UNIVERZITET CRNE GORE
ELEKTROTEHNIČKI FAKULTET

Damir Trnčić

# SEGMENTACIJA SLIKE POMOĆU K-MEANS METODE SA OPTIMALNIM CENTRIRANJEM KLASTERA PRIMJENOM PSO-SA-LEVY ALGORITMA

-MASTER RAD-

Podgorica, 2024. godine

PODACI I INFORMACIJE O MAGISTRANDU

Ime i prezime: **Damir Trnčič**
Datum i mjesto rođenja: **02. maj 1998. godine, Prijepolje**
Naziv završenog osnovnog studijskog programa i godina završetka studija: **Elektronika, telekomunikacije i računari, 2021. godine**

INFORMACIJE O MASTER RADU

Studijski program: **Računari**
Naslov rada: **Segmentacija slike pomoću K-means metode sa optimalnim centriranjem klastera primjenom PSO-SA-Levy algoritma**
Fakultet/akademija na kojoj je rad odbranjen: **Elektrotehnički fakultet**

UDK, OCJENA I ODBRANA MASTER RADA

Datum prijave master rada: **18. septembar 2023. godine**
Datum sjednice vijeća na kojoj je prihvaćena tema: **16. novembar 2023. godine**
Mentor: **Doc. dr Miloš Brajović**
Komisija za ocjenu/odbranu rada:

1. **Prof. dr Ljubiša Stanković,** predsjednik
2. **Doc. dr Miloš Brajović,** mentor
3. **Prof. dr Miloš Daković,** član

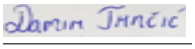
Datum odbrane: 27. decembar 2024. godine

Ime i prezime autora: Damir Trnčić

# Etička izjava

U skladu sa članom 22 Zakona o akademskom integritetu i članom 18 pravila studiranja na master studijama, pod krivičnom i materijalnom odgovornošću, izjavljujem da je master rad pod naslovom:
**“Segmentacija slike pomoću K-means metode sa optimalnim centriranjem klastera primjenom PSO-SA-Levy algoritma ”**
moje originalno djelo.

Podnosilac izjave:



Damir Trnčić

Podgorica, 18. Novembar 2024. godine

*Neizmjernu zahvalnost dugujem dragom Bogu koji mi je podario snage i strpljenja da istrajem na ovom putu školovanja.*

*Veliku zahvalnost dugujem mentoru doc. dr Milošu Brajoviću na pruženoj pomoći i posvećenosti u izradi ovog master rada.*

*Na kraju bih se posebno i jako zahvalio mojim divnim roditeljima, ocu Samedu i majki Fatimi, koji su bili konstantna podrška tokom mojeg školovanja i koji su uvjek vjerovali u moj uspjeh.*

# Predgovor

"Slika govori više od hiljadu riječi". Ova poznata izreka gotovo uvijek važi. Međutim, ponekad je teško izvući i razumjeti te riječi iz slike. To je upravo jedan od glavnih izazova u oblasti kompjuterske vizije. Automatska analiza slika ima značajan doprinos u razvoju mnogih oblasti, uključujući medicinu, tehnologiju, poljoprivredu i mnoge druge. Jedan od najvažnijih alata u kompjuterskoj viziji je segmentacija slike.

Segmentacija slike je proces koji omogućava dobijanje korisnih informacija iz slike. Kroz analizu piksela, osvijetljenosti i boje, kao i detekciju ivica i teksture, segmentacija slike jasno izdvaja određene oblasti na slici. Na primjer, kada segmentiramo medicinske slike ili slike biljaka, cilj algoritama segmentacije je da precizno izdvoje skup piksela koji sadrže određene strukture, kao što su tumori ili oštećeni dijelovi biljke.

Iako ne postoji stroga klasifikacija metoda segmentacije, algoritmi se razlikuju u pristupu analizi slike i fokusu na željeni rezultat. Do danas je razvijeno mnogo metoda segmentacije, među kojima su najpoznatiji segmentacija pomoću praga (engl. *thresholding*), pomoću klasterizacije (engl. *clustering*), pomoću regiona (engl. *region*), segmentacija pomoću granica regiona (engl. *edge*), segmentacija pomoću neuralnih mreža i tako dalje. Svaki od ovih metoda pronašao je svoje mjesto u nekoj praktičnoj primjeni. Na primjer, metoda pomoću praga se koristi u odvajanju pisanih i štampanih tekstova, metoda klasterizacije u analizi satelitskih slika, metoda pomoću regiona kod slika sa malo detalja. Iako su ove metode već uspješno primjenjene, postoji stalna potreba za njihovim unapređenjem. Način na koji se vrši poboljšanje metoda pomoću klasterizaije jeste pronalaženje optimalnog broja klastera kao i precizniji centar za svaki klaster. U tu svrhu, često se koriste popularne kombinacije sa metaheurističkim algoritmima koji mogu dosta precizno pronaći centre klastera.

Dobijeni rezultati segmentacije predstavljaju ključnu tačku u procesu kompjuterske vizije. Ovi rezultati omogućavaju dublje razumijevanje slike i otkrivanje informacija koje su često nevidljive golim okom. Preciznost i pouzdanost rezultata segmentacije opisuju kvantitativne i kvalitativne mjere koje uzimaju u obzir jedinstvenost i uniformnost regiona dobijenih segmentacijom.

# Sažetak

Segmentacija slike obavlja se analizom osvijetljenosti i boje piksela, nakon čega se ti pikseli označavaju i pripajaju klasterima sa sličnim osobinama. Svaki piksel se pridružuje klasteru na osnovu kriterijuma minimalne udaljenosti. Prvo se izračunava euklidska distanca između piksela i svih centroida klastera. Piksel se zatim pridružuje klasteru čiji je centar najbliži, tj. onom klasteru sa najmanjom euklidskom distancom u odnosu na taj piksel. Kada se svi pikseli pridruže odgovarajućim klasterima, centar svakog klastera se ažurira na osnovu srednje vrijednosti svih piksela unutar tog klastera, što rezultira novim pozicijama centroida za sljedeću iteraciju algoritma. Poznati algoritam koji sprovodi ove zadatke u metodama klasterizacije naziva se K-means (algoritam *K* srednjih vrijednosti).

K-means je veoma brz i jednostavan algoritam, pogodan za obradu velikih skupova podataka. Međutim, neke od glavnih mana ovog algoritma uključuju osjetljivost rezultata na početne vrijednosti i ignorisanje piksela koji su izvan uobičajenih vrijednosti (engl. *outliers*). Da bi se prevazišli ovi nedostaci, algoritam K-means se često kombinuje sa metaheurističkim algoritmima koji efikasno rješavaju ove probleme.

Optimizacija rojem čestica (engl. *Particle Swarm Optimization*) je metaheuristički algoritam zasnovan na prirodnom fenomenu traženja izvora hrane od strane roja (jata ptica, roja pčela, itd.). Kada jedna jedinka pronađe najbolji izvor hrane, ostale jedinke kreću za njom. Iako ovaj algoritam ima izvrsne performanse konvergencije, može zapasti u lokalni optimum. Zbog toga se vrlo često ovaj algoritam unapređuje sa tzv. Levy letom i Algoritmom simuliranog kaljenja (engl. *Simulated Annealing*), koji uspješno izvlače algoritam iz lokalnog optimuma.

U ovom radu predstavljena je primjena K-means algoritma za segmentaciju slike, uz optimizaciju centara klastera korišćenjem hibridnog modela zasnovanog na Optimizaciji rojem čestica (PSO), Simuliranom kaljenju (SA) i Levy letom. Cilj istraživanja bio je poboljšati preciznost segmentacije kroz kombinaciju različitih metaheurističkih pristupa, čime se unapređuje proces pronalaženja optimalnih centara klastera. Kao rezultat, razvijen je PSO-SA-Levy hibrid koji omogućava efikasniju segmentaciju slike, dok smanjuje rizik od zapadanja u lokalne minimume i unapređuje kvalitet rješenja.

Eksperimentalni rezultati pokazali su da predloženi hibridni model daje bolje rezultate u smislu PSNR, SSIM i FSIM metrika u poređenju sa standardnim K-means algoritmom. Ovaj rad takođe pruža uvid u značaj primjene slučajnih šetnji, kao što je Levy let, u metaheurističkim algoritmima, kao i prednosti i mane hibridnih pristupa. Doprinos rada ogleda se u demonstraciji efikasnosti kombinovanja PSO, SA i Levy leta za segmentaciju slike, što predstavlja potencijalno novu oblast primjene ovih algoritama.

**Ključne riječi**: segmentacija slike, klasterizacija, K-means algoritam, PSO, SA, Levy let

# Abstract

Image segmentation is performed by analyzing the brightness and color of pixels, after which these pixels are labeled and assigned to clusters with similar properties. Each pixel is assigned to a cluster based on the criterion of minimum distance. First, the Euclidean distance between the pixel and all cluster centroids is calculated. The pixel is then assigned to the cluster whose center is the closest, i.e., the cluster with the smallest Euclidean distance relative to that pixel. Once all pixels are assigned to their corresponding clusters, the center of each cluster is updated based on the average value of all pixels within that cluster, resulting in new centroid positions for the next iteration of the algorithm.

The well-known algorithm that performs these tasks in clustering methods is called K-Means.

K-Means is a very fast and simple algorithm, suitable for processing large datasets. However, some of its major drawbacks include sensitivity to initial values and the ignoring of pixels that are considered outliers. To overcome these limitations, the K-Means algorithm is often combined with metaheuristic algorithms that effectively address these issues.

Particle Swarm Optimization (PSO) is a metaheuristic algorithm based on the natural phenomenon of swarming in search of food sources (e.g., a flock of birds, a swarm of bees, etc.). When one particle finds the best food source, the others follow it. Although this algorithm has excellent convergence performance, it can fall into a local optimum. Therefore, it is often enhanced with mechanisms such as the Levy flight and Simulated Annealing (SA), which successfully help the algorithm escape local optima.

This paper presents the application of the K-means algorithm for image segmentation, with cluster center optimization using a hybrid model based on Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Levy flights. The goal of this research was to improve segmentation accuracy by combining different metaheuristic approaches, thereby enhancing the process of finding optimal cluster centers. As a result, a PSO-SA-Levy hybrid was developed, enabling more efficient image segmentation, reducing the risk of getting trapped in local minima, and improving solution quality.

Experimental results showed that the proposed hybrid model delivers better outcomes in terms of PSNR, SSIM, and FSIM metrics compared to the standard K-Means algorithm. This paper also provides insight into the significance of using random walks, such as Levy flights, in metaheuristic algorithms, as well as the advantages and disadvantages of hybrid approaches. The contribution of this study is reflected in demonstrating the effectiveness of combining PSO, SA, and Levy flights for image segmentation, representing a potentially new area of application for these algorithms.

**Keywords**: Image segmentation, Clustering, K-Means algorithm, PSO, SA, Levy Flight

# Sadržaj

# Lista slika

# Lista tabela

# Poglavlje 1

# Uvod i pregled aktuelnih istraživanja u oblasti

U domenu kompjuterske vizije, segmentacija slike igra ključnu ulogu, omogućavajući precizno identifikovanje i razdvajanje objekata ili regiona od interesa unutar slike. Ovaj proces je primjenjen u različitim aplikacijama, uključujući medicinsku dijagnostiku, poljoprivredu, kontrolu kvaliteta u industriji i prepoznavanje oblika. Do danas postoji mnoštvo dostupnih tehnika segmentacije slike. Koju tehniku odabrati, zavisi od problema odnosno zadataka, ali je bitno da bude zadovoljena definicija uspješne segmentacije [1]: *regioni dobijeni postupkom segmentacije budu uniformni i homogeni u odnosu na neke karakteristike, kao što su nivo sivog (ton) ili tekstura. Unutrašnjost regiona treba da bude jednostavna i bez malih rupa, a granice regiona jednostavne, glatke i prostorno tačne. Susjedni regioni treba da budu dovoljno različiti po onom atributu po kome su uniformni*.

Metod segmentacije pomoću jednog praga je vrlo jednostavan metod i koristi osvijetljenost piksela koje poredi sa jednim pragom, da bi na kraju kao rezultat dobili binarnu sliku. Ovaj metod se često koristi kod izdavjanja i prepoznavanja tekstova sa slika [2]. Međutim, na složenijim slikama, sa više objekata, možemo koristiti segmentaciju sa više pragova, gdje se osvijetljenost piksela poredi sa više pragova.

Većina tehnika segmentacije, uključujući metod pomoću praga i metod klasterizacije zasnovani su na analizi pojedinačnih piksela. Kada se u procesu segmentacije analiziraju grupe susjednih piksela, onda se pikseli, slični po osvijetljenosti ili boji, grupišu i nastaje *region*. Ova tehnika je poznata kao metoda pomoću regiona, a dvije poznate tehnike su segmentacija pomoću rasta regiona i segmentacija pomoću spajanja regiona [1].

Pored detekcije objekata na slici, kompjuterska vizija ima zadatak da detektuje i analizira linije, uglove, krive sa slike ili granice objekata. Jednom riječju, zadatak je detektovati ivice sa slike. Skup tačaka (piksela) gdje se osvijetljenost slike naglo mijenja nazivaju se *ivice*. Detekcija ivica je vrlo složen zadatak. Metod segmentacije pomoću granica regiona se bavi ovim problemom i jedna je od šire istraživanih tehnika, zbog brojnih izazova i problema koji se nameću tokom pomenutog zadataka. Neki od njih su lažne ivice i šum [3].

U ovom radu će biti posvećena pažnja poznatoj metodi, metodi segmentacije pomoću klasterizacije, koji je stekao popularnost zbog svoje jednostavnosti i efikasnosti. Metod klasterizacije je tipičan metod

nenadgledanog učenja. Nenadgledano učenje proces u kojem algoritam koristi samo ulazne podatke bez ciljnih varijabli kako bi otkrio skrivene obrasce ili strukture u podacima. Metod klasterizacije particionira sliku u klastere, gdje svaki klaster predstavlja grupu sličnih piksela. Primarni cilj je minimizirati varijansu unutar klastera, dok se varijansa između njih maksimizuje. Ovaj iterativni proces obično počinje sa inicijalizacijom centara klastera na slučajne vrijednosti, nakon čega slijedi dodjela piksela najbližem centru klastera i ažuriranje centara na osnovu srednje vrijednosti piksela unutar svakog klastera. Ove iteracije se nastavljaju sve dok ne dođe do konvergencije ili se dostigne unaprijed definisani maksimalni broj iteracija. Dva najpoznatija agoritama klasterizacije koji se koriste u praksi su K-means i Fuzzy C-means.

Metod klasterizacije, kao i mnogi optimizacioni problemi, podložan je lokalnim optimumima zbog svoje zavisnosti od inicijalizacije. Kako bi se prevazišla ova ograničenja i poboljšao kvalitet ovog metoda segmentacije slike, kombinacija sa metaheurističkim algoritmima optimizacije je pokazala značajno popravljene performnase. U radu [4], popularni K-means algoritam je kombinovan sa dinamičkim metodom Optimizacije roja čestica (DPSO). Istraživanja su pokazala poboljšane vizuelne rezultate u poređenju sa tradicionalnim metodom klasterizacije u segmentaciji slika. K-means algoritam je takođe kombinovan sa Algortimom svica (FA) [5]. Ova modifikacija rezultirala je značajnim poboljšanjima, posebno u smislu postizanja niže prosječne greške segmentacije, većeg odnosa signal-šum i povećane sličnosti strukture kada je primijenjena na medicinske slike. Takođe je kombinovan i sa Algoritmom sivog vuka (engl. *Grey Wolf Optimizer* - GWO) i primjenjen nad satelitskim slikama. Dobijeni su superiorni rezultati i tačnost u pogledu metrika kao što su Davies-Bouldin indeks, prosječno rastojanje između klastera i prosječno unutar-klastersko rastojanje [6]. U radu [7] je korišten Algoritam optimizacije hranjenja manta raža (engl. *Manta Ray Foraging Optimization* ili *MRFO* algoritam), za poboljšanje segmentacije slika bazirane na K-means algoritmu. Različite metrike kvaliteta segmentacije, kao što su PSNR, SSIM i FSIM, potvrdile su izvrsne performanse ovog pristupa.

PSO agoritam je veoma brz, efikasan i jenostavan algoritam sa malo parametara koje treba podešavati. Iako PSO algoritam ima dobre performanse konvergencije, često se dešava da ne konvergira ga globalnom optimumu, već zapada u lokalni optimum. Kada PSO dostigne lokalno optimalno rješenje, sve čestice su pozicionirane oko njega, što predstavlja poteškoću nastavka pretrage ka globanom optimumu. Drugim riječima, kada PSO zapadne u ovakvu situaciju, teško je napraviti veći "korak" čestica ili promijneti im smjer kretanja variranjem postojećih parametara PSO algoritma, kako bi izašle iz lokalnog optimuma nastavile dalju pretragu. Da bi se rješio ovaj problem, u ovom radu PSO algoritam je proširen sa povremenim uključivanjem Algoritma simuliranog kaljenja (SA) kao i Levy leta, koji na osnovu Levy distribucije generiše nove pozicije čestica što im omogućava dosta raznovrsnu i detaljnu pretragu prostora rješenja [8, 9].

Struktura rada je sljedeća. U Poglavlju 2 će biti opisane postojeće tehnike segmentacije slike sa svojom matematičkom pozadinom kao i svojim dosadašnjim primjenama. Takođe će biti definisane metrike koje se koriste za evaluaciju performansi segmentirane slike. Zatim u Poglavlju 3 će biti obrazloženi korišćeni metaheuristički algortimi a zatimu u Poglavlju 4 opisane karakteristike nasumičnog kretanja, a zatim će detaljno biti objašnjen K-means algoritam i njegova kombinacija sa predloženim metaheurisičkim algoritmom. U Poglavlju 5 će se predstaviti dobijeni rezultati i poređenje sa ostalim

dosadašnjim poboljšanim K-means algoritmima. U zaključku će biti prokomentarisani rezultati i opisane prednosti i mane predloženog metoda.

# Poglavlje 2

# Metode segmentacije digitalnih slika

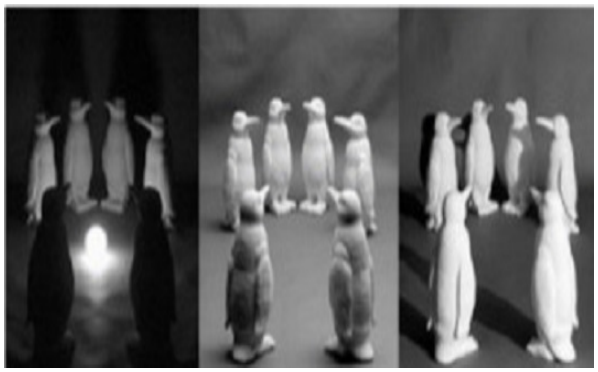
## 2.1 Inženjering slike i metodi segmentacije

Inženjering slike obuhvata sve metode i tehnike za obradu i analizu slika. Ovaj multidisciplinarni pristup ima tri glavna sloja, svaki sa svojim specifičnim zadacima [10]:

1. **Obrada slike (engl. *Image Processing*)**:
    - Ovaj sloj predstavlja osnovni nivo i fokusira se na manipulaciju pikselima slike.
    - Zadatak obrade slike je da primjeni različite transformacije, filtriranje i poboljšanja i slično, na slikama.
2. **Analiza slike (engl. *Image Analysis*)**:
    - Analiza slike ide korak dalje od obrade slike i ima za cilj razumijevanje sadržaja slike.
    - Ovdje se uglavnom uzimaju u razmatranje razna mjerenja koje se koriste u svrhu analize.
3. **Razumijevanje slike (engl. *Image Understanding*)**:
    - Ovo je najviši nivo inženjeringa slike i predstavlja najizazovniji zadatak.
    - Cilj je razumjeti kontekst slike, interpretirati složene scene i donositi zaključke na osnovu vizuelnih podataka.

