

ELEKTROTEHNIČKI FAKULTET SARAJEVO

SEMINARSKI RAD IZ PREDMETA

DIGITALNO PROCESIRANJE SIGNALA

---

**Korištenje tehnika morfološke obrade slika na  
prepoznavanje karaktera registarskih tablica  
automobila u BiH**

---

*Autori*

Adnan ARNAUTOVIĆ

Edvin TESKEREDŽIĆ

*Mentor*

Doc. Dr. Amila AKAGIĆ

21.5.2018.

# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
1.1	O morfologiji slike . . . . .	2
<b>2</b>	<b>Osnovni morfološki pojmovi i operacije</b>	<b>3</b>
2.1	Grayscale i binarne slike . . . . .	3
2.2	Thresholding . . . . .	4
2.3	Prepoznavanje granica (edge detection) . . . . .	4
2.4	Dilacija i erozija . . . . .	5
<b>3</b>	<b>Složene morfološke operacije</b>	<b>7</b>
3.1	Morfološko otvorenje i zatvorenje . . . . .	7
3.2	Top hat i bottom hat transformacije . . . . .	8
3.3	Hit-or-miss . . . . .	9
<b>4</b>	<b>Osnovni morfološki algoritmi</b>	<b>10</b>
4.1	Izdvajanje granica . . . . .	10
4.2	Punjenje regija . . . . .	10
4.3	Izdvajanje povezanih komponenti . . . . .	11
<b>5</b>	<b>Primjena i opis postupka za ANPR</b>	<b>12</b>
5.1	Ukratko o računarskom vidu i ANPR . . . . .	12
5.2	Generalni opis algoritma i korištenih heuristika . . . . .	14
5.3	Ekstrakcija tablica . . . . .	14
5.4	Prepoznavanje karaktera . . . . .	17
<b>6</b>	<b>Zaključak</b>	<b>18</b>
<b>7</b>	<b>Kodovi</b>	<b>19</b>

# 1 Uvod

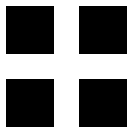
Cilj rada jeste ekstrakcija bosanskohercegovačkih registarskih tablica iz predefinisano seta slika, te prepoznavanje znakova same tablice korištenjem programskog alata MATLAB. Algoritam za automatsko prepoznavanje registarskih tablica se zasniva na tehnikama morfološke obrade slika, kao što su izdvajanje povezanih komponenti, i templating. Prvenstveno ćemo predstaviti neke od osnovnih morfoloških pojmova, te njihovu važnost u digitalnoj obradi slika. Zatim će biti dat kratki opis osnovnih algoritama u morfologiji, kao i njihova uloga i primjena u stvarnom životu. Na kraju ćemo demonstrirati predstavljene pojmove i algoritme na konkretnom primjeru, to jeste na prepoznavanju registarskih tablica automobila.

**Ključni pojmovi:** Morphology of images, ANPR, MATLAB.

## 1.1 O morfologiji slike

U digitalnoj obradi slika često je od važnosti izdvajanje oblika koji su nam od interesa. Ti oblici mogu biti različiti u ovisnosti od zadatka ispred nas: od najjednostavnijih kao što su, na primjer, kružnice, trogulovali, i ostale elementarne geometrijske figure, pa sve do onih složenijih oblika, kao što je oblik ljudske ruke, lica, itd. Oblast nauke koja se pokazala kao pogodna za rješavanje ovih problema jeste matematička morfologija. Matematička morfologija predstavlja veliku granu u digitalnoj obradi signala, koja se bavi prepoznavanjem, opisom, i izdvajanjem oblika (odnosno, regija od interesa) iz slika. Iste tehnike se također koriste za pre- i post-processing slika, u kombinaciji sa algoritmima iz drugih oblasti.

Osnovna ideja morfoloških algoritama je, u principu, veoma jednostavna: Neka je kao ulaz zadan neki nepoznat skup podataka (odnosno slika), nad kojim je potrebno izvršiti transformacije u cilju ekstrakcije ili naglašavanja određenih karakteristika tog skupa. Da bi se te transformacije mogle izvršiti uspješno, od velike važnosti je takozvani strukturni element (eng. *structuring element*). Strukturni element predstavlja unaprijed zadan skup podataka, čije su karakteristike dobro poznate programeru koji implementira algoritam. Dizajn adekvatnog strukturnog elementa za konkretan problem predstavlja temelj za uspješnost algoritma. Opisano najosnovnijim terminima, strukturni element predstavlja predefinisani oblik koji sa ulaznim setom podataka vrši interakciju, i kao povratni podatak daje nivo poklapanja oblika unutar ulaza sa samim strukturnim elementom. U računarskoj primjeni, strukturni element je obično zadan kao matrica (konkretno, matrica jedinica i nula gdje jedinice predstavljaju bijele piksele, a nule crne piksele). Primjer jednog strukturnog elementa oblika plusa i dimenzija 5x5 je dat ispod.

$$SE = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$


Slika 1: Strukturni element oblika plusa u memoriji, kao i interpretacija algoritma

Danas se morfološki algoritmi u primjenama često koriste u kombinaciji sa drugim granama računarskih nauka.

Posebno popularni su algoritmi koji za pre-processing slike koriste morfološke tehnike, da bi onda za prepoznavanje i ekstrakciju određenih karakteristika inkorporirali i tehnike iz oblasti mašinskog učenja. Neke od primjena matematske morfologije u sklopu digitalnog procesiranja signala se mogu naći u oblastima kao što su geologija, biomedicinsko mapiranje, video nadzor, analiza građevnih materijala, robotika, i mnogi drugi.

U narednom dijelu ćemo dati kratki opis osnovnih morfoloških pojmova i algoritama, kao i predstaviti njihovu vezu sa naučnom granom Računarskog vida (eng. *Computer vision*). Demonstrirat ćemo morfološke algoritme na primjeru automatkog prepoznavanja registarskih tablica (eng. *ANPR - automatic number plate recognition*).

## 2 Osnovni morfološki pojmovi i operacije

### 2.1 Grayscale i binarne slike

Većina morfoloških operatora se ne može primijeniti na uobičajenu RGB sliku na koju smo navikli, već zahtijeva posebne vrste slika: grayscale i binarne slike.

U mnogim algoritmima zasnovanim na morfološkim operatorima (pa tako i u ANPR sistemima), prvi korak predstavlja upravo konverziju RGB slike dobivene kao ulaz u grayscale. Grayscale predstavlja tip slike kod kojeg svaki piksel nosi isključivo informaciju o osvijetljenosti. Ovakve slike su sastavljene od sivih piksela, gdje nijanse sive koje su bliže bijeloj boji označavaju veću, a one koje su bliže crnoj označavaju manju osvijetljenost. Potpuno osvijetljeni dijelovi slike su predstavljeni bijelim pikselima, dok su potpuno tamni dijelovi slike crni pikseli. Grayscale slike dakle prenose informaciju o tome koliko je taman (odnosno svijetao) objekt ljudskom oku. Ako se RGB slika želi konvertovati u grayscale, koristi se sljedeća formula, koja daje dobre rezultate neovisno od prirode same slike (koeficijenti koji stoje uz odgovarajuće vrijednosti R, G, i B odgovaraju osjetljivosti ljudskog oka na odgovarajuću boju):

$$GrayLevel = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Druga važna vrsta slike u morfologiji jeste binarna slika. Za razliku od grayscale slika, kod kojih je svaki piksel određena nijansa sive boje i nosi informaciju o osvijetljenosti, binarne slike imaju samo dvije moguće vrijednosti piksela: potpuno bijelo ili potpuno crno. To znači da binarne slike nose isključivo informaciju o postojanju objekta u određenom dijelu slike, gdje obično bijeli pikseli označavaju da postoji objekt od interesa, dok crni označavaju pozadinu (odnosno dijelove slike koji za algoritam nisu od važnosti). U morfologiji, binarne slike igraju veoma važnu ulogu u identificiranju tzv. regija od interesa (eng. *ROI - Region of interest*), o kojima će više govora biti kasnije. Za produciranje binarne slike iz ulazne RGB slike postoji više metoda, ali su većina zasnovane na binarnom izobraženju (eng. *thresholding*). Na slici ispod je prikazana razlika između standardne RGB slike, grayscale slike, te binarne slike:



Slika 2: Ulazna RGB slika, njena grayscale verzija, te binarna verzija slike (slika preuzeta sa [www.pixabay.com](http://www.pixabay.com))

## 2.2 Thresholding

U digitalnoj obradi slika, thresholding predstavlja osnovnu metodu dobijanja binarne slike iz ulazne RGB slike (odnosno, iz grayscale slike dobivene konverzijom ulaza). Ideja na kojoj se zasniva thresholding je veoma jednostavna: svaki sivi piksel će se, u ovisnosti od toga koliko je blizu bijeloj ili crnoj boji, konvertovati u bijeli ili crni piksel. Granična vrijednost koja odlučuje o tome da li će piksel postati bijel ili crn naziva se prag (eng. *threshold*). Neka je, na primjer, data grayscale slika sa 64 nivoa sive boje, gdje je 0 potpuna tama (crno), a 63 potpuna osvijetljenost (bijelo). Ako sada želimo dobiti binarnu sliku, možemo postaviti prag na 50%. To znači da će se svaki piksel čija je nijansa sive između 0 i 31 (dakle, prva polovina vrijednosti), konvertovati u crni piksel, dok će pikseli vrijednosti od 32 do 63 biti pretvorene u bijele piksele. Razumije se da je odabir praga od iznimne važnosti u mnogim algoritmima, te da prag od 50% (koji je korišten na slici 3) neće uvijek dati idealne rezultate.

Kako bi se odredio optimalan prag u ovisnosti od same prirode slike, razvijene su mnoge metode. Jedna od najjednostavnijih je Metod Otsua (eng. *Otsu's method* - nazvan po japanskom naučniku Nobuyuki Otsuu). Obzirom da je i ovaj metod korišten u ANPR sistemima, ukratko ćemo ga opisati: ovaj metod koristi statističke podatke dobivene iz grayscale slike pomoću kojih se određuje optimalni prag za binarizaciju slike. Računa se varijansa, kao mjera 'rasutosti' nijansi sive boje, te se onda formiraju dvije klase (eng. *clustering*), i to na takav način da je varijansa unutar same klase što manja, ali varijansa između samih klasa što veća. Prag za koji se dobije takav rezultat je optimalni prag, koji algoritam vraća kao rezultat.

## 2.3 Prepoznavanje granica (edge detection)

Pojam usko povezan sa binarnim slikama i thresholding-om jeste prepoznavanje granica (eng. *edge detection*). Prepoznavanje granica predstavlja operaciju pomoću koje nalazimo najvažnije granice unutar slike. U morfološkim algoritmima i računarskom vidu, granice koje se dobiju se spajaju, da bi se slika mogla podijeliti u regije koje nam mogu dati više informacija o okolini (na primjer, ako je potrebno da sistem izbroji objekte koji su ispred njega). Prepoznavanje granica predstavlja težak problem, te je za slike koje nisu nastale pod tačno definisanim i kontrolisanim uslovima preporučljivo njihovo obrađivanje određenim filterima prije primjene algoritma prepoznavanja granica. Da bi se odredilo koje granice ostaju u konačnoj slici, a koje ne, koristi se već opisana metoda thresholding-a. U morfološkim algoritmima, prepoznavanje granica se obično primijeni na binarne slike, obzirom da je sa njima lakše raditi. Nakon što se nađu granice u binarnoj slici, zatvorene konture se popune tzv. *floodfill* postupkom, te kao rezultat dobijamo regije od interesa.

## 2.4 Dilacija i erozija

Kao što je već rečeno, svaka morfološka operacija se zasniva na dvije slike: ulazni set podataka, iz kojih je potrebno izvući relevantne informacije, te predefnisani set podataka, to jeste strukturni element (u primjenama, posebno ako je OpenCV u pitanju, često će se koristiti i pojam *kernel*). Kao što je ranije rečeno, svaki strukturni element ima određene dimenzije i oblik, koji zapravo opisuju na koji način će taj strukturni element djelovati sa ulaznom slikom (u tom smislu se na strukturni element može gledati kao na filter). Ulazna slika se dakle modificira tako što se na nju primjenjuju strukturni elementi različitih oblika i veličina. Elementarne operacije u morfologiji su dilacija i erozija, koje se dalje kombinuju u složenije operacije. U daljem razmatranju ćemo se fokusirati isključivo na binarne slike, pa govorimo o binarnoj dilaciji i eroziji (postoje i grayscale varijante ovih operacija).

Dilacija kombinuje dva skupa podataka na način da im sabere odgovarajuće elemente. Neka su data, na primjer, dva skupa uređenih n-torki A i B (gdje n-torke predstavljaju koordinate). Dilacija skupa A za skup B jeste skup svih mogućih vektorskih suma parova elemenata, pri čemu jedan element dolazi iz A, a drugi iz B. Operacija dilacije se obično označava simbolom  $\oplus_b$ , gdje je b element skupa na osnovu kojeg se vrši dilacija. Kao što se može naslutiti iz same definicije, prolazak strukturnog elementa B kroz zadanu sliku A uz operator dilacije kao rezultat ima da se regije slike sa bijelim pikselima proširuju (od toga i naziv dilacija, eng. *dilate* - proširiti). Ispod je prikazana matematska interpretacija dilacije, te zatim efekti dilacije na konkretnu (binarnu) sliku:

$$\begin{aligned}
 A &= \{(0, 1), (1, 1), (2, 1), (3, 1)\} \\
 B &= \{(0, 0), (0, 2)\} \\
 A \oplus_b B &= \{(0, 1), (1, 1), (2, 1), (3, 1), (0, 3), (1, 3), (2, 3), (3, 3)\}
 \end{aligned}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \quad A \oplus_b B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Operacija dilacije - matrični prikaz



Slika 3: Primjer dilacije. Dilacija je izvršena dva puta, a korišteni kernel je pravougaonik dimenzija 3x3

U morfologiji, dilacija je korisna kada želimo pojačati određene karakteristike binarne slike (npr. proširivanje regija od interesa). Također, operacija dilacije se može koristiti i za prepoznavanje granica slike, i to na način da se izvrši dilacija za neku malu vrijednost (odnosno male dimenzije strukturnog elementa), te se od dobivene slike oduzme polazna slika. Rezultat će tada biti oni pikseli koji su nastali dilacijom, što su efektivno samo granice.

Erozija predstavlja suprotnu operaciju od dilacije. Neka su A i B dva skupa uređenih n-torki, sa elementima a i b respektivno. Erozijska ova dva skupa  $A \ominus_b B$  jeste skup za čiji svaki element  $x$  važi relacija  $x + b \in A$ , za svako b iz B. Kao što dilacija predstavlja proširenje bijelih regija slike, tako erozija te regije smanjuje. Slijedi matični prikaz erozije, kao i efekt koji erozija ima na konkretnu sliku:

$$\begin{aligned}
 A &= \{(0, 1), (1, 1), (2, 1), (2, 2), (3, 0)\} \\
 B &= \{(0, 0), (0, 1)\} \\
 A \ominus_b B &= \{(2, 1)\}
 \end{aligned}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \quad A \ominus_b B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Operacija erozije - matični prikaz



Slika 4: Primjer erozije. Erozijska je izvršena dva puta, a korišteni kernel je pravougaonik dimenzija 3x3

Erozija je korisna operacija ukoliko želimo da uklonimo manje regije koje nam nisu bitne i koje mogu smetati algoritmu. Tada koristimo strukturni element takav da su mu dimenzije veće od dimenzija onih regija koje želimo da uklonimo.

