SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 438

PRONALAŽENJE PROMETNIH ZNAKOVA HOUGHOVOM TRANSFORMACIJOM ZA KRUŽNICE

Maja Šverko

Zagreb, siječanj 2009

Sadržaj:

1. Uvod

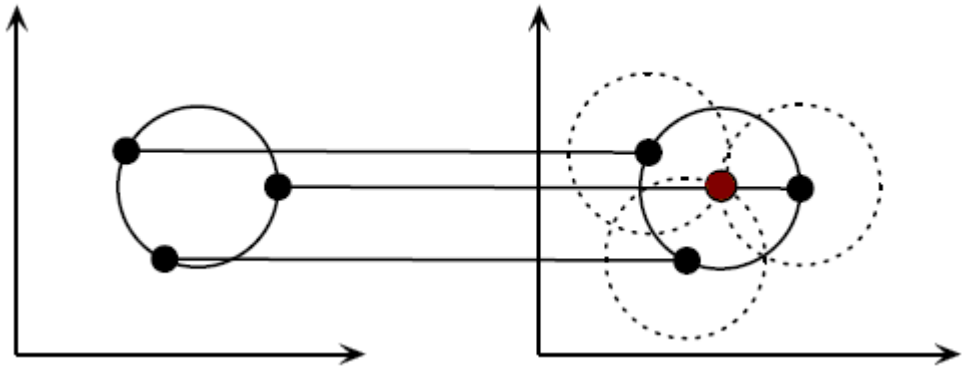
Cilj ovoga rada je prepoznavanje okruglih znakova izričitih naredbi s crvenim obrubom na snimkama ili slikama snimljenima iz automobila u pokretu. Metoda uz pomoć koje ćemo to pokušati ostvariti je Houghova transformacija za kružnice. U toj transformaciji svaki piksel crvene boje glasuje za sve kružnice kojima bi mogao pripadati. Ukoliko na slici imamo crvenu kružnicu, za nju će glasati svaki njezin piksel i tako će nam je označiti u akumulatorskom polju. Na ovaj ćemo način, uz pomoć kružnica od kojih se sastoji obrub znaka, pokušati strojno pronaći takve znakove.

Strojna detekcija znakova može biti korisna u automatiziranoj evaluaciji prometne signalizacije. Na taj način možemo brzo i efikasno provjeriti da li na nekom odsječku ceste znakovi zadovoljavaju standarde vidljivosti. Uz to postoje i neke druge moguće primjene. Takav sustav detekcije znakova mogao bi biti koristan u sustavu za pomoć vozačima ili u autonomnoj vožnji.

Rad je strukturiran kako slijedi. Prvo uvodimo teoretske osnove Houghove transformacije u poglavlju 2, a zatim, u poglavlju 3., potrebne pomoćne metode. U poglavlju 4. objašnjavamo programsku izvedbu, rad s programskim okruženjem i korisničkim sučeljem. Uspješnost algoritma i njegova ograničenja prikazujemo u poglavlju 5. Na kraju rada, u poglavlju 6. komentiramo postignute rezultate i ocjenjujemo primjenjivost algoritma.

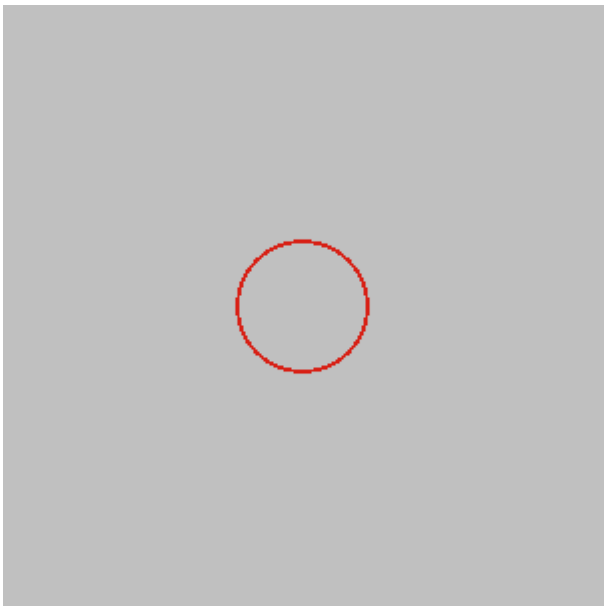
2. Pronalaženje kružnica Houghovom transformacijom

Houghova transformacija je robusna tehnika procjenjivanja parametara temeljena na načelu glasanja. U računalnom vidu, Houghovom transformacijom pronalazimo parametre nesavršenih primjeraka zadane klase oblika. Primjerci se pronalaze kao maksimumi akumulacijskog polja u kojem se skupljaju glasovi elemenata izvorne slike (piksela). Houghovom transformacijom mogu se pronaći oblici čak i u slučajevima kad je detektirano znatno manje od polovice slikovnih elemenata koji ih određuju [1].

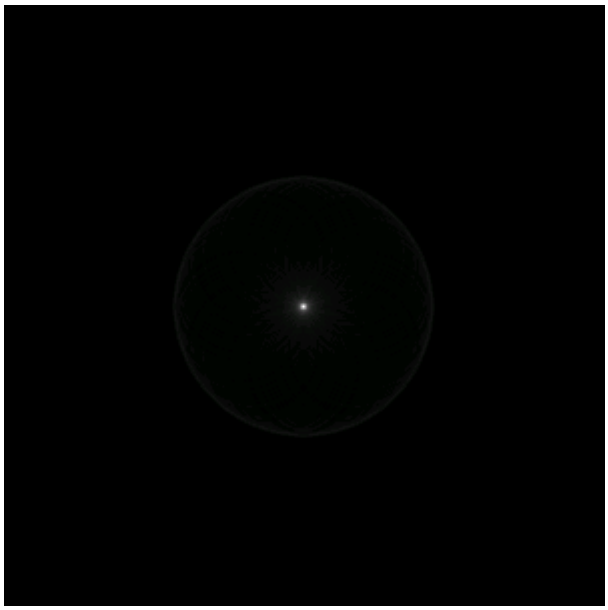


Slika 1: prikaz glasanja elemenata slike i maksimalne vrijednosti u središtu kružnice u akumulacijskom polju [2]

Houghova transformacija za kružnice služi za traženje kružnica određenih radijusa koji zadovoljavaju neke kriterije (u ovom je slučaju to boja slikovnih elemenata kružnice). To se ostvaruje tako da se oko svakog elementa (piksela) slike koji zadovoljava uvjete u akumulacijskom polju iscrta kružnica zadanog radijusa. Tako svaki takav element slike „glasuje“ za središte kružnica koje su na zadanoj udaljenosti od njega. Ukoliko imamo na slici kružnicu odgovarajućeg radijusa, svi će njezini elementi između ostaloga glasati i za središte kružnice. Time će se na tom mjestu pojaviti lokalni maksimum u akumulacijskom polju te tako „označiti“ centar kružnice.



Slika 2: kružnica crvene (zadane) boje radijusa približno R



Slika 3: akumulacijsko polje dobiveno glasanjem svakog crvenog piksela za svoju kružnicu radijusa R

Glasovi se u akumulacijskom polju dodaju na pozicije koordinata dobivenih sljedećim izrazima [3]:

$$x = a + R\cos(\theta)$$
$$y = b + R\sin(\theta)$$

Gdje je Θ element iz intervala [0, 360], x i y su koordinate elemenata slike u akumulacijskom polju, a i b su koordinate elemenata na slici, a R je zadani radijus kružnice.

