SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

PREPOZNAVANJE PROMETNOG ZNAKOVLJA U POKRETNOJ SLICI

Završni rad

Maja Soldo

Osijek, 2014.

Sadržaj

1. UVOD

1.1. Zadatak diplomskog rada

Veliki napori ulažu se u prepoznavanje objekata snimljene slike uporabom računalnog vida. Isto tako, u autoindustriji, uloženi su veliki napori primjene računalnog vida u prepoznavanju objekata na prometnici ispred vozila u pokretu. Zadatak ovog rada jest razraditi i implementirati metodu prepoznavanja objekata statičnog prometnog znakovlja iz snimke prometnice vozila u pokretu uporabom metodologija koje se oslanjaju na metode računalnog vida.

2. PREGLED KORIŠTENIH TEHNOLOGIJA I ALGORITAMA

2.1. Biblioteka OpenCV

Test citiranja literature [?]

Postoji više metoda odnosno algoritama koje se koriste za prepoznavanje različitih objekata na slici. Neke od njih su:

2.2. Algoritam usporedbe predloškom

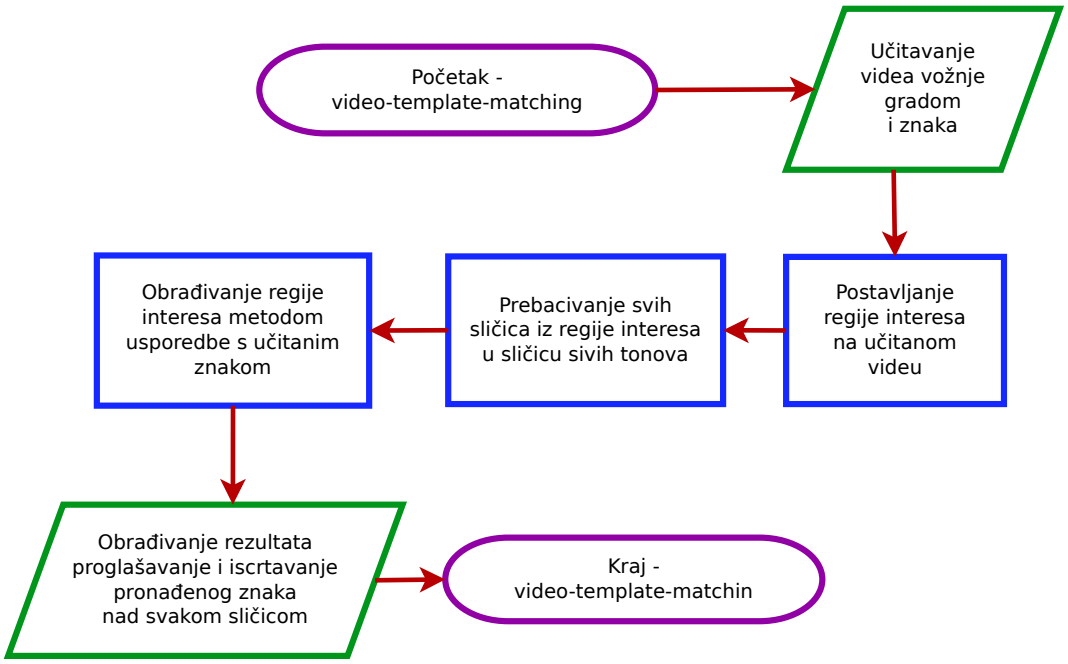
3. PREPOZNAVANJE PROMETNOG ZNAKOVLJA U POKRETNOJ SLICI

3.1. Prepoznavanje prometnog znakovlja upotrebom metode usporedbe s predloškom

Prepoznavanje prometnog znaka u pokretnoj slici implementirano je u programu nazvanom `video-template-matching`. Program se u potpunosti oslanja na biblioteku OpenCV koja je opisana u podpoglavlju 2.1. Program se sastoji od pet logičnih cijelina.

- Učitavanje videa vožnje gradom i učitavanje znaka.
- Postavljanje regije interesa na učitanom videu.
- Prebacivanje svake sličice iz regije interesa u sličicu sivih tonova.
- Obrađivanje takve regije interesa metodom usporedbe s učitanim znakom koji je isto slika sivih tonova.
- Obrađivanje rezultata, proglašavanje i iscrtavanje pronađenog znaka nad svakom sličicom.

