# Prepoznavanje i prebrojavanje pešaka koji su prešli semafor na crveno svetlo

Soft computing - Jelena Janković, Tamara Mrkšić
Profesor Đorđe Obradović, asistent Ivan Perić, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, Srbija

## Uvod

Na ovom papiru su predstavljeni algoritmi i tehnologije pomoću kojih je bila ideja da se izvrši prepoznavanje i prebrojavanje pešaka koji su prešli semafor na crveno svetlo.

Osnovna ideja samog postupka jeste detekcija semafora na video snimku sa upaljenim crvenim svetlom, zatim detekcija puta, detekcija pešaka koji prelaze put na upaljeno crveno svetlo kao i njihovo prebrojavanje

## Algoritmi

U postupku rešavanja korišteni su sledeći algoritmi: **Template matching, Hough transformation, Blobs detection, Canny (Edge) detection, Haar Cascades (haar cascades models, dpm cascade models), Background subtractor MOG, Background subtractor MOG2, Basic Thresholding Operations (kao i Bitwise AND operator).**

## Detekcija semafora sa upaljenim crvenim svetlom

**Template matching** je tehnika za pronalaženje dela slike koji se najviše podudara sa uzorkom koji se na slici pronalazi. Ova tehnika koristi dve slike:

- source image - slika na kojoj se pronalazi uzorak
- template image - uzorak koji se pronalazi na source image-u .

Primenom ove tehnike u **C++-u,** kao i u **Python-u**, dobijeni su rezultati prikazani na slici ispod:




U okviru **C++** implementacije Template Matching tehnike u funkciji **matchTemplate** je korištena **SQDIFF** metoda koja je davala najpreciznije rezultate. Na slici ispod je prikazana matematička formula pomenute metode, uz pomoć koje je detektovan semafor sa upaljenim crvenim svetlom, uz **threshold** koji iznosi **0.90.**

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

Takođe, ova tehnika je korištena i uokviru **Python** implementacije, s tim da su se najbolje pokazale formula poput **CCOEFF_NORMED,**na kojoj je korišten threshold od **0.90.** Razlog dobijanja različitih rešenja primenom iste formule u različitim programskim jezicima, jeste korištenje različitog OpenCv okruženja.

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

## Detekcija puta

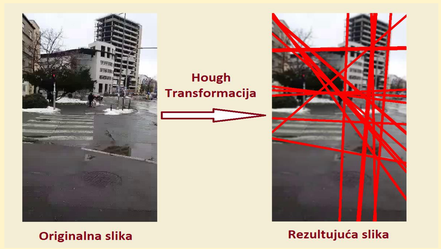
### Implementacija u C++ programskom jeziku

Kada je u pitanju detekcija puta, primenjen je sledeći postupak:

- Učitavanje slike u boji
- Pomoću Canny edge detection tehnike, detektovane su ivice, pa je dobijena crno bela slika
- Primena Hough transformacije na prethodno dobijenu sliku
- Detekcija svih linija primenom određenog praga
- Prikaz dobijene slike na ekranu

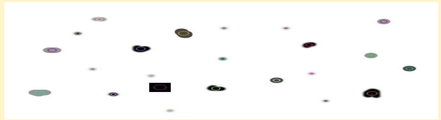
Iako OpenCV ima implementiranu Hough transformaciju, u ovom rešenju primenjen je kod sa sajta [1] .

Kada se na originalnu sliku primeni Haough transformacija, dobije se slika koja je prikazana ispod:




Takođe, korištena je i **Blobs detection tehnika**.

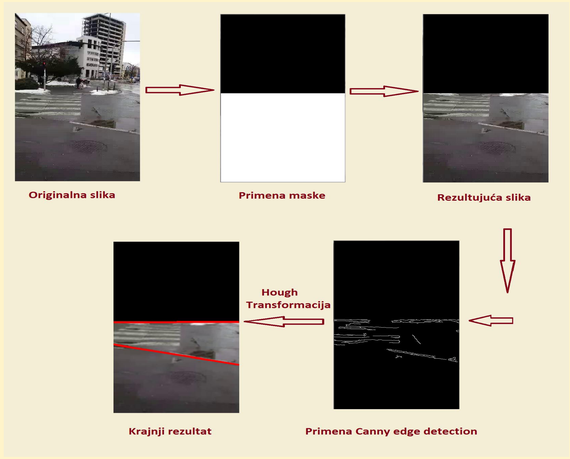
**Blob** predstavlja grupu povezanih piksela na slici koji dele neko svojstvo (npr. grayscale vrednost, itd.). OpenCV nudi pogodan način za detekciju blob-ova i njihovo filtriranje na osnovu različitih karakteristika, kao što su boja, veličina, oblik...Na sledećoj slici tamni povezani regioni su blob-ovi, dok je cilj blob detection tehnike da pomenuti blob-ovi kao regioni budu identifikovani i označeni.



