Metode segmentacije objekata i područja u digitalnoj slici

Danilović, Nikola

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

METODE SEGMENTACIJE OBJEKATA I PODRUČJA U DIGITALNOJ SLICI

Završni rad

Nikola Danilović

Osijek 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Nikola Danilović
Studij, smjer:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4640, 16.09.2019.
OIB studenta:	16953593304
Mentor:	Prof.dr.sc. Snježana Rimac-Drlje
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Metode segmentacije objekata i područja u digitalnoj slici
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	07.09.2020.
Datum potvrde ocjene Odbora:	11.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 14.09.2020.

Ime i prezime studenta:	Nikola Danilović
Studij:	Preddiplomski sveučilišni studij Elektrotehnika i informacijska tehnologija
Mat. br. studenta, godina upisa:	4640, 16.09.2019.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Metode segmentacije objekata i područja u digitalnoj slici**

izrađen pod vodstvom mentora Prof.dr.sc. Snježana Rimac-Drlje

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Digitalna slika je relativno nov izum koji se polako razvijao u drugoj polovici 20. stoljeća, u isto vrijeme kada su i mnoge druge digitalne tehnologije nastajale. U samim početcima digitalna slika je bila novitet koji je služio isključivo u znanstvene i istraživačke svrhe, no daljnjim napredovanjem tehnologije došlo je do postepenog smanjenja cijene digitalnih fotoaparata što je pridonijelo komercijalnoj upotrebi istih. Razvitkom računala i programa za obradu digitalne slike pronalazila se sve veća korist u digitalnim slikama, ne samo u industriji, gdje je digitalna slika omogućavala veći stupanj automatizacije i nadziranja industrijskog procesa, već i u umjetnosti, ali i u brojnim drugim primjenama. U većini tih primjena jedna od temeljnih metoda obrade digitalne slike je bila, a i dalje je, segmentacija. Pomoću segmentacije slika se može rastaviti na više regija, ili segmenata, na temelju nekog svojstva koje pojedine regije imaju.

U ovom završnom radu obrađeni su algoritmi pojedinih metoda segmentiranja te usporedba njihovih primjena na odabranom setu problema u Python programskom jeziku i OpenCV programskoj biblioteci. Nakon prva dva uvodna poglavlja razrađene su pojedine metode amplitudne segmentacije, teorija iza pripadnih algoritama te primjer primjene istih. U trećem poglavlju obrađene su sljedeće metode amplitudne segmentacije: amplitudna segmentacija s manualnim postavljanjem praga, amplitudna segmentacija Otsu metodom, adaptivna amplitudna segmentacija i amplitudna segmentacija digitalnih slika u boji. U četvrtom poglavlju obrađena je segmentacija na temelju rubova, objašnjena je teorija iza pripadnih segmentacija i prikazana primjena istih na računalu. Pojedine metode detekcije rubova pomoću kojih se može segmentirati digitalna slika i koje su razrađene u pripadnom poglavlju su: konvolucija digitalne slike sa Sobelovim i Prewittovim operatorom i *Canny* algoritam za detekciju rubova. Nakon toga je obrađena teorija i primjena segmentacije koja koristi digitalnu sliku s detektiranim rubovima te ju na temelju tih rubova segmentira. Riječ je o Houghovoj transformaciji, čijom primjenom se mogu detektirati pravci i kružnice pomoću kojih se vrši pripadna segmentacija. Sljedeća dva poglavlja se bave teorijskom pozadinom i primjenom segmentacije na temelju rasta područja i segmentacije grupiranjem. Na kraju, prije zaključka, u sedmom poglavlju dani su rezultati testiranja pojedinih obrađenih metoda na odabranom setu problema u cilju izdvajanja pozitivnih i negativnih karakteristika svake metode. Na temelju dobivenih rezultata testiranja napravljena je usporedba metoda. U zadnjem poglavlju dana su zaključna razmatranja.

1.1. Zadatak završnog rada

Razvoj algoritama za obradu slika kao i pristupačnost video kamera, omogućili su korištenje kamera kao senzora u mnogim primjenama kao što je autonomna vožnja, nadzor prometa, praćenje i prepoznavanje ljudi te mnoge druge. Ovisno o primjeni videa, iz slike se izdvajaju određene značajke. Jedan od važnih postupaka za izdvajanje značajki je segmentacija slike. U radu je potrebno opisati amplitudnu segmentaciju, segmentaciju na temelju rubova, segmentaciju na temelju rasta područja i segmentaciju grupiranjem. U programu Python napraviti algoritme za segmentiranje za svaku od opisanih metoda te napraviti usporedbu metoda na odabranom setu problema.

2. SEGMENTACIJSKE METODE U NOVIM TEHNOLOGIJAMA

U današnjem modernom dobu segmentacijske metode se koriste u više znanstvenih disciplina, najčešće kako bi se određeni objekti od interesa detektirali i razlikovali. Jedno od takvih područja primjene je medicinsko snimanje ili oslikavanje. U takvoj primjeni često se koriste segmentacijske metode obrade digitalne slike u svrhu lociranja tumora [1] te dijagnosticiranja i proučavanja anatomije [2]. Također, segmentacijske metode se koriste i za detektiranje objekata u prometu kao što su ceste [3] ,vozila [4] i neki drugi objekti koji se mogu pojaviti u prometu, a koji su od velike važnosti za njegovo pravilno funkcioniranje. Osim detekcije, segmentacijske metode se mogu koristiti i za prepoznavanje određenih značajki na slici zbog čega se koriste u svrhu prepoznavanja lica [5] i otisaka prstiju [6].

U ovom radu naglasak će biti na slikama prometa te će se stoga koristiti određene segmentacije u tu svrhu. Na primjer, za segmentiranje ceste koristit će se segmentacija na temelju rubova i segmentacija na temelju rasta područja dok se za istu svrhu koriste neke kompliciranije metode koje se temelje na konvolucijskim neuronskim mrežama [3]. Uz ceste, u ovom radu će se pokušati segmentirati i vozila, za što će se koristiti metoda amplitudne segmentacije i segmentacija grupiranjem pomoću kojih će se segmentirati vozila na temelju njihove boje. Za neke općenite slučajeve, kada se vozila ne ističu na slici zbog boje koja je slična pozadini, mogu se koristiti složenije metode kao što je segmentiranje na temelju pokretnih dijelova slike pri analizi video okvira. [4]

3. AMPLITUDNA SEGMENTACIJA

Amplitudna segmentacija je jedna od najkorištenijih i najintuitivnijih metoda segmentacije digitalne slike. To je zato što ima određena svojstva poput jednostavnosti implementacije i relativno velike brzine izvođenja na računalu. [7, str.760] Zbog tih svojstava se može koristiti u raznim slučajevima, npr. kada se traže određeni objekti slike koji su različito osvjetljeni ili kada se traže segmenti slike s određenim spektrom boja.

Za amplitudnu segmentaciju vrijede određena osnovna pravila koja se odnose i na ostale segmentacijske metode, a to su:

$$\bigcup_{i=1}^{n} R_i = R \tag{3-1}$$

gdje je: R – skup koji predstavlja cijelu sliku te se sastoji od unije svih segmenata slike R_i. Segmentni skup R_i je povezan skup, što znači da točke skupa moraju biti povezane kroz 4 ili 8 susjednih točaka. [7, str.760, str.90]

$$R_i \cap R_j = \emptyset, \qquad i \neq j, \qquad \forall i, j \tag{3-2}$$

gdje su: R_i i R_j disjunktni skupovi koji predstavljaju pojedine segmente slike. To znači da pri segmentiranju slike ne smije biti presjeka između segmenata, tj. svaki dio slike smije biti sadržan u točno jednom segmentnom skupu.

$$Q(R_i) = 1, \qquad i = 1,2, \dots, n \tag{3-3}$$

gdje je: $Q(R_k)$ – logički predikat definiran nad skupom točaka u segmentnom skupu R_k. Taj logički predikat nad određenim segmentom R_i je jednak logičkoj jedinici. To znači da će uvjet određenog svojstva na temelju kojega segmentiramo sliku biti ispunjen na svim točkama pojedinog segmentnog skupa R_i.

$$Q(R_i \cup R_j) = 0 \tag{3-4}$$

gdje je: $Q(R_i \cup R_j)$ – logički predikat definiran nad susjednim segmentnim skupovima R_i i R_j uvijek ima vrijednost logičke nule jer susjedni segmentni skupovi da bi se mogli razlikovati moraju imati drugačija svojstva. [7, str.712]

3.1. Amplitudna segmentacija s manualnim postavljanjem praga

Amplitudna segmentacija s manualnim postavljanjem praga je jednostavna amplitudna segmentacija jer se u algoritmu te segmentacije ne računa neki optimalni prag na temelju histograma slike, već se manualno odabire pri zvanju pripadne funkcije. Manualno odabrani prag će segmentirati monokromatsku sliku na dva segmenta, jedan s elementima slike koji imaju vrijednosti veće od praga, a drugi s onim elementima koji imaju manje vrijednosti. Tim segmentima će se moći manipulirati korištenjem parametra funkcije *threshold_type*. OpenCV funkcija koja se koristi za manualno postavljanje praga glasi: *ret, thresh = cv2.threshold(image, threshold_value, max_value, threshold_type)*. Varijabla *ret* će kasnije biti od koristi, a varijabla *thresh* sadrži segmentiranu sliku. Parametar *image* je varijabla u kojoj se nalazi slika, *threshold_value* je vrijednost manualno odabranog praga, a parametar *max_value* je maksimalna vrijednost elementa slike u segmentaciji. Na slikama 3.1. i 3.2. je prikazan programski kod u Python programskom jeziku u kojemu se primjenjuje *threshold* funkcija za amplitudnu segmentaciju s manualnim postavljanjem praga te su na slici 3.3. prikazani rezultati tih funkcija.

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Učitavanje crno-bijele slike
img = cv2.imread('red_mazda.jpg', 0)
cv2.imshow('Originalna slika', img)

# Primjena praga tipa THRESH_BINARY. Sve ispod praga postaje 0, a sve iznad postaje 255
ret, prag1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
cv2.imshow('THRESH_BINARY', prag1)

# Primjena praga tipa THRESH_BINARY_INV. Sve ispod praga postaje 255, a sve iznad postaje 0. Obrnuto od THRESH_BINARY
ret, prag2 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)
cv2.imshow('THRESH_BINARY_INV', prag2)

# Primjena praga tipa THRESH_TRUNC. Sve iznad praga poprima vrijednost praga, a sve ispod praga ostaje kao i prije.
ret, prag3 = cv2.threshold(img, 127, 255, cv2.THRESH_TRUNC)
cv2.imshow('THRESH_TRUNC', prag3)

# Primjena praga tipa THRESH_TOZERO. Sve ispod praga poprima vrijednost 0, a sve iznad ostaje kao što je i bilo.
ret, prag4 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO)
cv2.imshow('THRESH_TOZERO', prag4)
```

Sl. 3.1. Prvi dio koda za amplitudnu segmentaciju s manualnim postavljanjem praga uz opis primjena pojedinih *threshold_type* parametara i koda za spremanje pojedinih slika

```
# Primjena praga tipa THRESH_TOZERO_INV. Sve iznad praga poprima vrijednost 0, a sve iznad praga ostaje kao i prije.
ret, prag5 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO_INV)
cv2.imshow('THRESH_TOZERO_INV', prag5)

cv2.waitKey(0)
cv2.destroyAllWindows()

titles = ['Originalna slika','THRESH_BINARY','THRESH_BINARY_INV','THRESH_TRUNC','THRESH_TOZERO','THRESH_TOZERO_INV']
images = [img, prag1, prag2, prag3, prag4, prag5]

for i in range(6):
    cv2.imwrite('../Slike/Manualni pragovi/' + titles[i] + '.jpg', images[i])

for i in range(6):
    plt.subplot(2,3,i+1)
    plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Sl. 3.2. Drugi dio koda za amplitudnu segmentaciju s manualno postavljenim pragom na 127

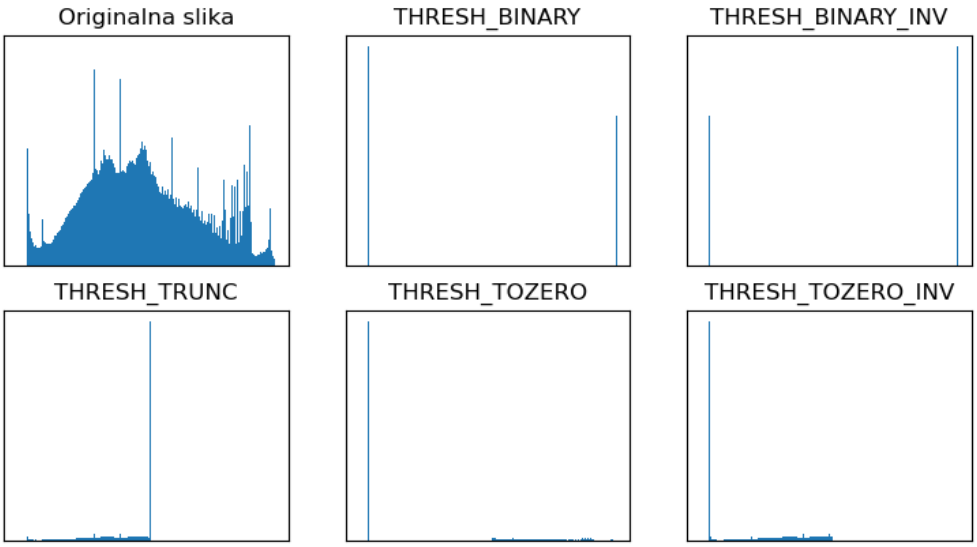
Sl. 3.3. Primjena amplitudne segmentacije s manualnim postavljanjem praga uz različite *threshold_type* parametre [8]

