Detekcija prometnih sudionika

Tehnička dokumentacija

Verzija 1.1

Studentski tim: Viktor Braut

Slavko Grahovac

Mirko Jurić-Kavelj

Melita Kokot

Igor Lipovac

Vedran Vukotić

Nastavnik: Prof. dr.sc. Siniša Šegvić

Sadržaj

1. Uvod

Računalni vid je polje računarstva koje uključuje metode za pristup, analizu i procesiranje slika iz stvarnog svijeta. Ulazni podaci u programskim implementacijama ovih metoda su slike ili njihova sekvenca, video-snimka. Kao izlaz se pojavljuju korisne numeričke ili simboličke informacije nastale nizom operacija nad ulaznim podacima. Jedna od primjena računalnog vida nalazi se u analizi snimaka prometa. Jedan od ključnih problema u prometnoj znanosti je minimizirati zagušenost i gužve na urbanim prometnicama, posebice na raskrižjima od većeg tranzitnog značaja. Za veće pomake u razrješavanju ovog problema važno je poznavati podatke o gustoći prometa na raskrižju. Upravo u tu svrhu može poslužiti ekstrakcija podataka iz snimke raskrižja pomoću računalnog vida. Iz tog razloga teži se automatizirati praćenje potrebnih statistika.

Svrha ovog projekta je naći rješenje problema praćenja prometnih objekata na području urbanog raskrižja. Ulazni podatak u ovom slučaju bit će video-snimka urbanog raskrižja. Snimka je pribavljena kamerom iz ptičje perspektive. Pomoću metoda računalnog vida na ulaznu snimku primijenjen je niz transformacija. Kroz taj niz koraka dobivena je konačna, izlazna snimka kao rezultat projekta. Najprije je primijenjena metoda pretprocesiranja ulazne snimke. Taj postupak uključuje zaglađivanje slike primjenom Gaussovog filtra. Svrha ovog koraka je transformacija ulazne snimke radi lakšeg obrađivanja u sljedećim fazama projekta. Pretprocesirana snimka je zatim korištena za modeliranje pozadine. Omogućeno je dinamički izgraditi i ažurirati pozadinsku sliku. Zahvaljujući ovom koraku bilo je moguće dalje analizirati regije slike na kojima se događa kretanje objekata. Prednji plan, odnosno dio slike koji se izdvaja od pozadine je identificiran u sljedećem koraku. Objekti koji su se kretali u usporedbi s pozadinom su definirani kao grupe piksela prednjeg plana. Imajući jasno odvojenu pozadinu i prednji plan krenulo se u izgradnju binarne slike. Pikselima pozadine na binarnoj slici pridružena je crna boja, dok su pikseli prednjeg plana označeni bijelom bojom. Zadnji cilj projekta bio je ostvariti praćenje objekata na binarnoj slici. Detektirani su bridovi objekata koji su se koristili za daljnje praćenje kretanja. Kroz implementaciju projekta pojavili su se i dodatni problemi koje je trebalo riješiti. Ostvarena je stabilizacija ulazne snimke zbog velikih utjecaja

na rezultate uslijed gibanja kamere.

Programsko ostvarenje projekta sastoji se od niza metoda koje su primijenjene na snimku. Jezik implementacije je C++, programski jezik opće namjene. U testiranju i prevođenju programskog koda korištena su razvojna okruženja Microsoft Visual Studio i G++ kao dio okruženja GNU Compiler Collection(za Linux operacijski sustav). U implementaciji se koristila biblioteka programskih funkcija OpenCV. To je open-source biblioteka namijenjena primjeni računalnog vida u stvarnom vremenu. OpenCV sadrži niz funkcija koje omogućavaju rješavanje problema projekta. U ovom projektu funkcije visoke apstrakcije, kao što su pretprocesiranje ulazne snimke i detektiranje objekata, ostvarene su uz pomoć osnovnih funkcija i struktura OpenCV biblioteke, odnosno, zasebno su implementirane. Dodatno ostvarene funkcionalnosti poput primjene određenih algoritama optimizacije i ubrzanja programskog produkta razlikuju ovaj projekt od ostalih proučavanih implementacija. Korišten je pristup koji omogućuje programsku kvalitetu u smislu djeljivosti i modularizacije programskog koda.

U sljedećim poglavljima bit će prikazan rad na projektu, proces testiranja i zaključci. Poglavlja 2-6 sadrže opis korištenih metoda, algoritama, kao i programsku izvedbu istih. Opisani su načini implementacije po fazama projekta. Sedmo poglavlje fokusira se na dobivenim rezultatima, u 8. poglavlju opisane su upute za korištenje programskog produkta, a u 9. poglavlju je opisan postupak izrade video snimki. Posljednje (10.) poglavlje sadrži popis korištene literature.

2. Gaussovo zaglađivanje

Zaglađivanje slike bitno utječe na detekciju bridova (koja će biti obrađena kasnije). Nezaglađene slike mogu rezultirati s velikom količinom bridova koji zapravo nisu potrebni i samo otežavaju kasnije prepoznavanje objekata. S druge strane, primjenom zaglađivanja uvelike se smanjuje broj nepotrebnih bridova te se time olakšava detekcija objekata. Postoje mnoge vrste zaglađivanja, a u ovom slučaju će nam najbolje poslužiti Gaussovo zaglađivanje.

Gaussovo zaglađivanje (Gaussian blur, Gaussian smoothing) je način zaglađivanja slika kod kojeg je filtar koji se primjenjuje Gaussova funkcija (normalna ili Gaussova distribucija). Gaussova funkcija glasi:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

u slučaju da ju promatramo jednodimenzionalno, dok dvodimenzionalna Gaussova funkcija glasi:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2.1 Implementacija konvolucijom kernel

Matematički gledano, primijeniti Gaussov filtar nad slikom jednako je konvoluiranju slike s Gaussovom funkcijom. Gaussova funkcija može se izračunati jednom te spremiti u matricu (koju nazivamo kernelom) nakon čega vršimo konvoluciju te matrice (kernela) sa matricom koja predstavlja sliku. Jedna takva matrica (u ovom slučaju 5x5) bi izgledala ovako:

$$G = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 4 & 8 & 4 & 1 \\ 2 & 8 & 16 & 8 & 2 \\ 1 & 4 & 8 & 4 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

Zaglađena se slika dobiva kao konvolucija te matrice i matrice slike:

*Slika_zamagljena = slika * G*

Programska implementacija diskretne konvolucije obavlja se iteriranjem kroz sve potrebne elemente slike i matrice (kernela) pomoću nekoliko pokazivača te sumiranjem umnožaka piksela s vrijednostima iz Gaussove matrice. Jasniji prikaz dati će pseudokod (u ovom slučaju za kovoluciju sa 3x3 Gaussovim kernelom):

```
za svaki redak slike i {
  inicijaliziraj pokazivaće redaka p_trenutni, p_prethodni, p_sljedeci i
p_izlazni
  za svaki stupac slike j {
    p_izlazni(j) =
   kernel(0,0) * p_prethodni(j-1) + kernel(0,1)*p_prethodni(j) +
kernel(0,2)*p_prethodni(j+1) +
   kernel(1,0) * p_trenutni(j-1) + kernel(1,1)*p_trenutni(j) +
kernel(1,2)*p_trenutni(j+1) +
   kernel(2,0) * p_sljedeci(j-1) + kernel(2,1)*p_sljedeci(j) +
kernel(2,2)*p_sljedeci(j+1) +
  }
}
```

Složenost ovog algoritama je $O(kernel_{širina} \cdot kernel_{visina} \cdot slika_{širina} \cdot slika_{visina})$ što nije najbolje rješenje, te je definitivno moguće postići manju složenost i bolju iskoristivost procesorskih resursa. Način na koji se to postiže je objašnjen u nastavku.