Segmentacije slike potpada pod zadatak obrade slike. Sa svojim rezultatima koje nosi, kao na primjer, detektovati objekat ili abnormalnosti, izdvojene ivice, i slično, spada u jedan od najbitnijih koraka koji se preduzimaju da bi se sproveli zadaci analize i razumijevanja slike. Umjesto analize ogromnog broja piksela, segmentirana slika biva predstavljena pomoću nekoliko vrijednosti piksela, dodijeljenih specifičnim detektovanim objektima sa slike. Međutim, da bi se došlo do kvalitetno segmentirane slike je vrlo izazovno, jer na primjer, detektovati objekte na slikama sa istim sadržajem ali snimljenim pod različitim osvijetljenjem može predstavljati ozbiljan problem. Drugim riječima, varijacija osvijetljenja ima veliki uticaj jer samim tim isto pozicionirani pikseli dobijaju različite vrijednosti, kao što je prikazano

na slici 2.1(a). Pored pomenute varijacije, postoje varijacije unutar klasa. Ista klasa može imati različite atribute, kao recimo boju ili oblik kao što je prikazano na slici 2.1(b). Pomenute varijacije predstavljaju jedan od glavnih izazova na koje nailaze tehnike klasterizacije i metode pomoću praga [11].



(a) Varijacija osvjetljaja prilikom fotografisanja

(b) Varijacije unutar istih klasa

**Slika 2.1:** Ilustracija glavnih izazova za nenadgledani metod segmentacije slike

Ne postoji egzaktna klasifikacija metoda segmentacije. Odabir metoda koji će se koristiti u konkretnom slučaju zavisi od tipa problema, kao i od vrste slika čiju je segmentaciju potrebno izvršiti. Neke od vrlo često korišćenih metoda, i to nenadgledanih, jesu, kao što je već pomenuto, metoda pomoću jednog ili više pragova, metoda pomoću regiona, metoda pomoću ivica i metoda pomoću klasterizacije.

### 2.1.1 Segmentacija pomoću praga

Binarana slika je slika koja se sastoji od samo dvije vrijednosti piksela, uglavnom minimalne (0) i maksimalne (255) osvjetljenosti. Takva slika je obično vrlo jednostavna jer je predmet od značaja na slici, obično u fokusu, i odvojen od pozadine. Segmentiranjem slike pomoću jednog praga možemo dobiti prethodno opisanu binarnu sliku. Prag koji se koristi u ovim metodama segmentacije zapravo predstavlja vrijednost piksela iznad koje se pojavljuju pomenuti predmeti od značaja, dok se ispod tog praga ostali manje bitni predmeti sa slike, kao što je pozadina, zanemaruju. Drugim riječima, piksel na poziciji $(m,n)$ na slici $I(m,n)$ će biti označen kao bitan ako je $I(m,n) > T$, gdje je $T$ prag, inače, piksel je klasifikovan kao pozadina. Segmentirana slika $S(m,n)$ bi bila definisana na sljedeći način:

$$S(m,n) = \begin{cases} 1, & \text{ako je} \quad I(m,n) > T, \\ 0, & \text{ako je} \quad I(m,n) \leq T. \end{cases} \tag{2.1}$$

Vrijednost praga $T$ zavisi od slike. Jedan prosti način određivanja praga bi izgledao ovako [12]:

1. Inicijalizovati vrijednost praga $T$ tako da bude veći od minimalne i manji od maksimalne vrijednosti piksela slike.

2. Izvršiti klasifikaciju piksela pomoću jednačine (2.1). Dobijene su dvije grupe piksela, $G_1$ koji su veći od praga $T$ i $G_2$ koji su manji ili jednaki od praga $T$.

3. Izračunati srednje (engl. *mean*) vrijednosti, srednju vrijednsot za prvu grupu, $k_1$, i srednju vrijednost za drugu grupu, $k_2$.

4. Izračunati novi prag na sljedeći način: $T = \frac{1}{2}(k_1 + k_2)$

5. Ponavljati korake 2 - 4 dok razlika između nove i prethodne vrijednosti praga ($\Delta T$) ne bude manja od neke unaprijed definisane vrijednosti $V$.

Algoritam iterativno pronalazi optimalan prag. Znajući da optimalan prag zavisi od vrijednosti $\Delta T$, koja teži da dostigne vrijednost $V$, možemo zaključiti da brzina ovog algoritma zavisi od $V$. Takođe je upitno kako izabrati početni prag $T$. S tim u vezi, dosta je pogodnije da prag bude određen samo jednom, a prethodnom analizom slike, i to pomoću histograma.

Takođe još jedan od prostijih metoda za određivanje praga poznat je kao $p-tile$ [13]. Ovaj metod se primjenjuje na slikama gdje je objekat dosta izraženiji od pozadine i približno poznato koliko procenata $p$ je zauzeo na slici, na primjer, tekst na listu papira. Ako je objekat zauzeo 30% slike, prag se postavlja na najveću vrijednost sive koji će omogućiti da 30% piksela slike bude klasifikovano kao objekat.

Iako ovaj metod spada u jedan od najjedonstavnijih, određivanje pomenutog praga može predstavljati ozbiljan zadatak i izazov. Slika koju treba segmentirati može prikazivati više predmeta, koji pritom nisu istih boja, osvjetljaja piksela, a jedan isti predmet može imati više različitih osvjetljaja. Za jasnu predstavu o nivoima osvijetljenosti, odnosno koliki broj piksela slike posjeduje određenu vrijednost, koristi se histogram. Za ovaj metod segmentacije, histogram ima bitnu ulogu. Jedan način da se odredi prag jeste kod histograma koji imaju dva maksimima, tj. bimodalni su. Onda je prag segmentacije minimum (dolina) koji se nalazi između njih. Ovaj način se može primjeniti ako su vrhovi tj. maksimumi približne veličine a dolina pravilnog oblika i bez ravnina. Jasno je da ovaj metod ima vrlo ograničenu primjenu, jer je malo slika sa ovakvim oblikom histograma.

Postoje brojni pristupi i algoritmi koji pokušavaju prevazići prethodne probleme, međutim najpoznatiji metod koji uspješno pronalazi optimalan prag, i to samo na osnovu histograma, diskriminatnom analizom, jeste Otsu metod [14]. U nastavku slijedi opis navedenog metoda. Pretpostavimo da slika sadrži $L$ nivoa sive boje. Broj piksela sa nivoom $i$ je $n_i$, dok je ukupan broj piksela $N = n_1 + n_2 + ... + n_L$. Histogram će biti normalizovan i predstavljaće distribuciju vjerovatnoće:

$$p_i = \frac{n_i}{N}, \quad \text{gdje je} \quad p_i \geq 0, \quad \sum_{i=1}^{L} p_i = 1 \tag{2.2}$$

Pretpostavimo sada da piksele dijelimo u dvije klase, pozadinu i objekat, $C_0$ i $C_1$, pragom $k$. $C_0$ klasi pripadaju pikseli sa nivoima $[1,2,\ldots,k]$, a $C_1$ klasi pripadaju pikseli nivoa $[k+1,...,L]$. Vjerovatnoće pojavljivanja klasa su:

$$\omega_0(k) = Pr(C_0) = \sum_{i=1}^{k} p_i = \omega(k) \tag{2.3}$$

$$\omega_1(k) = Pr(C_1) = \sum_{i=k+1}^{L} p_i = 1 - \omega(k) \tag{2.4}$$

a srednje vrijednosti:

$$\mu_0(k) = \sum_{i=1}^{k} iPr(i|C_0) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} = \frac{\mu(k)}{\omega(k)} \tag{2.5}$$

$$\mu_1(k) = \sum_{i=k+1}^{L} iPr(i|C_1) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1-\omega(k)} \tag{2.6}$$

gdje

$$\omega(k) = \sum_{i=1}^{k} p_i, \tag{2.7}$$

i

$$\mu(k) = \sum_{i=1}^{k} ip_i \tag{2.8}$$

predstavljaju kumulativne momente histograma nultog i prvog reda, izračunati do $k$-tog nivoa, a

$$\mu_T = \mu(L) = \sum_{i=1}^{L} ip_i \tag{2.9}$$

predstavlja srednju vrijednost cijele slike. Može se prosto dokazati da za svako $k$ važe sljedeće jednakosti:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T, \qquad \omega_0 + \omega_1 = 1. \tag{2.10}$$

Varijanse klasa su:

$$\sigma_0^2 = \sum_{i=1}^{k} (i-\mu_0)^2 Pr(i|C_0) = \sum_{i=1}^{k} (i-\mu_0)^2 \frac{p_i}{\omega_0} \tag{2.11}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i-\mu_1)^2 Pr(i|C_1) = \sum_{i=k+1}^{L} (i-\mu_1)^2 \frac{p_i}{\omega_1} \tag{2.12}$$

Da bi smo izračunali pogodnost, odnosno tačnost praga $k$, potrebno je uvesti nove mjere klasnih separabilnosti:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \qquad \kappa = \frac{\sigma_T^2}{\sigma_W^2}, \qquad \eta = \frac{\sigma_B^2}{\sigma_T^2},$$

gdje

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2 \tag{2.13}$$

predstavlja unutarklasnu varijansu, zatim

$$\sigma_B{}^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{2.14}$$

međuklasnu varijansu, dok

$$\sigma_T^2 = \sum_{i=1}^{L} (i-\mu_T)^2 p_i \tag{2.15}$$

predstavlja totalnu varijansu osvijetljenosti.

Maksimiziranjem jednog od parametara iz (2.13) dobijamo optimalni prag, jer važi da je:

$$\sigma_W^2 + \sigma_B^2 = \sigma_T^2. \tag{2.16}$$

Zatim slijedi da također važe sljedeće jednakosti:

$$\kappa = \lambda + 1, \qquad \eta = \frac{\lambda}{\lambda + 1}. \tag{2.17}$$

Zbog toga su diskrimanantni kriterijumi za maksimizaciju pokazatelja $\lambda, \kappa$ i $\eta$, iz (2.13) međusobno ekvivalentni. Međutim, primijetićemo da je $\sigma_W^2$ zasnovan na statistici drugog reda, a $\sigma_B^2$ na statstici prvog reda, i također $\sigma_W^2$ i $\sigma_B^2$ su funkcije koje zavise od praga $k$, dok $\sigma_T^2$ ne zavisi. Iz toga slijedi da je $\eta$ najpogodnija mjera za računanje optimalnog praga. Maksimizacijom parametra $\eta$, a samim tim i maksimizacijom $\sigma_B^2$ dobija se optimalan prag $k_{opt}$, rješavanjem jednačine (2.18) linearnim pretraživanjem:

$$\sigma_B^2(k_{opt}) = \max_{1 \le k < L} \sigma_B^2(k), \tag{2.18}$$

gdje je $\sigma_B^2(k)$:

$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]}. \tag{2.19}$$

Otsu algoritam za određivanje praga prikazan je u Algoritmu 1:

**Algoritam 1** Otsu algoritam za određivanje praga

1: Odrediti normalizovani histogram slike i komponente histograma označiti sa $p_i$.
2: Izračunati kumulativne sume, $\omega_0(k)$, za $k = 1, \ldots, L$ koristeći jednačinu (2.3)
3: Izračunati kumulativne srednje vrijednosti, $\mu_0(k)$, za $k = 1, \ldots, L$ koristeći jednačinu (2.8)
4: Izračunati globalnu srednju vrijednost intenziteta, $\mu$, koristeći jednačinu (2.9)
5: Izračunati varijansu između klasa, $\sigma_B^2(k)$, za $k = 1, \ldots, L$ koristeći jednačinu (2.19)
6: Odrediti Otsu prag, $k_{opt}$, kao vrijednost $k$ za koju je $\sigma_B^2(k)$ maksimalna. Ako maksimum nije jedinstven, odrediti $k_{opt}$ prosječnom vrijednošću $k$ koja odgovara različitim detektovanim maksimumima.
7: Odrediti mjeru separabilnosti $\eta$ pomoću jednačine:

$$\eta = \frac{\sigma_B^2(k_{opt})}{\sigma_T^2}.$$

Prethodni metod je imao zadatak da odredi jedan optimalan prag koji je dijelio sliku u dvije klase. Međutim, ukoliko slika sadrži više objekata, sa različitim nivoima osvijetljenosti, sliku treba segmentirati pomoću više pragova $k_0, k_1, \ldots k_N$ , odnosno izraziti je u više klasa $C_0, C_1, \ldots, C_{N+1}$. U tom slučaju, optimalne pragove $k_{0opt}, k_{1opt}, \ldots, k_{Nopt}$ dobijamo maksimiziranjem međuklasne varijanse, koja sada dobija

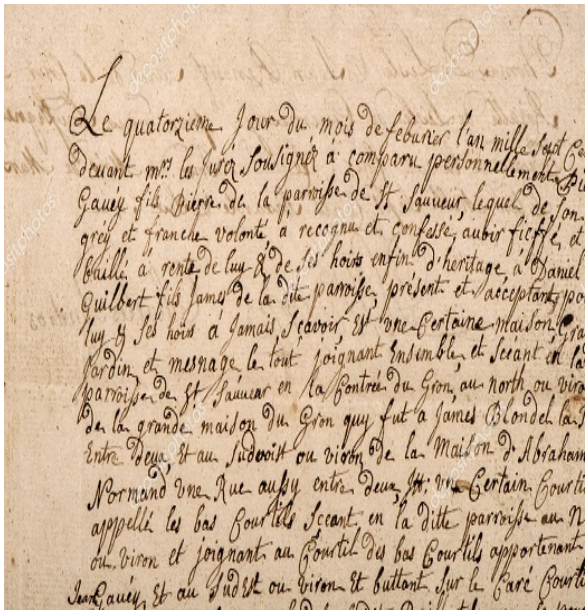
sljedeći oblik [15]:

$$\sigma_B^2(k_{0opt}, k_{1opt}, \ldots, k_{Nopt}) = \max_{1 \leq k_0 < \ldots < k_N < L} \sigma_B^2(k_0, k_1, \ldots, k_N) \tag{2.20}$$

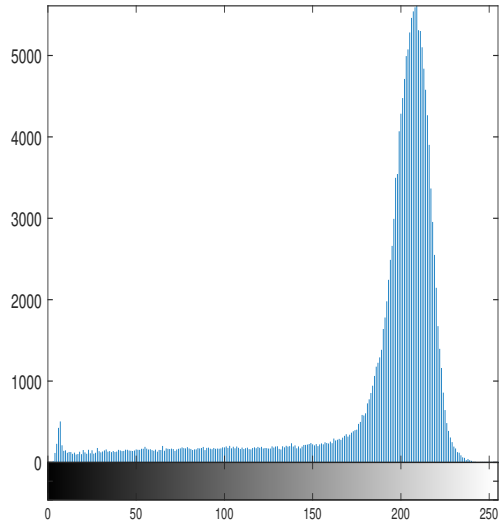
gdje su sada $\sigma_B^2 = \sum_{i=1}^{N} \omega_i(\mu_i - \mu_T)^2$, $\omega_i = \sum_{i \in C_i} p_i$ i $\mu_i = \sum_{i \in C_i} \frac{ip_i}{\omega(k)}$.

Može se primijetiti da Otsu metod radi samo s jednom varijablom, odnosno intezitetom osvijetljenosti. Uzimajući to u obzir, ovaj metod će uspješno segmentirati sliku na dvije ili tri klase. Međutim, povećanjem broj klasa ovaj metod ovaj metod gubi svoje performanse. Iz tog razloga razvijeni su drugi metodi koji uzimaju u obzir i druge parametre slike, na primjer, boju, ili prostorne parametre.

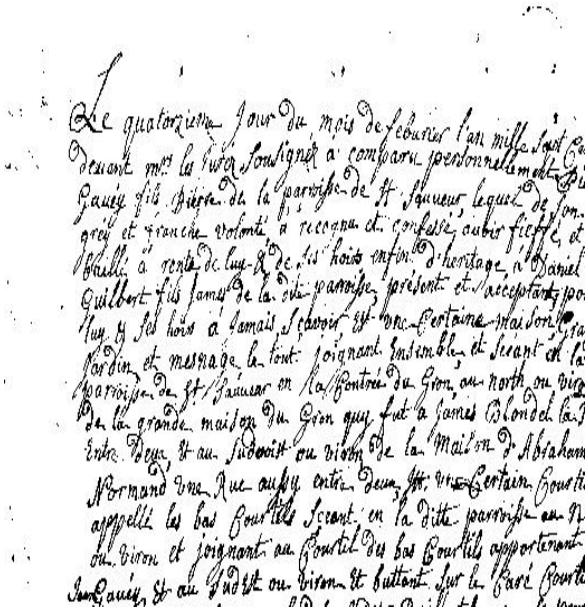
Jedan od naznačajnijih primjena metode segmemntacije pomoću praga jeste u optičkom prepoznavanju karaktera (OCR). Mnogi skenirani dokumenti i slike s tekstovima nisu jasne, ili imaju razmazane boje, i slično. U tu svrhu primijenjuje se ovaj metod pretvaranjem slika tekstova u čistu binarnu sliku, odnosno čist i čitljiv tekst, kao što je prikazano na slici 2.2.



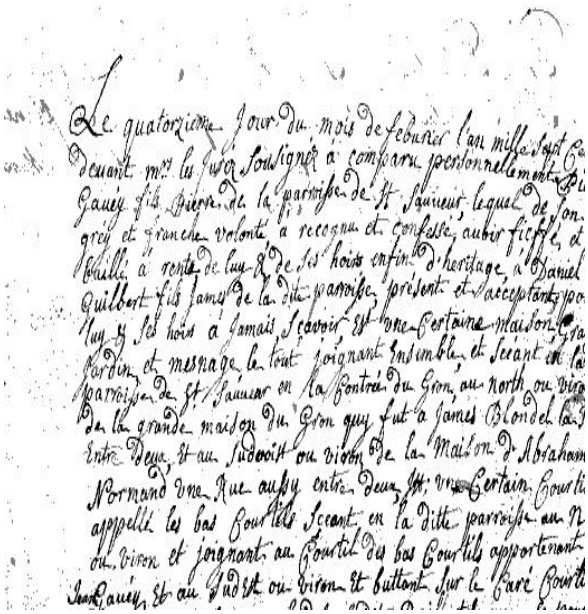
Orginalna slika




Histogram



Seg. slika pomoću jednog praga
Opt. prag: 0.5372,
Međuklasna varijansa: $\sigma_B = 0.8193$



Seg. slika pomoću dva praga
Opt. pragovi: 0.3412 i 0.6549,
Međuklasna varijansa: $\sigma_B = 0.9076$

**Slika 2.2:** Segmentacija slike i histogram, prikaz rezultata segmentacije pomoću jednog i dva praga.

### 2.1.2 Segmentacija pomoću regiona

Osnovni cilj svakog metoda segmentacije, pa i ovog, jeste identifikacija značajnih regiona na slici. Regioni nastaju grupisanjem piksela sa sa sličnim osobinama ili sličnim vrijednostima atributa. Iterativnim postupkom formiraju se regioni, koji, nadovezivanjem susjednih piksela ili sebi sličnih susjednih regiona, postaju veći i više razumljivi. Ovaj metod je poznat kao *metod rasta i spajanja regiona*. Pored pomenutog metoda, postoji i inverzan metod, poznat kao *metod razdvajanja regiona*. Kod ovog metoda, inicijalna slika se posmatra kao jedan region, i iterativnim postupkom algoritam sliku dijeli da podregione, sve dok podregion ne postane homogen. Metodi razdvajanja i spajanja regiona se često kombiniju kako bi se dobili što precizniji regioni.