Kako bi se bolje prikazala promjena vrijednosti elemenata slike opisana u komentiranom kodu na slici 3.1. i 3.2. za svaki pojedini *threshold_type* parametar, može se napraviti histogram segmentiranih slika u kojima se može vidjeti koliko ima elemenata slike (označeno ordinatom) s pojedinim vrijednostima monokromatske slike (označenim apscisom) koje mogu poprimiti

vrijednost od 0 do 255. Za crtanje histograma koji su prikazani na slici 3.5. potrebna je matplotlib programska biblioteka i dodatan kod koji je prikazan na slici 3.4.

```
for i in range(6):
    plt.subplot(2, 3, i + 1)
    plt.hist(images[i].ravel(),256,[-10,256])
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

Sl. 3.4. Dodatni kod za iscrtavanje histograma



Sl. 3.5. Histogrami segmentiranih slika amplitudnom segmentacijom s manualnim postavljanjem uz različite *threshold_type* parametre

3.2. Amplitudna segmentacija Otsu metodom

Kod funkcije pomoću koje se primjenjuje amplitudna segmentacija s manualnim postavljanjem praga spomenuta je dodatna varijabla *ret* koja se u tom slučaju nije koristila jer ona jednostavno vraća prag koji je postavljen za segmentaciju. No ta se varijabla može koristit pri

segmentiranju slike pomoću Otsu metode. Tada može biti korisna jer Otsu metoda služi za određivanje optimalnog praga bimodalne slike (slike koja ima dva globalna maksimuma u histogramu) te pri korištenju te metode funkcija vraća *ret* varijablu koja je jednaka optimalnom pragu navedene slike. Optimalni prag se određuje u cilju da se slika razdvoji na dvije klase pri čemu je varijanca između klasa maksimalna [7, str.768]. Pripadni algoritam ove metode se može sročiti kroz slijedeće korake:

1. Računanje normaliziranog histograma pomoću formule:

$$p(i) = \frac{n_i}{MN} \tag{3-5}$$

gdje je: n_i – broj elementa slike razine i , a MN je ukupan broj elemenata slike. $p(i)$ je vjerojatnost slučajnog odabira elementa određenog intenziteta i koji može poprimiti vrijednosti u intervalu $[0, L-1]$.

2. Odabiranje praga k tako da vrijedi:

$$T(k) = k, 0 \ < k < L-1 \tag{3-6}$$

Na temelju tog praga se klasificiraju elementi slike u područje C_1 koje ima elemente s intenzitetom u intervalu $[0, k]$ i u područje C_2 koje se sastoji od elemenata u intervalu $[k+1, L-1]$.

3. Za sve moguće vrijednosti praga k odrediti sve potrebne parametre kako bi se odredila varijanca između klasa $\sigma_B^2(k)$ pomoću formule:

$$\sigma_B^2(k) = \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))} \tag{3-7}$$

Parametri koji su potrebni su: kumulativna suma vjerojatnosti za prvu klasu $P_1(k)$, globalna srednja vrijednost m_G i $m(k)$,tj. srednja vrijednost intenziteta na intervalu $[0, k]$.

4. Pronalaženje intenziteta k^* pri kojem je varijanca između klase C_1 i C_2 maksimalna. Ako ima više takvih vrijednosti onda se traži njihova srednja vrijednost. [7, str.769]

Navedeni algoritam se primjenjuje pomoću OpenCV funkcije *threshold()* i dodatnog *THRESH_OTSU* parametra: *ret, thresh = cv2.threshold(image,0, max_value, cv2.THRESH_BINARY + cv2.THRESH_OTSU)*. Koristi se ista funkcija kao i u amplitudnoj

segmentaciji s manualnim postavljanjem praga, samo što se na parametar *THRESH_BINARY* dodaje *THRESH_OTSU* parametar te se umjesto vrijednosti praga (*threshold_value*) ostavlja broj nula zato što se prag ne postavlja manualno, već se pomoću Otsu metode traži njegova optimalna vrijednost k^*. Programski kod s kojim se primjenjuje Otsu metoda je prikazan na slici 3.6., a rezultat primjene, uspoređen s originalnom slikom, može se vidjeti na slici 3.7.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('red_mazda.jpg',0)

# Primjena Otsu metode na originalnoj slici
ret,th = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow('Otsu metoda', th)
cv2.waitKey(0)
cv2.destroyAllWindows()

plt.subplot(1,2,1)
plt.imshow(img,'gray')
plt.title('Originalna slika')
plt.xticks([]),plt.yticks([])

plt.subplot(1,2,2)
plt.imshow(th,'gray')
plt.title('Otsu metoda')
plt.xticks([]),plt.yticks([])

plt.show()

cv2.imwrite('../Slike/Otsu metoda/Otsu_threshold.jpg', th)
```

Sl. 3.6. Programski kod pomoću kojeg se primjenjuje Otsu metoda

Sl. 3.7. Originalna slika i slika segmentirana Otsu metodom.

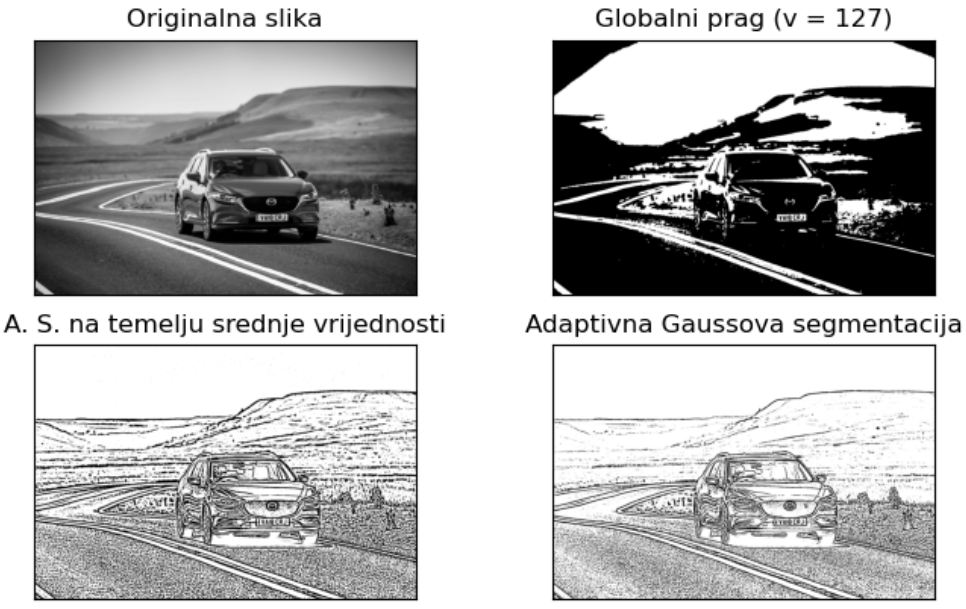
3.3. Adaptivna amplitudna segmentacija

Zbog utjecaja neujednačenog osvjetljenja i sjene koju objekti mogu stvoriti, na slici može biti slučaj da je nemoguće s jednim globalnim pragom ostvariti željene rezultate. Stoga je potrebna adaptivna amplitudna segmentacija koja određuje lokalne pragove na temelju vrijednosti okolnih elemenata u prozoru *N* x *N*. Od dvije vrste adaptivne segmentacije koje će se primijeniti u nastavku, prva se temelji na računanju srednje vrijednosti, a druga na računanju težinskog zbroja vrijednosti okolnih elemenata u prozoru *N* x *N* s dodijeljenim težinama prema Gaussovoj raspodjeli. [9]

Primjena obje vrste adaptivne segmentacije se može izvesti pomoću OpenCV funkcije: *slika = cv2.adaptiveThreshold(originalna_slika, maksimalna_vrijednost, adaptivna_metoda, cv2.THRESH_BINARY, veličina_prozora, konstanta).* Pomoću parametra *adaptivna_metoda* može se izabrati koja će se vrsta adaptivne segmentacije vršiti, dok parametri *veličina_prozora* i *konstanta* utječu na veličinu N x N prozora i težinsku sumu koja se dobije korištenjem pripadnog *N* x *N* kernela. Programski kod u kojemu se primjenjuju obje vrste adaptivne segmentacije te uspoređuju s originalnom i manualno segmentiranom slikom je prikazan na slici 3.8., a slika 3.9. prikazuje rezultate i usporedbu tih segmentacija.

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('red_mazda.jpg',0)
img = cv2.medianBlur(img,5)

ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

# Primjena adaptivne segmentacije na temelju lokalne srednje vrijednosti
thresh2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,11,2)

# Primjena adaptivne Gaussove segmentacije
thresh3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)

titles = ['Originalna slika', 'Globalni prag (v = 127)',
            'A. S. na temelju srednje vrijednosti', 'Adaptivna Gaussova segmentacija']
images = [img, thresh1, thresh2, thresh3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
    cv2.imwrite('../Slike/Adaptivna segmentacija/' + titles[i] + '.jpg', images[i])
plt.show()
```

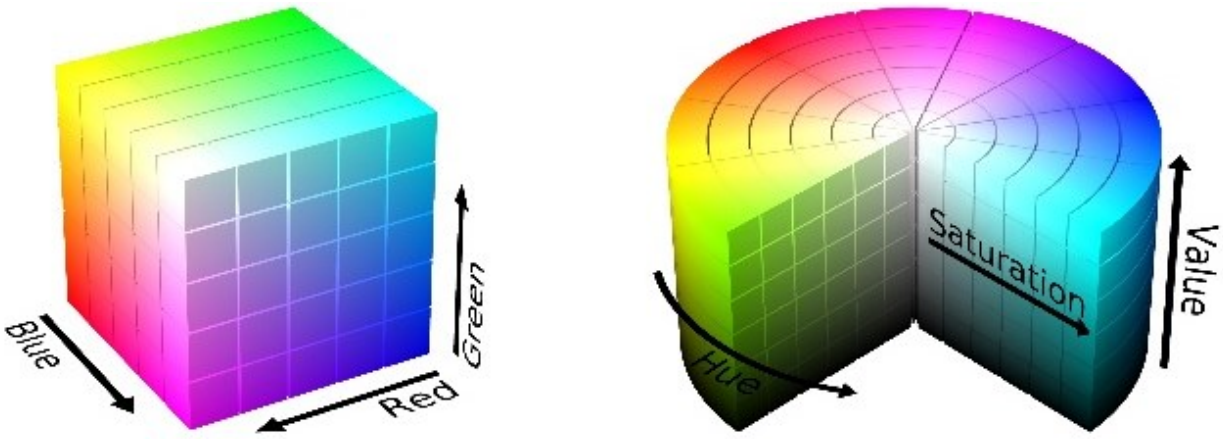
Sl. 3.8. Programski kod pomoću kojeg se primjenjuju dvije vrste adaptivnih amplitudnih segmentacija [9]



Sl. 3.9. Primjena dvije vrste adaptivnih amplitudnih segmentacija te njihova usporedba s amplitudnom segmentacijom s manualno postavljenim pragom

3.4. Amplitudna segmentacija digitalnih slika u boji

Amplitudna segmentacija digitalnih slika u boji je korisna jer većina objekata u stvarnom svijetu nisu samo monokromatskih tonova. S obzirom da su za prikaz slike u boji potrebna tri parametra, a ne samo jedan u slučaju monokromatske slike, amplitudna segmentacija takvih slika je malo kompliciranija. Format pomoću kojeg se brojevima često prikazuje digitalna slika je BGR ili RGB format te se pomoću njega prikazuje prisutnost plave (engl. *blue*), zelene (engl. *green*) i crvene (engl. *red*) boje trima brojevima s vrijednostima od 0 do 255. Takav format je koristan za prikaz na ekranima s LED tehnologijom jer se takav ekran sastoji od elemenata slike koji sadrže plave, zelene i crvene LED diode što se poklapa s navedenim formatom i čini njegovu primjenu intuitivnom. Takav format u digitalnoj obradi slike je manje koristan i intuitivan jer će se pri obradi lakše moći opažati svjetlina, jarkost i nijansa boje nego intenzitet pojedine crvene, zelene i plave komponente. [11, str.2263] Format koji bolje predočava takav prikaz boja je HSV format. HSV je kratica za parametre formata koji brojčano prikazuju nijansu (engl. *hue*), zasićenost ili jarkost (engl. *saturation*) i vrijednost ili intenzitet (engl. *value*) boje elemenata slike. Vizualizacija RGB i HSV formata je prikazana na slici 3.10.



Sl. 3.10. Vizualizacija RGB (lijevo) i HSV formata (desno) brojčanog prikaza boje u digitalnoj slici (Izvor: https://phenospex.helpdocs.com/plant-parameters/hue)

Kada se pomoću HSV formata želi izvršiti amplitudna segmentacija određenih boja na slici, potrebno je naći u kojim se kružnim isječcima sa slike 3.10. nalaze željeni tonovi boje te iz njih pronaći po dvije točke koje jednoznačno određuju pojedinačni kružni isječak. Nakon toga se može stvoriti maska regije koja sadrži željene tonove boja pomoću koje se mogu izolirati tražene

regije na originalnoj slici. Primjena navedenog algoritma je prikazana u programskom kodu na slici 3.11. i rezultati te primjene na slici 3.12.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread("red_mazda.jpg")

# Promjena formata slike (colorspace) iz BGR u HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Traženje maske pomoću koje se mogu izdvojiti crvene regije slike
# u rasponu (0, 120, 70) ~ (10, 255, 255) i (170, 120, 70) ~ (180, 255, 255)
# HSV vrijednosti:
mask1 = cv2.inRange(hsv, (0, 120, 70), (10, 255, 255))
mask2 = cv2.inRange(hsv, (170, 120, 70), (180, 255, 255))
mask = mask1 + mask2

# Primjenjivanje maske na originalnoj slici radi izdvajanja crvenih regija
imask = mask>0
red = np.zeros_like(img, np.uint8)
red[imask] = img[imask]

# S obzirom da matplotlib programska biblioteka koristi RGB format slike, a ne BGR,
# potrebno je slikama promijeniti format
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.subplot(1,2,1)
plt.imshow(img)
plt.title('Originalna slika')
plt.xticks([]),plt.yticks([])

red = cv2.cvtColor(red, cv2.COLOR_BGR2RGB)
plt.subplot(1,2,2)
plt.imshow(red)
plt.title('Izdvojeni crveni dijelovi slike')
plt.xticks([]),plt.yticks([])

plt.show()
```

Sl. 3.11. Programski kod pomoću kojeg se mogu izdvojiti crveni tonovi iz originalne slike [10]

Sl. 3.12. Prikaz amplitudnog segmentiranja crvenih tonova iz originalne slike

4. SEGMENTACIJA NA TEMELJU RUBOVA

4.1. Detekcija rubova

4.1.1. Sobelov operator

Metode detekcije rubova u digitalnoj obradi slike su algoritmi koji detektiraju rubove raznih objekata i regija slike zbog čega se koristi kao temeljni algoritam u određenim segmentacijskim metodama i algoritmima za detekciju objekata. Temeljni princip metoda detekcije rubova je detektiranje naglih promjena vrijednosti elemenata slike. To se matematički može realizirati pomoću gradijenta i prve derivacije. [7, str.728] Gradijent koji se koristi za detektiranje rubova na digitalnoj slici je prikazan formulom:

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{4-1}$$

gdje su g_x i g_y parcijalne derivacije vrijednosti elemenata slike po x i y osi.

Za detekciju rubova potrebno je vektor gradijenta iz formule (4-1) pretvoriti u skalarnu vrijednost, tj. modul tog vektora. Modul gradijenta se može izračunati formulom:

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{4-2}$$

Korištenjem formule (4-2) mogli bi se pronaći elementi slike koji sadrže rubove, no zbog izraza s korijenom i kvadratima, izvršavanje programskog koda koji opisuje takav matematički izraz bi bilo neprivlačno i kompliciranо. Stoga se koristi aproksimacija prikazana formulom:

$$M(x,y) \approx |g_x| + |g_y| \tag{4-3}$$

Navedena formula je puno prigodnija za izvođenje na računalu, ali isto tako zadržava relativne promjene vrijednosti derivacija. [7, str.732]

Za korištenje prethodno navedenih formula na digitalnoj slici potrebna je konvolucija. Konvolucija je matematička funkcija nastala integriranjem umnoška dviju funkcija po intervalu njihove definicije gdje su te funkcije ravnopravne tako da svaka beskonačno mala promjena jedne funkcije utječe na drugu funkciju u cijelome intervalu definicije. [12] Kako bi se koristila konvolucija na digitalnoj slici potrebno je aproksimirati parcijalne derivacije g_x i g_y filtrima (engl. *kernel*). Filtar je maska pomoću koje se vrši konvolucija na digitalnoj slici. Primjer jednog takvog 3x3 filtra se može vidjeti na slici 4.1.

w_0	w_1	w_3
w_4	w_5	w_6
w_7	w_8	w_9

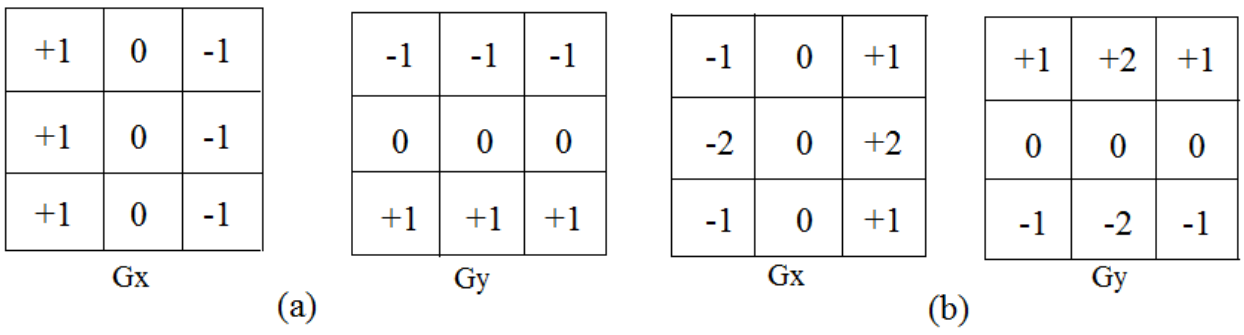
Sl. 4.1. Prikaz 3x3 filtera. [13]

Filtar sa slike 4.1. je primjer 3x3 filtra koji se može koristiti u konvoluciji na način da se iterativno njime pređe preko cijelog područja slike te da se u svakom koraku izvrši suma produkata koeficijenata filtera i vrijednosti elemenata slike koji su u tom koraku pod utjecajem filtera. Taj produkt predstavlja novu vrijednosti središnjeg elementa slike. [7, str.172] Općenito, matematički se to može izraziti formulom:

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_{mn} z_{mn} = \sum_{k=1}^{mn} w_k z_k \tag{4-4}$$

gdje je: w_k – koeficijent ili težina k-tog elementa filtera, z_k – vrijednost k-tog elementa slike pod filterom, R – suma produkata svih koeficijenata filtera i vrijednosti elemenata slike koji su u tom koraku pod utjecajem filtera.