2.2 Nezavisna implementacija

Karakteristika dvodimenzionalne Gaussove distribucije (funkcije) je kružna simetričnost. Također, funkciju je moguće obraditi najprije horizontalno, po svim retcima, te potom vertikalno po svim stupcima. (Dvodimenzionalna Gaussova distribucija se može odvojiti na dvije nezavisne Gaussove distribucije po svakoj od dimenzija). Vrijedi primijetiti da su jednodimenzionalne Gaussove distribucije simetrične. Iskorištavanje ovih pravilnih svojstava Gaussove distribucije (funkcije) bitno utječe na smanjivanje složenosti na:

$$O(kernel_{širina} \cdot slika_{širina} \cdot slika_{visina}) + O(kernel_{visina} \cdot slika_{širina} \cdot slika_{visina})$$

Osim smanjivanja složenosti, velika korist ovakvog pristupa je mogućnost paralelizacije. Korišten je *openmp,* kao ekstenzija prevodiocu, da bi se stvorilo onoliko dretvi koliko procesor ima jezgara, te paralelno izvršavala obrade slike.

U našoj implementaciji definirane su funkcije koje računaju Gaussovo zaglađivanje za širine i visine kernela od 3, 5 i 7 piksela. Svaka funkcija prvo računa horizontalnu transformaciju, a zatim vertikalnu. Kasnije je ispostavljeno da je veličina od 3 piksela definitivno premala za potrebe našeg projekta, 5 piksela jedva zadovoljavajuća, dok se 7 piksela pokazalo kao optimalni odabir.

Konačna je implementacija Gaussovog zamagljivanja takva da koristi kernel visine i širine od 7 piksela, odvojene horizontalne i vertikalne izračune te paralelizaciju uz pomoć openmp-a. Rezultat primjene takvog filtera je prikazan na slici 1.

Slika 2.1 Slika s Gaussovim zaglađivanjem (lijevo) te izvorna slika (desno)

3. Modeliranje pozadine

Model pozadine je slika koja teži što bolje predstaviti izgled raskrižja kada na njemu nema pokretnih objekata. Gradnja takvog modela se temelji na pretpostavci da pojedina fiksna točka najčešće poprima boju same pozadine (ne samo zato što se češće vidi cesta nego pojedino vozilo, već i iz razloga što su vozila međusobno neujednačenih boja pa će najučestalija boja biti upravo ona koja pripada pozadini). Problem se svodi na traženje najučestalije boje za pojedinu točku na slici. Druga mogućnost generiranja takvog modela bi se mogla temeljiti na prosječnoj boji pojedine točke kroz određeni vremenski interval (tzv. *moving average*), no ustanovljeno je da takav model daje lošije rezultate nego model temeljen na najučestalijoj boji.

Graf koji prikazuje učestalost pojedinih boja (ili njezinih komponenti) zove se histogram. U praksi se najčešće crtaju histogrami koji u sebi sadrže statistiku cjelokupne slike, no ovdje je bilo potrebno takav podatak sačuvati za svaku pojedinu točku na slici. Da bi se sačuvao takav podatak potrebna je trodimenzionalna matrica na koju dvije dimenzije otpadaju na širinu i visinu slike, a treća na pojedine boje. Svaka uređena trojka (x,y,boja) sadržava podatak o frekvenciji pojedine boje na položaju (x,y).

Testna video snimka ovog projekta je veličine 576 x 720 a svaka se točka sastoji od RGB komponenti (crvena, zelena i plava) od kojih svaka ima 256 razina. Zapis slike je takav da se, gledano po širini, prvo nalaze prva, druga i treća komponenta prve točke, zatim prva, druga i treća komponenta druge točke, itd. To znači da će širina jednog retka zapravo biti tri puta veća od širine slike. Iz svega toga slijedi da će u ovom slučaju veličina matrice koja sadrži frekvencije pojedinih boja za svaku točku slike biti 576 x 2160 x 256.

Da ne bi svaki put pretraživali najučestaliju boju svake točke, dodana je druga, dvodimenzionalna matrica koja čuva najučestalije vrijednosti. Iteriranjem kroz svaki redak i svaki stupac slike ažurira se vršna vrijednost, ukoliko je promatrana komponenta učestalija od trenutne. Na kraju, da bi se dobila koristiva slika pozadine, potrebno je samo iterirati kroz sve elemente matrice vršnih vrijednosti komponenata te zapisati to unutar jednog *IplImage* elementa.

I ovdje je, kao i u drugim pogodnim dijelovima programa korišten *openmp* kako paralelno vršila obrada na onoliko dretvi koliko procesor ima jezgara

Slika 3.1 Model pozadine nakon 25 obrađenih slika

Slika 3.2 Model pozadine nakon 50 obrađenih slika

Slika 3.3 Konačni model pozadine

Na kraju, pozadinu je moguće generirati u realnom vremenu (tokom izvršavanja) ili ju izračunati zasebno pa onda samo učitati, što zadnja inačica ovog projekta i radi.

4. Stabilizacija slike

Stabilizirati sliku podrazumijeva provesti postupke koji će reducirati efekte neželjenog pomicanja uhvaćenog okvira u odnosu na pozadinu, što je uzrokovano „treskanjem“ kamere. U našem primjeru događaju se neželjena rotacija i translacija okvira u usporedbi s okvirom koji određuje pozadinsku sliku. Do odluke da se provedu postupci stabilizacije došlo se nakon samog izvlačenja pozadine te generiranja prve verzije binarne slike. Tada su primijećeni gore spomenuti nedostaci koji su imali prilično destabilizirajući efekt na binarnu sliku te bi se često dogodilo da se kao objekti prednjeg plana pojavljuju sastavni dijelovi pozadinske slike, poput oznaka pješačkog prijelaza ili grana drveća na rubovima slike. Korak stabilizacije implementiran u našem programu se odvija u trenutku prije izgradnje binarne slike za trenutni okvir, a nakon što je izvučena osnovna pozadina te nakon što je slika zaglađena tako što je provučena kroz Gaussov filter.

Stabilna slika je preduvjet za svaki daljnji proračun te za postupak odvajanja objekata u pokretu od pozadine, tj. poglavito za izvlačenje binarne slike koja prikazuje detektirani prednji plan na crnoj pozadini. Efekti koje želimo izbjeći su detektiranje sastavnih dijelova pozadinske slike kao objekata prednjeg plana. Dodatni neželjeni efekti (i ideje za njihovo uklanjanje), koji će biti pojašnjeni na kraju ovog poglavlja, su dodirivanje/spajanje fizičkih objekata prednjeg plana te višestruki odzivi jednog fizičkog objekta u binarnoj slici.

