Prepoznavanje znakova

Tehnička dokumentacija

Verzija 1.0

Studentski tim: Tomislav Babić

Tomislav Lukinić

Damir Kovač

Kristina Popović

Dominik Rojković

Maja Šverko

Nastavnik: Siniša Šegvić

Sadržaj

Tehnička dokumentacija

1. Prilagođavanje programskog okruženja za Visual Studio 2005

1.1 Definiranje programskog projekta

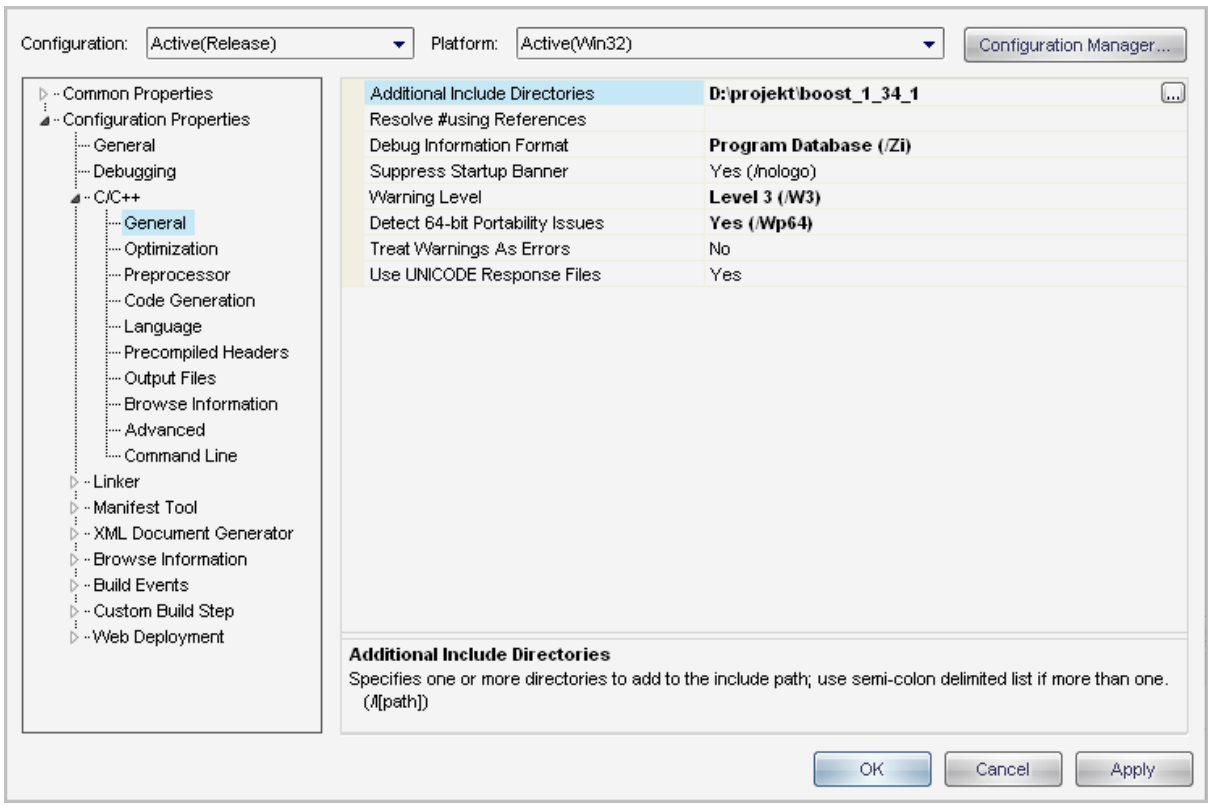
Prvi korak u prevođenju složenog programskog sustava je definiranje programskog projekta: određivanje komponenti koje valja povezati u konačni program, te konfiguracija parametara prevodioca i linkera. Najvažniji parametri prevodioca su lokacije sučelja vanjskih komponenti (xxx.h), dok su za linker to lokacije i imena korištenih vanjskih biblioteka (xxx.LIB). Na operacijskim sustavima Unix, programski projekt se definira generiranjem makefile-a uz pomoć alata kao što su autoconfigure, cmake ili tmake. Na Windowsima, to se postiže manualnim unošenjem podataka u grafičko sučelje korištenog razvojnog sustava, pri čemu veliki problem predstavlja nekompatibilnost formata datoteka projekta među različitim verzijama istog razvojnog sustava.