Postoji veliki broj filtara koji se mogu koristiti u obradi digitalne slike. Filtri koji su od posebnog značaja za ovaj završni rad su Prewittov i Sobelov operator. Oba operatora predstavljaju aproksimaciju parcijalnih derivacija vrijednosti elemenata slike po x i y osi. Na slici 4.2. prikazani su Prewittovi i Sobelovi filtri za x i y os.



Sl. 4.2. Prikaz Prewittovih filtera pod (a) i Sobelovih filtera pod (b) [14]

Iako se oba operatora koriste za aproksimiranje parcijalne derivacije vrijednosti elemenata slike po x i y osi, Sobelov operator je prigodniji za uporabu jer pri računanju parcijalnih derivacija smanjuje utjecaj šuma. [7, str.731] Primjena navedenih filtera se može napraviti pomoću OpenCV funkcije *filter2D( ).* Primjena te funkcije i definiranje filtara u Pythonu pomoću Numpy programske biblioteke je prikazano na slikama 4.3. i 4.4., dok je na slikama 4.5. i 4.6. prikazana originalna monokromatska slika i slike koje su nastale primjenom prethodno definiranih filtera.

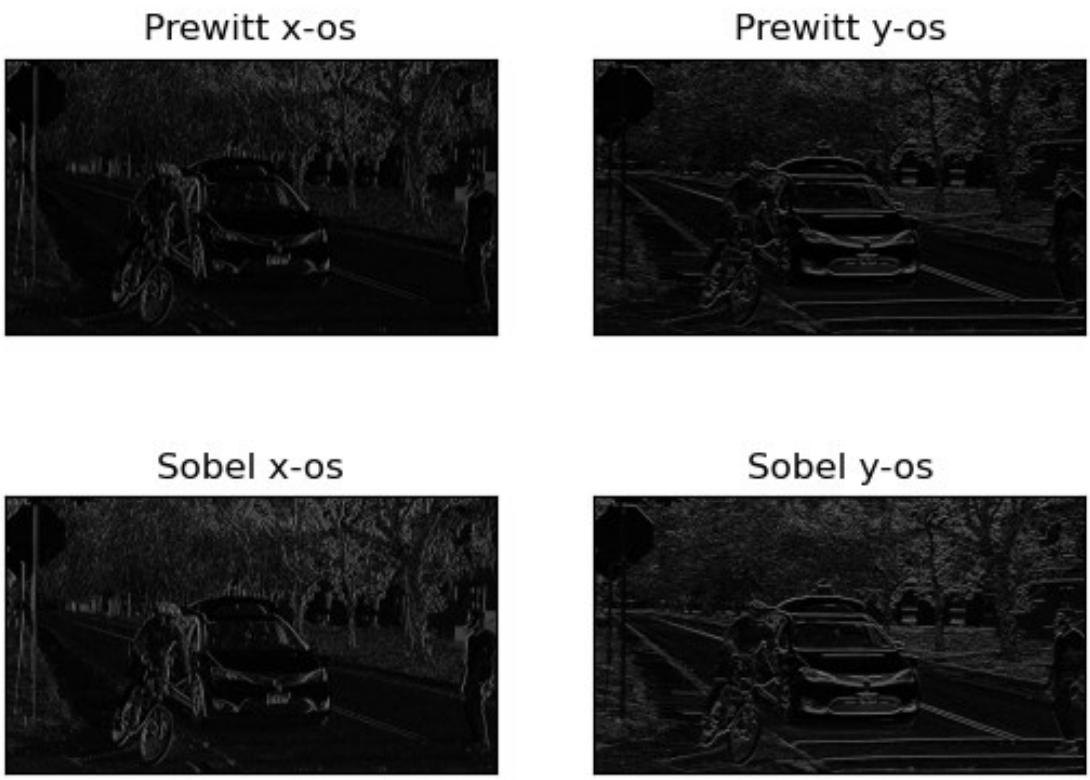
```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Sobelov filter za x-os
sobelX = np.array((
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]), dtype="int")

# Sobelov filter za y-os
sobelY = np.array((
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1]), dtype="int")

# Prewittov filter za x-os
prewittX = np.array((
    [-1, 0, 1],
    [-1, 0, 1],
    [-1, 0, 1]), dtype="int")

# Prewittov filter za y-os
prewittY = np.array((
    [-1, -1, -1],
    [0, 0, 0],
    [1, 1, 1]), dtype="int")

# Definiranje liste (preciznije engl. set of tuples) filtera (engl. kernel) radi lakšeg pristupanja pojedinom kernelu
kernelTuplesSet = (
    ("Prewitt x-os", prewittX),
    ("Prewitt y-os", prewittY),
    ("Sobel x-os", sobelX),
    ("Sobel y-os", sobelY)
)

# Učitavanje i promjena slike u monokromatsku sliku pomoću parametra 0
image = cv2.imread('white_car_and_bike.jpg', 0)

cv2.imwrite('../Slike/Primjena kernela/originalna_slika.jpg', image)
```

Sl. 4.3. Prvi dio programskog koda pomoću kojeg se primjenjuju Sobelovi i Prewittovi filtri

```
i = 1
for (kernelName, kernel) in kernelTuplesSet:
    # Primjena svih filtera iz prethodno definirane liste
    opencvOutput = cv2.filter2D(image, -1, kernel)

    # Prikaz svih slika na kojima su primijenjeni filtri iz krenelList
    plt.subplot(2, 2, i), plt.imshow(opencvOutput, 'gray')
    plt.title(kernelName)
    plt.xticks([]), plt.yticks([])

    # Spremanje pojedinih slika
    cv2.imwrite('../Slike/Primjena kernela/' + kernelName + '.jpg', opencvOutput)
    i = i + 1
plt.show()
```

Sl. 4.4. Drugi dio programskog koda pomoću kojeg se primjenjuju Sobelovi i Prewittovi filtri

Sl. 4.5. Prikaz originalne monokromatske slike (Izvor: https://www.ualrpublicradio.org/post/study-backs-getting-driverless-cars-road-waymo-ditches-backup-drivers)



Sl. 4.6. Prikaz slika na kojima su primijenjeni Prewittovi i Sobelovi filtri

Ako se ne žele manualno definirati filtri, mogu se koristiti i gotove OpenCV funkcije. Vrlo je korisna OpenCV funkcija *Sobel( )* pomoću koje se koristi Sobelov operator. Trećim i četvrtim parametrom navedene funkcije se određuje red derivacije vrijednosti elemenata slike po *x* i *y* osi. S obzirom da se funkcijom *Sobel( )* može pronaći *x* i *y* komponenta gradijenta, jedino što je preostalo je da se funkcijama *convertScaleAbs( ) i addWeighted( )* izračuna aproksimirani proračun iz formule (4-3). Na slici 4.8. je prikazana primjena prethodno navedenog algoritma čiji se programski kod može vidjeti na slici 4.7.

```python
import cv2

ddepth = cv2.CV_16S

# Učitavanje slike
gray = cv2.imread('white_car_and_bike.jpg', 0)

# Primjena Sobelovog kernela radi nalaženja rubova, tj. prvih derivacija po x i y osi slike.
# Pri tome se ddepth (dubina) vrijednost slike mijenja iz CV_8U u CV_16S radi očuvanja vrijednosti
# deriviranih vrijednosti
grad_x = cv2.Sobel(gray, ddepth, 1, 0)
grad_y = cv2.Sobel(gray, ddepth, 0, 1)

# vraćanje vrijednosti dubine slike iz CV_16S u CV_8U
abs_grad_x = cv2.convertScaleAbs(grad_x)
abs_grad_y = cv2.convertScaleAbs(grad_y)

# Aproksimiranje gradijenta zbrajanjem njegove x i y komponente
grad = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)

cv2.imwrite('../Slike/Primjena kernela/Sobel gradijent.jpg', grad)
cv2.imshow('Sobel', grad)
cv2.waitKey(0)
```

Sl. 4.7. Programski kod za proračun aproksimiranog modula vektora gradijenta svih elemenata slike pomoću Sobelovog operatora

Sl. 4.8. Slika na kojoj je primijenjen programski kod sa slike 3.7.

4.1.2. Canny algoritam za detekciju rubova

Canny algoritam je kompleksan algoritam za detekciju rubova na digitalnoj slici kojeg je osmislio John F. Canny. Cilj njegovog algoritma je trojak: [7, str.741]

1. Ostvarivanje niske stope pogreške pri detekciji rubova.
2. Detektirane točke rubova bi trebale biti dobro lokalizirane što znači da detektirane točke moraju biti što bliže središnjem dijelu ruba.
3. Rub koji se detektira bi trebao imati debljinu jednaku jednom elementu slike.

Nakon što se navedena tri kriterija matematički izraze te optimiziraju za korištenje na računalu, algoritam koji se dobije se može objasniti tako što se rastavi na četiri osnovna koraka:

1. Primjena Gaussovog filtra na originalnu sliku tako što se konvoluira funkcija vrijednosti elemenata slike $f(x,y)$ i Gaussova funkcija $G(x,y)$ prikazana formulom:

$$G(x,y) = Ae^{-\left(\frac{(x-x_0)^2}{2\sigma_X^2}+\frac{(y-y_0)^2}{2\sigma_Y^2}\right)} \tag{4-5}$$

   gdje je: A – amplituda, x_0 i y_0 – središnje točke distribucije, σ_X i σ_Y parametri kojim se opisuje raširenost funkcije.
2. Izračun modula vektora gradijenta prema izrazu (4-2) i pripadne kutove gradijenta prema formuli:

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{4-6}$$

3. Suzbijanje i brisanje dijelova ruba na slici modula gradijenta $M(x,y)$ koji nisu njegova maksimalna vrijednost. Drugim riječima, vrši se stanjivanje rubova dok se ne dobije dio ruba koji sadrži elemente slike u kojima je prisutna najveća promjena vrijednosti elemenata slike s obzirom na jednu od četiri glavne orijentacije (horizontalna, vertikalna ili dvije dijagonalne). Ta orijentacija ovisi o kutu gradijenta α pripadnog ruba te se zaokružuje na najbližu glavnu orijentaciju. Izvođenjem ovog koraka se dobije $g_N(x,y)$[7, str.743]
4. Postavljanje višeg i nižeg praga na $g_N(x,y)$. Oni rubni elementi slike koji su viši od višeg praga se označavaju sa $g_{NH}(x,y)$,tj. kao „jaki“ rubovi, a oni koji su između gornjeg i donjeg praga se označavaju sa $g_{NL}(x,y)$,tj. kao „slabi“ rubovi. Oni rubni elementi slike koji su manji od nižeg praga se postavljaju na vrijednost nula. John Canny je preporučio da omjer višeg i nižeg praga treba biti 2:1 ili 3:1. [7, str.744] Nakon toga slijedi spajanje dužih rubova u slučaju da su isprekidani. To se vrši pomoću sljedeće procedure:

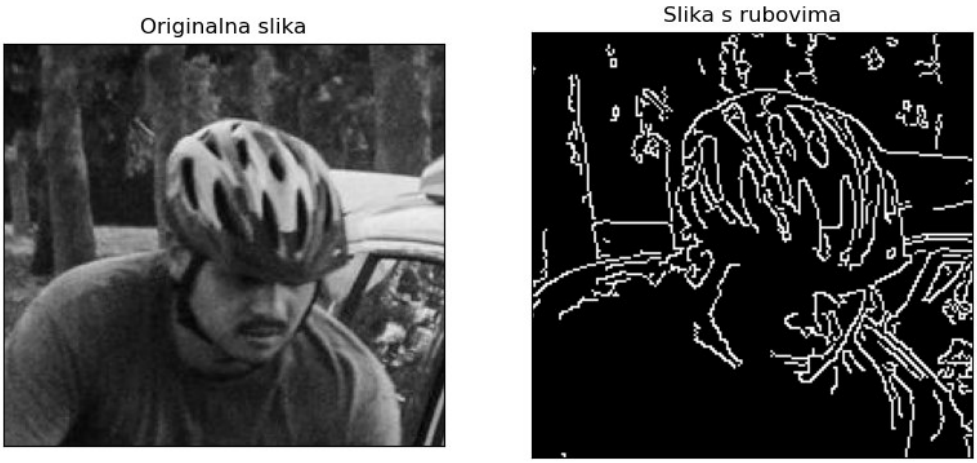
a. Lociraj slijedeći neposjećeni rubni element slike koji se nalazi unutar $g_{NH}(x, y)$.
b. Označi sve slabe rubne elemente slike, koji su povezani s jakim rubnim elementom slike osmim stupnjem povezanosti (engl. *8-connectivity*), kao ispravnima. To znači da se spomenuti element slike mora dodirivati u bilo kojem smjeru (horizontalnom, vertikalnom ili dijagonalnom) s jakim rubnim elementom slike [7, str.90, str.744].
c. Ako su svi elementi slike iz $g_{NH}(x, y)$ posjećeni, onda se može nastaviti na korak 4.d, a ako nisu onda se treba vratiti na korak 4.a.
d. Svi rubni elementi slike iz $g_{NL}(x, y)$ koji nisu označeni u koraku 4.b kao ispravni, se postavljaju na vrijednost nula.

Navedeni algoritam se u Python programskom jeziku (Slika 4.9.) može vrlo lagano primijeniti pomoću gotove OpenCV funkcije *Canny( )*. Drugi i treći parametar te funkcije služi za manualno postavljanje nižeg i višeg praga spomenutog u 4. koraku algoritma. Na slikama 4.10. i 4.11. su prikazane usporedbe originalne slike i slike na kojoj je primijenjen Canny algoritam.

```
import cv2
from matplotlib import pyplot as plt

# Učitavanje slike
img = cv2.imread('white_car_and_bike.jpg',0)

# Primjena Canny algoritma
edges = cv2.Canny(img,100,200)

plt.subplot(1, 2, 1),plt.imshow(img,cmap = 'gray')
plt.title('Originalna slika'), plt.xticks([]), plt.yticks([])
plt.subplot(1, 2, 2),plt.imshow(edges,cmap = 'gray')
plt.title('Slika s rubovima'), plt.xticks([]), plt.yticks([])

plt.show()
```

Sl. 4.9. Programski kod u kojemu se upotrebljava Canny algoritam

Sl. 4.10. Usporedba originalne slike i slike na kojoj je primijenjen Canny algoritam



Sl. 4.11. Usporedba uvećanih dijelova sa slike 3.10. kako bi se uočila dobra lokaliziranost rubnih elemenata slike.

