MSVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1500

Sustav za gusto označavanje prometnih znakova u video sekvencama

Martin Morava

Zagreb, srpanj 2010

Zahvaljujem se svojem mentoru prof.dr.sc. Zoranu Kalafatiću na stručnom vodstvu te doc.dr.sc. Siniši Šegviću na iskazanoj pomoći tijekom izrade ovog rada.

Sadržaj

1. Uvod

U nazad zadnjih dvadesetak godina razvojem znanosti, ali i sklopovskih komponenata područje računalnog vida doživjelo je znatan napredak. Algoritmi računalnog vida počeli su sve više ulaziti u široku upotrebu, prvenstveno zahvaljujući eksploziji proizvodnje jeftine potrošačke elektronike poput mobitela, kamera, fotoaparata itd., gdje su algoritmi računalnog vida prvenstveno korišteni za zabavu većinom kao pomoć pri detekciji lica i osmijeha na slikama. Osim ovih zabavnih primjena, algoritmi računalnog vida mogu se svakako koristiti i za mnogo ozbiljnije i naprednije stvari nego što je to detekcija osmijeha. Jedna od takvih ozbiljnijih primjena je i primjena algoritama računalnog vida u sustavu za automatizirano detektiranje prometnih znakova u video sekvencama. Naime, na FER-ovom zavodu ZEMRIS, pod vodstvom prof.dr.sc. Zorana Kalafatića i doc.dr.sc. Siniše Šegvića već par godina razvija se sustav za automatiziranu detekciju prometnih znakova u video sekvencama. Cilj takvog sustava je omogućiti kvalitetno vođenje evidencije prometnih znakova na cestama Republike Hrvatske. Zbog velikog područja koje treba nadzirati, bez računala nije moguće kvalitetno voditi evidenciju o ispravnosti prometnih znakova na cestama, te pravovremeno reagirati na sve probleme koji sa znakovima nastaju (oštećenja, krađe...) i stoga se krenulo u razvoj jednog ovakvog sustava. Glavna ideja projekta je proći sve ceste automobilom opremljenim video kamerom, GPS prijemnikom i računalom. Nakon završetka vožnje u računalu imamo spremljenu video sekvencu u kombinaciji sa podacima iz GPS prijemnika te na temelju tih podataka pomoću našeg sustava možemo označiti položaj prometnih znakova koje smo snimili na digitalnoj karti. Nakon proteka nekog vremena, ponovno prođemo cestu, snimimo rutu kojom se vozimo te unesemo nove podatke u sustav, a sustav nam nakon toga automatski na karti pokaže lokacije na kojima se dogodila određena promjena na znakovima (znaka nema, pošaran je ili je obrastao lišćem, itd.). Ovakav sustav bio bi iznimno koristan te zasigurno povećao sigurnost na hrvatskim prometnicama i zbog toga se na njemu intenzivno radi. Sustav se sastoji od više dijelova, a ovaj konkretan rad bavi se poboljšanjem rada dijela koji izvodi samu detekciju znakova u video sekvencama.

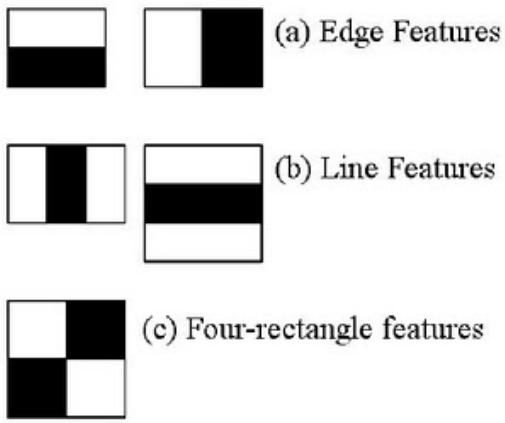
2. Algoritmi računalnog vida koji se koriste

Kao što sam napomenuo u uvodu, sustav za automatsku detekciju prometnih znakova složen je sustav koji se sastoji od više međusobno povezanih komponenti. U ovom dijelu završnog rada opisati ću dva algoritma računalnog vida koja se koriste u sustavu, a to su algoritam *Viola i Jones*[1] koji služi za samu detekciju znakova te algoritam *KLT* [4] koji nam služi za praćenje detektiranog znaka.

2.1. Detekcija prometnih znakova algoritmom Viola i Jones

Paul Viola i Michael J. Jones 2003. godine objavili su članak [1] u kojemu su objasnili svoju novu metodu detekcije vizualnih elemenata na slikama korištenjem kaskade pojačanih Haarovih klasifikatora[6]. Iako je primarna svrha metode isprva bila samo primjena na detekciju lica, zbog svoje implementacije koja koristi algoritme strojnog učenja ova metoda može se primijeniti na bilo koju detekciju vizualnih objekata na slikama; naprimjer za detekciju prometnih znakova, što nam je iznimno važno u sklopu gore spomenutog projekta. S tipičnim parametrima algoritam Viola i Jones radi brzinom od 15 slika po sekundi, no uz pažljivu prilagodbu parametara može se dobiti veća brzina obrade.

2.1.1. Haarovi klasifikatori



Ključ uspješnosti i brzine detekcije algoritma Viola Jones je to što se na slici ne radi provjera piksel po piksel što je metoda izuzetno velike vremenske složenosti, već se koriste tri tipa Haarovih značajki. Ona se računa tako da se odabere određeno područje na slici, to područje se podijeli na podregije i za svaku regiju se izračuna suma vrijednosti piksela u njemu.

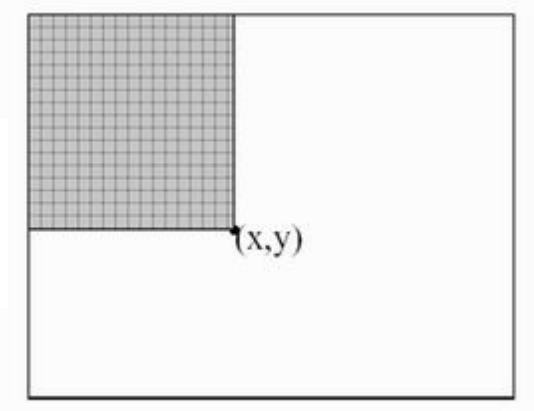
Slika 1. Tri tipa Haarovih klasifikatora

Nakon toga, Haarova značajka dobije se kao razlikavrijednosti određenih područja u regiji.Na prethodnoj slici prikazana su tri tipa Haarovih klasifikatora koji se koriste u algoritmu Viola Jones, a računaju se kao razlika zbroja vrijednosti piksela bijelih i crnih pravokutnika. Iznimka je značajka označena slovom b) na prethodnoj slici koja se računa kao zbroj vrijednosti piksela rubnih pravokutnika od kojeg se oduzme zbroj vrijednosti piksela središnjeg pravokutnika.

