SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

**DETEKCIJA LINIJA NA CESTI**

Seminarski rad

David Hodak

Osijek, 2020.

SADRŽAJ

[1. UVOD 1](#_Toc64835867)

[1.1. Pregled područja rada 1](#_Toc64835868)

[2. OBRADA SLIKE 2](#_Toc64835869)

[2.1. Model boja 2](#_Toc64835870)

[2.1.1. Hsl model boja 2](#_Toc64835871)

[2.2. Područje interesa na slici 6](#_Toc64835872)

[2.3. Grayscale 7](#_Toc64835873)

[2.4. Detekcija rubova 8](#_Toc64835874)

[2.4.1. Gaussov filtar 8](#_Toc64835875)

[2.4.2. Pronalaženje gradijenta intenziteta slike 9](#_Toc64835876)

[2.5. Houghova transformacija 10](#_Toc64835877)

[3. OBJAŠNJENE PROGRAMSKOG RJEŠENJA 13](#_Toc64835878)

[3.1. Problemi 14](#_Toc64835879)

[4. ZAKLJUČAK 15](#_Toc64835880)

[5. LITERATURA 16](#_Toc64835881)

# UVOD

Detekcija linija na cesti poznati je problem koji je već kvalitetno riješen te se konstantno unaprjeđuje s obzirom na sve veću popularnost autonomnih vozila. U ovom projektu bilo je potrebno implementirati metodu za detekciju srednje i rubne linije koristeći OpenCV datoteke i prilagoditi rješenje za video sekvencu. S obzirom da nije bilo dozvoljeno koristi neuronske mreže, rješenje se bazira na pronalaženju odgovarajućeg modela boja za prepoznavanje linija i primjene Houghove transformacije. Programsko rješenje je napisano u Python programskom jeziku.

## Pregled područja rada

Autonomna vozila postaju dio svakodnevnice, te sa svakom godinom vidimo sve veći napredak u tom području tehnologije. Za to je i zaslužan sve veći broj korisnika automobila ali i nažalost sve veći broj nesreća godišnje. S obzirom da je ovo područje dosta kompetentno, teško da ćemo pronaći gotova izvješća kako rade autonomna vozila Tesle, BMW-a ili Mercedesa. Ono što znamo je da takva vozila koriste određene radare, senzore za vid itd. Primjerice, Lidar senzor[7] koji baca laserske zrake na objekt i mjeri udaljenost odbijenog lasera natrag do izvora. Nešto napredniji Lidar sutavi mogu pružiti sliku okoline koju skenira. Dobro djeluje na velike udaljenosti.

Za pronalaženje ravnih linija koristi se detekcija rubova i Houghova transformacija. Ostale metode koje se temelje na značajkama poput Haar-like feature, neuronske mreže, HOG(Histogram of oriented gradienst) te SVM(support vector machine) su dostojne detektirati objekte i cestu s visokom preciznošću te se koriste u realnim uvjetima no same po sebi nisu dovoljne za kompletan sustav pomoći u vožnji.

Što se tiče sustava za pomoć u vožnji jedan takav je i LDWS ili sustav upozorenja napuštanja trake. Kao što samo ime kaže, on upozorava vozača ako se vozilo počne kretati izvan vlastite linije.  
Sistem je nastao još 2000 godine od strane tvrtke Iteris koja je ugradila sustav u komercijalne kamione. Radi na principu Houghove transformacije i Canny detekcije rubova gdje se slike dohvaćaju iz kamera postavljenih na prednjoj i stražnjoj strani. Naravno, ovaj sustav je s godinama napredovao te postoje dodaci gdje se sustav brine da auto stalno ostaje u svojoj traci.[8]

# OBRADA SLIKE

Rastav problema detekcije linija u ovom projektu možemo podijeliti u šest značajki:

1. Primjena odgovarajućeg model boja
2. Označiti regiju interesa na slici
3. Sliku pretvoriti u sive tonove
4. Provesti detekciju rubova
5. Provesti Houghovu transformaciju