4.2. Segmentacija slike pomoću Houghove transformacije

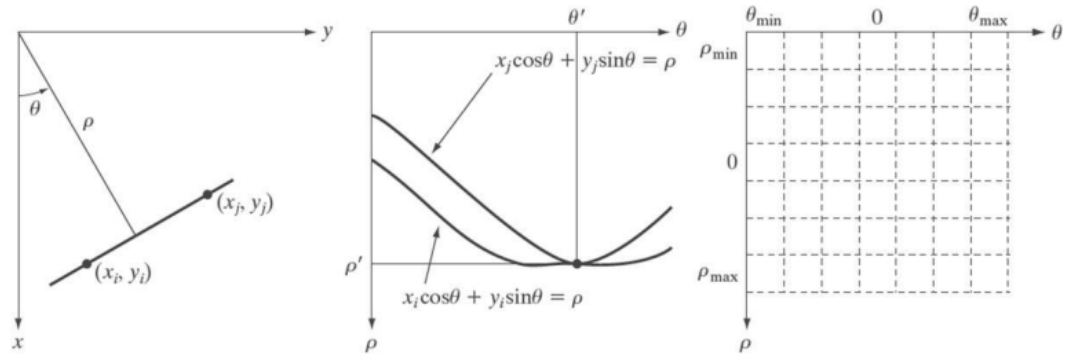
4.2.1. Detekcija pravaca pomoću Houghove transformacije

Houghova transformacija je metoda segmentacije slike na temelju rubova te se zbog toga u prvom koraku pripadnog algoritma koristi neka od spomenutih metoda detekcije rubova. Ona služi za detektiranje i pronalaženje pravaca, kružnica, ali i nekih drugih krivulja. Algoritam Houghove transformacije za detekciju pravca se može opisati kroz nekoliko koraka:

1. Detektiranje rubova i stvaranje binarne slike pomoću neke od prethodno navedenih detekcijskih metoda.
2. Svaki element slike određen x i y vrijednostima se može prikazati u parametarskom $\rho\theta$ koordinatnom sustavu (slika 4.12., srednji graf) kao sinusoida koja opisuje sve linije koje prolaze kroz tu točku u xy koordinatnom sustavu. [7, str.756] Veza između parametara ρ i θ te vrijednosti x i y je prikazana formulom:

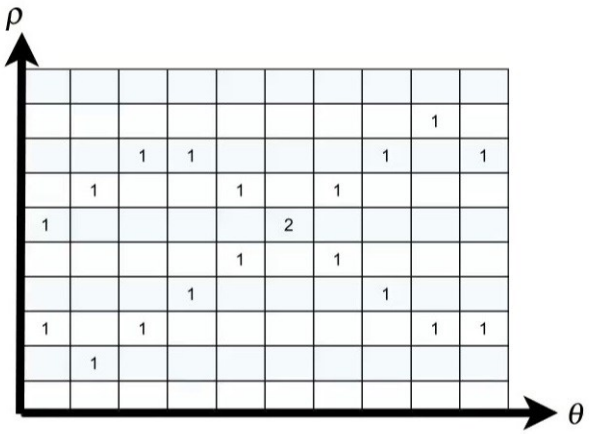
$$\rho = x\,cos\theta + y\,sin\theta \tag{4-7}$$

Taj $\rho\theta$ koordinatni sustav se treba podijeliti na akumulacijska polja označena s $A(i,j)$ gdje su i i j broj retka i stupca pripadnog polja (slika 4.12., desni graf). Ta polja će se nalaziti unutar minimalnih i maksimalnih vrijednosti ρ i θ parametara. Stoga, za ρ mora vrijediti nejednakost $-D \leq \rho \leq D$, gdje je D najveća moguća udaljenost elemenata slike, a za θ mora vrijediti $-90° \leq \theta \leq 90°$. [7, str.756]



Sl. 4.12. Prikaz xy koordinatnog sustava (lijevi graf), parametarskog $\rho\theta$ koordinatnog sustava (srednji graf) i parametarskog $\rho\theta$ sustava podijeljenog na akumulacijska polja (desni graf) [15]

3. Postavlja se $A(i,j) = 0$ za sve i i j vrijednosti.
4. Za svaku rubnu točku binarne slike iz prvog koraka odrediti ρ mijenjajući vrijednost parametra θ od minimalne do maksimalne vrijednosti po izrazu (4-7).
5. Svaki dobiveni ρ pridružiti određenim poljima u kojima se ta vrijednost nalazi. (Slika 4.13.)
6. Zbrajati vrijednosti polja $A(i,j)$ kada se više ρ vrijednosti može pripisati istom polju. (Slika 4.13.)



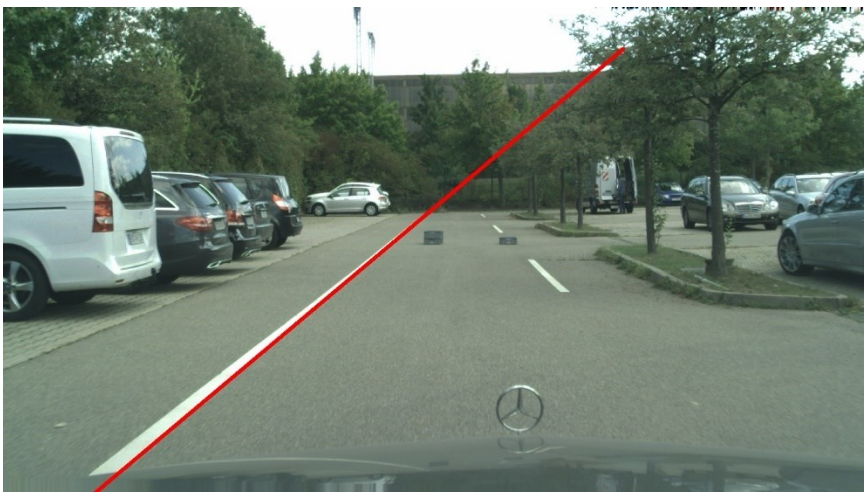
Sl. 4.13. Prikaz 5. i 6. koraka Houghove transformacije [16]

7. Maksimumi vrijednosti $A(i,j)$ se nalaze na presjecima kvantiziranih sinusoida različitih rubnih točaka. Pripadna polja tih maksimuma su jednoznačno definirana s vrijednostima θ_i i ρ_j, što su zapravo parametri detektiranog pravca čiji se izraz dobije kada se uvrste dobiveni parametri u izraz (4-7). [7, str.756]
8. Potrebno je ispitati kontinuiranost detektiranog pravca tako što se računaju udaljenosti između elemenata slike koji leže na pravcu, ali koji nisu spojeni. Ako je udaljenost manja od definiranog praga, ti elementi se spajaju pravcem.[7, str.758]

Primjena navedenog algoritma se može realizirati u Python programskom jeziku (slika 4.14.) pomoću OpenCV funkcije *cv2.HoughLines( )*. Prvi parametar te funkcije je binarna slika na kojoj je potrebno detektirati pravce. Drugi i treći parametar su preciznosti ρ i θ vrijednosti, a četvrti parametar je prag iznad kojeg mora biti vrijednost $A(i,j)$ da bi se pripadni pravac mogao uzeti u obzir pri detektiranju. Nakon što se izvrši funkcija, potrebno je iscrtati detektirani pravac na originalnoj slici. [17] Da bi se to moglo napraviti prvo se treba iz dobivenih ρ i θ vrijednosti izračunati jednadžba pravca prema izrazu (4-7) te pomoću dvije točke tog pravca i funkcije *line( )* ucrtati detektirani pravac. Rezultat toga se može vidjeti na slici 4.15. gdje se može uočiti da je detektiran pravac kojeg čini puna crta na cesti.

```python
import cv2
import numpy as np

# Učitavanje i pretvaranje slike u monokromatsku sliku
img = cv2.imread('road_pic.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

# Primjena Canny algoritma za detektiranje rubova
edges = cv2.Canny(gray,50,150,apertureSize = 3)

# Primjena Houghovog algoritma za detektiranje pravaca
lines = cv2.HoughLines(edges,1,np.pi/180,200)
for rho,theta in lines[0]:
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))
    #iscrtavanje detektiranih pravaca na originalnoj slici
    cv2.line(img,(x1,y1),(x2,y2),(0,0,255),8)

cv2.imwrite('../Slike/Houghove linije i krugovi/houghlinije.jpg',img)
```

Sl. 4.14. Prikaz programskog koda pomoću kojeg se primjenjuje Houghova transformacija za detektiranje pravca [17]



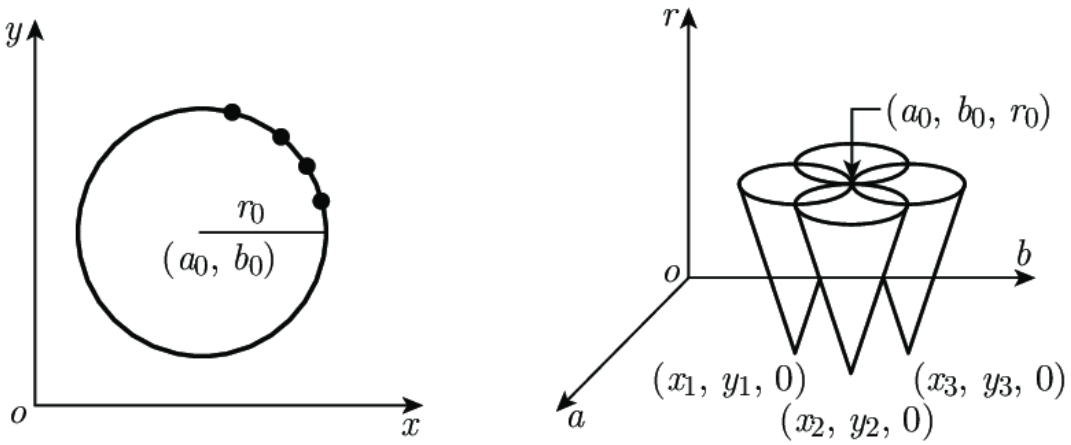
Sl. 4.15. Slika na kojoj je detektiran pravac kojeg stvara puna linija na cesti [18]

4.2.2. Detekcija kružnica pomoću Houghove transformacije

Houghova transformacija se može koristiti za detektiranje bilo koje krivulje opisane funkcijom $g(\vec{v}, \vec{c}) = 0$, gdje je $\vec{v}$ vektor koordinata kojima se opisuje krivulja, a $\vec{c}$ vektor koeficijenata. Na primjer, u prošlom odjeljku, pri korištenju Houghove transformacije za detektiranje pravaca, vektor $\vec{v}$ je sadržavao x i y koordinate, a vektor $\vec{c}$ se odnosio na ρ i θ parametre. Na taj način se može gledati i pri detektiranju nekih kompliciranijih krivulja poput kružnice. Razlika u takvom korištenju transformacije je ta što se za detekciju kružnice moraju koristiti dvije koordinate i tri koeficijenta, što je prikazano izrazom: [7, str.758]

$$(x - a)^2 + (y - b)^2 = R^2 \tag{4-8}$$

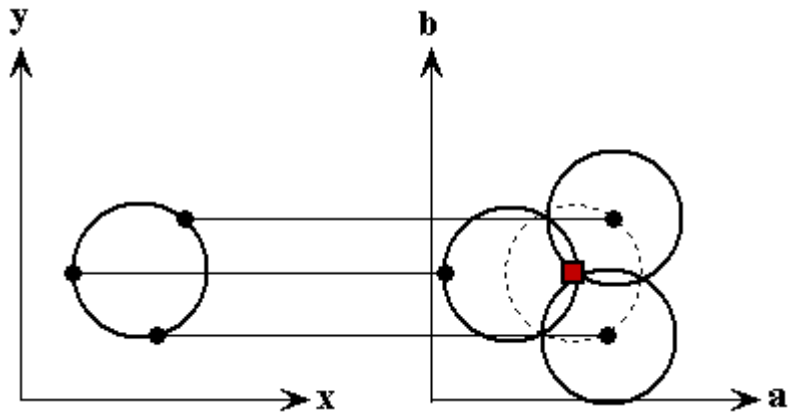
To znači da bi u tom slučaju, kada se ne zna niti jedan od tri koeficijenta, bilo potrebno tri dimenzije za opis parametarskog prostora, što čini ovu transformaciju kompliciranijom od Houghove transformacije za detekciju pravaca. Unatoč tome, analogija se može uspostaviti između te dvije vrste transformacija. Kao i pri detekciji pravaca, parametarski prostor se može rastaviti na akumulacijske kvadre. Također, svaka točka koja pripada kružnici u xy koordinatnom sustavu ima pripadni stožac u parametarskom prostoru. Sjecišta tih stožaca određuju zajedničke kružnice točaka (slika 4.16.). Zbog toga se analogno može definirati da akumulacijski kvadar s lokalnim maksimumom broja sjecišta stožaca označava onaj akumulacijski kvadar s parametrima pripadne detektirane kružnice. [19]



Sl. 4.16. Prikaz koordinatnog sustava (lijevi graf) i parametarskog prostora (desni graf) u kojemu se sijeku stošci pripadnih točaka iz detektirane kružnice u lijevom grafu [20]

Ako je ipak polumjer iz izraza (4-8) poznat, onda se ne mora koristiti 3D parametarski prostor nego se može koristiti samo 2D parametarska ravnina. U toj ravnini svaka točka iz

koordinatnog sustava će imati svoju pripadnu kružnicu. Točka u kojoj će se sjeći te kružnice u parametarskoj ravnini će određivati zajedničku kružnicu pripadnih točaka u xy koordinatnom sustavu (slika 4.17.). Nastavljajući analogiju iz prošlog odjeljka, u parametarskoj ravnini se mogu definirati akumulacijska polja s početnim vrijednostima jednakim nuli, čije će se vrijednosti povećavati za jedan svaki put kad se točka kružnice u parametarskoj ravnini nađe u pripadnom polju. Ako polje posjeduje lokalni maksimum broja sjecišta kružnica i ako je taj maksimum veći od praga akumulatora, onda se pripadni parametri tog polja mogu označiti kao parametri detektirane kružnice.