Houghovu transformaciju za kružnice iskoristiti ćemo kao metodu za pronalaženje okruglih znakova izričitih naredbi s crvenim obrubom.

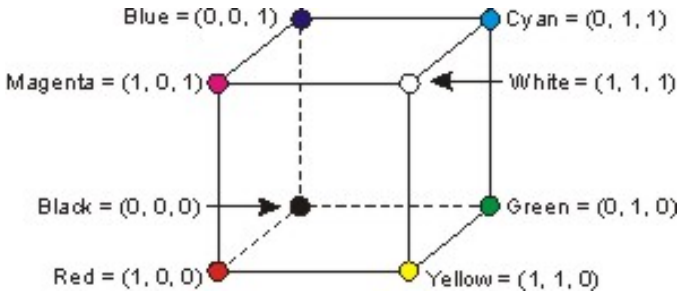
Slika 4: primjeri znakova izričitih naredbi s crvenim obrubom [4]

3. Pomoćne metode

3.1. Prevođenje iz modela RGB u model HSI

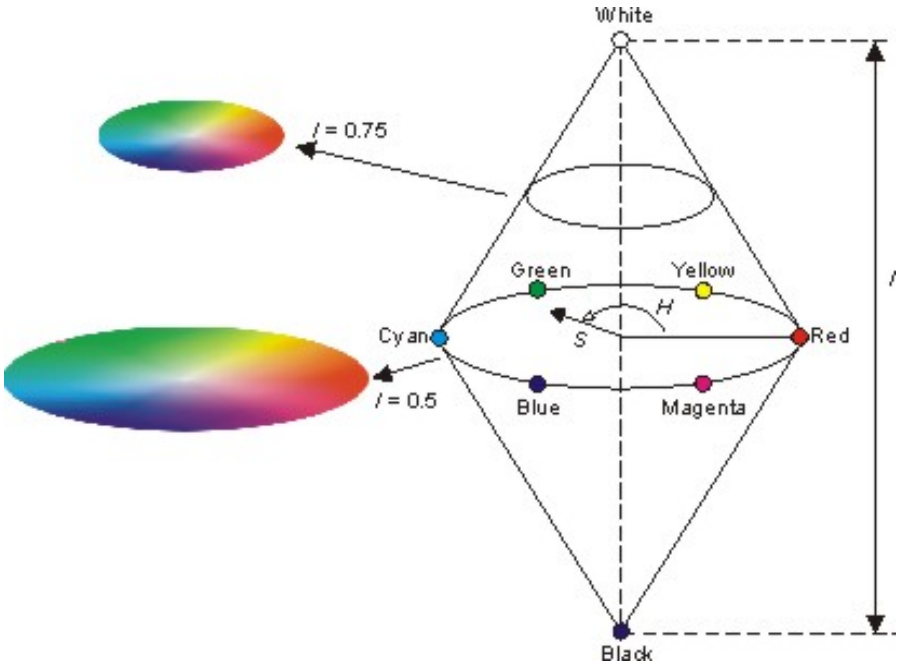
RGB je najpoznatiji i raširenij prostor boja. U tom modelu boja svaka je boja predstavljena s tri vrijednosti; crvena (red), zelena (green) i plava (blue). U računalnim sustavima najčešće se koristi model prikaza u kojem se vrijednosti komponenti RGB kreću od 0 do 255 tako da je crna boja prikazana kao (0,0,0) dok je bijela prikazana kao (255,255,255). Sivi su tonovi prikazani jednakim količinama komponenti R, G i B.

Vrijednosti RGB se također mogu prikazivati vrijednostima u intervalu od 0 do 1, gdje se bijela boja prikazuje kao (1, 1, 1).



Slika 5: prikaz prostora boja RGB [5]

Model boja HSI je važan i praktičan za obradu slika jer predstavlja boje na način sličan ljudskom viđenju boja. HSI model boja svaku boju predstavlja s tri komponente; nijansa (hue), zasićenje (saturation) i intenzitet (intensity).



Slika 6: prikaz prostora boja HSI [6]

HSI vrijednosti se računaju uz pomoć normaliziranih R, G i B vrijednosti koje dobijemo tako da svaku od njih podijelimo sa njihovim zajedničkim zbrojem [7]:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}. \quad (1)$$

Vrijednost nijanse (hue) se dobiva u stupnjevima. U slučaju da je komponenta B veća od komponente G, njena je vrijednost u intervalu od π do 2π dok je u slučaju da je komponenta B manja ili jednaka komponenti G njena vrijednost u intervalu od 0 do π [7].

$$H = cos^{-1}\left[\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right] \quad \text{za B<G} \quad (2)$$

$$H = 360 - H \quad \text{za B>G} \quad (3)$$

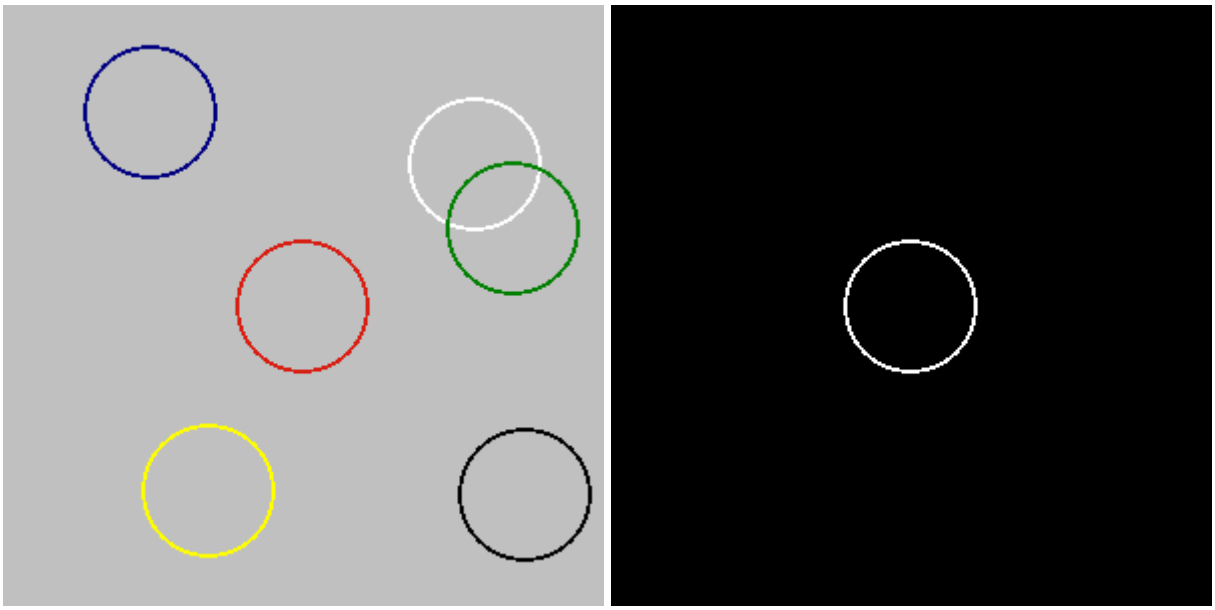
Intenzitet (intensity) i zasićenje (saturation) imaju vrijednosti u intervalu od 0 do 1 i računaju se sljedećim izrazima [7]:

$$S_p = 1 - 3\ min(r, g, b) \tag{4}$$

$$I = \frac{1}{3}(R + G + B) \tag{5}$$

3.2. Detekcija boje u prostoru HSI i binarna slika

Za detektiranje elemenata slike koji smiju glasati za svoje kružnice koristimo boju. Boja se detektira u HSI sustavu unutar gornjih i donjih granica za hue, saturation i intensity. Rezultat detekcije boje je binarna slika, točnije slika koja se sastoji od samo dvije vrijednosti; minimalne i maksimalne odnosno crne i bijele. Binarnu sliku dobijemo tako da ispitamo svaki element slike i ukoliko se on nalazi unutar zadanih intervala H, S i I, označimo ga bijelom bojom dok ga u suprotnom označimo crnom. Takva slika bijelom bojom prikazuje sva područja koja zadovoljavaju naše kriterije boje.