4.1 Stabilizacija kadra pomoću Harrisovog izračuna uglova na slici

Osnovna ideja sljedećeg postupka je praćenjem određenih karakterističnih značajki slike uhvaćenog okvira doći do razlike između trenutnog okvira i modela pozadine. Postoje različiti algoritmi koji izvlače karakteristične značajke na slici koja se obrađuje; te značajke mogu biti rubovi, uglovi, takozvani patch (mali dio slike koji se ističe bojom i teksturom) itd. Najpogodnija značajka za nas je ugao. Uglovi su „dobra“ značajka za pratiti, za razliku od ruba, budući da ako slikom prolazimo gledajući neki manji okvir putujući rubom nećemo primjećivati velike razlike između konsekventnih okvira pomaknute slike te nam to ne daje nikakvu korisnu informaciju za

posao koji moramo obaviti. Kod uglova se pri pomicanju slike gledajući isti okvir u prethodnoj i trenutnoj slici događa velika varijacija u vrijednostima piksela koji su isto pozicionirani unutar okvira izvučenih iz dvije različite slike.

Ideja implementiranog algoritma stabilizacije je da nakon što pomoću određenih algoritama dođemo do pozicija (koordinata) pronađenih uglova na uhvaćenoj slici, usporedimo pozicije tih istih uglova na pozadinskoj slici, zatim izračunamo prosječni pomak uglova između te dvije slike, po x i y osi, te ovisno o prosječnom pomaku virtualno „pomaknemo“ kameru namještajući regije interesa na slikama tako da se uglovi na trrenutnoj slici i podlozi približno poklope.

Prvi algoritam koji se koristi za pronalazak uglova je tzv. Harrisov detektor uglova. To je matematički operator koji je pri pronalasku uglova invarijantan na utjecaje rotacije, skaliranja slike te promjene u osvjetljenju. Ideja je da se prati intenzitet malog okvira slike te intenzitet malo pomaknutog drugog okvira. Izračunava se razlika dolje navedenom osnovnom formulom te se u slučaju intenziteta većeg od neke zadane dovoljne vrijednosti označava detektirani ugao.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

E – izračunata razlika u intenzitetu. Funkcija pomaka.

u – pomak drugog okvira po x osi.

v – pomak drugog okvira po y osi.

w(x,y) - pozicija promatranog okvira na slici, djeluje kao maska i osigurava da se djeluje na samo taj određeni okvir na slici.

I – intenzitet slike na poziciji (x,y)

Ta relacija prolazi kroz određene postupke pretvorbe, razlika intenziteta u uglatim zagradama razvija se u Taylorov red iz kojeg relevantni ostaju samo prva tri člana te se dobiveni izraz kvadrira i cijela se relacija prebacuje u matrični zapis te dobivamo sljedeću relaciju:

$$E(u,v) \approx [u \quad v]\left(\sum \begin{bmatrix} I_x{}^2 & I_xI_y \\ I_xI_y & I_y{}^2 \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix}$$

4.2 Implementacija

Biblioteka OpenCV sadrži više funkcija i metoda koje implementiraju Harrisov izračun, ali posebno interesantnom se pokazala:

`cvGoodFeaturesToTrack(image, eigImage, tempImage, corners, cornerCount, qualityLevel, minDistance, mask=NULL, blockSize=3, useHarris=0, k=0.04)`

Njena prednost je brzina izvođenja, mogućnost korištenja još jedne funkcije za izračun uglova uz Harrisa, te način zapisivanja pozicije uglova u polje `corners` koji sadrži koordinate x i y zapisane u obliku `CvPoint2D32f` strukture podataka.

Parametri koji se predaju gore navedenoj funkciji su redom: izvorna slika na kojoj se traže uglovi; novostvorena slika jednakih dimenzija kao izvorna u koju se pohrani izračun minimalnih svojstvenih vrijednosti za svaki piksel izvorne slike (služi se za Harrisu alternativni izračun uglova); novostvorena slika jednakih dimenzija kao izvorna koja služi kao pomoćna; polje u koje će se zapisati koordinate uglova; brojač uglova; nivo kvalitete – prag razlike za gore navedenu osnovnu jednadžbu; minimalna udaljenost između 2 značajke koje ćemo prihvatiti kao ugao; maska; veličina okvira; true ili false vrijednost – koristimo true, znači Harrisov algoritam; te parametar za Harrisov algoritam – koristimo zadanu postavku.

Implementirane su metode `ComputeSrc` i `ComputeBack` koje izvode izračun uglova na uhvaćenoj slici te na trenutnoj pozadini. Izračuni su pohranjeni u privatna polja osnovne klase `Background`. Na temelju izračunatih pozicija uglova računaju se udaljenosti između najbližih uglova i prosječna udaljenost uglova pronađenih na pozadini od uglova pronađenih na uhvaćenom kadru.

Pseudokod:

```
// ovaj dio spoji najbliže uglove i njihove udaljenosti po  x i y osi
      za svaki ugao izvora i{
            minDistanca=sirinaPozadine^2 + visinaPozadine^2
            za svaki ugao pozadine j{
                  razlikax = izvor.x-pozadina.x
                  razlikay = razliku izvor.y-pozadina.y
                  a= razlikax^2 + razlikay^2
                  ako je a < minDistanca  distance[i]=(razlikax,razlikay)
                  }
            Izračunaj prosječnu udaljenost po x i y iz distance
      }
```

Izračunata prosječna udaljenost po x i y se koristi u metodama koje vrše određivanje zajedničkih regija interesa slike pozadine i trenutne slike. Prvo se postavlja regija interesa za jednu, a zatim za drugu sliku te se one zatim predaju na daljnju obradu i iz njih se generira binarna slika. Dijelovi slika koji su izvan regije interesa zadanih slika samo se uklanjaju, visina i širina slike pozadine i slike uhvaćenog kadra ostaju jednake u tom procesu. Način na koji se određuje okvir za regije interesa tih dviju slika dan je u nastavku:

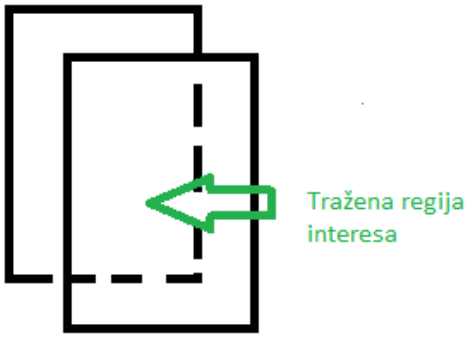
- Okvir je cjelobrojno polje koje sadrži sljedeći skup elemenata

`{pozicijaX,pozicijaY,širina,visina}`

- Postoje 4 slučaja za svaku sliku, ovisno o izračunatim prosječnim pomacima između uglova. Izračunati prosječni pomak može biti pozitivan ili negativan, ako je pomak po x osi pozitivan – to je pomak udesno, inače ulijevo. Ako je pomak po y osi pozitivan – to je pomak prema gore, inače prema dolje.

