SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 459

Pronalaženje prometnog traka analizom profila slikovnog gradijenta

Andrea Žabčić

Zagreb, siječanj 2009

Sadržaj

1. Uvod

Cilj rada je pronalaženje prometnog traka na slikama koje su snimljene iz automobila u pokretu. Metoda kojom ćemo se poslužiti prilikom pronalaska traka je veoma jednostavna. Analizirati ćemo profil slikovnog gradijenta i potražiti na slici vodoravna područja tamno-svijetlo-tamno, odnosno područja gdje dolazi do nagle promjene boje iz tamne u svijetlu (i obrnuto). Takva područja uglavnom predstavljaju promjene iz sive ceste u bijelu liniju i natrag u sivu cestu. Pokušat ćemo grupiranjem takvih područja pronaći linije prometnog traka.

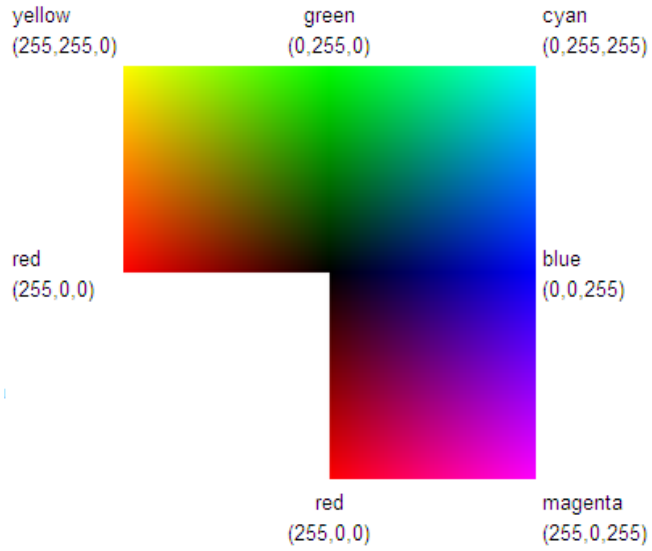
Strojna detekcija linija nam može biti korisna u automatiziranoj evaluaciji prometne signalizacije. Na taj način možemo brzo i efikasno provjeriti da li na nekom odsječku ceste linije traka zadovoljavaju standarde vidljivosti. Uz to postoje i neke druge moguće primjene. Sustav detekcije traka mogao bi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

Rad je strukturiran kako slijedi. Prvo uvodimo teoretske osnove algoritma, u poglavlju 2. Programsku realizaciju algoritma ćemo objasniti u poglavlju 3. U poglavlju 4 ćemo testirati program na testnim slikama. Testiranje otkriva koliko je naša metoda pouzdana, te u kojim slučajevima ju ne valja koristiti. Na kraju rada komentiramo postignute rezultate te donosimo ocjenu primjenjivosti algoritma.

2. Metode

2.1 Konverzija slike u boji u sivu sliku

Slike na kojima ćemo tražiti trak su predstavljene elementima modela RGB [3]. Model boja RGB se temelji se na trikromatskom svojstvu ljudskog oka. Pojedine boje se dobivaju kombinirajući crvenu, zelenu i plavu boju u drugačijim omjerima. Opseg veličina je od 0 do 1, za digitaliziranu sliku od 0 do 255. Na slici 1 je prikazan RGB model boja sa opsegom veličina [0,255].

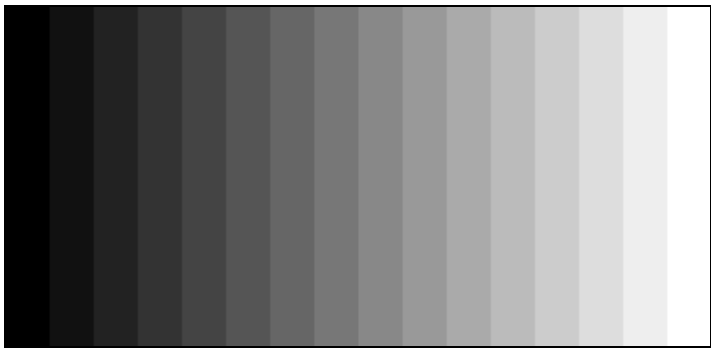


Slika 1: RGB model boja

Nama će za potrebe našeg algoritma biti potrebna siva slika. Siva slika za svaki piksel ima samo jednu vrijednost i to vrijednost intenziteta. Vrijednosti se nalaze u rasponu od 0 do 255, s time da je vrijednost 0 crna boja a 255 bijela. Vrijednost intenziteta se iz RGB vrijednosti dobiva sljedećom formulom:

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

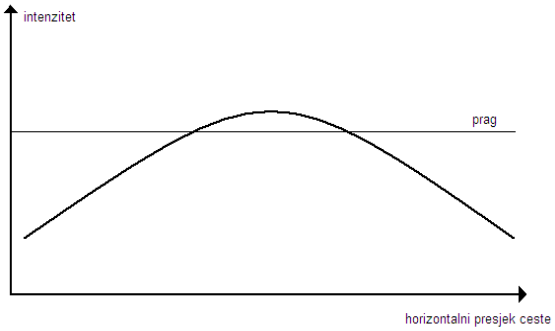
Na slici 2 je prikazan raspon boja u sivoj slici.