Slika 7: binarna slika dobivena detekcijom crvene boje na prethodnoj slici

3.3. Narastanje područja u binarnoj slici

Narastanje područja (flood fill) je algoritam koji određuje područje obojeno istom bojom na slici kao jednu cjelinu. Svaki se pronađeni bijeli element na binarnoj slici označava nekom vrijednošću te se ulazi u rekurzivnu funkciju, gdje se za svaki njegov susjedni element provjerava je li bijele boje i rekurzivno, jesu li bijele boje njegovi susjedi. Svi susjedi se označavaju istom vrijednošću, iz čega slijedi je skupina elemenata jedinstveno označena. Kada funkcija pronađe novi bijeli element, ponavlja se isti postupak, ovoga puta sa novom vrijednošću.

Pseudokodom možemo postupak opisati na sljedeći način:

narastanje(*element, ciljna bolja, zamjenska boja*)

1. ako boja *elementa* nije jednaka *ciljnoj boji*, izađi iz petlje
2. ako je boja *elementa* jednaka *zamjenskoj boji*, izađi iz petlje
3. oboji *element zamjenskom bojom.*
4. pokreni funkciju **narastanje***(jedan korak lijevo od elementa, ciljna boja, zamjenska boja)*

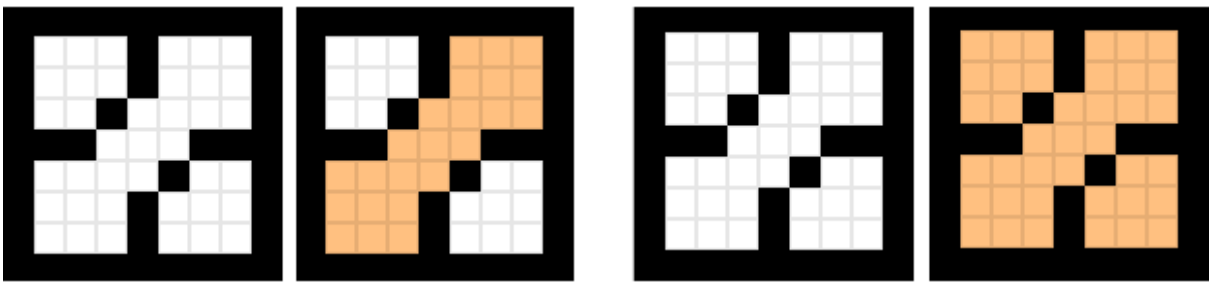
pokreni funkciju **narastanje***(jedan korak desno od elementa, ciljna boja, zamjenska boja)*

pokreni funkciju **narastanje***(jedan korak iznad elementa, ciljna boja, zamjenska boja)*

pokreni funkciju **narastanje***(jedan korak ispod elementa, ciljna boja, zamjenska boja)*

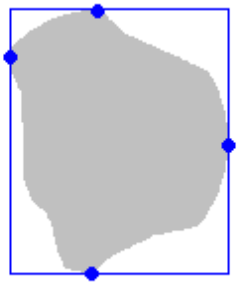
5. izađi iz petlje

Koristimo algoritam narastanja područja u četiri smjera (gore, dolje, lijevo i desno), no postoji i algoritam narastanja područja u osam smjerova koji provjerava i elemente slike koji se dodiruju vrhovima a ne samo stranicama.



Slika 8: Prikaz razlike između narastanja područja u 4 smjera i u 8 smjerova. [8]

Pomoću označenih vrijednosti područja iste boje traže se skupine elemenata slike. Svakoj se skupini pritom određuje opisani pravokutnik pronađenog područja kao na slici 10. Ako je veličina skupine veća od minimalne zadane, funkcija vraća koordinate opisanog pravokutnika. Postupak se ponavlja za svaku skupinu bijelih elemenata na slici.



Slika 9: primjer određivanja krajnjih točaka skupine

[1] Preuzeto s: http://en.wikipedia.org/wiki/Flood_fill

4. Programska izvedba

Programsko okruženje dobavlja sliku i zapisuje ju u memoriju u obliku: „RGBRGBRGB…“. U istom obliku sa novim vrijednostima predstavlja se i HSI slika gdje su vrijednosti zapisane u obliku: „HSIHSIHSI...“.

Pri obradi slike, ljuska poziva funkciju `alg_znakovi::process` koja je deklarirana u klasi `alg_znakovi` i to na sljedeći način:

```
virtual void process(
  const img_vectorAbstract& src,
  const win_event_vectorAbstract& events,
  int msDayUtc);
```

4.1. Konverzija RGB slike u HSI sliku

Radi intuitivnijeg i jednostavnijeg izdvajanja traženih nijansi na samom početku vrijednosti RGB elemenata izvorne slike preračunavamo u vrijednosti HSI. Nove se vrijednosti dobivaju uz pomoć izraza opisanih u poglavlju 3.1.

Konverzija slike RGB u sliku HSI izvodi se uz pomoć funkcije RGBtoHSI:

```
void RGBtoHSI(
            int width,
            int height,
            const unsigned char* psrc,
            float* pdst );
```

Funkciji predajemo širinu i visinu slike, dobivene atributima `imgs_[0].fmt().height()`, odnosno `imgs_[0].fmt().width()`, izvornu sliku RGB `psrc` i odredišno polje za spremanje slike HSI `pdst`.

Funkcija preko predane visine i širine prolazi kroz sve elemente izvorne slike i za svaki pojedini element izračunava vrijednosti H, S i I te ih sprema na mjesto tog elementa u odredišnoj slici, točnije u `float* pdst`.

4.2. Detekcija boje u prostoru HSI i binarna slika

Iz izvorne slike za početak moramo na neki način izdvojiti elemente slike koji zadovoljavaju naše kriterije. To su u ovom slučaju elementi crvene boje, točnije nekoliko nijansi crvene boje koje variraju u uskom pojasu vrijednosti oko točne boje obruba okruglih znakova izričite opasnosti. Izdvajamo ih tako da sve elemente koji zadovoljavaju kriterije obojimo bijelom bojom, dok sve ostale obojimo crnom. Na taj način nastaje binarna slika; slika na kojoj su bijeli elementi oni s kojima ćemo dalje raditi. Taj se postupak odvija u funkciji nadjiBoju:

```
void nadjiBoju (
                int width,
                int height,
                float* NorSlika,
                char* BinSlika,
                double hueLo,
                double hueHi,
                double satLo,
                double satHi,
                double intLo,
                double intHi);
```

Funkciji predajemo visinu i širinu slike, zatim izvornu sliku u obliku HSI `NorSlika` te binarnu sliku `BinSlika` koja za svaki element ima samo jednu vrijednost (za razliku od prijašnje tri za slike HSI i RGB, binarna slika ima samo jednu vrijednost na svakom elementu koja može biti minimalna – crna, ili maksimalna – bijela). Preostalih šest vrijednosti koje predajemo funkciji su zapravo granične vrijednosti HSI. (hueLo, hueHi, satLo, satHi, intLo i intHi)

U funkciji hueLo i hueHi predstavljaju granice intervala topline boje, satLo i satHi predstavljaju granice intervala zasićenja, a intLo i intHi predstavljaju granice intervala za intenzitet boje. Također treba primijetiti da se za crvenu boju (koja se nalazi na oba kraja spektra HSI) zadaju obrnuto vrijednosti za toplinu (hue) boje. Dakle, ako je hueLo zadan tako da bude veći od hueHi onda interval u kojem će se gledati neće biti [hueLo, hueHi], već [0, hueHi] U [hueLo, 360˚].

