzitet boje te je njen prototip sada:

   ```
   void nadjiBoju_signs (int width, int height, float* NorSlika, char*
   BinSlika, double hueLo1, double hueHi1, double satLo1, double
   satHi1, double intLo1, double intHi1);
   ```

   Umjesto hardkodiranih vrijednosti, funkcija sada koristi primljene argumente:

   ```
   double satLo = satLo1;
   ```

```
double satHi = satHi1;
double intLo = intLo1;
double intHi = intHi1;
```

5. U poziv funkcije `nadji_boju` iz postupka `alg_signs` dodani su parametri `satLo_`, `satHi_`, `intLo_` i `intHi_`.

Ovako modificirani postupak preko komandne linije može primiti šest parametara o kojima ovisi kakvi prometni znakovi će se detektirati:

1. hueLo – donja granica vrijednosti nijanse boje
2. hueHi – gornja granica vrijednosti nijanse boje
3. satLo – donja granica vrijednosti zasićenja boje
4. satHi – gornja granica vrijednosti zasićenja boje
5. intLo – donja granica vrijednosti intenziteta boje
6. intHi – gornja granica vrijednosti intenziteta boje

2.2. Pronalaženje prometnih znakova Houghovom transformacijom za kružnice

Ovaj postupak omogućuje pronalaženje prometnih znakova s crvenim rubovima. Algoritam radi tako da svaki crveni piksel glasuje za piksele koji su od njega udaljeni za zadanu vrijednost radijusa kružnice. Ako se na slici nalazi crvena kružnica, svi pikseli kružnice će glasati za njeno središte pa će postupak proglasiti taj piksel središtem prometnog znaka. Zatim se na osnovu zadanog radijusa izračunavaju koordinate točaka pravokutnika koji opisuje taj prometni znak.

Implementacija ovog postupka nije omogućavala konfiguraciju parametara preko komandne linije. To je riješeno tako da je u funkciji `config` razreda `alg_houghCir` odsječak koda:

```
if (strlen(bufR)>0){
      std::istringstream iss(bufR);
      iss >>circRad;
      ossR <<std::endl;
}
```

zamijenjen sa sljedećim:

```
if (strlen(bufR)>0){
```

```
        std::istringstream iss(bufR);
        iss >>circRad >>hueLo_ >>hueHi_ >>satLo_ >>intLo_;
        ossR <<std::endl;
}
```

Postupak pronalaženje prometnih znakova Houghovom transformacijom za kružnice preko komandne linije može konfigurirati 5 parametara:

1. circRad – radijus kružnice
2. hueLo – donja granica vrijednosti nijanse boje
3. hueHi – gornja granica vrijednosti nijanse boje
4. satLo – donja granica vrijednosti zasićenja boje
5. intLo – donja granica vrijednosti intenziteta boje

2.3. Prilagodba postupaka programskoj okolini za evaluaciju

Da bi evaluacija postupaka detekcije objekata bila moguća treba ih modificirati tako da programska okolina za evaluaciju može pročitati rezultate njihova rada. U spomenuta dva postupka bilo je potrebno napraviti izmjene tako da ispisuju koordinate pravokutnika koji opisuje označeni prometni znak. To je postignuto dodavanjem sljedeće naredbe u funkciji `addRectangle` razreda `win_ann_util`:

```
if (ispisStipeGrbic) printf("\n***Evaluation data: x=%d y=%d w=%d
h=%d\n", p10.x(), p10.y(), p01.x()-p00.x(), p11.y()-p00.y());
```

Naredba ispisuje gornju lijevu točku te širinu i visinu pravokutnika. Ovaj ispis se može uključiti ili isključiti postavljanjem varijable ispisStipeGrbic na true odnosno false.

Svi budući postupci detekcije koje se želi evaluirati ovim programom moraju koristiti spomenutu funkciju `addRectangle` na način da prva točka u pozivu funkcije bude gornja lijeva, a druga donja desna. Na taj način funkcija će ispisati koordinate točaka pravokutnika onim redoslijedom koji očekuje program za evaluaciju.

3. Implementacija programske okoline

Program za evaluaciju razvijen je u programskom jeziku Python. Program iz komandne linije prima ime tekstualne datoteke s parametrima koje će koristiti za evaluaciju. U toj datoteci se može zadati sedam parametara:

- adresa direktorija sa ispitnim slikama
- identifikator postupka koji se evaluira
- zadani udio preklapanja pravokutnika da bi se detekcija proglasila točnom
- konfiguracijski parametri za postupak detekcije
- regularni izrazi vrste prometnih znakova na kojima se želi evaluirati postupak
- ime tekstualne datoteke s opisom točnih pozicija prometnih znakova na slikama
- ime programa koji će program koristiti za evaluaciju