Sl. 4.17. Prikaz koordinatnog sustava (lijevi graf) i parametarske ravnine (desni graf) u kojemu se sijeku kružnice pripadnih točaka iz detektirane kružnice u lijevom grafu [19]

Primjena navedenog algoritma se može izvršiti u Python programskom jeziku (slika 4.18.) pomoću OpenCV funkcije *cv2.HoughCircles( )*. Prvi parametar navedene funkcije je monokromatska slika koja je prethodno zamućena *medianBlur* filterom pomoću istoimene OpenCV funkcije. Drugi parametar je parametar *method*, tj. metoda detekcije (samo je cv2.*HOUGH_GRADIENT* podržan). Treći parametar (*dp*) je inverzni omjer rezolucije akumulatora. Četvrti parametar (*minDist*) je minimalna udaljenost između središta detektiranih kružnica. Peti parametar (*param1*) se odnosi na viši prag koji se prosljeđuje Canny detektoru ruba sadržanog unutar same funkcije. Šesti parametar (*param2*) je prag za vrijednosti akumulatora. Što je manji, to se više lažnih krugova može detektirati. Sedmi i osmi parametar je minimalni i maksimalni polumjer detektiranih kružnica. [21] Nakon što se detektira kružnica, potrebno ju je ucrtati u originalnoj slici pomoću naredbe *circle()* (slika 4.19.).

```python
import cv2
import numpy as np

# Učitavanje slike
img = cv2.imread('prometni_znak.jpeg',0)

# Odstranjivanje šuma na slici pomoću medianBlur funkcije
img = cv2.medianBlur(img,5)
cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)

# Primjena Houghovog algoritma za detektiranje kružnica
circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,20,
                           param1=50,param2=80,minRadius=50,maxRadius=170)
circles = np.uint16(np.around(circles))

# Iscrtavanje detektiranih kružnica na originalnoj slici
for i in circles[0,:]:
    # Crtanje kružnice
    cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
    # Crtanje centra pripande kružnice
    cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)

cv2.imwrite('../Slike/Houghove linije i krugovi/houghkrugovi.jpg',cimg)
cv2.imshow('detected circles',cimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Sl. 4.18. Programski kod u kojem se primjenjuje Houghova transformacija za detekciju kružnica [22]

Sl. 4.19. Slika s ucrtanom detektiranom kružnicom na prometnom znaku [23]

5. SEGMENTACIJA NA TEMELJU RASTA PODRUČJA

Segmentacija na temelju rasta područja je metoda segmentacije pomoću koje se grupiraju pojedini dijelovi ili elementi slike u veće regije na temelju nekih prethodno definiranih kriterija za rast. Jedan od jednostavnijih načina za izvođenje rasta područja je kroz niz početnih točaka (engl. *seeds*) iz kojih rastu pojedine regije. One rastu na način da se iz svake početne točke promatraju pripadni susjedni elementi slike koji imaju prethodno definirano svojstvo slično kao i početna točka. Ako imaju slično svojstvo, onda se dodaju novonastaloj regiji pripadne početne točke te se iterativno testiraju nove susjedne točke. [7, str.785] Pripadni algoritam se može detaljnije opisati u četiri koraka:

1. Trebaju se pronaći sve povezane početne točke iz dvodimenzionalnog polja $S(x, y)$ te svaku pripadnu grupu povezanih početnih točaka treba smanjiti na samo jednu početnu točku u toj grupi. Na koordinatama preostalih početnih točaka u S polju treba postaviti maksimalnu vrijednost elementa slike, a na svim ostalim elementima slike postaviti nulu.
2. Treba se formirati slika f_Q, takva da se za svaki par koordinata (x, y) pridruži maksimalna vrijednost pripadnom elementu ako ispunjava određeni kriterij Q, a inače treba pridružiti nulu.
3. Slika g se formira tako što se binarnoj slici definiranoj dvodimenzionalnim poljem S dodaju elementi slike f_Q koje imaju maksimalne vrijednosti i koji su 8 – stupanjskom povezanosti (engl. *8-connectivity*) povezani s početnim točkama.
4. Označiti svaku spojenu regiju na slici g drugačijom oznakom. Na taj način svaka označena regija predstavlja pojedini segment segmentirane slike. [7, str.786]

Kriterij rasta područja koji će se koristiti u algoritmu ne ovisi samo o problemu koji se pokušava riješiti, već i o tipu slike koja se analizira. Na primjer, za analizu satelitskih slika Zemlje potrebno je uzimati u obzir sva tri parametra elemenata slika u boji. S druge strane, kod monokromatskih slika mogu se promatrati kriteriji koji se odnose na intenzitet elemenata slike, kao što su:

- sličnost između dviju susjednih točaka
- sličnost okolina dviju susjednih točaka
- sličnost točke i centroida regije (aritmetičke sredine vrijednosti točaka regije) [24]
- sličnosti točke i vrijednosti početne točke regije. [7, str.788]

Primjer opisanog algoritma će se izvršiti pomoću kriterija kojim će se ispitivati sličnost vrijednosti točaka i vrijednosti početne točke pripadne regije kojoj se ispitane točke trebaju pridružiti. To se može postići na način da se proizvoljno odredi prag koji će određivati maksimalnu apsolutnu razliku dvije prethodno navedene vrijednosti. Ako je apsolutna razlika manja ili jednaka od praga onda će se pripadni element slike uzimati u obzir pri rastu područja, a u suprotnome neće. S obzirom da se skup početnih točaka može izabrati proizvoljno jer ovisi o problematici koja se proučava [7, str.785], u ovoj primjeni će se određivati lijevim klikom miša na regije slike koje je potrebno istaknuti.

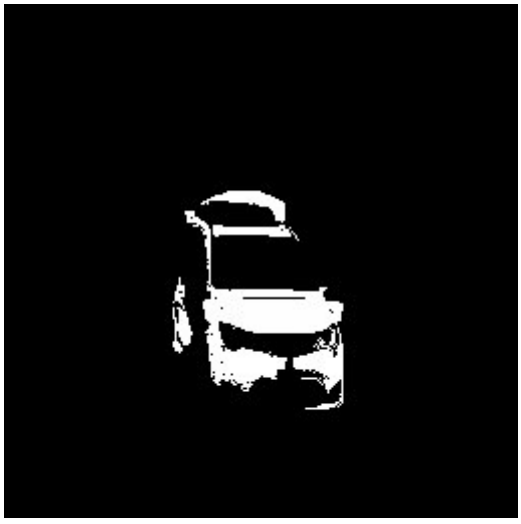
Navedena primjena algoritma će se izvršavati u Python programskom jeziku pomoću OpenCV i Numpy programskih biblioteka. Iako zbog svoje specifičnosti navedena primjena nema pripadnu OpenCV funkciju, ona se može realizirati koristeći određene funkcije iz Numpy biblioteke. To se može učiniti pomoću *region_growing()* funkcije koja koristi pomoćnu funkciju *get8n()* [25] koja služi za traženje 8 susjednih točaka elementa slike koji joj se proslijede pomoću x i y koordinata. Obje funkcije se mogu naći u prilogu P.4.1. Funkcija *region_growing()* vrši segmentaciju, omogućava praćenje rasta područja te prima tri parametra. Prvi parametar je varijabla koja sadrži sliku, a drugi parametar je koordinata početnog elementa slike koja je generirana lijevim klikovima miša na originalnoj slici pomoću funkcije *on_mouse()*. Nakon što se klikne na željene početne točke potrebno je stisnuti tipku na tipkovnici kako bi se pokrenula segmentacija. Treći parametar je prethodno navedeni prag koji se koristi u kriteriju algoritma. Slike dobivene koristeći navedene funkcije (slika 5.1.) na originalnoj slici (slika 5.2.) su prikazane na slikama 5.3. i 5.4.

```python
import cv2
import numpy as np

def get8n(x, y, shape):...

def region_growing(img, seed, threshold):...

def on_mouse(event, x, y, flags, params):
    if event == cv2.EVENT_LBUTTONDOWN:
        print('Seed: ' + str(x) + ', ' + str(y), img[y,x])
        clicks.append((y,x))

clicks = []
img = cv2.imread('white_car_and_bike.jpg', 0)

img = cv2.resize(img,(256,256))

cv2.namedWindow('Input')
cv2.setMouseCallback('Input', on_mouse, 0, )
cv2.imshow('Input', img)
cv2.waitKey()
seed = clicks[-1]
out = np.zeros_like(img)
for seed in clicks:
    out += region_growing(img, seed, 10)

cv2.imshow('Region Growing', out)
cv2.waitKey()
cv2.destroyAllWindows()
```

Sl. 5.1. Programski kod [25]

Sl. 5.2. Originalna slika na kojoj se vrši odabir početnih točaka i segmentacija

Sl. 5.3. Segmentirana slika s tri odabrane početne točke

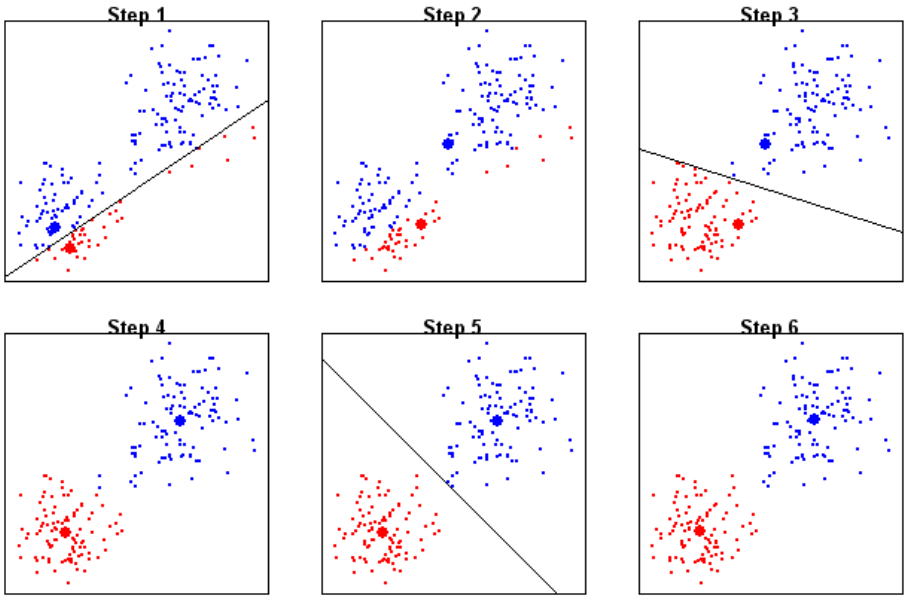


Sl. 5.4. Segmentirana slika s nekoliko odabranih početnih točaka na području automobila

6. SEGMENTACIJA GRUPIRANJEM

Segmentacija grupiranjem je metoda segmentiranja digitalne slike koja se svodi na sljedeći problem: kako neke N - dimenzionalne vektore x_i, koji opisuju značajke slike u određenim točkama, grupirati tako da značajke unutar jedne grupe budu uniformne. [24] Jedna od poznatih segmentacija na temelju grupiranja je grupiranje s K – srednjih vrijednosti. To je iterativni algoritam kojemu je cilj podijeliti skup podataka u K grupa koje se međusobno ne preklapaju. To se može postići u 5 koraka:

1. Definira se broj grupa K, pri čemu je $u_K(n)$ centar K – te grupe u n – toj iteraciji.
2. Za svaku se grupu u nultom koraku se određuje proizvoljno centar grupe $u_K(0)$.
3. Za svaki x_i provjerava se u n – toj iteraciji kojemu je centru najbliži te se pripadnoj grupi taj vektor pridružuje.
4. Ponovno se računaju centri grupa kao vektori koji minimiziraju udaljenost za vektore pojedine grupe.
5. Prethodni postupci, koji su prikazani na slici 6.1., se ponavljaju se sve dok se položaji centara više ne mijenjaju. [24]



Sl. 6.1. Vizualizacija grupiranja podataka u dvije grupe ($K = 2$) [26]

U primjeni će se koristiti OpenCV funkcija *kmeans()*. Prvi parametar funkcije je slika čije su vrijednosti prethodno prilagođene za rad s tom funkcijom. Drugi parametar je proizvoljni broj grupa na koje je potrebno podijeliti vrijednosti slike. Četvrti parametar je uređena trojka koja opisuje kriterij po kojemu se zaustavljaju iteracije. To je nekad potrebno iz razloga što može biti velik broj podataka za grupiranje iz slika veće rezolucije te se na taj način smanjuje vrijeme potrebno za izvedbu segmentacije, iako je u petom koraku algoritma naznačeno da se u idealnom slučaju iteracije završavaju kada se položaj centara više ne mijenja. [27] Uređenom trojkom se određuje maksimalni broj iteracija nakon kojih će se segmentacija zaustaviti ili preciznost *epsilon* koja opisuje minimalni pomak centara u pripadnoj iteraciji nakon kojega se segmentacija zaustavlja. Također se u uređenoj trojci može naznačiti da li se želi uzimati u obzir prvi, drugi ili oba kriterija pomoću zastavica *cv2.TERM_CRITERIA_EPS* i *cv2.TERM_CRITERIA_MAX_ITER*. Peti i šesti parametri se odnose na početno postavljanje centara, gdje peti parametar označava koliko će se puta segmentacija izvršiti uz različite početne položaje centara, a šesti je zastavica kojom se specificira način postavljanja početnih centara. Takav način manipuliranja početnim postavljanjem centara je korisno jer se u određenim uvjetima može dogoditi situacija da su početni centri postavljeni na način da segmentacija neće moći razdvojiti vrijednosti slike na korisne grupe. Tada opcija višestrukog izvršavanja algoritma pomoću različitih početnih centara može pomoći.

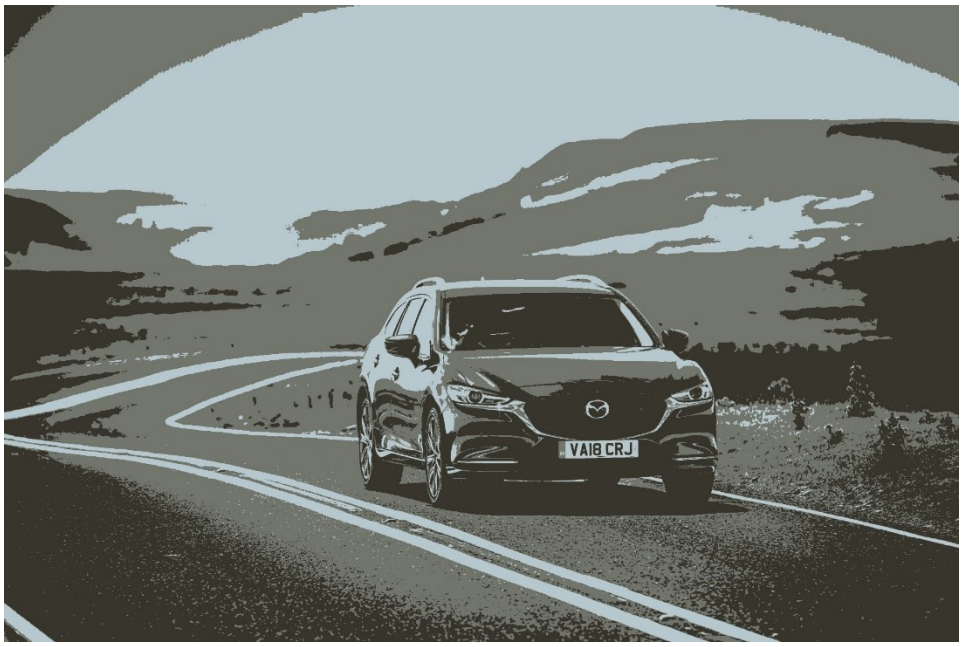
Funkcija *kmeans()* sadrži i tri povratna parametra. Prvi označava zbroj kvadriranih udaljenosti od svake točke do pripadnog centra. Drugi označava svaki element slike s pridruženim brojem grupe kojoj taj element pripada, uzimajući u obzir da brojevi grupe počinju s nulom. Treći parametar označava polje centara grupa, te zajedno s drugim parametrom može se prilagoditi za prikaz novonastale segmentirane slike. Prikaz navedene funkcije s potrebnim prilagodbama se može vidjeti na slici 6.2. Na slici 6.3. je prikaz originalne slike, a na slikama 6.4. i 6.5. su prikazane segmentirane slike s različitim brojem grupa K.