Primjer vrijednosti za traženje okruglih znakova izričitih naredbi s crvenim obrubom:

```
hueLo_(5.95),
hueHi_(0.05),
satLo_(0.09),
satHi_(1),
intLo_(0.15),
intHi_(0.833)
```

4.3. Houghova transformacija za kružnicu

Kada jednom imamo binarnu sliku, sljedeći je korak traženje kružnica zadanog radijusa. To se ostvaruje glasanjem elemenata binarne slike. Glasovi se prikupljaju u akumulacijsko polje u okviru funkcije `houghCirc`:

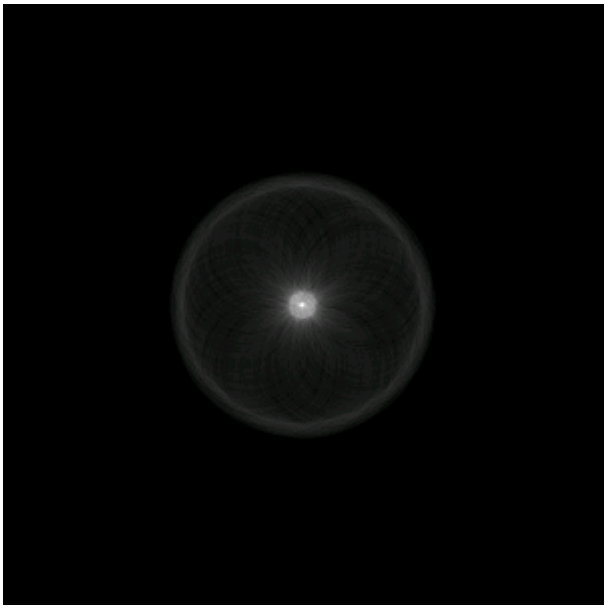
```
void houghCirc (
                    int width,
                    int height,
                    char* imgBin,
                    float* imgAcc,
                    int circRad);
```

Funkciji predajemo visinu i širinu slike, dobivenu binarnu sliku `imgBin` i akumulacijsko polje za upis rezultata `imgAcc` te radijus kružnica koje tražimo `circRad`. Funkcija prolazi po svim elementima slike ta za svaki bijeli element u akumulacijskom polju upisuje jedan „glas“ na svaki element koji je za zadani radijus udaljen od elementa koji glasuje. Ti elementi stoje u obliku kružnice tog istog radijusa oko elementa koji glasuje, a izračunavaju se izrazima iz poglavlja 2.

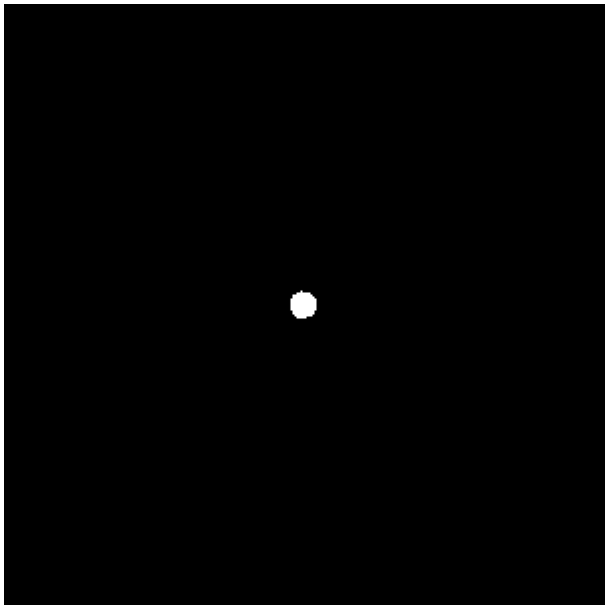
U akumulacijskom polju glasovi se akumuliraju, tj zbrajaju se svi glasovi na istome mjestu. Ukoliko na slici postoji kružnica zadanog radijusa svi će njegovi elementi između ostaloga glasati i za središte te kružnice te će tako vrijednost na poziciji središta kružnice u akumulacijskom polju biti maksimalna.

4.4. Narastanje područja

U idealnoj situaciji, kružnica koji ima širinu crte jedan element ovim postupkom u akumulacijskom polju na maksimalnu bi vrijednost postavio točno jedan element slike, tj. točno središte kružnice. Međutim znakovi imaju široke obrube koji nisu idealne kružnice zbog kuta snimanja, utjecaja osvjetljenja, smetnji i ostalih vanjskih utjecaja te oni u akumulacijskom polju postavljaju na maksimum nekoliko elemenata. Da bi mogli odrediti točno središte kružnice, za početak moramo izdvojiti maksimalne vrijednosti u akumulacijskom polju.



Slika 10: centar kruga pronađen Houghovom transformacijom



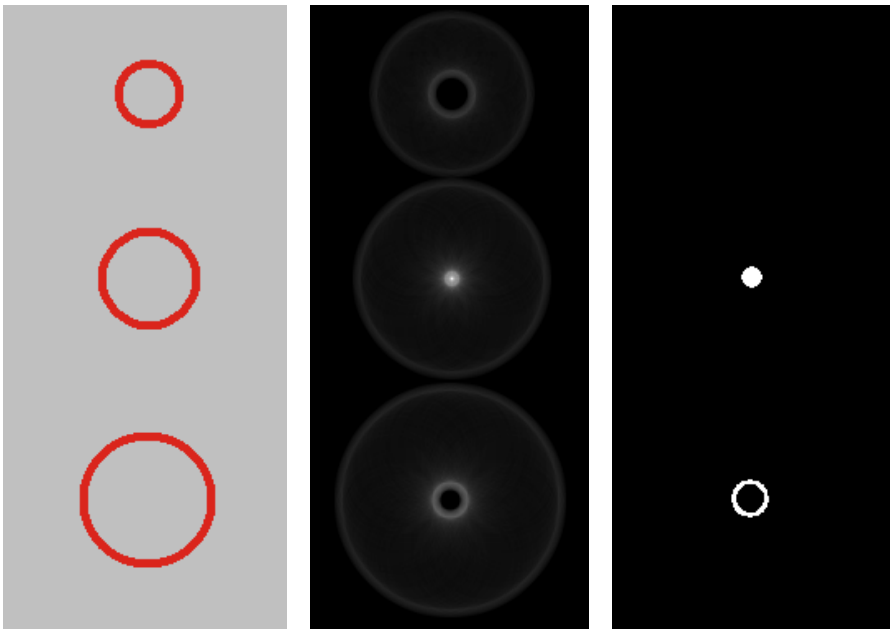
Slika 11: centar kruga istaknut izdvajanjem maksimalnih vrijednosti u akumulacijskom polju

Nakon što izdvojimo maksimalne vrijednosti polja u obliku binarne slike (minimalna i maksimalna vrijednost). Uz pomoć funkcije narastanja područja izdvojimo sve takve skupine elemenata veće od neke minimalne vrijednosti.

```
int *narastanje (
                    int width,
                    int height,
                    char* imgBin,
                    int circRad,
                    int thSzRegion);
```