Kad se učitaju parametri, program otvara datoteku s točnim pozicijama prometnih znakova te iz nje pročita vrstu, poziciju i veličinu prometnog znaka za svaku sliku. Zatim slijedi provjera odgovara li vrsta prometnog znaka na slici jednoj od zadanih vrsta u parametrima. Ako odgovara, pokreće se postupak detekcije s odabranom slikom, a inače se ta slika preskače i učitava sljedeća. To radi sljedeći programski odsječak:

```
for confstr in konfiguracija:
      output = subprocess.Popen([evaluiraniProgram, '-sf='+adresaSlike,
      algoritam, '-c='+confstr], stdout=subprocess.PIPE).communicate()[0]
      evalDataList.append(re.compile('\*\*\*Evaluation
      data:.*').findall(output))
```

Pokretanje postupka detekcije radi se naredbom `subprocess.Popen` koja rezultate detekcije sprema u varijablu output. Iz cijelog ispisa postupka detekcije treba izvući samo redak u kojem se nalaze koordinate označenog pravokutnika. Koordinate se nalaze zapisane u retku koji počinje s "***Evaluation data:" te ih sljedeća naredba sprema u listu koordinata pravokutnika evalDataList. Pokretanje postupka detekcije i učitavanje rezultata se ponavlja nad istom slikom za svaku

zadanu konfiguraciju postupka te se koordinate označenih pravokutnika dodaju u evalDataList.

Za svaki označeni prometni znak iz datoteke s točnim pozicijama znakova radi se usporedba sa svim pravokutnicima koje je označio postupak detekcije. U slučaju da postupak nije označio nijedan prometni znak na slici, program postavi zastavicu nistaPronadeno te sve znakove koji se nalaze na obrađenoj slici proglasi nedetektiranima.

Koordinate pravokutnika učitavaju se funkcijom ucitajKoordinate(oznaka). Funkcija u znakovnom nizu "oznaka" traži i vraća vrijednosti za koordinate i dimenzije pravokutnika. U implementaciji postupaka detekcije koordinate slike definirane su tako da je točka (0,0) u donjem lijevom kutu te koordinate rastu prema gore i desno, a u tekstualnoj datoteci s točnim pozicijama znakova točka (0,0) je u gornjem lijevom kutu te Y koordinata raste prema dolje. Zbog toga, nakon što se pročitaju koordinate pravokutnika iz postupka detekcije program ih usklađuje tako da promijeni Y koordinatu ovisno o visini slike:

```
x1,y1,w1,h1 = ucitajKoordinate(evalDataList[i][j])
y1 = visina-y1
```

Nakon toga, određuju se koordinate pravokutnika koji predstavlja presjek pravokutnika koji je označio postupak detekcije i pravokutnika koji predstavlja točnu poziciju prometnog znaka. Preklapanje pravokutnika se određuje funkcijom preklapanje:

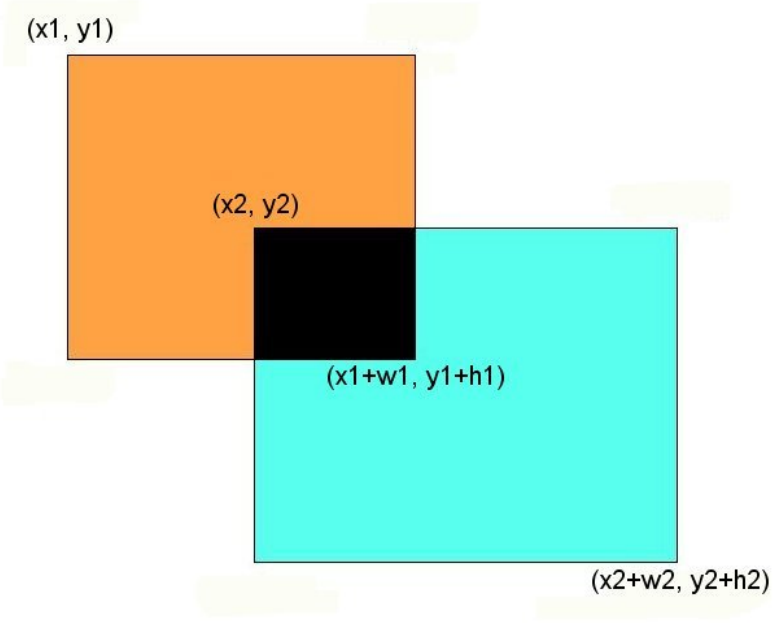
```
if preklapanje(x1,y1,w1,h1,x,y,w,h,zadanoPreklapanje):
      pronadeni[imeSlike + '_' + oznaka] += 1
else:
      brojNetocnih = brojNetocnih + 1)
```

Prvih osam argumenata su x i y koordinata te širina i visina prvog odnosno drugog pravokutnika. Zadnji argument je zadani postotak preklapanja pravokutnika za koji će se preklapanje proglasiti istinitim.