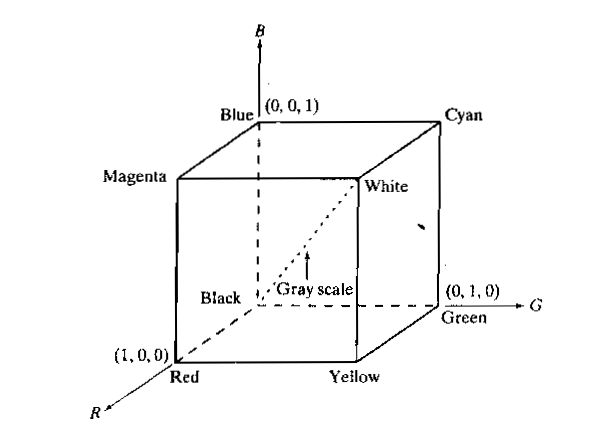
## Model boja

Linije na cesti su obično bijele ili žute boje te je potrebno pronaći odgovarajući model boja u kojoj bi bilo najlakše napraviti filter za te boje. To često nije jednostavan zadatak jer intenzitet boja varira pri vanjskim utjecajima, primjerice, utjecaj sunca, utjecaj prljavštine na cesti, kiša, oblačan dan, sjena drveća itd. Preuzimanje video sekvenci s interneta dodatno komplicira ovaj zadatak jer nisu svi snimljeni s istom kamerom, odnosno jednakom kvalitetom te će često jedan prostor boja dobro odgovarati za jedan video, a za drugi neće. Kako bi se ovo pojednostavilo, korištena su tri videa snimljena istom kamerom. U praksi se koristi više modela boja pa ih se uspoređuje no ovdje je korišten samo jedan koji se pokazao najboljim, a to je HSL.

### Hsl model boja

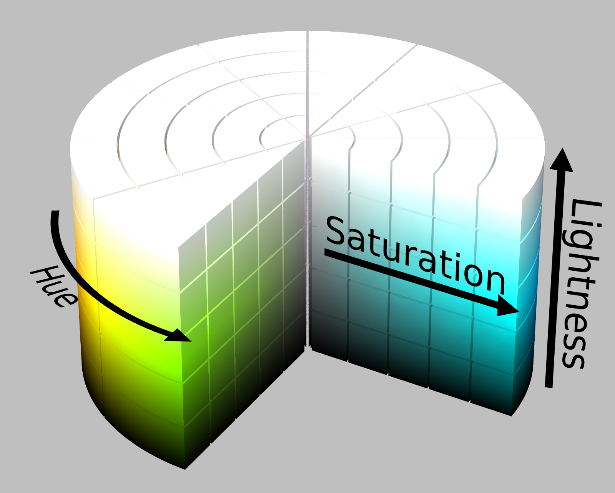
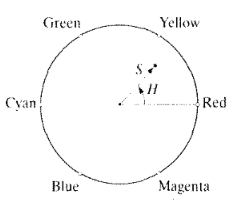
U RGB modelu svaka boja se pojavljuje kao primarna spektralna komponenta crvene, zelene i plave boje. Model je baziran na Kartezijevom koordinatnom sustavu te se prema slici 2.1. može prikazati kao kocka gdje vrhovi predstavljaju određene boje. Radi jednostavnijeg prikaza vrijednosti su normalizirane u raspon [0,1]. Različite boje su predstavljene kao točke unutar kocke i definirane kao vektori koji se pružaju od početne vrijednosti.

Slike predstavljene u RGB modelu sastoje se od tri komponente slike koje se kombiniraju kada ih se prikazuje na monitoru. Broj bitova koji predstavlja svaki piksel u RGB prostoru naziva se dubina piksela. Ako imamo 8 bitnu sliku tada svaki piksel ima dubinu od 24 bita ( 8 bita za svaku od vrijednosti R, G, B kanala).



**Slika 2.1.** Prikaz RGB modela boja u Kartezijevom koordinatnom sustavu

HSL je alternativna reprezentacija RGB modela boja. Dok je RGB dobar za računalni prikaz, nije prilagođen za opis boja kako ih ljudi interpretiraju. Opisujući boju s nijansom(eng. *hue*), zasićenjem (eng*. saturation*) i intenzitetom (eng. *lightness* ili *brightness)* bliže smo opisu boje kako bi ju čovjek opisao. HSL je predstavljen kao cilindar (slika 2.2.) s nijansom koja se očitava u stupnjevima počinjući od 0° (crvena), prolazeći kroz 120°(zelena) i 240°(plava) te opet u 0°.

**Slika 2.2.** HSL cilindar

Zasićenje je dužina vektora od početne točke do neke druge točke, odnosno udaljenost od okomite osi. Ishodište (eng. *origin*) je definirano kao sjecište ravnine boja s okomitom osi intenziteta(svjetline). Možemo zaključiti da su tri bitne komponente HSL modela boja duljina vektora do određene točke, kut koji taj vektor zatvara s referentnom osi (crvena boja) i os intenziteta.