Prethodni korak nije doveo do željenog rezultata, iz razloga što je Hough transformacija prepoznala previše linija, pa se moralo pribeći korišćenju maske kako bi se uklonili nepotrebni delovi slike koji su uzrokovali detekciju velikog broja linija primenom Hough transformacije. Na taj način izdvojen je samo deo od interesa na slici, slika je konvertovana u crno belu sliku, pa je na nju primenjen Canny edge detektor kako bi se izdvojile sve ivice na slici. Na sliku dobijenu pomoću Canny edge detektora primenjena je Hough transformacija kako bi se detektovale ivice (linije) puta.

Nakon navedenog postupka, dobijene su tražene ivice tj. linije puta i zatim tu iscrtane na originalnoj slici.

Prethodni postupak prikazan je na sledećoj slici.



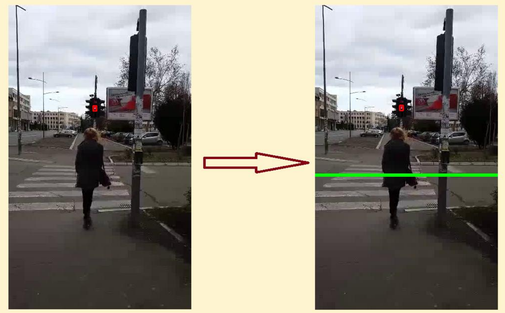

Međutim primenjene metode nisu dobro „radile“ u dovoljno velikom broju slučajeva, jer primenom Hough transformacije nisu se uvek dobijale dve linije, nekad detektovan veći broj linija, a nekad nije dobijena nijedna linija. Prethodno navedeno je bilo uzrok da ovaj pristup problemu bude odbačen i da se nastavi sa radom u **Python** programskom jeziku.

### Implementacija u Python programskom jeziku

Kao što je prethodno i spomenuto, korištena je **Template matching** tehnologija u okviru Python programskog jezika i pomoću nje je omogućeno prepoznavanje semafora na slici.

S obzirom da su korišteni snimci napravljeni sa mesta koje je veoma blizu semafora i pešačkog prelaza koji se posmatra, ta informacija je iskorištena na način što je na osnovu pozicije semafora detektovana približna pozicija sredine pešačkog prelaza.

Na osnovu dobijene pozicije sredine semafora, određena je jednačina prave koja predstavlja sredinu pešačkog prelaza i nakon toga je data jednačina iscrtana na frejmovima, što je prikazano na slici ispod.



## Detekcija pešaka

Implementacija ovog koraka je pokušana na više načina.

Prvo čemu je pribegnuto jeste korišćenje **Background subtractor MOG** metode, kao i **Background subtractor MOG2** metode, a razlika među njima je korištenje određenih dodatnih thresholda.

To su širokorištene tehnologije za generisanje crne maske, odnosno binarna slika sadrži piksele koji pripradaju pokretu objekata u datoj sceni koja se posmatra, ali pomoću statičke kamere.

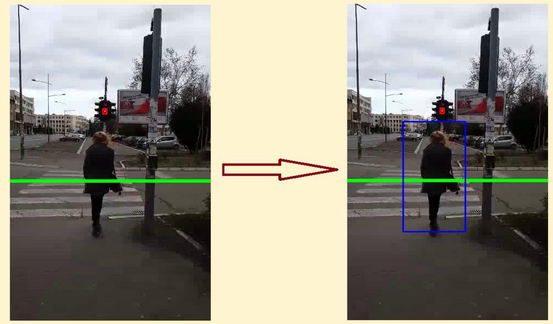
To je upravo bio ključ zbog čega se u ovom projektu ova tehnika nije pokazala kao dobro rešenje. Naime, snimci pravljeni pomoću kamere mobilnog telefona, koji su snimljeni držeći telefon u ruci, su pokazali da ova metoda u navedenim uslovima snimanja nije mogla dobro da uoči pokrete, odnosno, dobijalo se rešenje koje je pokazalo da je za ova metoda skoro sve na frejmovima prepoznala kao pokret, što je i logično pošto ruka nije dovoljno statička. S obzirom na navedene razloge, moralo se odustati od ove metode.