Kombinirajući operacije erozije i dilacije, možemo dobiti složenije morfološke operacije, od kojih su neke najpoznatije: morfološko otvorenje (eng. *opening*) i zatvorenje (eng. *closing*), hit-or-miss transformacija, kao i top hat i bottom hat transformacije.

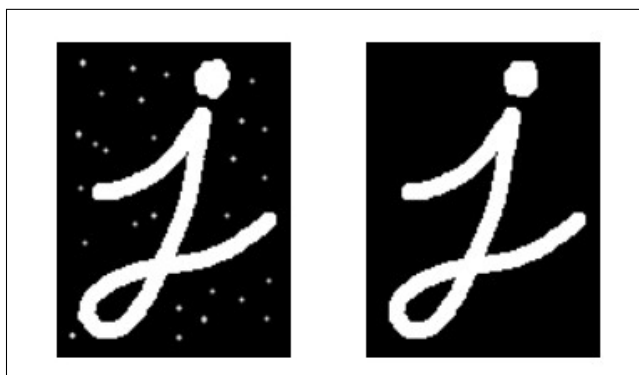
### 3 Složene morfološke operacije

#### 3.1 Morfološko otvorenje i zatvorenje

Morfološko otvorenje skupa A od skupa B (označava se kao  $A \circ B$ ) predstavlja složenu morfološku operaciju takvu da važi:

$$A \circ B = (A \ominus B) \oplus B$$

Dakle, otvorenje predstavlja primjenu operacije erozije sa kernelom B, pa zatim primjenu operacije dilacije, ponovo sa istim kernelom. Ova operacija je izuzetno korisna ako želimo da uklonimo šumove iz slike, a da pri tome ostatak ostane nepromijenjen. Otvorenje se može tumačiti na sljedeći način: sve regije koje su manje od strukturnog elementa će biti uklonjene u potpunosti, oštri uglovi će se izgladiti, tanke izbočine će biti uklonjene. Otvorenje se također može tumačiti i kao low-pass filter, u slučaju da se kao strukturni element koristi disk, dok će high-pass filter biti razlika same početne slike i otvorenja. Ispod je prikazano korištenje morfolškog otvorenja pri eliminaciji šuma sa slike.

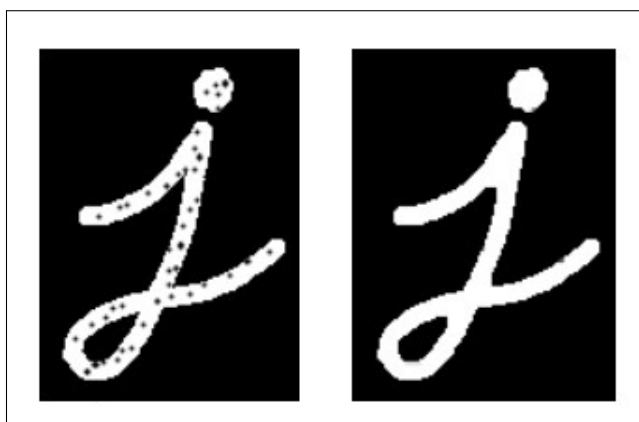


Slika 5: Primjer otvorenja - šumovi oko slova su uklonjeni

Morfološko zatvorenje ( $A \bullet B$ ) predstavlja složenu morfološku operaciju gdje važi:

$$A \bullet B = (A \oplus B) \ominus B$$

Za razliku od otvorenja, zatvorenje je složena operacija gdje se u prvom koraku primijeni operacija dilacije, da bi se nakon toga primijenila operacija erozije. Zatvorenje koristimo kada želimo da popunimo manje rupe unutar slike, ili da zatvorimo uske prolaze između dvije regije. Slijedi primjer morfolškog zatvorenja.



Slika 6: Primjer zatvorenja - rupe unutar slova su popunjene. Ovo je posebno korisno za OCR algoritme koji prepoznaju rukopis



Jasno je da su se slični efekti mogli postići i samom primjenom dilacije i erozije. Postavlja se pitanje, zašto onda koristiti otvorenje i zatvorenje? Iako je tačno da se osnovnim morfološkim operacijama može postići željeni efekt, one imaju jedan veliki nedostatak - kako dilacija tako i erozija mijenjaju originalnu sliku. To znači da bi se čistom primjenom npr. dilacije za uklanjanje rupa sa slova (kao npr. u slici 7), ne samo uklonile rupe, već bi se povećalo (proširilo) i samo slovo. Isto tako, ako koristimo eroziju za uklanjanje šumova, slovo bi se smanjilo (istanjilo). Ovakvi 'bočni efekti' operacija erozije i dilacije često nisu poželjni, te je u svakom slučaju bolje upotrijebiti operacije otvorenja i zatvorenja.

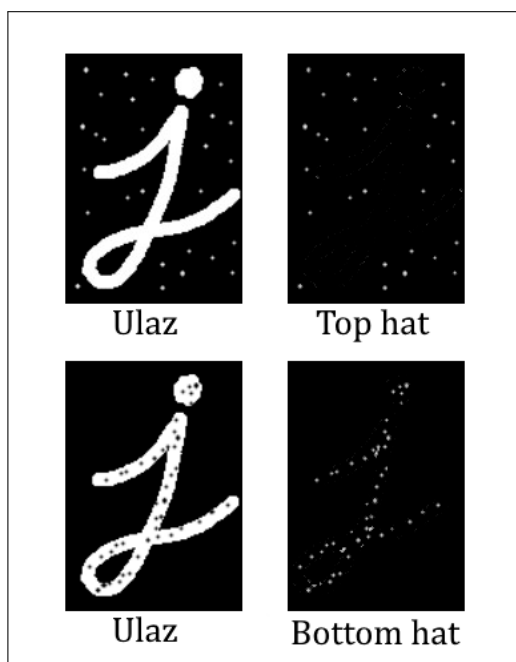
### 3.2 Top hat i bottom hat transformacije

Top i bottom hat transformacije kombinuju operacije morfološkog otvorenja i zatvorenja sa oduzimanjem slika. U literaturi se često može naći i termin black hat umjesto bottom hat.

Top hat transformacija predstavlja razliku između ulazne slike i njenog otvorenja, dok je bottom hat transformacija razlika između zatvorenja ulazne slike i same slike. To se može zapisati na sljedeći način:

$$\begin{aligned} \text{Top} - \text{Hat}(f) &= f - (f \circ B) \\ \text{Bottom} - \text{Hat}(f) &= (f \bullet B) - f \end{aligned}$$

Top hat transformacija u osnovi iz slike izdvaja manje regije i detalje, te ostatak slike odbacuje. Top hat transformacija se često koristi korekciju osjenčenosti (eng. *shading correction*). Nepoželjna osjenčenost može nastati neuniformnom raspodjelom distribucijom osvjjetljenja na slici. Top hat transformacija može da kompenzuje nedostatak osvjjetljenja. Bottom hat transformacija ima istu ulogu kao top hat, uz jednu značajnu razliku. Dok top hat transformacija izdvaja one detalje iz slike koji spadaju u prvi plan (to jeste, bijeli pikseli na binarnoj slici), bottom hat transformacija izdvaja detalje koji spadaju u pozadinu (odnosno crne piksele u binarnoj slici). Laički rečeno, top hat iz slike izdvaja male skupine bijelih piksela, dok ostatak odbacuje, dok bottom hat izdvaja male skupine crnih piksela, a ostatak odbacuje (razumije se da se te skupine crnih piksela zatim pretvore u bijele, jer nam je nakon bottom hat transformacije to faktički novi prednji plan slike). Ispod je dat primjer primjene top hat i bottom hat transformacije:



Slika 7: Top hat i bottom hat transformacije. Korišteni kernel je disk dimenzija 2x4

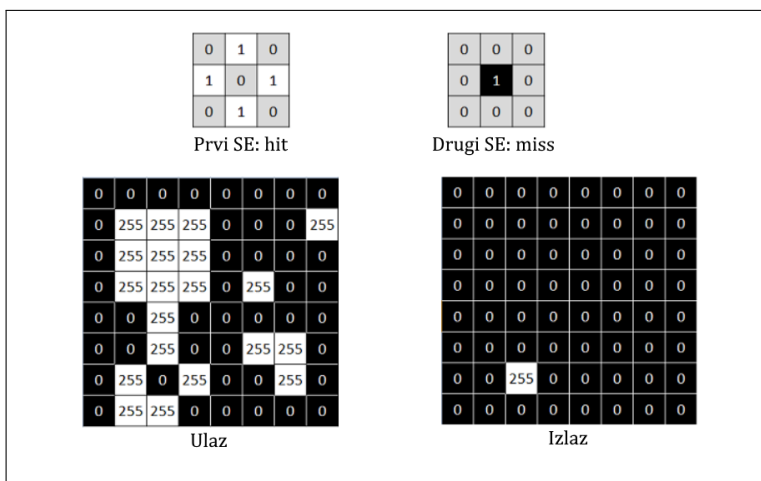
Treba napomenuti da se u određenoj literaturi top hat i bottom hat transformacija još nazivaju i white top hat i black top hat transformacija respektivno.

### 3.3 Hit-or-miss

Hit-or-miss transformacija (skraćeno HoM ili HMT) binarne slike  $A$  je definisana na sljedeći način: Neka je zadan strukturni element  $B$  koji je definisan kao par binarnih slika  $B_1$  i  $B_2$ , i to takav da važi  $B_1 \subseteq A, B_2 \subseteq \bar{A}$ . HMT slike  $A$  u odnosu na  $(B_1, B_2)$  se označava kao  $A \odot (B_1, B_2)$ , i računa se po formuli:

$$A \odot (B_1, B_2) = (A \ominus B_1) \cap (\bar{A} \ominus B_2)$$

Iz formule se može zaključiti sama priroda hit-or-miss transformacije: ova transformacija će da ostavi sve piksele koji se poklapaju sa prvim strukturnim elementom  $B_1$ , ali će također da ostavi sve one piksele koji se *ne* poklapaju sa drugim strukturnim elementom  $B_2$ , od čega dolazimo i do naziva hit-or-miss: hit (pogodatak) za prvi strukturni element ili miss (promašaj) za drugi. Hit-or-miss transformacija se pokazala kao moćan alat za detekciju oblika unutar slike. Primjer hit-or-miss transformacije je dat ispod.



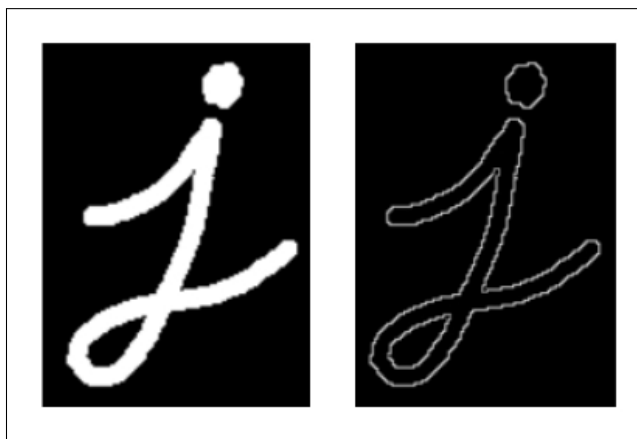
Slika 8: Primjer hit or miss transformacije. U ovom slučaju tražimo 3x3 strukture u kojima centralni piksel pripada pozadini, dok sjeverni, istočni, zapadni, i južni pikseli pripadaju prednjem planu. Ostali pikseli (u uglovima 3x3 strukturnog elementa) su nebitni, i mogu biti ili bijeli ili crni. Jedina 3x3 površina piksela koja ispunjava ove uslove svoj centar ima u predzadnjem redu i trećoj koloni.

## 4 Osnovni morfološki algoritmi

Morfološki algoritmi se najčešće predstavljaju preko ranije uvedenih morfoloških operatora, te ih je lako implementirati na računar. Najjednostavniji način implementacije svaki piksel predstavlja kao ćeliju sa različitim stanjima. Ako se sada definiše funkcija tranzicije za ćeliju u ovisnosti od ćelija koje graniče sa njom, primjenom te funkcije se aktualizira stanje svih ćelija (piksela) unutar slike. Ovakav način implementacije omogućava izvršavanje kompleksnih morfoloških algoritama u relativno kratkom vremenu. Neki od najosnovnijih algoritama u morfologiji su: izdvajanje granica, punjenje regija, i izdvajanje povezanih komponenti.

### 4.1 Izdvajanje granica

Kao što je već ranije rečeno, ulazna grayscale slika se može transformisati u binarnu pomoću thresholdinga. Nakon što je stvorena binarna slika, počinje izdvajanje granica slike (eng. *boundary extraction*). Granični pikseli će sada biti oni koji unutar 8 piksel koji graniče sa njima imaju barem jedan piksel koji je crn (odnosno, čija je vrijednost nula). Izdvajanje granica u svom prvom koraku zahtijeva eroziju slike sa nekim strukturnim elementom (najčešće matrica 3x3), te zatim računanje razlike između originalne slike i slike dobivene erozijom. U rezultirajućoj slici ostaju samo granični pikseli. Razumije se da debljina granice ovisi o dimenzijama strukturnog elementa. Tako se za 3x3 strukturni element dobija granica debljine 1, za 5x5 debljine 3, itd. Ispod je prikazano morfološko izdvajanje granica na primjeru.



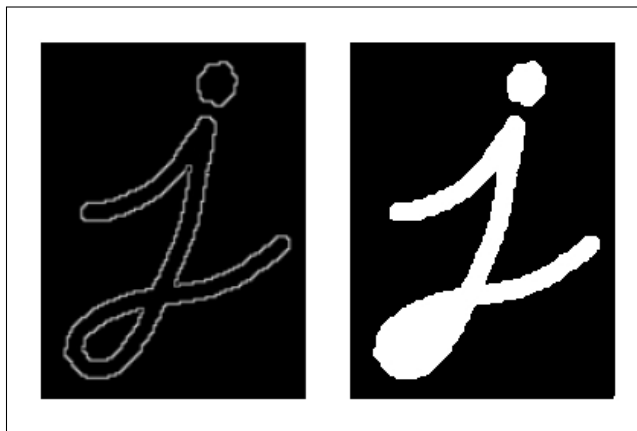
Slika 9: Primjer izdvajanja granica iz slike. Korišteni kernel je 3x3 kvadrat.