Ako imamo sliku u RGB formatu, komponentu H svakog piksela možemo dobiti prema formuli

gdje je kut

.

Zasićenje se može izračunati prema

.

Svjetlina prema

.

Pretpostavlja se da su RGB vrijednosti normalizirane u [0,1] i da se kut promatra u odnosu na referentu os.

Na slici 2.3. možemo vidjeti kako treći kanal HSL slike(svjetlina) dobro izdvaja svjetlije boje što nam pomaže pri detekciji žutih i bijelih linija. No to funkcionira dok nema utjecaja vanjskih elemenata kao što je sunce (slika 2.4.).



**Slika 2.3.** HSL kanal svjetline



**Slika 2.4.** Utjecaj sunca

Za filtriranje žute i bijele boje (slika 2.5.) izabran je HSL model boja jer je intuitivan za pronaći niske i visoke granice žute i bijele boje, ali i ukloniti donekle smetnje sunčevih zraka jer možemo utjecati na svjetlinu i zasićenje slike.



**Slika 2.5**. Maska za nijanse žute i bijele boje

## Područje interesa na slici

Za detekciju linija potrebna nam je samo cesta, okolni dijelovi slike se mogu ukloniti. Na taj način dobivamo regiju interesa (eng. reign of interest) (slika 2.6.) na kojoj će se obavljati ostale operacije algoritma. Osim što se ubrzava proces obrade slike, uklanja se i utjecaj okoline gdje bi Houghova transformacija zasigurno pronašla mnoštvo linija. Uvjet je da kamera bude fiksna te će se područje ceste uvijek nalaziti između istih točaka. U praksi se detekcija ne oslanja samo na kameru koja snima cestu već i na mnoštvo drugih senzora jer je ovakav princip nepouzdan za primjerice, nizbrdice, uzbrdice i oštre zavoje.



**Slika 2.6.** Područje interesa na slici

## Grayscale

Grayscale slika predstavlja sliku koja se sastoji samo od sivih tonova. Razlog zašto bi uopće koristili takvu sliku je što je za svaki piksel potrebno pružiti puno manje vrijednosti. Siva boja je ona koja kod crvene, zelene i plave komponente ima isti intenzitet te je potrebno navesti samo jednu vrijednost za svaki piksel, a ne tri kao što je potrebno kod slike u boji. Najčešće se intenzitet sive pohranjuje kao 8-bitni cijeli broj koji daje 256 mogućih različitih nijansi sive od crne(0) do bijele(255). Ovakve slike su dovoljne za mnoge zadatke, pa nema potrebe za korištenjem složenijih i teže obradivih slika u boji. Iz istog razloga se koristi i u ovom projektu. Nakon što se pronašla maska za žutu i bijelu boju primjenjuje se pretvorba slike u sliku sivih tonova gdje ćemo dobiti svjetlije tonove, odnosno nijanse više prema bijeloj gdje su prethodno bile žuta i bijela boja, a sve ostalo će ostati u tamnijim tonovima, odnosno crnoj. Time se dobiva matrica s brojem redova i stupaca koji predstavljaju veličinu slike s vrijednostima od 0 do 255, što predstavlja računalu mnogo lakšu i bržu obradu podataka, a za problem detekcije linija je dostatno.



**Slika 2.7.** Grayscale slika

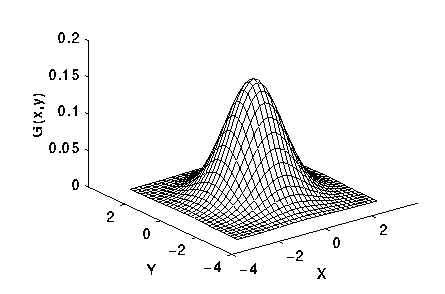
## Detekcija rubova

Za detekciju rubova (slika 2.9.) koristit će se Canny algoritam. Razvijen je 1986. godine od John F. Cannya. S obzirom da je detekcija rubova osjetljiva na šum u slici, moramo primijeniti Gaussov filtar.