Kako bi dodatno smanjili vremensku složenost detekcije, prilikom izračuna sume piksela u određenom pravokutniku Haarove značajke[6] koristimo integralnu sliku. Svaki piksel slike dobije se tako da se izračuna zbroj svih piksela lijevo i iznad tog piksela uključujući i sam piksel.

$$S_i(x,y) = \sum_{x' \le x, y' \le y} S(x',y')$$

Slika 2. Formula za izračun piksela



Slika 3. Prikaz jednog piksela

Binarni Haarov klasifikator *Hi* dobivamo usporedbom vrijednosti značajke *Hi* s pragom *θi*. Ako je značajka veća od praga, tada klasifikator daje pozitivan odgovor *„a“* odnosno negativan odgovor „b“ u obrnutom slučaju. Dakle, Haarov klasifikator može se predstaviti slijedećim izrazom:

$$H_i = \begin{cases} a & \text{ako } h_i \ge \theta_i \\ b & \text{inač e} \end{cases}$$

Međutim, iz oblika Haarovih značajki (slika 1), može se naslutiti da jedan takav Haarov klasifikator ne bi bio izrazito uspješan u detekciji objekata. Takav klasifikator naziva se slabim klasifikatorom. Iz toga slijedi zaključak da je očito nužno kombinirati više značajki kako bi postupak detekcije bio zadovoljavajuć. Međutim, broj značajki za okno od 24x24 piksela iznosi oko 45 000, stoga bi bilo neprimjereno i skupo

ispitati ih sve u postupku detekcije. U tu svrhu koristi se boostanje, odnosno meta-algoritam strojnog učenja koji iz skupa slabih klasifikatora gradi jedan snažni klasifikator. Tijekom procesa detekcije svaki slabi klasifikator će pridonositi pozitivno u slučaju kada detektira traženi objekt, odnosno negativno u obrnutom slučaju. Na temelju doprinosa svih slabih klasifikatora, boostani klasifikator može detektirati objekt u oknu ili javiti da ga nema. Ako *Hi* predstavlja Haarov slabi klasifikator, *tj* prag boostanog klasifikatora dobiven tijekom postupka učenja, *Bj* boostani snažni klasifikator, a *n* broj slabih klasifikatora koji čine snažni klasifikator, boostani klasifikator se može opisati sljedećom formulom:

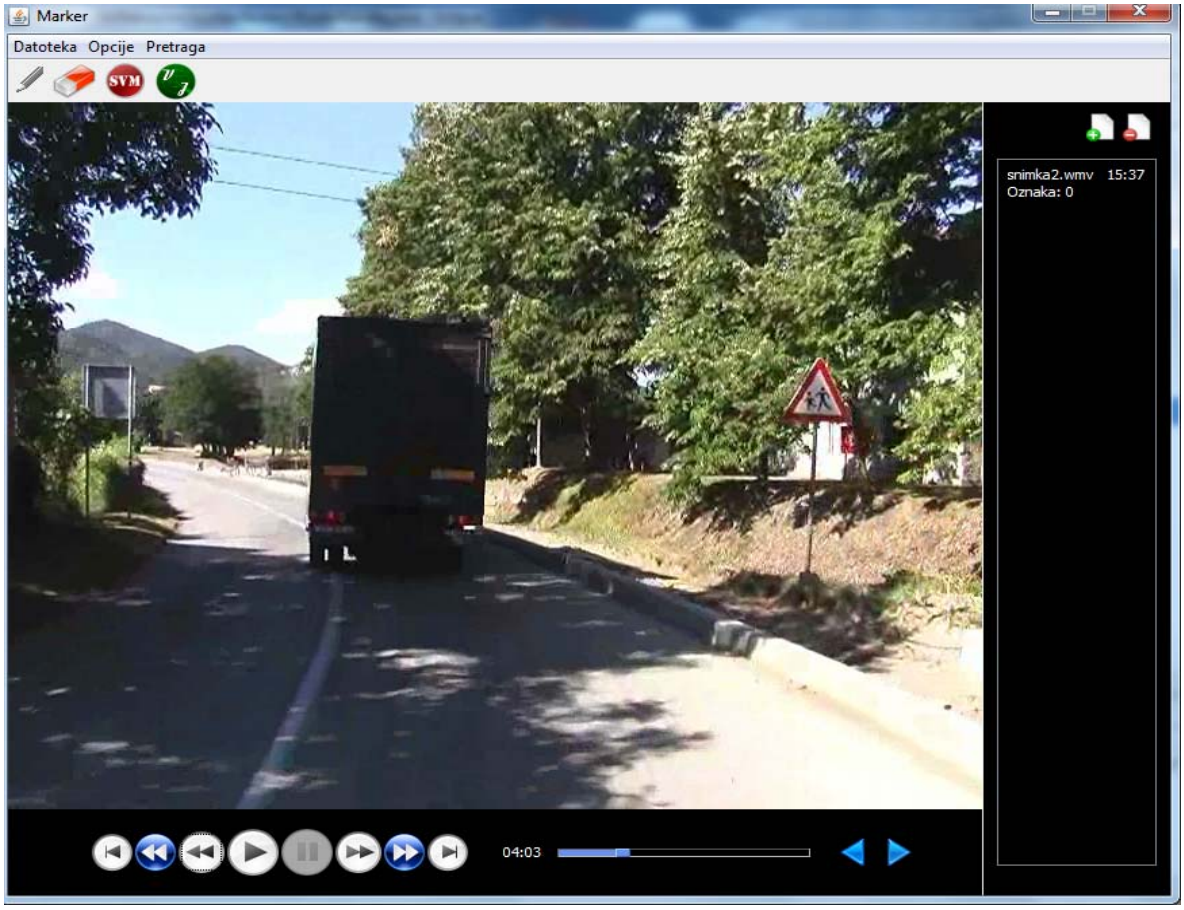
$$B_j = \begin{cases} 1 & \text{ako} \sum_{i=1}^{n} H_{ji} > t_j \\ 0 & \text{inače} \end{cases}$$