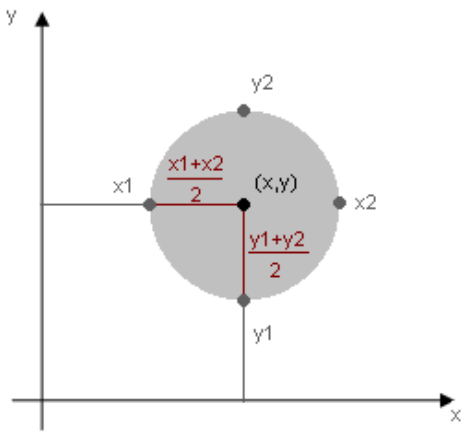
Funkcija prima pet argumenata; to su visina i širina slike, binarna slika `imgBin` s izdvojenim maksimumima, zadani radijus kružnice `circRad` te minimalnu veličinu skupine elemenata slike `thSzRegion`. Funkcija u obliku polja vraća krajnje vrijednosti svakog pronađenog područja, to jest visinu najgornjeg i najdonjeg elementa te širinu onoga koji je najlijeviji i najdesniji (točke x1, x2, y1 i y2 na slici 13).

Kod detekcije kružnica, izdvojena vrijednost najčešće ima oblik kruga ako je radijus kružnice približno jednak zadanom ili kružnice ukoliko radijus pronađene kružnice nije isti kao zadani. To je prikazano na slici 12.

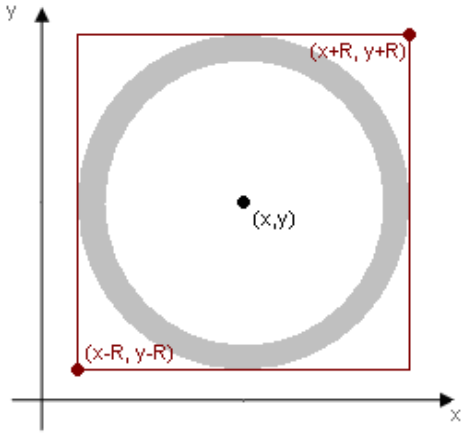


Slika 12: prikaz izdvajanja maksimuma kad je radijus kružnice manji, jednak i veći od zadanog radijusa

Budući da izdvojena područja nisu precizna iz njihovih krajnjih koordinata koje je vratila funkcija moramo izračunati njihovo središte. Kao što je prikazano na slici 13, središte tražimo tako da na pola podijelimo zbroj krajnjih x koordinata i krajnjih y koordinata područja.



Slika 13: traženje središta u izdvojenom području

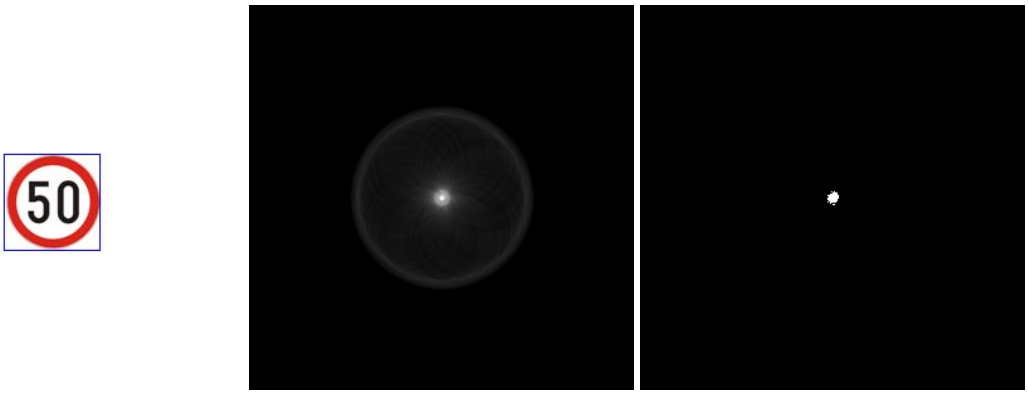


Slika 14: iscrtavanje kvadrata oko kružnice preko središta i radijusa

Pronađene znakove označavamo na izvornoj slici. Ljuska omogućava iscrtavanje vektorskih elemenata (anotacija) preko slike, uz pomoć njih iscrtati ćemo kvadrate stranice 2R oko kružnice, kao na slici 14. Za to koristimo funkciju ljuske `addRectangle`.

```
static void addRectangle(win_ann_abstract& ann,
                         const grid::Pixel& p0,
                         const grid::Pixel& p1,
                         int col, int width);
```

Algoritam po svom završetku predaje ljusci izvornu sliku s označenim znakovima, i dva međurezultata; akumulacijsko polje i središta pronađenih kružnica u binarnoj slici. Na kraju ljuska ispisuje rezultate kao što je prikazano na slici 15.



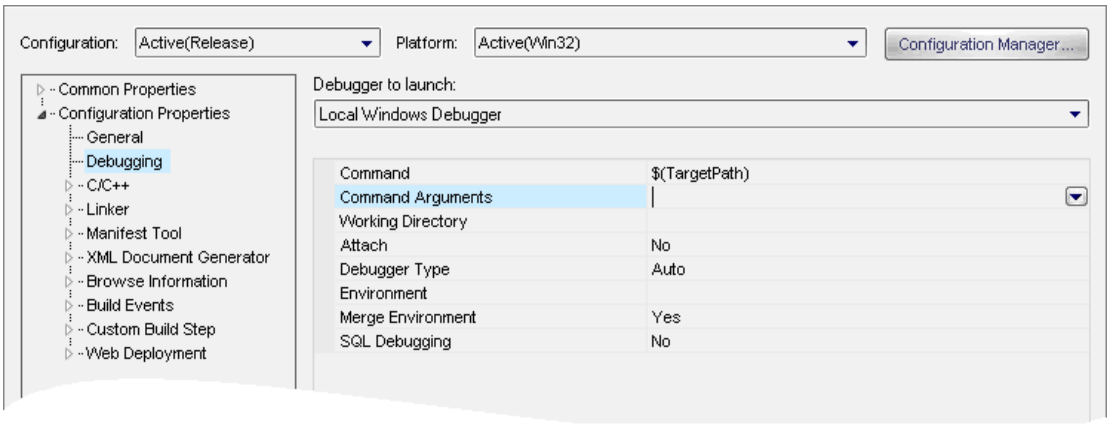
Slika 15: Rezultat izvođenja algoritma nad slikom znaka za ograničenje brzine

4.5. Rad s programskim okruženjem

Parametri naredbenog retka u programskom okruženju Visual Studio 2005 zadaju se u: *Project Properties* → *Debugging* → *Command Arguments* (Slika 15)

Naredbeni redak ljuske se sastoji od nekoliko osnovnih dijelova:

- putanja izvorne datoteke (source) `-sf`
- odredište dobivene datoteke (destination) `-df`
- algoritam koji se koristi `-a`
- interakcija za vrijeme provođenja algoritma (interactive) `-i`
- konfiguracija parametara (config) `-c`



Slika 16: Command Arguments - upis naredbe

Putanja izvorne datoteke se zadaje na sljedeće načine:

```
-sf=…\video.avi
-sf=…\video.avi#početna_slika-završna_slika
-sf=...\cijela_mapa/
```

Odredište dobivene datoteke:

```
-df=…\rezultat.avi
```

Algoritam koji se koristi:

```
-a=algoritam
```

Interakcija za vrijeme provođenja algoritma:

```
-i =“p n“
```

Neke opcije koje se mogu koristit su „`p n`“ obradi sljedeću (process next), „`p p`“ obradi prethodnu (process previous), „`p a 10`“ obradi sliku na poziciji (adresi) 10 (process address), „`p t`“ obradi trenutnu (process this).