Slika 3.1. prikazuje dijagram toka programa odnosno logičke cijeline od kojih se program sastoji.



Slika 3.1.: Dijagram toka programa video-template-matching

3.1.1. Učitavanje videa vožnje i učitavanje znaka

Ispis koda 3.1.: Izvorni kod za učitavanje videa i znaka

```
#include "opencv2/imgproc/imgproc.hpp"

int main (int argc, char *argv[])
{
    // kreiranje objekta cap za ucitavanje videa
    VideoCapture cap("video/znakich2.mp4");
    if(!cap.isOpened())  // provjera uspjeha ucitavanja
        return -1;

    // kreiranje objekta Mat za spremanje znakova
    Mat znak1, znak2, znak3, znak4;

    // ucitavanje izrezanih znakova u razlicitim velicinama
    // trenutno se koristim samo znak2
    znak1 = imread ("roi/01_roi.png");
    znak2 = imread ("roi/02_roi.png");
    znak3 = imread ("roi/03_roi.png");
    znak4 = imread ("roi/04_roi.png");

}
```

Izlistanje koda 3.1. prikazuje primjer učitavanja videa i učitavanje znaka upotrebom klase `VideoCapture` i funkcije `imread` koji su uključeni dodavanjem biblioteke `imgproc.hpp`. Kreiranom objektu `cap` predana je putanja do videa kojeg treba učitati. Ukoliko video nije uspješno učitan program završava i vraća `-1`. Kreiranim objektima `znak1`, `znak2`, `znak3`, `znak4` pridjeljene su slike učitane upotrebom funkcije `imread` kojoj je predana putanja do slike koju treba učitati.

3.1.2. Postavljanje regije interesa

Ispis koda 3.2.: Izvorni kod za postvljanje regije interesa

```
    // prebacivanje znak2 u sliku sivih nijansi
2   cvtColor (znak2, znak2, CV_BGR2GRAY);

4   // ucitavanje slicice iz videa u frame
    // postavljanje regije interesa u roi
6   cap >> frame;
    rect = Rect (600, 150, 480, 120);
8   roi = frame(rect);
```

Izlistanje koda 3.2. prikazuje upotrebu funkcije `cvtColor` kojom se učitani znak prebacuje u sliku sivih tonova. Zatim se učitava sličica u objekt `frame` iz objekta `cap`. Tada se kreira objekt `rect` pomoću funkcije `Rect()` kojom definiramo pravokutnik koordinatama gornjeg lijevog kuta te širinom i visinom. Takav pravokutnik upotrebljava se za definiranje regije interesa nad učitanom sličicom odnosno objektom `frame` te se regija interesa sprema u objekt `roi`. Regija interesa se kasnije upotrebljava u algoritmu usporedbe s predloškom.

3.1.3. Pozivanje algoritma usporedbe s predloškom

Ispis koda 3.3.: Izvorni kod pozivanja algoritma usporedbe s predloškom

```
    // postavljanje velicine results ovisno o velicini roi i znak2
    int resultRows, resultCols;
    resultRows = roi.rows - znak2.rows + 1;
    resultCols = roi.cols - znak2.cols + 1;
    results.create (resultRows, resultCols, CV_32FC1);

    // pretvaranje roi u greyscale
    cvtColor(roi, groi, CV_BGR2GRAY);
    // trazenje znaka u greyscale groi i spremanje u results
    matchTemplate (groi, znak2, results, 5);
    // normaliziranje rezultata
    normalize (results, results, 0, 1, NORM_MINMAX, -1);
    // traznje lokacije najveceg rezultata
    double minVal; double maxVal;
    minMaxLoc (results, &minVal, &maxVal, &minLoc, &maxLoc, Mat() );
    // spremanje rezultata u vektor tocaka
    vPoint.push_back (maxLoc);
```

Prije pozivanja algoritma usporedbe s predloškom implementiranog u funkciji `matchTemplate()` potrebno je kreirati matricu/sliku u koju će funkcija spremati rezultate. Matrica `results` se kreira ovisno o veličini matrica `roi` i `znak2` kao što prikazuje izlistanje koda 3.3.. Kako su svi parametri koje funkcija `matchTemplate()` kreirani, slijedi njeno pozivanje. Rezultat rada algoritma je spremljen u `results` te se takvi rezultati normaliziraju korištenjem funkcije `normalize()`. Zatim slijedi lociranje najvećih i najmanjih vrijednosti piksela u rezultantnoj matrici upotrebom funkcije `minMaxLoc()`. Korištena metoda usporedbe sprema najbolje rezultate kao maksimalne vrijednosti i zato se `maxLoc` sprema u vektor točaka za daljnju obradu.