- Ovisno o izračunatim prosječnim pomacima određuje se dio pozadinske

slike i dio slike kadra koji se poklapa. Odredivši taj dio koji se poklapa vršimo virtualni pomak kamere i stabilizaciju slike uklanjajući negativne efekte neželjene **translacije.**



Slika 4.1 Tražena regija interesa za stabilizaciju

Stabilizacija implementirana ovim algoritmom rješava probleme translacije slike uhvaćenog kadra u odnosu na sliku pozadine, ali ne rješava probleme nastale neželjenom rotacijom. Razlika između binarne slike dobivene bez i sa stabilizacijom je nažalost veoma mala budući da su problemi uglavnom uzrokovani rotacijom kamere tj. glavnog kadra našeg primjernog video zapisa.

4.3 Ostali algoritmi i predložena rješenja za neke probleme

4.3.1 Fazna korelacija

Osnovna ideja ove metode je računanje pomaka (translacija) između dviju slika. Metoda se oslanja na Fourierovu transformaciju i njena svojstva (točnije, na svojstvo pomaka).

Postupak započinje s računanjem Fourierove transformacije na slikama[1] (g_a i g_b) za koje želimo

[1] U teoretskom razmatranju smo pretpostavili da su slike cirkularno-translatirane, drugim riječima pikseli se pri translaciji slike u lijevo pojavljuju na desnoj strani slike kao što se i pri translacijij prema dolje pikseli pojavljuju na vrhu slike i obratno (slika se "vrti u krug")

utvrditi koliki je pomak:

$$G_a = F\{g_a\}$$

$$G_b = F\{g_b\}$$

Zatim se izračuna „*cross-power spectrum"* (* označava kompleksnu konjugaciju):

$$R = \frac{G_a G_b^*}{|G_a G_b^*|}$$

Potom slijedi računanje normalizirane „*cross-correlation"* pomoću inverzne Fourierove transformacije:

$$r = F^{-1}\{R\}$$

Dobivena matrica r sadrži vrijednosti između 0 i 1. Na kraju, da bi odredili pomak, tražimo poziciju „peak-a", tj. poziciju maksimuma u matrici:

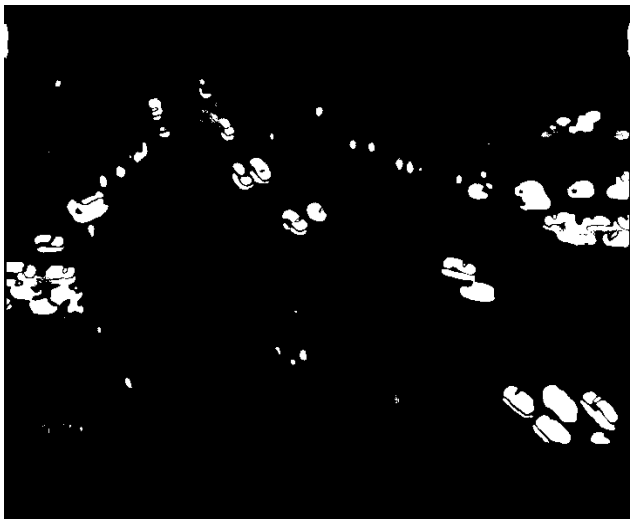
$$(x_{max}, y_{max}) = \arg\max_{(x,y)}\{r\}$$

S obzirom na dobivenu poziciju maksimuma možemo odrediti pomak slike (1 od 4 moguća; lijevo – desno; gore – dolje). Kad saznamo pomak možemo odrediti regiju od interesa, tj. presjek pozadinske slike i trenutne slike, kao i kod implementacije pomoću Harrisa.

5. Odvajanje objekata u pokretu od pozadine

5.1 Binarna slika

Binarna slika je slika koja za svaki piksel sadrži jednu od dvije vrijednosti. Binarna sliku potrebno je stvoriti za odvajanje objekata u pokretu od pozadine. U našem slučaju pozadini je pridijeljena crna boja, a objektima koji se kreću bijela. Funkcija koja stvara binarnu sliku uspoređuje vrijednost svakog piksela trenutne slike sa vrijednošću pripadajućeg piksela pozadine. Ukoliko je razlika tih dviju vrijednosti veća od definirane granice znači da piksel trenutne slike pripada nekom objektu te se vrijednost binarne slike za taj piksel postavlja na 255. U suprotnom, kad je razlika vrijednosti tih piksela manja od definirane granice, pikselu binarne slike pridjeljuje se vrijednost 0 i on predstavlja dio pozadine.



Slika 5.1 Binarna slika

5.2 Morfološka obrada binarne slike – Closing (zatvaranje) i Opening (otvaranje)

Za poboljšanje odziva objekata u pokretu koji su detektirani na binarnoj slici dodatno se radi morfološka obrada slike, vršimo tzv. „**zatvaranje**“ koje se sastoji od dva postupka – dilatacije i erozije. **Prvo se vrši dilatacija, zatim erozija zadane binarne slike.** Tim postupkom dolazi do povezivanja nepovezanih regija odziva koji pripadaju istom objektu prednjeg plana te

redukcije suvišnih piksela na binarnoj slici koja će nakon odrađenog „zatvaranja“ biti mnogo pogodnija za detekciju bridova i daljnju obradu. Nedostaci ovog postupka su neuklanjanje „noise-a“ (odziva koji nastaju malim pomacima kamere, a pripadaju pozadini), te zatvaranje bliskih automobila i pješaka u jednu regiju. Metoda koja rješava problem „noise-a“ je tzv. „**otvaranje“**, koja se, kao i „zatvaranje“ sastoji od dva postupka – dilatacije i erozije. Za razliku od „zatvaranja“, kod „otvaranja“ se **prvo vrši erozija, a zatim dilatacija binarne slike**. S obzirom da smo stavili veći prioritet na detekciji i praćenju automobila (dakle, većih objekata) nego na uklanjanje šumova u slici, odlučili smo iskoristiti „**zatvaranje**“ zbog krajnjeg rezultata koji dobivamo tom metodom, a to je povezivanje regija koji pripadaju istom objektu.

5.2.1 Dilatacija

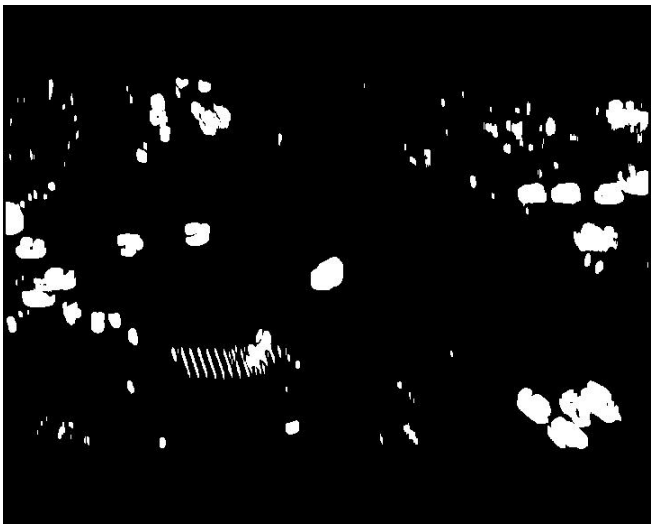
Glavni efekt morfološkog postupka dilatacije je postupno proširenje regija piksela objekata prednjeg plana te na taj način dolazi do spajanja do tada nepovezanih dijelova odziva istog objekta prednjeg plana.

