SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 811

PRONALAŽENJE PROMETNOG TRAKA U ODZIVU UPRAVLJIVOG FILTRA

Matija Gulić

Zagreb, lipanj 2009.

Sadržaj

Uvod 1
1. Pregled korištenih tehnika 2
1.1. Inverzna perspektivna transformacija 2
1.2. Obrada slike upravljivim filtrima 3
1.3. Houghova transformacija 4
1.3.1. Općenito o Houghovoj transformaciji 4
1.3.2. Houghova transformacija za pravce uz parametrizaciju (k, c) 5
1.3.3. Houghova transformacija za pravce uz parametrizaciju (ρ, θ) 6
1.3.4. Jednodimenzionalna Houghova transformacija za pravce 7
1.4. Odabrani pristup detekcije prometnog traka 7
2. Programska implementacija 9
2.1. Inverzna perspektivna transformacija 9
2.2. Primjena upravljivog filtra 10
2.3. Detekcija dominantne orijentacije prometnog traka 11
2.3.1. Binarizacija odziva upravljivog filtra 12
2.3.2. Histogram orijentacije upravljivog filtra 12
2.4. Detekcija rubnih linija jednodimenzionalnom Houghovom transformacijom 13
3. Eksperimentalni rezultati 16
3.1. Postavljanje parametara naredbenog retka ljuske 16
3.2. Utjecaj odabira parametara na rezultate postupka 16
3.3. Ilustracija koraka obrade 19
3.4. Procjena ispravnosti postupka 23
3.5. Brzina izvođenja razvijenog postupka 24
Zaključak 25
Literatura 26

Uvod

Povećanjem broja motornih vozila na cestama došlo je i do pojačanog razvoja kako aktivne tako i pasivne sigurnosti. Pod pojmom aktivna sigurnost smatramo sve ono što može spriječiti nesreće (bilo da one izvana nepredvidljivo djeluju na vozača ili ih on sam uzrokuje - nepažnjom). Primjeri sustava aktivne sigurnosti su: ABS, senzor za kišu i automatska regulacija razmaka. Također u takve sustave spada i sustav za upozoravanje prilikom nenamjernog prelaska u drugi trak. Ovakvi sustavi bi u budućnosti uz konstantni razvoj vozila i prometnica mogli značajno doprinijeti sigurnosti na cestama. Međutim, preduvjet za ostvarivanje takve funkcionalnosti su postupci za detekciju prometnog traka temeljeni na računalnom vidu.

Kako bi bilo moguće detektirati prometni trak potrebno je prepoznati linije koje omeđuju taj trak. Za čovjeka je takav posao vrlo jednostavan dok je za računalo vrlo složen. Računalne metode detekcije uglavnom se baziraju na kontrastu signalizacije i podloge. Upravo taj kontrast u lošim uvjetima (sjena, neodržavana signalizacija) dovodi do loših rezultata. Zbog toga prije same detekcije prometnog traka potrebno je na slici ukloniti šum, te naglasiti tražene informacije filtriranjem, detekcijom rubova ili nekim drugim metodama. Na tako predobrađenim slikama, granicu prometnog traka pronalazimo kao povezani slijed piksela koji odgovaraju ili rubovima ili elementima linije prometnog traka.

U ovom radu razmatramo postupak za pronalaženje linija koje ograničavaju prometni trak. Prvo provodimo inverznu perspektivnu transformaciju, zatim koristimo upravljivi filtar. Nakon toga određujemo dominantnu orijentaciju i pomoću Houghove 1D transformacije detektiramo prometne linije.

Rad je strukturiran kako slijedi. U 1. poglavlju opisane su korištene tehnike. Nakon toga u 2.poglavlju slijedi programska implementacija (opis bitnih dijelova programa) i u 3.poglavlju su eksperimentalni rezultati te subjektivna procjena dobivenih rezultata.

1. Pregled korištenih tehnika

1.1. Inverzna perspektivna transformacija

Metoda inverzne perspektivne transformacije je primjerena kao prvi korak predobrade pri detekciji prometnih linija, uz pretpostavku da je cesta lokalno planarna [1]. Ova metoda omogućava transformaciju izvornih slika snimljenih iz perspektive vozača u „inverznu perspektivnu“ sliku, kakva bi bila snimljena kamerom čiji je smjer gledanja paralelan s normalom ravnine ceste. Inverzne perspektivne slike su korisne zbog toga što prometne linije postaju paralelne linije s konstantnom širinom, kao što je prikazano na slici 1.

Slika 1. Izvorna slika iz perspektive vozača (lijevo) i odgovarajuća inverzno perspektivna slika (desno)

Postojanje inverzne perspektivne transformacije je jednostavno dokazati. Označimo s q_{Ci} neku točku na cesti, na perspektivnoj slici odgovarajuću točku sa q_{Pi} i na inverzno perspektivnoj slici sa q_{Ii}. Dobro je poznato da se odnos točke ravnine i njihove perspektivne slike može opisati bijektivnim preslikavanjem koje se naziva homografija [1]. Ukoliko točke izrazimo u homogenim koordinatama [2] homografija se može prikazati linearnim preslikavanjem kao što slijedi:

$$q_{Pi} = H_{CP} \cdot q_{Ci}, \forall i \qquad (1)$$

$$q_{Ii} = H_{CI} \cdot q_{Ci}, \forall i \qquad (2)$$

Kombiniranjem (1) i (2) inverznu perspektivnu transformaciju možemo prikazati kao kompoziciju H_{CP}^{-1} i H_{CI} :

$$q_{Ii} = H_{IPT} \cdot q_{Ci}, \forall i \qquad (3)$$

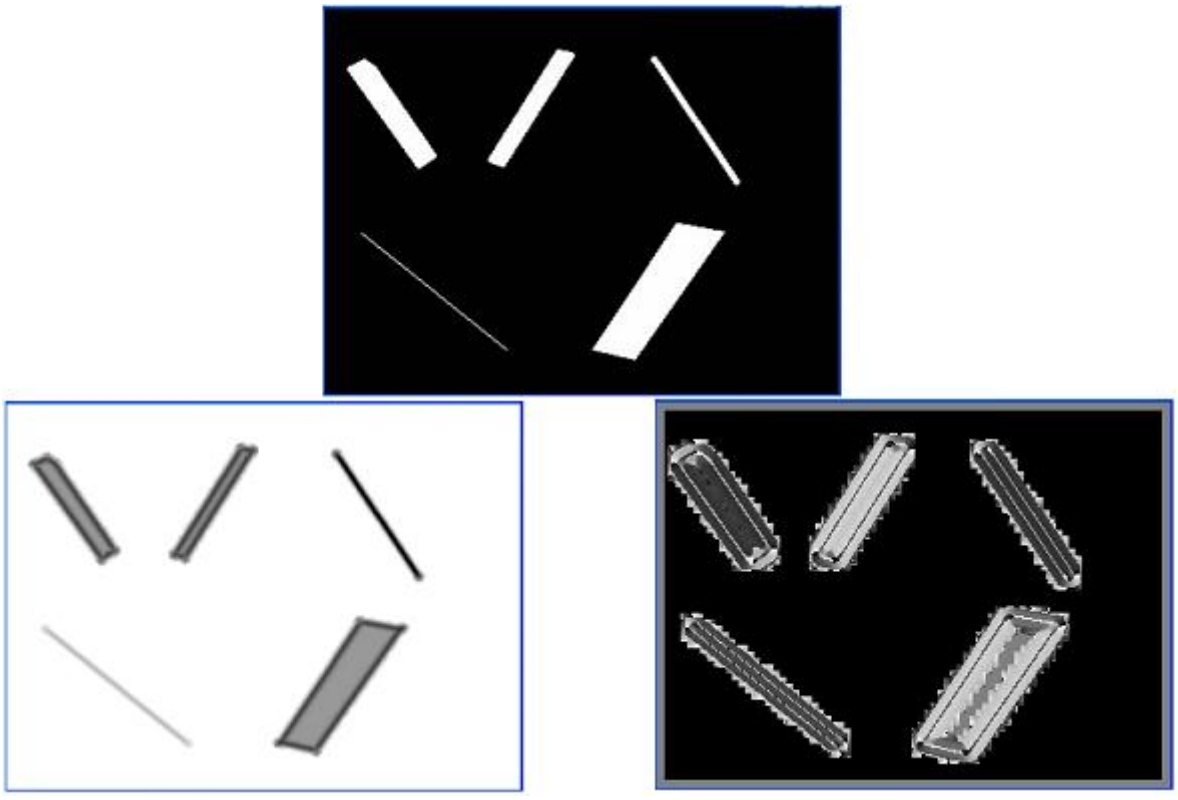
$$H_{IPT} = H_{CI} \cdot H_{CP}^{-1} \qquad (4)$$

Parametri matrice H_{IPT} odgovaraju geometriji kamere s obzirom na ravninu ceste. Ovi parametri mogu biti kalibrirani prije upotrebe odgovarajućim postupkom kalibracije [2]. Kalibracija inverzne perspektivne transformacije je zahtjevan postupak koji neće biti detaljnije razmatran u ovom radu.