3.1.4. Eliminacija lažno pozitivnih rezultata

Zbog velike količine lažno pozitivnih rezultata osmišljen je kod za eliminaciju istih računanjem geometrijske udaljenosti između dvije točke. Kod za računanje geometrijske udaljenosti je prikazan u ispisu kod 3.4.. Pretpostavka je da se znak pojavljuje postepeno što znači da bi rezultati trebali biti u blizi prošlog rezultata odnosno ne bi trebali iznenadno mjenjati veliku udaljenosti.

Ispis koda 3.4.: Izvorni kod računanja geometrijske udaljenosti između dvije točke

```
int calcPointDist (Point maxLoc, Point prevMaxLoc)
{
    int dist = 0;
    dist = sqrt ((maxLoc.x - prevMaxLoc.x) * (maxLoc.x - prevMaxLoc.x) +
                 (maxLoc.y - prevMaxLoc.y) * (maxLoc.y - prevMaxLoc.y));
    cout << dist << endl;
    return dist;
}
```

Kao što se vidi iz ispisa koda 3.5. program ima beskonačnu petlju u kojoj učitava sličicu po sličicu te nad svakoj do njih ponovo poziva algoritam usporedbe s predloškom, pronalazi najveće vrijednosti te iste sprema u vektor točka `vPoint`.

Ispis koda 3.5.: Izvorni kod obrade svih učitanih sličica

```
for (;;){
    // ucitaj frame, postavi roi, prebaci u greyscale
    cap >> frame;
    roi = frame(rect);
    cvtColor(roi, groi, CV_BGR2GRAY);

    matchTemplate (groi, znak2, results, 5);
    normalize (results, results, 0, 1, NORM_MINMAX, -1);
    minMaxLoc (results, &minVal, &maxVal, &minLoc, &maxLoc, Mat() );
    vPoint.push_back (maxLoc);
    int size = vPoint.size();
```

Za svaki rezultat veći od postavljene vrijednosti (0.75) izvršava se logika koja određuje hoće li se prepoznati znak odnosno iscrtati pravokutnik oko njega. Za prvih 10 `maxVal` vrijednosti većih od 0.75 znak se prepoznajue ako je udaljenosti između početne i trenutne vrijednosti manja od 10. Za ostale rezultate uspoređuje se trenutni i sedmi prije. Ukoliko je udaljenost između njih veća od 15 znak se prepoznaje i iscrtava. Kod vidljiv u ispisu 3.6. još prikazuje ponovno stavljanje iste točke u vektor zbog dužeg prikaza/praćenja znaka.