U nastavku teksta precizno ćemo popisati sve korake u definiranju programskog projekta u razvojnom okruženju MS Visual Studio.

1.1.1 Za otvaranje novog projekta u Visual Studiu 2005 odaberite *File → New → Project.* Za *Project type* odaberite *Win32*, a za *Template* odaberite *Win32 Console Application* te upišite ime novog projekta. U sljedećem prozoru odaberite *Application Settings,* a unutar njega *Console application* i označite *Empty project.* Time je stvoren novi projekt.

Datoteke programskog okruženja koje se nalaze u mapi *src* u projekt možemo dodati tako da u *Project Exploreru* desnim klikom kliknemo na *Source Files* i odaberemo *Add → Existing Item.* Jednostavniji je način da željene datoteke jednostavno odvučemo (drag) u Source Files. (Komponente koje je potrebno dodati u projekt za pokretanje programskog okruženja pogledajte u točki 1.2.1)

1.1.2 Prilikom prevođenja datoteke cvsh_main.cpp nedostajao nam je boost/smart.hpp library. Cijeli boost skup library-a je dostupan na Internetu na: http://www.boost.org/ . Boost library se u projekt uključuje tako da se u Project Properties → Configuration Properties → C/C++ → General → Additional Include Directories upiše lokacija (path) na kojoj su instalirane. (slika 1)



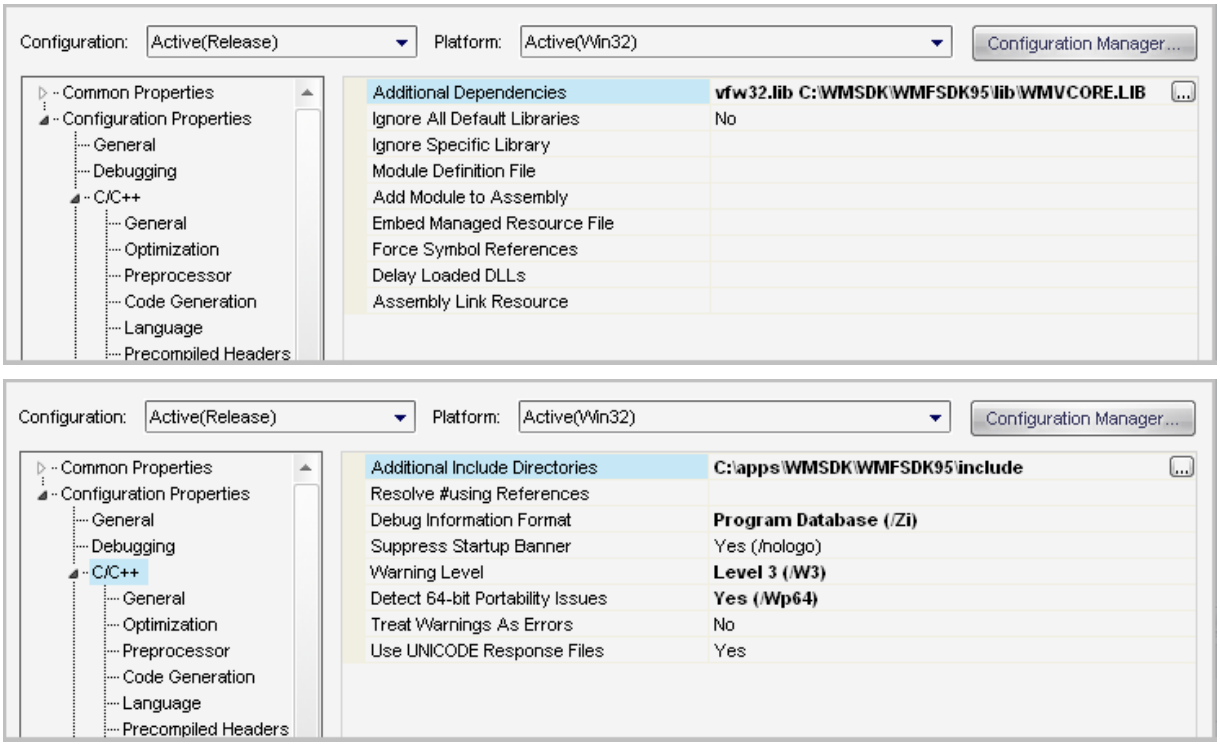
Slika 1: uključivanje boost library-a u project