Implementirane su dvije metode koje vrše dilataciju. Prva koristi oblik maske 5x5 matrice, druga koristi 3x3 matricu. Implementacija koristi logičke operacije nad bitovima za promjenu vrijednosti odgovarajućeg piksela.

Pseudokod druge metode:

```
Dilatacija3x ( izvor, dilatirana, maska){  // jednoprolazni algoritam
      Ako je maska == 1
            okvir ={{255, 255, 255}, //255 bijeli piksel, 0 crni
                  {255, 255, 255},
                  {255, 255, 255}};
      Inicijaliziraj pointere za piksele izvorne i  dilatirane slici
      Za svaki redak izvorne slike {
                  Za svaki stupac izvorne slike{
                  Odgovarajući piksel (redak,stupac)
```

```
                    Ako je odgovarajući piksel == „pozadinski“ piksel
                    (vrijednost 0){
                    Postavi masku preko njega, ako se poklapa vrijednost
                    piksela maske s vrijednošću piksela izvorne slike:
                    Onda  dati „pozadinski“ postavi u prednji plan (255)
                    Zapiši odgovarajući piksel u dilatiranu sliku
                    }
}
```



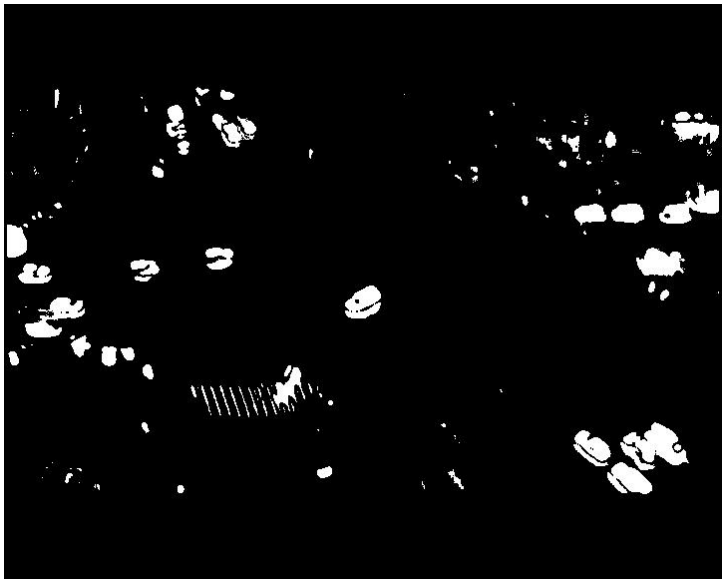
Slika 5.2 Binarna slika nakon dilatacije

5.2.2 Erozija

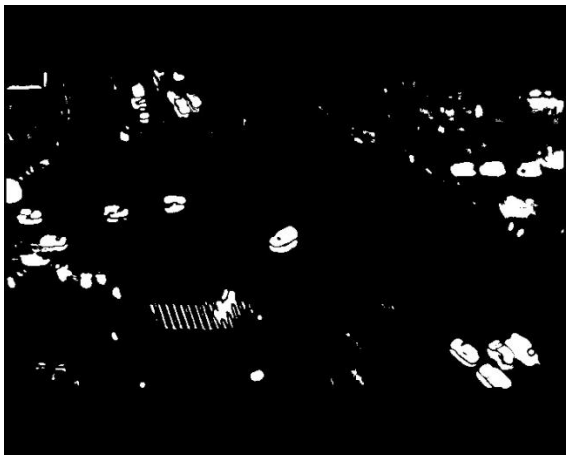
Glavni efekt morfološkog postupka erozije je erodiranje piksela koji se nalaze na granicama binarnih odziva objekata prednjeg plana, smanjuje odziv objekata i povećava razmak između dva različita odziva tako da olakša njihovu detekciju.

Implementirane su, kao kod dilatacije, metode sa 3x3 i 5x5 maskom. Pseudokod erozije je identičan pseudokodu dilatacije osim u dijelovima gdje se pomiče maska i prate pikseli. Maska dilatacije i erozije je jednaka. Implementacija koristi logičke operacije nad bitovima za promjenu vrijednosti odgovarajućeg piksela.

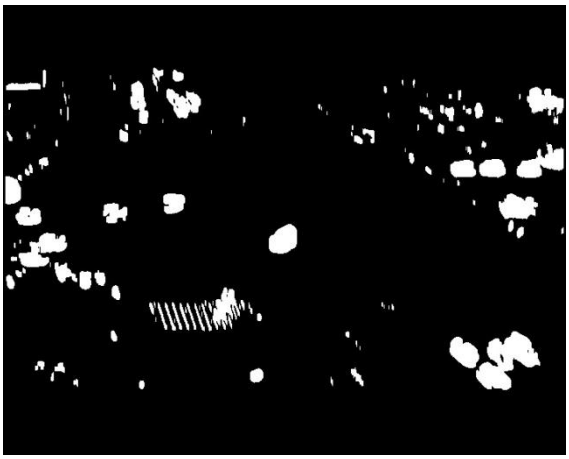
```
Za svaki redak izvorne slike {
                    Za svaki stupac izvorne slike{
                    Odgovarajući piksel (redak,stupac)
                    Ako je odgovarajući piksel == piksel „prednjeg plana"
                    (vrijednost 255){
                    Postavi masku preko njega, ako se poklapa vrijednost
                    piksela maske s vrijednošću piksela izvorne slike:
                    Onda odgovarajući piksel postavi u pozadinu (0)
                    Zapiši odgovarajući piksel u dilatiranu sliku
                    }
```



Slika 5.3 Binarna slika nakon erozije



Slika 5.4 Binarna slika



Slika 5.5 „ Zatvorena“ slika

6. Detekcija bridova

Detekcija bridova uvelike može olakšavati daljnje prepoznavanje objekata na slici. Kroz samu detekciju bridova izdvajaju se dijelovi koji se odvajaju od pozadine i vjerojatno čine zasebne objekte. Prvi pristup detekcije bridova napravljen je kroz sobel operator.