2.1.2. Postupak učenja i detekcije

Gore prikazani Haarovi klasifikatori koriste se pri procesu detekcije vizualnog objekta na slici, no postavlja se pitanje s čime se uspoređuju elementi trenutne slike. Odgovor na to pitanje je vrlo jednostavan. Na temelju unaprijed označenih slika vizualnih objekata za koje želimo naučiti naš detekcijski algoritam te unaprijed označenih elemenata slike koji nisu objekti koje želimo pratiti, kreira se kaskada boostanih Haarovih klasifikatora na temelju koje se prosuđuje prilikom postupka detekcije prometnog znaka. Algoritam koristi „okno“ okvir određene veličine koje prolazi kroz sliku i računa Haarove klasifikatore. Nakon što izračuna klasifikator, algoritam uspoređuje izračunati klasifikator sa svim vršnim klasifikatorima kaskade, ukoliko se dogodi određeno poklapanje (recimo detektiran je krug), prolazi se u drugi sloj kaskade i tako rekurzivno dok se ne dođe do samog lista kaskade. Ukoliko se i na listu kaskade dogodi preklapanje, detektirali smo znak, i tada iz opisa znamo točno ime i klasu znaka koji smo pronašli; naprimjer detektiran je znak A22. Postoji gotova implementacija Viola i Jones[1] algoritma u biblioteci OpenCV, međutim za našu primjenu koristimo algoritam posebno prilagođen za naše potrebe koji je u sklopu svog završnog rada izradio Tomislav Babić [3], čiji rad sam i citirao u ovom poglavlju.

2.1.3. Označavanje znakova programom Marker

Program Marker izradila je dipl.ing. Karla Brkić sa FER-ovog zavoda ZEMRIS te skupina studenata vođena studenticom Anom Bulović tijekom predmeta Projekt. Program Marker koristi se kao pomoćni alat za pripremu slika za učenje algoritma Viola Jones[1]. Program nam omogućuje učitavanje i premotavanje videa, označavanje znakova i određivanje klase znakova. Naime, kako bi detektor bio sposoban detektirati znakove u video sekvencama, prvo ga moramo „naučiti“ kako se to radi.



Slika 4. Izgled sučelja programa Marker

Princip je jednostavan. Prvotno sami prođemo kroz nekoliko video sekvenci, ručno odaberemo skup elemenata slike na kojima se nalazi prometni znak tzv. pozitive te skup znakova na kojima se ne nalazi nikakav prometni znak, već se nalazi samo pozadina tzv. negative. Ta dva skupa svaki zasebno dijelimo na skup za učenje kojemu dodijelimo 90-95% svih označenih znakova, a preostale znakove smjestimo u

skup znakova za testiranje detektora. Na temelju ovih unaprijed označenih znakova i pozadina, algoritam detekcije će naučiti kako prepoznati pojavljivanje određenog znaka na slici, pa čak i na video sekvencama koje nismo koristili prilikom izrade skupa za učenje što je zapravo i cilj učenja. Važno je također naglasiti kako skupovi znakova za učenje moraju biti disjunktni sa skupom znakova za testiranje uspješnosti detektora. Intuitativno je jasno da što je broj elemenata u skupu za učenje veći, to algoritam detekcije nakon postupka učenja ima bolja svojstva, odnosno raste mu postotak uspješnosti detekcije znakova. Upravo iz tog razloga potrebno je mnogo vremena provesti sa programom Marker kako bi se označilo na tisuće i tisuće znakova kojima ćemo učiti detektor. Ako promotrimo postupak označavanja vidimo kako se on sastoji od niza koraka:

- zaustaviti video na slici gdje uočimo znak
- dvaput kliknuti na sliku
- odabrati željeni znak iz padajuće liste
- ponovno pokrenuti reprodukciju videa

Ovime uviđamo da je sam postupak označavanja vrlo spor pogotovo ako uzmemo u obzir da je potrebno preko 5000 označenih znakova za kvalitetno učenje detektora i trebalo bi naći načina kako taj postupak ubrzati ili još bolje automatizirati. U ovom radu problemu ubrzanja procesa pripreme skupa znakova za učenje detektora pristupio sam kombinirajući sustav oznaka koji kao svoj izlaz sprema program Marker te algoritam praćenja objekata u video sekvencama KLT[4]. Postupak koji sam razvio i opisao u ovom radu učitati će listu oznaka iz .vse datoteke koju je generirao program Marker, te će na temelju tih oznaka pokrenuti i premotati video na mjesto svake oznake. Na temelju postojeće oznake moj algoritam će inicijalizirati algoritam za praćenje objekata predavši mu oznaku znaka na trenutnoj slici kao značajku koju je potrebno pratiti. Nakon što pustimo reprodukciju videa od mjesta gdje smo inicijalizirali praćenje, algoritam za praćenje kao svoj izlaz vraćat će nam novu poziciju znaka koji pratimo, te je naša zadaća samo taj izlaz zapisati u novu datoteku. Tako smo u mogućnosti postići da na temelju samo jedne ili dvije oznake dobije zapravo 10 do 15 označenih slika ovisno o kvaliteti snimke i poziciji znaka na slici i time znatno povećamo broj slika za učenje detektora i ubrzamo cijeli proces pripreme znakova za učenje te naravno dobijemo kvalitetniji skup za učenje. Međutim, kasnije

u radu vidjet ćemo da ova dva iako naizgled disjunktna sustava ipak nisu potpuno odvojena, jer uspješnost programa za gusto označavanje znakova u video sekvencama jako ovisi o načinu na koji smo označili pojedine znakove, a to je uglavnom uvjetovano algoritmom za praćenje koji koristimo. Naime, u praksi se pokazalo da ovisno o pozadini i okružju u kojemu se prometni znak nalazi nije svejedno kako ćemo označiti znak jer ukoliko loše označimo znak, algoritam praćenja ili uopće neće naći značajke za praćenje ili će pratiti pozadinu, umjesto onoga što mi želimo pratiti, a to je sam prometni znak.

2.2. Algoritam za praćenje Kanade-Lucas-Tomasi

U prethodnim odlomcima objasnio sam princip rada algoritma detekcije Viola i Jones[1] te načine na koji se provodi priprema samog detekora. Međutim, za ispravan rad programa za zgušnjavanje zapisa neophodno je potrebna i implementacija nekog algoritma za praćenje objekata na video sekvencama. Ja sam se odlučio za primjenu algoritma KLT koji su Carlo Tomasi i Takeu Kanade objasnili u svom radu 1991. godine.[4]

2.2.1. Što je to dobra značajka?