### Gaussov filtar

Gaussov filtar je niskopropusni filtar koji se koristi za zamućivanje slika i uklanjanje detalja i šuma. Šum se uglavnom sastoji od visokih frekvencija koje Gaussov filtar ne propušta. Jezgra filtera se popunjava s vrijednostima koje su izračunate po 2D Gaussovoj funkciji (slika 2.8):

Gdje je:  
- – standardna devijacija Gaussove funkcije  
- x – udaljenost od centralnog piksela po x osi  
- y – udaljenost od centralnog piksela po y os



**Slika 2.8.** Gaussova funkcija

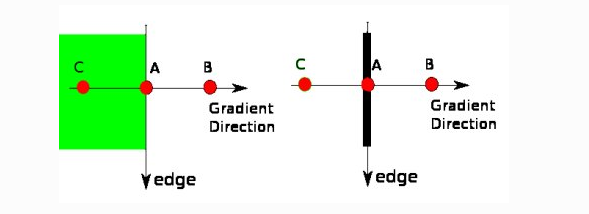
S obzirom da je slika spremljena kao skup diskretnih piksela trebamo uraditi diskretnu aproksimaciju Gaussove funkcije prije nego se primjeni konvolucija. Gledajući teoriju, Gaussova razdioba imati će vrijednost različitu od nula u svakoj točki što bi zahtijevalo beskonačno veliki konvolucijski kernel, no u praksi sve više od 3 od središnje vrijednosti se može smatrati nulom pa se i kernel može smanjiti. Nakon što je pronađen odgovarajući kernel provodi se konvolucija. 2D Gaussova funkcija se dijeli na x i y komponente te se provodi 1D Gaussova funkcija u x smjeru te zatim isti postupak u y smjeru.

### Pronalaženje gradijenta intenziteta slike

Zaglađena slika je filtrirana sa Sobelovim kernelom u okomitom i vertikalnom pravcu kako bi se dobile derivacije u horizontalnom smjeru () i vertikalnom smjeru (. Koristeći te dvije slike možemo pronaći gradijent i smjer svakog piksela:

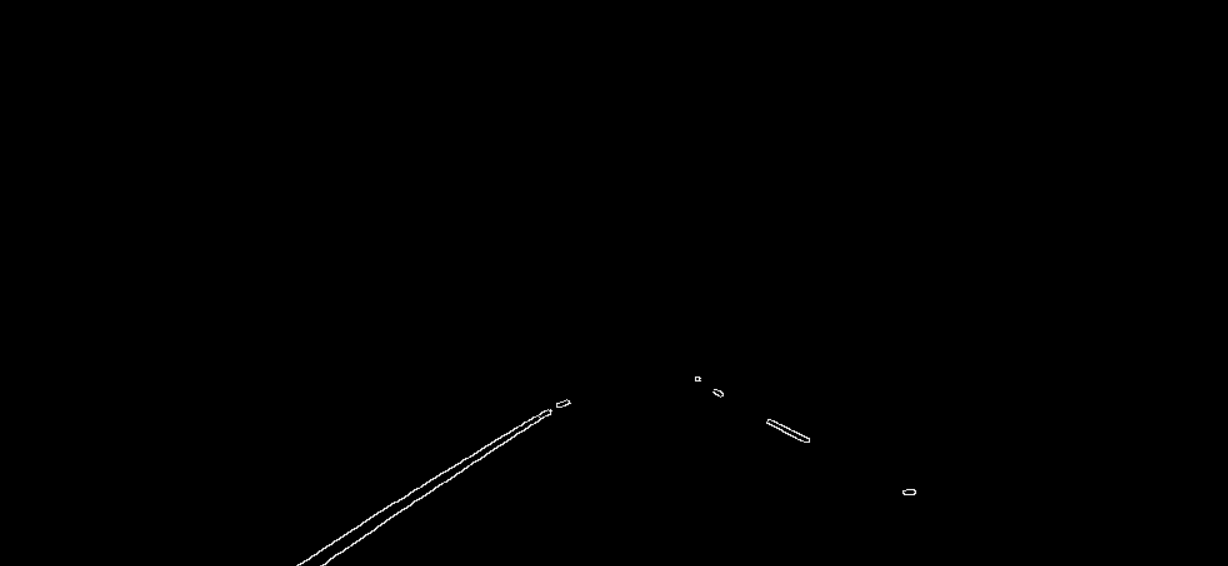
Kut(

Smjer gradijenta je uvijek okomita prema rubovima. Nakon dobivanja veličine i smjera gradijenta, pretražuje se slika kako bi se uklonili svi neželjeni pikseli koji možda ne predstavljaju rub. Da bi se to utvrdilo, za svaki piksel provjerava se je li on lokalni maksimum u svom susjedstvu i nalazi se u smjeru gradijenta.