1.2. Obrada slike upravljivim filtrima

Upravljivi filtri se mogu koristiti za detekciju linearnih struktura na slici. U ovom radu korišten je upravljivi filtar koji se temelji na drugoj derivaciji Gaussove funkcije [3].

Glavno svojstvo upravljivog filtra je da se odziv sastoji od intenziteta i orijentacije. Korišteni upravljivi filtar daje veliki odziv na izduženim i kontrastnim strukturama na slici, kao što je prikazano na slici 2.



Slika 2. Ulazna slika (gore), odziv intenziteta (lijevo) i odziv orijentacije (desno) upravljivog filtra

Najveći odziv intenziteta na izlaznoj slici dogodio se kod područja crno-bijelo-crno [11]. Takva promjena rezultira crnom bojom na mjestima gdje se na ulaznoj slici nalaze spomenute promjene. Na ostatku slike nema promjena intenziteta, zbog toga je ostatak slike bijele boje (odziv je jednak nuli). Orijentacija piksela ovisi o nagibu pojedine strukture sa ulazne slike.

Kod upravljivog filtra bitan je odabir „debljine“ filtra, odnosno parametar σ. Prilikom izračuna odziva filtra ta vrijednost je fiksna. Upravo zbog toga smo ulaznu sliku inverzno perspektivno transformirali kako bi debljine prometnih linija bile fiksne. Više o odabiru vrijednosti parametra σ kao i primjeri s različitim vrijednostima istog parametra u potpoglavlju 3.2.

1.3. Houghova transformacija

Houghova transformacija je robusna tehnika procjenjivanja parametara temeljena na načelu glasanja. Ovom transformacijom pronalaze se parametri nesavršenih primjeraka zadane klase oblika. Isprva je korištena za detekciju ravnih linija, a kasnije je proširena i na neke kompleksnije parametarske oblike poput kružnica i elipsa [4].

Primjerci klase oblika definiraju se n-torkom parametara. Svaki element slike (piksel) glasuje za parametre svih oblika kojima bi mogao pripadati. Glasovi se zbrajaju u akumulacijskom polju čija dimenzionalnost ovisi o broju parametara koji definiraju oblik. Traženi oblici se pronalaze kao maksimumi tog polja.

1.3.1. Općenito o Houghovoj transformaciji

Houghova transformacija sastoji se od tri osnovna koraka.

Prvi korak je detekcija primitivnih elemenata slike. Svaki element slike ne sadrži jednako vrijedne vrijednosti. Zbog toga prije postupka glasanja trebamo provesti razna filtriranja kako bismo naglasili one informacije koje nam trebaju, odnosno prigušili one informacije koje nam smetaju. Preciznije, izdvajamo elemente koji su prošli određenu selekciju.

Drugi korak je preslikavanje slike u parametarski prostor. Svaki element slike koji je prošao selekciju iz prvog koraka preslikava se u parametarski prostor definiran akumulacijskim poljem. Ovisno o tome koje oblike detektiramo određena je n-

dimenzionalnost akumulacijskog polja. Ako se radi o pravcu tada je akumulacijsko polje 2-dimenzionalno, dok je kod elipse akumulacijsko polje 5-dimenzionalno. Preslikavanja elemenata u parametarski prostor ostvaruje se glasovanjem, odnosno povećanjem vrijednosti u akumulacijskom polju čije su koordinate jednake paru parametara traženog oblika.

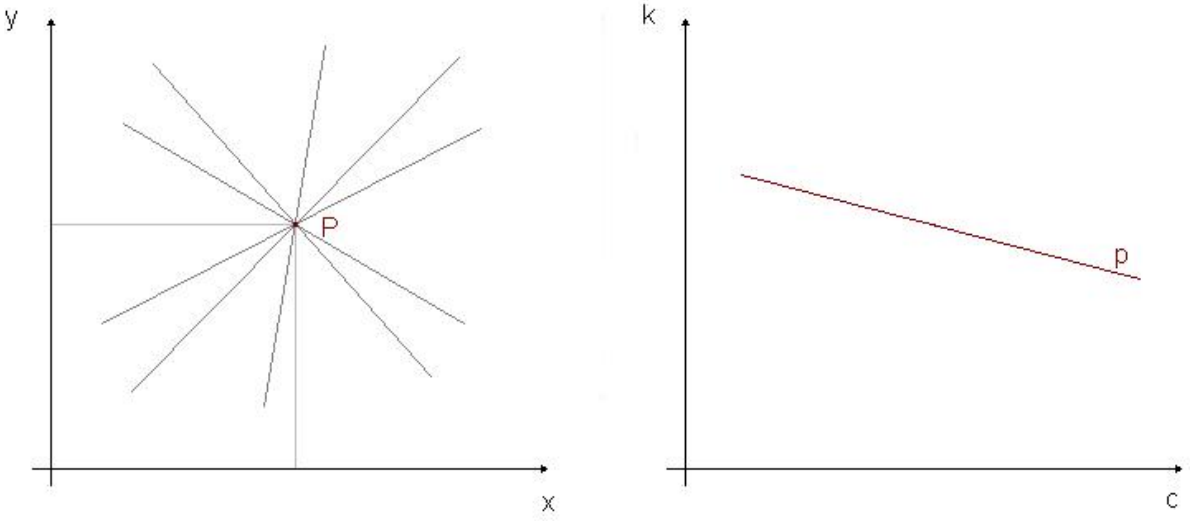
Treći korak je detekcija lokalnih maksimuma. Rezultat dobiven drugim korakom je akumulacijsko polje u kojemu se lokalni maksimumi nalaze na koordinatama koje predstavljaju parametre traženog oblika. Jedan od načina izdvajanja maksimuma je izdvajanje svih vrijednosti akumulacijskog polja koje prelaze neku graničnu vrijednost.

1.3.2. Houghova transformacija za pravce uz parametrizaciju (k, c)

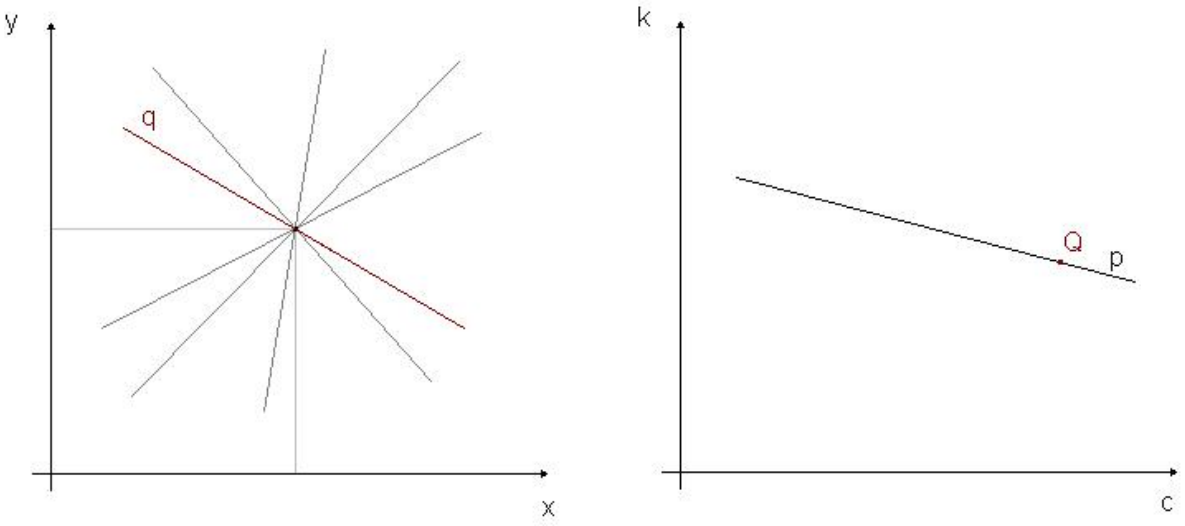
Pravac možemo prikazati parametrima nagiba i odsječka na osi y (k, c). Tada pravcu pripadaju točke koje zadovoljavaju slijedeću jednakost:

$$y = kx + c \qquad (5)$$

Neka točka P u prostoru *x*, *y* može pripadati svakom od skupa pravaca koji prolaze kroz nju. U prostoru *k*, *c* po jednakosti (5), ta točka definira pravac *p*. Točka Q toga pravca u *k*, *c* prostoru definira točno jedan pravac *q* od skupa pravaca koji prolaze kroz točku P u prostoru *x*, *y*.



Slika 3. Odnos točke u prostoru O(x, y) i pravca u prostoru O(k, c) (preneseno iz [4])



Slika 4.Odnos pravca u prostoru O(x, y) i točke u prostoru O(k, c) (preneseno iz [4])

Koordinate točke u prostoru *x*, *y* predstavljaju vrijednosti parametara pravca u prostoru *k*, *c* (slika 3). Analogno tome, koordinate točke u prostoru *k*, *c* određuju pravac u prostoru *x*, *y* (slika 4), stoga se ova metoda može smatrati transformacijom između prostora *x*, *y* i *k*, *c*.

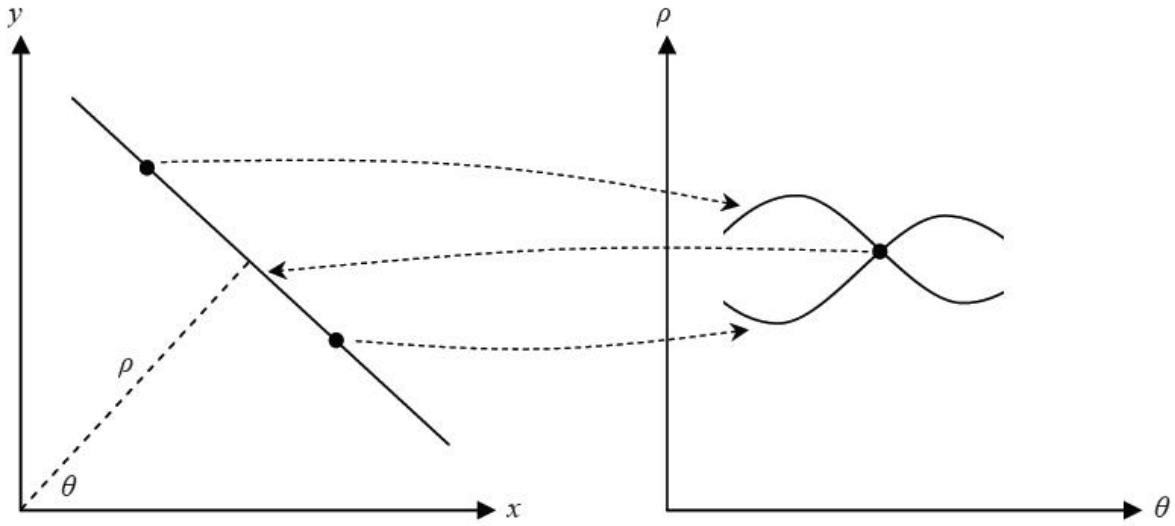
Svaki element slike (piksel) na traženom pravcu *q* u parametarskom prostoru iscrtava pripadajući pravac, tj. glasuje za njega. Time se na sjecištu Q tih pravaca ostvaruje lokalni maksimum. Parametri koji definiraju traženi pravac na slici su koordinate lokalnog maksimuma u parametarskom prostoru.

1.3.3. Houghova transformacija za pravce uz parametrizaciju (ρ,θ)

Budući da jednakost (5) ima beskonačni nagib za vertikalni pravac, u praksi se češće koristi alternativna parametrizacija (ρ , θ):

$$p \equiv \rho = x\cos\theta + y\sin\theta \qquad (6)$$

U jednadžbi (6) ρ je udaljenost pravca od ishodišta, a θ nagib normale pravca s obzirom na x os. Opisana jednadžba pridružuje vrijednosti prostora *x*, *y* vrijednostima parametarskog prostora ρ , θ .



Slika 5. Prikaz parametara pravca ρ i θ i povezanost pravca s parametarskim prostorom

Za razliku od jednadžbe (5), elementi slike u parametarskom prostoru ρ, θ definiraju sinusoidne krivulje, a ne pravce. Pravac u prostoru *x*, *y* pridružen je sjecištu sinusoidnih krivulja u parametarskom prostoru.

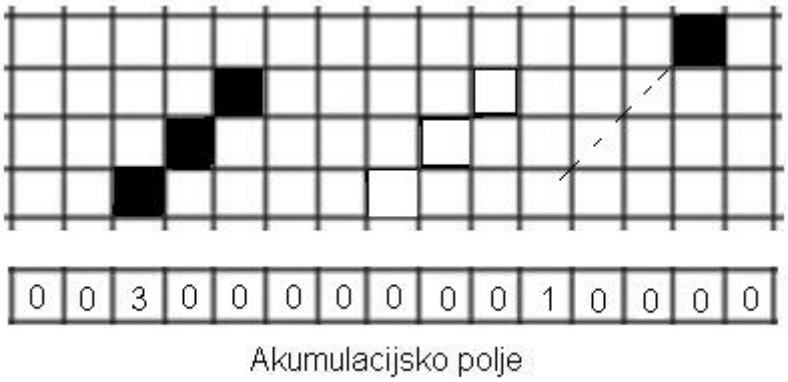
1.3.4. Jednodimenzionalna Houghova transformacija za pravce

Važno je spomenuti da ponekad možemo smanjiti prostor parametara nekog oblika. To radimo na način da unaprijed izračunamo neki od parametara, te taj parametar fiksiramo. U slučaju pravca fiksiranjem jednog parametra akumulacijsko polje postaje jednodimenzionalno. Fiksirani parametar može biti orijentacija pravca ukoliko tu orijentaciju možemo odrediti nekom drugo metodom.

1.4. Odabrani pristup detekcije prometnog traka

Naš postupak detekcije prometnog traka počinje inverznom perspektivnom transformacijom, a zatim tako transformiranu ulaznu sliku obrađujemo upravljivim filtrom. Nakon toga radimo binarizaciju odziva upravljivog filtra, i računamo dominantnu orijentaciju. Slijedeći korak je popunjavanje akumulacijskog polja.

Kod detekcije pravaca parametarski prostor se može reducirati na jednodimenzionalan korištenjem nagiba (θ) normale pravca s obzirom na os x. Pokažimo na jednostavnom primjeru kako izgleda popunjavanje akumulacijskog polja (slika 6).



Slika 6. Prikaz popunjavanja akumulacijskog polja

Pikseli slike koji glasaju prikazani su kao kvadratići popunjeni crnom bojom. Akumulacijsko polje je jednodimenzionalno polje čije su početne vrijednosti za sve elemente postavljene u nulu. Za svaki „crni“ piksel (x_1, y_1) računamo apscisu x_0 za koju je ordinata jednaka nuli. Vrijednost apscise se računa prema jednadžbi:

$$x_0 = x_1 + y_1 \cdot tg\theta \qquad (7)$$

U jednadžbi (7) x_0 je tražena apscisa, (x_1, y_1) su koordinate trenutnog piksela, θ je poznati nagib normale pravca s obzirom na x os. Pokažimo da jednakost (7) vrijedi. Iz jednadžbe (6) slijedi ako su dvije točke na istom pravcu mora vrijediti:

$$x_0 \cos\theta + y_0 \sin\theta = x_1 \cos\theta + y_1 \sin\theta \quad (8)$$

Vrijednost y_0 postavljamo u 0, po pretpostavci. Sada podijelimo cijelu jednadžbu (8) sa $\cos\theta$ i time dobivamo jednadžbu (7).