```python
import numpy as np
import cv2

img = cv2.imread('red_mazda.jpg')

# S obzirom da cv2.kmeans() funkcija koristi 2D format slike s float
# vrijednostima, a ne 3D format slike u boji, potrebno je promijeniti
# format slike i pretvoriti brojeve u float tip
Z = img.reshape((-1,3))
Z = np.float32(Z)

# Definiranje kriterija pomoću uređene trojke parametara
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
K = 8

# Upotreba cv2.kmeans() funkcije
ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)

# Nakon segmentacije potrebno je vratiti vrijednosti elemenata slike
# koji su pohranjeni u drugom i trećem povratnom parametru kmeans() funkcije iz
# float tipa u 8-bitni int tip vrijednosti te vratiti u početni oblik slike
# kako bi se slika mogla prikazati pomoću funkcije cv2.imshow()
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))

cv2.imshow('res2',res2)
cv2.imwrite('../Slike/Segmentacija grupiranjem/original.jpg', img)
cv2.imwrite('../Slike/Segmentacija grupiranjem/kmeans8.jpg', res2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Sl. 6.2. Programski kod pomoću kojeg se vrši segmentacija slike grupiranjem s K srednjih vrijednosti [28]

Sl. 6.3. Originalna slika

Sl. 6.4. Segmentirana slika s 8 grupa ($K = 8$)



Sl. 6.5. Segmentirana slika s 3 grupe ($K = 3$)

7. USPOREDBA SEGMENTACIJSKIH METODA

7.1. Amplitudne segmentacije

Sve prethodno navedene segmentacijske metode imaju svoje pozitivne i negativne karakteristike koje određuju na koji način i u kojim ih je slučajevima prigodno koristiti. Kako bi se pojasnile te karakteristike, u ovom poglavlju će se vršiti usporedba metoda na odabranom setu problema.

Zbog svoje jednostavne primjene, uspoređivanje će se započeti s amplitudnom segmentacijom s manualnim postavljanjem praga. Ta segmentacijska metoda je korisna za odvajanje dijelova slike s različitim vrijednostima elemenata te nudi mogućnost manualnog mijenjanja praga po kojem će se to odvajanje vršiti, što omogućuje prilagodbu metode za specifične probleme. Zbog navedenih svojstava ta metoda je prigodna za korištenje na jednostavnijim problemima kada je potrebno odvojiti svjetliju pozadinu od nekog tamnijeg objekta ili dijela slike, što se može vidjeti na slici 7.1.

Sl. 7.1. Prikaz amplitudne segmentacije s manualnim postavljanjem praga uz različite *threshold_type* parametre. [29]

Amplitudna segmentacija s manualnim postavljanjem praga se može koristiti samo na monokromatskim slikama zbog čega je pri obradi slika u boji potrebno koristiti analognu metodu

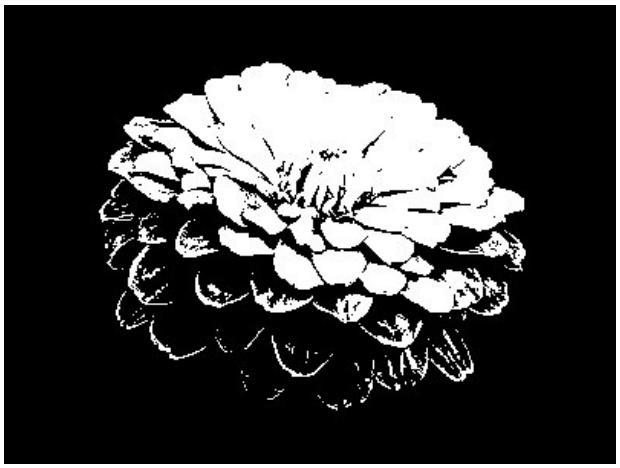
koja postavlja pragove na vrijednosti boja elemenata slike. Takva segmentacijska metoda je korisna kada je potrebno istaknuti dijelove slike koji se sastoje od nijansi određene boje. (slika 7.2.)

Sl. 7.2. Prikaz izdvojenih crvenih nijansi iz originalne slike. [30]

Pri korištenju amplitudne segmentacije s manualnim postavljanjem pragova na slici 7.1. bilo je potrebno prilagoditi prag kako bi se što više zahvatila samo pozadina u segmentiranoj slici. Takav način segmentiranja ide u prilog prilagodljivosti te metode, no u velikom broju slučajeva neće biti očigledno gdje treba postaviti prag kako bi se ravnomjerno segmentirali tamniji i svjetliji dijelovi slike. To se može vidjeti na slici 7.3. gdje je bilo potrebno segmentirati svjetliji cvijet od tamnije pozadine, no to se nije realiziralo jer prag segmentacije nije dobro postavljen.

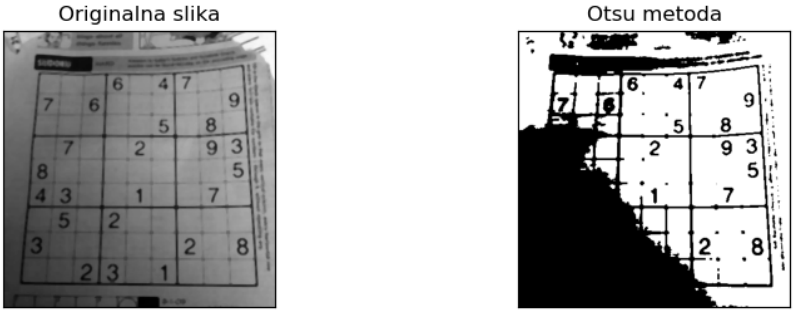
Sl. 7.3. Primjena amplitudne segmentacije s manualno postavljenim pragom. [31]

Taj se problem može riješiti amplitudnom segmentacijom pomoću Otsu metode koja je opisana u potpoglavlju 3.2. Tom se metodom mogu odvojiti dijelovi slike koji čine dva globalna maksimuma u histogramu slike te se na taj način bolje segmentiraju tamnije i svjetlije regije. Primjer takve segmentacije je prikazan na slici 7.4.



Sl. 7.4. Slika segmentirana Otsu metodom

Na slici 7.4. je prikazana segmentirana slika cvijeta koja puno bolje odražava konture cvijeta nego što je to napravljeno amplitudnom segmentacijom s manualnim pragom na slici 7.3. Iako su prethodno uspoređene metode korisne u spomenutim slučajevima, velika mana tih metoda je što koriste određene pragove na cijelom području slike. Ta mana brzo postane očigledna kada se analizira neki objekt ili dio slike koji je potrebno segmentirati, ali je na slici različitog intenziteta, tj. nije ravnomjerno osvjetljen. To se može uočiti na slici 7.5.



Sl. 7.5. Primjena Otsu metode na slici sudoku zagonetke. [32]

Rješenje navedenog problema se može postići korištenjem jedne od obrađenih adaptivnih amplitudnih segmentacija slike (slika 7.6.).

Adaptivna segmentacija na temelju srednje vrijednosti

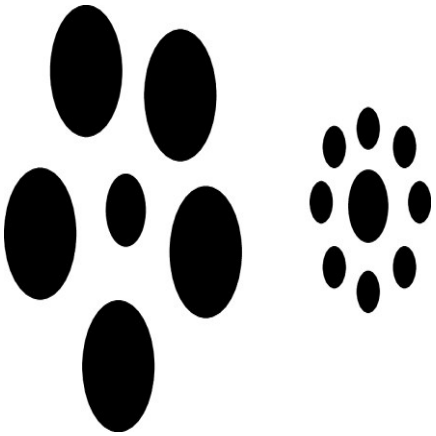
Adaptivna Gaussova segmentacija

Sl. 7.6. Primjena metoda adaptivnih amplitudnih segmentacija

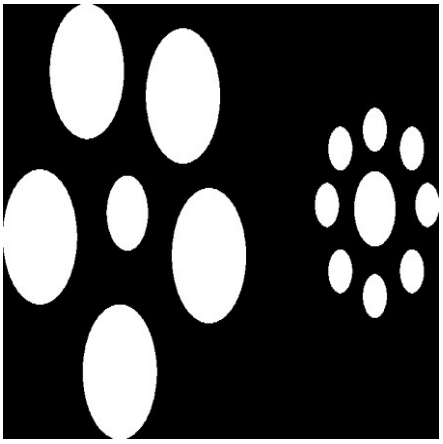
Na slici 7.6. je vidljiva sudoku zagonetka koju je na slici 7.5. teže uočiti zbog promjena osvjetljenja, tj. intenziteta elemenata slike te zbog različitih odsjaja koji također utječu na segmentaciju. Isto tako se može uočiti manje šuma u adaptivnoj Gaussovoj segmentaciji u odnosu na adaptivnu segmentaciju na temelju srednje vrijednosti.

7.2. Segmentacija na temelju rasta područja

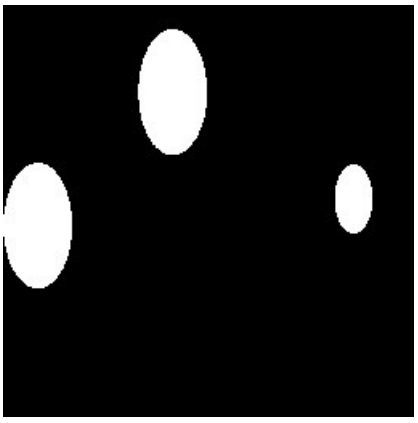
Sve dosad uspoređene metode su djelovale na cijelom području slike te nisu pružale mogućnost obrade samo dijelova slike koji su spojeni osmim stupnjem povezanosti (engl. *8-connectivity*) i koji su ključni za rješavanje određenih problema. Dok je moguće manualno stvoriti određene maske pomoću kojih će se djelovanje tih metoda svesti na određenu regiju slike, često se neće automatski znati u kojem dijelu slike se nalazi željeni objekt. Ako je potrebno naći određeni objekt ili dio slike koji je cjelovit, što znači da su mu svi elementi slike povezani, i pod uvjetom da je područje tog objekta homogeno s obzirom na određeni kriterij, onda se može koristiti segmentacija na temelju rasta područja. Navedena problematika je prikazana na slikama 7.7., 7.8. i 7.9.



Sl. 7.7. Slika optičke iluzije. [33]



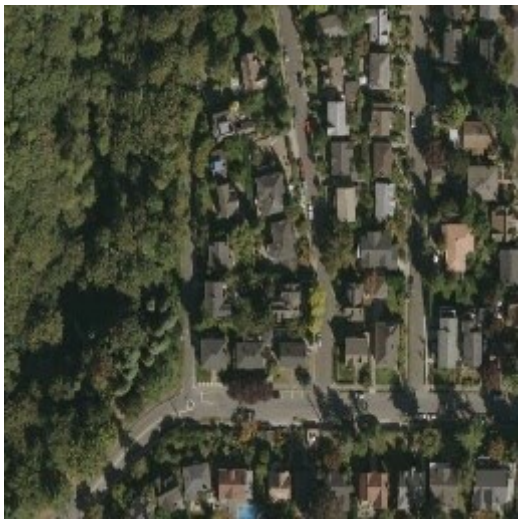
Sl. 7.8. Slika segmentirana amplitudnom segmentacijom s *THRESH_BINARY_INV* parametrom



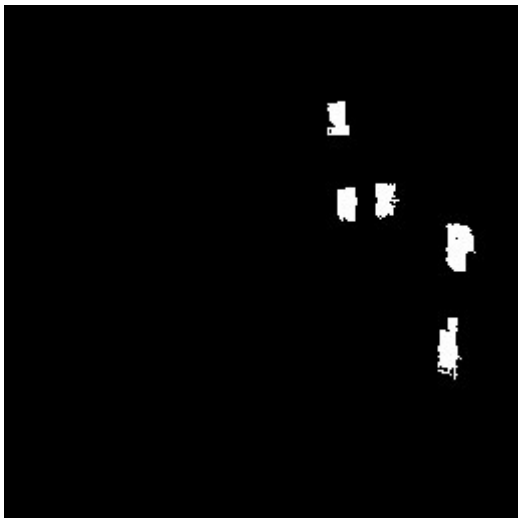
Sl. 7.9. Slika segmentirana metodom segmentacije koja se temelji na rastu područja

Na slici 7.8. je prikazana amplitudno segmentirana slika s parametrom *THRESH_BINARY_INV*. Na njoj se može uočiti da su istaknute sve regije koje ispunjavaju određeni kriterij, u ovom slučaju to je da vrijednosti elemenata slike moraju biti manje od određenog praga. Ako nas zanimaju samo određene cjelovite i istaknute regije te slike, onda se može koristiti segmentacija na temelju rasta područja. Na slici 7.9. su postavljene početne točke (engl. *seed points*) na onim područjima koji su od interesa za problem i koji zadovoljavaju prethodno navedeni kriterij. Na taj način su ti dijelovi slike izolirani od ostalih dijelova koji zadovoljavaju isti kriterij te se može samo njih analizirati.

Metoda rasta područja se može koristiti i u nekim realnim situacijama, kao npr. detektiranje krovova kuća, sagrađene infrastrukture ili geografskih regija u satelitskim slikama. Na slikama 7.10. i 7.11. je prikazan takav način korištenja metode, gdje su korištene početne točke na određenim krovovima kuća.

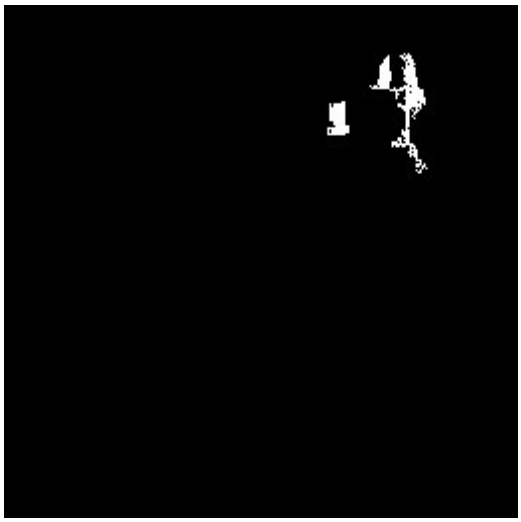


Sl. 7.10. Satelitska slika predgrađa. [34]



Sl. 7.11. Slika predgrađa segmentirana segmentacijom na temelju rasta područja

Ponekad kriterij širenja područja neće biti prigodan za problematiku koja se pokušava riješiti sa segmentacijom zbog čega mogu nastati greške kao naprimjer na slici 7.12. Na toj slici se koristi kriterij sličnosti točke i početne točke pripadnog područja. U slučaju kada se vrijednosti elemenata dijelova slike koji označuju krov ne razlikuju od okolnih elemenata za određenu graničnu vrijednost, onda će regija nekontrolirano rasti izvan željenog područja, što je situacija koju je potrebno izbjeći.