Ispis koda 3.6.: Izvorni kod eliminacije lažno pozitivnih rezultata

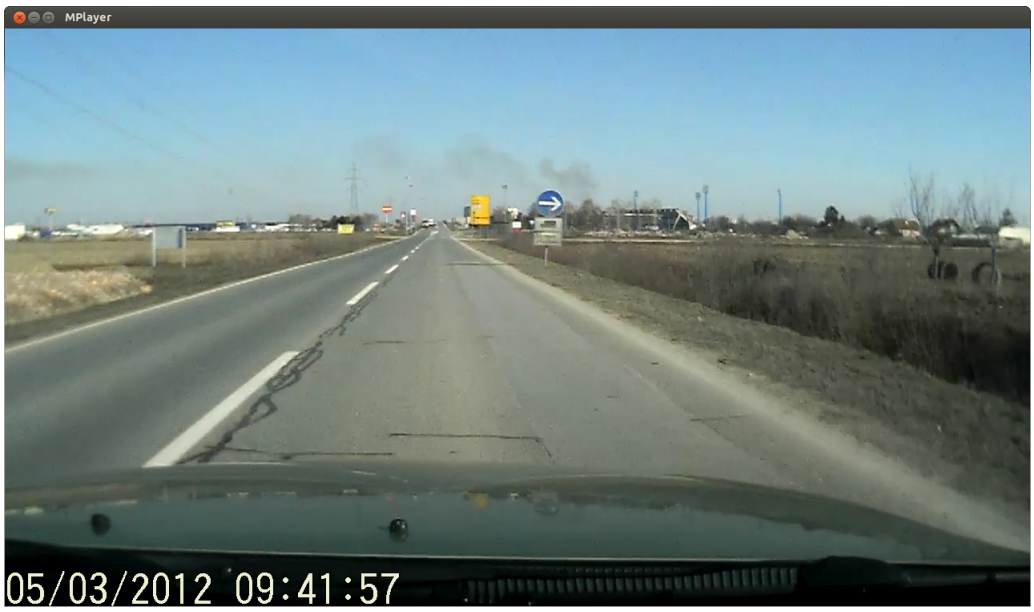
```
    // Za svaki rezultat veci od 0.75
    if (maxVal > 0.75) {
```

```
3           // racunaj udaljenost izmedu dva rezultata
            // Ako je vektor tocaka manji od 10 clanova
5           if (size < 10) {
                // racunaj udaljenost izmedu trenutne i prve tocke
7               dist = calcPointDist (maxLoc, vPoint.at(0));
                // ako je udaljenost manja od 10 isrctaj znak
9               if (dist < 10) {
                    rectangle (groi, maxLoc, Point(maxLoc.x + znak2.cols,
11                  maxLoc.y + znak2.rows), Scalar::all(0), 2, 18, 0 );
                }
13          }
            // Ako je vektor tocaka veci od 10 clanova
15          else {
                // racunaj udaljenost izmedu trenutnog i sedomog prije
17              dist = calcPointDist (maxLoc, vPoint.at(size-7));
                // iscrtaj pravokutnik ako je udaljenost manja od 15
19              if (dist < 15) {
                    rectangle (groi, maxLoc, Point(maxLoc.x + znak2.cols,
21                  maxLoc.y + znak2.rows), Scalar::all(0), 2, 18, 0 );
                    // ponovi istu tocku u vektor da se duze pracati znak
23                  vPoint.push_back (maxLoc);
                }
25              else {
                // ponovi istu tocku u vektor da se duze pracati znak
27              vPoint.push_back (maxLoc);
                }
29          }
        }
31  }
```

4. REZULTATI

U ovom poglavlju prikazani su rezultati rada razvijenog programa za prepozavanje statičnog znaka iz snimke prometnice vozila u pokretu. Za ispitivanje funkcionalnosti metode odabrana je jedna scena voženje i jedan znak kojeg metoda treba pronaći. Znak je prikazan na slici 4.1., a scena vožnje je snimljena na osječkoj obilaznici i prikazana na slici 4.2..

Slika 4.1.: Prikaz korištenog znaka - obvezan smjer kretanja u desno



Slika 4.2.: Prikaz testirane scene voznje

4.1. Prikaz rezultata

Rezultati su prikazani na slikama s dva prozora. U prvom prozoru prikazana je regija interesa sa scene unutar koje algoritmom usporedbe tražimo znak. U drugom prozornu prikazana je matrica rezultata koju metoda koristi za pronalazak znaka. Postoji nekoliko tipova rezultata: pozitivni, negativni, lažno pozitivni i lažno negativni. Svi tipovi rezultata su predstavljeni dalje u tekstu. Slika 4.3. prikazuje sličicu iz videa sa scene na kojoj nema znaka niti ga je metoda/program našao što je pozitivan rezultat.

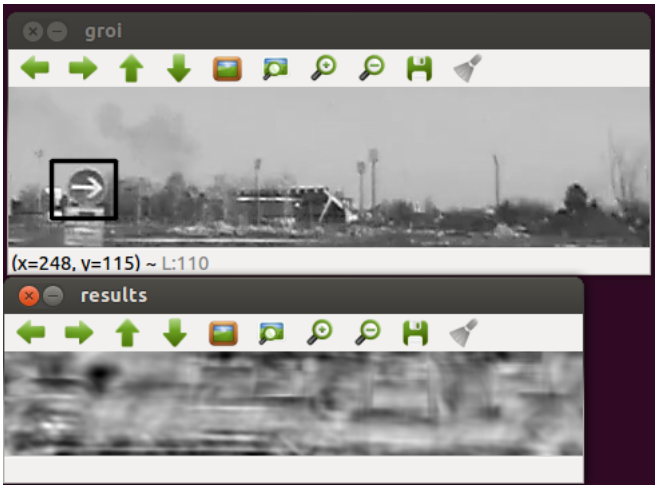
Slika 4.3.: Prikaz sličice videa i rezultata - pozitivan rezultat