1.1.3 Da bi se omogućio rad s .wmv datotekama, potrebno je instalirati Windows Media Format SDK kojeg se može naći na:

http://msdn2.microsoft.com/en-us/windowsmedia/bb190309.aspx

U opcijama za povezivanje Project Properties → Configuration *Properties* → *Linker* → *Input* → *Additional Dependencies* treba upisati lokaciju (path) na koju smo ga instalirali

Isto treba napraviti i u: *Project Properties* → *Configuration Properties* → *C/C++* → *General* → *Additional Include Directories (slika 2)*



Slika 2: uključivanje Windows Media Format SDK

1.1.4 U *Project Properties → Configuration Properties → General → Character Set treba odabrati Use Multi-Byte Character Set*

1.2 Prenosivost postojećeg programskog okruženja

Pri izvedbi projekta korišteno je programsko okruženje cvsh (ljuska) koja je na ZEMRIS-u razvijana da olakša eksperimentiranje i razvoj novih postupaka računarskog vida. Ljuska cvsh je oblikovana da bude što je moguće više prenosiva. Tamo gdje prenosivost nije moguće postići zbog poziva koji su specifični operacijskom ustavu (grafika, pristup sklopovlju, baratanje direktorijima, ...), razvijane su paralelne komponente za operacijske sustave Windows (sufiks `_w32`) i Unix (sufiks `_unix`). Tamo gdje se nije mogla postići uniformnost za Unix platforme, razvijane su konkretne implementacije za Linux i MacOS (sufiksi `_linux`, `_macx`). Nažalost, ljuska već duže vrijeme nije bila korištena pod Windowsima, pa je na početku rada valjalo identificirati:

- dijelove koda koji su u međuvremenu dodani a ne mogu se prevesti pod Windowsima (npr `chdir`)
- nove komponente čije implementacije za Windowse još nisu napisane; te komponente se dijele na:

  - one koje nisu kritične pa mogu imati praznu implementaciju (`util_fs::readDir`)
  - kritične komponente koje treba implementirati (čitanje formata wmv, `vs_file_vmw_w32`)
- komponente koje su prestale funkcionirati nakon usuglašavanja konzistencije nomenklature, i organizacijskih intervencija (`../../pd/util_w32.hpp`)
- neispravno napisane dijelove koda koje prevodilac pod Linuxom tolerira (nedostajući `#include`)

Slijedi detaljan popis intervencija koje su napravljene prilikom ponovnog prilagođavanja ljuske Windowsima.

1.2.1 U projekt treba uključiti sve .cpp datoteke programskog okruženja izuzev `grid_draw_x.cpp`, datoteka u mapi `win_x`, datoteka koje u sebi imaju naziv `linux, unix, macx` ili `test` te svih datoteka s riječi `grab` u nazivu izuzev `vs_grab.cpp` i `vs_grabUtil.cpp`.

1.2.2 U datoteci cvsh_main.cpp bilo je potrebno izbrisati iz koda include-ove (`#include<…>`) sljedećih library-a: `sys/type.h`, `sys/stat.h` i `unistd.h`. To povlači još jednu grešku u dijelu koda koje koristi jednu od njih. Za rješavanje tog problema dovoljno je zakomentirati liniju chdir (`tmpdir.c_str ()`); u datoteci `cvsh_main.cpp`.

1.2.3 U datoteci `win_w32_worker.cpp` liniju koda `#include "../../pd/util_w32.hpp"` treba zamijeniti sa `#include "../../util/util_w32.hpp"`.

1.2.4 U datoteci `util_fs_w32`:

- `bool readDir (char const* path,` treba zamijeniti s `bool util_fs::readDir (char const*path,`
- `bool isDir (char const* path){` zamijeniti sa `bool util_fs::isDir (char const*path){`
- `std::vector<std::string> entries)` zamijeniti sa `std::vector<std::string>& entries)`
- obrisati:

```
Struct stat buf;
Stat (path, &buf);
Return S_ISDIR (buf.st_mode)
```

1.2.5 U datoteku `img_dbg.cpp` treba uključiti `#include <string>`

2. Prepoznavanje znakova

2.1 Prevođenje boje RGB u boju HSI

RGB slika koristi cjelobrojne vrijednosti u intervalu od 0 do 255, dok HSI koristi vrijednosti iz pozitivnog dijela skupa R, uključujući nulu. Funkcija koja pretvara vrijednosti boja iz područja RGB u područje HSI kao ulaz dobiva položaj i veličinu dvodimenzionalne matrice u memoriji koja predstavlja sliku s RGB vrijednostima, te pokazivač na mjesto u memoriji gdje će rezultat biti spremljen u HSI obliku.