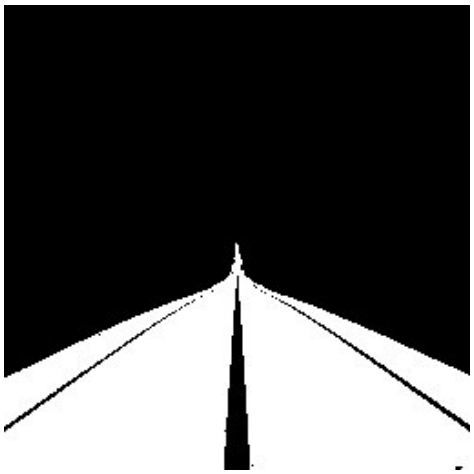


Sl. 7.12. Segmentirana slika predgrađa s neželjenim rastom područja izvan ciljanog objekta

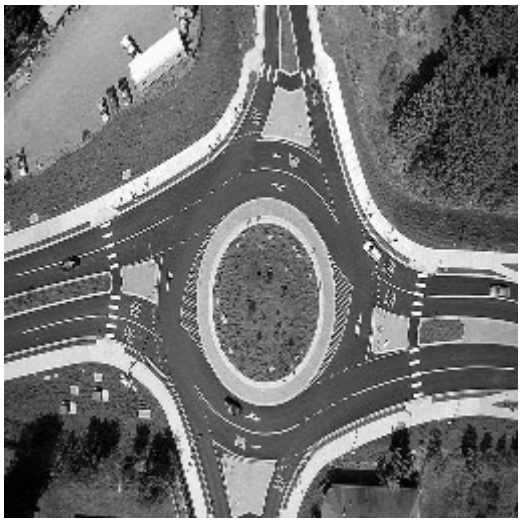
Slučaj na slici 7.12. je dobar pokazatelj mana navedene metode. Metoda rasta područja može imati više kriterija koji se moraju zajedno sa svojim parametrima prilagoditi određenoj problematici za čije rješavanje se koriste. Često će određena područja rasta biti omeđena cjelovitim ili isprekidanim rubovima što može biti dobra značajka za daljnje segmentiranje određenih objekata ili dijelova slike. Kao što je vidljivo na slici 7.12., ako su rubovi krova na slici isprekidani te se elementi slike oko krova ne razlikuju za određenu graničnu vrijednost, to odmah dovodi do nekontroliranog i neželjenog širenja područja.

Unatoč navedenoj mani, segmentacije na temelju rasta područja mogu biti vrlo korisne u prometnim situacijama. To se može vidjeti u sljedećim primjerima prikazanima na slikama 7.13., 7.14., 7.15. i 7.16.

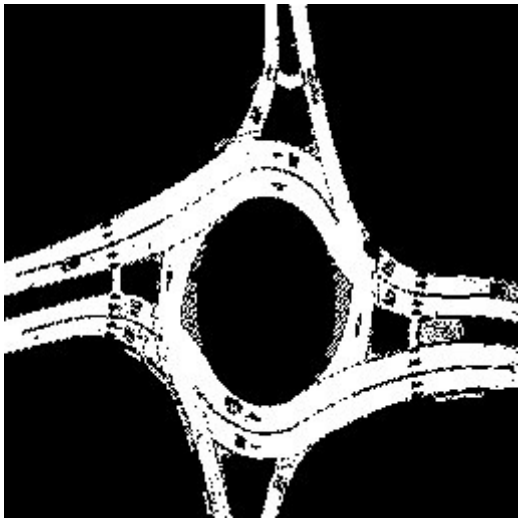
Sl. 7.13. Monokromatska slika ceste. [35]



Sl. 7.14. Segmentirana slika ceste pomoću segmentacije na temelju rasta područja



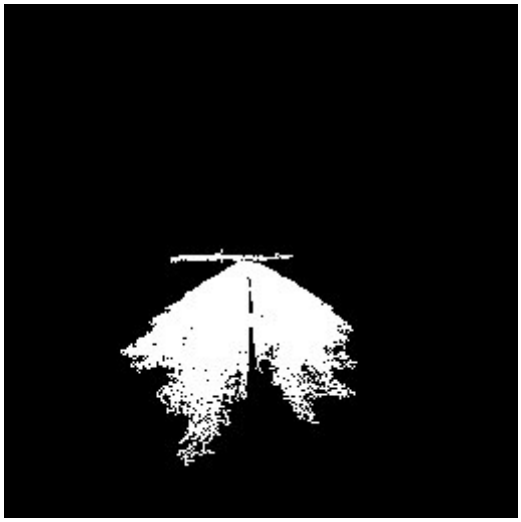
Sl. 7.15. Satelitska slika kružnog toka. [36]



Sl. 7.16. Segmentirana slika kružnog toka pomoću segmentacije na temelju rasta područja

Kao što je vidljivo u prethodnim primjerima, segmentacija na temelju rasta područja je dobra za izdvajanje cesta na slici kada su uvjeti vidljivosti ispunjeni i kada je manjak odsjaja ili nekih drugih smetnji prisutan što bi dovelo do nekontroliranog širenja segmenta ili pak nepotpune segmentacije. Takav slučaj je prikazan na slikama 7.17. i 7.18.

Sl. 7.17. Monokromatska slika ceste s odsjajem. [37]



Sl. 7.18. Segmentirana slika ceste sa slike 7.17. pomoću segmentacije na temelju rasta područja

U slučaju kada segmentacija na temelju rasta područja da krive rezultate, kao na slici 7.18., potrebno je promijeniti ili parametre rasta područja ili njegove uvjete da bi se dobili željeni rezultati, ako je to uopće moguće.

Ponekad će biti potrebno analizirati samo dio slike koji se ne može uvjetima i parametrima rasta područja pravilno izolirati u određenim situacijama. Naprimjer, ako želimo segmentirati samo desnu polovicu ceste na slici 7.13. ili samo područje kružnog toka na slici 7.15. U tim slučajevima želimo ograničiti rast područja u dijelovima slike koji inače zadovoljavaju uvjete rasta i jedan od načina za to je pomoću detektiranih geometrijskih pravilnosti. U tom slučaju je potrebna

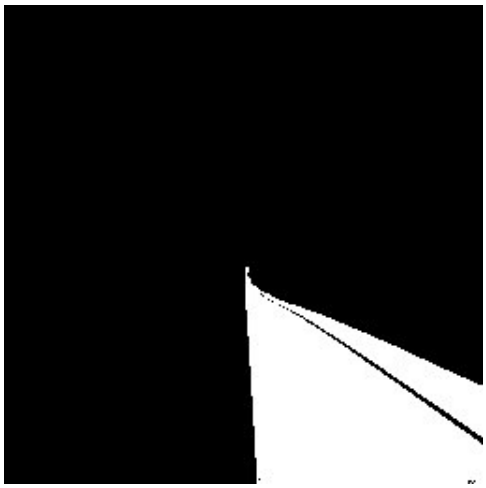
segmentacija pomoću koje će se moći detektirati rubovi i određene pravilnosti na slici koje se mogu ekstrapolirati iz isprekidanih rubova, kao npr. pravci i kružnice.

7.3. Segmentacije na temelju rubova

Za razliku od prethodno uspoređenih metoda za segmentaciju digitalne slike, segmentacijska metoda na temelju rubova ne ispituje samo vrijednosti elemenata slike i njihovu povezanost, već pomoću kompleksnih algoritama opisanih u 4. poglavlju detektira rubove i geometrijske pravilnosti koje su prisutne na promatranoj slici. To se može demonstrirati na problemu iz prošlog potpoglavlja.

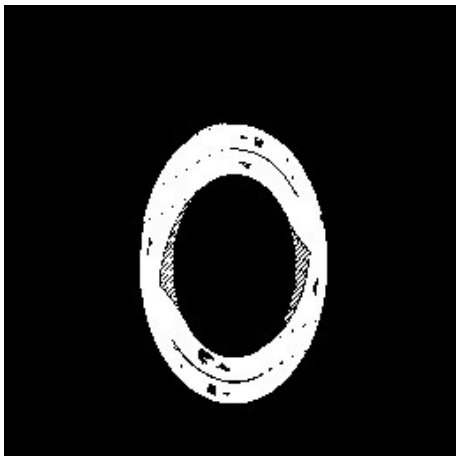
Ako je potrebno u određenim okolnostima analizirati manji dio slike koji se može dobiti njegovim odvajanjem od ostatka slike pomoću ekstrapoliranih pravaca i kružnica, onda je ova segmentacija prikladna. To je upravo potrebno napraviti na slici 7.13. kako bi se analizirala samo desna polovica ceste na kojoj se vozilo većinu vremena nalazi i na slici 7.15. kako bi se segmentiralo samo područje unutar kružnog toka. U tu svrhu se koristi metoda detekcije pravaca i kružnica pomoću Hough-ove transformacije. Detektirani pravac i kružnica su prikazani na slikama 7.19. i 7.21., dok su na slikama 7.20. i 7.22. prikazane segmentirane slike na temelju detektiranog pravca i kružnice pomoću segmentacije na temelju rasta područja.

Sl. 7.19. Cesta sa slike 7.13. s ucrtanim detektiranim pravcem pomoću Hough-ove transformacije



Sl. 7.20. Cesta s detektiranim pravcem (slika 7.19.) kojoj je segmentirana desna polovica pomoću segmentacije na temelju rasta područja

Sl. 7.21. Kružni tok sa slike 7.15. s ucrtanom detektiranom kružnicom pomoću Hough-ove transformacije



Sl. 7.22. Kružni tok s detektiranom kružnicom (slika 7.21.) koji je izdvojen od okolnih cesti pomoću segmentacije na temelju rasta područja.

Iako su u prethodnim primjerima segmentacije na temelju rubova bile korisne, to nije uvijek slučaj. Ponekad se može dogoditi da se neće detektirati kružnice, pravci ili krivulje koje je potrebno pronaći, ali se isto tako može dogoditi da se detektiraju, ali na krivim mjestima. To je upravo mana ovakve vrste segmentacije jer je osjetljiva na uvjete na slici i parametre koji se koriste pri pozivu funkcije. Na slikama 7.23. i 7.24. su prikazane ovakve vrste neželjenih situacija pri detektiranju pravaca i kružnica.

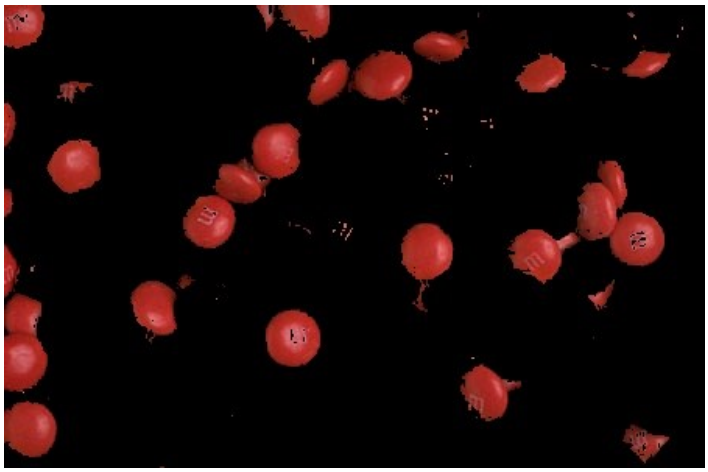
Sl. 7.23. Cesta sa slike 7.13. na kojoj su ucrtani pravci koji su nepotrebni za detekciju ceste ili pojedinih prometnih traka

Sl. 7.24. Kružni tok sa slike 7.15. s ucrtanim kružnicama koje su nepotrebne za nalaženje područja kružnog toka

7.4.Segmentacija grupiranjem

Metoda segmentacije grupiranjem, opisana u 6. poglavlju, izvršava se na slikama u boji te će se stoga uspoređivati s amplitudnom segmentacijom digitalnih slika u boji. Dok je amplitudna segmentacija prigodna za isticanje određenog raspona nijansi boja, segmentacije grupiranjem služe za diferenciranje određenog broja grupa nijansi algoritmom opisanim u 6. poglavlju. Razlika se može vidjeti kada se primijene navedene segmentacije na slici 7.25. te se dobiju segmentirane slike 7.26. i 7.27.

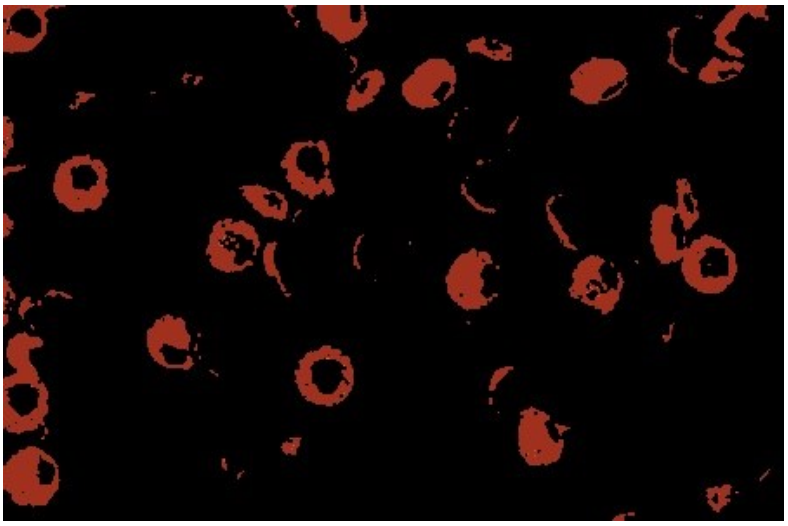
Sl. 7.25. Slika slatkiša. [38]



Sl. 7.26. Slatkiši crvene boje segmentirani pomoću amplitudne segmentacije

Sl. 7.27. Segmentirana slika slatkiša pomoću segmentacije grupiranjem ($K = 8$)

Pri segmentiranju slike grupiranjem potrebno je bilo odrediti broj grupa na koje će se svesti sve nijanse boja. U slučaju segmentirane slike 7.27. bilo je određeno 8 grupa zbog šest boja slatkiša, crnih područja između njih i boje odsjaja. Nakon što se na dobivenoj slici 7.27. segmentira crvena boja, dobije se nova segmentirana slika 7.28.



Sl. 7.28. Primijenjena amplitudna segmentacija crvenih nijansi slike 6.27.

Uspoređujući slike 7.26. i 7.28. može se zaključiti da je za detekciju objekata pomoću određenih boja bolja i fleksibilnija amplitudna segmentacija jer je bolje obuhvatila pojedine objekte od segmentacije grupiranjem te se pri tome manualno mogu odabirati željene nijanse. Unatoč tome, segmentacija grupiranjem se može koristiti kada je potrebno segmentirati objekt određene boje, a da pri tome na slici nema objekata koji su iste te boje (slika 7.29. i 7.30.). Pri takvom korištenju segmentacije također je potrebno paziti da broj grupa nije prenizak jer se onda može dogoditi da boje traženog objekta nisu stavljene u željenu grupu ili su stavljene u grupu s bojama okoline, što se može vidjeti na slici 7.31.

Sl. 7.29. Crveni autobus na kat. [39]

Sl. 7.30. Segmentirana slika crvenog autobusa pomoću segmentacije grupiranjem ($K = 3$)

Sl. 7.31. Segmentirana slika crvenog autobusa pomoću segmentacije grupiranjem ($K = 2$)

8. ZAKLJUČAK

U ovom završnom radu obrađene su i uspoređene metode segmentacije digitalnih slika. Prvo su opisane njihove teorijske podloge, a zatim i način njihove primjene u Python programskom jeziku pomoću OpenCV programske biblioteke. Nakon što je proučena svaka od metoda pojedinačno, slijedilo je uspoređivanje metoda na odabranom setu problema, što je glavni zadatak ovog završnog rada.

Uspoređivanje je započelo s metodama amplitudne segmentacije. Amplitudna segmentacija s manualno postavljenim pragom se pokazala kao fleksibilna metoda zbog mogućnosti postavljanja praga, no isto tako nije prigodna za korištenje kada nije očito na slici gdje bi prag trebao biti postavljen kako bi se segmentirao objekt ili regija slike koju je potrebno izdvojiti. U tom slučaju se pokazalo da je bolje koristiti amplitudnu segmentaciju Otsu metodom jer se pomoću te segmentacije pronalazi optimalan prag koji razdvaja svjetlije i tamnije dijelove slike. Iako se Otsu metodom riješio navedeni problem, u određenim situacijama regije i objekti slike nisu jednako osvjetljeni što je stvaralo probleme pri njihovom segmentiranju samo s jednim pragom. Kako bi se riješio problem u navedenoj situaciji je bolje koristiti metode adaptivne amplitudne segmentacije. Takve metode su bolje u segmentiranju slika s različitim osvjetljenjem jer određuju više lokalnih pragova na temelju lokalnih svojstava.