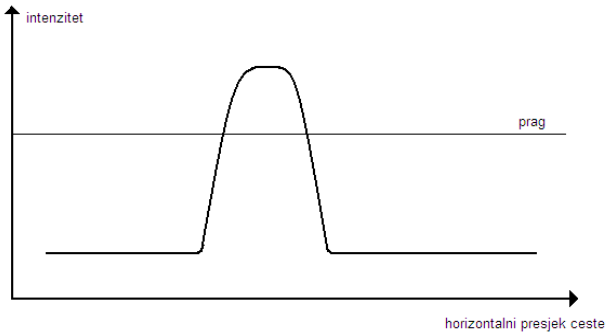
Slika 2: raspon boja u sivoj slici

2.2 Detekcija uzorka "tamno-svijetlo-tamno"

Gledajući vrijednosti intenziteta na slici primijetili smo da bijele linije imaju veliki intenzitet u odnosu na sivi asfalt koji ih okružuje. Kako bi izdvojili svijetle piksele koji su dijelovi linija moramo pronaći neku minimalnu vrijednost intenziteta koja može biti dio linije traka. Detaljan opis izračuna vrijednosti praga će biti opisan u poglavlju 3.3. Jednom kada izračunamo prag potrebno je izdvojiti sve piksele čiji je intenzitet iznad praga. No samo to nam neće biti dovoljno za detekciju linija traka, jer će se tada izdvojiti i slučajevi gdje se sporo povećava intenzitet (poput primjera na slici 3). U slučaju linija traka promjena intenziteta će biti nagla (vidi sliku 4): iz sive ceste u bijelu liniji (i obrnuto). Zato nam je potrebno proučavanjem slikovnog gradijenta pronaći takva područja nagle promjene intenziteta.



Slika 3: primjer neželjenog prijelaza t-s-t



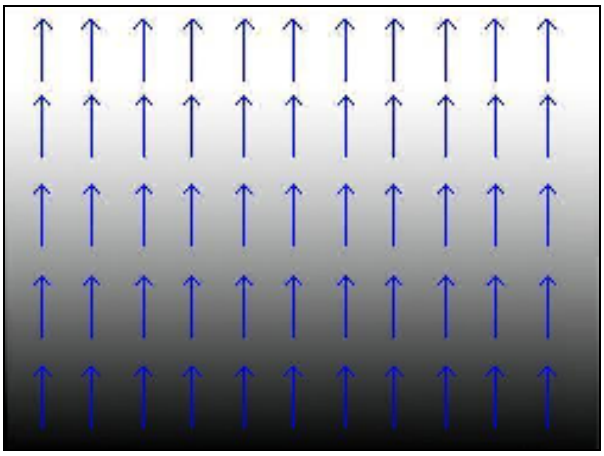
Slika 4: primjer željenog prijelaza

2.2.2 Gradijent

Gradijent funkcije je vektor koji pokazuje u smjeru najbržeg povećanja vrijednosti te funkcije. U našem slučaju u svakoj točki slike gradijent će pokazivati u smjeru u kojem se najbrže povećava vrijednost intenziteta. Važan podatak o vektoru gradijenta je njegova amplituda (veličina). Mala amplituda označava područje gdje se vrijednost sive veličine u slici sporo mijenja, dok velika amplituda označava područje gdje se naglo povećava vrijednost. Matematički je gradijent prikazan formulom (1).

$$gradI(x,y) = \nabla I(x,y) = \left[\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right] \tag{1}$$

Na slici 5 prikazan je primjer gradijenta na sivoj slici, gdje se postepeno boja mijenja iz crne prema bijeloj.



Slika 5: primjer gradijenta

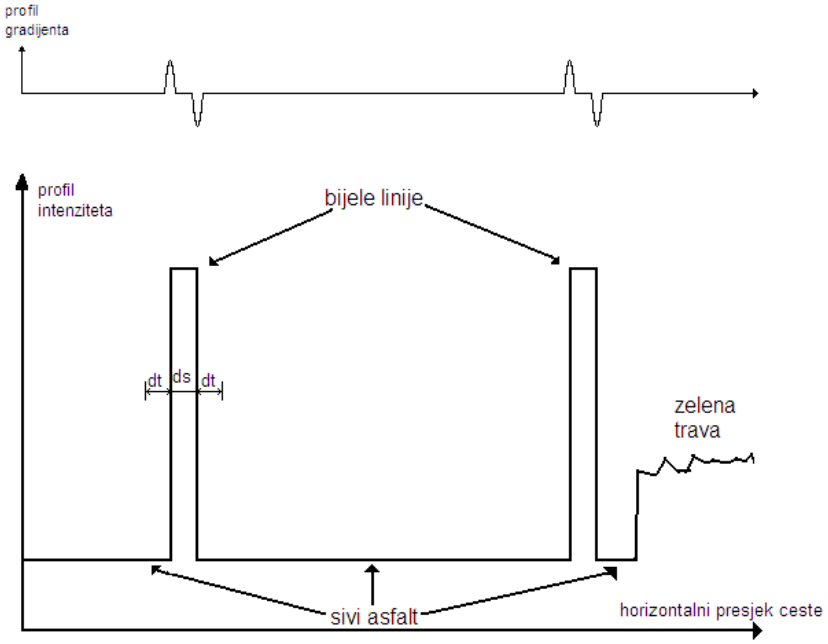
2.2.2 Model uzorka tamno-svijetlo-tamno

Nama neće biti dovoljno pronaći u svakoj točki smjer najbržeg povećanja vrijednosti intenziteta, nego ćemo morati pronaći ona područja u kojima dolazi do nagle promjene intenziteta. Neka od mjesta naglih promjena intenziteta su dijelovi gdje se prelazi sa sivog asfalta u bijelu liniju na cesti (ili obrnuto).