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

Funkcija petljom vrti po elementima matrice i za svaki piksel poziva posebnu funkciju koja radi pretvorbu nad učitanim vrijednostima. Vrijednosti H, S i I se dobivaju sljedećim formulama:

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

$$S = 1 - \frac{3 \times \min(R,G,B)}{R+G+B}$$

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

Varijabla H se dobiva u stupnjevima i u slučaju da je B > G, njena vrijednost se oduzima od punog kruga, tj. H = 360˚ – H. Isto tako, ako se pokaže da je dobivena vrijednost S < 0, varijabli H se pridružuje 180˚. Nakon pretvorbe, u pozivajućoj funkciji se dobiveni rezultati spremaju na zadano mjesto u memoriji (`float* pdst`) i petlja se tako vrti sve dok se svi elementi matrice ne učitaju.

2.2 Detekcija boje u prostoru HSI (stvaranje binarne slike)

Sljedeći korak je stvaranje binarne slike. Prilikom toga se koristi HSI slika koja se dobiva od RGBtoHSI funkcije. Binarna slika je digitalna slika koja za svaki piksel ima jednu od samo dvije vrijednosti. U ovom slučaju to služi da bismo izdvojili boje karakteristične za prometne znakove. Te boje na binarnoj slici prikazujemo bijelom bojom, dok su sve ostale crne.

Kao što smo već rekli, slika je predstavljena kao dvodimenzionalna matrica. Funkcija provjera element po element, gdje svaki element predstavlja jedan piksel. Ukoliko se H, S i I vrijednosti tog piksela nalaze unutar zadanih vrijednosti, intenzitet mu se postavlja na vrijednost 255 (postaje bijel) dok se ostalim pikselima postavlja na 0 (postaje crn). Taj postupak se ponavlja dok se ne obrade svi pikseli. Time se dobiva slika u kojoj su područja koja bi mogla biti znakovi obojana bijelom bojom.

Funkcija je deklarirana u nadjiBoju.h na sljedeći način:

```
void nadjiBoju(int width, int height, float* NorSlika, float* BinSlika,
hueLo, hueHi);
```

Kao što se vidi, funkcija prima četiri argumenta, cjelobrojne vrijednosti za širinu i visinu, te izvornu sliku u HSI formatu i alocirani prostor za binarnu sliku. Primjer pozivanja funkcije:

```
nadjiBoju (sirina, visina, imgHsi, imgBin, hueLo, hueHi );
```

Postavljen uvjet s parametrima za detekciju plavih znakova:

```
hueLo1 = (float) hueLo; // vrijednost parametra hueLo1 = 3.3
hueHi1 = (float) hueHi; // vrijednost parametra hueHi1 = 4

satLo = 0.1;
satHi = 2;
intLo = 40;
intHi = 120;
```

Postavljen uvjet s parametrima za detekciju crvenih znakova:

```
hueLo1 = (float) hueLo; // vrijednost parametra hueLo = 5.5
hueHi1 = (float) hueHi; // vrijednost parametra hueHi = 0.3

satLo = 0.1;
satHi = 2;
intLo = 40;
intHi = 120;
```

U funkciji hueLo1 i hueHi1 predstavljaju granice intervala topline boje, satLo i satHi predstavljaju granice intervala zasićenja, a intLo i intHi predstavljaju granice intervala za intenzitet boje. Također treba

primijetiti da se za crvenu boju (koja se nalazi na oba kraja HSI spektra) zadaju obrnuto vrijednosti za toplinu (hue) boje. Dakle, ako je hueLo zadan tako da bude veći od hueHi onda interval u kojem će se gledati neće biti [hueLo, hueHi], već [0, hueHi] U [hueLo, 360˚].

2.3 Narastanje područja u binarnoj slici

Sljedeći korak je detektiranje skupine piksela unutar slike i označavanje takvih skupina. To se radi funkcijom narastanja područja. Narastanje područja je algoritam koji određuje područje obojano istom bojom na slici.[1]

Naš algoritam narastanja se sastoji od dva dijela. U prvom dijelu za svaki piksel provjerava je li bijele boje, u slučaju da je to točno, piksel se označava novom vrijednošću te se ulazi u rekurzivnu funkciju, gdje se provjerava za svaki njegov susjedni piksel je li bijele boje i rekurzivno, jesu li bijele boje njegovi susjedi. Svi susjedi se označavaju istom vrijednošću, iz čega slijedi da se i skupina piksela označava jedinstvenom vrijednošću. Kada funkcija pronađe novi bijeli piksel, ponavlja se isti postupak, ovoga puta sa novom vrijednošću.