#### Metod rasta i spajanja regiona

Metod rasta regiona intuitivno je jasan i može se zamisliti kao i svaki prirodni proces rasta, gdje se rast odvija nadovezivanjem piksela ili grupe piksela sa sličnim osobinama, odnosno sličnom osvjetljnošću, bojom ili teksturom. Rast započinje inicijalizacijom jedne ili više tačaka ondnosno piksela koji se nazivaju "sjeme" (engl. *seed*). Međutim, izbor sjemena je poseban zadatak i veoma bitan jer utiče na kvalitet segmentacije. Jedan od najjednostavnijih načina izbora inicijalnog piksela, odnosno sjemena, jeste da se prvi piksel slike $I(m,n)$, za $m = 1$ i $n = 1$, proglasi kao sjeme prvog regiona [16]. Nakon toga, upoređivaće se vrijednosti osvjetljaja 8 susjednih piksela: $I(m,n+1)$, $I(m+1,n)$, $I(m,n-1)$, $I(m-1,n)$, $I(m-1,n-1)$, $I(m-1,n+1)$, $I(m+1,n-1)$ i $I(m+1,n+1)$. Ako susjedni piksel ispunjava uslov homogenosti, onda biva pridružen regionu i vrijednost osvjetljaja mu se mijenja na vrijednost osvijetljaja piksela sjemena. Prvi region će rasti na način što će se svaki novi pridruženi piksel upoređivati sa svojim susjedima i pridružiti ih svom regionu ako zadovoljavaju kriterijum koji je i on sam zadovoljio. Kada prvi region pridruži sebi sve odgovarajuće piksele, i odbaci one koji to nisu, može se početi sa stvaranjem novog regiona.

Krećući se kroz redove matrice piksela slike, s desna na lijevo, pronalazi se prvi piksel koji nije dodijeljen nijednom regionu. Taj piksel se postavlja kao sjeme novog regiona. Postupak provjere 8 susjeda i dodjeljivanja regionu se ponavlja za novi region. Ukoliko u susjedstvu postoje pikseli koji su već dodijeljeni nekom regionu, ti pikseli se ne uzimaju u obzir, tj. ne razmatraju se. Ovaj proces se iterativno ponavlja za svaki novi region. Slika će biti segmentirana kada svaki piksel bude dodijeljen nekom regionu.

Uslov homogenosti, odnosno pripadnosti regionu, ispunjavaju pikseli kod kojih je razlika u osvjetljenju između njih i sjemena $S$ manja od unaprijed definisanog praga $T$, odnosno:

$$|I(m,n) - S| \leq T \tag{2.21}$$

Prag $T$ se bira ručno u zavisnosti od slike.

Nakon segmentacije slike na regione, sledeći korak je spajanje susjednih regiona sa sličnim svojstvima. Ovaj korak omogućava dalju kontrolu nad brojem regiona koje slika sadrži. Proces spajanja regiona može se ostvariti pomoću dvije procedure:

1. Uklanjanje regiona čiji su pikseli šum

   Regioni koji imaju $N$ piksela ili manje smatraju se regionima čiji su pikseli šum. To su regioni sa abnormalnim nivoima sive koje se znatno razlikuju od vrijednosti okolnih regiona. Ove regione možemo ukloniti tako što ćemo ih pridružiti obližnjim regionima. Vrijednosti piksela u ovim regionima se zatim mijenjaju tako da odgovaraju vrijednostima susjednih regiona.

2. Spajanje susjednih regiona

   Ova procedura podrazumjeva spajanje regiona koji su susjedi i koji zadovoljavaju uslov homogenosti 2.21 sa novim pragom spajanja $M$. Vrijednost praga $M$ biće nešto veća od praga $T$ koji je korišten za rast regiona. Spajanje regiona omogućava da se regioni koji su susjedi, a koji se nisu mogli spojiti tokom procesa rasta regiona (zbog nešto niže vrijednosti praga T), spoje u jedan region. Da bi se dva regiona smatrala susjedima, najmanje $J$ piksela sa granice svakog regiona mora biti 8-povezano. Niska vrijednost $J$ rezultirala bi niskim stepenom segmentacije i većim brojem spojenih regiona, dok bi visoka vrijednost $J$ rezultirala suprotnim efektom, tj. većim brojem odvojenih regiona.

Kao što je već ranije pomenuto, izbor sjemena, kao i način rasta regiona, kod ovog metoda je od velikog značaja. Nekada je dobro recimo ručno izabrati sjemena, kao i njihov broj, a to su situcije kada korisnik jasno može identifikovati reprezentativne piksele za različite regione [17]. Pored ručnog odabira takođe se sjemena mogu odabrati automatski.

Automatski odabir sjemena podrazumjeva identifikaciju početnih tačaka za segmentaciju na osnovu karakteristika slike. Algoritam analizira sliku da bi pronašao piksele koji predstavljaju dobre početne tačke za rast regiona. Ove tačke se biraju tako da predstavljaju različite karakteristične regione unutar slike, što omogućava algoritmu da preciznije segmentira sliku. Zatim se koristi prioritetni red da bi obrada piksela bila nezavisna od redoslijeda, omogućavajući konzistentne rezultate segmentacije bez obzira na sekvencu obrade piksela. Prioritetni red osigurava da se pikseli sa najvećim prioritetom (najvećom sličnošću prema kriterijumima kao što su intenzitet ili tekstura) obrađuju prvi, što doprinosi efikasnosti i tačnosti segmentacije slike [18].

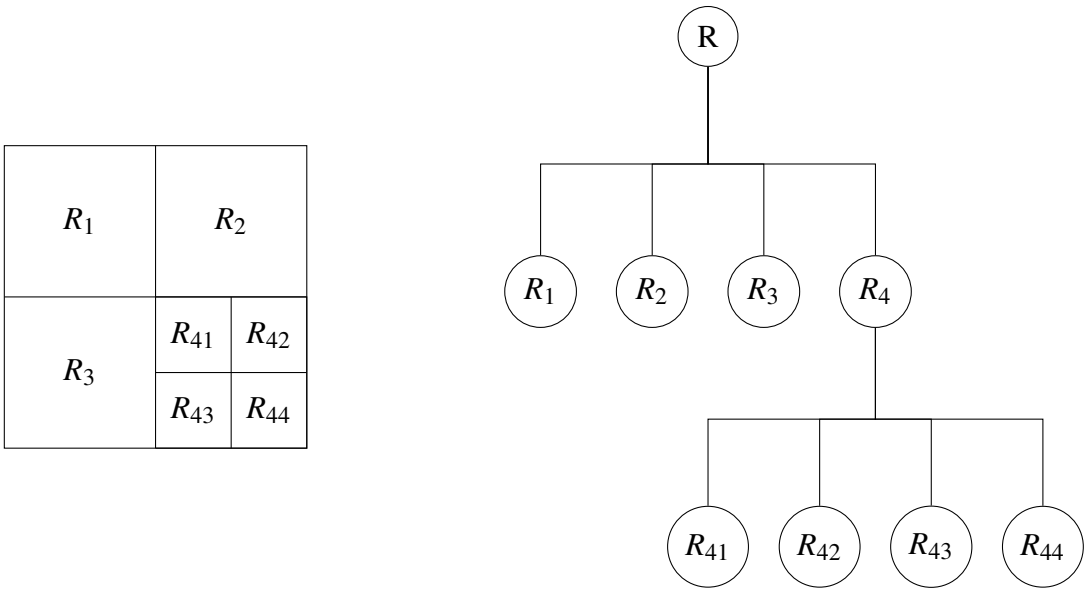
Ovaj metod je našao značajnu primjenu medicini kao što je segmentacija tumora i organa. Metod se široko koristi za segmentaciju tumora u MRI i CT slikama za automatsku detekciju tumora [21]. Ovaj metod se takođe koristi za 3D rekonstrukciju organa, gdje se pomoću ovog metoda iz volumetrijskih podataka segmentiraju organi što omogućava 3D rekonstrukciju [22].

Metod rasta regiona je jednostavan za implementaciju i ne zahtjeva kompleksne algoritme. Može se lako prilagoditi različitim vrstama slika i specifičnim aplikacijama podešavanjem kriterijuma homogenosti. Metod je posebno efikasan kada je potrebno segmentirati povezane regije koje se razlikuju od okoline, kao što je već pomenuto, za tumore na medicinskim slikama. Međutim, jedna od mana jeste osjetljivost na izbor sjemena. Rezultati segmentacije mogu znatno varirati u zavisnosti od izbora početnih tačaka. Takođe, metod može biti osjetljiv na šum u slici, što može dovesti do pogrešnog rasta regiona, a može imati i poteškoće sa segmentacijom regija sa složenim granicama ili nejasnim prelazima između regija.

### Metod razdvajanja regiona i spajanja regiona

S obzirom da u većini slučajeva metod razdvajanja regiona se usavršava i poboljšava pomoću spajanja, otuda je i ovaj metod ima dati naslov (eng. split and merge). U nastavku je dat opis metoda [12].

Ovaj metod počinje proces s pretpostavkom da je početna slika jedan (nehomogen) region. Homogenost se provjerava pomoću takozvanog predikata $P$. Predikat ustvari predstavlja kriterijum koje homogeni regioni treba da zadovolje i uglavnom se oslanjaju na analizu srednje vrijednosti i/ili standardne devijacije regiona. Ako početna slika nije homogena, slijedi postupak dijeljenja slike na 4 regiona, uz uslov da je slika dimenzija $N \times N$, gdje je $N = 2^n$, a regioni dimenzija $M \times M$, gdje je $M = 2^m$, $m \leq n$, kao što je prikazano na slici 2.3. Ovaj postupak je poznat kao *quadtree*, ilustrovan na primjeru 2.3 gdje $R$ predstavlja početnu sliku. Ona se zatim dijeli na 4 jednaka podregiona, od kojih $R_4$, u ovom primjeru, nije zadovoljio kriterijum homogenosti, odnosno $P(R_4) = FALSE$ pa se dalje dijeli na 4 nova podregiona, i tako dalje, sve dok slika bude sastavljena samo od homogenih regiona, odnosno $P(R_i) = TRUE$, za $i = 1, 2, \ldots, k$, gdje je $k$ broj regiona. Takođe je veoma bitno postaviti granice ispod koje se ne može region dijeliti, onosno definisati najmanju veličinu regiona.




**Slika 2.3:** Slika podijeljena u regione (lijevo) i Quadtree struktura (desno).

Ovako segmentirana slika najvjerovatnije neće biti potpuno jasna jer postoji regioni koji su identičnih osobina a ipak su označeni kao posebni regioni. U ovom slučaju se primjenjuje prethodno opisani postupak spajanja regiona. Regioni će zadovoljiti uslov spajanja samo ako zajednički ispunjavaju uslove predikata. Na primjer, homogeni regioni $R_{41}$ i $R_{42}$ mogu se spojiti ako važi $P(R_{41} \cup R_{42}) = TRUE$. Postupak je opisan u Algoritmu 2. Kako bi prethodni algoritam postao još brži i jednostavniji, kriterijum za spajanje regiona se može pojednostaviti na način da ako dva susjedna regiona pojedinačno zadovoljavaju predikat, mogu se odmah spojiti u jedan.

Predikat $P$ možemo definisati na osnovu standardne devijacije regiona $\sigma$ i njegove srednje vrijednosti

**Algoritam 2** Algoritam za razdvajanje i spajanje regiona

**Input:** Početni region $R$
**Output:** Segmentirana slika na regione
**function** QUADTREESPLITANDMERGE($R$)
    **if** $P(R) =$ FALSE **then**
        Podijeliti $R$ na četiri kvadranta: $R_1, R_2, R_3, R_4$
        **for** svaki kvadrant $R_i$ **do**
            QUADTREESPLITANDMERGE($R_i$)
        **end for**
    **else**
        **return** $R$
    **end if**
**end function**
**function** SPOJISUSJEDNEREGIONE($R$)
    **Inicijalizacija** lista regiona $\mathscr{R} = \{R_1, R_2, \ldots, R_n\}$
    **while** bilo koji susjedni regioni $R_i$ i $R_j$ za koje važi $P(R_i \cup R_j) =$ TRUE **do**
        Spojiti $R_i$ i $R_j$ u $R_{novo}$
        Ažurirati listu regija $\mathscr{R}$
    **end while**
**end function**
**function** MAIN($R$)
    QUADTREESPLITANDMERGE($R$)
    SPOJISUSJEDNEREGIONE($R$)
**end function**

$s$, na sljedeći način [12]:

$$P(R_i) = \begin{cases} TRUE, & \text{ako je} \quad \sigma > a \quad \text{i} \quad 0 < s < b, \\ FALSE, & \text{u suprotnom.} \end{cases} \tag{2.22}$$

Konstante $a$ i $b$ se biraju u zavisnosti od slike.

Drugi način definisanja predikata se zasniva samo na vrijednosti piksela regiona, i to maksimalne i minimalne vrijednosti [20]:

$$P(R_i) = \begin{cases} TRUE, & \text{ako je} \quad [\max_{m,n \in R_i} R_i(m,n) - \min_{m,n \in R_i} R_i(m,n)] \leq T, \\ FALSE, & \text{u suprotnom,} \end{cases} \tag{2.23}$$

gdje $R_i(m,n)$ predstavljaju vrijednost piksela na lokaciji $m,n$ a $T$ prestavlja prag, koji se ručno zadaje ili računa posebnima metodama.

Nakon izvršene metode dijeljenja i spajanja regiona, u velikom broju slučajeva će, pored korektnih regiona, postojati i oni mali regioni, dobijeni od piksela sa viskofrekventnim šumom, ili se nalaze na prelazu velikih regiona. Kako bi eliminisali njih, izvršava se spajanje ovih regiona sa svojim većim susjedom, što i zapravo predstavlja poslednji korak ovog metoda. Međutim, predikat je u tom slučaju

definisan na malo drugačiji način:

$$P(R_i \cup R_j) = \begin{cases} TRUE & \text{ako je} \quad |\sum_{(m,n)\in R_i} R_i(m,n)/S_i - \sum_{(m',n')\in R_j} R_j(m',n')/S_j| \leq T_1, \\ FALSE & \text{u suprotnom,} \end{cases} \tag{2.24}$$

gdje $R_i(m,n)$ i $R_j(m',n')$ predstavljaju vrijednosti piksela na poziciji $(m,n)$ u regionu $R_i$ i $(m',n')$ u regionu $R_j$, koji se sumiraju i dijele sa odgovarajućim brojem piksela $S_i$ i $S_j$ respektivno.

Metod razdvajanja i spajanja regiona našao primjenu u segmentaciji slike slike terena zemljišta što ima izuzetnu važnost za spektralnu analizu i za poboljšanu preciznost klasifikacije [23].

Ovaj metod, kombinovanjem procesa dijeljenja i spajanja omogućava bolju kontrolu nad segmentacijom, što može rezultirati preciznijim identifikovanjem homogenih regija. Takođe je manje osjetljiv na šum u poređenju sa prethodnim metodom, jer koristi globalne kriterijume homogenosti. Međutim proces dijeljenja i spajanja može biti računski intenzivan, naročito za velike slike ili slike sa velikim brojem regija, i može imati teškoća sa finim detaljima i preciznom segmentacijom zbog kvadratnog dijeljenja prostora.

### Metod označavanja regiona

Prethodno opisani metodi pronalaze značajne regije i objekte na slici, čije postojanje i razliku možemo uočiti na osnovu različite vrijednosti piksela. Na primjer, kod binarne slike dobijene, recimo, opisanom metodom pomoću praga, objekat od značaja je crne a pozadina bijele boje. Kako bi još jasnije definisali i uočili objekat, može se sprovesti postupak iscrtavanja granica oko već pronađenog objekta. Ovim postupkom sprovodimo završnu fazu segmentacije.

Prije svega, da bi razumjeli proces označavanja regiona, treba se upoznati sa terminom *lančanog kodiranja* (eng. *Chain Code*). Pretpostavimo da se krećemo po granici jednog od regiona slike, i da krećemo od piksela $P$ čija je pozicija $(m,n)$. Kretanje nastavljamo ako neki od piksela iz 8-susjedstva zadovoljava predefinsani uslov. Ako se pomjerimo udesno, odnosno na pozicji $(m,n+1)$ pomjeraj kodiramo sa 0, pomjeraj na poziciji $(m-1,n+1)$ kodiramo sa 1, i tako dalje, kroz cijelo 8-susjedstvo. Prethodni postupak se takođe može sprovoditi i na 4-susjedstvo, odnosno piksel sa pozicije $(m,n+1)$ kodiramo sa 0, piksel sa $(m-1,n)$ sa 1, $(m,n-1)$ sa 2, i piksel sa $(m+1,n)$ sa 3, odnosno pikseli koji se nalaze dijagonalno u odnosu na piksel $P$ se ne uzimaju u obzir. Ove dvije vrste susjedstva su ilustrovane na slici 2.4.

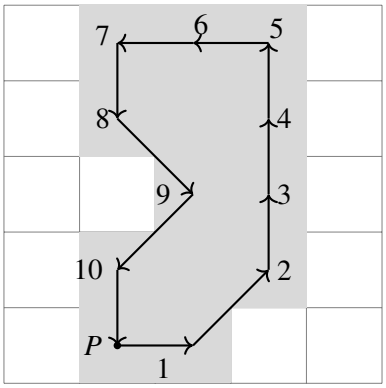
| | | |
|---|---|---|
| 3 | 2 | 1 |
| 4 | P | 0 |
| 5 | 6 | 7 |

| | | |
|---|---|---|
| | 1 | |
| 2 | P | 0 |
| | 3 | |

**Slika 2.4:** Ilustracija 8-susjedstva (lijevo) i 4-susjedstva (desno)

Na slici 2.5 predstavljena je ilustracija jdnostavnog primjera lančano kodiranja uzimajući u obzir

8-susjedstvo.