Proučavati ćemo sliku samo uzduž horizontalnih pravaca, te tražiti dijelove gdje dolazi do nagle promjene intenziteta. I to iz tamnog u svijetlo i onda nakon nekoliko desetaka piksela ponovo u tamno. Takva mjesta naglih promjena intenziteta su uglavnom promjene iz sive

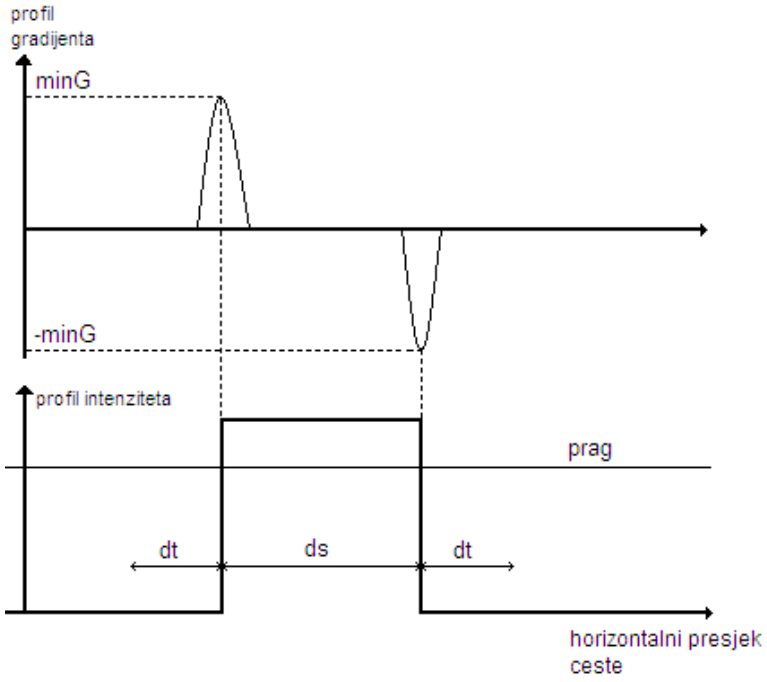
ceste u bijelu liniju i natrag u sivu cestu. Naš algoritam se upravo bazira na toj očitoj razlici u intenzitetima između ceste i linija [1].

Jedan primjer (idealni) vrijednosti intenziteta uzduž jednog horizontalnog pravca je prikazan na slici 6. Širina svijetlog područja označena je sa d_s, dok d_t označava minimalnu širinu tamnog područja.



Slika 6: vrijednosti intenziteta uzduž horizontalnog pravca

Na slici 7 je detaljnije prikazano samo jedno područje tamno-svijetlo-tamno (u nastavku: područje t-s-t). Primjećujemo da se profil gradijenta iznenada mijena na dijelovima slike gdje dolazi do nagle promjene intenziteta. S obzirom na predznak gradijenta možemo zaključiti da li dolazi do rasta, ili pada intenziteta, te ukoliko je veći od neke minimalne vrijednosti (minG) pronašli smo t-s-t područje.



Slika 7: područje t-s-t

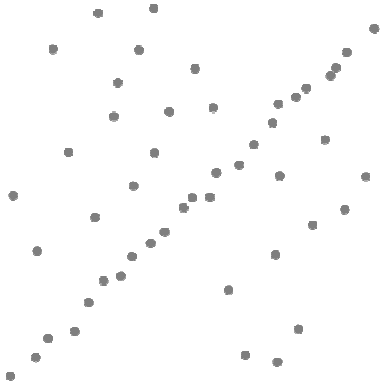
2.3 Pronalaženje pravca slučajnim uzrokovanjem (RANSAC)

Kada smo pronašli t-s-t područja uzduž horizontalnih linija potrebno ih je nekako grupirati, odnosno provjeriti koje od njih zajedno tvore liniju prometnog traka. To ćemo napraviti korištenjem metode poznate pod nazivom konsenzus slučajnih uzoraka ili RANSAC [5, 6].

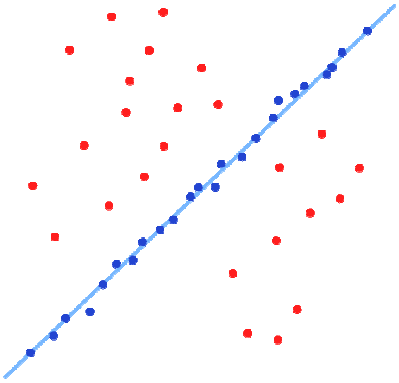
RANSAC (RANdom SAmple Consensus) je iterativna metoda za procjenu parametara matematičkog modela iz skupa podataka koja uključuju i *outlier*e[1]. To je nedeterministički algoritam, te pronalazi točne rezultate samo sa određenom sigurnošću, koja se povećava sa povećavanjem broja iteracija. Algoritam je prvi puta objavljen od strane Fischlera i Bollesa 1981. Osnovna pretpostavka je da se podaci sastoje od inliera (podataka čija se distribucija može objasniti skupom parametara modela) i outliera (oni ne pripadaj populaciji koju modeliramo). Glavna ideja je: slučajnim odabirom minimalnog uzorka hipotetizirati traženi model. Hipoteze evaluirati nad svim elementima ulaznog skupa podataka. Algoritam vraća najbolju hipotezu iz skupa inliers. Primjer jednog takvog matematičkom modela je pravac. Na slici 8 je prikazan skup točaka kroz koji je potrebno provući pravac, a na slici 9 rezultat

[1] Outlier je podatak iz skupa koji promatramo koji ne pripada razmatranoj populaciji

RANSAC algoritma. Plave točke su inlieri – točke koje otprilike tvore pravac, a crvene outlieri – točke koje nikako ne mogu tvoriti taj pravac. Jednostavnija metoda za pronalaženje takvoga pravca bi uglavnom generirala lošiji pravac, jer za razliku od RANSAC algoritma ne bi zanemarila točke koje su predaleko od pravca (outliere). Jedan takav primjer bila bi metoda koja minimizira ukupne kvadratne udaljenosti od pravca. Ona će pronaći pravac kod kojega je suma kvadratnih udaljenosti svih točaka od pravca minimalna, uključujući i outliere.