Na slici 1 prikazan je primjer preklapanja pravokutnika. Plavi pravokutnik predstavlja točno označeni prometni znak, a narančasti predstavlja rezultat postupka detekcije. Potrebno je odrediti koordinate crnog pravokutnika tj. presjeka, a to rade sljedeće četiri naredbe:

```
x1_preklapanje = max(x1, x2)
y1_preklapanje = max(y1, y2)
x3_preklapanje = min(x1+w1, x2+w2)
y3_preklapanje = min(y1+h1, y2+h2)
```

Prva naredba od dva pravokutnika traži maksimalnu vrijednost X koordinate za lijevu gornju točku. Maksimalnu zato što lijeve točke presjeka imaju X koordinatu desnog pravokutnika. Druga naredba traži maksimalnu vrijednost Y koordinate za istu točku. Opet maksimalnu zato što gornje točke presjeka imaju Y koordinatu donjeg pravokutnika. Slično rade i druge dvije naredbe, s izuzetkom da traže minimum vrijednosti koordinata donjih desnih točaka pravokutnika. Minimum zato što su koordinate donje desne točke presjeka uvijek gore lijevo od promatranih točaka.



Slika 1 – Primjer preklapanja pravokutnika

Slijedi izračun površine presjeka:

```
povrsinaPreklapanja = (x3_preklapanje - x1_preklapanje) * (y3_preklapanje - y1_preklapanje)
if ((x3_preklapanje < x1_preklapanje) or (y3_preklapanje < y1_preklapanje)):
        povrsinaPreklapanja = 0
```

Ako je X koordinata desne točke presjeka manja od X koordinate lijeve točke presjeka, to znači da se pravokutnici ne preklapaju. Isti je slučaj i kad je Y

koordinata donje točke presjeka manja od Y koordinate gornje točke presjeka. U oba slučaja površina presjeka se postavlja na nulu.

Slijedi usporedba omjera površine presjeka i pravokutnika točno označenog znaka. Ako je taj omjer veći ili jednak udjelu preklapanja zadanom u parametrima, znak se proglašava točno detektiranim, inače je to netočna detekcija i brojač netočnih detekcija se povećava za jedan.

Za svaki prometni znak postoji vrijednost u rječniku pronadeni koja označava koliko puta je taj znak bio označen. Ključevi rječnika su u obliku B43_0002.bmp_B43@(x=564,y=201,w=76,h=74) koji ga jedinstveno definira jer na jednoj slici može biti više različitih znakova i mogu biti na različitim pozicijama na slici.

Nakon što su obrađene sve slike, program prolazi kroz listu pronadeni i prebrojava koliko ima detektiranih i nedetektiranih znakova. Na kraju ispisuje rezultate rada: broj obrađenih znakova, broj detektiranih i nedetektiranih znakova te broj pogrešnih detekcija.

4. Instalacija i korištenje programske okoline

4.1. Instalacija programskog jezika Python

Python je programski jezik koji je besplatno dostupan za više platformi uključujući Microsoft Windows, Macintosh, Linux i Unix. Instalacija se može skinuti s adrese http://www.python.org/download/. Program za evaluaciju razvijen je u verziji jezika 2.6.2. te je potrebno skinuti instalaciju te verzije. Nakon instalacije moguće je pokrenuti python skripte iz komandnog prozora jednostavnim navođenjem imena skripte koja se želi pokrenuti.

Zbog različitog načina zapisivanja Y koordinate između ljuske u kojoj su razvijeni postupci za detekciju i datoteke u kojoj su zapisane točne pozicije znakova potrebno je svaku sliku otvoriti kao Image objekt, odrediti visinu i uskladiti Y koordinate. Za otvaranje slika u Pythonu treba instalirati biblioteku Python Image Library (PIL) koja je dostupna na adresi http://www.pythonware.com/products/pil/. U postupku instalacije biblioteke treba odabrati direktorij u koji je instaliran Python.

4.2. Datoteka s konfiguracijskim parametrima programa

Tekstualna datoteka s konfiguracijskim parametrima mora se nalaziti u istom direktoriju kao i program za evaluaciju. U datoteci se može zadati sedam parametara:

- Adresa direktorija sa slikama zadaje se preko parametra -s=,

  npr. -s=C:\okrugli
- Ime postupka koji se evaluira zadaje se parametrom -a=,
  npr. -a=signsHSI
- Zadani udio preklapanja pravokutnika zadaje se parametrom -perc=, npr. -perc=60%
- Konfiguracija postupka zadaje se parametrom -c=, npr. -c=30 5.95. 0.05 0.15 0.85. Moguće je zadati više različitih konfiguracija za postupak detekcije tako da se konfiguracije napišu odvojene znakom "&", npr. -c=30

5.95 0.05 0.15 0.85 & 20 5.2 0.1 0.2 0.6. Program će pokrenuti obradu slike sa svakom konfiguracijom posebno. Ako se ne zadaju konfiguracijski parametri, koriste se oni koji su podrazumijevani u određenom postupku.

- Regularni izrazi vrste znakova s kojima se evaluira postupak zadaje se parametrom -type=, npr. -type=B01. Na taj način iz direktorija sa slikama koriste se samo one na kojima se nalaze prometni znakovi zadanog tipa. Ako se želi koristiti više vrsta prometnih znakova, regularne izraze treba odvojiti znakom "|", npr. -type=B31|B43. Ako se ne navedu regularni izrazi u datoteci s parametrima, koristit će se sve slike iz direktorija.
- Ime tekstualne datoteke s točnim pozicijama prometnih znakova zadaje se preko parametra -file=, npr. -file=okrugli.txt. Ako se ovaj parametar ne navede, program će tražiti datoteku istoimenu direktoriju sa slikama koja ima ekstenziju .seq. U oba slučaja tekstualna datoteka se mora nalaziti u direktoriju sa slikama
- Ime programa koji se koristi za evaluaciju može se zadati parametrom -prog=, npr. -prog=znakovi.exe. Ako se ne navede ovaj parametar, program će koristiti podrazumijevano ime cvsh.exe.