**Slika 2.5:** Ilustracija lančanog kodiranja duž granice regiona.

Lančani kod za granice regiona sa 2.5 korišćenjem provjere 8-susjedstva dat je u tabeli 2.1

| $p$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $C[p]$ | 0 | 1 | 2 | 2 | 2 | 4 | 4 | 6 | 7 | 5 | 6 |

**Tabela 2.1:** Lančani kod za granice regiona slike 2.5

Neka imamo za cilj da odredimo *unutrašnju* granicu nekog regiona $R$. Unutrašnja granica je kontura koja je nalazi na granici regiona i podskup je regiona, za razliku od *spoljašnje* granice koja nije podskup regiona. Krenućemo se kroz piksele slike počevši od donjeg lijevog ugla (od piksela $(0,0)$). Potrebno je pronaći piksel $P$ koji pripada novom regionu, odnosno taj piksel još uvjek nije obrađivan, ili pripada trenutnom regionu koji se obrađuje i nalazi se što više dolje i što više lijevo na slici u odnosu na sve druge piksele tog regiona. Kada se pronađe takav piksel $P$, može početi postupak označavanja granica. Zapamtićemo njegovu poziciju $(m,n)$ i inicijalizovati varijablu $\vec{d}$ gdje je potrebno čuvati kodove pri pomjeraju odnosno provjeri susjedstva. Sada se sprovodi prethodno opisani postupak lančanog kodiranja počevši od piksela $P$, dok u varijabli $\vec{d}$ smještamo pravce odnosno kodove, koje smo dobili ispitivanjem susjedstva. Uslov koji bi trebao susjedni piksel da zadovolji jeste da ima istu vrijednost kao piksel $P$. Međutim, umjesto zahtjeva da susjedni piksel mora imati identičnu vrijednost kao piksel $P$, može se koristiti uslov koji dozvoljava slične, ali ne identične vrijednosti. Na primjer, u algoritam se može dodati tolerancija za odstupanje u intenzitetu boje između piksela $P$ i njegovih susjeda. Na ovaj način, algoritam ima manju šansu da zaglavi i može nastaviti i ako nema piksela sa potpuno istom vrijednošću. Kretanje po granici odnosno konturi regiona se vrši sve dok se ne dođe u početani položaj, odnosno u piksel $P$. Ovaj algoritam određivanja granica se može primijeniti na sve regione veće od jednog piksela. Međutim ukoliko postoji "rupa" u regionu odnosno region je unutar nekog regiona, u tom slučaju se prethodno opisani algoritam ne može primjeniti.

Ukoliko bismo izvršili prethodni algoritam i koristili 4-susjedstvo, lako možemo označiti i *spoljašnju* granicu regiona. Kada se odredi unutrašnja granica, onda se spoljašnja granica sastoji od piksela koji ne

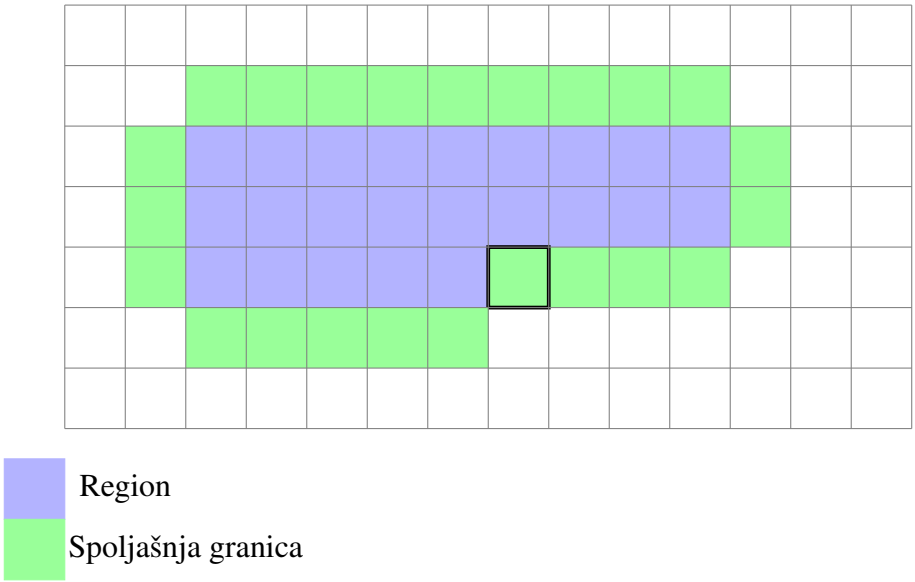
pripadaju regiji ali su uzeti u razmatranje, odnosno provjeravani su tokom analize 4-sujsedstva.

Spoljašnja granica regiona može biti vrlo korisna jer omogućava računanje obima regiona. Takođe, spoljašnja granica regije nikada nije dio samog regiona, što znači da dva susjedna regiona nemaju zajedničku granicu. Ova osobina pomaže da se izbjegnu problemi prilikom ostalih nivoa obrade slike, kao što je spajanje regiona, jer se ne javlja konflikt u prepoznavanju gdje jedan region završava a drugi počinje. Na taj način, svaki region je jasno definisan i odvojen od svojih susjeda.

Veoma često se prilikom određenih granica regiona na prethodni način, uočava da veoma bliski susjedni regioni imaju preklopljene granice ili se uočava mala praznina između njih. Ovakav problem bi se mogao riješiti formiranjem *proširenih granica*. Proširena granica se lako izvodi iz spoljašnje granice. Pretpostavimo da je spoljašnja granica dobijena analizom 8-susjedstva. Postupak proširene granice se sprovodi na sljedeći način:

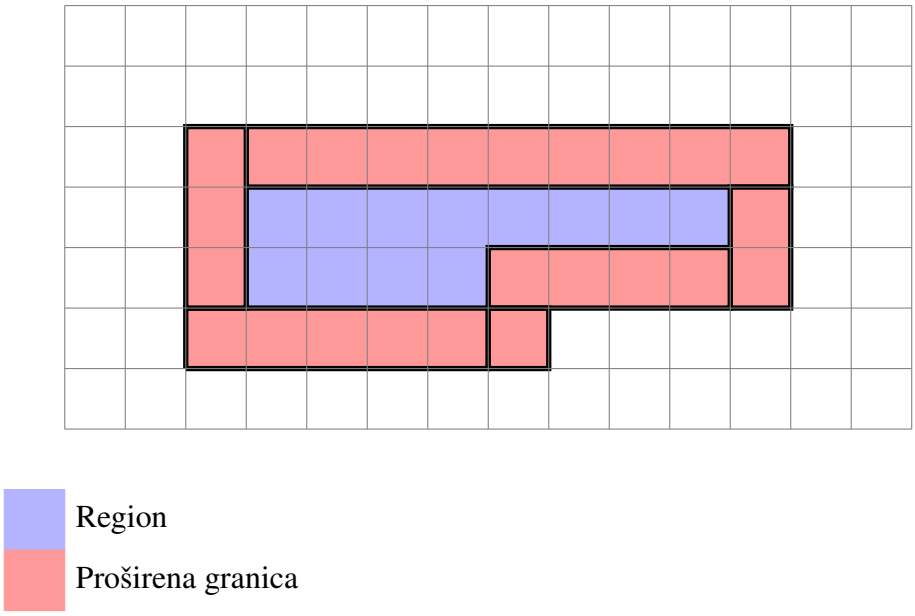
- sve piksele spoljašnje granice koji su kodirani sa 2, odnosno to su *gornji* susjedi, treba pomjeriti jedan piksel dolje a zatim jedan piksel desno,
- pikseli kodirani sa 4, odnosno lijevi pikseli, pomjeraju za jedan piksel desno,
- pikseli kodirani sa 0, odnosno desni susjedi, pomjeraju se jedan piksel dolje,
- svi donji pikseli, odnosno susjedi kodirani sa 6, zadržavaju svoju poziciju.

Na slikama 2.6 i 2.7 je dat jednostavan primjer određivanja proširene granice izvedene iz spoljašnje granice regiona.




**Slika 2.6:** Prikaz glavnog regiona sa spoljašnjim granicama

Proširena granica je ilustrovana na slici 2.7:




**Slika 2.7:** Prikaz proširenog regiona

Metod označavanja regiona predstavlja ključan korak u procesiranju slika, omogućavajući precizno razlikovanje objekata unutar slike. Kod binarnih slika, gdje su vrijednosti piksela obično crne ili bijele, postupak segmentacije i identifikacije objekata je jednostavan i efikasan jer postoje jasne granice između različitih regiona. Međutim, kada radimo sa slikama koje nijesu binarne, tj. slikama sa više nivoa sive ili boja, proces postaje složeniji. Razlikovanje objekata u ovakvim slikama zahtjeva dodatne kriterijume za identifikaciju i iscrtavanje granica jer nema oštrog prelaza između piksela različitih regiona.

U ovakvim situacijama, potrebni su sofisticiraniji pristupi kao što su adaptivne metode segmentacije, algoritmi zasnovani na intenzitetu ili boji, i naprednije tehnike poput klasterovanja piksela. Ovi pristupi pomažu u analizi nijansi i glatkih prelaza unutar slike, ali su često kompleksniji i računski zahtjevniji.

### 2.1.3 Segmentacija pomoću detekcije ivica

Za razliku od metoda koji iscrtava granice već identifikovanih regiona, ovaj pristup se fokusira na pronalaženje i označavanje ivica objekata na slici. Ivice su karakteristične po tome što predstavljaju uske oblasti diskontinuiteta u osvijetljenosti ili boji piksela. Zbog ove osobine, ivice su veoma korisne za označavanje granica i objekata na slici, pa se ovaj postupak može smatrati jednim od metoda segmentacije slike [24].

S obzirom na to da ivice na slici odgovaraju oblastima sa značajnim promjenama u intenzitetu ili boji, prvi izvod funkcije osvetljenja ili boje dostiže svoj maksimum na tim mjestima. Stoga, računanje prvog izvoda postaje ključni alat za detekciju ivica. Ovaj princip je osnova za mnoge algoritme za detekciju ivica. Kada primjenimo prvi izvod na sliku, možemo identifikovati oblasti gdje dolazi do naglih promjena u intenzitetu piksela, što ukazuje na prisustvo ivica.

Detektori ivica, kao što su Sobel-ov, Prewitt-ov i Canny-ev detektor, zasnovani su na analizi prvog

izvoda.

Pretpostavimo da osvjetaljaj slike na poziciji $(x,y)$ ima vrijednost *I(x,y)*. Prvi izvod u toj tački se računa kao:

$$\nabla I(x,y) = [I_x(x,y), I_y(x,y)], \tag{2.25}$$

gdje su:

$$I_x(x,y) = \frac{\partial I(x,y)}{\partial x}, \qquad I_y(x,y) = \frac{\partial I(x,y)}{\partial y}, \tag{2.26}$$

Ivicu onda možemo detektovati tako da detetektor ivica:

$$e(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2}, \tag{2.27}$$

bude veći od nekog već predefinasnog praga. Onda tačku $(x,y)$ proglašavamo kao ivičnu tačku.

Kako su slike koje se obrađuju na računaru diskretne, pa umjesto kontinualnih koordinata $x$ i $y$ imamo diskretne indekse $m$ i $n$. Za proračun izvoda se koriste diskretne aproksimacije zasnovane na konačnim razlikama. Gradijent se aproksimira korišćenjem razlika u vrijednostima piksela. Aproksimacija gradijenta slike najčešće se radi pomoću diskretnih razlika duž kolona (horizontalni pravac) i vrsta (vertikalni pravac). Ova aproksimacija se može predstaviti na sljedeći način:

- *Horizontalna aproksimacija gradijenta*: Izračunava se razlika između vrijednosti susjednih piksela u istoj vrsti: $I_m(m,n) = I_m(m,n) - I_m(m,n+1)$,
- *Vertikalna aproksimacija gradijenta*: Izračunava se razlika između vrijednosti susjednih piksela u istoj koloni: $I_n(m,n) = I_n(m,n) - I_n(m+1,n)$.

Detektori ivica, kao što su Sobel-ov i Prewitt-ov operator, koriste ove diskretne razlike za aproksimaciju gradijenta. Na primjer, Sobel-ov operator koristi konvolucione kernele koji računaju razlike u horizontalnom i vertikalnom pravcu kako bi identifikovali ivice:

$$S_H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix},$$

$$S_V = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix},$$

gdje $S_H$ i $S_V$ predstavljaju Sobel-ov horizontalni i vertiklani kernel respektivno.

Kada se ovi kernali primijene na sliku, rezultati su aproksimacije gradijenata u horizontalnom i vertikalnom pravcu. Kombinovanjem ovih gradijenata, moguće je identifikovati oblasti sa velikim promjenama u intenzitetu piksela, tj. ivice objekata na slici. Sličan Sobel-ovom, je Prewitt-ov, ali koristi drugačije vrijednosti za konvolucione kernale. Prewitt-ov operator je jednostavniji za implementaciju,

ali je manje robustan na šumove:

$$P_H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix},$$

$$P_V = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

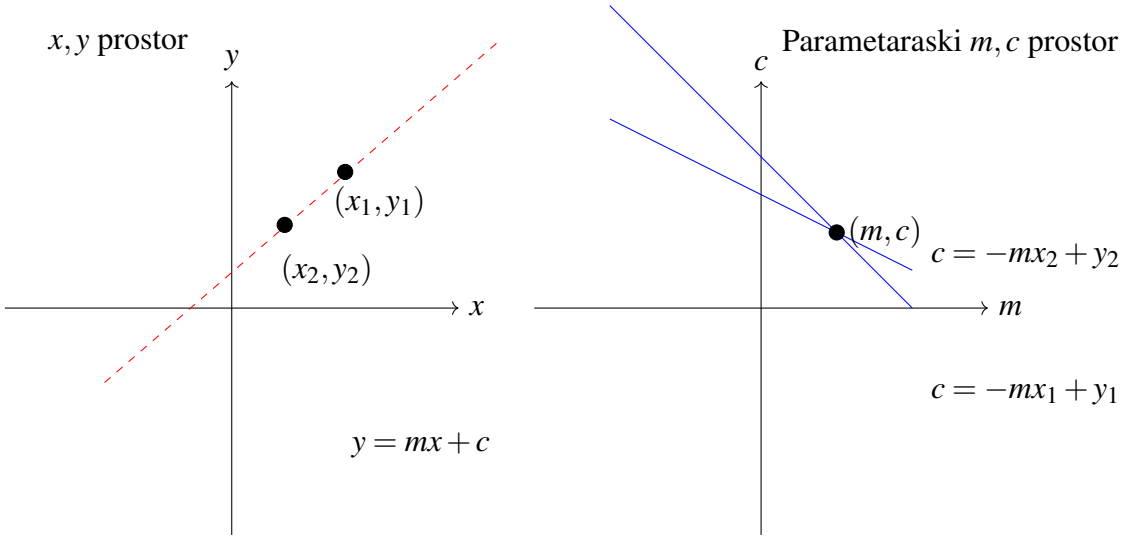
gdje $P_H$ i $P_V$ predstavljaju Perwitt-ov horizontalni i vertiklani kernel respektivno.

Jedan od do sada najboljih detektora koji se koristi jeste *Kanny-ev detektor*. Ovaj algoritam prolazi kroz nekoliko faza kako bi identifikovao ivice sa velikom tačnošću i robusnošću:

- smanjenje šuma: prvo se primijenjuje Gausov filtar kako bi se smanjio šum na slici. Na taj način se smanjuje rizik od detekcije lažnih ivica,
- računanje promjene u intenzitetu slike (gradijenta) nakon što je šum smanjen primjenom Gaussovog filtra,
- primjena dvostrukog praga: detektor koristi dva praga, visok i nizak. Pikseli sa gradijentima većim od visokog praga se označavaju kao jake ivice, dok se oni sa gradijentima između visokog i niskog praga označavaju kao slabe ivice. Pikseli sa gradijentima ispod niskog praga se odbacuju. Slabe ivice se uključuju u finalni rezultat samo ako su povezane sa jakim ivicama, čime se osigurava kontinuitet ivica.

Prilikom detekcije ivica često se javljaju neki faktori koji utiču na smanjeni kvalitet odnosno preciznost detektovanih ivica. Na primjer, šum može izazvati nepravilnosti u gradijentu slike, što može dovesti do prekinutih ili pogrešnih detekcija ivica. Još jedan od problema mogu biti blage ivice. Kada je promjena intenziteta između objekta i pozadine mala, detektori ivica mogu imati problema da precizno identifikuju ivicu. Ove blage ivice mogu se pojaviti kao isprekidane linije jer detektor ne može jasno razlikovati granice objekta. Takođe, nizak kontrast između objekta i pozadine može uzrokovati probleme pri detekciji ivica. Kada su razlike u osvetljenju ili boji suptilne, detektori ivica mogu propustiti djelove ivica, što dovodi do isprekidanih linija. Kompleksni oblici i zakrivljene ivice može biti teško detektovati. Algoritmi za detekciju ivica koji koriste jednostavne kernele, poput Sobelo-vog ili Prewitt-ovog, mogu imati problema sa preciznim praćenjem zakrivljenih ivica, što dovodi do prekinutih segmenata.

Može se zaključiti da svaki od navedenih faktora uzrokuje isprekidane ivice. Iz tog razloga razvijene su brojne metode za spajanje ivica. Jedan od najefikasnijih metoda za rješavanje problema detekcije geometrijskih oblika na slici jeste *Hough-ova transformacija*. Ovaj metod prepoznaje geometrijske oblike tako što transformiše njihove jednačine u parametarski prostor, koristeći pri tom veoma mali broj parametara. Na taj način, osnovni oblici na slici mogu biti prepoznati i analizirani kroz njihove parametarske




**Slika 2.8:** Ilustracija prave u $(x,y)$ prostoru i prostoru parametra $(m,c)$

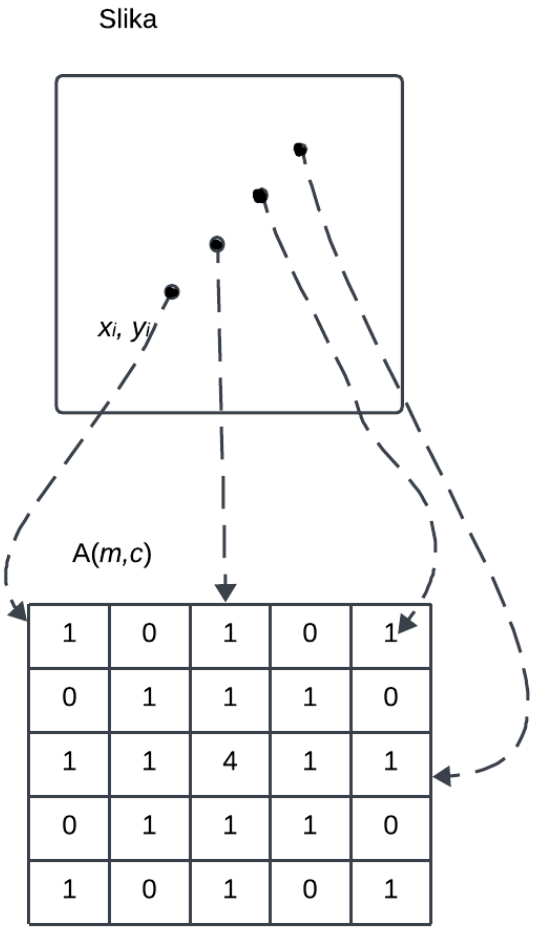
predstave. Krenućemo od osnovnog primjera sa pravom. Posmatrajmo pravu u $(x,y)$ kooridnatnom sistemu koja prolazi kroz dvije tačke $x_1, y_1$ i $x_2, y_2$, i koja je još određena parametrima $m, c$. Svaka tačka u $(x,y)$ prostoru može biti predstavljena kao prava u $(m,c)$ prostoru (parametarskom prostoru). Dakle, za svaku tačku $x_i, y_i$ na slici, postoji beskonačno mnogo pravih koje mogu proći kroz tu tačku, ali sve te prave zadovoljavaju parametarsku jednačinu $c = -mx_i + y_i$. Ako više tačaka na slici leže na istoj pravoj, odgovarajući pravci u parametarskom prostoru će se presjeći u jednoj tački, koja predstavlja parametre $m, c$ te prave, kao što je prikazano na slici 2.8.