Slika 8: skup podataka kojima se treba provući pravac



Slika 9: pravac provučen metodom RANSAC

Mi ćemo za naš algoritam iskoristiti upravo ovu inačicu RANSAC-a sa provlačenjem pravca kroz točke. Skup ulaznih podataka će biti točke koje predstavljaju sredinu područja t-s-t koje smo pronašli kao potencijalne dijelove traka. Kroz taj skup točaka provući ćemo pravac, te točke koje se na kraju postupka nalaze u skupu inliers su dijelovi linije traka. Na taj način ćemo pronaći pravac koji prolazi kroz najviše točaka, te područja t-s-t čija su središta točke iz inliers označiti kao dijelove linije traka.

3. Programska interpretacija

3.1 Konverzija u sivu sliku

Prvi korak u programskoj interpretaciji algoritma je pretvorba slike iz početne slike u boji (RGB model) u sivu sliku. Tri vrijednosti R,G, i B iz početne slike se preračunavaju u samo jednu – intenzitet. Za to koristimo funkciju koja je razvijena u okviru predmeta Projekt [7]. Funkcija sliku pretvara iz RGB modela boja u HSI model. HSI model boja ima tri

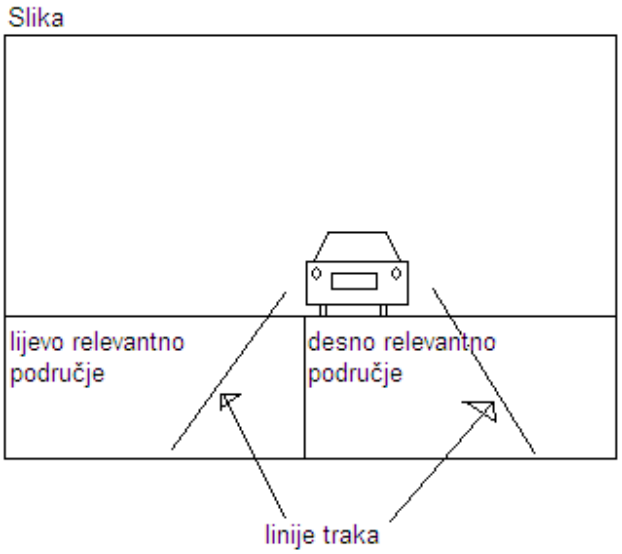
vrijednosti: *Hue*, *Saturnation* i *Intensity*. *Intensity* je ekvivalent intenzitetu u sivoj slici, te ćemo tu vrijednost koristiti.

Sučelje funkcije:

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

3.2 Odabir relevantnog dijela slike

U postupku pronalaženja traka nepotrebno je razmatrati cijelu sliku, te ćemo se fokusirati samo na donju trećinu slike, jer upravo tamo očekujemo da će se nalaziti cesta. Gornju polovicu slike je nepotrebno gledati jer tamo se nikako ne nalazi cesta, zbog načina na koji su snimljene slike, te bi bio samo gubitak vremena tražiti trakove tamo. Radi točnosti smanjili smo relevantni dio slike sa pola na trećinu kako bi eliminirali neke smetnje koje bi se mogle pojaviti (npr. drugi automobil). Dodatno ćemo podijeliti sliku popola kako bi zasebno tražili lijevi kraj traka, i zasebno desni (vidi sliku 10).

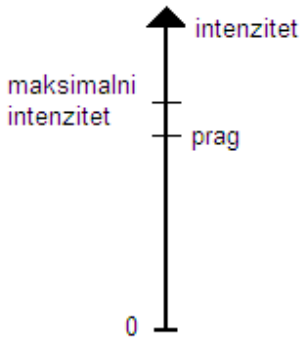


Slika 10: odabir relevantnog dijela slike

Kako bi algoritam bio što brži nećemo razmatrati cijelu donju trećinu slike, nego ćemo tražiti linije traka samo uzduž horizontalnih linija, koje su međusobno udaljene za 1/40 visine slike.

3.3 Odabir praga za svijetle piksele

Nakon što smo izračunali vrijednosti intenziteta pojedinih piksela slike, potrebno je na temelju tih vrijednosti pronaći dijelove koji su potencijalne linije traka. Krenuti ćemo tako da prvo odredimo najniži intenzitet koji je prihvatljiv za liniju traka, te izdvojiti sve svjetlije piksele, tj. piksele sa većim intenzitetom. Taj prag ćemo izračunati tako da pronađemo najveću vrijednost intenziteta (maxI) na relevantnom dijelu slike, posebno za lijevu stranu slike i posebno za desnu, te ju umanjimo za 20% njezine vrijednosti, odnosno prag će biti 0.8*maxI.



Slika 11: izračun praga

Prag se računa posebno za lijevu i posebno za desnu stranu radi povećanja točnosti algoritma. Kada bi bio jedan prag zajednički cijeloj slici u slučajevima kada bi jedna linija traka bila znatno svjetlija od druge maksimalni intenzitet bi bio pronađen u toj svjetlijoj liniji, te druga linija ne bi uopće bila detektirana, jer bi joj vrijednosti intenziteta bile manje od praga.