Glavni problem s kojim se suočavamo je način na koji možemo u slici odrediti dobre značajke koje možemo pratiti te kako ih zapravo pratiti iz slike u sliku. Tomasi i Kanade svoju metodu za praćenje baziraju na metodi koju su predložili Lucas i Kanade 1981.[5] godine koja se temelji na usporedbi dvije uzastopne slike. Njihov pristup temelji se na označavanju malog okvira unutar slike te na minimiziranju sume razlike u svjetlini piksela u prošlom i trenutnom okviru. Upravo zahvaljujući malom pomaku označenog okvira usporedbu možemo izvršiti translatirajući novu sliku kako bi se poklopila sa prethodnom slikom, te uz to možemo intenzitet svjetline piksela u novoj slici zapisati slično zapisu u prvoj slici te dobiti rezidualni izraz koji linearno ovisi o vektoru pomaka elementa slike. Koristeći ove dvije aproksimacije moguće je napisati 2x2 matricu, čija je nepoznanica upravo vektor pomaka praćenog elementa. Pitanje koje ostaje neodgovoreno je kako zapravo odabrati dobar uzorak na slici koji se može kvalitetno pratiti odnosno koji će se pokazati kao dobra karakteristika za

praćenje. Mnogi pristupi ovom problemu temelje se na *a priori* vrijednosti kojom se računa je li značajka „dobra“. Postoje razni pristupi ovom problemu. Tako su Moravec i Thorpe predložili da se kao dobre značajke uzimaju dijelovi slike koji imaju veliki iznos standardne devijacije u svojim histogramima; Marr, Poggio, and Ullman preferiraju križanje intenziteta slike sa Laplacijanom, dok su Kitchen, Rosenfeld, Dreschler, and Nagel predložili kako je najbolje pratiti kutove na slici temeljeći se na prvoj i drugoj derivaciji intenziteta slike. Bilo kako bilo, Tomassi i Kanade u svom radu definirali su kao dobrom značajkom onu značajku koja se može dobro pratiti ne ograničavajući se na nijednu metodu posebno.

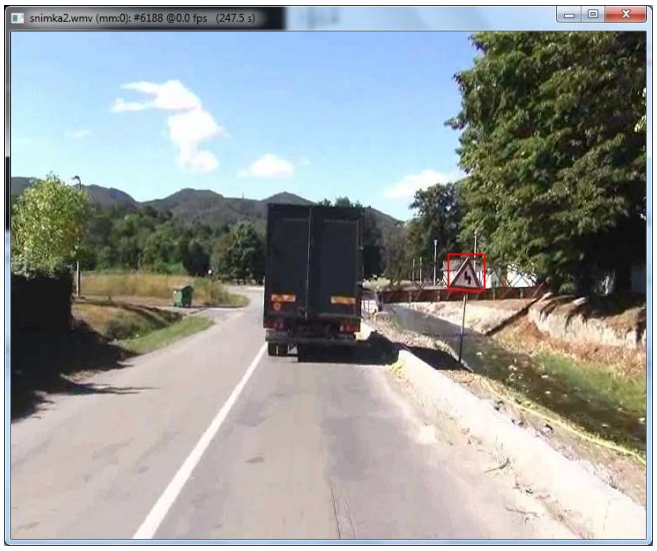
2.2.2. Praćenje dobre značajke

Kako se pomiče kamera, tako se intenzitet uzoraka na slici mijenja na kompleksan način. Općenito bilo koja funkcija od tri varijable I(x,y,t) gdje su diskretne varijable x i y prostorne varijable, a t predstavlja varijablu vremena može prikazati pokretnu sliku. Srećom, slike koje su uslikane u vrlo kratkom vremenskom intervalu uobičajeno su snažno povezane, iz razloga što prikazuju istu scenu koja je samo malo pomaknuta u odnosu na prethodni položaj. Pomak dijela slike opisanog funkcijom I(x,y,t) možemo zapisati kao:

$$I(x, y, t \mid \tau) = I(x - \varepsilon, y - \rho, t)$$

Prirodnim rječnikom to znači da sliku snimljenu u trenutku T+ τ možemo dobiti tako da svaku točku slike translatiramo za određeni iznos.Količina pokreta d =() naziva se pomak slike između vremena t te je t + τ općenita funkcija varijabla x, y i t. Svejedno čak i kod statičnih slika pod konstantnim osvjetljenjem, gornje svojstvo je često povrijeđeno. Naime, točke na slici ne rade samo pomake, već nestaju i pojavljuju se, također svjetlina određenih točaka mijenja se s vremenom ovisno o kutu pod kojim pada svjetlo na tu točku. Bez obzira na ovu činjenicu, gornja formula u većini slučajeva vrijedi kada radimo sa relativno malim prozorima unutar slike koju promatramo uz uvjet da je dio te slike dovoljno udaljen od samog ruba slike, čime se izbjegava nestanak dijelova promatranog prozora izvan slike.

Vrlo važan problem koji se javlja prilikom izračuna distance d pri pomaku iz jedne slike u drugu je taj što nije moguće pratiti pomak samo jednog pojedinog piksela, osim ukoliko taj piksel izrazito odudara po svojoj svjetlini od piksela koji ga okružuju. Dapače, moguće je da piksel potpuno i promjeni boju zbog utjecaja šuma u slici te kao posljedica ovih činjenica većinom je nemoguće odrediti pomak samo jednog određenog piksela temeljeći se samo na lokalnim informacijama. Zbog ovih problema KLT[4] algoritam ne prati samo jedan piksel nego prozor piksela.



Slika 5. Primjer praćenog prozora unutar slike

Međutim, ovisno o slici, određena područja mogu se kretati drugačijim razinama; naprimjer kada na desnoj slici vidimo označeni trokutasti znak. S lijeve strane znaka većinu označenog područja prekriva slika kuće daleko u pozadini, dok je s desne strane zastupljeno granje i lišće od drveta neposredno iza znaka. Možemo primjetiti da s obzirom na udaljenost objekta koji se nalazi u okviru, lijeva strana i desna strana našeg označenog prozora neće se kretati na jednak način odnosno njihove brzine kretanja biti će različite. Dolazimo do problema na koji način biti siguran da pratimo pravi prozor ukoliko se njegovi dijelovi neprestano mijenjanju, te kako iz skupa različitih vektora pomaka dobiti jedinstven vektor pomaka za cijeli prozor. Tomasi i Kanade prvi su problem riješili tako što su usporedili sadržaj prozora u dvjema slikama te ukoliko se sadržaj prozora previše promjenio odbacili su prozor i prestali ga pratiti. Drugi problem u principu se može riješiti tako da pomake ne opisujemo običnim vektorima translacije, nego da ih opisujemo kompleksnijim matricama afinih transformacija. U tom slučaju različite vektore brzina možemo pridjeliti različitim dijelovima u prozoru.