Da bismo praktično implementirali Hough-ovu transformaciju, koristimo akumulacionu matricu koja broji koliko pravih prolazi kroz svaku tačku u parametarskom prostoru. Maksimumi u akumulacionoj matrici odgovaraju pravama u prostoru slike $(x,y)$ koje sadrže mnogo tačaka. Maksimumi u akumulacionoj matrici odgovaraju parametrima linija koje sadrže najveći broj tačaka u prostoru slike. Ove dominantne linije predstavljaju glavne ivice u slici [20]. Jedan prost primjer prikazan je na slici 2.9, gdje četiri piksela predstavljaju četiri tačke. Akumulaciona matrica se inicijalizuje nulama. Tačke sa slike konvertujemo u parametarski oblik $(m,c)$. Kada se tačka sa slike parametrizuje, odgovarajuća vrijednost u akumulacionoj matrici za dati parametarski par $(m,c)$ se inkrementira za 1. Ova inkrementacija beleži koliko linija može da prođe kroz konkretni par parametara $(m,c)$. Maksimumi u akumulacionoj matrici, u našem primjeru 4, odgovaraju parametrima linija koje prolaze kroz najveći broj tačaka u prostoru slike.

Kod parametrizacije u $m,c$ prostoru, gdje $m$ predstavlja nagib a $c$ presjek, vertikalne linije postaju problematične jer nagib $m$ teži beskonačnosti. Nasuprot tome, koriste se polarne koordinate koje omogućavaju jedinstvenu reprezentaciju svih linija pa i vertikalnih. Polarni oblik je

$$\rho = x\cos\theta + y\sin\theta,$$

što je jednostavan linearni izraz u parametarskom prostoru. Parametrizacija sa $\rho$ i $\theta$ omogućava stabilnije i preciznije računanje, posebno u prisustvu šuma i diskretizovanih podataka. Koordinate $\theta$ (ugao) i $\rho$




**Slika 2.9:** Primjer akumulacione matrice

(udaljenost od početka koordinatnog sistema) omogućavaju robustniju detekciju linija.

Pored detekcije pravih linija, Hough-ova transformacija može se koristiti za prepoznavanje i drugih geometrijskih oblika na slikama, kao što su krugovi, elipse, i pravougaonici, koristeći odgovarajuće parametarizacije. Pošto Hough-ova transformacija identifikuje linije na osnovu akumulacije glasova iz više tačaka, ona može uspešno spojiti isprekidane linije. Čak i ako linija nije kontinualna u prostoru slike, sve tačke koje pripadaju toj liniji će se akumulirati u istu ćeliju u parametarskom prostoru, omogućavajući detekciju kompletne linije.

### 2.1.4 Segmentacija pomoću neuralnih mreža

Upotreba dubokog učenja, posebno neuralnih mreža, značajno je unaprijedila performanse segmentacije slike, omogućavajući preciznije i brže rezultate. Pomoću raznih, do sada razvijenih modela ovog metoda, moguće je izvršiti kako *semantičku*, tako i *segmentaciju instanci* slika. Semantička segmentacija klasifikuje svaki piksel u slici u određenu klasu (npr. osoba, automobil, drvo), dok segmentacija instanci ne samo da klasifikuje piksele, već i razlikuje pojedinačne instance unutar iste klase (npr. dva različita automobila).

Konvolucione neuralne mreže (engl. *CNN - Convolutional Neural Networks*) predstavljaju temelj za većinu savremenih pristupa segmentaciji slike. CNN se sastoje od niza slojeva za konvoluciju (konvolucija - operacija koja izdvaja karakteristike iz slike), aktivaciju (aktivaciona funkcija koja uvodi nelinearnost u mrežu) i pododabiranje (engl. *sub-sampling* ili *pooling* - operacija koja smanjuje dimenzije slike), koji zajedno omogućavaju ekstrakciju hijerarhijskih karakteristika slike. Osnovne arhitekture kao što su *VGG*, *ResNet* i *Inception* često služe kao osnova za mreže koje se koriste za segmentaciju [25].

Jedan od najpoznatijih modela za semantičku segmentaciju slike je **U-Net**. U-Net se sastoji od simetrične arhitekture sastavljene od enkodera i dekodera. Enkoder je niz konvolucionih i pooling slojeva koji smanjuju dimenzije slike i izvlače karakteristike, dok dekoder koristi niz dekonvolucionih (engl. *upsampling* - povećavanje dimenzija slike) slojeva za rekonstrukciju slike na originalnu rezoluciju. U-Net koristi skip veze (engl. *skip connections* - preskok veze) koje omogućavaju prenošenje informacija iz ranijih slojeva direktno u odgovarajuće slojeve dekodera, čime se poboljšava preciznost segmentacije [26].

Za segmentaciju instanci, **Mask R-CNN** je jedan od najefikasnijih modela. Mask R-CNN proširuje Faster R-CNN tako što, pored regije interesa (RoI) za detekciju objekata, dodaje i granu za predikciju maske piksela za svaku detektovanu instancu. Ovaj pristup omogućava ne samo prepoznavanje objekata, već i njihovu preciznu segmentaciju unutar slike [27].

Postoje različite tehnike koje se koriste za poboljšanje performansi modela za segmentaciju slike:

- *Augmentacija podataka*: tehnike kao što su horizontalno i vertikalno okretanje, promjene osvjetljenja i kontrasta, te dodavanje šuma, mogu značajno povećati robusnost modela.
- *Transfer učenje*: korišćenje unaprijed treniranih modela na velikim datasetovima (kao što je ImageNet) može poboljšati efikasnost i smanjiti vrijeme treniranja.

- *Postprocesiranje*: tehnike kao što se CRF (engl. *Conditional Random Fields*) često koriste za rafiniranje ivica segmenata i poboljšanje preciznosti segmentacije.

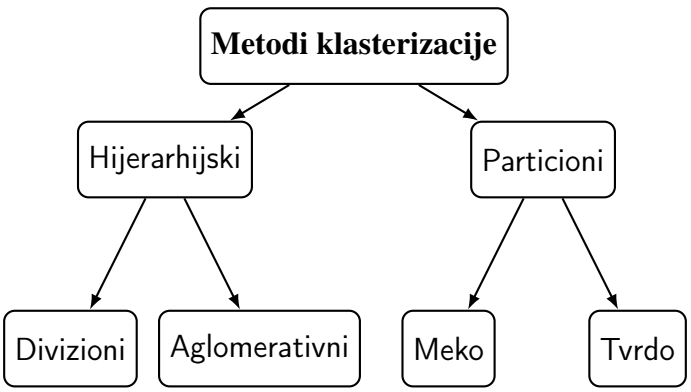
Duboko učenje i neuralne mreže revolucionizirale su segmentaciju slike, omogućavajući postizanje visokih nivoa tačnosti i robusnosti. Modeli kao što su U-Net i Mask R-CNN predstavljaju ključne korake naprijed u ovoj oblasti, omogućavajući preciznu segmentaciju i segmentaciju instanci. Sa daljim razvojem novih arhitektura i tehnika, očekuje se da će performanse segmentacije slike nastaviti da se poboljšavaju.

### 2.1.5 Segmentacija pomoću klasterizacije

Prethodni metod segmentacije slike zasnovan na neuralnim mrežama zahtjeva proces treniranja, što znači da model mora biti obučen na unaprijed označenim podacima. Ovaj proces obuke omogućava modelu da nauči karakteristike različitih klasa objekata na osnovu velikog broja primeraka, čime se postiže visoka preciznost segmentacije. Ovaj pristup pripada modelu nadgledanog učenja (engl. *supervised learning*), gde su podaci za obuku obilježeni, tj. unaprijed se zna kojoj klasi svaki primjerak pripada.

S druge strane, metod klasterizacije je primjer nenadgledanog učenja (engl. *unsupervised learning*). Kod klasterizacije, model ne zahtjeva unaprijed označene podatke za obuku. Umjesto toga, algoritmi klasterizacije sami analiziraju podatke i grupišu ih u klastere na osnovu sličnosti među podacima. Ovo znači da se slični pikseli u slici grupišu zajedno bez potrebe za prethodnim označavanjem, što omogućava segmentaciju slike bez potrebe za velikim brojem označenih podataka.

Metode klasterovanja imaju za cilj da grupišu nepovezane piksele slike u homogene grupe, odnosno klastere, u kojima su elementi unutar svake grupe što sličniji jedni drugima, dok su elementi između grupa što različitiji. Na taj način se postiže jasna identifikacija i razdvajanje različitih objekata ili oblasti na slici [11]. Uzimajući u obzir da metod klasterizacije postoji već odavno, do sada su razvijeni brojni oblici ovog metoda [28]. Iako u literaturi nije jasno definisano, moglo bi se ipak reći da postoji neka gruba podjela (klasifikacija) algoritama za klasterizaciju, koja je definisana na osnovu načina formiranja klastera [11]. Metod koji prilikom formiranja klastera teži ka tome da zadovolji neku funkciju cilja naziva se *particioni*, za razliku od *hijerarhijskog* koji ne "preispituje odluku" koju je donio prilkom procesa klasterizacije. Detaljnija podjela prikazana je na slici 2.10:




**Slika 2.10:** Vrste i podjela klasterizacije

Hijerarhijske metode ne preispituju odluke jer su zasnovane na izgradnji hijerarhijskog modela klastera koji se kreira korak po korak. Svaki korak zavisi od prethodnog, što znači da se jednom napravljeni koraci ne mijenjaju. Na primjer, u *aglomerativnim* metodama, svaka tačka podataka počinje kao zaseban klaster, a zatim se iterativno spajaju dva najbliža klastera dok se ne formira jedno klaster drvo. Slično tome, u *divizionim* metodama, svi podaci počinju u jednom klasteru, koji se zatim iterativno dijeli na manje klastere. Ovo čini metode jednostavnijim i bržim, ali može rezultirati suboptimalnim klasterima ako se početne odluke pokažu kao loše.

S druge strane, particioni metod je dosta pouzdaniji, jer prilikom razmatranja elementa (u našem slučaju piksela), preispituje pripadnost svakom od klastera, gdje nakon zadovoljenja funkcije cilja biva pridružen jednom od njih, ili biva označen vjerovatnoćom pripadnosti svakom od klastera. S tim u vezi, postoje dvije vrste ovog metoda, *meko* i *tvrdo* klasterisanje. Meko klasterisanje se odnosi na tehniku u kojoj svaki element ima pridruženu vrijednost pripadnosti odnosno vjerovatnoću pripadnosti različitim klasterima. Drugim riječima, element može pripadati više klastera istovremeno, ali sa različitim stepenima pripadnosti a ukupna suma mora biti jednaka 1, kao što je prikazano na slici 2.11. Na primjer crna zvjezdica pripada najviše prvom klasteru sa vrijednostima pripadnosti (0.85, 0.1, 0.05), što znači da zvezdice pripadaju prvom klasteru sa verovatnoćom od 85 %, drugom sa 10% i trećem sa 5%.

Najpoznatiji predstavnik ovog metoda jeste **Fuzzy C-means**

Fuzzy C-means (FCM) je algoritam koji dozvoljava meke granice, što je korisno tamo gdje postoji preklapanje između klastera, gdje podaci nisu jasno razdvojeni ili kada je potrebna fleksibilnost u određivanju klastera [29].

Algoritam FCM minimizira sljedeću funkciju cilja:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \|x_i - c_j\|^2 \tag{2.28}$$

gdje je:

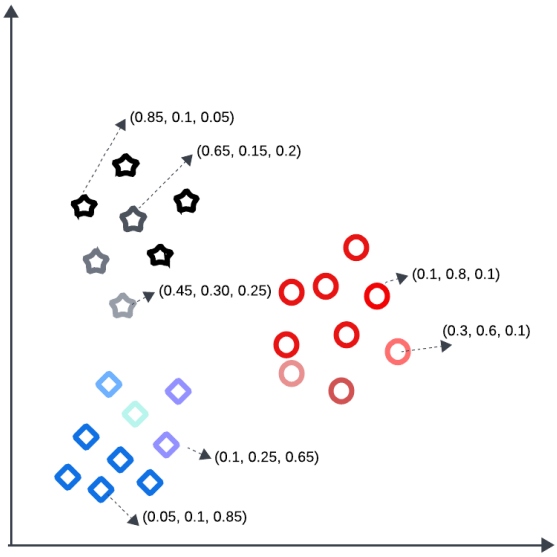
- $N$ broj podataka
- $C$ broj klastera
- $u_{ij}$ stepen pripadnosti podatka $x_i$ klasteru $j$
- $m$ je stepen zamućenosti (fuzziness exponent) (obično $1 < m < 2$)
- $x_i$ je $i$-ti podatak
- $c_j$ je centroid $j$-tog klastera
- $\|x_i - c_j\|^2$ je Euklidska distanca između podatka $x_i$ i centroida $c_j$

Centroidi klastera se ažuriraju koristeći izraze:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}, \tag{2.29}$$

dok stepen pripadnosti $u_{ij}$ se ažurira koristeći:

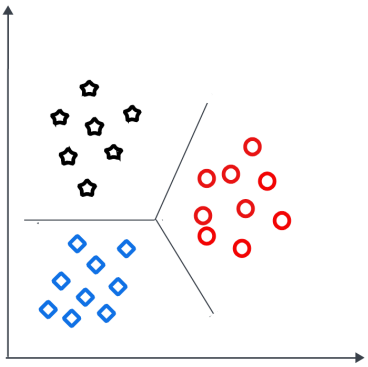
$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left( \frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}. \tag{2.30}$$




**Slika 2.11:** Primjer mekog klasterisanja

FCM je jako dobar alat za klasterizaciju koji se široko koristi u različitim domenima zbog svoje fleksibilnosti, i može se efikasno koristiti u prisustvu šuma i nepreciznih podataka, međutim može biti računski intenzivan za velike skupove podataka.

Za razliku od mekog klasterisanja, koje omogućava da element pripada više klastera sa različitim stepenima pripadnosti, tvrdo klasterisanje dodjeljuje svaki element samo jednom klasteru. U tvrdom klasterisanju, pripadnost elementa klasteru je stroga i binarna: element ili pripada klasteru (pripadnost je 1) ili ne pripada klasteru (pripadnost je 0), kao što je prikazano na slici 2.12. Ovaj pristup se zasniva na funkciji cilja koja minimizira ukupnu varijaciju unutar klastera, odnosno sumu kvadrata Euklidske distance između tačaka i centroida njihovih klastera.



**Slika 2.12:** Primjer tvrdog klasterisanja

Funkcija cilja koja se minimizuje u ovom metodu je suma kvadrata Euklidske distance:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{2.31}$$

gdje je:

- $k$ broj klastera,
- $C_i$ $i$-ti klaster ,
- $x$ podatak koji pripada klasteru $C_i$,
- $\mu_i$ centar (centroid) klastera $C_i$.

Centroid klastera $C_i$ definisan je kao:

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x, \tag{2.32}$$

gdje $|C_i|$ označava kardinalnost, odnosno, broj elemenata skupa $C_i$.

Ova funkcija cilja osigurava da su elementi unutar klastera što bliže centroidu klastera, čime se postiže maksimalna homogenost unutar klastera i minimalna sličnost između različitih klastera. Najpoznatiji algoritam ovog metoda poznat je kao **K-means** algoritam, čiji je pseudokod dat u algoritmu 3.

**Algoritam 3** K-means algoritam

1: **Input:** Slika dimenzija $m \times n$, broj klastera $K$
2: **Output:** Slika segmentirana na $K$ klastera
3: Nasumično izabrati različite tačke podataka kao početne centre klastera, $\mu_i$ ;
4: **while** uslov za klasterizaciju nije zadovoljen ili se centri klastera ne mijenjaju **do**
5: Izračunati funkciju cilja, definisanu u (2.31);
6: Dodijeliti svaku tačku podataka klasteru koji je najbliži;
7: Ažurirarati centre klastera, definisan u 2.32 ;
8: **end while**

Ovaj postupak počinje s ručnim unosom broja klastera. Početne srednje vrijednosti klastera (centroidi) se nasumično postavljaju, što može uticati na broj iteracija potrebnih za postizanje stabilnosti (konvergencije). Zatim se svaki podatak dodijeljuje najbližem centroidu. Nakon dodjele svih podataka, centroidi se ponovo izračunavaju na osnovu trenutne raspodjele podataka. Ovaj proces se ponavlja dok se dodjele podataka ne stabilizuju, odnosno dok se ne prestanu mijenjati. Na taj način algoritam konvergira ka minimumu prosječne udaljenosti unutar klastera [28].

**Primjer 1.** *U nastavku je dat primjer koji ilustruje proces K-means algoritma na dvodimenzionalnim podacima. Neka je cilj grupisati podatke u 3 klastera.*

*Podaci su dati u tabeli 2.2:*

| Tačka | $x$ | $y$ |
|---|---|---|
| A | 1 | 2 |
| B | 2 | 1 |
| C | 3 | 4 |
| D | 5 | 7 |
| E | 3 | 3 |
| F | 6 | 5 |
| G | 8 | 9 |
| H | 7 | 8 |

**Tabela 2.2:** Podaci u $(x,y)$ koordinatama

*Izaberemo 3 početna klaster centra. U ovom primjeru, inicijalizujemo sa tačkama A, D i G kao centroidima. Centroidi su:*

$$C_1 = (1,2), \quad C_2 = (5,7), \quad C_3 = (8,9)$$

*Računamo udaljenost svake tačke od svih centara klastera koristeći Euklidsku distancu (tabela 2.3).*

| Tačka | $C_1(1,2)$ | $C_2(5,7)$ | $C_3(8,9)$ | Najbliži Centroid |
|---|---|---|---|---|
| A | 1.35 | 6.02 | 9.19 | $C_1$ |
| B | 1.52 | 6.10 | 9.30 | $C_1$ |
| C | 1.68 | 3.20 | 6.36 | $C_1$ |
| D | 5.27 | 1.12 | 2.92 | $C_2$ |
| E | 0.90 | 3.91 | 7.11 | $C_1$ |
| F | 4.51 | 1.12 | 3.81 | $C_2$ |
| G | 8.68 | 3.91 | 0.71 | $C_3$ |
| H | 7.27 | 2.50 | 0.71 | $C_3$ |

**Tabela 2.3:** Udaljenosti od centroida

*Novi centroidi:*

$$C_1 = \left(\frac{1+2+3+3}{4}, \frac{2+1+4+3}{4}\right) = (2.25, 2.5)$$

$$C_2 = \left(\frac{5+6}{2}, \frac{7+5}{2}\right) = (5.5, 6.0)$$

$$C_3 = \left(\frac{8+7}{2}, \frac{9+8}{2}\right) = (7.5, 8.5)$$

*Računamo udaljenost svake tačke od novih centara klastera (tabela 2.4):*

| Tačka | $C_1(2.25, 2.5)$ | $C_2(5.5, 6.0)$ | $C_3(7.5, 8.5)$ | Najbliži Centroid |
|---|---|---|---|---|
| A | 1.25 | 6.99 | 9.59 | $C_1$ |
| B | 1.80 | 6.5 | 10.10 | $C_1$ |
| C | 1.80 | 4.5 | 7.10 | $C_1$ |
| D | 5.92 | 1.11 | 3.61 | $C_2$ |
| E | 1.03 | 4.30 | 7.70 | $C_1$ |
| F | 4.90 | 1.11 | 5.0 | $C_2$ |
| G | 8.10 | 3.61 | 1.11 | $C_3$ |
| H | 7.30 | 2.83 | 0.71 | $C_3$ |

**Tabela 2.4:** Udaljenost od novih centroida

*Konačni klasteri dati su u tabeli 2.5:*

| Klaster | Tačke | Centroid |
|---|---|---|
| $C_1$ | A, B, C, E | (2.25, 2.5) |
| $C_2$ | D, F | (5.5, 6.0) |
| $C_3$ | G, H | (7.5, 8.5) |

**Tabela 2.5:** Konačni klasteri

*Kroz iterativno ažuriranje centroida i dodjelu tačaka, K-means uspješno grupiše podatke u klastere sa minimalnim varijacijama unutar klastera.*

K-means algoritam je jednostavan za implementaciju, ali ima određene nedostatke. Kvalitet konačnih rezultata klasterizacije zavisi od početnog izbora centroida, koji se često bira nasumično. Različiti početni centroidi mogu dovesti do različitih rezultata i uticati na brzinu konvergencije. Stoga je pažljiv izbor početnih centara ključan za postizanje željene segmentacije. Pored toga, K-means algoritam često pronalazi lokalne, umjesto globalnih optimuma. Zbog ovih razloga, algoritam se često nadograđuje kombinovanjem sa metaheurističkim algoritmima, što se pokazalo kao vrlo uspješna metoda za poboljšanje performansi i rezultata segmentacije.