Nakon određivanja x_0, relevantni element akumulacijskog polja ($\lfloor x_0 \rfloor$) se uvećava za 1.

Nakon što je završio postupak glasanja, tražimo onaj indeks za koji element polja ima najveću vrijednost. Na temelju tog indeksa može se detektirati pravac (prometna linija). Poznat je nagib pravca i poznata je jedna točka pravca (koordinata x točke je indeks elementa s najvećom vrijednosti, dok je koordinata y jednaka nuli).

2. Programska implementacija

Eksperimentalni dio završnog rada rađen je u programskom jeziku C++. Prilikom izrade rada korištena je programska ljuska cvsh [12]. Ljuska učitava slijed slika, na svaku ulaznu sliku primjenjuje odabrani postupak te prikazuje rezultate obrade. Svojstvo slika je da su definirane kao jednodimenzionalno polje. Svaki piksel slike je najčešće određen sa tri vrijednosti. Navedene vrijednosti su R (crveno) , G (zeleno) i B (plavo) [6]. Svaka od ovih vrijednosti predstavlja zastupljenost određene boje na određenom pikselu. Pomoću kombinacije ove tri boje mogu se dobiti sve ostale boje.

Korisnički algoritam se poziva preko metode `process` korisničkog razreda `alg_laneSteer`:

```
void alg_laneSteer::process (
      const img_vectorAbstract& imgs,
      const win_event_vectorAbstract& events,
      int msDayUtc, int frame)
```

2.1. Inverzna perspektivna transformacija

Kao prvi korak obrade pozivamo metodu `process` korisničkog razreda `ext_ipt`:

```
void ext_ipt::process (
      const img_wrap& imgSrc,
      img_wrap& imgIPT)
```

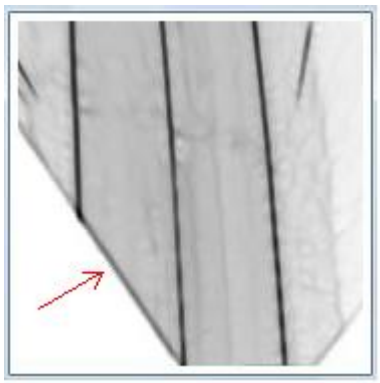
Ovoj metodi predajemo ulaznu sliku, te zatim ona provodi inverznu perspektivnu transformaciju [7] nad ulaznom slikom i vraća sivu sliku [8]. Ova transformacija nam je vrlo važna zato jer prometne linije nakon inverzne perspektivne transformacije postaju konstante debljine (bitno za upravljivi filtar). Također rezultat transformacije je taj da linije postaju paralelne što nije slučaj u ulaznoj slici. Ova metoda ima i nekih negativnih posljedica, a to su „trokuti“ o kojima će biti riječi u potpoglavlju 2.2.

2.2. Primjena upravljivog filtra

Slijedeća metoda koja se poziva je metoda `process` korisničkog razreda `ext_steer [3]`:

```
int ext_steer::process (
      const img_wrap& imgSrc,
      double sigma)
```

Ova metoda prima ulaznu sliku i parametar `sigma` pomoću kojega će se vršiti filtriranje temeljeno na drugoj derivaciji Gaussove funkcije [9]. Ovoj metodi šaljemo sliku koju smo dobili nakon inverzne perspektivne transformacije. Nakon obrade postojat će nekoliko novih slika od kojih su dvije ključne za daljnje korake. Vratimo se sada na problem „trokuta“ koji je prošao kroz ovaj filtar. Pogledajmo na slici 7 kako izgledaju trokuti nakon ovog filtriranja:

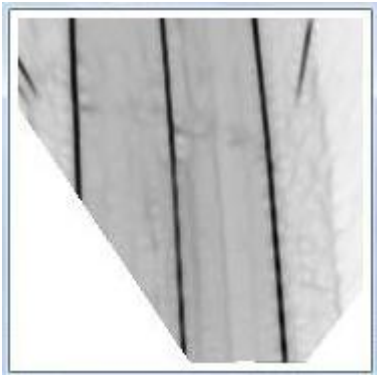


Slika 7. Problematični trokut

Crvenom strelicom označena je stranica trokuta koja će u daljnjoj obradi slike pridonositi pogrešnoj detekciji, te je zbog toga moramo maknuti. Prolazimo svim pikselima slike i kada naiđemo na prvi piksel kojem vrijednost nije jednaka 0, odnosno bijela boja, tada sve piksele koji se trenutnom pikselu nalaze bliže od polovine dimenzije upravljivog filtra [3] mijenjamo u bijelu boju i nastavljamo dalje na idući red. Opisani postupak smješten je u funkciji `triangles`:

```
void triangles(
       const img_wrap& imgIPT,
       const img_wrap& imgSteerResponse,
       const img_wrap& imgSteerOrientation,
       const int margin)
```

Sa priložene slike (Slika 7) vidi se da i u donjem desnom kutu postoji trokut, problem ovog trokuta rješavamo na isti način kao i za veliki trokut u donjem lijevom kutu. Na Slika 8 vide se rezultati nakon micanja trokuta na donjem lijevom i donjem desnom dijelu slike.



Slika 8 . Rezultat micanja trokuta

Spomenuti pristup za rješavanje problema trokuta pretpostavlja da valjani pikseli slike nisu crni, a to ne mora biti slučaj. Ovaj problem možemo riješiti na način da inverznom perspektivnom transformacijom „napadnemo“ sliku koja se sastoji od samih bijelih piksela. Time bi rezultirajuća slika („maska“) bila crna samo u nevidljivim pikselima. Međutim ako ovaj pristup primijenimo pri obradi svake slike obrada će se nezanemarivo usporiti.

Sada kad je riješen problem trokuta pozivamo funkciju `equalize`. Definicija ove funkcije:

```
void equalize(
      const img_wrap& src,
      img_wrap& dst)
```

Ovu funkciju koristimo iz razloga što su slike dobivene dosadašnjim filtriranjem popunjene podacima tipa double, te su ove vrijednosti veće ili manje od 0 odnosno 255 što su vrijednosti za sivu sliku. Ova funkcija skalira dobivene vrijednosti na željeni raspon od 0 do 255.

2.3. Detekcija dominantne orijentacije prometnog traka

Kao što je navedeno u 1.4. selekcija za odabir dominantne orijentacije se sastoji od odabira piksela koji se nalaze u binarnoj slici [11], i zatim ispitivanja orijentacije tih piksela [7].

2.3.1. Binarizacija odziva upravljivog filtra

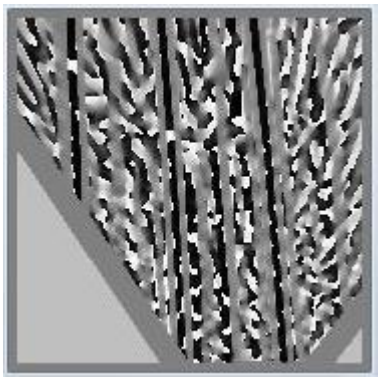
Binarizacija slike je postupak kojim u slici potiskujemo smetnje, te naglašavamo one piksele koji su nam važniji. Za svaki piksel ulazne sive slike postavljamo vrijednost na minimalnu ili maksimalnu vrijednosti ovisno o tome da li je trenutna vrijednost iznad ili ispod zadanog praga. Dobar rezultat binarizacije uvelike ovisi o odabiru praga. Ako prag postavimo previsoko tada će rezultat imati premalo piksela i binarna slika neće biti dobra. Ako prag postavimo prenisko tada će binarizaciju proći previše piksela, i samim time i više smetnji koje ovim postupkom želimo riješiti. Više o postavkama praga i primjeri slika s različiti pragom u potpoglavlju 3.2.