**Haar Cascade Classifiers** metoda je sledeća bila na redu za korištenje da bi se omogućila detekcija pešaka.

Ona radi po principu primene različitih klasifikatora koji se primenjuju na slici jedan po jedan. Ukoliko odredjeni deo slike ne prodje kroz neki klasifikator, onda se taj deo slike i odbacuje kao rešenje. Na kraju se dobije rezultat koji predstavlja deo slike koji je prošao kroz svaki klasifikator.

U ovom rešenju su prvo primenjeni klasifikatori za gornji deo tela, zatim za donji, međutim nisu se pokazali kao dovoljno dobri , stoga su uzeti i klasifikatori za prepoznavanje celog tela,.

Klasifikatori za prepoznavanje celog tela su dali dovoljno dobre rezultate. Na sledećoj slici je prikazan rezultat prepoznavanja pešaka na jednom frejmu primenom ovog klasifikatora.



## Prebrojavanje pešaka koji su prešli na crveno

Za ovaj korak je iskorištena dobijena linija, kao i njena jednačina. Sa druge strane, iskorištena je i pozicija prepoznatog pešaka, odnosno njegov uokvireni deo, gde je uzeta tačka iz donje četvrtine pravougaonika, u delu sredine nogu pešaka jer su te tačke bile najlogičnije s obzirom na pozicije pešaka i pešačkog prelaza.

Dobijena tačka je uvrštena u jednačinu prave, nakon čega se vršila provera. Ukoliko provera pokaže da je jednačina zadovoljena, to znači da je pešak prešao na crveno svetlo, nakon čega se broj pešaka koji su prešli na crveno svetlo uvećava.

## Zaključak

Na osnovu svega navedenog, zaključujemo da je detekcija semafora sa upaljenim crvenim svetlom uspešno obavljena primenom Template matching tehnike i u C++ i u Python programskom jeziku.

Kada je u pitanju detekcija puta, zaključujemo da rezultati Hough transformacije u velikoj meri zavise od lokacije na kojoj je postavljen semafor, kao i od vidljivosti ivica puta, pa se ovaj algoritam pokazao kao ne previše koristan. U ovom radu, detekcija puta je određena pomoću pozicije semafora koji je detektovan na slici tako što je određena približna pozicija sredine pešačkog prelaza. Na taj način se mogu detektovati pešaci koji prelaze put.

Kada je u pitanju detekcija pešaka, Background subtractor MOG i Background subtractot MOG2 tehnologije se nisu pokazale kao pogodne za detekciju pešaka iz razloga što su snimci korišteni u ovom radu snimljeni mobilnim telefonom, dok vremenski uslovi takođe nisu bili odgovarajući, pa se na osnovu prethodno navedenog može zaključiti da je slika dobijena primenom navedene dve tehnologije lošeg kvaliteta iz pomenutih razloga, kao i da bi rezultati bili mnogo kvalitetniji i precizniji da su korišteni snimci snimljeni stacionarnom kamerom. Na slici je bilo previše šumova i iz tog razloga ove dve tehnologije nisu dovele do željenog rezultata. Detekcija pešaka je ostvarena primenom Haar Cascade Classifiers metode koja radi po principu različitih klasifikatora koji se primenjuju na slici jedan po jedan. Ova metoda se pokazala kao najbolja kada je u pitanju detekcija pešaka u ovom radu.

S obzirom da su svi korišteni algoritmi i tehnologije u ovom radu testirani na skupu od 3 video snimka, budući rad na ovom problemu će se bazirati na proširenju domena problema i isprobavanju što više metoda i tehnologija kako bi se došlo do optimalnog rešenja.

## Literatura i reference

Literatura korišćena u ovom radu može se pronaći na sledećim linkovima:

1. http://www.keymolen.com/2013/05/hough-transformation-c-implementation.html
2. http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html
3. https://www.youtube.com/watch?v=8-3vl71TjDs
4. http://docs.opencv.org/2.4.10/doc/tutorials/imgproc/threshold/threshold.html
5. http://alereimondo.no-ip.org/OpenCV/34/
6. http://alereimondo.no-ip.org/OpenCV/34/

I ostali materijali sa interneta.