## 2.2 Evaluacija performansi segmentirane slike

Procijeniti kvalitet segmentirane slike golim okom je vrlo teško, jer se subjektivni osjećaj često pokazuje kao nepouzdan i podložan greškama. Subjektivna procjena može varirati između različitih osoba i zavisi od mnogih faktora, kao što su iskustvo i percepcija ocjenjivača. Stoga, postoji potreba za objektivnim, kvantitativnim metrikama koje mogu pružiti tačnu procjenu performansi segmentacije.

S obzirom na veliki broj metoda segmentacije koje su razvijene, pojavile su se i brojne metrike za procjenu kvaliteta segmentacije. Ove metrike omogućavaju poređenje različitih algoritama segmentacije i pomažu u izboru najprikladnijeg pristupa za određeni zadatak. Korišćenje odgovarajućih metrika zavisi

od specifičnog metoda segmentacije koji se koristi kao i ciljeva analize, jer na primjer, u medicinskoj slici, preciznost i pouzdanost segmentacije mogu biti od vitalnog značaja za dijagnozu. Stoga, postoji potreba za objektivnim, kvantitativnim metrikama koje mogu pružiti tačnu procjenu performansi segmentacije. Postoje dvije osnovne vrste evaluacije: *nadgledana* i *nenadgledana*.

### 2.2.1 Nadgledana evaluacija

Nadgledane metrike pružaju kvantitativnu mjeru razlike između segmentirane slike i očekivanog rezultata segmentacije (engl. *ground truth*) [30]. Očekivani rezultat može biti unaprijed poznat ili mora biti definisan od strane stručnjaka, kao na primjer od strane doktora, u slučaju medicinskih slika. Međutim ovdje se javlja problem objektivnosti i varijabilnosti među stručnjacima.

Nakon očekivanog rezultata, definišu se mjere razlike između segmentirane slike i očekivanog rezultata. Ove mjere kvantifikuju tačnost klasifikacije objekata u slici. Što je stopa tačno klasifikovanih objekata veća, to je kvalitetniji metod odnosno algoritam segmentacije [30].

Koristeći pomenute metrike, možemo tačno odrediti koliko dobro algoritam segmentira određene djelove slike u poređenju sa očekivanim rezultatom segmentacije. Ova kvantitativna procjena je ključna za razvoj algoritama koji pružaju pouzdane rezultate u različitim praktičnim primjenama. U nastavku su date neke od poznatih nadgledanih metrika [11].

#### *Matrica konfuzije*

- prikazuje broj tačno klasifikovanih (*True Positive, TP; True Negative, TN*) i pogrešno klasifikovanih (*False Positive, FP; False Negative, FN*) podataka 2.6.

| | **Predviđeno pozitivno** | **Predviđeno negativno** |
|---|---|---|
| **Tačno pozitivno** | TP | FN |
| **Tačno negativno** | FP | TN |

**Tabela 2.6:** Matrica konfuzije

Na osnovu matrice konfuzije, mogu se izračunati sledeće metrike:

$$\text{Preciznost} = \frac{TP}{TP+FP}, \tag{2.33}$$

$$\text{Odziv} = \frac{TP}{TP+FN}, \tag{2.34}$$

$$\text{Tačnost} = \frac{TP+TN}{TP+TN+FP+FN}. \tag{2.35}$$

#### *Presjek nad unijom* (engl. *Intersection over Union - IoU*)

- IoU mjeri koliko se poklapaju segmentirani dio slike $X$ i očekivanog rezultata $Y$ u odnosu na sve piksele koji su prisutni u oba segmenta zajedno:

$$\text{IoU} = \frac{|X \cap Y|}{|X| + |Y|}.$$

- sličan IoU je **Dice koeficijent**, koji je definisan kao dvostruki broj zajedničkih piksela podijeljen sa ukupnim brojem piksela u $X$ i $Y$:

$$\mathrm{DC} = \frac{2|X \cap Y|}{|X| + |Y|}.$$

***PRI* (engl. *Probability Rand Index*)** mjeri usklađenost oznaka piksela između segmentirane slike i očekivanog rezultata. Izračunava udio parova piksela koji su konzistentno označeni kao isti ili različiti u oba segmenta, pružajući mjeru konzistentnosti između njih:

$$\mathrm{PRI} = \frac{a+b}{a+b+c+d},$$

gdje su:

- $a$ – broj parova piksela koji su ispravno označeni kao isti u oba segmenta;
- $b$ – broj parova piksela koji su ispravno označeni kao različiti u oba segmenta;
- $c$ i $d$ – broj parova piksela koji su pogrešno označeni kao isti ili različiti.

### 2.2.2 Nenadgledana evaluacija

Nenadgledani kriterijumi evaluacije omogućavaju kvantifikaciju kvaliteta rezultata segmentacije bez potrebe za unaprijed poznatim podacima ili očekivanog rezultata. Ovo je posebno korisno u situacijama kada očekivani rezultat nije dostupan ili ga je teško definisati [30].

Ovi kriterijumi se oslanjaju na unutrašnje karakteristike segmentirane slike kako bi procijenili njen kvalitet, kao što su *standardna devijacija*, *varijansa* i *kontrast regije*.

Standardna devijacija i varijansa mjere raspršenost ili varijaciju nivoa sive unutar određene regije slike, pri čemu manja vrijednost ukazuje na uniformniju regiju, što je poželjno za određene tipove segmentacije. Takođe, varijansa, mjeri kvadrat odstupanja svakog piksela od srednje vrijednosti unutar regije; manja varijansa znači veću homogenost regije, dok veća varijansa može ukazivati na prisustvo teksture ili neujednačenosti unutar segmentirane regije. Kontrast regije, koji mjeri razliku u nivou sive između različitih regija, ukazuje na efikasniju segmentaciju kada je viši, jer obično znači bolje odvojene regije.

Međutim, korišćenje ovih kriterijuma za procjenu segmentacije teksturiranih slika može nositi određeni rizik. Slike često sadrže područja sa visokim stepenom varijacije unutar regija, što može dovesti do visokih vrijednosti standardne devijacije i varijanse čak i za ispravno segmentirane regije. Zbog toga se mogu pojaviti pogrešni zaključci o kvalitetu segmentacije.

U nastavku je dato nekoliko mjera zasnovanih da prethodnim karakteristikama.

#### SSIM (engl. *Structural Similarity Index*)

SSIM mjeri sličnost između dvije slike uzimajući u obzir sljedeće parametre:

- $\mu_I$ i $\mu_S$ – srednje vrijednosti originalne i segmentirane slike,
- $\sigma_I^2$ i $\sigma_S^2$ – varijanse originalne i segmentirane slike,
- $\sigma_{IS}$ – kovarijansa između originalne i segmentirane slike,
- $c_1$ i $c_2$ – konstante za stabilizaciju,

$$\text{SSIM} = \frac{(2\mu_I\mu_S + c_1)(2\sigma_{IS} + c_2)}{(\mu_I^2 + \mu_S^2 + c_1)(\sigma_I^2 + \sigma_S^2 + c_2)}. \tag{2.36}$$

Vrijednost SSIM je između 0 i 1, i sto je veća to je kvalitet segmentacije bolji.

### NCC (engl. *Normalized Cross-Correlation* )

NCC je tehnika koja se koristi za poređenje sličnosti između dvije slike ili djelova slike. Ona kvantifikuje koliko su dva segmenta slike slična, ali prije toga se segmentirana slika i orginalna slika normalizuju kako bi se eliminisale razlike u osvijetljenju i kontrastu. NCC se koristi u mnogim aplikacijama za obradu slike, uključujući detekciju defekata, praćenje objekata i poređenje šablona. Parametri uključuju:

- $I(m,n)$ – vrijednost piksela originalne slike,
- $S(m,n)$ – vrijednost piksela segmentirane slike
- $\mu_I$ i $\mu_S$ – srednje vrijednosti originalne i segmentirane slike,
- $\sigma_I$ i $\sigma_S$ – standardne devijacije originalne i segmentirane slike,

$$\text{NCC} = \frac{1}{k}\sum_{m,n}\frac{(I(m,n) - \mu_I)(S(m,n) - \mu_S)}{\sigma_I\sigma_S}.$$

### Maksimalni odnos signal-šum (PSNR)

PSNR definsan na sljedeći način:

$$\text{PSNR} = 20 \cdot \log_{10}\left(\frac{255}{\text{RMSE}}\right). \tag{2.37}$$

RMSE (srednja kvadratna greška) se izračunava kao:

$$\text{RMSE} = \sqrt{\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}(I(m,n) - \text{S}(m,n))^2},$$

gdje:

- $I(m,n)$ predstavlja vrijednost sivog piksela originalne slike,
- $\text{S}(m,n)$ predstavlja vrijednost sivog piksela segmentirane slike,
- $M \times N$ predstavlja veličinu slike.

PSNR ocijenjuje sličnost orginalne i segmentirane slike, i što je veća PSNR vrijednost, bolji je kvalitet segmentirane slike. Kriterijumi za procjenu kvaliteta segmentacije slike korišćenjem PSNR mogu se predstaviti na sljedeći način:

- $PSNR > 40dB$ – odličan kvalitet segmentirane slike,
- $30dB < PSNR < 40dB$ – dobar kvalitet slike, sa vidljivim ali prihvatljivim izobličenjima,
- $PSNR < 30dB$ – značajno vidljiva izobličenja, što ukazuje na loš kvalitet segmentacije [7].

### FSIM (engl. *Feature Similarity Index*)

FSIM koristi faznu kongruenciju (PC), koja je dimenzionalna mjera koja pokazuje značaj lokalne strukture. Parametri uključuju:

- $SL(x)$ – mjerilo sličnosti lokalnih struktura,
- $PC_m(x)$ – maksimalna fazna kongruencija na lokaciji $x$,
- $I$ – cjelokupna slika,

$$\text{FSIM} = \frac{\sum_{x \in I} SL(x) \cdot PC_m(x)}{\sum_{x \in I} PC_m(x)}. \tag{2.38}$$

# Poglavlje 3

# Metaheuristički optimizacioni algoritmi

## 3.1 Uvod u metaheurističke optimizacione algoritme

Metaheuristički optimizacioni algoritmi razvijeni su na način da se mogu primijeniti na širok spektar problema. Njihova fleksibilnost postiže se korišćenjem stohastičkih elemenata, što omogućava istraživanje većeg prostora rješenja i pomaže u pronalaženju globalnog optimuma, izbegavajući zaglavljivanje u lokalnim optimumima. Ovi algoritmi iterativno poboljšavaju rješenja tokom vremena, balansirajući između eksploatacije trenutno najboljih rješenja i istraživanja novih oblasti u prostoru rješenja. Metaheuristički algoritmi oponašaju prirodne fenomene [31]. Postoji nekoliko prirodnih fenomena koje ti algoritmi oponašaju, što je dovelo do razvoja različitih tipova algoritama a zasnovanih na:

- evoluciji,
- inteligenciji roja,
- fizičkim pojavama,
- ljudskim osobinama.

Algoritmi zasnovani na evoluciji oponašaju procese evolucije opisane Darvinovom teorijom, i zato ćemo se u ovim algoritmima sresti sa pojmovima kao što su *generacija, selekcija, rekombinacija i mutacija*. Ovi algoritmi su, kao i svi ostali metaheuristički, iterativni, i svaka iteracija predstavlja jednu generaciju. Generacija (populacija) predstavlja skup mogućih rješenja i unutar svake generacije vrše se procesi selekcije, rekombinacije i mutacije, kako bi se stvorila nova generacija rješenja. Selekcija odabira najbolja rješenja (roditelje) za stvaranje novih rješenja (potomaka). Ovaj proces preferira rješenja sa boljim vrijednostima funkcije cilja. Zatim se vrši proces rekombinacije odnosno kombinovanje karakteristika odabranih roditelja kako bi se generisala nova rješenja (potomci). Potom dolazi do mutacije odnosno nasumične promjene u nekom dijelu rješenja kako bi se povećala raznolikost populacije i omogućilo istraživanje novih rješenja koja nisu mogla biti dobijena rekombinacijom. Zatim se vrši izbor najboljih rješenja koja će preživjeti i formirati novu generaciju. Ovi procesi se ponavljaju kroz generacije, pri čemu algoritam iterativno poboljšava rješenja, sve dok se ne postigne zadovoljavajući nivo optimizacije [32]. Najpoznatiji algoritam iz ove grupe jeste Genetički algoritam (GA).

Algoritmi zasnovani na inteligenciji roja simuliraju ponašanja životinja koje se kreću u grupi ili roju, i to posmatrajući osobine njihovih kretanja i strategija pronalska hrane. Ptica, ili riba na primjer, ima svoju trenutnu poziciju i brzinu kretanja, i u svakom trenutku prilagođava svoj pravac i brzinu na osnovu *ličnog iskustva*, odnosno traži lokaciju gdje je već ranije pronašla hranu ili da istraži detaljnije, oslanjajući se na sopstveno iskustvo. Ptice takođe prate ponašanje svojih susjeda u jatu, odnosno prilagođavaju se na osnovu *globanog (društvenog) iskustva*. Ako neka ptica primijeti da druge ptice u jatu idu ka boljem izvoru hrane, ona će prilagoditi svoju putanju i brzinu kako bi pratila jato. Na taj način, ptice kombinuju lična i kolektivna znanja da bi se efikasno kretale prema najpovoljnijim lokacijama [33]. Na ovom principu razvijen jedan od najpopularnijh algoritama ove grupe, poznat kao Algoritam *optimizacije roja čestica*, odnosno PSO (engl. *Particle Swarm Optimization*) algoritam.

Još jedna grupa algoritama koja počiva na metodama fizičkih pojava, pokazala se kao veoma uspješna u zadacima optimizacije, a posebno algoritam koji se naziva *Simulirano kaljenje* odnosno SA (engl. *Simulated Annealing*) algoritam. Ovaj algoritam oponaša postupak kaljenja koji se sprovodi u metalurgiji. Ovaj proces se odvija na principu zagrijavanja i kontrolisang hlađenja materijala. Materijal, kao što je čelik, se zagrijava na veoma visokoj temperaturi, često blizu tačke topljenja jer na taj način atomi unutar materijala dobijaju mnogo energije, što im omogućava da se kreću slobodnije. Nakon što je materijal zagrijan, postepeno se hladi i atomi postepeno gube energiju i počinju da se organizuju u stabilnije konfiguracije i na kraju "zamrzavaju" u položajima koji formiraju kristalnu strukturu materijala.

U SA algoritmu, visoka početna temperatura omogućava rješenju (koje predstavlja skup parametara optimizacije) da slobodno istražuje prostor rješenja. Slično kao atomi u zagrijanom metalu, rješenje može preći u gotovo bilo koju poziciju, čak i ako je trenutna vrijednost funkcije cilja relativno loša. Kako algoritam napreduje, temperatura se postepeno smanjuje. Sa smanjenjem temperature, rješenje postaje "stabilnije" i manje sklono velikim promjenama. Ovaj korak odgovara hlađenju metala, gde se atomi postepeno smiruju u energetski povoljnijim pozicijama. Čak i kada algoritam naiđe na lošije rješenje (analogno energetskom skoku u atomima tokom hlađenja), postoji određena vjerovatnoća da će ga prihvatiti. To je slično tome kako se atomi u metalu, iako se uglavnom smiruju u stabilne pozicije, povremeno premještaju u manje stabilne konfiguracije, dok temperatura još nije pala dovoljno nisko.

Poslednja grupa algoritama oponaša neke intelignetne radnje ili osobine ljudi [31]. Na primjer, proces generisanja ideja i donošenje najboljeg zaključka odnosno rješenja je poznati BSO (engl. *Brainstorm Optimization*). Zatim, TBLO (engl. *Teaching Learning Based Optimization*) algoritam, opnaša proces učenja u učionici, gdje učitelj prenosi znanje učenicima. U ovom algoritmu, populacija rješenja se poboljšava kroz fazu nastave i fazu učenja. U fazi nastave se rješenja unapređuju uz "vođenje" najboljeg rješenja (učitelja). U fazi učenja se rješenja poboljšavaju kroz interakciju sa drugim rješenjima (učenicima).

Svaki od ovih algoritama ima svoje prednosti i ograničenja, pa su neki efikasniji za specifične vrste problema dok drugi bolje funkcionišu u drugačijim okolnostima. Zbog toga se pojavila ideja o kombinovanju sa metaheurističkih algoritama, s ciljem razvoja sveobuhvatnijeg i snažnijeg algoritma koji može pružiti tačnija rješenja za širi spektar problema ili za neki specifični problem.

## 3.2 PSO algoritam

PSO (Algoritam optimizacije roja čestica) nastao je 1995. godine, kada su socijalni psiholog *James Kennedy* i inženjer elektrotehnike *Russell Eberhart* razvili ovaj algoritam posmatrajući ponašanje prirodnih sistema. Inspiraciju su dobili iz načina na koji jata ptica i riba kao i rojevi insekata sarađuju u grupama kako bi pronašli hranu ili izbjegli predatore [33].

U matematičkom zapisu, jato se može predstaviti kao skup slučajno generisanih brojeva koji čine skup *swarm*. Svaka čestica u jatu zauzima određenu poziciju u prostoru, koja se može opisati pomoću koordinata. Na primjer, u dvodimenzionalnom prostoru, pozicija čestice $i$ u iteraciji $j$ može biti predstavljena vektorom $\mathbf{x}_i^j = (x_i^j, y_i^j)$. U $n$-dimenzionalnom prostoru, ovaj vektor se generalizuje na $\mathbf{x}_i^j = (x_i^j, y_i^j, z_i^j, \ldots)$.

Svaka čestica takođe ima pridruženu brzinu $\mathbf{v}_i$. Brzina se ažurira u svakoj iteraciji na osnovu ličnog iskustva (najbolje pozicije koju je čestica postigla) i globalnog iskustva (najbolje pozicije postignute u cijelom jatu). Ove brzine se ažuriraju prema sledećoj jednačini:

$$\mathbf{v}_i^{j+1} = \omega \mathbf{v}_i^j + c_1 r_1 (\mathbf{p}_i^j - \mathbf{x}_i^j) + c_2 r_2 (\mathbf{g}^j - \mathbf{x}_i^j), \tag{3.1}$$

gdje su:

- $\omega$ inercijski faktor, koji kontroliše uticaj prethodne brzine,
- $c_1$ i $c_2$ koeficijenti ubrzanja koji kontrolišu uticaj ličnog i globalnog iskustva,
- $r_1$ i $r_2$ slučajne vrijednosti između 0 i 1 koje uvode stohastički element u kretanje čestica.