Konfiguracija parametara:

```
-c = „<circRad><hueLo><hueHi><satLo><intLo>“
```

Parametri koje možemo mijenjati u konfiguraciji (config) ovise o samom algoritmu koji se koristi. Gore je naveden primjer za algoritam „`houghCirc`“. Parametri konfiguracije za houghCirc algoritam su redom; radijus, donja granica vrijednosti nijanse boje, gornja granica vrijednosti nijanse boje, donja granica vrijednosti zasićenja boje, donja granica vrijednosti intenziteta boje te radijus kružnice.

Nekoliko primjera potpune naredbe:

```
-sf=E:\video.wmv -a= houghCir
-sf=E:\video.wmv#1-5 -df=E:\temp\video.avi -a=houghCirc
-sf=C:\OkrugliZnakovi\znak.bmp -i="p t" -a=houghCir
-sf=C:\OkrugliZnakovi\znak.bmp -i="p n" -a=houghCir -c=“30 5.95 0.05
0.09 0.15“
```

4.6. Korisničko sučelje

Korisničko sučelje nam pruža nekoliko korisnih i praktičnih mogućnosti. Neke od tih primjerice kretanje kroz slike koje se obrađuju i one nisu ovisne o algoritmu koji se koristi. Konfiguracija algoritma s druge strane ovisi o algoritmu i omogućava nam da bez ponovnog pokretanje programa promijenimo neke parametre.

Postoji nekoliko načina na koje se možemo kretati kroz slike ili video. U prvom slučaju krećemo se po slikama unutar neke mape, u drugom po slikama nekog videa.

„p n“ - obradi sljedeću (process next)

„p p“ - obradi prethodnu (process previous)

„p a n“ - obradi sliku na poziciji (adresi) n (process address)

„p t“ - obradi trenutnu (process this)

Za izlazak iz programa koristimo kraticu „q“ (quit)

U algoritmu HoughCir možemo u konfiguraciji mijenjati nekoliko varijabli. To su radijus kružnice, granice nijanse boje (hue), donja granica zasićenja (saturation), donja granica intenziteta (intensity). Konfiguraciji pristupamo kraticom „c“ nakon čega se ispisuju trenutne vrijednosti parametara i traži se unos novih.

5. Eksperimentalni rezultati

5.1. Odabir reprezentativnog skupa slika

Odabirom reprezentativnog skupa slika pokušavamo pronaći ograničenja ovog algoritma, to jest osjetljivost algoritma na smetnje poput slabog osvjetljenja, kuta snimanja i zamućenja. Također, želimo ispitati koliko je algoritam osjetljiv na promjenu radijusa znakova budući se u algoritmu zadaje fiksni radijus.

Prvenstveno odabiremo slike sa jasno istaknutim znakovima, čistih boja i bez zamućenja za provjeru rada algoritma u dobrim uvjetima. Osim toga isprobati ćemo algoritam na nekim teže prepoznatljivim znakovima. Tu odabiremo znakove koji su zbog kontrasta prilikom snimanja ili zbog lošeg osvjetljenja teže raspoznatljivi po kriteriju boje. Zatim biramo znakove koji se teže raspoznaju od svoje okoline, one koji su snimani pod nekim kutom i znakove koji su zamućeni zbog loše kvalitete snimke i kretanja automobila.

Zbog provjeravanje osjetljivosti algoritma na pogreške, isprobati ćemo ga i na nekim slikama koje sadrže objekte crvene boje.

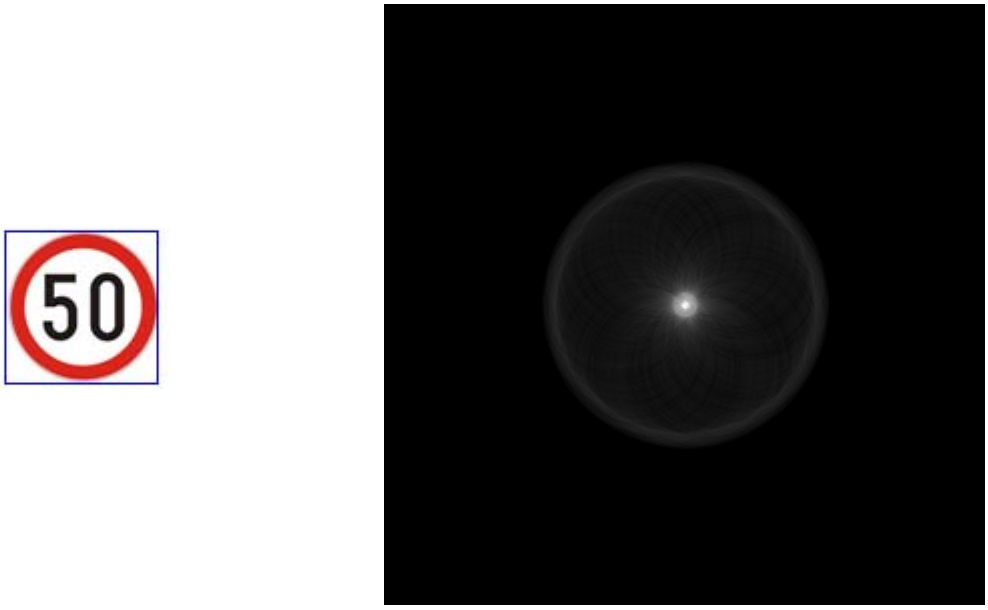
5.2. Postignuti rezultati

Uspješnost ovog algoritma u različitim uvjetima ovisi o mnogo podesivih parametara. Jedini parametar koji je podesiv u korisničkom sučelju je boja, to jest intervali nijanse, zasićenosti i intenziteta boje. Ako povećamo intervale parametara boje, algoritam će naći i znakove koji zbog različitih utjecaja nisu jasne crvene boje, ali će se tada pojaviti i više smetnji. Osim boje na rezultate utječe i zadani radijus kružnica koje tražimo. Ukoliko je na snimci ili seriji slika kut snimanja naspram ceste približno isti, u jednom bi trenutku svi znakovi trebali biti prepoznati, barem što se ovisnosti o radijusu tiče.

Sljedeći parametar o kojem ovise rezultati je osjetljivost na intenzitet slike prilikom izdvajanja središta kružnice u akumulacijskom polju (poglavlje 4.4). Ta osjetljivost zapravo označava koliki broj elemenata slike mora glasati za središte da bi ga mi izdvojili. U idealnim uvjetima ta bi se vrijednost trebala kretati oko $2*r*\pi$. Ako smanjimo broj piksela koji mora

glasati za središte, tada će algoritam biti manje osjetljiv na promjenu radijusa, no tada će također biti osjetljiviji na smetnje. Također u slučaju smanjenja broja potrebnih glasova, algoritam će moći detektirati znakove koji su mutni, imaju smetnje ili im zbog kriterija boje ne mogu glasati svi pikseli.

Minimalna veličina područja koju odlučimo smatrati znakom je kriterij kojim uglavnom smanjujemo da algoritam smetnje detektira kao znakove. Ona također ovisi o kvaliteti slike i veličini znaka, a njenim smanjenjem algoritam nalazi manje uočljive znakove, ali i ostale smetnje sličnih boja.