U funkciji `binarize` sadržan je opisani postupak:

```
void binarize(
       const img_wrap& imgSteerResponseEq,
       int prag,
       int margin,
       img_wrap& imgBin)
```

2.3.2. Histogram orijentacije upravljivog filtra

Filtriranje upravljivim filtrom [3] uz već spomenutu sliku odziva (Slika 8) kreira i sliku na kojoj se nalaze orijentacije svakog piksela. Upravo ova slika omogućit će nam pronalazak dominantnog smjera. Pogledajmo kako izgleda slika orijentacija:



Slika 9 . Orijentacija piksela

Vrijednosti elemenata sa slike kreću se od -1,57 (-π/2) do 1,57 (π/2). Kako bi lakše kreirali histogram vrijednosti se pretvaraju u stupnjeve, i još im se dodaje 90 kako bi u histogramu jednostavnije pristupali elementima vektora. Dakle da bi kreirali histogram potrebno je prolaziti po svim pikselima slike, i povećati brojač vrijednosti kojoj trenutno

pristupamo. Na taj način dobit ćemo broj pojavljivanja svakog kuta na sa slike. Pogledajmo sada kako izgleda histogram orijentacije upravljivog filtra za dijelove slike 9 u kojima je iznos odziva upravljivog filtra veći od zadanog praga.

Slika 10. Histogram orijentacije

Iz histograma možemo pročitati da ima najviše piksela čija orijentacija je 90°, odnosno -90°. Ovo je očekivani rezultat zato jer tražimo linije koje su na slikama vertikalne - prometne linije. Kako bi dobili dominantni smjer koji nam treba potrebno je provjeriti sve kutove, te među njima naći onaj koji u histogramu ima najveću vrijednost. Upravo taj kut je dominantni smjer (orijentacija).

Postupak opisan u ovom odjeljku nalazi se u funkciji `histOrientation` :

```
double histOrientation(
       const img_wrap& imgSteerOrientation,
       const img_wrap& imgBin,
       const int margin,
       img_wrap& imgHistOrientation)
```

Vezano za navedenu funkciju važno je spomenuti da funkcija vraća vrijednost dominantne orijentacije.

2.4. Detekcija rubnih linija jednodimenzionalnom Houghovom transformacijom

Sada imamo binarnu sliku i dominantni smjer. Upravo to nam je bitno za određivanje akumulacijskog polja kojeg koristimo za jednodimenzionalnu Houghovu transformaciju. Postupak je vrlo sličan kao kod histograma. Samo što sada nemamo samo jednu informaciju pomoću koje povećavamo vrijednost određenog indeksa polja, nego više informacija.

Postupak je opisan u poglavlju 1.4. Pokažimo sada pomoću pseudokoda kako je taj postupak implementiran:

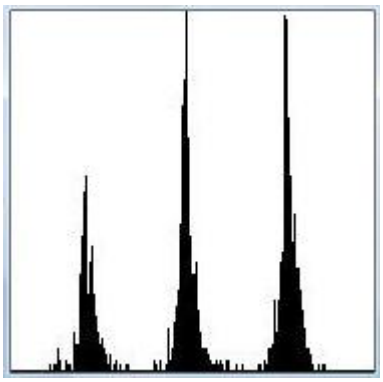
```
za x=1 do brojStupaca
   za y=1 do brojRedaka
      ako binarnaSlika(x,y)==255
         ako (orijentacijaSlika(x,y) >= dominantniSmjer -odstupanje &
              orijentacijaSlika(x,y) <= dominantniSmjer +odstupanje)
                  i = x + y*tan(-pi/2+orijentacijaSlika(x,y));
                  akPolje(i)++;
                  novaSlika(x,y) = 255;
         kraj
      kraj
   kraj
kraj
```

Pojasnimo sada ovaj pseudokod. Prolazimo svim pikselima slike, i prvo provjeravamo da li trenutni piksel ima vrijednost 255 u binarnoj slici. Na taj način riješili smo neke od smetnji koje bi se pojavljivale da radimo sa sivom slikom. Zatim provjeravamo da li je orijentacija određenog piksela približno jednaka dominantnoj orijentaciji. Na ovaj način riješili smo se još jednog dijela smetnji (spomenute smetnje su svi pikseli osim onih koji se nalaze na prometnoj liniji). Sada u novu sliku postavljamo vrijednosti piksela na 255 (odnosno bijelu boju). Ova slika nam služi za ilustraciju kako bismo provjerili koliko smetnji je uklonjeno usporedbom s dominantnim smjerom. I sada računamo indeks akumulacijskog polja po slijedećoj jednadžbi:

$$i_A = x_1 + y_1 \cdot tg\theta \qquad (7)$$

Na taj način dobiva se indeks elementa akumulacijskog polja čiju vrijednost trebamo povećati za jedan. Izvod formule opisan je u 1.4.

Sada u akumulacijskom polju imamo postavljene vrijednosti za svaki indeks, i prikladno bi bilo grafički to prikazati.



Slika 11 . Grafički prikaz akumulacijskog polja

Iz Slika 11 možemo rekonstruirati položaj linija na slici. Ovaj postupak izvodi se na način da pronađemo lokalne maksimume, u ovom konkretnom primjeru ima 3 lokalna maksimuma. Zatim odredimo indekse u akumulacijskom polju u kojem se nalaze lokalni maksimumi. Od prije je poznata vrijednost dominantnog smjera, te još dodatno uzimamo dvije točke pravca. Odaberimo prvu točku na način da joj je x koordinata jednaka indeksu akumulacijskom polja na kojemu se nalazi lokalni maksimum, y koordinata ove točke je 0: $T_1 = (x_1, 0)$. Za drugu točku odaberimo koordinatu y takvu da bude jednaka visini h prozora u kojem prikazujemo slike. x koordinatu druge točke izračunamo po formuli:

$$x_2 = x_1 + h \cdot tg\theta \qquad (9)$$

Sada imamo poznate dvije točke kojima prolazi prometna linija, te ih možemo lako anotacijski označiti. Ovime smo prepoznali prometnu liniju.

Postupak opisan u ovom potpoglavlju nalazi se u funkciji `hough1D`:

```
void hough1D(
       const img_wrap& imgBin,
       const img_wrap& imgSteerOrientation,
       const int dominantna_orijentacija,
       const int odstupanje,
       const int margin,
       img_wrap& imgAkPolje,
       img_wrap& imgHough,
       std::vector<int>& lines)
```

3. Eksperimentalni rezultati

Kako bismo mogli ocijeniti uspješnost prepoznavanja prometnog traka potrebno je testirati algoritam. Kako nismo omogućili automatsko testiranje, potrebno je ručno gledati rezultate za svaku sliku te subjektivno procijeniti točnost algoritma.

3.1. Postavljanje parametara naredbenog retka ljuske

Za ispravan rad s ljuskom potrebno je prvo ručno postaviti određene postavke [12]. Lijevim klikom miša treba izabrati Project , te zatim zavrad (ime projekta) Properties.