Nužni parametri za rad programa su adresa direktorija sa slikama, ime postupka i udio preklapanja pravokutnika zadan u postotku. Parametri mogu biti poredani bilo kojim redoslijedom.

4.3. Datoteka s točnim pozicijama prometnih znakova

Za evaluaciju postupaka detekcije objekata potrebno je imati datoteku sa zapisima točnih pozicija tih objekata. Datoteku u kojoj su zapisane točne pozicije prometnih znakova moguće je napraviti programom Marker [3]. Taj program za svaki označeni prometni znak u datoteku zapisuje redak sljedećeg formata:

[oznaka_slike]:opis_znaka@(x=x_koordinata,y=y_koordinata,w=sirina,h=visina)[@opis_drugog_znaka(...]

U uglatim zagradama nalazi se ime slike, slijedi znak ":" pa vrsta prometnog znaka, znak "@" te u oblim zagradama koordinate gornje lijeve točke pravokutnika te širina i visina. Slike za koje ne postoji zapis u ovakvom formatu neće se koristiti

u evaluaciji. Datoteka s točnim pozicijama prometnih znakova mora se nalaziti u istom direktoriju sa slikama kojima se evaluira postupak.

4.4. Pokretanje evaluacije

Program za evaluaciju, tekstualna datoteka s parametrima i izvršna datoteka postupaka detekcije moraju biti u istom direktoriju. Program se pokreće tako da joj se u komandnoj liniji zada ime datoteke s parametrima. Npr. na Windows platformi treba ići na *Start → Run...,* upisati "cmd", u novootvorenom prozoru pozicionirati se u direktorij sa programom te ga pozvati naredbom "evaluacija.py parametri.txt". Program će tada iz zadane tekstualne datoteke pročitati parametre i u ovisnosti o njima pokrenuti evaluaciju. Na početku rada ispisuju se konfiguracijski parametri trenutne evaluacije. Rezultat rada je ispis broja obrađenih znakova, točnih detekcija, pogrešnih detekcija i nedetektiranih znakova.

5. Eksperimentalni rezultati

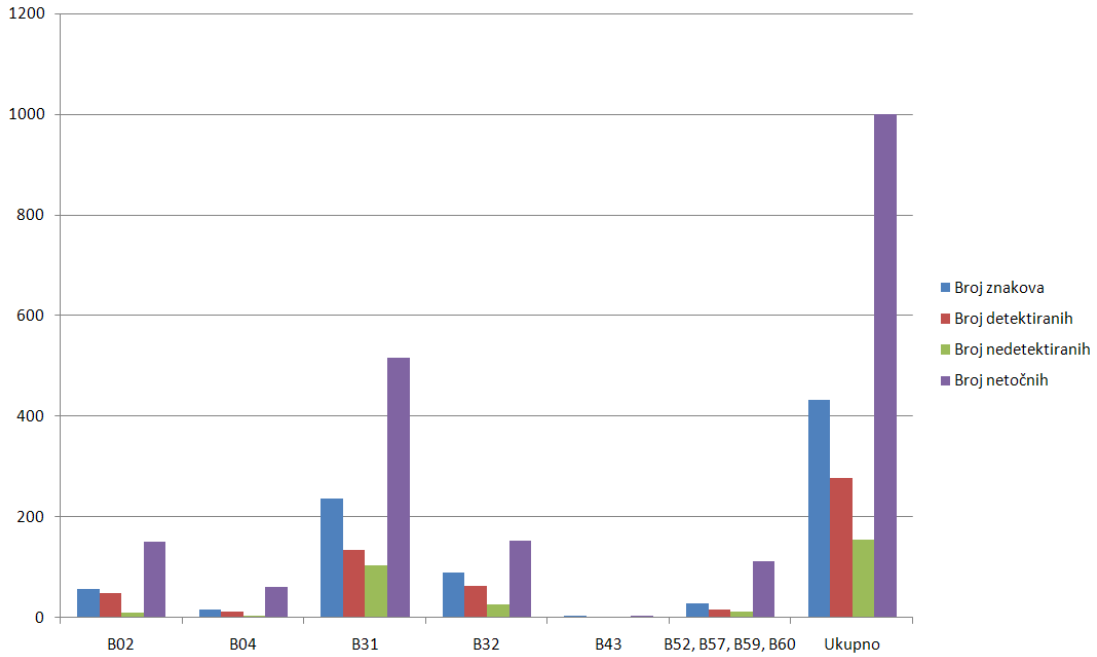
U ovom poglavlju prikazani su rezultati detekcije prometnih znakova na velikom skupu slika. Spomenuta dva postupka detekcije evaluirana su na skupu okruglih prometnih znakova koji su prikazani u tablici 1.