Kao što vidimo iz gornjeg opisa algoritam za praćenje nije savršen i ima svoja ograničenja na koja se treba paziti prilikom izrade skupa oznaka znakova na temelju kojega će algoritam za zgušnjavanje zapisa generirati nove oznake znakova, jer uspješnost algoritma zgušnjavanja zapisa neposredno ovisi o algoritmu za praćenje

koji koristimo. Ukoliko početna oznaka znaka nije pažljivo označena, tada ta oznaka vjerojatno neće biti prihvaćena kao dobra značajka te samim time znak neće biti praćen niti će biti moguće generirati zapis novih oznaka na temelju prethodne loše oznake.

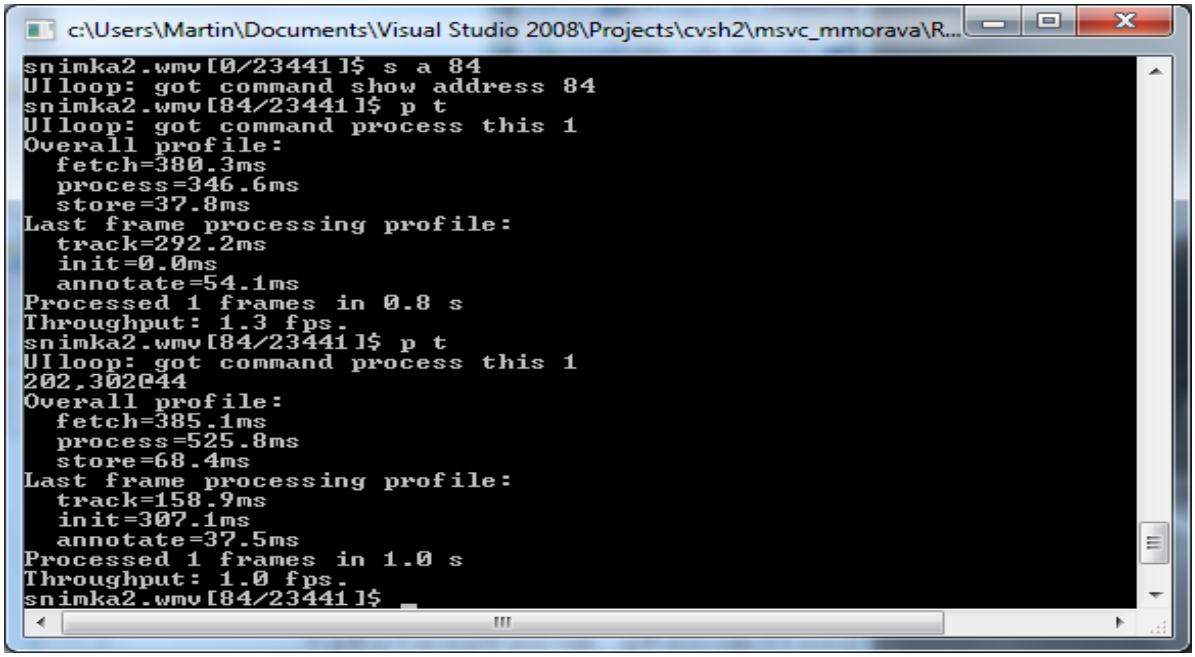
Ovi problemi mogu se riješiti pažljivim odabirom načina na koji će se znakovi označavati tako kod trokutastih znakova možemo iskoristiti činjenicu da ti znakovi uglavnom sadrže bitne značajke u samoj sredini znaka (jelen, strelica itd.) te je u tom slučaju bolje označiti samo taj središnji dio znaka, umjesto cijelog znaka jer ćemo tako povećati izglede da će znak biti kvalitetno praćen, a uz to izbjeći ćemo probleme kao što su praćenje pozadine umjesto samog znaka. Kod okruglih znakova princip označavanja je sličan samo što je prozor označavanja veći pošto se za razliku od trokutastih u okruglim znakovima većinom nalaze detalji koji su veći i jednako udaljeni od rubova samoga znaka. Kod nekih drugih tipova znakova kao što su primjerice table, znak STOP ili znak ceste s prednosti prolaska stvar nije tako jednostavna i potrebno je uzeti u obzir kontekst u kojemu se znak nalazi kako bi se pametnim označavanjem omogućio dobar uzorak za skup znakova za učenje, te kako bi odabrani uzorak mogao biti praćen kao dobra karakteristika za praćenje.

3. Implementacija algoritma za gusto označavanje znakova

Kao što sam napomenuo o uvodu, označavanje znakova u video sekvencama neophodan je postupak za izradu kvalitetnog skupa znakova za učenje algoritama detekcije. Proporcionalno sa njegovom važnošću ovaj postupak je vrlo vremenski ali i psihički zahtjevan jer zahtjeva visoku koncentraciju kroz veoma dug period kako bi se napravio što kvalitetniji skup za učenje. Upravo iz ovih razloga razvila se potreba za razvojem algoritma za gusto označavanje znakova u video sekvencama kako bi se proces označavanja što više automatizirao i kako bi se minimiziralo vrijeme koje je potrebno da čovjek provede pred računalom ručno označavajući znakove. Ideja i implementacija ovog algoritma opisani su u ovom poglavlju.

3.1. Ljuska cvsh2

Ljuska cvsh2 dinamična je i prilagodljiva okolina razvijena na zavodu ZEMRIS i posebno prilagođena isprobavanju algoritama računalnog vida. Također sama ljuska već ima gotove implementacije mnogih algoritama računalnog vida (Viola i Jones, KLT...) kao i mnoštvo primjera koji te algoritme koriste te sam iz tih razloga odlučio iskoristiti upravo nju kao okolinu pod kojom ću razviti algoritam za gusto označavanje znakova. Ljuska cvsh2 omogućuje rad u takozvanom interaktivnom modu u kojemu predajemo naredbe algoritmima računalnog vida pomoću komandno-linijskih naredbi, a rezultati su nam odmah vidljivi u odvojenom prozoru, ili možemo koristiti neinteraktivni način rada kojemu unaprijed odredimo video sekvencu koju treba obraditi, te algoritam kojim će se obrada vršiti, a potom će sama ljuska prolaziti kroz cijelu video sekvencu od početka do kraja i nad svakom slikom pozivati unaprijed određeni algoritam računalnog vida.