6.1 Sobel operator

Sobel operator predstavlja numeričku aproksimaciju računanja gradijenta slike. Tako će dijelovi slike koji imaju „najbržu“ promjenu vrijednosti imati veću vrijednost od piksela koji su gotovo sličnih vrijednosti susjednim. Kako se i gradijent prikazuje kao derivacije po pojedinim dimenzijama, tako se i sobel operator računa zasebno za vertikalnu os, zasebno za horizontalnu os te se na kraju izračuna intenzitet na temelju oba. Vertikalni i horizontalni sobel se, kao i Gaussovo zaglađivanje, računaju konvolucijom matrice (kernela) i slike. Horizontalni sobel se dobiva na sljedeći način:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I$$

Gdje * označava operator konvolucije, a I matricu koja predstavlja sliku nad kojom primjenjujemo sobel operator. Na analogni način dobivamo i vertikalni sobel:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

Sama izvedba konvolucije slike sa kernelom jednaka je opisanom pseudokodu iz poglavlja 2.1, samo što se ovdje umjesto Gaussovog kernela koriste vertikalni i horizontalni kerneli sobel operatora. Na kraju, nakon što su izračunate vrijednosti za vertikalni i horizontalni sobel, određuje se amplituda gradijenta koristeći sljedeći izraz:

$$G = \sqrt{G_x + G_y}$$

Međutim, kako je korjenovanje relativno skupa operacija, moguće je koristiti sljedeću aproksimaciju uz jednako dobre rezultate:

$$G = |G_x| + |G_y|$$

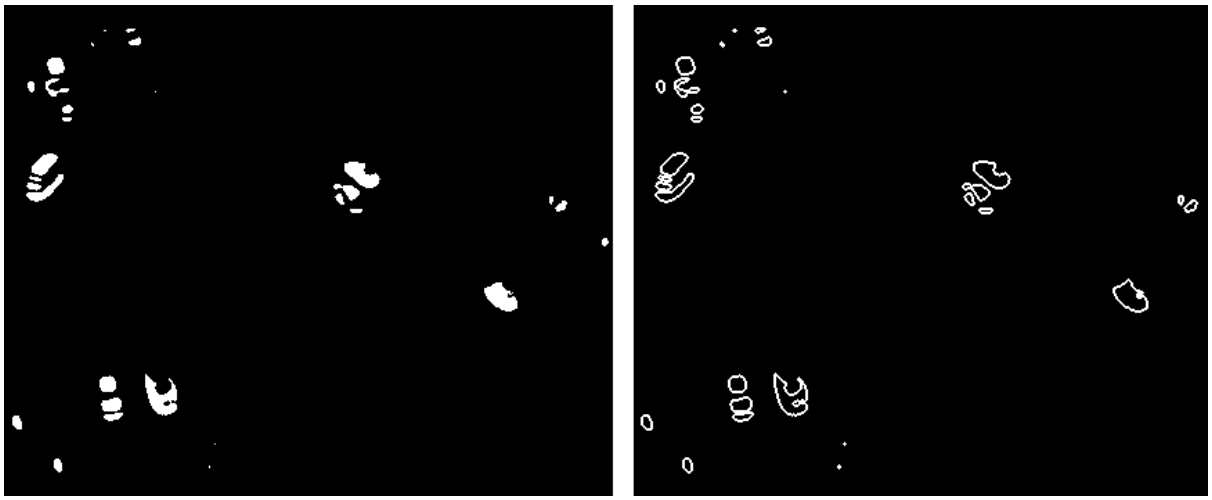
6.2. Binarna detekcija bridova

S obzirom da je slika koju dobivamo uspoređivanjem trenutne slike i modela pozadine zapravo jedino bitna (ili se piksel poklapa sa modelom pozadine ili se razlikuje) moguće je odrediti bridove na jednostavniji i manje rastrošan način od korištenja Sobela. Da bi to učinili, možemo koristiti binarne operacije nad susjednim pikselima. Piksel je dio brida ukoliko je bilo koji njemu susjedni piksel njemu suprotne boje (boje u ovom slučaju mogu biti samo 0 ili 1). Zapis u pseudo kodu izgleda ovako:

```
za svaki redak i {
  za svaki stupac j {
    o(i,j) =
     ( p(i-1,j) xor p(i,j) ) or
     ( p(i,j) xor p(i+1,j) ) or
     ( p(i,j-1) xor p(i,j) ) or
     ( p(i,j) xor p(i,j+1) );
  }
}
```

Po pitanju oznaka: *o(i,j)* označava piksel na odredišnoj slici (gdje se iscrtavaju konture), a *p(i,j)* označava izvorišnu sliku (razlika modela pozadine i tekuće slike). Ovakva implementacija, za razliku od sobel operatora, ne koristi nikakve aritmetičke operacije već čiste binarne

operacije, što ju čini puno bržom i pogodnijom u našem slučaju. Svakako, vrijedi napomenuti, da ovakva funkcija ne bi bila od nikakve koristi na višebitnoj slici (*grayscale* ili *color*). Ovo je pogodnija i brža metoda, samo zato što se radi o jednobitnoj slici. U svakom drugom slučaju sobel, neka njegova optimizacija (kernel se može rastaviti na dva manja vektora) ili druge, slične, metode bi predstavljale bolji odabir.



Slika 6.1 Detekcija bridova (desno) nad jednobitnom slikom razlike trenutne slike i modela pozadine (lijevo)

7. Eksperimentalni rezultati

7.1 Kvalitativni problemi

Kroz razvoj i produkciju rezultata projekta pojavilo se nekoliko problema iz kvalitativne domene. Ovi problemi se glavninom odnose na pogrešno detektiranje objekata koji se kreću raskrižjem. Ishod ovih kvalitativnih grešaka su neprecizni, odnosno netočni rezultati kod praćenja statistika, odnosno broja objekata od interesa. Razvoj i testiranje su pokazali 2 tipa pogreške:

1) lažno pozitivne rezultate, odnosno detektiranje i brojanje nepostojećih objekata

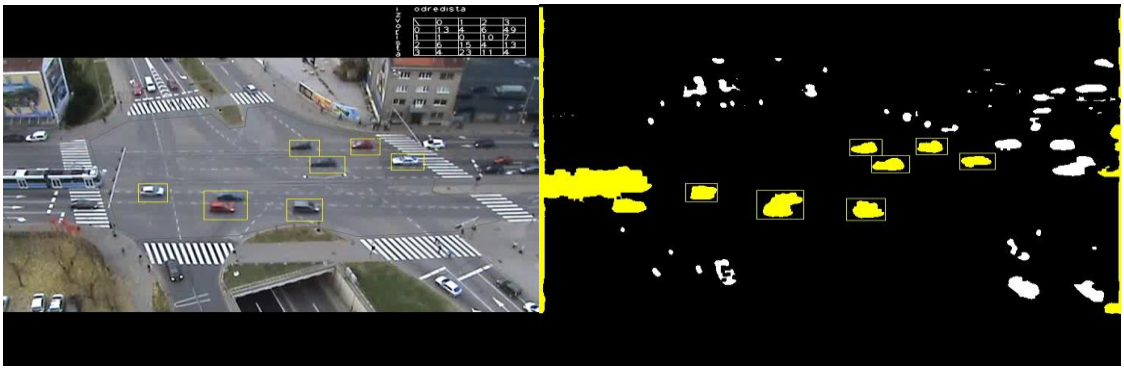
2) lažno negativne rezultate, odnosno odsustvo detektiranja postojećih objekata

Prvi tip pogreške je detektiranje objekata koji nisu vozila koja se kreću raskrižjem. Kroz poboljšanja projekta ovaj tip pogreške je uklonjen uz zadovoljavajuću učinkovitost. Ovaj je problem nastajao kao posljedica načina na koji je implementirano pronalaženje objekata prednjeg plana. Objekti su detektirani kao pikseli koji se razlikuju od pozadine. Zbog toga je nekad pojava gibanja(trešnje) kamere koja pribavlja snimku prepoznata kao gibanje na pozadini. Također, prepoznati su objekti u kretanju koji se nalaze van samog raskrižja. Primjer ovakve greške je na slici 7.1.