3.4 Izdvajanje svijetlih piksela

Sada kad smo našli donji prag za svijetle piksele potrebno je sa slike izdvojiti sve piksele koji su iznad tog praga, te kao takvi su potencijalno dijelovi linije na cesti. Sliku ćemo popola podijeliti i razmatrati zasebno lijevu i desnu polovicu slike, pretpostavljajuću da će se na lijevoj polovici nalaziti linija za lijevi kraj traka, odnosno na desnoj strani desna linija.

Dijelove traka tražimo individualno na svakoj od horizontalnih linija. Kada pronađemo piksel čiji je intenzitet iznad odgovarajućeg praga (praga za lijevu polovicu slike, odnosno praga za desnu polovicu) vršimo odgovarajuće kalkulacije. Ukoliko je to prvi „bijeli“ piksel nakon „crnog“ pamtimo ga kao početak crte koja bi mogla biti horizontalni presjek linije

traka. Nakon toga pogledamo koliko još bijelih piksela ima u nizu iza njega, te zadnji od njih zapamtimo kao kraj crte. Kada smo zapamtili jednu crtu potrebno je provjeriti da li je ona odgovarajuće duljine da bi mogla biti dio traka. Sve crte između 10 i 50 piksela prolaze u daljnju provjeru, jer se očekuje da će debljina linije traka biti u tome rasponu (vidi sliku 7 iz poglavlja 2.2.2., gdje je $d_s \in [10,50]$).

Zadnja provjera prije nego li zapamtimo crtu kao dio linije traka je da provjerimo da li se lijevo i desno od te nakupine svijetlih piksela nalaze tamniji pikseli koji predstavljaju sivu cestu. To ćemo napraviti tako da provjerimo da li su prosjeci intenziteta piksela na pola duljine crte lijevo i desno od crte barem za 35 manji od prosjeka intenziteta crte (vidi sliku 7).

Kada se uvjerimo da smo doista naišli na područje t-s-t potrebno je zapamtiti njegovu poziciju. Podaci koje ćemo zapamtiti su koordinate x i y središta crte (odnosno nakupine bijelih piksela) te polovica širine. Za potrebe toga stvorili smo klasu `MyLine` gdje su `x_` i `y_` koordinate središta crte a `width_` polovica širine.

Klasa:
```
class MyLine{
public:
      MyLine(int x, int y, int w):
                  x_(x), y_(y), width_(w)
      {}
public:
      int x_,y_;
      int width_;
};
```

Podatke o područjima t-s-t pospremamo u niz kojemu su pojedini članovi tipa `MyLine`. Stvorili smo dva takva niza za potrebe odvojenog spremanja podataka o linijama lijeve i desne polovice slike.

3.5 Pronalaženje traka

Nakon što smo u niz smjestili podatke o područjima t-s-t, koja su prošle naše provjere i vjerojatno su dijelovi linije traka, potrebno je proučavanjem svih područja i njihovog prostornog razmještaja ustanoviti koje od njih predstavljaju liniju traka. To ćemo obaviti funkcijom `nadjiTrak`.

Sučelje funkcije:
```
void nadjiTrak(const std::vector<MyLine>&
                  podaci,std::vector<int>& inliers)
```

Funkcija kao argumente prima pokazivač na niz u kojem se nalaze podaci o područjima t-s-t, te pokazivač na poziciju u memoriji na koju će funkcija spremiti indekse podataka o područjima koje predstavljaju liniju traka. Funkciju ćemo pozvati dva puta: sa podacima o područjima lijeve strane slike, te sa podacima o područjima desne strane.

Funkcija se temelji na algoritmu RANSAC opisanom u poglavlju 2.3, koji kroz skupinu točaka provlači pravac, zanemarujući točke koje su predaleko od pravca.

Postupak koji smo implementirali je sljedeći: iz skupa `podaci` slučajnim odabirom se izaberu dva područja t-s-t (koja su oblika `MyLine`) kroz koje se zatim provuče pravac.

Zatim se poziva funkcija `evaluirajPravac` koja nam služi za evaluaciju pravca. Ona uzduž pravca traži točke iz početnog skupa (`podaci`) koje su dovoljno blizu pravca, te iste stavlja u niz `tocke`.

Sučelje funkcije:
```
int evaluirajPravac(int x0,int y0, double k,
                    const std::vector<MyLine>& podaci,
                    std::vector<int>& tocke)
```

Njezini ulazni parametri su koordinate prve točke iz koje smo povukli pravac, koeficijent smjera pravca, te mjesto u memoriji gdje se nalaze podaci o područjima t-s-t. Zadnji argument je pokazivač na mjesto u memoriji gdje će funkcija spremiti podatke inliers, odnosno indekse podatka o točkama (područjima t-s-t) koje se nalaze u blizini pravca kojeg trenutno promatramo. Povratna vrijednost funkcije je brojač inliera, odnosno broj pronađenih točaka. Funkcija prolazi kroz niz `podaci` te računa udaljenost svake točke od pravca, te ukoliko se nalazi u blizini pravca (maksimalna dopuštena udaljenost od pravca je 6 piksela) dodaje njen indeks u niz `tocke`.

Funkcija `nadjiTrak` poziva 200 puta funkciju `evaluirajPravac` i to sa različitim pravcima slučajno odabirajući po dvije točke iz skupa podaci. Među njima nalazi onaj pravac koji ima najveći broj točaka (područja t-s-t) u blizini, te indekse tih područja smješta u niz `inliers` i vraća u glavni program.