Za upisivanje naredbenog retka u razvojno okruženje potrebno se pozicionirati u Configuration properties →Debugging →Command Arguments

Primjer kako treba izgledati naredba u naredbenom retku:

-sf=C:/desktop/slike_i_filmovi/V-24_9_2007-16_22_58-D5-A.wmv -i="p a 200" -a=laneSteer -c="2 90 7 (-533,0)(605,0)(309,296) 5 20 fixed:ipv-panasonic-720"

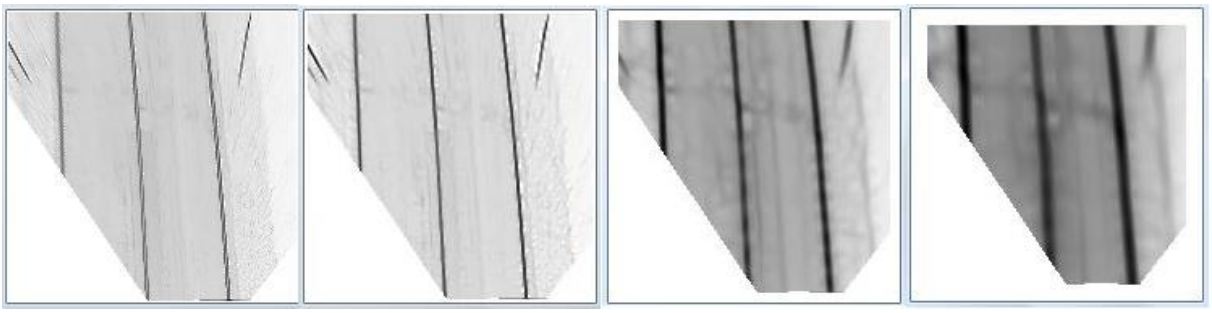
Prvo se navodi putanja do izvorne datoteke slijeda slika (-sf), zatim se navodi prva naredba koja se prosljeđuje korisničkom sučelju (-i), te nakon toga naziv algoritma koji se koristi (-a), i na kraju konfiguracija parametara kamere pomoću koje je snimljen ulazni slijed slika (-c). Naziv algoritma povezuje se sa klasom alg_base, tada se pomoću metode alg_base::create stvara instanca pripadajućeg algoritma kojeg ljuska dalje koristi.

3.2. Utjecaj odabira parametara na rezultate postupka

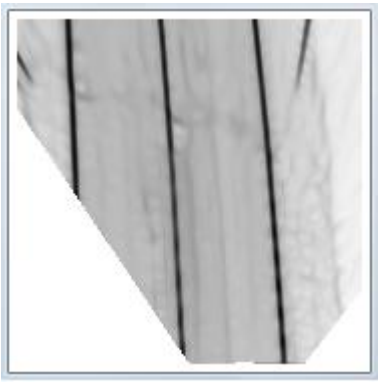
Prilikom izvođenja algoritma vrlo je važno odabrati dobre parametre kako bi dobili što bolje rezultate (odnosno kako bi pronalaženje prometne trake bilo što preciznije). Kako bi olakšali eksperimentiranje, omogućeno je dinamičko konfiguriranje tri parametra (`sigma, thBin, epsTheta`).

Izbor parametra `sigma` bitan je za upravljivi filtar [3]. Upravo pomoću ovog parametra određujemo širinu prometne linije. Zbog toga nam je važna inverzna perspektivna

transformacija, jer nakon te transformacije linije postaju konstantno široke. Pokažimo na nekoliko primjera (slika 12) utjecaj izbora parametra `sigma`.



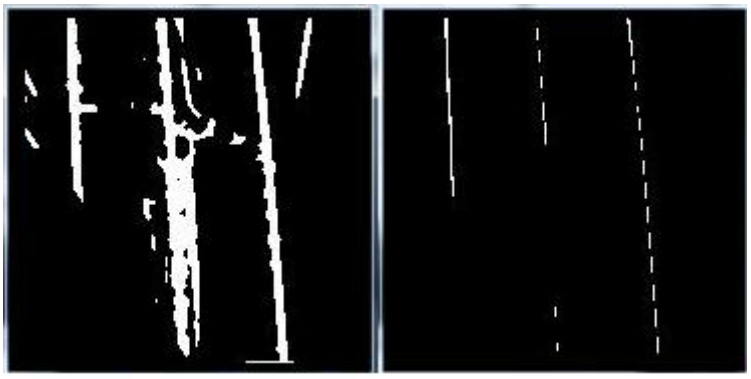
Slika 12. Prikaz važnosti odabira parametra `sigma` (s lijeva na desno σ=0.5 , σ=1 , σ=3 , σ=5)



Slika 13. Najprikladniji odabir parametra `sigma` (σ=2)

Smanjivanjem vrijednosti parametra `sigma` filtriranje ne daje tražene rezultate te pronalaženje prometnog traka postaje otežano. Povećanjem vrijednosti `sigme` odziv filtra postaje sve mutniji, što također nije dobro za daljnju obradu. Parametar `sigma` postavljen u vrijednost 2 daje najbolje rezultate. Ovdje je još važno spomenuti da vrijeme izvođenja programa uvelike ovisi o parametru `sigma`. Kada parametar `sigma` iznosi 2 tada vrijeme obrade upravljivog filtra traje 120 ms, peterostrukim povećanjem parametra (na 10) vrijeme izvođenja se deset puta poveća (na 1.3 s). Smanjenjem parametra `sigma` na 0.5 vrijeme izvođenja se smanji na 35 ms. Više o trajanjima faza algoritma u potpoglavlju 3.5.

Izbor parametra praga (`thBin`) bitan je za binarizaciju, tj. za selekciju piksela koji glasuju. Pokažimo to na nekoliko primjera:

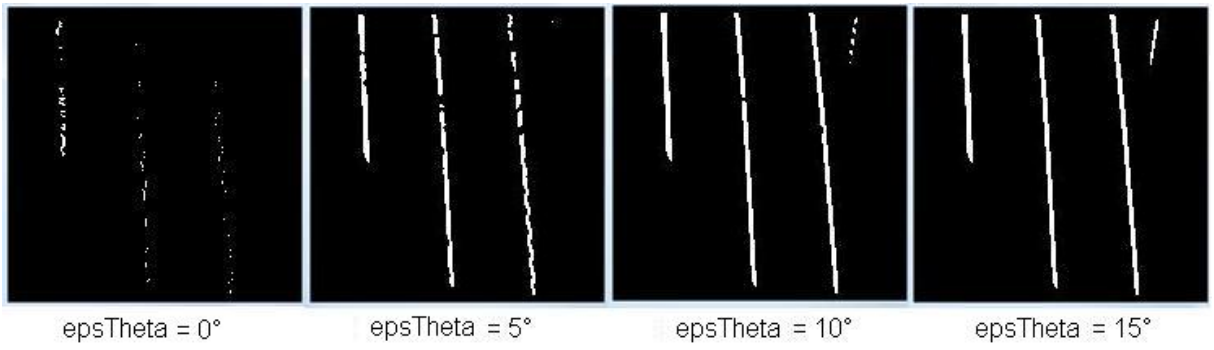


Slika 14. Odabir vrijednosti `thBin` (`thBin`=120 lijevo, `thBin`=60 desno)

Za navedene granice rezultati su prilično loši, kod izbora previsokog praga prošlo je previše smetnji. Kod izbora preniske granice nemamo šumova ali također nema ni onih piksela koji nam trebaju. Ispitivanjem više vrijednosti zaključeno je da idealna vrijednost za prag između 90 i 120 (slika 15).

Slika 15. `thBin` = 120

Treći parametar je dozvoljeni kut odstupanja (`epsTheta`) od dominantne orijentacije. Ukoliko ovaj parametar postavimo prenisko nećemo dobiti dobre linije. Ako postavimo parametar previsoko tada će i smetnje proći kroz ovaj postupak. Pogledajmo na slici 16 utjecaj odabira parametra `epsTheta`:



Slika 16. Prikaz utjecaja izbora parametra `epsTheta`

Slika 16 je prikaz binarne slike uz dodatno filtriranje orijentacije. Provjerava se da li je orijentacija piksela približno jednaka ($\pm$ `epsTheta`) dominantnoj orijentaciji. Vidimo da bez odstupanja većina korisnih piksela nestaje. Uz odstupanje od 15° filtriranje su prošle i smetnje koje želimo ukloniti. Znači idealno bi bilo da je odstupanje između 5° i 10° .

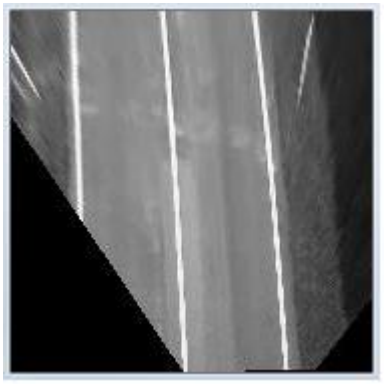
Eksperimentalnom metodom pokazalo se da su dobri parametri `sigma`=2, `thBin`=120 i `epsTheta`=5°. Navedeni parametri će biti korišteni u nastavku rada.