Tablica 1 – Vrste prometnih znakova

				
B02	B04	B31	B32	B38
				
B43	B52	B57	B59	B60

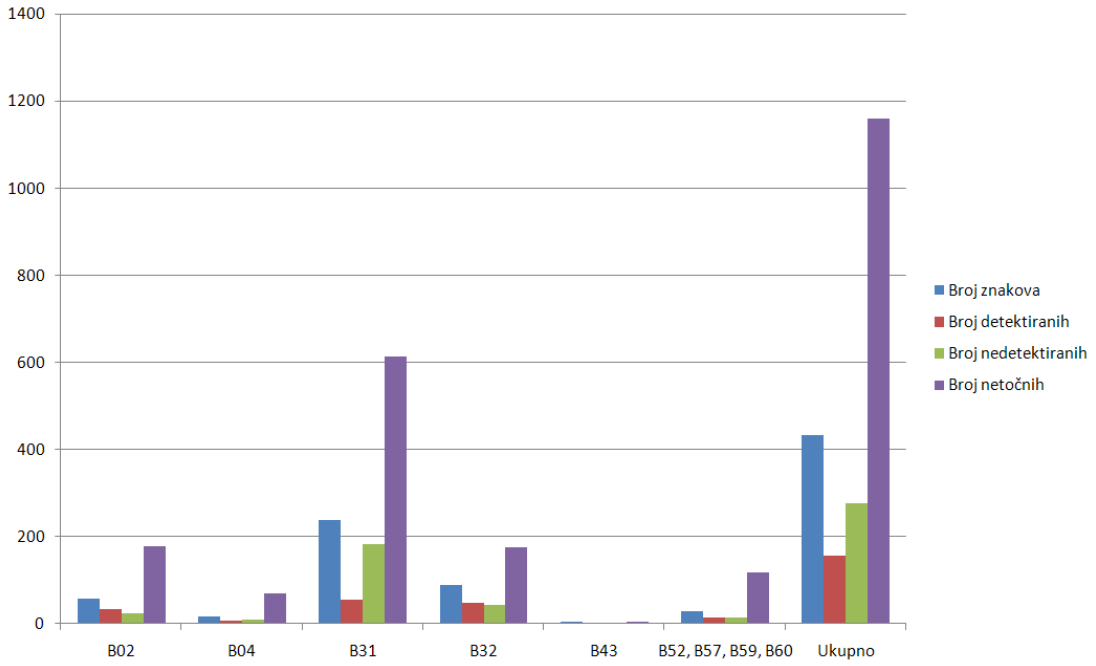
5.1. Pronalaženje prometnih znakova na temelju boje

Crveni znakovi evaluirani su sljedećom konfiguracijom: hueLo = 5.5, hueHi = 0.3, satLo = 0.1, satHi = 2, intLo = 0.15, intHi = 0.85, a plavi: hueLo = 3.3, hueHi = 4, satLo = 0.1, satHi = 2, intLo = 0.15, intHi = 0.85. Grafički prikaz rezultata za 10-postotno preklapanje nalazi se na slici 2. Ukupno je od 433 prometna znaka točno detektiran 281 (64%), a 1099 detekcija je bilo pogrešno.



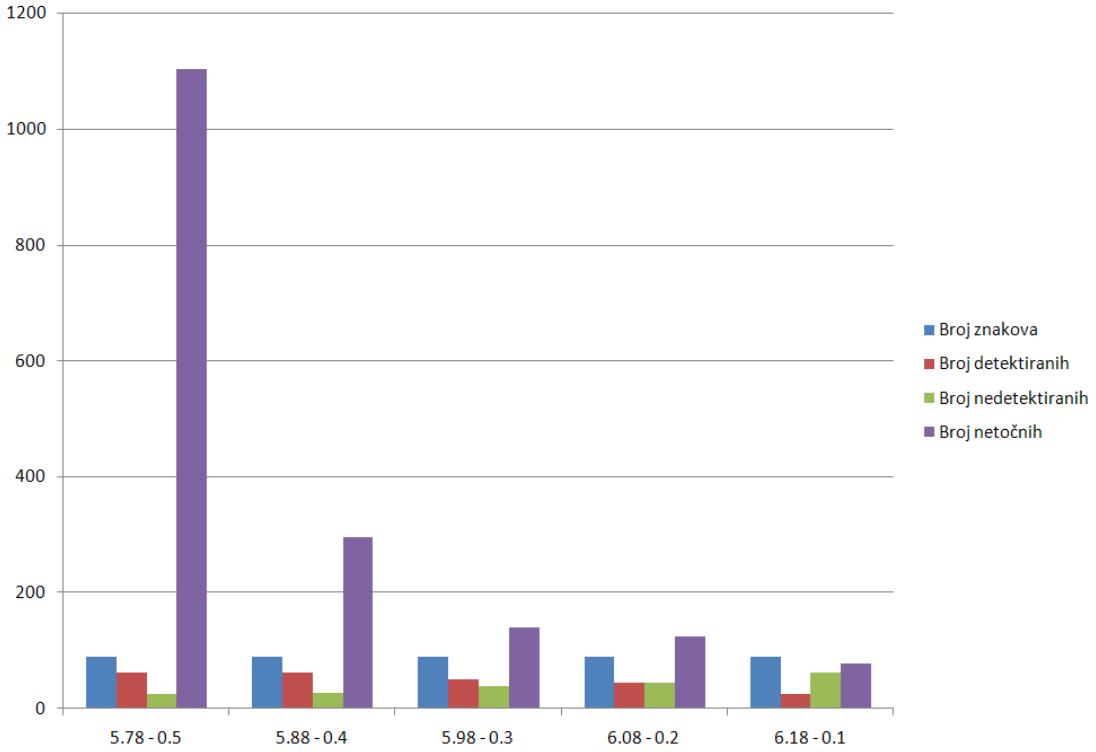
Slika 2 – Rezultati detekcije algoritmom signsHSI uz zadano preklapanje od 10%