Slika 6. Primjer pokretanja cvsh2 ljuske

3.2. Ideja

Sama ideja za implementaciju sustava za gusto označavanje znakova temelji se na ulaznim podacima koje imamo na raspolaganju, a to su zapisi već unaprijed rijetko označenih znakova u video sekvencama. Naime, sustav je vođen mišlju da ukoliko već imamo gotove oznake prometnih znakova na pojedinim slikama u video sekvenci te uz to imamo algoritme računalnog vida koji nam omogućavaju praćenje objekata kroz video, da kombinacijom tih dvaju elemenata možemo automatski generirati gušći zapis oznaka znakova temeljen na već unaprijed postojećem rijetkom zapisu. Glavna ideja je vrlo jednostavna. Potrebno je:

- parsirati listu znakova iz .vse datoteke
- za svaki znak premotati video na sliku u kojoj je taj znak označen
- inicijalizirati algoritam praćenja te pokrenuti praćenje
- prikupiti svaki izlaz iz algoritma za praćenje i spremiti ga u novu datoteku
- ponavljati postupak za svaki znak pronađen u prvotnoj .vse datoteci

Na kraju ovog postupka imat ćemo dvije .vse datoteke. Prva datoteka biti će orginalna .vse datoteka sa rijetkim zapisom oznaka prometnih znakova koju je

generirao program Marker, dok će druga datoteka biti .vse datoteka sa zgusnutim zapisom istih tih znakova koju generira algoritam za gusto označavanje znakova.

3.3. Implementacija

Nakon što smo objasnili ideju na kojoj se sam sustav temelji, potrebno je objasniti samu implementaciju. Prvi korak je naravno samo konfiguriranje algoritma. Kako bismo konfigurirali naš algoritam za gusto označavanje znakova, potrebno mu je kao parametar predati putanju do datoteke u kojoj se nalazi skup prethodno rijetko označenih znakova. Za ovu primjenu iskoristiti ćemo mogućnosti ljuske cvsh2 i postaviti opciju –c na vrijednost putanje te naše .vse datoteke. Primjer poziva ljuske cvsh2 koji ispravno konfigurira i pokreće algoritam je sljedeći:

cvsh2 -sf=D:\apps\snimka2.wmv -a=mm -c=d:\apps\snimka2.vse

Ovaj poziv će pokrenuti ljusku cvsh2, predati joj putanju do video sekvence snimka2.wmv te odrediti tu video sekvencu kao objekt nad kojim će raditi algoritmi računalnog vida, kreirati će se instanca algoritma za gusto označavanje znakova (mm) te će se tom algoritmu kao konfiguracijski parametar predati putanja do odgovarajuće .vse datoteke koja pripada videu koji smo odredili kao glavni objekt. Nadalje, preostaje nam iz predane .vse datoteke parsirati listu oznaka, te spremiti pročitanu listu oznaka u povezanu listu kako bi bili u mogućnosti na temelju tih oznaka generirati nove. Jedan redak oznaka zapisuje se u sljedećem formatu:

[Broj_slike]:sifra_znaka@(koordinate znaka)

Primjer jednog takvog zapisa je sljedeći zapis:

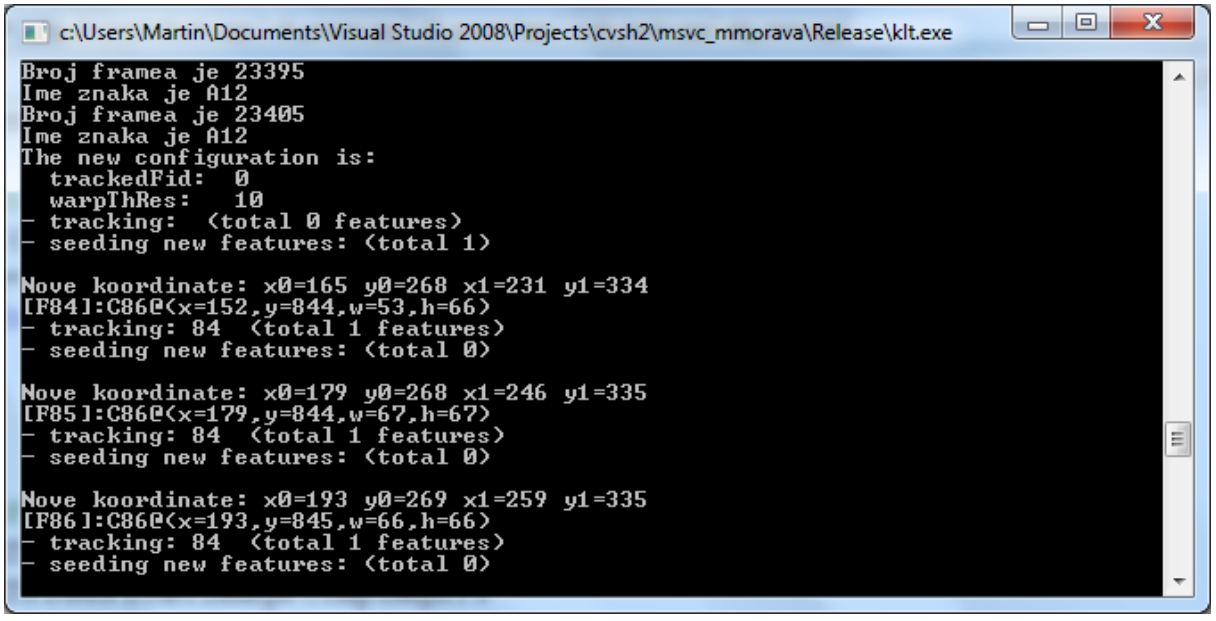
[F3104]:A11@(x=443,y=254,w=32,h=32)

U ovom zapisu F3104 označava da se znak nalazi na slici broj 3104 u video sekvenci, oznaka A11 označava da se radi o trokustastom znaku (A), broj 11, čiji je donji lijevi ugao na koordinatama x=443 i y=254 te da su visina i širina znaka jednake 32 piksela. Na temelju pročitanih oznaka algoritam za gusto označavanje znakova

kreirati će povezanu listu objekata tipa znak, pri čemu je objekt znak zapravo C stuktura deklarirana kao:

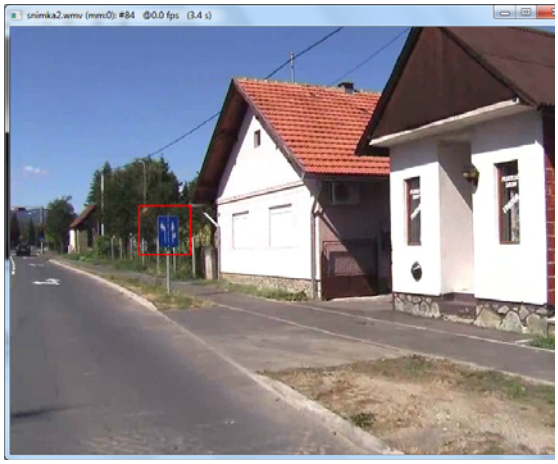
```
typedef struct
{
  coordinates sign_coor;/*Top left coordinates of a sign on a
picure*/
  int frame_number;/*Frame in wich sign is marked*/
  string sign_name;
} traffic_sign;

typedef struct
{
  int x;
  int y;
  int h;
  int w;
} coordinates;
```