3.3. Ilustracija koraka obrade

Prvo prikazujemo ulaznu sliku (Slika 17), kako bi mogli pratiti rad algoritma.

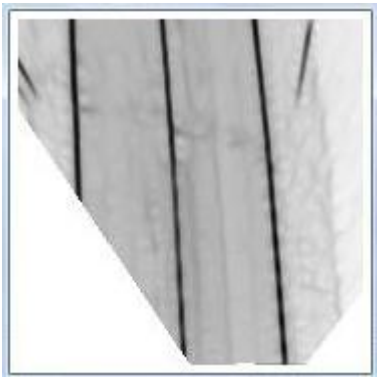
Slika 17. Ulazna slika

Druga slika nastaje inverznom perspektivnom transformacijom prve slike. Sada su širine prometnih linija jednake, dok kod prve slike to nije slučaj. Ova slika je manja zbog toga jer smo komponentu `ext_ipt` konfigurirali na način da naš algoritam bude brži.

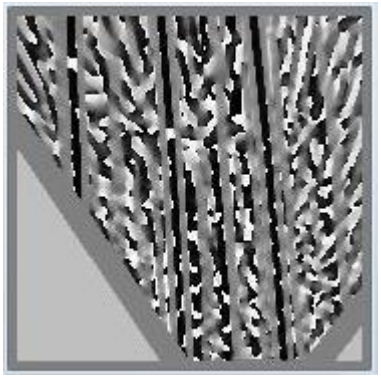


Slika 18. Slika nastala inverznom perspektivnom transformacijom

Iduće dvije slike (Slika 19 i Slika 20) nastaju nakon filtriranja temeljenog na Gauss-ovoj drugoj derivaciji. Prva od njih je odziv filtra, dok je druga slika orijentacije piksela (pomoću nje određujemo histogram).



Slika 19. Odziv upravljivog filtra

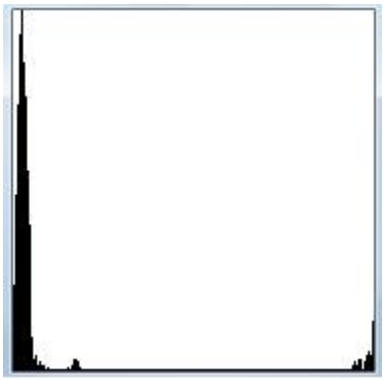


Slika 20. Orijentacija piksela

Slijedeća slika (Slika 21) nastaje binarizacijom slike 19.

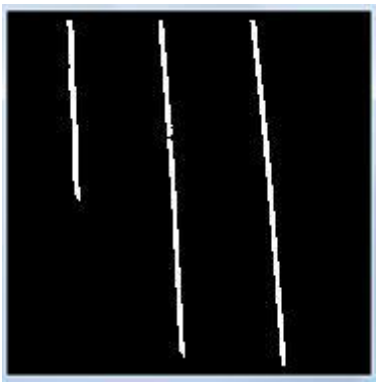
Slika 21. Binarna slika odziva upravljivog filtra

Iduću sliku (Slika 22) dobivamo iz slike 20, a to je histogram orijentacije. Odnosno grafički prikaz frekvencije pojavljivanja pojedinih kutova.

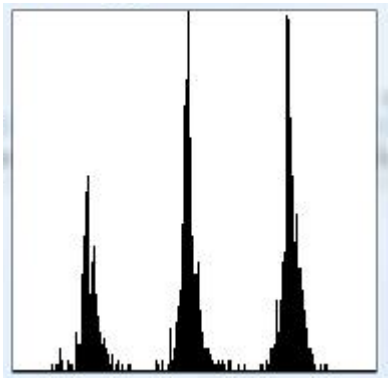


Slika 22. Histogram orijentacije piksela sa binarne slike

Iduće dvije slike (Slika 23 i Slika 24) nastaju Houghovom transformacijom. Prva od njih je binarna (Slika 21) uz dodatni uvjet da je orijentacija piksela približno jednaka dominantnoj orijentaciji. A druga slika je grafički prikaz akumulacijskog polja.

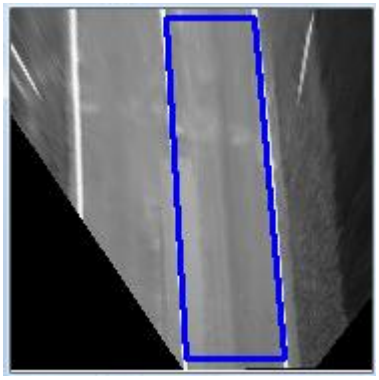


Slika 23. Binarna slika (uz dodatni uvjet da je orijentacija piksela približno jednaka dominantnoj orijentaciji)

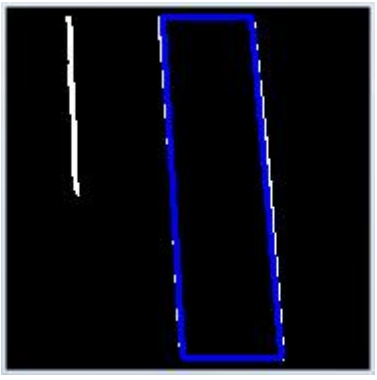


Slika 24. Grafički prikaz akumulacijskog polja

Sada pomoću akumulacijskog polja na ranije opisani način anotacijski označimo sliku 18 i sliku 23. I to će biti konačni rezultat po kojem se ocjenjuje uspješnost algoritma. Na taj način dobivaju se konačne slike (Slika 25. i Slika 26.).



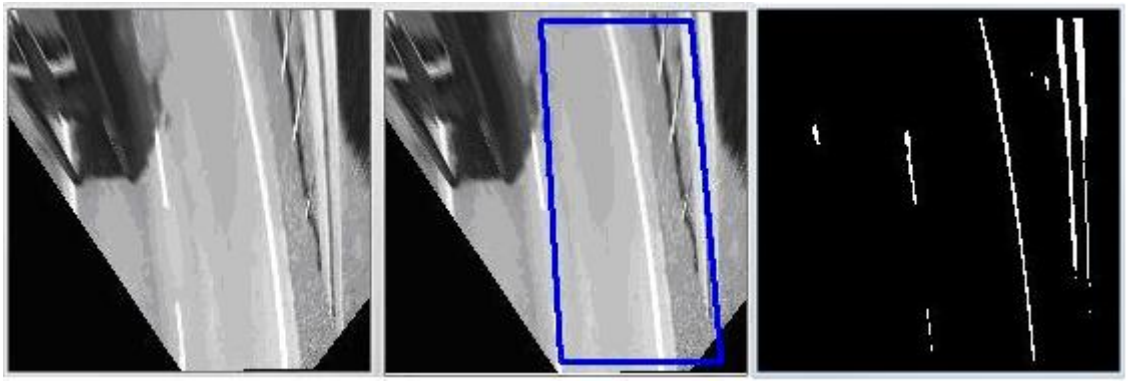
Slika 25. Detektiran prometni trak u inverznoj perspektivnoj slici



Slika 26. Detektiran prometni trak na binarnoj slici odziva upravljivog filtra

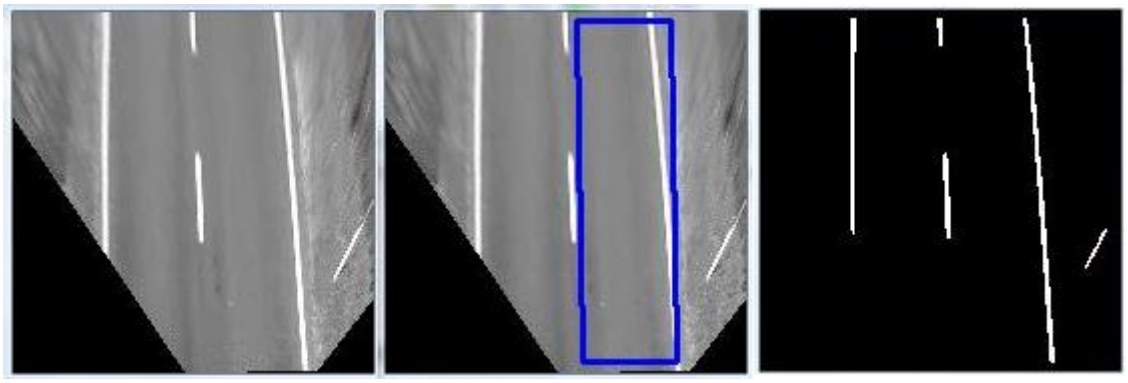
3.4. Procjena ispravnosti postupka

U ispitnom slijedu od 2552 slike algoritam je davao dobre rezultate za preko 2200 slika. U specifičnim uvjetima na početku (prvih stotinjak slika) i kraju (zadnjih stotinjak slika) ispitnog slijeda dolazilo je do grešaka. Na početku slijeda je automobil, odnosno kamera, smješten pokraj ceste i tada su stupići prepoznati kao linija. Na kraju slijeda se nalazi dodatna signalizacija na kolniku koja je također zbunila algoritam i došlo je do pogrešnog prepoznavanja. Još jedno pogrešno prepoznavanje se dogodilo kada je zaštitna ograda s desne strane ceste prepoznata kao linija. Ovaj primjer može se vidjeti na slici 27. Propuštene detekcije u ovom algoritmu nije bilo.