Nakon što se izračuna nova brzina, pozicija svake čestice se ažurira na sljedeći način:

$$\mathbf{x}_i^{j+1} = \mathbf{x}_i^j + \mathbf{v}_i^{j+1}. \tag{3.2}$$

Svaka čestica takođe pamti svoju najbolju poziciju do sada, označenu kao $\mathbf{p}_i$, dok jato kao cjelina održava najbolju poziciju $\mathbf{g}(t)$. Ažuriranje pozicija zavisi od funkcije cilja $f(\mathbf{x})$. Ako nova pozicija daje bolji rezultat (manju vrijednost funkcije cilja), tada se najbolja lična pozicija ažurira. Inače, zadržava se stara pozicija:

$$\mathbf{p}_i^{j+1} = \begin{cases} \mathbf{x}_i^{j+1}, & \text{ako je } f(\mathbf{x}_i^{j+1}) < f(\mathbf{p}_i^j), \\ \mathbf{p}_i^j, & \text{inače .} \end{cases} \tag{3.3}$$

Na ovaj način, čestice kombinuju lična i kolektivna iskustva kako bi se efikasno kretale prema najpovoljnijim lokacijama. Pseudokod PSO algoritma dat je u 4.

**Algoritam 4** PSO algoritam

**Input:** Broj čestica $N$, broj iteracija $T$, funkcija cilja $f(\mathbf{x})$
**Output:** Najbolje pronađeno rješenje $\mathbf{g}^*$
**for** svaka čestica $i = 1$ do $N$ **do**
    Nasumično inicijalizovati poziciju $\mathbf{x}_i$ i brzinu $\mathbf{v}_i$
    Postaviti najbolju ličnu poziciju $\mathbf{p}_i \leftarrow \mathbf{x}_i$
**end for**
Postaviti najbolju globalnu poziciju $\mathbf{g}^* \leftarrow \arg\min_i f(\mathbf{p}_i)$
**for** svaka iteracija $t = 1$ do $T$ **do**
    **for** svaku česticu $i = 1$ do $N$ **do**
        Ažurirati brzinu čestice pomoću 3.1
        Ažurirati poziciju čestice pomoću 3.2
        Izračunati vrijednost funkcije cilja: $f(\mathbf{x}_i(t+1))$
        **if** $f(\mathbf{x}_i^{j+1}) < f(\mathbf{p}_i^j)$ **then**
            Ažurirati najbolju ličnu poziciju: $\mathbf{p}_i^{j+1} \leftarrow \mathbf{x}_i^{j+1}$
        **end if**
    **end for**
    Ažurirati najbolju globalnu poziciju:
    $\mathbf{g}^* \leftarrow \arg\min_i f(\mathbf{p}_i(^{j+1})$
**end for**
**return** $\mathbf{g}^*$

## 3.3 Algoritam Simuliranog Kaljenja

Simulirano kaljenje je algoritam inspirisan procesom termodinamičkog kaljenja iz statističke mehanike, zasnovan na Bolcmanovoj distribuciji koja opisuje vjerovatnoću da sistem zauzme određeno energetsko stanje. Vjerovatnoća da sistem bude u stanju sa energijom $e$ data je formulom:

$$P(e) = \exp\left(-\frac{e}{k_b T}\right),$$

gdje je $T$ temperatura, a $k_b$ Bolcmanova konstanta [34]. Kako temperatura opada, vjerovatnoća prelaska sistema u stanje više energije se smanjuje, ali nikada ne postaje nula, što omogućava algoritmu da izbjegne lokalni optimum i istražuje globalni prostor rješenja.

Algoritam simulira ovaj prirodni proces kako bi optimizovao funkcije, počevši od visokih vrijednosti temperature i smanjujući ih tokom vremena. Prvi algoritam ovog tipa bio je Metropolisov algoritam [34]. U ovom algoritmu se vjerovatnoća prelaska iz jednog energetskog stanja $e_1$ u drugo stanje $e_2$ računa kao:

$$P = \min\left(\exp\left(-\frac{e_2 - e_1}{k_b T}\right), 1\right).$$

U kontekstu SA algoritma, energija sistema odgovara vrijednosti funkcije cilja koju treba minimizirati. Tokom jedne iteracije, algoritam ima trenutno rješenje $x$ i vrijednost funkcije cilja $f(x)$, a novo rješenje $x_{\text{novo}}$ se bira nasumično iz susjedstva trenutnog rješenja. Ako novo rješenje smanji vrijednost funkcije cilja, automatski se prihvata. U suprotnom, prihvata se sa vjerovatnoćom:

$$P = \exp\left(-\frac{f(x_{\text{novo}}) - f(x)}{T}\right).$$

Ova osobina omogućava SA algoritmu da prihvati lošija rješenja, što mu pomaže da izbjegne zaobljavanje u lokalnim minimumima.

**Hlađenje** je ključni aspekt SA algoritma, koji se realizuje smanjenjem temperature tokom iteracija. Postoji nekoliko različitih strategija hlađenja, od kojih su najčešće:

1. **Eksponencijalno hlađenje**

$$T_k = T_0 \cdot \alpha^k, \tag{3.4}$$

   gdje je $T_0$ početna temperatura, $\alpha$ faktor hlađenja (obično između 0 i 1), a $k$ broj iteracija;

2. **Polinomsko hlađenje**

$$T_k = T_0 \left(1 - \frac{k}{k_{\max}}\right)^{\beta}, \tag{3.5}$$

   gdje je $\beta$ parametar (obično između 1 i 4) koji određuje brzinu hlađenja, a $k_{max}$ predstavlja maksimalan broj iteracija tokom kojih se proces hlađenja odvija.

Eksponencijalno hlađenje je češće u praksi jer omogućava brže opadanje temperature, dok polinomsko hlađenje omogućava algoritmu više vremena na nižim temperaturama.

Pored ovih standardnih metoda, u svom istraživanju [35], Hajek je izveo rigorozniju analizu procesa hlađenja kako bi osigurao konvergenciju algoritma ka globalnom minimumu. Hajek je predložio alternativno pravilo hlađenja:

$$T_k = \frac{\gamma}{\log(k+2)} \tag{3.6}$$

Ovdje je $\gamma$ konstanta koja zavisi od specifičnosti problema, i mora biti dovoljno velika da omogući algoritmu da "izađe" iz regiona lokalnih minimuma koji nisu globalni. Parametar $k$ predstavlja broj iteracija i uzima vrijednosti iz skupa ne-negativnih cijelih brojeva $k = 0, 1, 2, \ldots$.

U generalnom smislu, kako temperatura opada, vjerovatnoća prihvatanja lošijih rješenja se smanjuje, što dovodi do postepenog stabilizovanja algoritma na optimalnom ili blizu optimalnom rješenju.

Na kraju, kvalitet rješenja zavisi od početne temperature $T_0$, faktora hlađenja $\alpha$ i broja iteracija, pa je važno pažljivo postaviti ove parametre u zavisnosti od specifičnosti problema. Za određivanje početne temperature može se koristiti formula:

$$T_0 = \frac{\Delta E}{P_S/2},$$

gdje je $\Delta E$ razlika između najveće i najmanje vrijednosti funkcije cilja u prvoj generaciji rješenja, a $P_S$ veličinu populacije, odnosno broj članova roja (engl. *population size*) [36].

# Poglavlje 4

# Poboljšani K-means algoritam baziran na PSO-SA-Levy algoritmu

## 4.1 Nasumično kretanje

Nasumično kretanje (engl. *random walk*) može se definisati kao proces u kojem se položaj čestice, tačke ili objekta nasumično mijenja u vremenu. Ovo kretanje može se uporediti sa kretanjem pijanog čovjeka koji u svakom koraku ima nepredvidljiv pravac i smjer kretanja. Gledano s matematičke strane, ako sa $S_N$ označimo sumu $N$ uzastopnih nasumičnih koraka $X_i$, tada se $S_N$ može definisati na sljedeći način:

$$S_N = \sum_{i=1}^{N} X_i = X_1 + X_2 +, ..., X_N, \tag{4.1}$$

gdje je $X_i$ nasumični korak, generisan izabranom distribucijom [37]. Još prostiji način da opišemo nasumično kretanje jeste:

$$S_{i+1} = S_i + \omega_i, \tag{4.2}$$

gdje je $S_i$ trenutna lokacija u $i$-tom trenutku, a $\omega_i$ korak, ili nasumično generisana varijabla određenom distribucijom. U ovom slučaju $S_1$ je početna lokacija objekta ili čestice, koja može biti unaprijed zadana ili nasumično odabrana, zavisno od konteksta problema. Pomenuti korak $\omega$ nema i ne mora da ima uvjek istu vrijednost, već u svakom trenutku, odnosno iteraciji, varira. Ukoliko ovaj korak varira prema definiciji Gausove distribucije:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

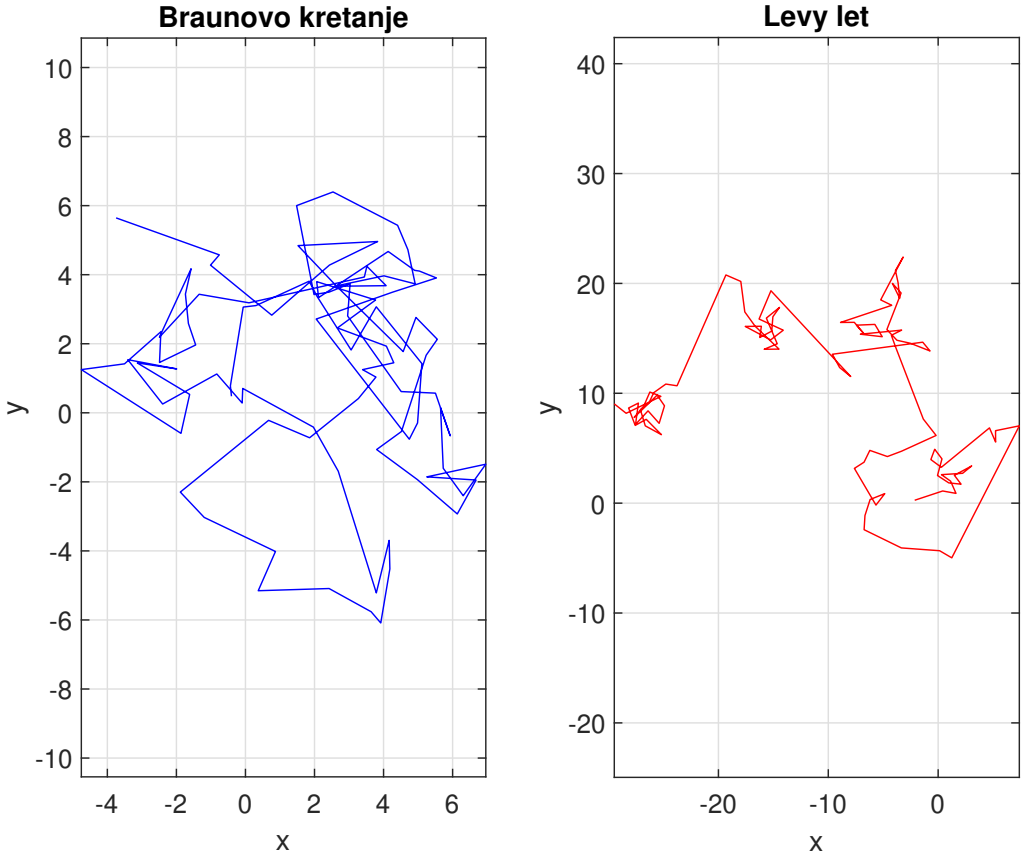
gdje je:

- $\mu$ – srednja vrijednost (engl. *mean*),
- $\sigma$ – standardna devijacija,
- $x$ - varijabla,

onda se takvo kretnaje naziva *Braunovo kretanje*. Gausova distribucija se takođe naziva normalna dis-

tribucija i označava se kao $\mathcal{N}(\mu, \sigma^2)$. Kada su $\mu = 0$ i $\sigma = 1$, to predstavlja specijalan slučaj, i naziva se *standardna normalna distribucija*, u oznaci $\mathcal{N}(0,1)$.

Iako je Gausova distribucija moćan alat za opisivanje nasumičnih procesa, ona ima svoja ograničenja. U ovom modelu, varijansa je konačna, što znači da su ekstremno dugi koraci u prostoru malo vjerovatni. To ograničava sposobnost ovog modela da opiše procese koji uključuju velike, neočekivane promjene. Za modeliranje nasumičnih procesa koji uključuju veće promjene i nepredvidljive skokove koristi se *Levy distribucija*. Ona spada u grupu distribucija sa "teškim repovima", što znači da vjerovatnoća većih koraka ne opada eksponencijalno kao kod Gauss-ove distribucije, već polinomijalno. Levy let je proces u kojem se koraci nasumičnog hoda biraju prema Levy distribuciji, omogućavajući modeliranje složenih pojava sa naglim promjenama, kao što je prikazano na slici 4.1.




**Slika 4.1:** Braunovo kretanje i Levy let

Furijeova transformacija $N$ slučajnih nezavisnih varijabli Levy distribucije ima sljedeći oblik:

$$F_N(k) = e^{-N|k|^\beta}. \tag{4.3}$$

U formuli (4.3) $k$ predstavlja frekvencijsku varijablu, dok je $\beta$ eksponent koji opisuje osobine Levy distribucije. Vrijednost $\beta$ obično leži između 0 i 2, gdje veće vrijednosti $\beta$ označavaju brže opadanje distribucije i veću koncentraciju oko srednje vrijednosti. Vrijednosti u rasponu $1 < \beta \leq 2$ posebno su relevantne za stabilne distribucije s beskonačnom varijansom, što je karakteristično za Levy procese koji modeluju fenomen poput Levy leta.

Inverzno, dobijanje Levy distribucije $L(s)$, gdje $s$ označava udaljenost ili pomjeraj u prostoru koji želimo opisati ovom distribucijom, nije jednostavno jer integral

$$L(s) = \frac{1}{\pi}\int_0^\infty \cos(\tau s)e^{-\alpha\tau^\beta}d\tau, \qquad 0 < \beta \leq 2, \tag{4.4}$$

nema svoju analitičku formu, osim u nekoliko specijalnih slučajeva. Za većinu slučajeva, radi jednostavnosti, $\alpha$ ima vrijednost 1. Za slučaj kada je $\beta = 1$, onda (4.4) postaje Košijeva distribucija, dok za $\beta = 2$ postaje Gausova odnosno normalna distribucija, gdje Levy let postaje ustvari Braunovo kretanje.

U generalnom slučaju, inverzni integral 4.4, možemo izraziti kao asimptotski niz, a njegova aproksimacija prvog reda za dužinu leta jeste:

$$L(s) = |s|^{-1-\beta}, \quad 0 < \beta < 2 \tag{4.5}$$

što je ustvari čini distribucijom s "teškim repom".

Matematičkim jezikom, prosta formula za Levy distribucije se može predstavti na sljedeći način:

$$L(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}}\frac{e^{-\frac{\gamma}{2(s-\mu)}}}{(s-\mu)^{\frac{3}{2}}}, & 0 < s < \mu < \infty, \\ 0, & \text{inače}, \end{cases} \tag{4.6}$$

gdje

- $\mu > 0$ predstavlja minimalni korak u Levy letu što znači da vrijednosti koraka $s$ ne mogu biti manje od $\mu$, odnosno, distribucija definiše najmanji mogući korak koji čestica može napraviti,
- $\gamma$ je parametar skale koji određuje širinu distribucije, odnosno, koliko su veliki ili mali koraci. Veća vrijednost $\gamma$ povećava vjerovatnoću većih koraka, dok manja vrijednost $\gamma$ daje veću vjerovatnoću manjih koraka.

Sa stanovišta implementacije, generisanje slučajnih brojeva sa Levy letovima obuhvata dva osnovna koraka: izbor nasumičnog pravca i generisanje koraka koji odgovaraju izabranoj Levy distribuciji. Prvi korak uključuje izbor pravca, koji se uzima iz uniforme distribucije, dok je generisanje koraka složeniji zadatak. Postoji nekoliko metoda za postizanje ovog cilja, a jedna od najučinkovitijih i jednostavnih metoda je upotreba Mantegninog algoritma za simetričnu Levy stabilnu distribuciju [37]. U ovom kontekstu, simetričnost označava da koraci mogu biti i pozitivni i negativni. U Mantegninom algoritmu, dužina koraka $s$ može se izračunati pomoću sledeće formule:

$$s = \frac{u}{|v|^{1/\beta}}, \tag{4.7}$$

gde su $u$ i $v$ slučajni brojevi izvedeni iz normalne distribucije:

$$u \sim \mathcal{N}(0, \sigma_u^2), \tag{4.8}$$

$$v \sim \mathcal{N}(0, \sigma_v^2). \tag{4.9}$$

Specifične vrijednosti za varijanse su:

$$\sigma_u = \left( \frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \beta/2)}{\Gamma((1+\beta)/2) \cdot \beta \cdot 2^{(\beta-1)/2}} \right)^{1/\beta}, \tag{4.10}$$

gdje je $\Gamma$ gama funkcija i definisana je kao $\Gamma(x) = (x-1)!$ za prirodne brojeve, i

$$\sigma_v = 1. \tag{4.11}$$

Ova distribucija za $s$ zadovoljava očekivanu Levy distribuciju za $|s| \geq |s_0|$, gde je $s_0$ najmanji korak. U teoriji, $|s_0|$ bi trebao biti nula, ali u praksi se $s_0$ može postaviti na neku fiksnu vrijednost iz opsega od $s_0 = 0.1$ do $s_0 = 1$ [9].

## 4.2 PSO-SA-Levy algoritam

PSO algoritam odlikuje se svojom jednostavnošću, koja mu omogućava visoku efikasnost, posebno kada je u pitanju brzina konvergencije. Ta jednostavnost pomaže PSO-u da brzo pronađe rješenja, naročito kod manje zahtjevnih optimizacionih problema, gdje čestice mogu lako pratiti najuspješnije pozicije u prostoru rješenja. Kada je riječ o globalnom optimumu, PSO je efikasan jer konstantno usmjerava čestice ka najboljim globalnim rješenjima, što omogućava bržu konvergenciju ka optimalnom rješenju u manje složenim okruženjima. Međutim, algoritam ima tendenciju stabilizacije u lokalnim minimumima, što može ograničiti njegovu sposobnost da pronađe globalno optimalno rješenje. Kao što znamo, sve čestice pored svog ličnog iskustva ($P_{best}$), uzimaju u obzir i globalnu, najbolju poziciju ($G_{best}$). Međutim, sve čestice, odnosno cijela populacija uzima u obzir ovaj parametar, što može dovesti do bliskog i brzog grupisanja oko ove pozicije. Nakon određenog broja iteracija, čestice postaju vrlo slične jedna drugoj i kreću se prema istom cilju. Njihove pozicije postaju vrlo bliske, čime se smanjuje sposobnost algoritma da istražuje nove, udaljene regije u prostoru pretrage. Čestice se mogu zaglaviti u lokalnim minimumima jer nisu u stanju da naprave značajnije skokove u druga, udaljena područja koja bi mogla sadržati bolje rješenja. Zbog navedenih nedostataka koje PSO algoritam sadrži, u nastavku ćemo ovaj algoritam nadograditi sa već nekim postojećim tehnikama koje se koriste kako bi se prevazišle pomenute nesigurnosti i izbjegli nedostaci.