3.6 Integracija s ljuskom cvsh

Programsko okruženje dobavlja sliku u obliku polja koje predstavlja dvodimenzionalnu matricu slikovnih elemenata u modelu RGB. Slika je u polju zapisana u sljedećem obliku: „RGBRGBRGBRGB…“. U istom obliku se predstavlja i HSI slika, gdje su vrijednosti zapisane u obliku: „HSIHSIHSI...“.

Sve gore navedene funkcije se odvijaju u metodi `alg_trak::process` koja je deklarirana u klasi `alg_trak` i to na sljedeći način:

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

Izvorna slika nalazi se u nultom elementu polja src (src[0]). Za HSI sliku se zauzme obično float polje dimenzija 3*visina*širina, gdje su visina i širina dimenzije slike. Nakon toga poziva se funkcija `RGBtoHSI`, te nakon određenih kalkulacija i funkcija `nadjiTrak` unutar koje se poziva funkcija `evaluirajPravac`.

Na izvornoj slici je potrebno na neki način označiti dijelove traka koje smo pronašli. Ljuska omogućava iscrtavanje vektorskih elemenata (anotacija) preko slike, te ćemo koristeći te mogućnosti iscrtati crte koje smo pronašli uzduž horizontalnih linija i koje su dio linije traka. Za to koristimo funkciju ljuske `addLine`, koja dodaje liniju na anotaciju. Ona je deklarirana u klasi `win_ann` na sljedeći način:

```
void win_ann::addLine(
      const grid::Line& line,
      int col, char const* str,
      LineStyle style, int width);
```

Mi ju pozivamo na sljedeći način (gdje su `l.x_` i `l.y_` x i y koordinate središta crte, a `width_` polovica duljine crte) :

```
ann.addLine(grid::Line(
            grid::Pixel(l.x_-l.width_, l.y_),
            grid::Pixel(l.x_+l.width_, l.y_)),
            0, "", win_ann_abstract::LS_Normal, 2);
```

Napominjemo da u funkciju nisu ispravno implementirane konfiguracijske mogućnosti programskog okruženja, što znači da se svi parametri moraju ručno mijenjati, te se nakon svake promjene okruženje mora ponovo prevesti.

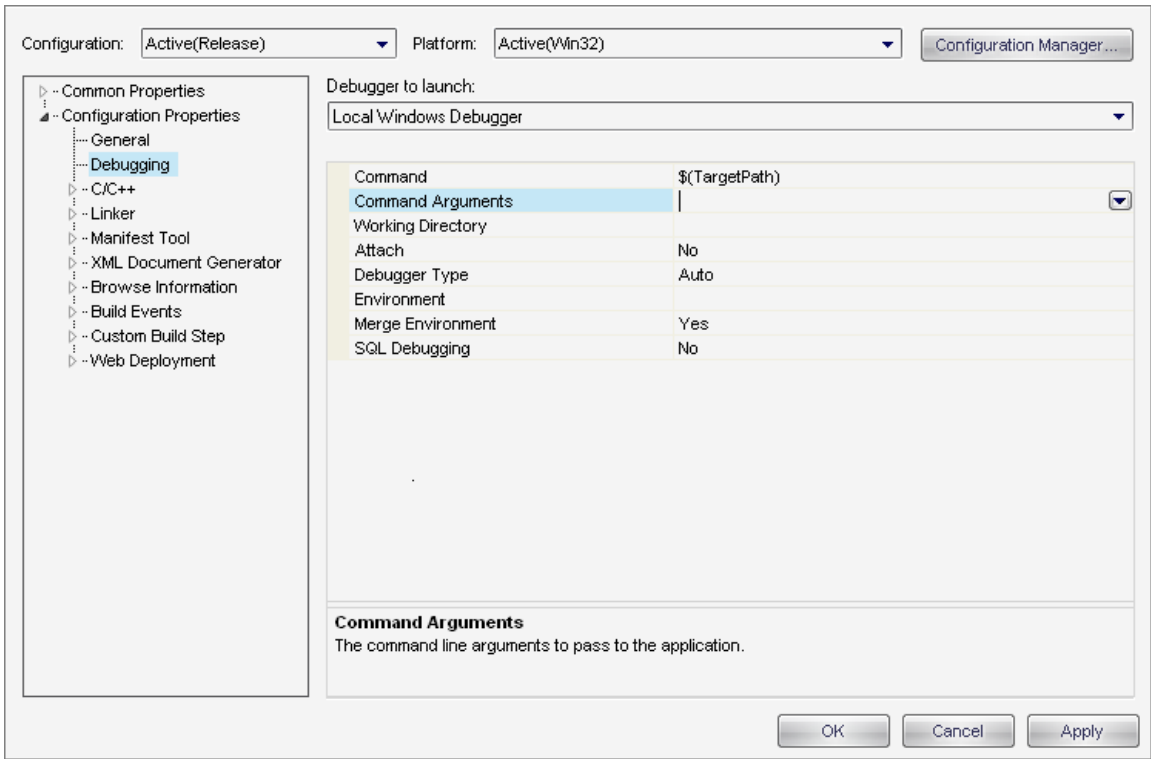
4. Eksperimentalni rezultati

Da bi ocijenili uspješnost algoritma potrebno ga je testirati. Najprije moramo stvoriti reprezentativni uzorak slika na kojima će se vršiti testiranja. To ćemo detaljnije opisati u poglavlju 4.2.