U drugom dijelu se pomoću označenih vrijednosti traže jedna po jedna, skupine piksela i pritom se određuje krajnja lijeva, krajnja gornja, krajnja desna i krajnja donja točka skupine. Također se određuje broj piksela u skupini, te ako taj broj zadovoljava uvjet veličine skupine, funkcija vraća krajnje točke skupine. Postupak se ponavlja za svaku skupinu bijelih piksela na slici.

Funkcija narastanja prima tri argumenta. Prva dva argumenta su cjelobrojni brojevi koji predstavljaju širinu odnosno visinu slike, a treći argument predaje binarnu sliku. Također funkcija kao rezultat vraća jednodimenzionalno polje u kojem je smješten niz koordinata. Spremaju se po četiri koordinate za svaki pravokutnik to jest detektiranu skupinu piksela. Prva koordinata predstavlja krajnju lijevu točku (x1), krajnju gornju točku (y1), krajnju desnu točku (x2), krajnju donju točku (y2), iz čega slijedi da je gornji lijevi kut pravokutnika (x1, y1), gornji desni (x2,y1), donji lijevi (x1,y2) te donji lijevi kut (x2,y2). Kraj niza koordinata se označava s -1.

Memorija zauzeta za polje koordinata pravokutnika se oslobađa u glavnoj funkciji naredbom:

```
free (pPravokutnici);
```

Funkcija je deklarirana u narastanje.h i to na sljedeći način:

[1] Wikipedia, Flood Fill

```
int *narastanje (int width, int height, float* imgBin);
```

Primjer poziva funkcije:

```
pPravokutnici = narastanje (sirina, visina, imgBin);
```

2.4 Integracija

Programsko okruženje dobavlja sliku u obliku jednodimenzionalnog polja koje predstavlja trodimenzionalnu RGB matricu. Slika je u polju zapisana u sljedećem obliku: „RGBRGBRGBRGB…“. U istom obliku se predstavlja i HSI slika.

Cijeli postupak gore navedenih funkcija se odvija u metodi `alg_znakovi::process` koja je deklarirana u klasi alg_znakovi i to na sljedeći način:

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

Poziva se na sljedeći način:

```
void alg_znakovi::process(
  const img_vectorAbstract& src,
        const win_event_vectorAbstract&, int)
```

Izvorna slika se dobavlja pomoću objekta *src* koji je tipa *img_vectorAbstract*, a zatim se sprema u *imgs_[0]*, gdje je *imgs_* tipa *img_vector*. HSI slika se sprema u obično polje dimenzija 3*visina*širina. Isti prostor se zauzima i za binarnu sliku. Zatim se poziva funkcija RGBtoHSI. Nakon dobivene slike poziva se funkcija nadjiBoju, te funkcija narastanja.

Također treba napomenuti 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.

2.5 Selekcija reprezentativnog skupa slika

Selekcija reprezentativnog skupa slika je proces odabira slika (frameova) iz zadanog videa (konkretno ovdje se radi o VMW formatu) i to na takav način da skup tih slika bude reprezentativan. Reprezentativnost u ovom slučaju znači da iz skupa od nekoliko tisuća ili desetaka tisuća frameova odaberemo mnogo manji skup slika (reda nekoliko stotina) na način da poštujemo kriterije koje smatramo zanimljivima u svrhu ocjenjivanja performansi nad početnim slijedom. Kriteriji zavise o ulaznom slijedu i performansama koje želimo ocijeniti.

Kvaliteta slike:

- realne boje
- nepostojanje ili vrlo slabo postojanje zamućenja
- nepostojanje refleksija, svjetlosnih varki ili zrcalnih slika (refleksije od glatkih površina)
- nepostojanje objekata sa sličnim bojama kao traženi objekti
- nepostojanje objekata koji potpuno ili djelomično zakrivaju tražene objekte

sadržavanje traženih objekata:

- postojanje što više traženih objekata
- postojanje raznovrsnih traženih objekata na jednoj slici
- postojanje raznovrsnih traženih objekata na cijelom skupu slika
- postojanost istih traženih objekata u nekoliko veličina i položaja
- postojanje objekata s graničnim vrijednostima koje se uzimaju u mjerenja

slike namijenjene ispitivanju osjetljivosti na pogreške:

- skup slika koje imitiraju tražene objekte ili njihove dijelove
- skup slika koje sadrže tražene objekte djelomično iskrivljene ili prekrivene
- ekstremni uvjeti poput vremenskih nepogoda ili refleksija

Poštujući gore navedene kriterije smanjili smo više od pola sata filma na skup od nešto manje od dvije stotine reprezentativnih slika. Skup smo izradili koristeći program zvan Frameshots u verziji 2.2. Program je dostupan kao shareware na http://www.frame-shots.com/ . Prilikom izrade projekta je korištena puna verzija.