**Slika 2.8.** Provjera piksela

Točka A nalazi se na rubu, te je rub okomit na smjer gradijenta. Točke B i C se nalaze u smjeru gradijenta. Zatim se točka A, zajedno s B i C provjerava je li tvori lokalni maksimum. Ako da, ostavlja se netaknuta, u suprotnom se postavlja na 0. U sljedećom koraku odlučuje se koji su rubovi stvarno rubovi, a koji nisu. Za to nam trebaju dvije granične vrijednosti, minimalna i maksimalna. Svi rubovi s gradijentom intenziteta većim od maksimalne granične vrijednosti su sigurno rubovi, a oni ispod minimalne granične vrijednosti se odbacuju. Svi pikseli koji se nalaze između graničnih vrijednosti klasificirani su kao rubovi s obzirom na povezanost s pikselima koji su sigurno rubovi. Ako su povezani, smatra ih se rubovima, u suprotnom se isto odbacuju.

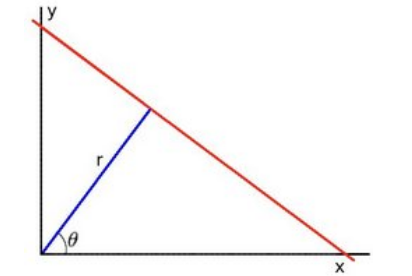


**Slika 2.9.** Izgled detekcija rubova na linija ceste

## Houghova transformacija

Houghova transformacija je tehnika koja se koristi za izoliranje određenog oblika unutar slike. Klasična Houghova transformacija najčešće se koristi za otkrivanje pravilnih krivulja kao što su crte, krugovi, elipse itd. Glavna prednost ove tehnike je tolerancija na praznine u obilježjima i na nju relativno ne utječe šum slike.

Linija u prostoru može biti opisana s dvije varijable. U Kartezijevom koordinatnom sustavu to su k(nagib) i c(točka gdje linija sječe y os). U polarnom sustavu to su r(udaljenost linije od ishodišta) i θ(kut koji okomica na liniju zatvara s x osi). S obzirom da za vertikale linije nagib teži k beskonačnosti koristi se prikaz linije u polarnom koordinatnom sustavu.



**Slika 2.10.** Prikaz linije u polarnom koordinatnom sustavu

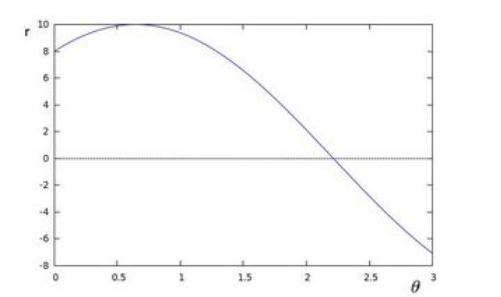
Formula kojom se opisuje takva linija je :

r = x cos θ + y sin θ.

Za svaku točku ( možemo definirati beskonačno linija koje prolaze kroz nju kao:

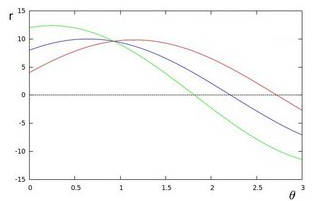
.

To znači da svaki par ( predstavlja jednu liniju koja prolazi kroz (. Ako za danu točku prikažemo cijelu familiju linija koje prolaze kroz nju, dobijemo sinusoid na grafu (slika 2.11).



**Slika 2.11.** Sinusoid

Koristeći isti postupak na sve točke možemo zaključiti ako se krivulje dvije ili više točaka sjeku u ravnini , to znači da te točke pripadaju istoj liniji.



**Slika 2.12.** Sjecište krivulja u ravnini

Na ovakav način radi Houghova transformacija. Prati sjecišta krivulja za svaki piksel slike. Ako je broj sjecišta iznad parametra minimalnog broja sjecišta potrebnog za detekciju linija, tada ju deklarira kao liniju s parametrima () točke sjecišta.

Nakon svih prijašnjih metoda obrade slike, na sliku s detektiranim rubovima je primijenjena Houghova transformacija te se rezultat može vidjeti na slici 2.13.