Slika 17: Detekcija znaka u idealnom slučaju

Postavke koje su korištene za idealan slučaj (slika 17):

hue: (5.95 - 0.05)

saturation: (0.09 - 1)

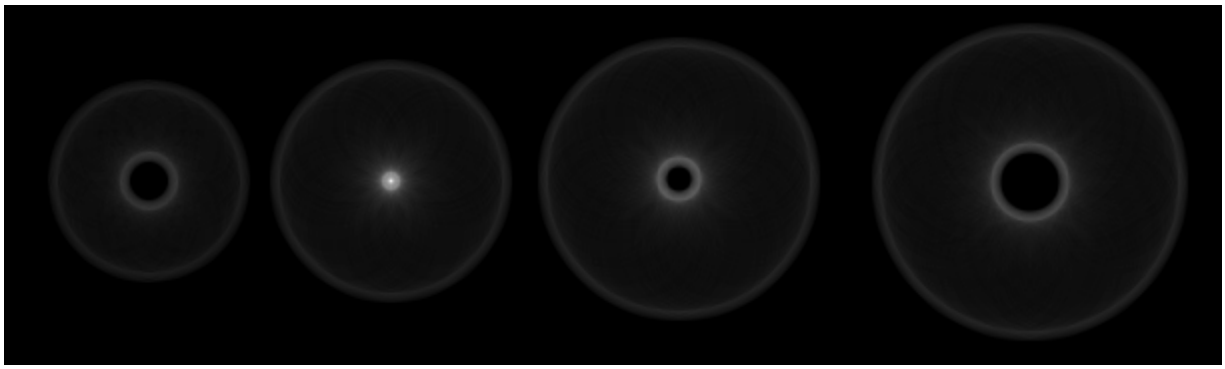
intensity: (0.15 - 0.833)

radius: 35

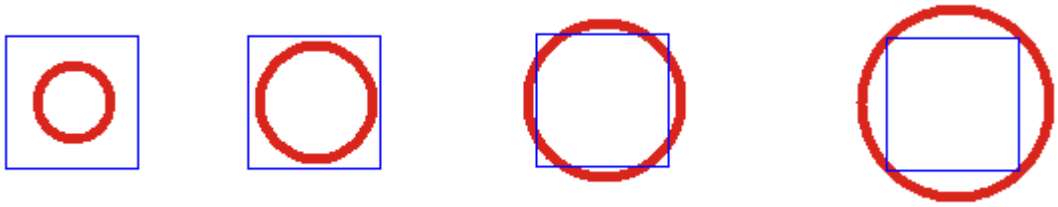
osjetljivost: 0.85*2*r*π

minimalna veličina prihvatljivih područja: 40

Vrijeme obrade jedne slike dimenzije 720 x 576 piksela kreće se od 0.1s do 0.5s, a u prosjeku iznosi 0.2s.



Slika 18: akumulacijsko polje slike 19



Slika 19: prikaz otpornosti algoritma na promjenu radijusa

Na slici 19 nalaze se kružnice radijusa približno 10, 20, 30 i 40 piksela. Sa zadanim radijusom od 30 piksela detektirane su sve četiri kružnice, no da bi se to postiglo potreban broj elementa slike koji glasuju za središte (osjetljivost) morao se smanjiti na samo 35 posto naspram idealnog slučaja.

Postavke koje su korištene na slici 19:
hue: (5.95 - 0.05)
saturation: (0.09 - 1)
intensity: (0.15 - 0.833)
radius: 30
osjetljivost: 0.35*2*r*π
minimalna veličina prihvatljivih područja: 15

5.2.1. Primjeri uspješne detekcije:

Slika 20: ispravno detektirani znak

Slika 21: ispravno detektirani znak

Slika 22: ispravno detektirani znak

Postavke koje su korištene na slikama 20, 21 i 22:

hue: (5.95 - 0.05)
saturation: (0.09 - 1)
intensity: (0.15 - 0.833)
radius: 32
osjetljivost: 0.7*2*r*π
minimalna veličina prihvatljivih područja: 25

Sa istim postavkama kao i slika 20 pronađene i slike 21 i 22 iako je na slici 21 boja znaka tamnija i kontrast je manji, dok na slici 22 znak okružuje crvenkasto lišće.
Iste postavke, s izuzetkom radiusa koji je smanjen na 25, bile su dovoljne za detekciju znakova na slikama 23 i 24. Navedene slike su izdvojene kao tipični predstavnici najvećeg skupa unutar baze slika znakova izričitih naredbi kojom raspolažemo.

Slika 23: ispravno detektirani znak

Slika 24: ispravno detektirani znak

Slika 25: ispravno detektirani znakovi

Slika 25 je mutna zbog kretanja automobila no boja znakova je dovoljno jasna i dovoljno se ističe od okoline. Također slika pokazuje mogući način izbjegavanja interferencije među znakovima (pojavljivanje nepostojećih znakova, slika 29) povećanjem broja elemenata koji moraju glasati za središte.

Postavke koje su korištene na slici 25:

hue: (5.95 - 0.05)
saturation: (0.09 - 1)
intensity: (0.15 - 0.833)
radius: 26
osjetljivost: 0.8*2*r*π (povećan da bi se spriječila interferencija znakova)
minimalna veličina prihvatljivih područja: 25

Slika 26: ispravno detektirani znak

Slika 26 je dobar primjer otpornosti ovog Hoghove transformacije za kružnice na zamućenje slike.

Postavke koje su korištene na slici 26:

hue: (5.95 - 0.05)
saturation: (0.09 - 1)
intensity: (0.15 - 0.833)
radius: 20
osjetljivost: 0.7*2*r*π
minimalna veličina prihvatljivih područja: 25

Slika 27: ispravno detektirani znak

Slika 27 je primjer slike u kojoj je teško detektirati samu boju znaka. Zbog toga je bilo potrebno povećati intervale parametara boje, smanjiti broj piksela koji moraju glasati da bi detektirali središte, i smanjiti minimalnu veličinu skupine elemenata koju smatramo središtem.

Postavke koje su korištene na slici 27:

hue: (5.2 - 0.1)
saturation: (0.01 - 1)
intensity: (0.15 - 0.833)
radius: 33
osjetljivost: 0.4*2*r*π
minimalna veličina prihvatljivih područja: 15

Slika 28: ispravno detektirani znak

Znak na slici 28 je također teže pronaći zbog loše kvalitete slike i dosta tamne boje samog znaka.

Postavke koje su korištene na slici 28:

hue: (5.2 - 0.1)
saturation: (0.01 - 1)
intensity: (0.15 - 0.833)
radius: 30
osjetljivost: 0.55*2*r*π
minimalna veličina prihvatljivih područja: 15

Na bazi slika znakova izričitih naredbi s crvenim obrubom kojom raspolažemo, na približno 90% slika koje sadrže znakove znakovi su uspješno detektirani.

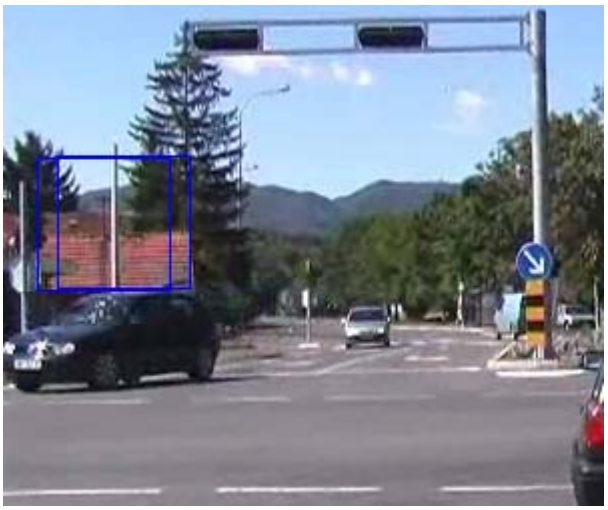
5.2.2. Primjeri neispravne detekcije (false positive):

Neispravna detekcija je zapravo nalaženje znakova tamo gdje ih nema. Do neispravne detekcije najčešće dolazi zbog smetnji poput automobila i ostalih prijevoznih sredstava, krovova kuća, reklama, ali i drugih prometnih znakova iste boje.