### 4.2 Punjenje regija

Osnovni cilj algoritma punjenja regija jeste da se unutar cijelog prostora nekog objekta na slici dijelovi koji su 0 (crni) pretvore u bijele piksele. Neka binarna slika kao granice regije bijele piksele, a sve piksele koji nisu dio granice regije crno. Algoritam punjenja regije počinje tako što se dodijeli vrijednost 1 nekom pikselu  $p$  koji nije granica (odnosno, taj piksel postaje bijel). Zatim se proširuje tako što se iterativno primjenjuje operacija dilacije, ali pod restrikcijom  $\bar{A}$ . Da ove restrikcije nema, punjenje ne bi bilo zaustavljeno na granicama regije, i algoritam ne bi terminirao sve dok svaki piksel slike ne bude imao vrijednost 1 (odnosno, dok ne bude bijel). Proces se može opisati sljedećom formulom:

$$X_k = (X_{k-1} \oplus B) \cap \bar{A}, k = 1, 2, 3, \dots$$

Gdje važi  $X_0 = p$  i  $B$  je strukturni element oblika plusa. Proces terminira kada važi  $X_k = X_{k-1}$ , to jeste kada se nije desila nikakva promjena na slici. Iz formule se lako vidi da se algoritam može implementirati i rekursivno. Primjer punjenja regija je dat ispod.



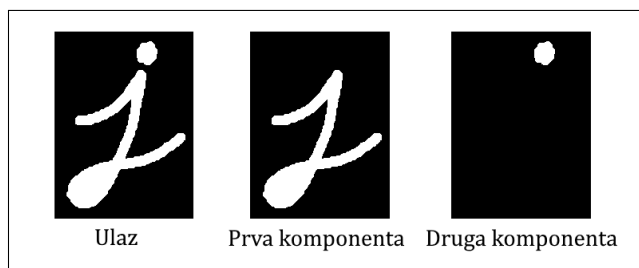
Slika 10: Punjenje regija unutar slike. Korišteni kernel je 3x3 kvadrat.

### 4.3 Izdvajanje povezanih komponenti

U 2D slikama, povezana komponenta predstavlja skupinu piksela iste vrijednosti, koji su međusobno povezani. Izdvajanje tih komponenti igra važnu ulogu u procesiranju slika, jer nam omogućava da svaki objekt unutar slike izdvojimo i sagledamo posebno. Iako zvuči slično, izdvajanje poveznih komponenti nije isto što i segmentacija slike. Dok se kod segmentacije slike sama slika modificirana na način da se iz nje izvuku neki novi podaci, izdvajanje povezanih komponenti ne mijenja ulaznu sliku, već samo označava komponente unutar nje, te kao izlaz daje set slika, za svaku komponentu po jednu, ili eventualno jednu sliku sa različitim labelama za svaku komponentu (npr. svaka komponenta označena u nekoj drugoj boji). Osnovna ideja je sljedeća: neka je ulazna binarna slika  $A$  takva da su crni pikseli označeni sa nula, a bijeli pikseli sa jedan. Sada možemo unutar neke komponente uzeti nasumičan piksel  $p$ , i proširivati ga iterativno koristeći dilaciju pod restrikcijom  $A$  (restrikcija je tu iz istih razloga kao i kod punjenja regija). Iterativni proces je opisan formulom:

$$X_k = (X_{k-1} \oplus B) \cap A, k = 1, 2, 3, \dots$$

Gdje je  $X_0 = p$  i  $B$  je strukturni element 3x3 kvadrat. Algoritam terminira kada  $X_k = X_{k-1}$ . Isti proces se sada nastavlja za sljedeću regiju u slici, te se kao konačni rezultat dobija niz slika sa svim povezanim komponentama. Primjer toga je dat na slici ispod.



Slika 11: Izdvajanje povezanih komponenti iz slike. Algoritam kao rezultat daje niz slika - u ovom slučaju su to dvije slike, za dvije povezane komponente. Korišteni kernel je 3x3 kvadrat.

## 5 Primjena i opis postupka za ANPR

### 5.1 Ukratko o računarskom vidu i ANPR

Računarski vid predstavlja, u najosnovnijem obliku, interdisciplinarnu granu nauke koja za cilj ima računarsko rješavanje problema orijentisanih oko sposobnosti ljudskog vida. Cilj računarskog vida je zapravo iz ulaznih podataka (to može biti jedna ili više slika) opisati ono što vidimo i rekonstruisati osobine, kao što su oblici prisutni na slikama, osvjetljenje, distribucija boja, dubina prostora, itd. Obzirom da ljudski vid nema nikakvih problema sa ovakvim zadatkom, postavka problema se na prvi pogled čini veoma jednostavnom. Međutim, složenost ovakvog zadatka je ogromna, te je i dan-danas sistem za računarski vid koji bi radio na nivou na kojem radi vid prosječnog dvogodišnjaka još uvijek nedostižan za današnju nauku.

Veoma mali dio radova iz oblasti računarskog vida se trenutno bavi samim *razumijevanjem* sadržaja ulaznih podataka; najčešći cilj je zapravo detekcija objekata unutar samih slika, njihov opis, procjena atributa kao što su veličina i udaljenost, njihova klasifikacija, te pokretanje određenih radnji na osnovu rezultata tih mjerenja. Iako se, kao što je već rečeno, radi o jako kompleksnoj problematici, značajan napredak danas se ostvario u nekoliko oblasti, i to između ostalog:

- Prepoznavanje karaktera (eng. *OCR - Optical charecter recognition*)
- Automatizirana inspekcija uređaja (u avioindustriji)
- Prodaja (npr. supermarketi bez zaposlenih, gdje se sva kupovina odvija automatski)
- 3D modeliranje - automatsko kreiranje 3D modela iz podataka dobivenim zračnim fotografijama
- Prepoznavanje otisaka prstiju
- Nadzor - prepoznavanje opasnosti

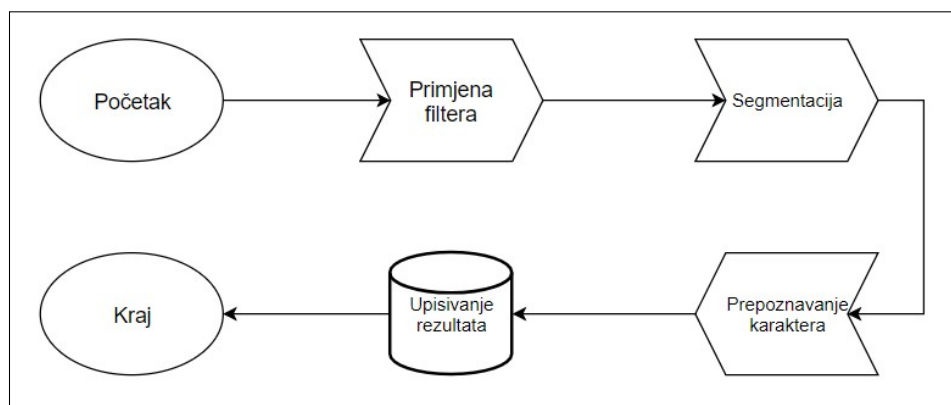
Postoje mnogi alati i programski jezici koji algoritme za računarski vid čine pristupačnima za programere. Najpoznatiji su MATLAB, te OpenCV biblioteka koja je dostupna za jezike Python i C++. Dok je MATLAB popularan u akademskoj zajednici, C++ je više orijentisan ka industriji, a Python postaje sve popularniji kako u primjeni tako i u istraživanju (posebno zato što sa sobom donosi veoma moćne biblioteke za računarstvo, kao što je numpy).

Jedna primjena računarskog vida koje ćemo se posebno dotaći u ovom radu jeste automatsko prepoznavanje registarskih tablica (ANPR). ANPR predstavlja tehniku videonadzora koja je zasnovana na prepoznavanju karaktera registarskih tablica vozila. Neke primjene ANPR danas uključuju:

- Provjera da li je vozilo registrovano na osnovu tablica
- Automatska naplata (npr. na autoputevima i u parking garažama)
- Potraga za osobama od strane policije
- Dokumentovanje kretanja saobraćaja

Dobar ANPR softver je u stanju da obradi jedno vozilo svake sekunde pri brzini kretanja vozila od 160 km/h. Za dobijanje fotografija se koristi infracrveno svjetlo i blic, da bi se bez obzira na vremenske uslove dobila što jasnija slika vozila. Sistem se također može uvezati sa postojećom bazom podataka u cilju identifikacije vozača i ostalih osobina koji su u vezi sa prepoznatim registarskim tablicama.