2.6 Ručno označavanje znakova

Pogledajmo prvo nekoliko definicija testiranja:

- Testiranje je proces izvođenja programa sa svrhom pronalaženja pogrešaka.
- Cilj testiranja je pokazati da program ispravo obavlja željene funkcije.
- Aktivnost s ciljem otkrivanja informacija o ispravnosti i kvaliteti, te poboljšanja pronalaženjem kvarova i problema testiranog produkta.

Kako bismo izvršili testiranje našega programa i dobili korisne povratne informacije, najprije smo morali stvoriti reprezentativni uzorak slika na kojem ćemo vršiti testiranja. To je podrobnije opisano u prošlom poglavlju. Zatim smo se trebali odlučiti za metodu.

Ideja je inicijalno bila da napravimo pomoćni program koji će automatski rezultate rada glavnog programa uspoređivati s bazom podataka o slikama koja je izrađena ručno, no zbog nedostatka vremena to nije u potpunosti zaživjelo.

Kako bi se testiranje obavilo automatski bilo je ipak prethodno potrebno napraviti bazu podataka iz izabranog reprezentativnog skupa slika ručno. Prilikom određivanja koordinata služili smo se programom zvanim PixelRuler koji se može pronaći u trial verziji na stranici:
www.mioplanet.com/products/pixelruler/index.htm

Za spomenutu bazu podataka poslužila je jednostavna .txt datoteka u kojoj je broj retka ujedno značio i broj slike na koju se taj redak odnosi. Tako npr. redak 39 u uzlaznom slijedu slika predstavlja 39. sliku u nizu. Možda nije loše napomenut da kako bismo ovo postigli potrebno je isključiti opciju word wrap u notepadu.

Koristili smo sljedeći format pisanja retka datoteke:

- redak sa znakom „X“ označavao je sliku koja ne sadrži nijedan znak.
- Slike koje sadrže barem jedan (ili više znakova) imale su redak sljedećeg formata:
  `<ime znaka>@<Xkoordinata u px>,<Ykoordinata u px>:<max veličina znaka od ruba do ruba>`

  Značenja pojedinih dijelova formata su sljedeća:
  - ime znaka – pripadajuća oznaka znaka prema važećem pravilniku o prometnim znakovima, opremi i signalizaciji na cestama. Popis svih oznaka može se pronaći na: http://www.nn.hr/clanci/sluzbeno/2003/0450.htm
  - X i Y koordinate su koordinate sjecišta dijagonala znaku opisanog kvadrata izražene u pikselima
  - Veličina znaka od ruba do ruba – označava veličinu pronađenog znaka i to uzimajući vrijednost dulje stranice opisanog mu kvadrata izraženu u pikselima

- Ako je u nekoj slici pronađeno više od jednog znaka tada se zapisi o pojedinom znaku odvajaju sa simbolom „#”

Primjer zapisa znakova u datoteci (jedan redak na ekranu je ekvivalentan jednom retku datoteke):

```
D14@368,233:46#D15@415,229:47
X99
D15@612,92,114#D20@483,265:61#D06@570,269:21
A07@437,277:19
```

2.7 Ocjena performanse

Iako je u početku bilo zamišljeno da se ocjena performanse obavi programski, odnosno automatski, zbog tehničkih razloga je to obavljeno ručno, promatrajući izlaz iz ljuske i ocjenjujući njegovu točnost. Metoda kojom se radila ocjena performanse je takozvana metoda preciznosti i odziva (eng. precision and recall).

Kao što je već navedeno, izvršen je odabir reprezentativnog skupa slika i nad njima ispitana funkcionalnost programa. Ispitivanje se odnosilo na dvije različite vrste znakova: plavi i crveni (odnosno bijeli s crvenim rubom). Za plave znakove, rasponi prihvatljivih vrijednosti varijabli H, S i I varijabli zadani su ovako:

H = <3.3, 4>

S = <0.2, 2>

I = <40, 150>

Za crvene:

H = [0,0.3> + <5.5, 2*PI>

S = <0.05, 0.6>

I = <10, 150>

Parametri kojima je filtrirana veličina znaka koju program smije uzeti u obzir za plave znakove

20 < P < 3700

Za crvene:

90 < P < 3700

(parametri su konfigurirani promjenom uvjeta u kodu funkcije za stvaranje binarne slike, a ne preko naredbe config kojom ljuska omogućuje da se to radi preko komandne linije, zbog kroničnog nedostatka vremena).