Nakon toga se odlučujemo za način testiranja. S obzirom na narav algoritma nemoguće ga je testirati automatski, te ćemo stoga morati ručno gledati rezultate algoritma za svaku sliku, te ocijeniti točnost algoritma.

4.1 Rad s programskim okruženjem

Naredbeni redak se u programskom okruženju Visual Studio 2005 zadaju u: *Project Properties → Debugging → Command Arguments*



Slika 12: mjesto za upis naredbe

Svaka se naredba sastoji od tri osnovna dijela: putanje izvorne datoteke (source), odredišta dobivene datoteke (destination) i algoritma koji se koristi. Dodatno možemo koristiti naredbu –i (interactive) za interakciju za vrijeme provođenja algoritma.

Putanja izvorne datoteke se zadaje na sljedeći način:

-sf=…\video.avi

-sf=...\video.wmv

ili

-sf=…\video.avi#početna_slika-završna_slika

Odredište dobivene datoteke:

-df=…\rezultat.avi

Algoritam koji se koristi:

-a=algoritam

Nekoliko primjera potpune naredbe:

-sf=C:\video.wmv -a=trak

-sf=C:\video.wmv#1-20 -df=C:\temp\video.avi -a=trak

-sf=C:\video.wmv -a=trak –i

4.2 Odabir reprezentativnog skupa slika

Odabir reprezentativnog skupa slika je proces u kojem se iz skupa svih dostupnih slika odabire puno manji skup slika i to tako da se izaberu slike koje smatramo zanimljivima za ocjenjivanje performansi.

Da bi što bolje ocijenili performanse algoritma izabiremo slike iz tri snimke (vmw formata). Na prvom slijedu slika (*autoput*) je snimljeno putovanje autoputom, na drugom (*grad*) je snimljena gradska cesta, te na trećem (*selo*) ruralna sredina. Iz njih izdvajamo stotinjak slika. Prilikom tog postupka obraćamo pažnju na to da su slike odgovarajuće kvalitete (realne boje, nepostojanje refleksija, te nepostojanje zamućenja), te da budu reprezentativne, odnosno da obuhvaćaju različite situacije.

4.3 Postignuti rezultati

Sada kada smo izdvojili stotinjak slika krećemo sa testiranjem algoritma na njima. Pokretat ćemo algoritam za sve slike i proučavanjem izlaza za svaku sliku zaključiti koliko je algoritam uspješan, te u kojim situacijama podbacuje.

Kroz testiranje algoritma na slikama izvučenim iz tri pripremljena slijeda zaključujemo da algoritam radi zadovoljavajuće na slijedu slika *autoput* i slijedu *grad*, no kod slijeda *selo* su rezultati prilično loši. Razlog tome je nedovoljna razlika intenziteta između boje asfalta i boje linija na cesti u kombinaciji sa jakim sunčanim osvjetljenjem koje još dodatno posvjetljuje asfalt. Osnovna metoda ovog algoritma se bazira na pretpostavci da će razlika prosječnog intenziteta između bijele linije i sivog asfalta biti najmanje 35. Ta je brojka dobivena isprobavajući razne intenzitete u rasponu od 20 do 40, te se pokazala kao najbolja na većini slika. Išlo se za time da se pronađu što točnije linije na cesti, ali i da se što više eliminiraju pogreške. Ukoliko se za potrebe slijeda slika selo smanji ta razlika na 20 ili 25, na dijelovima se puno bolje pronalazi prometni trak, ali u isto vrijeme je količina pogrešno pronađenih trakova (odnosno prepoznavanje trakova tamo gdje ih nema) veća. Na slici 13 je prikazan primjer propuštene detekcije traka, slika je izvađena iz slijeda slika *selo* (uz razliku intenziteta postavljenu na 35).

Slika 13: propuštena detekcija

Slika 14: neispravna detekcija traka

Testirajući algoritam na slijedu *selo*, snimljenom u ruralnoj sredini, primjećujemo da je na otprilike trećini slika pogrešno detektirani trak (upravo na onima na kojima je jako sunčano

osvjetljenje), te je očito da ova metoda nije zadovoljavajuća za taj slijed slika, te se dalje fokusiramo na slike izvučene iz druga dva slijeda.

Nakon proučavanja izlaza algoritma za ostale slike pronalazimo nekoliko čestih uzročnika pogreška u određivanju traka. To su:

- slabo vidljive zebre
- te strelice za usmjeravanje prometa nacrtane na kolniku

Najčešći uzročnici pogrešnog određivanja traka su slabo vidljive zebre, izbljeđene te oštećene, te kao takve se često neki njezini dijelovi pogrešno detektiraju kao linija. Takve probleme je nemoguće potpuno izbjeći ovom metodom. Moglo bi ih se malo ublažiti modifikacijom algoritma na način da u traženju traka uzduž jedne horizontalne linije algoritam uzima u obzir rezultate nekoliko prethodnih linija, te zahtjeva da na tekućoj horizontalnoj liniji trak prati smjer kojim je krenuo u prethodnim linijama. No ni takva modifikacija ne bi mogla otkloniti pogrešno nađeni trak na slici 15.

Jedino rješenje koje bi potpuno uklonilo takve pogreške bi bilo redovito bojanje zebra, koje u tome slučaju ne prolaze provjeru te se ne detektiraju pogrešno kao linije traka.

Slika 15: dijelovi zebre pogrešno detektirani kao linije traka

Sljedeći problem su strelice za usmjeravanje prometa nacrtane na kolniku (vidi sliku 16). Debljina im je jednaka očekivanoj debljini linije traka, te u slučaju izostanka jedne od linija vrlo je vjerojatno da će se strelica detektirati kao linija traka. To bi mogli spriječiti sličnom modifikacijom kao kod problema zebri. Uz tamo opisanu modifikaciju uputno bi bilo zadati minimalnu udaljenost lijevog i desnog kraja traka, koju bi odredili eksperimentalno, kako bi eliminirali strelice.