Lažno pozitivni rezultat prikazuje slika 4.4. na kojoj je metoda pronašala znak gdje nije trebala. Takvi rezultati su očekivani ali nisu poželjni te ih se pokušalo smanjiti na što manji broj metodom opisanom u podpoglavlju 3.1.4.

Slika 4.4.: Prikaz sličice videa i matrice rezultata - lažno pozitivan rezultat

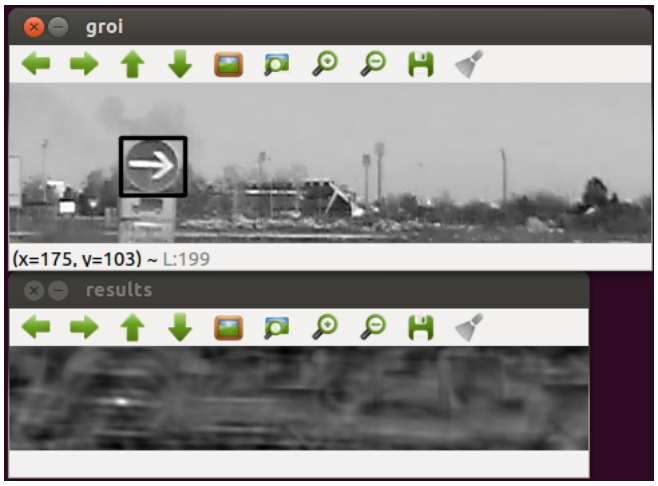
Slika 4.5.: Prikaz sličice videa i matrice rezultata - pozitivan rezultat

Na slikama 4.5., 4.6. i 4.7. vidi se da je metoda uspješno pronašla znak na različitim udaljenostima odnosno veličinama znaka iako se koristila samo jedna veličina znaka za pronalazak. Metoda odnosno program bih se mogao unaprijediti tako da se ugradi uspoređivanje s različitiim veličinama predloška odnosno znaka.



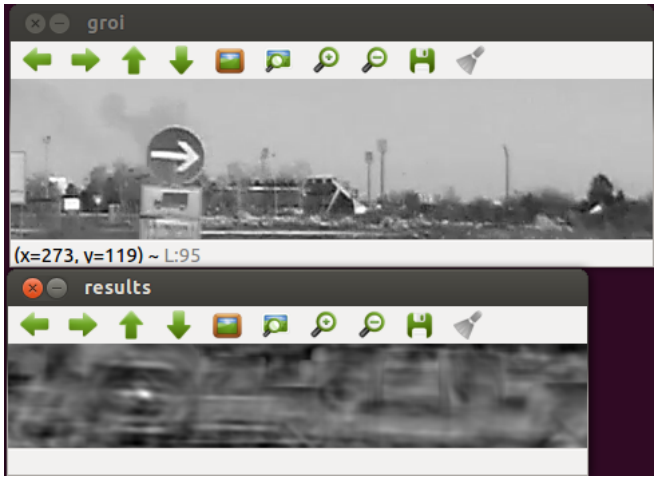
Slika 4.6.: Prikaz sličice videa i matrice rezultata - pozitivan rezultat

Matematička metoda korištena u algoritmu prikazuje pozitivne rezultate bijelom bojom te se na slikama 4.6. i 4.7. jasno mogu vidjeti "žarišta" u matrici rezultata.



Slika 4.7.: Prikaz sličice videa i matrice rezultata - pozitivan rezultat

Slika 4.8.: Prikaz sličice videa i matrice rezultata - pozitivan rezultat



Slika 4.9.: Prikaz sličice videa i matrice rezultata - pozitivan rezultat

5. ZAKLJUČAK

SAŽETAK

ŽIVOTOPIS

PRILOZI

Izlistanje koda