Također su proučene i metode na temelju rasta područja. Takve metode su korisnije od metoda amplitudne segmentacije ako je potrebno segmentirati i analizirati pojedinu regiju slike čiji su elementi međusobno dodiruju, što može biti vrlo korisno za razlikovanje pojedinih objekata. No s obzirom da se uvjet rasta područja i pojedini parametri mogu definirati manualno unutar pripadne funkcije, ponekad ih je potrebno podešavati kako bi se dobio potreban rezultat. U određenim situacijama je potrebno ograničiti rast područja na temelju određenih značajki slike koje se ne mogu opisati uvjetima rasta područja. Neke od tih značajki, kao što su pravci, kružnice ili neke druge krivulje, mogu se detektirati pomoću segmentacija na temelju rubova te zajedno s metodama na temelju rasta područja čine koristan alat za detekciju i prepoznavanje objekata koji su određeni spomenutim ograničenjima. Iako su na taj način korisne segmentacije na temelju rubova, ponekad pri korištenju pripadnih algoritama će se detektirati određene značajke slike koje su nepotrebne ili se pak neće detektirati one značajke slike koje je potrebno detektirati. U takvoj situaciji se mogu promijeniti parametri segmentacije kako bi se detekcija mogla prilagoditi očekivanim uvjetima na slici.

Pri samom kraju uspoređivanja promatrane su i metode koje vrše segmentaciju na temelju vrijednosti boja elemenata slike. U usporedbi je uočeno kako je amplitudna segmentacija digitalne slike u boji fleksibilnija i bolje segmentira objekte na temelju boje nego segmentacija grupiranjem.

Iz provedenih usporedbi metoda segmentacije može se zaključiti da svaka od analiziranih metoda ima određene pozitivne i negativne karakteristike u ovisnosti o parametrima koji su joj proslijeđeni i o karakteristikama slike koja se segmentira. Također je uočeno da se metode mogu kombinirati u određenim situacijama kako bi se iskoristile pozitivne karakteristike više metoda. Fleksibilnost i raznovrsnost metoda i njihovih karakteristika idu u prilog njihovoj širokoj primjeni te činjenici da se te metode koriste kao temelj nekih naprednijih algoritama za detekciju objekata i područja digitalne slike.

LITERATURA

[1] W. Wu, A. Y. C. Chen, L. Zhao and J. J. Corso: "Brain Tumor detection and segmentation in a CRF framework with pixel-pairwise affinity and super pixel-level features", International Journal of Computer Aided Radiology and Surgery, Vol. 9., pp. 241–253, Srpanj, 2013.

[2] S. Kamalakannan ; A. Gururajan ; H. Sari-Sarraf ; R. Long ; S. Antani. "Double-Edge Detection of Radiographic Lumbar Vertebrae Images Using Pressurized Open DGVF Snakes". IEEE Transactions on Biomedical Engineering. No. 6, Vol. 57, pp. 1325–1334., Veljača 2010.

[3] J. M. Alvarez; T. Gevers; Y. LeCun; A. M. Lopez, Road Scene Segmentation from a Single Image, Lecture Notes in Computer Science, Vol. 7578, pp. 376 – 389, Springer, Berlin, Heidelberg, 2012.

[4] P. A. Kandalkar; G. P. Dhok, Image Processing Based Vehicle Detection And Tracking System, International Advanced Research Journal in Science, Engineering and Technology, No. 11, Vol. 4, pp. 186 – 190, Studeni 2017.

[5] A. Cheddad; D. Mohamad, Segmentation Methods in Face Recognition: Face segmentation, semanticscholar.org, dostupno na: https://pdfs.semanticscholar.org/8872/8ec89044a4706737e68c1bb7a16450615f1c.pdf [19.06.2020.]

[6] S. Bana; Dr. D. Kaur, Fingerprint Recognition using Image Segmentation, International Journal of Advanced Engineering, Sciences and Technologies, No. 1, Vol. 5, pp. 12-23

[7] R. C. Gonzales, R. E. Woods, Digital Image Processing Third Edition, Pearson Education, Inc., Upper Saddle River, New Jersey, 2008.

[8] Is the Mazda 6 one of the best built cars on the road?: Northern Ireland Business First, 2018., dostupno na: https://www.businessfirstonline.co.uk/articles/is-the-mazda-6-one-of-the-best-built-cars-on-the-road/ [19.06.2020.]

[9] A. Mordvintsev, Abid K., Image Thresholding: Adaptive Thresholding, OpenCV – Python Tutorials, 2013. dostupno na: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html [19.06.2020.]

[10] How to define a threshold value to detect only green colour objects in an image, Stack overflow, 2017., dostupno na: https://stackoverflow.com/questions/47483951/how-to-define-a-threshold-value-to-detect-only-green-colour-objects-in-an-image [19.06.2020.]

[11] H. D. Cheng, X. H. Jiang, Y. Sun, W. Jingli, Color image segmentation: Advances and prospects, Pattern Recognition 34, Department of Computer Science, Utah State University, 2001.

[12] D. Gilbušić, Struna: Konvolucija, Institut za hrvatski jezik i jezikoslovlje, 2011., dostupno na: http://struna.ihjj.hr/naziv/konvolucija/19228/ [19.06.2020.]

[13] J. Padarian, Example of the first 3 steps of a convolution of a 3x3 filter over a 5x5 array image: Fig 1, ResearchGate, 2018., dostupno na: https://www.researchgate.net/figure/Example-of-the-first-3-steps-of-a-convolution-of-a-3x3-filter-over-a-5x5-array-image_fig1_329241581 [19.06.2020.]

[14] S. Madenda, Kernel of a Prewitt operator and b Sobel operators: fig 2, Researchgate, 2016., dostupno na: https://www.researchgate.net/figure/Kernel-of-a-Prewitt-operator-and-b-Sobel-operators_fig2_311605340 [19.06.2020.]

[15] M. Thurley, The Hough Transform: The Hough transform – polar coordinates, Itu.se, , dostupno na: https://www.ltu.se/cms_fs/1.36192!/e0005e_lecture05_hough_transform.dvi.pdf [19.06.2020.]

[16] P. Corke, Finding Line Features: QUT Robot Academy, dostupno na: https://robotacademy.net.au/lesson/finding-line-features/ [19.06.2020.]

[17] A. Mordvintsev, Abid K., Hough Line Transform: Hough Transform in OpenCV, OpenCV-Python Tutorials, 2013., dostupno na: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html [19.06.2020.]

[18] ADAS and Autonomous Vehicles: Case Study – Out-of-Domain Object Detection, Level five safety, dostupno na: http://level-five-safety.ai/automotive/ [19.06.2020.]

[19] Circle Hough Transform (CHT): Cis.rit.edu, dostupno na: https://www.cis.rit.edu/class/simg782.old/talkHough/HoughLecCircles.html [19.06.2020.]

[20] X. Zou, Method of the standard Hough transform circle detection: Researchgate.net, 2014., dostupno na: https://www.researchgate.net/figure/Method-of-the-standard-Hough-transform-circle-detection_fig1_334093731 [19.06.2020.]

[21] Feature Detection: HoughCircles(), opencv.org, dostupno na: https://docs.opencv.org/master/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d [19.06.2020.]

[22] A. Mordvintsev, Abid K., Hough Circle Transform: Theory, OpenCV-Python Tutorials, 2013., dostupno na: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html [19.06.2020.]

[23] Traffic sign detection – selefdriving car – deep learning series 3: Introduction, mc.ai, 2018., dostupno na: https://mc.ai/traffic-sign-detection-selefdriving-car-deep-learning-series-3/ [19.06.2020.]

[24] S. Lončarić, Segmentacija slike: Fakultet elektrotehnike i računarstva, Zagreb, dostupno na: https://www.fer.unizg.hr/_download/repository/09-OI-SegmentacijaSlike[2].pdf [19.06.2020.]

[25] Region Growing python: Stackoverflow.com, 2017., dostupno na: https://stackoverflow.com/questions/43923648/region-growing-python [19.06.2020.]

[26] S.Towers, K means clustering: Polymatheia, 2013., dostupno na: http://sherrytowers.com/2013/10/24/k-means-clustering/ [19.06.2020.]

[27] A. Rockikz, How to Use K-Means Clustering for Image Segmentation using OpenCV in Python: PythonCode, 2020., dostupno na: https://www.thepythoncode.com/article/kmeans-for-image-segmentation-opencv-python [19.06.2020.]

[28] A. Mordvintsev & Abid K., K-Means Clustering in OpenCV: Color Quantization, OpenCV-Python Tutorials, 2013., dostupno na: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html [19.06.2020.]

[29] do-not-fear (originalna slika), dostuno na: http://www.christianmessenger.in/wp-content/uploads/2019/03/do-not-fear-696x392.jpg [19.06.2020.]

[30] Toyota yaris (originalna slika), dostupno na: http://www.kojiauto.hr/toyota-yaris-1-33-bi-tone-red/ [19.06.2020.]

[31] Zinnia elegans (originalna slika), dostupno na: https://en.wikipedia.org/wiki/Wikipedia:Featured_pictures/Plants/Flowers [19.06.2020.]

[32] Sudoku (originalna slika), dostupno na: https://medium.com/analytics-vidhya/smart-sudoku-solver-using-opencv-and-tensorflow-in-python3-3c8f42ca80aa [19.06.2020.]

[33] Optička iluzija, dostupno na: https://d2gn4xht817m0g.cloudfront.net/p/platform/story_thumbnails/images/giant/000/004/397/4397-22d080b04fc38288e06ad03a49ebfc170b8bd644.?1565972320 [19.06.2020.]

[34] Satelitska slika predgrađa, dostupno na: https://phys.org/news/2012-08-satellite-view-house.html [19.06.2020.]

[35] Monokromatska slika ceste, dostupno na: https://stepfeed.com/this-saudi-highway-is-the-world-s-longest-straight-road-2869 [19.06.2020.]

[36] Satelitska slika kružnog toka, dostupno na: https://pbs.twimg.com/media/EVv0-vbWAAEqbkc.jpg:large [19.06.2020.]

[37] Monokromatska slika ceste s odsjajem, dostupno na: https://plaviured.hr/wp-content/uploads/2015/10/road-220058_640.jpg [19.06.2020.]

[38] Slika slatkiša, dostupno na: https://1.bp.blogspot.com/_aiBihDyfxYE/TGUuaqQjnCI/AAAAAAAABLI/2Li7DDMyqSQ/w1200-h630-p-k-no-nu/PH02176J.jpg [19.06.2020.]

[39] Crveni autobus na kat, dostupno na: https://media-cdn.tripadvisor.com/media/photo-s/14/e8/71/85/bus-on-the-move.jpg [19.06.2020.]

SAŽETAK

U ovom radu opisana je teoretska podloga segmentacijskih metoda i algoritama za obradu digitalne slike te je napravljena njihova usporedba na odabranom setu problema. Obrađene segmentacijske metode su: amplitudne segmentacije, segmentacije na temelju rubova, segmentacije na temelju rasta područja i segmentacija grupiranjem. Primjena i usporedba metoda je provedena u Python programskom jeziku pomoću OpenCV programske biblioteke. Uspoređivanje se vršilo na metodama sa zajedničkim karakteristikama kako bi se istakla njihova pozitivna i negativna svojstva pri određenim uvjetima na slici. Uspoređivanjem segmentacijskih metoda došlo se do zaključka da svaka od metoda ima pozitivne i negativne karakteristike koje ovise o karakteristikama slike i vrsti problema kojeg je potrebno riješiti segmentacijom, tj. vrsti objekta ili područja kojeg je potrebno segmentirati.

Ključne riječi: segmentacijske metode, amplitudna segmentacija, segmentacija na temelju rubova, segmentacija na temelju rasta područja, segmentacija grupiranjem

ABSTRACT

This paper describes theoretical basis of segmentation methods and algorithms for digital image processing and compares them on selected set of problems. Main segmentation methods that are analysed in this paper are: thresholding, segmentation based on edge detection, segmentation based on region growing and segmentation based on clustering. Application and comparison of the segmentation methods was made using Python programming language and OpenCV library. Comparison was made on methods with common characteristics to point out their positive and negative properties in regards to the properties of the image that was being segmented. After that it was concluded that every method that was analysed had positive and negative characteristics which depended on properties of the segmented image and type of the problem that needed to be solved using a particular method.

Keywords: segmentation methods, thresholding, segmentation based on edge detection, segmentation based on region growing, segmentation based on clustering

ŽIVOTOPIS

Nikola Danilović rođen je 1. kolovoza 1998. u Vukovaru, Republika Hrvatska. Pohađao je Osnovnu školu Tenja te nakon završetka osnovne škole pohađa III. gimnaziju u Osijeku, prirodoslovno – matematički smjer. Godine 2017. upisuje Preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku te je trenutno na trećoj godini studija.

PRILOG

P.4.1. [25]:

```
import cv2
import numpy as np

def get8n(x, y, shape):
    out = []
    maxx = shape[1]-1
    maxy = shape[0]-1

    #top left
    outx = min(max(x-1,0),maxx)
    outy = min(max(y-1,0),maxy)
    out.append((outx,outy))

    #top center
    outx = x
    outy = min(max(y-1,0),maxy)
    out.append((outx,outy))

    #top right
    outx = min(max(x+1,0),maxx)
    outy = min(max(y-1,0),maxy)
    out.append((outx,outy))

    #left
    outx = min(max(x-1,0),maxx)
    outy = y
    out.append((outx,outy))

    #right
    outx = min(max(x+1,0),maxx)
    outy = y
    out.append((outx,outy))

    #bottom left
    outx = min(max(x-1,0),maxx)
    outy = min(max(y+1,0),maxy)
    out.append((outx,outy))

    #bottom center
    outx = x
    outy = min(max(y+1,0),maxy)
    out.append((outx,outy))

    #bottom right
    outx = min(max(x+1,0),maxx)
    outy = min(max(y+1,0),maxy)
    out.append((outx,outy))

    return out

def region_growing(img, seed, threshold):
    list = []
    outimg = np.zeros_like(img)
    list.append((seed[0], seed[1]))
```

```
    processed = []
    while(len(list) > 0):
        pix = list[0]
        outimg[pix[0], pix[1]] = 255
        for coord in get8n(pix[0], pix[1], img.shape):
            if img[coord[0], coord[1]] >= img[seed[0], seed[1]] - threshold and \
                    img[coord[0], coord[1]] <= img[seed[0], seed[1]] + threshold:
                outimg[coord[0], coord[1]] = 255
                if not coord in processed:
                    list.append(coord)
                processed.append(coord)
        list.pop(0)
        cv2.imshow("progress",outimg)
        cv2.waitKey(1)
    return outimg

def on_mouse(event, x, y, flags, params):
    if event == cv2.EVENT_LBUTTONDOWN:
        print('Seed: ' + str(x) + ', ' + str(y), img[y,x])
        clicks.append((y,x))

clicks = []
img = cv2.imread('white_car_and_bike.jpg', 0)

img = cv2.resize(img,(256,256))

cv2.namedWindow('Input')
cv2.setMouseCallback('Input', on_mouse, 0, )
cv2.imshow('Input', img)
cv2.waitKey()
seed = clicks[-1]
out = np.zeros_like(img)
for seed in clicks:
    out += region_growing(img, seed, 15)

cv2.imshow('Region Growing', out)
cv2.waitKey()
cv2.destroyAllWindows()
```