Slika 16: strelica pogrešno detektirana kao linija traka

Na slici 17 i slici 18 nalaze se primjeri dobrog rada algoritma.

Slika 17

Slika 18

Još se pojavio jedan slučaj pogrešne detekcije traka, ali on je bio vezan isključivo za slike izvučene iz slijeda *grad*. Tamo se na dijelovima gdje nije iscrtan desni kraj traka pogrešno svjetliji rub ceste prepoznao kao trak (vidi sliku 19). Tome nema pomoći kod ove

metode, jer na obje strane tog svijetlog ruba se nalaze tamniji dijelovi. No ovaj slučaj nije toliko problematičan jer je specifičan za ovu cestu.

Slika 19:pogrešna detekcija desnog ruba traka

5. Zaključak

Algoritam temeljen na detekciji uzorka tamno-svijetlo-tamno je poprilično jednostavan algoritam koji ima kratko vrijeme izvođenja, te bi kao takav mogao biti koristan kao pomoć prilikom vožnje, jer brzo obrađuje pristigle podatke, te gotovo isti čas daje rezultate. No ipak prije takve primjene bi bilo potrebno napraviti dosta dodatnih preinaka kako bi se više prilagodio toj upotrebi. Trenutni algoritam radi u jednostavnim situacijama, a to je normalna vožnja sredinom traka. On unaprijed pretpostavlja da se lijeva linija traka nalazi na lijevoj polovici slike, a desna na desnoj, te ukoliko bi došlo do složenijih situacija – kao što je prestrojavanja iz jedne trake u drugu ili pretjecanje drugih vozila algoritam više ne bi dobro pronalazio linije traka.

Za točan rad algoritma potrebna je dovoljno velika razlika intenziteta tamno-svijetlo-tamno područja, te ukoliko ta razlika nije dovoljno naglašena, kao što je prikazano na primjeru slijeda slika snimljenih u ruralnoj sredini, gdje je uz neobično svijetli asfalt cesta još i jako osvijetljena suncem, algoritam neće odgovarajuće raditi.

Algoritam na jednostavnijim slikama uglavnom točno pronalazi linije traka izuzev pojedinih slika na kojima se nalaze izblijedjele zebre ili strelice za usmjeravanje prometa. No on pretpostavlja da će linije traka biti pravilne, te to ne provjerava dodatno. Tako da ukoliko bi se negdje na cesti umjesto ravne linije nacrtala blago valovita linija algoritam bi i nju detektirao kao liniju traka, jer detektira liniju koristeći RANSAC algoritam koji će pronaći točke koje se nalaze u blizini pravca, a ne nužno na pravcu. Zato bi, ukoliko bi htjeli precizniji algoritam koji bi takve slučajeve eliminirao, morali preinačiti algoritam tako da on na temelju smjera u kojem je krenula linija traka očekuje nastavak linije na određenom mjestu.

6. Literatura

[1] Goldbeck J., Image and Vision Computing: Lane following combining vision and DGPS, 18 izdanje,Elsevier, 2000

[2] HSL and HSV, 25.6.2003., *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/HSI_color_space, 10.11.2008.

[3] RGB color model, 26 November 2008, *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/RGB_color_model, 29.11.2008

[4] D.Begusic, Inžinjerska grafika, *Boja u računarskoj grafici,* http://lab405.fesb.hr/igraf/Frames/fP5_1.htm, 15.11.2008

[5] RANSAC, 14.11.2008., *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/RANSAC, 20.10.2008

[6] Charles V. Steward, Robust parameter estimation in computer vision, SIAM Review, Vol. 41, No 3 (1999), str. 513-537

[7] Babić T.,Lukinić T., Kovač D., Popović K., Rojković D.,Šverko M., Prepoznavanje prometnih znakova na temelju boje, tehnička dokumentacija predmeta Projekt iz programske potpore, Fakultet elektrotehnike i računarstva, 2008

Pronalaženje prometnog traka analizom profila slikovnog gradijenta

Sažetak

U ovome radu bavimo se pronalaženjem prometnog traka na slijedu slika snimljenih iz vozila u pokretu. Metoda kojom se koristimo bazira se na analizi profila intenziteta i gradijenta slike. Ta metoda na cesti traži područja tamno-svijetlo-tamno koja predstavljaju promjene iz sive ceste u bijelu liniju i obrnuto. Uz pomoć RANSAC algoritma grupiraju se više tamno-svijetlo-tamno područja koja zajedno tvore jednu liniju traka. Prvo se opisuju teorijske osnove, nakon toga programska realizacija, te na kraju eksperimentalna evaluacija postupka.

Ključne riječi: pronalaženje prometnog traka, detekcija uzorka „tamno-svijetlo-tamno“, RANSAC

Traffic lane detection analysing gradient profile

Summary

In this paper we are occupied with traffic lane detection based on video recorded from vehicle in motion. Our method is based on analysis of horizontal intensity and gradient profile. The method searches for dark-light-dark areas on the road which represent changes from gray road to white lane and back to gray road. Using the RANSAC algorithm we are grouping several dark-light-dark areas which are together forming one traffic line. First we describe theoretical basics, than we describe program realization, and finally provide experimental results.

Keywords: traffic lane detection, detection of “dark-light-dark” pattern, RANSAC