Rezultati za 70-postotno preklapanje prikazani su na slici 3. Od 433 prometna znaka detektirano ih je 157 (36%), a 1162 detekcije su bile pogrešne. Kao što vidimo rezultati su znatno lošiji nego za 10-postotno preklapanja. Razlog tomu je što prometni znakovi nisu jednobojni pa zbog drugih boja unutar znaka postupak često ne označi cijeli prometni znak nego samo dio, npr. gornji i donji dio odvojeno. Zbog toga udio preklapanja s točnim znakom nije dovoljno velik pa program to proglašava pogrešnom detekcijom. Osim toga, na određenom broju slika prometni znakovi su premali ili preslabih boja pa ih postupak ne može pronaći.



Slika 3 – Rezultati detekcije algoritmom signsHSI uz zadano preklapanje od 70%

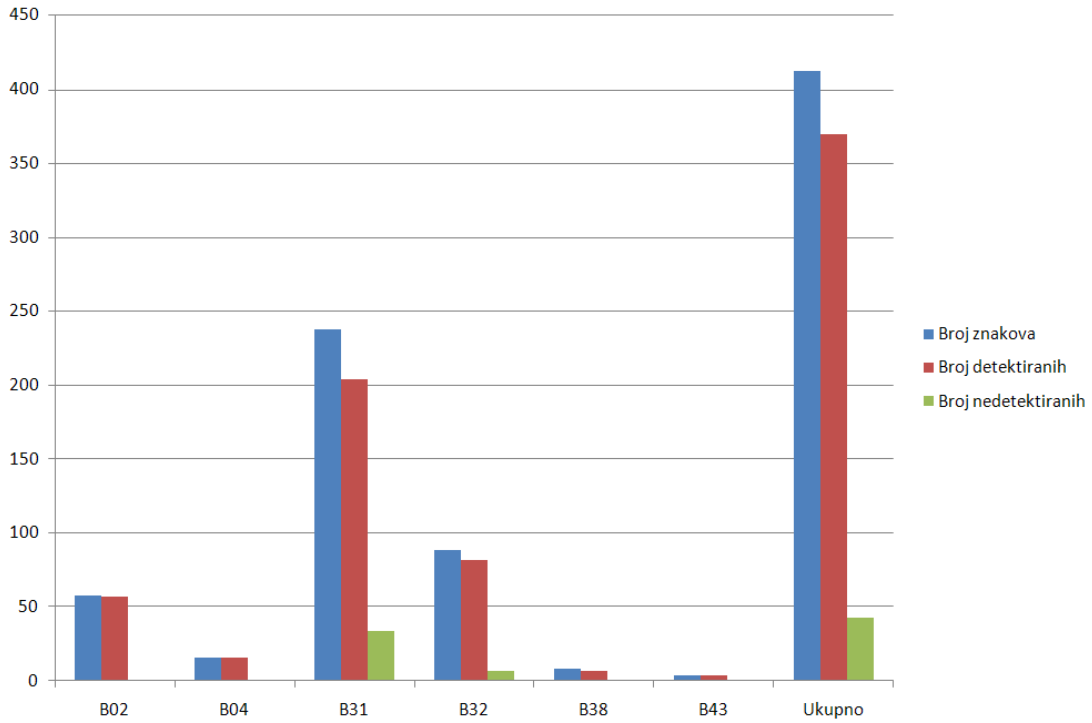
Kako rezultati detekcije ovise o intervalu za nijansu boje prikazano je na slici 4. Rezultat je dobiven za znakove B32 uz 10-postotno preklapanje. Vidimo da što je veći interval za nijansu boje veći je i postotak detektiranih znakova, ali i broj pogrešnih detekcija. Rezultat je logičan ako znamo da postupak detekcije na osnovu intervala za nijansu boje određuje koji pikseli mogu biti prometni znak, a koji ne pa za veći interval daje i više detekcija – točnih i netočnih.