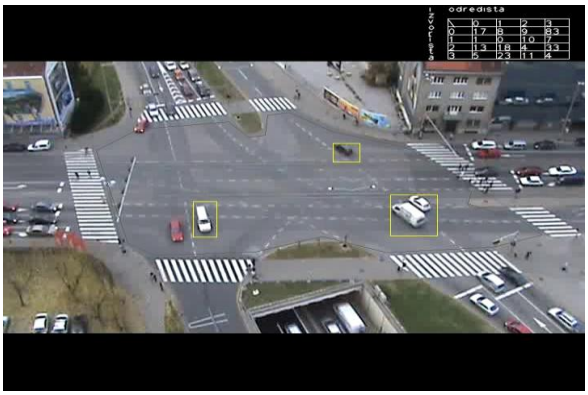
Slika 7.1.

Problem je uklonjen uvođenjem ograničenja u vidu definiranja interesnog područja snimke. Definiranjem istog, na snimci se promatraju i detektiraju samo objekti unutar raskrižja kao područja kretanja vozila.

Drugi tip pogreške se javlja u obliku spajanja bliskih objekata na snimci u jedan objekt pri detektiranju. Ovakav problem je puno teži za eliminirati i pojavljuje se u više formi iz različitih uzroka. Između ostalog ova je greška posljedica pozicije kamere. Snimka se pribavlja pod određenim kutom u odnosu na horizontalu raskrižja i stoga trodimenzionalni objekti prividno zauzimaju veću površinu na pozadini u odnosu na stvarnu površinu koju zauzimaju automobili na cestovnim trakovima. Konkretnije, zbog visine automobila na snimci nisu vidljivi određeni pikseli pozadine. Idealni kut snimanja bio bi okomito na samo središte raskrižja. Sljedeći uzrok je u samoj detekciji. Binarne slike bliskih objekata uslijed krivo detektiranih piksela prednjeg plana dobiju dodirne točke i spajaju se u jedan objekt. Ovakav problem je djelomično uzrokovan i kvalitetom snimke. Analizom snimke utvrđeno je da se ovaj tip pogreške najčešće pojavljuje tijekom dva stanja raskrižja. Prvo stanje, odnosno prva forma pogreške manifestira se u situaciji kada su dva automobila u susjednim prometnim trakovima, i kreću se djelomično paralelno ili vrlo blisko. Takav primjer može se vidjeti na slikama 7.2. i 7.3.

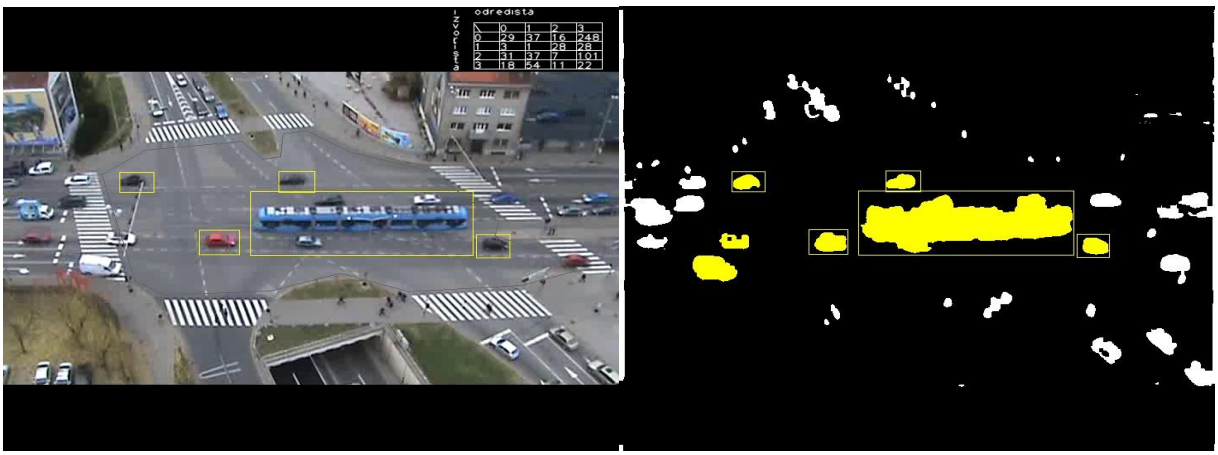


Slika 7.2.



Slika 7.3.

Greška na ovakvom stanju nastaje uslijed oba navedena uzroka. Kroz analizu niza spojenih objekata, pokazalo se da se ova kriva funkcionalnost pojavljuje najčešće kada su dva bliska automobila slične boje ili kada su boje automobila tamnije, odnosno sličniji boji pozadine. Kao što je vidljivo iz slike 7.2, zbog kuta snimanja nije vidljiva pozadina između 2 automobila i dolazi do spajanja u jedan objekt. Druga forma ovog tipa pogreške je pojava tramvaja na raskrižju. Uzroci pogreške su isti kao i u prvoj formi pogreške. Primjer je na slici 7.4.



Slika 7.4.

Osobito problematična je situacija kada raskrižjem prolaze 2 tramvaja, svaki u svojem smjeru. Tada dolazi do većih odstupanja od pravih vrijednosti.

7.2 Kvantitativna svojstva

Cilj konačne implementacije programske izvedbe projekta je da ekstraktira određene podatke o detektiranom prometnom toku iz primjerne video snimke. To je učinjeno izgradnjom tzv. Origin – Destination (OD) matrice. U njoj se prate određeni statistički podaci o samom prometnom toku. Konkretno – prati se broj detektiranih objekata u svakom definiranom odredištu i diferencira ih se po tome iz kojeg izvorišta su „izašli“. Rezultat je matrica podataka čiji redovi predstavljaju izvorišta, stupci odredišta, a vrijednost na (izvorište x, odredište y) poziciji u matrici predstavlja broj detektiranih automobila koji su detektirani da su prošli kroz izvorište x te zatim detektirani kako prolaze kroz odredište y. Izvorišta, odredišta te regija interesa unutar koje se objekti detektiraju određuju se za vrijeme izvođenja programa (upute za to su dane u sljedećem poglavlju), rezultati se pohranjuju posebnim formatom zapisa u tekstualnu datoteku. OD – matrica se prikazuje u gornjem desnom kutu izvorne slike programa nad kojom se vrši detekcija.

Rezultati su obrađivani u 3 različita intervala trajanja primjerne video snimke. Ti intervali su odabrani jer su na njima zanemarive smetnje uzrokovane lošom kvalitetom snimke ili je učestalost prije navedenih kvalitativnih problema niska te program tada daje zadovoljavajuće rezultate. Brojanje objekata je izvršeno i „ručno“ te se ti rezultati uspoređuju s rezultatima detekcije samog programa. Retci – izvorišta; stupci – odredišta.