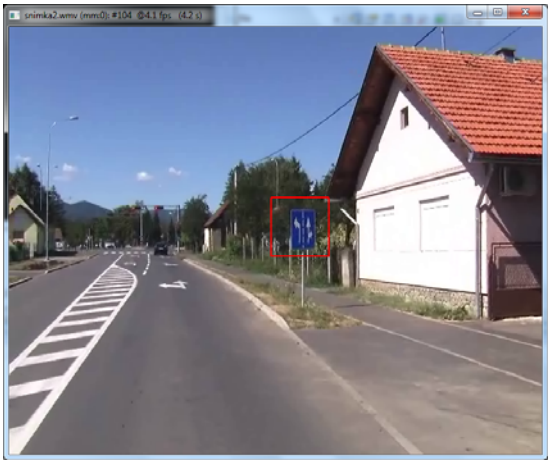
Nakon uspješno provedene konfiguracije, algoritam je spreman za rad. Pošto je ljuska cvsh2 pokrenuta bez opcije –i to znači da će ljuska raditi u neinteraktivnom modu. Kao što sam napomenuo u uvodu ovog poglavlja, kod neinteraktivnog moda ljuska cvsh2 sama pokreće reprodukciju video zapisa koji joj je predan pomoću opcije –sf te svaku sliku tako pokrenutog video zapisa šalje na obradu algoritmu računalnog vida kojim je ljuska inicijalizirana, a to je u ovom slučaju naš algoritam za gusto označavanje znakova alg_mm.



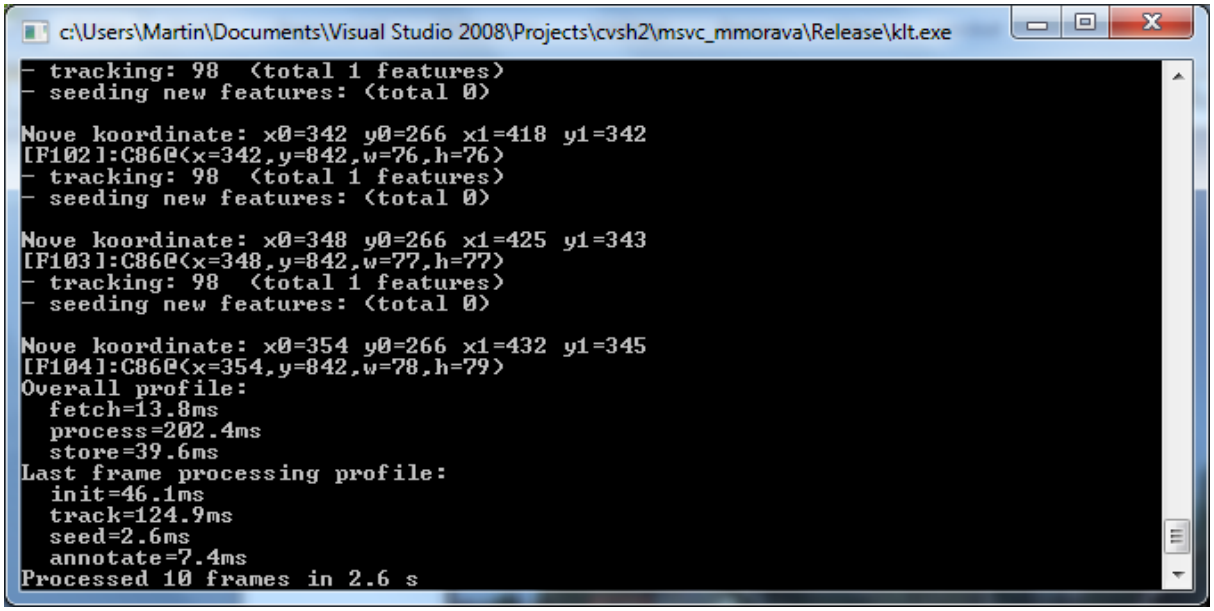
Slika 7. Primjer rada algoritma

Kao što je vidljivo na slici iznad, ljuska će algoritmu prosljeđivati sliku po sliku pokrenutog videa. Nakon što ljuska pozove funkciju process() i algoritmu preda trenutnu sliku te redni broj slike unutar videa, algoritam će proći kroz povezanu listu znakova generiranu prilikom parsiranja .vse datoteke i ukoliko ustanovi da u povezanoj listi postoji spremljena oznaka znaka koji se nalazi na slici koju trenutno obrađujemo, algoritam će inicijalizirati novu značajku za praćenje algoritma KLT[4] te pokrenuti praćenje. Nakon što je ispravno inicijaliziran algoritam KLT[4] će pratiti značajku dokle god je ona „dobra“ , odnosno dokle god ju se može pratiti, te će nam kao svoj izlaz vraćati koordinatu donjeg lijevog ugla pomaknute značajke. Ovime ćemo na temelju samo jedne oznake moći dobiti i do deset novih oznaka u svim slikama koje slijede nakon one jedne slike u kojoj je znak prvotno bio označen. Jasno je da ćemo ovim postupkom dobiti višestruko više zapisa u .vse datoteci te ćemo samim time imati višestruko veći skup znakova za učenje algoritma Viola i Jones.
Primjer rada algoritma dan je na sljedećim slikama:





Slika 7. i 8. Primjer rada algoritma na slikama iz videa



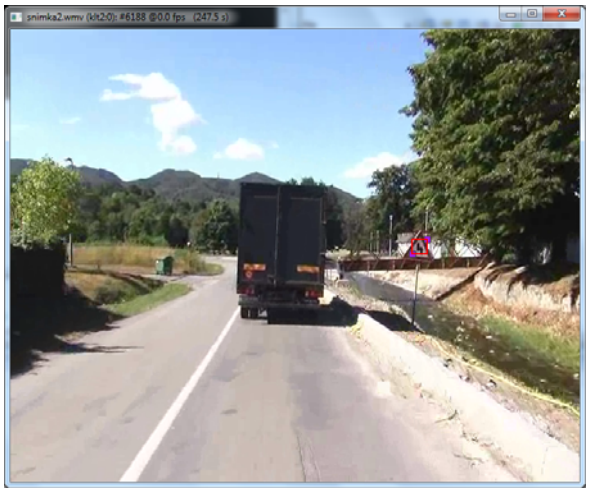
Slika 9. Primjer rada algoritma (2)

Kao što na gornjim slikama možemo primjetiti, na temelju samo jedne oznake na slici broj 7, generirali smo novih 20 oznaka znaka C86 i time višestruko povećali skup označenih znakova koji se koristi za učenje detektora Viola i Jones. Međutim, gornje slike stavljene su s razlogom, a taj je da zapravo ovakav oblik praćenja i nije poželjan jer na gornjim slikama algoritam prati zapravo pozadinu, a ne sam znak što je nepoželjno. Naime, zbog oblika i izgleda samog znaka C86 koji je plava tabla sa dvije velike bijele strelice na sebi algoritam KLT[4] prave značajke za praćenje pronalazi zapravo u pozadini iza znaka koja se giba drugačije nego sam znak, te naš algoritam zapravo prati pozadinu a ne prati znak što je loše i to je jedan od problema s kojim se suočava i algoritam za gusto označavanje znakova, jer kvaliteta njegova rada jako ovisi o svojstvima algoritma za praćenje koji se koristi.