U reprezentativnom uzorku sadržano je 175 slika, a ispitivanje je obavljeno na procesoru Intel Core Duo T2050 gdje prosječna obrada jedne slike rezolucije 720x576 piksela traje 212 ms.

Rezultati su obrađeni tako da su podijeljeni u tri skupine: znakovi koje je program ispravno prijavio

(spremljeni u varijablu tp), znakovi koje program nije prijavio (fn) i na kraju, znakovi koje je program neispravno prijavio (odnosno prepoznao je znak gdje ga zapravo nema) koji su spremljeni u varijablu fp. Nadalje, iz tih varijabli smo dobili preciznost i odziv u postocima prema sljedećim formulama:

Preciznost = tp / (tp + fp)
Odziv = tp / (tp + fn)

Za plave znakove, preciznost rada programa je 50,89% dok je odziv 89,06%. Što se tiče crvenih znakova (odnosno znakova s crvenim obrubom), preciznost je 27,93% dok je odziv 76,09%. Znatno manja preciznost programa u radu s crvenim znakovima je posljedica toga da je vrijednost S njihove crvene boje vrlo niska tako da se pri kalibraciji intervala moralo to uzeti u obzir. Time se otvorila veća mogućnost da se neke ne-prihvatljive regije označe kao znakovi, budući da se snižavanjem S-a gubi značajka pojedine boje i približava se crnoj boji, odnosno apsolutnom nedostatku boje. Preciznost u označavanju plavih znakova je dosta veća jer su oni pretežito svjetlijeg tona. Osim toga, češće se na slikama pojavljuju ne-znakovni elementi crvene boje nego plave (poput automobilskih svjetala, koja su nerijetko dovoljno blizu objektivu kamere da svojom veličinom budu propušteni kroz filtar veličine čije smo parametre gore naveli). No, ni odziv nije stopostotan (ni u jednom slučaju), a to je pak iz razloga što smo u reprezentativnom uzorku odabrali i neke slike s odsjajem ili slike koje su snimljene u bržem kretanju što je uzrokovalo njeno zamućenje.

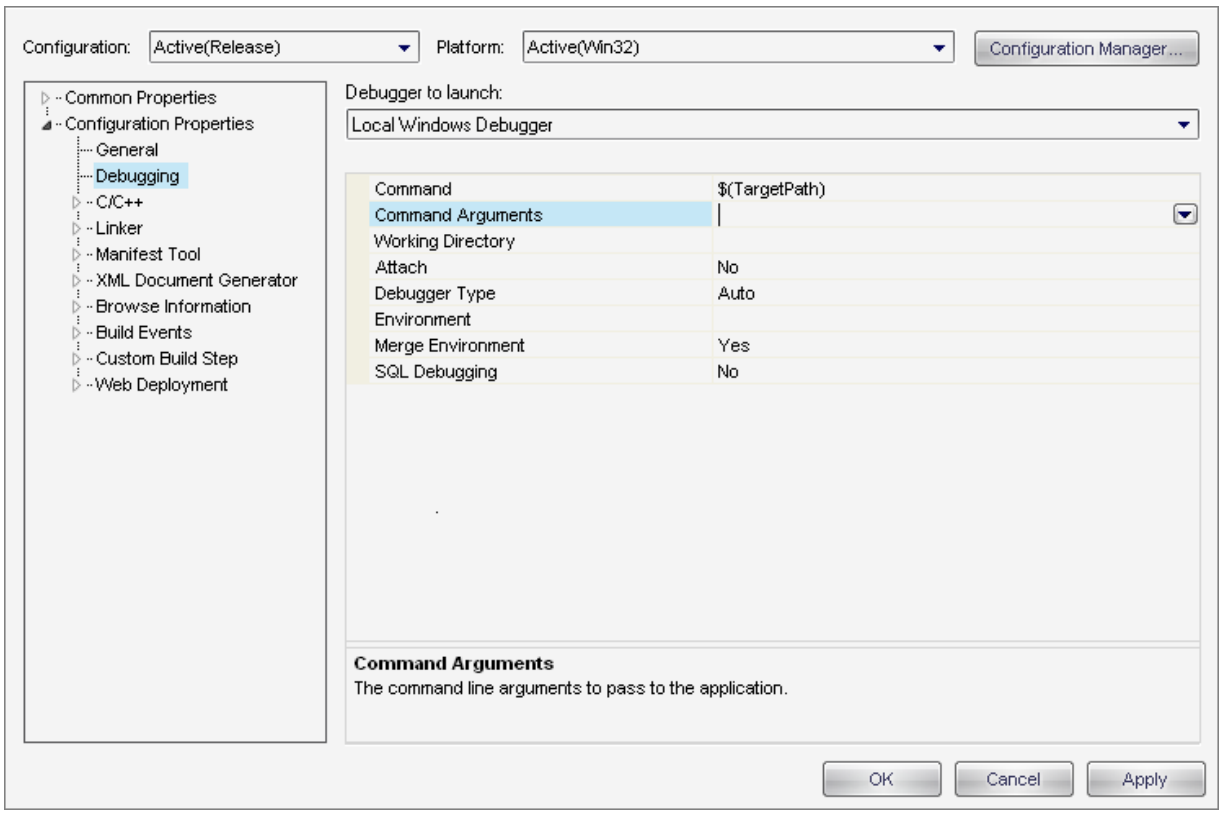
Jedan od načina kojima bi mogli poboljšati performanse programa je smanjiti njegovu osjetljivost na određen tip boje, a da pritom sačuvamo kvalitativni rezultat. Budući da to ovisi o intervalima HSI područja, to bi možda dobro bilo obaviti statističkom obradom boja velikog broja znakova iste vrste (npr. znakova s