Slika 4 – Rezultati detekcija algoritmom signsHSI za različite intervale nijanse boje

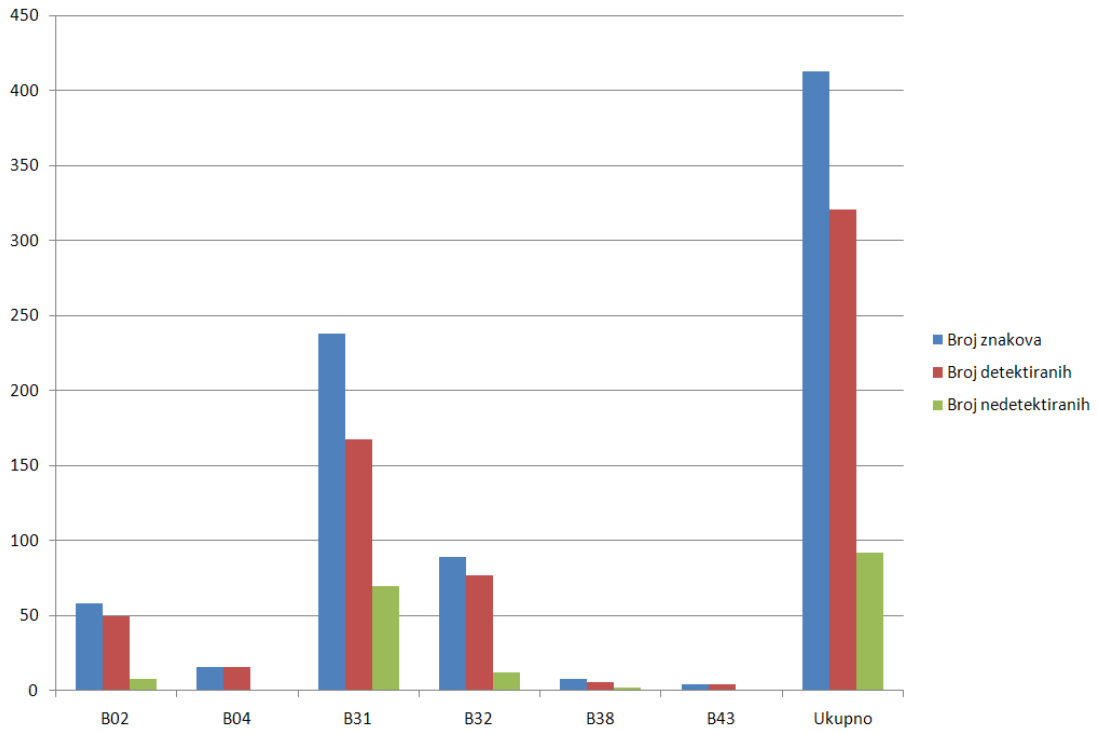
5.2. Pronalaženje prometnih znakova Houghovom transformacijom za kružnice

Mogućnost korištenja više konfiguracija ljuske u jednoj evaluaciji pokazala se vrlo korisnom pri evaluaciji ovog postupka zato što ovaj postupak omogućuje promjenu radijusa kružnice prometnog znaka. Budući da u skupu slika ima prometnih znakova različitih veličina i različitog intenziteta boje evaluacija sa samo jednom konfiguracijom postupka nije davala zadovoljavajuće rezultate. Sa 16 različitih konfiguracija s različitim vrijednostima za radijus, nijansu i intenzitet boje te uz 10-postotno preklapanje od 413 znakova detektirano ih je 370 (90%). Broj pogrešnih detekcija sveukupno je 7856 zbog toga što je svaka slika evaluirana sa 16 konfiguracija pa je i vjerojatnost za pogrešne detekcije velika. Rezultati su grafički prikazani na slici 5.



Slika 5 – Rezultati detekcije algoritmom houghCir uz zadano preklapanje od 10%

Rezultati za 70-postotno preklapanje prikazani su na slici 6. Od 413 znakova detektiran je 321 (78%), a ukupno je bilo 9384 pogrešne detekcije.



Slika 6 – Rezultati detekcije algoritmom houghCir uz zadano preklapanje od 70%

Ovaj postupak se pokazao jako osjetljivim na parametre. Budući da na slikama ima prometnih znakova različitih veličina i da su slikani pod različitim vremenskim uvjetima pa imaju različite nijanse i intenzitete boja, trebaju se i evaluirati s različitim parametrima. Na primjeru evaluacije sa 16 konfiguracija pokazano je da ovaj postupak može detektirati veliki postotak znakova, ako mu se zadaju odgovarajući parametri.

6. Zaključak

Razvijeni program koristan je za automatiziranu evaluaciju postupaka detekcije objekata na velikom skupu slika. Primjena programa može biti u ispitivanju kvalitete implementiranih postupaka detekcije objekata, usporedbi rezultata različitih postupaka ili pri razvoju takvih postupaka za lakšu i bržu provjeru rezultata koje daju. Osim toga, primjena bi mogla biti u provjeravanju vidljivosti prometnih znakova na određenoj prometnoj dionici. Pri tome je vrlo korisna mogućnost odabira vrste prometnih znakova regularnim izrazima tako da se traže samo oni prometni znakovi koje korišteni postupak detekcije može pronaći.