Sam proces se može podijeliti u nekoliko faza. Prva faza podrazumijeva dohvaćanje same slike. To se može desiti na licu mjesta (ako se vrši real-time obrada), ili centralni računar može dohvatiti sliku sa udaljene kamere, i pohraniti istu. Druga faza jeste primjena određenih filtera i morfoloških operatora na sliku, u cilju isticanja regija od interesa. Od velike je važnosti da su parametri algoritma pažljivo odabrani, jer je ova faza ključna za uspjeh algoritma. Nakon toga, u trećoj fazi se na dobivenoj slici traži regija od interesa, to jeste registarske tablice, koje se onda izrežu od ostatka slike, te se za ostatak algoritma posmatra samo izrezana regija. Konačno, u četvrtoj fazi se unutar tablica prepoznaju karakteri, te se oni zapisuju u određenu datoteku ili bazu podataka. Treba napomenuti da određeni napredni ANPR sistemi pored tablica čuvaju i fotografiju vozača, u slučaju da je istog potrebno identificirati. Sljedeća slika prikazuje gore opisani proces:



Slika 12: Osnovni dijagram toka ANPR sistema

Pri prepoznavanju registarskih tablica mogu se javiti mnogi problemi. Neki od njih su:

- Loš kvalitet ulaznih slika
- Slabo osvijetljena okolina kamere koja pravi snimke
- Nemogućnost identificiranja karaktera usljed prljavih tablica
- Manipulisanje registarskim tablicama

Za kraj treba napomenuti da je ANPR sistem često kritikovan. Mnogi aktivisti smatraju da su ovakvi sistemi direktno narušavanje privatnosti i metod za masovni nadzor stanovništva.

U narednom poglavlju ćemo se upoznati sa nekim osnovnim pojmovima u matematičkoj morfologiji, koji su od velike koristi za ANPR sisteme.

## 5.2 Generalni opis algoritma i korištenih heuristika

U ovom dijelu ćemo dati opis i prezentirati sve korake algoritma za prepoznavanje registarskih tablica koji smo implementirali na osnovu već uvedenih pojmova. Korišteni alat je MATLAB, kao i njego image processing toolbox. Sam algoritam se može podijeliti u dvije glavne faze:

- Prepoznavanje i ekstrakcija registarskih tablica sa slike (to jeste dijela slike koji sadrži tablice)
- Prepoznavanje odgovarajućih karaktera iz izdvojenog dijela slike, te njihovo upisivanje u .txt datoteku

Također treba napomenuti da, pošto se isključivo fokusiramo na BiH automobilske tablice, možemo da koristimo nekoliko činjenica kao dodatne heuristike u algoritmu:

- Sve tablice su u formatu *SBB-S-BBB* gdje je *S* slovo, a *B* broj
- Slova koja se javljaju u tablicama su isključivo ona koja se pišu isto kako na latinici tako i na ćirilici. Ta slova su: *A, E, J, K, M, O*, i *T*
- Registarske tablice u BiH koriste font *FE-Schrift*, koji pravi jasnu razliku između slova 'O' i broja nula (0)

U programu ćemo sve ove činjenice uzeti u obzir. Slijedi opis svake faze programa.

## 5.3 Ekstrakcija tablica

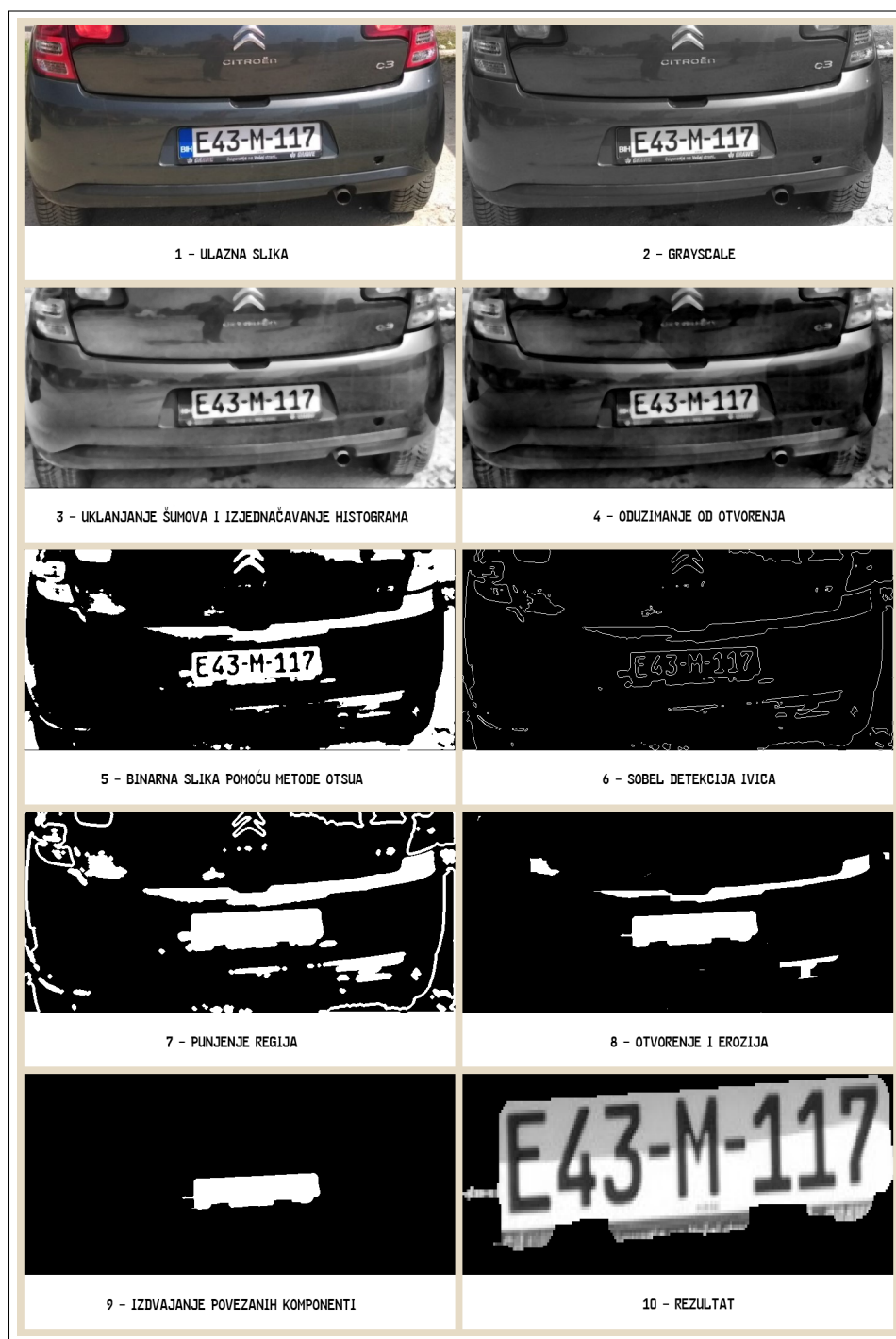
Prvu cjelinu algoritma predstavlja ekstrakcija tablica. Ovaj dio podrazumijeva prihvatanje ulazne slike, primjenu određenih filtera i morfoloških algoritama za pre-processing, te na koncu izdvajanje regije registarske tablice iz slike. Dakle, za ulaznu sliku mi kao rezultat dobijemo isječen dio slike sa tablicama, koji je spreman za dalju obradu i prepoznavanje karaktera. Slijedi opis svakog koraka programa, kao i konkretan primjer:

1. Prvi korak predstavlja dobavljanje slike. Sve slike koje smo koristili su napravljene pomoću mobilne HTC kamere (13 MP), i sačuvane u .jpg formatu. Nakon što se slike spase na predviđenoj lokaciji, program ih učitava pomoću naredbe *imread*
2. Učitana slika (koja je RGB) se sada pretvara u grayscale. U svrhu toga se koristi komanda *rgb2gray*, koja je implementirana pomoću već opisane formule za dobijanje nivo sive boje. Osnovna uloga ovog koraka jeste da smanji broj boja koje se koriste (sa 3 kanala na jedan 8-bitni gray-channel). Sada se može lakše manipulirati slikom, što je ključno za uspjeh algoritma. Upravo zato je ovo prva operacija koju vršimo nad slikom.
3. Naredni korak predstavlja potrebne operacije za uklanjanje šumova sa slike. Tako dobijamo bolju binarnu sliku u ostatku algoritma (to jeste, dobijamo manje 'sitnih' regija unutar slike, koje nam nisu od važnosti a mogu prouzrokovati netačne rezultate). Šumove uklanjamo pomoću tzv. *median filtera*, koji zapravo predstavlja jednu vrstu moving average point filtera. Pomoću komande *medfilt2* učitavamo sliku i matricu dimenzija 6x6. Svaki piksel će sada dobiti srednju vrijednost njegovog 6x6 okruženja. Kao drugi korak u uklanjanju šumova se vrši popravljavanje kontrasta slike. Kontrast se prepravlja na lokalnom nivou (dakle, posebno za svaki dio slike). Takav proces se naziva CLAHE (eng. *Contrast-limited Adaptive Histogram Equalization*). Konkretno, korištena je funkcija *adapthisteq*, koja realizira ovu tehniku. Nakon što se na sliku primijeni CLAHE postupak, kontrast u svim regijama slike je izbalansiran, te su šumovi svedeni na minimum. Treba napomenuti da je samo u teoriji moguće ukloniti sve šumove. U praksi je, uslijed velike raznolikosti ulaza, gotovo neizvodivo dobiti optimalne rezultate
4. Sada se na dobijenoj slici vrši morfološko otvorenje realizovano pomoću funkcije *imopen*, koristeći strukturni element oblika disk (dobiven ili ručnim kreiranjem matrice, ili pozivanjem funkcije *strel*), da bi se onda od slike oduzelo njeno otvorenje. Ovim dobijamo tamniju sliku, na kojoj je izraženija regija registarskih tablica

5. Nakon što su primijenjeni ovi filteri, možemo sliku konvertovati u binarnu. Obzirom da su ulazi nepredvidive prirode, ne možemo zadati fiksni prag za pretvaranje u binarnu sliku, već za svaku primljenu sliku moramo da računamo optimalni prag za binarizaciju. Da bi to izveli, koristimo metodu Otsu, te pomoću funkcije *graythresh* dobijamo željeni prag. Sada se slika vrlo lako pretvara u binarnu, i to na način da se koristi *imbinarize* funkcija. Alternativno se mogla napisati i petlja koja postavlja svaki piksel na odgovarajuću vrijednost u ovisnosti od prethodno izračunatog praga
6. Naredni korak podrazumijeva detekciju ivica na dobijenoj binarnoj slici. Koristimo Sobel filter, koji radi pomoću 3x3 strukturnog elementa, i detektuje ivice na onim mjestima na kojima je gradijent maksimalan. To se postiže pomoću funkcije *edge*. Razlog zbog kojeg tražimo ivice je da bi se riješili određenih manjih regija unutar binarne slike, kao i crnih regija koja su ostala od slova na prostoru registarske tablice. Također, uz jasne ivice algoritam punjenja regija, koji predstavlja sljedeći korak, radi bolje
7. Na sliku se primjenjuje dilacija uz mali kernel (da se podebljaju granice dobivene sobel operatorom), pa se zatim izvrši algoritam punjenja regija pomoću funkcije *imfill*. Sada imamo jasno definisane regije na slici, predstavljene bijelim pikselima. Posebno treba naglasiti razliku između obične binarne slike, dobijene u koraku 5. Sve rupe (odnosno, svi crni pikseli koji su okruženi bijelim pikselima) su sada popunjeni
8. U ovom koraku se prvo primjenjuje morfološko otvorenje (opet u cilju uklanjanja preostalih manjih regija), te se zatim vrši erozija, da bi se (dijelom) poništio efekt dobiven dilacijom iz prethodnog koraka. Ove operacije se primjenjuju u cilju eliminisanja što više regija, da bi se dobila veća vjerovatnoća nalaska regije koja predstavlja registarske tablice
9. Na dobijenoj slici primjenjujemo algoritam izdvajanja povezanih komponenti, pomoću funkcije *bwconncomp*. Tako dobijamo niz slika sa posebnim komponentama. Cilj je sada, pomoću tog seta slika, na ulaznoj slici ostaviti samo onu komponentu koja je površinom najveća. Upravo ta komponenta predstavlja našu registarsku tablicu. To radimo na sljedeći način: prvo ćemo na atribut 'PixelIdxList' od objekta dobivenog funkcijom *bwconncomp* primijeniti *cellfun*. Atribut PixelIdxList predstavlja listu čiji je svaki element lista piksela koji su postavljeni na 1 (odnosno, koji su bijeli na slici). Kao drugi atribut funkciji *cellfun* šaljemo funkciju *numel*, koja vraća broj elemenata nekog niza. Funkcija *cellfun* će dakle na svaki element list PixelIdxList primijeniti funkciju za broj elemenata, te vratiti listu kao rezultat. Time efektivno dobijamo listu u kojoj k-ti element predstavlja broj piksela koji čine k-tu komponentu naše slike. Sada pomoću funkcije *max* možemo naći indeks maksimalnog elementa te liste, te tako znati koja komponenta je najveća. Kreiramo novu sliku istih dimenzija kao stara, ali ćemo joj svaki piksel postaviti na 0 (dakle, cijela slika će biti crna). To postizemo funkcijom *zeros*. Konačno ćemo na toj slici promijeniti na 1 one piksele koji se nalaze unutar liste na prethodno pronađenom indeksu liste PixelIdxList. Tako dobijamo novu sliku u kojoj je isključivo prikazana najveća povezana komponenta, što predstavlja registarsku tablicu
10. U posljednjem koraku jednostavno nalazimo prvi i posljednji red, kao i prvu i posljednju kolonu u kojoj se javlja bijeli piksel na slici, te niz piksela izrežemo na način da ostavimo samo prostor koji je između tih redova (odnosno kolona). Na taj način dobijamo isključivo sliku registarskih tablica. Razumije se da ovo rezanje primijenimo na početnu sliku (dakle onu iz koraka 1), i to na način da prvo sve piksele na originalnoj slici koji nisu unutar dobijene regije pretvorimo u crne piksele

Time je dio algoritma koji se bavi identifikacijom i ekstrakcijom regije registarskih tablica okončan. Na slici ispod je dat i pregled svih prethodno opisanih koraka na konkretnom primjeru:





Slika 13: Prikaz svih navedenih koraka za ekstrakciju regije registarskih tablica sa ulazne slike

## 5.4 Prepoznavanje karaktera

Naredna cjelina algoritma jeste prepoznavanje samih karaktera na registarskoj tablici. Na ulazu imamo sliku dobivenu u prethodnom koraku, a kao izlaz dobijamo zapis pročitanih karaktera u *.txt* datoteku. Slijede osnovni koraci:

1. Prvo se učitava slika koja se dobila iz prethodnog dijela algoritma. Ta slika je već u grayscale formatu, i spremna je na obradu
2. Sada se na sliku primjenjuje thresholding, sa pragom od 125. To znači da će svaki piksel na slici koji ima vrijednost manju od 125 postati bijel, dok će ostali pikseli postati crni. Ovim efektivno izdvajamo objekte koji su u prvom planu, što kod nas predstavlja same karaktere na tablici
3. Pomoću funkcije *imclearborder* se čisti slika na način da se potisnu svijetli dijelovi oko granica slike, te na taj način dobijamo još izraženije karaktere
4. U ovom koraku iz slike izdvajamo 9 najvećih objekta iz slike. To radimo pomoću funkcije *bwareafilt*. Razlog zbog kojeg izdvajamo 9 objekta jeste to što svaka registracija ima tačno 9 karaktera (kada se broje i '-' karakteri). Nakon toga, dobijamo sliku na kojoj se nalazi samo 9 najvećih objekta
5. Slijedi najbitniji korak ovog dijela algoritma. Pomoću funkcije *ocr* iz Computer vision toolbox-a se vrši prepoznavanje pojedinačnih karaktera. Da bi se postili optimalni rezultati, nije korištena podrazumijevana konfiguracija parametara za funkciju *ocr*, već smo pomoću *OCR Trainer* alata i tehnika templatinga istrenirali OCR data file koji prosljedimo kao parametar algoritmu. Taj fajl je dobijen treniranjem seta podataka baziranih na fontu FE-Schrift koji koriste registarske tablice, i to isključivo sa onim karakterima koji se javljaju na BiH tablicama. Nakon što se obavi funkcija, kao izlaz dobijamo tekst na osnovu prepoznatih karaktera
6. Na kraju se konačni rezultat upisuje u datoteku *rezultati.txt*

Ispod je dat primjer fajla rezultati, nakon pokretanja dva testna ulaza:

```
1  REZULTATI:  
2  =====  
3  test1.jpg -> O91-K-036  
4  test2.jpg -> E43-M-117
```

Slika 14: Konačni rezultat algoritma

## 6 Zaključak

Za kraj možemo reći da je korištenje tehnika morfološke obrade slika od velike važnosti za ANPR sisteme, pa i za računarski vid kao cjelinu. Obzirom da je ovo problem koji je česta pojava u stvarnim situacijama, on je veoma značajan, i svaki napredak u razvoju ANPR sistema predstavlja veliki korak prema naprijed.

Konkretno u ovom radu je predstavljena efikasna metoda za ANPR, te možemo zaključiti da je MATLAB dobar alat za ovakve vrste obrade slika. Implementirani algoritam je baziran isključivo na strukturi BiH registarskih tablica automobila (standardni oblik pravougaonika dimenzija 520 × 110 mm), ali se on lako može proširiti i na druge vrste tablica. Što se ekstrakcije tablica tiče, moguće je napraviti poboljšanje dodavajući heuristiku za prepoznavanje regija koje podsjećaju na pravougaonik (umjesto trenutne metode koja samo uzima najveću regiju). Također, OCR algoritam ponekad daje pogrešne rezultate (konkretno može da pomiješa 'O' i '0' u nekim specijalnim primjerima). Uzrok toga je što je OCR data file treniran na nedovoljno velikom setu podataka. Proširenje training set-a bi donijelo i bolje rezultate.

Čitav kod se može naći u nastavku, kao i na sljedećem github linku:

<https://github.com/eteskeredzic/ANPR-MATLAB>

## 7 Kodovi

Slijedi kompletan listing kodova korištenih u radu:

Ekstrakcija tablica:

---

```
%% Izdvajanje regije registarskih tablica
clf
% korak 1 – učitaj sliku
path = input('Unesi naziv slike i format: ', 's');
img = imread(strcat('INPUT/',path));

% korak 2 – konvertuj u grayscale
img_gray = rgb2gray(img);

% korak 3 – CLAHE
img_median=medfilt2(img_gray,[6 6]);
img_hist = adapthisteq(img_median);

% korak 4 – oduzimanje od otvorenja
SE = strel('disk',35,4);
img_opening = imopen(img_hist,SE);
img_sub = img_hist - img_opening;

% korak 5 – binarna slika pomocu metode otsua
level = graythresh(img_sub);
img_bin = imbinarize(img_sub,level);

% korak 6 – sobel detekcija ivica
sobel = edge(img_bin, 'Sobel');

% korak 7 – punjenje regija
SE = strel('disk',2,4);
dilate = imdilate(sobel, SE);
fill = imfill(dilate, 'holes');

% korak 8 – otvorenje i erozija
SE = strel('disk', 6, 6);
img_opening2 = imopen(fill,SE);
img_opening2 = imerode(img_opening2,SE);

% korak 9 – izdvajanje povezanih komponenti
CC = bwconncomp(img_opening2);
numOfPixels = cellfun(@numel,CC.PixelIdxList);
[unused,indexOfMax] = max(numOfPixels);
biggest = zeros(size(img_opening2));
biggest(CC.PixelIdxList{indexOfMax}) = 1;

% korak 10 – crop + spasi rezultat
lnew = bsxfun(@times, img, cast(biggest, 'like', img));
[rows, columns] = find(biggest);
row1 = min(rows);
row2 = max(rows);
col1 = min(columns);
col2 = max(columns);
croppedImage = lnew(row1:row2, col1:col2);
figure, imshow(croppedImage);
imwrite(croppedImage, strcat('OUTPUT/',path));
```

---

Prepoznavanje karaktera:

---

```
% korak 1 – učitaj sliku
path = input('Unesi naziv slike i format: ', 's');
img = imread(strcat('OUTPUT/',path));

% korak 2 – izdvajanje backgrounda i foregrounda koristeći thresholding
% ako je intensity piksela ispod 125, bit će bijel, u suprotnom je crn
mask = img < 125;

% korak 3 – iscjenje svijetlih dijelova slike(suma) oko najvećeg dijela
% foregrounda(sto bi trebale biti tablice)
mask = imclearborder(mask);

% korak 4 – izdvajanje 9 najvećih područja na slici
mask = bwareafilt(mask, 9);
imshow(mask);

% korak 5 – korištenje OCR algoritma istreniranog za prepoznavanje BH
% registarskih tablica zajedno sa templateingom ugrađenim u OCR funkciju
ocrRes = ocr(mask, 'CharacterSet', 'AEJKMOT-0123456789', 'Language', 'RES\OCR_TRAINER\
    BH_Registarske\tessdata\BH_Registarske.traineddata');

% korak 6 – upisivanje rezultata u datoteku
S = [path, ' -> ', ocrRes.Text];
fid = fopen('rezultati.txt', 'a+');
fprintf(fid, '\r\n%s', S);
fclose(fid);
```

---

## Reference

- [1] D. Ballard, C. M. Brown, *Computer Vision*. Prentice Hall, 1982.
- [2] Oge Marques, *Practical image and video processing using MATLAB*. John Wiley & Sons, 2011.
- [3] Richard Szeliski, *Computer Vision: Algorithms and Application*. Springer, 2011.
- [4] Richard J. Goutsias, L. Vincent, D. S. Bloomberg, *Mathematical morphology and its applications to image and signal processing*. Kluwer academic, 2000.
- [5] Frank Y. Shih, *Image processing and mathematical morphology: Fundamentals and applications*. CRC Press, 2009.
- [6] W. Burger, J. Burge, *Principles of Digital Image Processing: Core Algorithms*. Springer, 2009.