Slika 3: Primjer pogrešnog izlaza iz programa

crvenim obrubom) i tako omogućiti što finiju kalibraciju intervala. S druge strane, tom problemu bi se moglo pristupiti programskom obradom neke regije koja sadrži u sebi traženu boju – npr. provjera čine li male povezane regije boje neki geometrijski lik određene veličine (poput trokuta ili kruga) koji bi označavao da se zapravo radi o prometnom znaku.

3. Rad s programskim okruženjem

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



Slika 4: 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 tj. -sf=…\video.avi#početna_slika-završna_slika

Odredište dobivene datoteke:

-df=…\rezultat.avi

Algoritam koji se koristi:

-a=algoritam

Nekoliko primjera potpune naredbe:

-sf=E:\video.wmv#1-5 -df=E:\temp\video.avi -a=znakovi

-sf=E:\video.wmv -a=znakovi

-sf=E:\video.wmv -a=znakovi –i

4. Literatura

- S. Šegvić, ``Višeagentsko praćenje objekata aktivnim računarskim vidom'', doktorska disertacija, Fakultet elektrotehnike i računarstva, Zagreb, Hrvatska, lipanj 2004.
- Conversion from RGB to HSI, 31.10.2007., *Color processing*, http://fourier.eng.hmc.edu/e161/lectures/color_processing/node3.html, 8.11. 2007.
- HSL and HSV, 25.6.2003., *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/HSI_color_space, 10.11.2007.
- Precision and Recall, 21.11.2007., *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Precision_and_recall, 7.1.2008.
- Binary image, 2.10.2002., *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Binary_image, 30.11.2007.
- Flood fill, 13.12.2001., *Wikipedia, the free encyclopedia*, http://en.wikipedia.org/wiki/Flood_fill, 8.12.2007.
- John R. Shaw., QuickFill: An efficient flood fill algorithm, 2.2.2004., *CodeProject*, http://www.codeproject.com/KB/GDI/QuickFill.aspx, 13.12.2007.
- Flood Fill, 2004, *Lode's Computer Graphics Tutorial - Flood Fill,* http://student.kuleuven.be/~m0216922/CG/floodfill.html, 10.12.2007.
- Pixel Ruler v3.1, Pixel Ruler, *Your Free Virtual Screen Ruler*, www.mioplanet.com/products/pixelruler/index.htm, 03.01.2007.
- Pravilnik, 03.03.2005.,*Pravilnik o prometnim znakovima, signalizaciji i opremi na cestama,* http://www.nn.hr/clanci/sluzbeno/2005/0662.html , 3.12.2007
- D.Begusic, Inžinjerska grafika, *Boja u računarskoj grafici,* http://lab405.fesb.hr/igraf/Frames/fP5_1.htm, 15.11.2007
- YUV, 29.9.2002., *wikipedia, the free encyclopedia,* http://en.wikipedia.org/wiki/YUV, 2.12.2007.