Budući da svaki postupak ima parametre kojima se određuje kakvi objekti se traže vrlo bitno je da se u jednoj evaluaciji mogu koristiti različite postavke parametara. Ta se mogućnost može koristiti ako u skupu slika ima prometnih znakova različitih dimenzija, različitog intenziteta boje ili ako se evaluacija radi na raznobojnim znakovima pa treba koristiti različite nijanse boje. Za što bolje rezultate evaluacije važno je znati koji parametri postupka odgovaraju zadanim objektima na slikama.

Program za evaluaciju razvijen je i testiran na dva postupka detekcije prometnih znakova, međutim može se koristiti i za druge postupke koji su prilagođeni tako da ispisuju koordinate pravokutnika koji označava detektirani objekt. Dakle, programska okolina može se koristiti za puno različitih postupaka detekcije neovisno o njihovoj implementaciji i upravo to je njena najvažnija karakteristika.

Postupak detekcije koji pronalazi prometne znakove na temelju boje dao je sljedeće rezultate: za 10-postotni udio preklapanja od ukupno 433 prometna znaka točno je detektirao 281 (64%), a 1099 detekcija je bilo pogrešno. Veliki broj pogrešnih detekcija se može objasniti tako što postupak za detekciju u obzir uzima samo boju pa ako mu se zada npr. crvena boja on će sve crvene predmete na slici (automobili, krovovi kuća, reklame i sl.) proglasiti prometnim znakovima. Drugi postupak detekcije pomoću Houghove transformacije za kružnice uzima u obzir i

oblik prometnog znaka. Rezultati ovog postupka su vidljivo bolji: za 10-postotno preklapanje od 413 znakova detektirao je 370 (90%). Veliki broj (7856) pogrešnih detekcija rezultat je toga što je svaka slika evaluirana sa 16 različitih konfiguracija parametara, a budući da se radilo na velikom skupu raznovrsnih slika to je bilo nužno da bi se dobio ovoliki postotak detektiranih znakova.

Program bi se mogao proširiti tako da traži prometne znakove samo na određenim pozicijama na slici. Evaluacijama za različite pozicije mogla bi se pronaći ona na kojoj se pojavljuje najveći broj prometnih znakova. U postupku u kojem se može zadati radijus kružnice prometnog znaka mogle bi se napraviti evaluacije s različitim vrijednostima za radijus i tako odrediti najveći, najmanji i prosječni radijus prometnog znaka. U postupku detekcije prometnih znakova na temelju boje evaluacijama s različitim intervalima nijanse boje mogao bi se odrediti optimalan interval za koji postupak detektira najveći broj znakova.

Literatura

[1] Šverko, M. Pronalaženje prometnih znakova Houghovom transformacijom za kružnice. Završni rad, Fakultet elektrotehnike i računarstva, 2009.

[2] Babić, T. Prepoznavanje prometnih znakova na temelju boje. Projekt, Fakultet elektrotehnike i računarstva, 2008.

[3] Brkić, K. MASTIF Marker – upute za korištenje, 2009.

[4] The Python Tutorial, 25.4.2009., *The Python Tutorial*, http://docs.python.org/tutorial/index.html, 7.5.2009.

[5] The Python Standard Library, 5.5.2009., *The Python Standard Library*, http://docs.python.org/tutorial/index.html, 15.5.2009.

[6] Fredrik Lundh, Tutorial, 12.3.2002., *Python Imaging Library Overview*, http://www.pythonware.com/library/pil/handbook/introduction.htm, 21.5.2009.

Sažetak

U ovom radu proučavaju se postojeći postupci detekcije objekata u računalnom vidu te se razvija programska okolina za evaluaciju takvih postupaka. Postupci detekcije se prethodno prilagođavaju da bi se mogli koristiti s razvijenom programskom okolinom. Evaluacija postupaka detekcije radi se na velikom skupu slika za koje postoji datoteka sa zapisima točnih pozicija objekata te se one uspoređuju s pozicijama objekata koje pronađe postupak detekcije. Ako je udio preklapanja pravokutnika veći ili jednak zadanom, objekt je točno detektiran. Opisana je procedura prevođenja i korištenja razvijene programske okoline te je dana definicija datoteke s parametrima. Prikazani su rezultati za nekoliko evaluacija na dva postupka detekcije prometnih znakova.

Ključne riječi: računalni vid, detekcija objekata, evaluacija postupaka detekcije, prometni znakovi

Abstract

In this work we analyze existing algorithms of object detection in computer vision and implement software environment for evaluation of those algorithms. Before it's possible to evaluate algorithms, it is necessary to modify them so that implemented software environment can use them. Evaluation is performed on large set of images for which there is a file with exact positions of objects. If percentage of overlapping between the square of detection and square of exact object exceeds or is equal to given percentage, detection is correct. Usage of software enviroment is described and the definition of file with parameters is given. Results for few evaluations of two algorithms of object detection are also provided.

Keywords: computer vision, object detection, evaluation of algorithms, traffic signs