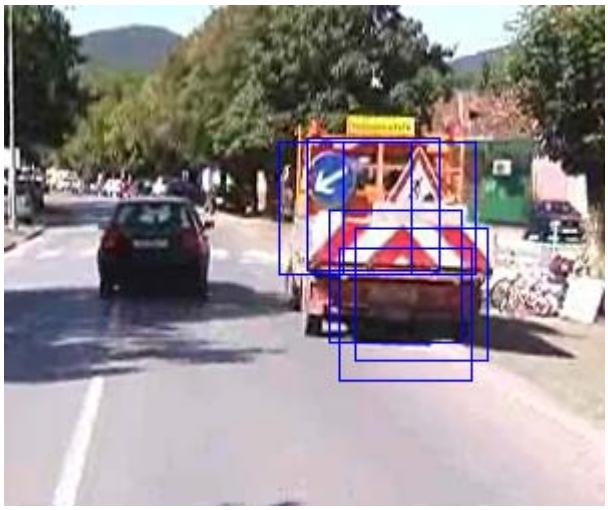
Slika 29: neispravna detekcija

Slika 30: neispravna detekcija

Na slici 29 vidimo da iako je okrugli znak uspješno detektiran, zbog glasanja elemenata drugoga znaka (koji je gotovo ispravno detektiran iako nije okrugao) pronađen i treći, nepostojeći znak. Na slici 30 vidimo da je osim ispravno detektiranog znaka, neispravno detektiran i automobil crvene boje.



Slika 31: neispravna detekcija



Slika 32: neispravna detekcija

Na slici 31 je detekciju nepostojećeg znaka izazvao krov kuće, dok je na slici 32 neispravno detektiran znak.

Na bazi slika znakova izričitih naredbi s crvenim obrubom kojom raspolažemo, na približno 40% slika znakovi su neispravno detektirani.

5.2.3. Primjeri propuštene detekcije (false negative):

S obzirom na veliku mogućnost prilagođavanja postavki slici, većinu znakova se uspješno može pronaći ovim algoritmom. To najčešće nije moguće samo onda kad se znak stopi s pozadinom (npr. zbog kontrasta, tamne slike ili crvenih pozadina) Primjer znaka koji nije pronađen ovim algoritmom je slika 33.

Slika 33: propuštena detekcija

Na bazi slika znakova izričitih naredbi s crvenim obrubom kojom raspolažemo, na približno 10% slika koje sadrže znakove znakovi nisu detektirani. Više od dvije trećine takvih slučajeva uzrokovano je velikom udaljenošću znaka na slici.

Zaključak

Houghova transformacija za kružnice je jednostavan algoritam koji je k tome prilično otporan na smetnje. Ne zahtjeva kompleksne izračune i ima kratko vrijeme izvođenja. Ovim se algoritmom uz prilagodbu postavki može pronaći veliki postotak znakova. Algoritam nije jako osjetljiv na zamućenja slike, na promjenu svijetlosti i kontrasta kao ni na blage promjene kuta snimanja. S druge strane ukoliko je znak stopljen s pozadinom, bilo zbog svjetlosti, kontrasta ili pozadine koja je boje znaka, algoritam ga neće moći precizno izdvojiti.

Iako veliki broj prilagodljivih postavki osigurava uspješnost ovog algoritma, to mu je ujedno i mana, budući da se one moraju ručno prilagođavati slici. Osim toga u ovom je algoritmu radijus znakova koji se traže fiksan, pa iako algoritam nije jako osjetljiv na manje promjene radijusa znakova, za veće mu se radijus mora ručno mijenjati.

Algoritam je također osjetljiv na ostale objekte istih nijansi crvene boje koji bi se mogli naći na slici poput krovova, reklama, automobila i drugih znakova. Također, ukoliko su znakovi postavljeni jedni iznad drugog, zbog međudjelovanja algoritam često pronađe još jedan ili više nepostojećih znakova među njima.

Houghova transformacija nije algoritam koji bi se samostalno mogao koristiti u nekim autonomnim sustavima, ali u kombinaciji s drugim algoritmima koji bi pokrili njegove nedostatke mogao bi dati dobre rezultate.

6. Literatura

[1] Stewart, C. V. Robust Parameter Estimation in Computer Vision: Robust Estimation Techniques Developed in Computer Vision, SIAM Review, svezak 41, broj 3, (1999), 520-521.

[2] Harvey Rhody, Hough Circle Transform, 11.10.2005., *Hough Circle Transform* www.cis.rit.edu/class/simg782/lectures/lecture_10/lec782_05_10.pdf, 27.9.2008.

[3] Robyn Owens, The Hough Transform, *29.10.1997., CVonline* http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT6/node3.html, 27.9.2008.

[4] Znakovi izričitih naredbi, 28.9.2008., *Prometni znakovi,* http://www.prometna-zona.com/prometni_znakovi-izricitih_naredbi.html, 03.10.2008.

[5] The RGB color space, 22.8.2008, *Blackice,* http://www.blackice.com/colorspaceHSI.htm, 20.10.2007.

[6] The HSI color space, 22.8.2008, *Blackice,* http://www.blackice.com/colorspaceHSI.htm, 20.10.2007.

[7] Conversion from RGB to HSI, 31. 10. 2007., *Color Processing*, http://fourier.eng.hmc.edu/e161/lectures/color_processing/node3.html, 23.9.2007.

[8] Flood fill, 13.12.2001., *Wikipedia, the free encyclopedia,* http://en.wikipedia.org/wiki/Flood_fill, 23.9.2007.

Pronalaženje prometnih znakova Houghovom transformacijom za kružnice

Sažetak

U ovome radu razmatramo pronalaženje okruglih znakova izričitih naredbi s crvenim obrubom u slikama snimljenima iz automobila u pokretu. Predstavljamo implementaciju metode za detekciju baziranu na Houghovoj transformaciji. U ovoj implementaciji obrube znakova pronalazimo kao kružnice fiksnog radijusa. Za izdvajanje elemenata slike koji zadovoljavaju uvjet boje obruba okruglih znakova izričitih naredbi, koristimo detekciju boje u prostoru HSI. Područja maksimalne vrijednosti uz akumulacijskog polja izdvajamo narastanjem područja. Implementacija je napisana u programskom jeziku C++. Uspješnost implementacije ispitana je i prikazana na stvarnim slikama

Ključne riječi

Houghova transformacija za kružnice, prometni znakovi, detekcija boje, narastanje područja, računalni vid, detekcija objekata

Detection of traffic signs using Hough transform for circles

Abstract

In this work we consider detecting circular traffic signs with a red frame in images taken from a moving vehicle. We present an implementation of a detection method based on Hough transform. In this implementation traffic signs are detected as circles with fixed radius. Red pixels corresponding to the traffic signs are detected using the HIS color model. The regions with maximal intensity in the accumulation array are extracted by the flood fill algorithm. The method has been implemented in the programming language C++. The efficiency of the developed system was evaluated and demonstrated on real images.

Key words

Hough transform for circles, traffic signs, region growing, color detection, computer vision, object detection