3.4. Problemi s kojima se algoritam susreće te mogućnosti poboljšanja

Ranije u tekstu opisao sam ograničenja s kojima se susreće algoritam za pračenje KLT[4], te načine na koji se ti problemi rješavaju i posljedice koje iz toga proizlaze. Zapravo jedno od glavnih ograničenja moje implementacije algoritma za gusto označavanje znakova je njegova ovisnost prvo o skupu već unaprijed označenih znakova, te načina na koji su ti znakovi označeni, a potom i o samom algoritmu za praćenje koji se koristi. Korisnik koji označava znakove ručno pomoću

programa Marker možda i nije svjestan ograničenja koja sa sobom donosi algoritam KLT[4], ili je krivo uvjeren da je bolje označavati veću površinu u okolini znaka kako bi se taj znak mogao kasnije bolje detektirati što je krivo, a ta se greška potom automatski prenosi u moj algoritam. Upravo vodeći se ovim problemom, parsiranje u algoritmu sam kreirao na takav način da se za svaki znak točno poznaje tip znaka koji je označen, x i y korodinate njegovog doljnjeg lijevog kuta te visina i širina. Poznavajući ove parametre možemo slobodno eksperimentirati i napisati naš algoritam za zgušnjavanje znakova na taj način da recimo za trokutaste znakove oznake pročitane iz .vse datoteke skalira na takav način da se ozačava zapravo samo sredina našega znaka, za okrugle znakove pak oznake možemo skalirati na način da se označi površina znaka tik do stvarnog ruba znaka itd. Ovakvim prilagodbama i posebnim načinima označavanja ovisno o tipu znaka maksimizirao bi se učinak algoritma za gusto označavanje znakova.



Slika 10. Primjer dobro označene značajke u trokutastom znaku

4. Zaključak

Vođenje kvalitetne evidencije prometnih znakova na prometnicama od velike je važnosti za sigurnost cestovnog prometa svake zemlje. Upravo iz tih razloga razvijaju se sustavi koji bi taj posao što više ubrzali i olakšali kroz automatizaciju cijelog procesa. U ovom radu prikazao sam ograničenja koja sa sobom donose algoritmi računalnog vida koji se koriste u takvim sustavima, opisao sam same algoritme i predložio načine kako eliminirati ta ista ograničenja. Opisao sam ideju i implementaciju dijela sustava koji služi kao pomoć u izgradnji kvalitetnijeg skupa znakova za učenje algoritama detekcije, i time automatski pomaže kvaliteti cijelog sustava. U radu je navedena glavna ideja kako je zamišljeno da sustav funkcionira, prikazana je konkretna implementacija sustava te su navedena postojeća ograničenja i predloženi načini kako bi se ta ograničenja u budućnosti zaobišla i time se povećala učinkovitost cijeloga sustava.

5. Literatura

[1] Viola P., Jones M., Robust Real-Time Face Detection, International Journal of Computer Vision 57(2), 137–154,http://lear.inrialpes.fr/people/triggs/student/vj/viola-ijcv04.pdf, 2004.

[2] Čuljak, M.: Primjena strojno naučenih klasifikatora u računalnom vidu, Seminar, Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[3] Babić, T.:Programska implementacija pronalaženja objekata korištenjem kaskadom boostanih Haarovih klasifikatora, Završni rad,Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[4] Tamasi C., Kanadeu T., Detection and tracking of point features,Techical report CMU-CS-91-132,1991

[5] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.

[6] Brkić, K., Pinz, A., Šegvić, S.: Traffic sign detection as a component of an automated traffic infrastructure inventory system, OAGM'09, Stainz, 2009.

[7] Haar-like features – Wikipedia, http://en.wikipedia.org/wiki/Haar-like_features, 29. lipnja 2010.

[8] Robust real time object detection – Wikipedia, http://en.wikipedia.org/wiki/Robust_real-time_object_detection, 12. prosinca 2007.

Sustav za gusto označavanje prometnih znakova u video sekvencama

Sažetak

U ovom radu objašnjen je princip djelovanja i način na koji radi program za zgušnjavanje zapisa označenih znakova u video sekvencama. U uvodnom dijelu dan je osvrt na širi kontekst projekta čiji dio je i ovaj rad kako bi se shvatila sama potreba za izradom jednog ovakvog programa te su objašnjeni razlozi zašto se projekt uopće provodi. U trećem poglavlju opisani su algoritmi računalnog vida koji se u projektu koriste te razlozi zašto su nam ti algoritmi potrebni. U četvrtom poglavlju opisujem svoju implementaciju algoritma za zgušnjavanje zapisa prometnih znakova, principe na kojima se temelji te način na koji se od rijetkog zapisa već unaprijed označenih znakova stvara novi, mnogo zgusnutiji zapis. U petom poglavlju prikazujem rezultate samoga rada, opisujem probleme na koje sam naišao tijekom rada na algoritmu i predlažem načine na koji se ti problemi u budućnosti mogu riješiti.

Ključne rjeći:

Računalni vid, prometni znakovi, detekcija, pračenje, realno vrjeme,Kanade Lucas Tomasi, Haarove značajke,KLT,cvsh2,Viola Jones,Marker

System for dense annotation of traffic signs in video sequences

Summary

This paper considerate implementation of algorithm for dense annotation of traffic signs in video sequences. Paper explains the principals and the way that algorithm do the job. In the beginning we give the overview of wider context of the project that this algorithm is also a part, and in a third part we explain the way that computer vision algorithms used in this project work. In part for the main idea and implementation of the algorithm are explained also with an accent on some problems that we introduced while working on this project with a proposal how to solve them.

Keywords:

Computer vision, traffic signs, detection, tracing, real time, Kanade Lucas Tomasi, Haar, features, KLT, cvsh2, Marker