**Slika 2.13.** Rezultat Houghove transformacije

# OBJAŠNJENE PROGRAMSKOG RJEŠENJA

Programsko rješenje je pisano u Python programskom jeziku koristeći OpenCV biblioteku za rad sa slikama.

def processImage(image):  
 imgMasked = filterWhiteYellow(image)  
 vertices = getVertices(imgMasked)  
 roiImage = get(imgMasked, vertices)  
  
 grayImage = cv2.cvtColor(roiImage, cv2.COLOR\_BGR2GRAY)  
 gaussianImage = cv2.GaussianBlur(grayImage, (7, 7), 0)  
 cannyImage = cv2.Canny(gaussianImage, threshold1=50, threshold2=150)  
  
 houghLines = cv2.HoughLinesP(cannyImage, rho=1, theta=np.pi/180, threshold=20, minLineLength=20, maxLineGap=300)  
  
 lineImage = np.zeros((image.shape[0], image.shape[1],3), dtype=np.uint8)  
  
 lineImage = findAverageLines(lineImage, houghLines)  
  
 output = cv2.addWeighted(lineImage, 1, image, 1, 0.)  
  
 return imgMasked, grayImage, cannyImage, roiImage, output

**Slika 3.1.** Glavna funkcija za detekciju linija

Proces obrade slike ide redoslijedom kako je navedeno na početku. Funkcija *filterWhiteYellow* prebacuje BGR format u HLS format te pronalazi maske za žutu i bijelu boje koje onda ujedinjuje u jednu masku s *bitwise\_or* funkcijom. Za rezultat vraća sliku gdje su prikazane samo žuta i bijela boja. Funkcija *getROI* postavlja željeno područje prema predanim točkama kao jedino vidljivo, dok je sve ostalo crno. Sljedeće se slika prebacuje u grayscale sliku, primjenjuje se Gaussov filtar te se radi detekcija rubova. Završno na detektirane rubove primjenjuje se vjerojatnosna Houghova transformacija koja radi na istom principu kao i klasična Houghova transformacija no nešto je efikasnija jer nam u parametrima omogućava da postavimo minimalan broj točaka koje tvore liniju i minimalnu vrijednost praznine između dvije točke što će se smatrati istom linijom, a u detekciji linija na cesti je izrazito korisno s obzirom da srednje linije obično imaju praznine. Kako ova funkcija vraća mnoštvo linija, bilo je potrebno naći aproksimaciju za linije koje predstavljaju desnu i lijevu cestovnu liniju. Funkcija *findAverageLines* osnovu nagiba linija pronalazi linije koje se nalaze na lijevoj i desnoj strani, te radi aproksimaciju tih linija s prednosti onih linija koje su duže, kako bi pronašla dvije linije koje najbolje predstavljaju skup lijevih i desnih linija te ih ocrtava na slici.

## Problemi

Ovaj algoritam dobro radi detekciju linija na ravnoj cesti ili na cesti s blagim zavojima, primjerice autoput. Kod oštrih zavoja pojavljuju se problemi što je i očekivano jer Houghova transformacija pronalazi ravne linije. Vrlo je važan i utjecaj kvalitete kamere, vrijednosti kojima očitavamo žutu i bijelu boju možda neće biti iste ako koristimo različite kvalitete video sekvenci. Isto tako i utjecaj sjena, sunca, kiše, snijega itd. uveliko otežava detekciju linija ako primjenjujemo samo ovaj tip detekcije.

# ZAKLJUČAK

Korištenjem nekoliko metoda obrade slike kako bi pripremili sliku za Houghovu transformaciju dovelo je do rješenja koje zadovoljava za specifične uvjete, no nije primjereno za realnu uporabu. Unatoč tome, ovakvo rješenje je prije svega edukacijskog karaktera gdje se na realnom problemu dobiva osnovno znanje o obradi slike i funkcioniranju metoda koje se može translatirati na korištenje kompliciranijih i zahtjevnijih procesa obrade slike, bilo za sličan problem ili za nešto drugo.

# LITERATURA

[1] R. C. Gonzalez, R. E. Woods: Digital Image Processing. Addison-Wesley, Second Edition, 2002. str. 295 – 302.

[2] <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html>

[3] <https://en.wikipedia.org/wiki/HSL_and_HSV>

[4] Matija Forko, *Nefotorealistična tehnika prikaza video zapisa*, Završni rad, Fakultet elektrotehnike i računarstva, Zagreb 2009.

[5] Maja Šverko, Houghova transformacija, Zavrspni rad, Fakultet elektrotehnike i računarstva, Zagreb 2009.

[6] <https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html>

[7] <https://www.sciencedirect.com/topics/engineering/optical-radar>

[8] <https://en.wikipedia.org/wiki/Lane_departure_warning_system>