1. interval - (00:00-00:52)

„Ručna“ obrada	0	1	2	3
	0	1	2	10
1	3	0	0	0
2	4	4	1	0
3	10	0	0	0

Programska obrada	0	1	2	3
0	0	1	2	10
1	2	0	1	0
2	3	4	0	0
3	8	0	0	0

2. interval - (01:06-02:24)

„Ručna“ obrada	0	1	2	3
0	2	2	2	16
1	6	0	2	0
2	8	4	0	0
3	8	2	6	1

Programska Obrada	0	1	2	3
0	10	3	2	10
1	6	0	2	0
2	7	3	0	0
3	4	2	6	1

3. interval - (02:40-05:47)

„Ručna“ obrada	0	1	2	3
0	0	6	12	22
1	6	0	8	17
2	15	9	2	0
3	19	36	20	2

Programska obrada	0	1	2	3
0	0	9	11	19
1	6	0	8	13
2	13	12	2	0
3	18	37	20	2

Zapažena su određena odstupanja između ručno dobivenih rezultata te rezultata koje daje program. Ta odstupanja su izazvana već navedenim kvalitativnim problemima poput detektiranja nepostojećih objekata ili spajanja više detektiranih objekata u jedan.

Drugi problem koji se primjećuje je u samom definiranju područja izvorišta i odredišta. Česta situacija koja se događa kod nekih izvorišta i odredišta je da se pri detekciji nekog automobila u izvorištu dogodi da rub detektiranog automobila dotiče regiju interesa najbližeg definiranog odredišta te program to automatski zapisuje u OD matricu te tako stvara fiktivne prolaze između nekih parova izvorište-odredište te na taj način izaziva odstupanja od stvarne situacije.

Poboljšanje rezultata je moguće izazvati pažljivijim određivanjem regija izvorišta i odredišta te prilagođavanjem tih regija ovisno o situaciji u kojoj se prometni tok nalazi. U cilju efektivnog poboljšanja rezultata moralo bi se poraditi na kvalitativnim problemima koji se pojavljuju.

8. Upute za korištenje

Za potpunu funkcionalnost programske izvedbe projekta, potrebno je definirati određena ograničenja i koristiti dodatne opcije. Da bi promatrali vozila na raskrižju potrebno je najprije definirati tzv. područje interesa (ROI-region of interest). To je područje na snimci na kojem detektiramo objekte. Definira se poligonom kojeg čini određeni broj točaka, odnosno vrhova poligona. Crtanje vrhova omogućeno je na dijelu prozora na kojem je originalna slika. Vrhovi se označavaju lijevim klikom miša uz pritisak na tipku *Control* (Ctrl) na tipkovnici. Program automatski povezuje označene vrhove u poligon koji predstavlja područje interesa. Ukoliko nacrtani poligon nije zadovoljavajući ili ga je potrebno promijeniti, moguće ga je izbrisati desnim klikom miša uz držanje tipke *Control*.

Definiranje odredišta i izvorišta je potrebno radi točnosti matrice križanja. Izvorišta i odredišta se crtaju kao pravokutnici na dijelu prozora sa originalnom snimkom. Izvorišta se crtaju povlačenjem miša uz stisnutu lijevu tipku, dok se odredišta crtaju povlačenjem uz stisnutu desnu tipku. Uz pravokutnik će biti ispisan njegov redni broj. Ukoliko želimo izbrisati neko odredište ili izvorište, potrebno je pronaći točku koja je najbliža pravokutniku koji definira to područje (najjednostavnije odabrati točku unutar pravokutnika). Klikom na lijevu ili desnu tipku miša uz pritisnutu tipku *Shift* na tipkovnici, najbliže područje se briše. Program također omogućuje spremanje trenutnih postavki pozicija područja interesa, izvorišta i odredišta, kao i učitavanje spremljenih postavki. Pritiskom na tipku "S" spremaju se trenutno nacrtane pozicije u tekstualnu datoteku imena "ODmatrix.txt". Ukoliko se pritisne tipka "L" učitavaju se pozicije spremljene u datoteku "ODmatrix.txt". Iscrtavaju se spremljeni pravokutnici i poligon. Postavke pozicije su u datoteci spremljene na sljedeći način: prvi red predstavlja broj izvorišta, zatim po redovima podaci koji definiraju pravokutnik izvorišta (x i y koordinata gornjeg lijevog vrha, širina i visina u pikselima, odvojeni razmakom), slijedi red s ukupnim brojem odredišta i niz redova s podacima koji definiraju pravokutnike odredišta, zatim red s brojem točaka poligona i na kraju niz redova koji definiraju x i y koordinatu tih točaka.

U programu je opcionalno moguće promijeniti vrijednost koja određuje koja razina razlikovnosti diferencira piksele prednjeg plana od piksela pozadine. Inicijalna vrijednost je

postavljena na 20, pritiskom na tipku "+" ta se razina povećava za 1, dok se pritiskom na "-" razina smanjuje za 1. Konačno, prikazivanje snimke se prekida pritiskom na tipku "z".

Tablica 8.1: Kombinacije tipki i njihove akcije

Kombinacija tipki	Akcija
Ctrl + lijeva tipka miša	Označavanje vrhova poligona za područje interesa
Ctrl + desna tipka miša	Brisanje poligona
Lijeva tipka miša + povlačenje	Definiranje izvorišta
Desna tipka miša + povlačenje	Definiranje odredišta
Shift + lijeva ili desna tipka miša	Brisanje najbližeg područja (izvorišta ili odredišta)
S	Spremanje trenutnih područja
L	Učitavanje spremljenih područja
+	Povećavanje vrijednosti razlikovnosti za 1
-	Smanjivanje vrijednosti razlikovnosti za 1
z	Izlazak iz programa

9. Izrada video snimki

Za potrebe prezentacije bilo je potrebno izraditi video uratke pojedinih dijelova snimki u kojima se program ponaša dobro, kao i dijelova u kojima se pojavljuje određena greška. Najprije su ručno popisani interesni intervali, a zatim su oni spremljeni kao JPEG slike na disku. To je učinjeno dodavanjem brojača slika (eng. *frame*) pomoću kojeg bi program znao kada obrađuje interesni interval. Ukoliko je trenutno obrađena slika unutar interesnog intervala, tada se ona sprema u datoteku čije ime sadrži redni broj slike: *frame_nnnn.jpg*. Redni broj je bitan da bi se kasnije mogao rekonstruirati redoslijed prilikom kreiranja video snimke. Spremanje snimke je

učinjeno *cvSaveImage* koja prima dva argumenta: naziv datoteke te pokazivač na trenutnu sliku koja se sprema. Kreiranje video snimke od niza JPEG slika je učinjeno uz pomoć programskog alata *mencoder* i to na sljedeći način:

```
mencoder "mf://*.jpg" -mf fps=15 -o video.avi -ovc x264 -x264encopts bitrate=500
```

Korišten je x264 codec iz razloga što pruža puno veću kvalitetu slike od MPEGv2 koji je prvotno bio korišten. Također, postavljen je manji *framerate* (broj slika po sekundi) kako bi se dobila preglednost detalja koji bi teže bili uočili na bržoj snimci.

10. Literatura

OpenCV 2 Computer Vision Application Programming Cookbook, Robert Laganière, svibanj 2011.

OpenCV 2 Computer Vision Application Programming Cookbook

http://en.wikipedia.org/wiki/Phase_correlation, 12. siječanj 2012.

http://en.wikipedia.org/wiki/Gaussian_blur, 12. siječanj 2012.

http://opencv.willowgarage.com/wiki/, 12. siječanj 2012.

http://en.wikipedia.org/wiki/Corner_detection, 12. siječanj 2012.