Slika 27. Pogrešna detekcija

Još je potrebno pokazati jedan slučaj koji se povremeno javljao. Ta situacija se javlja kada linije nisu paralelne. Pretpostavka dobrog rada ovog postupka je da su prometne linije paralelne.



Slika 28. Prometne linije nisu paralelne

3.5. Brzina izvođenja razvijenog postupka

Konačno, razmatramo brzinu izvođenja. Izvođenje se može podijeliti na nekoliko faza. Izmjerit ćemo trajanje za pojedine faze. Izvest ćemo 10 mjerenja za svaku pojedinu fazu.

Najdulje traje filtriranje temeljeno na Gaussovoj drugoj derivaciji (prosječnih 119.5 ms), nakon toga najviše vremena zauzima inverzna perspektivna transformacija (prosječnih 57.5 ms). Ukupno vrijeme izvođenja inverzne perspektivne transformacije i upravljivog filtra je 177 ms.

Ukupno trajanje kompletnog algoritma traje prosječnih 191.8 ms. Što znači da na Houghovu transformaciju otpada svega 14.8 ms.

Zaključak

Postupak korišten za detekciju prometnog traka temelji se na inverznoj perspektivnoj transformaciji, upravljivom filtru i jednodimenzionalnoj Houghovoj transformaciji. Ulaznu sliku prvo inverzno perspektivno transformiramo. Zatim na tako obrađenoj slici primijenimo upravljivi filtar. Nakon toga binariziramo odziv upravljivog filtra, te pomoću dobivene binarne slike i histograma orijentacije odredimo dominantnu orijentaciju. Slijedeći korak je popunjavanje akumulacijskog polja Houghove transformacije. Nakon toga pronalazimo lokalne maksimume u akumulacijskom polju. Na temelju tih vrijednosti pronalaze se linije koje omeđuju prometni trak. Opisani postupak za detekciju prometnog traka kombinira tri različite metode, od kojih svaka zasebno ne bi davala posebno dobre rezultate. No njihovim spajanjem dobiva se ohrabrujući rezultat.

Opisani postupak za detekciju prometnog traka u većini slučajeva na ispitnom slijedu daje dobre rezultate. No algoritam ipak ne bi mogao u konkretnu primjenu, zbog toga jer krećemo od nekoliko temeljnih pretpostavki koje ne moraju uvijek biti točne. Pretpostavljamo da su prometne linije paralelne i da se vozilo (kamera) nalazi u svom traku. No uz određene izmjene i usavršavanje algoritam ima potencijal za stvarnu upotrebu.

Ovaj postupak daje najbolje rezultate kada postoji kontrast između boje asfalta i prometne linije. No čak i uz smanjeni kontrast (jako osvjetljenje, sjene) dobivaju se zadovoljavajući rezultati. Jedan od mogućih problema je neodržavana prometna signalizacija, odnosno blijeđenje prometnih linija. Također problematično je kada prometne linije nisu paralelne.

Razvijeni postupak bi mogao poslužiti kao temelj za daljnji rad na području detekcije horizontalne prometne signalizacije. Neki od postupaka koje bi se mogli realizirati su: preciznije određivanje linija na kolniku, određivanje strukture svih trakova ceste (širokokutna kamera), prepoznavanje vrste prometnih linija (isprekidana, dvostruka).

Literatura

[1] Šegvić S., Brkić K., Kalafatić Z., Stanistavljević V., Budimir D., Dadić I.: Towards automatic assessment and mapping of traffic infrastructure by adding vision capabilities to a geoinformation inventory, Proceedings of Mipro, Opatija, 2009.

http://www.zemris.fer.hr/~ssegvic/pubs/mipro09.pdf , 2009.

[2] Šegvić S.: Višeagentsko praćenje objekata aktivnim računarskim vidom, Fakultet elektrotehnike i računarstva, doktorska disertacija, Zagreb, 2004.

[3] Majić A.: Pronalaženje prometnog traka korištenjem upravljivih filtara, Završni rad, Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[4] Šverko M.: Houghova transformacija, seminar, Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[5] Fitch A.J., Kadyrov A., Christmas W.J.,Kittler J.: Orientation correlation, Proceedings of BMVC , Guildford, Velika Britanija, 2002.

http://www.bmva.ac.uk/bmvc/2002/papers/95/full_95.pdf

[6] RGB color model, November 2008, Wikipedia, the free encyclopedia,

http://en.wikipedia.org/wiki/RGB_color_model, 10.06.2009.

[7] Simond N.: Obstacle Detection from Inverse Perspective Mapping and Super – Homography, Proceedings of IROS, San Diego, California, 2007.

http://hal.archives-ouvertes.fr/docs/00/16/58/00/PDF/iros07_nsimond.pdf

[8] Grayscale, November 2008, Wikipedia, the free encyclopedia,

http://en.wikipedia.org/wiki/Grayscale , 09.06.2009.

[9] Zisserman A.: Two-Dimensional Signal Analysis, Lecture Works

www.robots.ox.ac.uk/~az/lectures/sa/lect12.pdf , 2003.

[10] Bovik A.C. : Handbook of image and video processing

Academic Press, 2005, vol.2 , str.21.-57.

[11] Papić, V.: Obrada slika i računalni vid, 2005.

[12] Dostal D., et al: Cannyev detektor rubova, tehnička dokumentacija predmeta Projekt, Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[13] Hough transform, Wikipedia, the free encyclopedia,

http://en.wikipedia.org/wiki/Hough_transform , 13.05.2009.

[14] Hough transformation, the java applet

http://www.rob.cs.tu-bs.de/content/04-teaching/06-interactive/HNF.html , 12.06.2009.

[15] Fisher R., Perkins S., Walker A., Wolfart E.: Hough Transform, 2003

http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm , 10.06.2009.

Pronalaženje prometnog traka u odzivu upravljivog filtra

Sažetak

U ovom radu bavimo se pronalaženjem prometnog traka u slikama pribavljenim iz vozila u pokretu. Postupak koji koristimo se temelji na inverznoj perspektivnoj transformaciji, upravljivom filtru i Houghovoj transformaciji. Postupak je izveden u programskom jeziku C++, koristeći prethodno razvijene module za određivanje inverzne perspektivne transformacije, te odziva upravljivog filtra. Razvijena implementacija je evoluirana na slijedu od 2000 stvarnih slika, a dobiveni rezultati su prikazani i komentirani.

Ključne riječi: računalni vid, pronalaženje prometnog traka, inverzna perspektivna transformacija, upravljivi filtar, Houghova transformacija , obrada slike

Traffic lane detection using steerable filter response

Summary

In this work we consider traffic lane detection in images recorded from vehicle in motion. Algorithm which we use is based on inverse perspective transform, steerable filter and Hough transform. This algorithm is implemented in programming language C++, using previously developed modules for inverse perspective transform and steerable filter. Developed component was evolved in 2000 real images, and results are provided and discussed.

Keywords: computer vision, traffic lane detection, inverse perspective transform, steerable filter, Hough transform, image processing