Prvi metod kojim se mogu poboljšati performanse globalnog pretraživanja PSO algoritma predložen je u radu [38], i odnosi se na inercijski faktor koji kontroliše koliko čestica u trenutnoj iteraciji zadržava svoju prethodnu brzinu:

$$\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) \cdot k}{N}, \tag{4.12}$$

gdje su:

- $k$ – trenutna iteracija algoritma,
- $N$ – ukupan broj iteracija,
- $\omega_{\max}$– početna vrijednost inercijskog faktora, koja omogućava veće kretanje čestica na početku,

- $\omega_{\min}$ – krajnja vrijednost faktora, koja smanjuje brzinu kretanja čestica kako se približavaju rješenju.

Kada je inercijski faktor veliki, čestice imaju tendenciju da se kreću većim brzinama i prave veće korake, čime istražuju veći dio prostora pretrage. Kada je inercijski faktor mali, čestice prave manje korake i fokusiraju se na lokalnu pretragu oko trenutnog najboljeg rješenja. To pomaže u finom podešavanju rješenja i pronalaženju boljih rješenja unutar određenog prostora. Na početku pretrage, potrebna je veća vrijednost $\omega$ kako bi se obezbijedila dobra globalna pretraga. Kako se algoritam približava kraju (u kasnijim iteracijama), $\omega$ se smanjuje kako bi se fokus prebacio na lokalnu pretragu.

Drugi predloženi aspekat hibridizacije uključuje primjenu Levy distribucije odnosno osobine dugih nasumičnih koraka [39]. U višedimenzionalnim optimizacijskim problemima, PSO algoritam često zna biti problematičan. Kako broj dimenzija raste, broj lokalnih minimuma u prostoru pretrage obično eksponencijalno raste. PSO se oslanja na iterativno ažuriranje položaja čestica na osnovu njihove brzine i pozicije, ali kada je u pitanju višedimenzionalni problem, čestice ne mogu lako da razlikuju lokalni optimum od globalnog, jer im je "vidljivost" u prostoru ograničena. Zbog toga se čestice mogu zaglaviti u lokalnom optimumu, jer nemaju dovoljno informacija o širem prostoru. Levy distribucija generiše korake različitih dužina, pri čemu povremeno dolazi do velikih skokova. Ovi skokovi mogu omogućiti česticama da "iskoče" iz lokalnog optimuma i istraže nova rješenja koja PSO inače ne bi mogao da pronađe. Levy let uvećava diverzitet čestica jer čestice mogu praviti neočekivano velike korake.

Da bi PSO algoritam dobio prethodno opisano poboljšanje, potrebno je na poziciju 3.2, dodati još korak Levy distribucije 4.7, kao što je već urađeno u radu [39] :

$$\mathbf{x}_i^{j+1} = \mathbf{x}_i^j + \mathbf{v}_i^{j+1} + k, \tag{4.13}$$

gdje su:

- $\mathbf{x}_i^{j+1}$ – nova pozicija čestice $i$ u iteraciji $j+1$,
- $\mathbf{x}_i^j$ – trenutna pozicija čestice $i$ u iteraciji $j$,
- $\mathbf{v}_i^{j+1}$ – nova brzina čestice $i$ u iteraciji $j+1$,
- $k = 0.01 \cdot s$ – dodatni Levy korak, gdje je $s$ nasumična vrijednost dobijena Levy distribucijom, definisana u formuli 4.7.

Još jedan metod koji će nam u velikoj mjeri osigurati efikasnost PSO algoritma jeste hibridizacija sa Algoritmom simuliranog kaljenja. SA algoritam uvodi sporiji proces konvergencije, održavajući mogućnost da čestice istražuju nova rješenja čak i nakon velikog broja iteracija. Temperatura u SA kontroliše vjerovatnoću prihvatanja lošijih rješenja, čime se omogućava održavanje diverziteta čestica. Algoritam simuliranog kaljenja se primjenjuje u PSO algoritmu, ukoliko $G_{best}$, odnosno globalna pozicija stagnira, odnosno nema primijećenog poboljšanja $G_{best}$ nakon $K$ iteracija [8].

Na osnovu prethodnog teorijskog objašnjenja hibridni PSO-SA-Levy algoritma prikazan je u 5.

**Algoritam 5** PSO-SA-Levy algoritam

Inicijalizacija: $N$, $MaxIter$, $D$, $Lb$, $Ub$ ▷ N – Broj čestica, D – dimenzionalnost problema
PSO parametri: $c_1, c_2, \omega_{\max}, \omega_{\min}, \alpha_{\text{adaptive}}, k$
SA parametri: $K$, $T$, $T_{\min}$ ▷ $T$–početna temperatura, $T_{min}$–minimalna temperatura
Inicijalizacija populacije: $X$, $V$, $P_{\text{best}}$, $G_{\text{best}}$ ▷ pozicija, brzina, lokalno i globalno najbolje rješenje
Definisanje funkcije cilja: $f_{\text{obj}}$
**Output** $G_{best}$
**while** $iter < MaxIter$ **do**
    Ažurirati $\omega$ sa (4.12)
    $\alpha_{\text{adaptive}} = \alpha_{\text{adaptive}}/(1+iter)$
    **for** j = 1 : N **do**
        **if** $rand < 0.7$ **then**
            **if** $rand < \alpha_{\text{adaptive}}$ **then**
                Ažurirati $V(j,:)$ sa (3.1), $X(j,:)$ sa (4.13))
            **end if**
        **else**
            Ažurirati $V(j,:)$ sa (3.1), $X(j,:)$ sa (3.2)
        **end if**
        Izračunati funkciju cilja $f_{\text{obj}}$ za trenutnu poziciju čestice
        **if** $f_{\text{obj}}(X(j,:)) < P_{\text{best}}(j)$ **then**
            Ažurirati lokalno najbolje rješenje $P_{\text{best}}(j) = f_{\text{obj}}(X(j,:))$
        **end if**
        **if** $P_{\text{best}}(j) < G_{\text{best}}$ **then**
            Ažurirati globalno najbolje rješenje $G_{\text{best}} = P_{\text{best}}(j)$
            k = 0
        **else**
            k = k + 1;;
        **end if**
    **end for**
    **if** $k$ == $K$ **then** ▷ Primijeni SA
        $c = f_{obj}(G_{best})$ ▷ Izračunati funkciju cilja trenutnog $G_{best}$
        **while** $(T > T_{\min})$ **do**
            susjed = generisanjeSusjeda($G_{\text{best}}, Lb, Ub, D$)
            $c_{\text{susjed}} = f_{\text{obj}}$(susjed)
            **if** prihvatanjeRješenja($c_{\text{susjed}}, c, T$) **then**
                sol = susjed
                $c = c_{\text{susjed}}$
            **end if**
            Ažurirati $T$ koristeći eksponencijalno hlađenje (jedn. (3.4))
        **end while**
        Ažurirati $G_{\text{best}}$
    **end if**
    iter = iter + 1
**end while**
**return** $G_{\text{best}}$

Prvo se inicijalizuju parametri za algoritme PSO i SA, zatim se inicijalizuje populacija čestica, pri čemu svaka čestica ima svoju poziciju, brzinu, najbolju ličnu poziciju ($P_{best}$) i globalno najbolje rješenje ($G_{best}$). Nakon toga, algoritam ulazi u glavnu petlju (*while* petlju) koja će se imati $MaxIter$ broj iteracija. U svakoj iteraciji ažurira se inercijalni faktor $\omega$. Zatim se provjerava uslov $rand < 0.7$. Funkcija *rand* je funkcija u programiranju koja generiše nasumični realan broj između 0 i 1, što znači da u uslovu $rand < 0.7$ postoji 70% vjerovatnoće da će algoritam isprobati novi korak koristeći Levy let. U preostalih 30% slučajeva, algoritam koristi samo standardne PSO formule za ažuriranje položaja i brzine čestica. Vrijednost 0.7 je eksperimentalno utvrđena kao optimalna za problem segmentacije slike. Takođe se koristi se dinamički parametar $\alpha_{adaptive}$ za primjenu Levy koraka. Okarakterisan je kao *dinamički* jer na taj način imamo kontrolu, odnosno možemo balansirati nad eksploracijom prostora i eksploatacijom optimalnih rješenja. Kada je $\alpha_{\text{adaptive}}$ visok, u početnim iteracijama, algoritam ima tendenciju da istražuje nova rješenja koristeći Levy let. Suprotno tome, kada je $\alpha_{\text{adaptive}}$ nizak, algoritam više koristi postojeće znanje koristeći komponente PSO. Generisanje Levy koraka se može izvesti pomoću funkcije 4.1.

**Funkcija 4.1** Levy let

1: **function** LEVYLET($D$)
2: $\beta \leftarrow 1.5$
3: $\sigma \leftarrow \left(\frac{\Gamma(1+\beta)\cdot\sin(\pi\cdot\beta/2)}{\Gamma((1+\beta)/2)\cdot\beta\cdot 2^{(\beta-1)/2}}\right)^{1/\beta}$
4: $u \leftarrow \texttt{randn}(1,D) \times \sigma$
5: $v \leftarrow \texttt{randn}(1,D)$
6: step $\leftarrow 0.01 \times u/|v|^{1/\beta}$
7: **return** step
8: **end function**

Funkcija *randn* je funkcija u programiranju koja generiše niz (u našem slučaju niz dužine $D$) slučajnih brojeva iz normalne (Gausove) distribucije sa srednjom vrijednošću 0 i standardnom devijacijom 1. Veličine $\sigma$, $u$, $v$ i step već definisane u (4.10), (4.8), (4.9) i (4.7) respektivno. U skladu sa zadovoljenim uslovom, pozicija $X$ i brzina $V$ j-te čestice iz skupa $N$ ažuriraju se u svim dimenzijama. Izraz : je čest u programiranju i označava cio red (ili kolonu) matrice. U našem slučaju, $X(j,:)$ i $V(j,:)$ predstavljaju vektore koji sadrže vrijednosti pozicije i brzine $j$-te čestice u svim dimenzijama problema, pri čemu dimenzionalnost problema označava varijabla $D$. Zatim se računa funkcija cilja za trenutnu poziciju čestice. Ovo je važno jer vrijednost funkcije cilja određuje da li je nova pozicija čestice bolja u odnosu na prethodne. Ako je trenutna vrijednost funkcije cilja bolja od prethodnog najboljeg rješenja čestice ($P_{\text{best}}$), ažuriramo lokalno najbolje rješenje i poziciju. Ako je novo lokalno najbolje rješenje bolje od trenutnog globalnog najboljeg ($G_{\text{best}}$), onda se globalno najbolje rješenje i pozicija ažuriraju. Ovaj korak omogućava da se čestice prilagođavaju ne samo na osnovu svog prethodnog iskustva, već i na osnovu cjelokupnog kolektivnog znanja populacije. Međutim, ako se globalno najbolje rješenje $G_{\text{best}}$ nije poboljšalo u ovoj iteraciji, varijabla $k$ se povećava za 1. Na taj način pratimo broj uzastopnih iteracija tokom kojih nije došlo do poboljšanja $G_{\text{best}}$. Ukoliko $G_{\text{best}}$ ostane nepromijenjen nakon $K$ uzastopnih iteracija, pokreće se SA algoritam kako bi se izbjegla lokalna optimalnost i istražile još novije regije pretraživačkog prostora. Počinje proces "hlađenja" (ili "kaljenja"), pri čemu se temperatura

$T$ postepeno smanjuje tokom iteracija dok ne dostigne minimalnu vrijednost $T_{\min}$. SA generiše susjedna rješenja. Ova susjedna rješenja se dobijaju perturbacijom trenutnog rješenja unutar definisanih granica pretraživačkog prostora. Perturbacija se ostvaruje dodavanjem nasumičnog Gausovog šuma trenutnom rješenju, pri čemu je opseg perturbacije proporcionalan veličini pretraživačkog prostora. Da bi se osigurala validnost rješenja, granice pretraživačkog prostora ($Lb$ i $Ub$) se koriste da ograniče susjedno rješenje na odgovarajuće prostor pretrage, kao što je prikazano u funkciji4.2.

**Funkcija 4.2** Generisanje susjednih rješenja

```
1: function GENERISANJESUSJEDA(G_best, Lb, Ub, D)
2:     perturbacija ← (Ub - Lb) × 0.5          ▷ Podešavanje opsega perturbacije
3:     susjed ← G_best + perturbacija × randn(1, D)
4:     susjed ← max(min(susjed, Ub), Lb)   ▷ Provjeravanje da li je rješenje unutar granica
5:     return susjed
6: end function
```

Funkcija cilja se izračunava za novo susjedno rješenje, $c_{\text{susjed}} = f_{\text{obj}}(\text{susjed})$. Zatim dolazimo do funkcije 4.3 koja ima zadatak da ispita da li prihvatiti novo rješenje. Ako je razlika između funkcije cilja susjednog i najboljeg globalnog rješenja, ($\delta$), manja ili jednaka nuli, funkcija odmah prihvata novo rješenje, jer je susjedno rješenje bolje ili isto kao trenutno. Ako je susjedno rješenje lošije, izračunava se vjerovatnoća prihvatanja prema formuli $\exp(-\delta/T)$. Ova vjerovatnoća zavisi od razlike $\delta$i temperature. Ukoliko je nasumično generisana vrijednost manja od izračunate vjerovatnoće, tada se lošije rješenje prihvata. U funkciji 4.3 exp je matematička funkcija koja označava eksponencijalnu funkciju, odnosno funkciju sa osnovom $e$, gdje je $e$ Eulerov broj. Na višim temperaturama ($T$), vrijednost $p$ je veća, što znači da se lošija rješenja lakše prihvataju. Ovo omogućava algoritmu da istražuje širi prostor rješenja i izbjegne zaglavljivanje u lokalnim optimumima. Kako temperatura opada, $p$ postaje manja, pa algoritam sve rjeđe prihvata lošija rješenja, čime se povećava fokus na pronalaženje boljih rješenja u blizini trenutnog rješenja.

**Funkcija 4.3** Funkcija prihvatanja rješenja

```
1: function PRIHVATIRJEŠENJE(c_susjed, c, T)
2:     δ ← c_susjed − c          ▷ Izračunaj razliku između trenutnog i susjednog rješenja
3:     if δ ≤ 0 then
4:         return true                ▷ Prihvati ako je susjedno rješenje bolje ili jednako
5:     else
6:         p ← exp(−δ/T)          ▷ Izračunaj vjerovatnoću za prihvatanje lošijeg rješenja
7:         a ← rand < p                ▷ Prihvati lošije rješenje sa vjerovatnoćom p
8:         return a
9:     end if
10: end function
```

Nakon primjene SA algoritma, dobija se novo potencijalno globalno najbolje rješenje ($G_{\text{best}}$). Algoritam zatim nastavlja sa radom sve do dostizanja maksimalnog broja iteracija $MaxIter$, nakon čega $G_{\text{best}}$ predstavlja optimalno, tj. globalno najbolje rješenje pronađeno tokom pretrage.

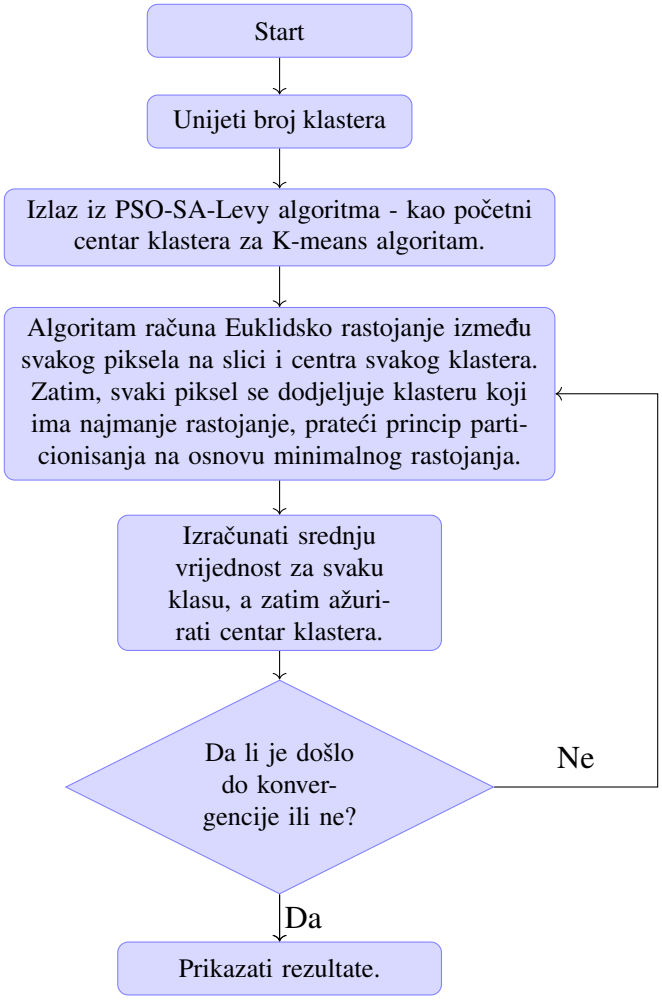
### 4.2.1 Poboljšani K-means algoritam

K-means algoritam je poznat po jednostavnosti implementacije i primjeni, uz nisku računsku složenost $O(NK)$, gdje je $N$ broj podataka, a $K$ broj klastera. Ova karakteristika omogućava njegovu primjenu na velike i raznovrsne skupove podataka, s brzim konvergiranjem ka rješenju, pogotovo ako su početni centri klastera pažljivo odabrani. Ipak, jedan od glavnih izazova K-means algoritma je osjetljivost na inicijalizaciju centara klastera. Loša inicijalizacija može dovesti do lokalnih minimuma i loših rezultata. Također, algoritam se loše ponaša u prisustvu šuma ili *outlier* -a, jer oni mogu narušiti proces formiranja klastera. Ovi nedostaci proizlaze iz toga što K-means nema mehanizam za globalnu pretragu prostora rješenja.

Da bi se ovi problemi prevazišli, koristićemo hibridni PSO-SA-Levy algoritam, koji nudi efikasniju globalnu pretragu prostora rješenja. PSO-SA-Levy koristi kombinaciju tehnika globalne optimizacije i to PSO algoritam, SA algoritam i Levy let – što poboljšava sposobnost algoritma da izbjegne lokalne minimume i pronađe bolja globalna rješenja. PSO-SA-Levy algoritam omogućava pronalaženje centara klastera kroz minimizaciju funkcije cilja, što omogućava rješavanje kompleksnijih problema optimizacije.

U PSO-SA-Levy algoritmu, svaka čestica predstavlja potencijalno rješenje, tj. skup centara klastera. Na primjeru segmentacije slike, i to slike sa tri kanala (RGB) i tri klastera, svaka čestica ima dimenziju $3 \times 3 = 9$. Čestice u jatu ažuriraju svoje pozicije na temelju prethodno deinisanog PSO-SA-Levy algoritma. Kroz iteracije, brzina svake čestice se prilagođava kako bi pratila najbolja rješenja, čime se postiže ravnoteža između istraživanja novog prostora i eksploatacije trenutno poznatih rješenja. Svaka čestica procjenjuje se putem funkcije cilja 2.31 koja mjeri kvalitet klasterisanja, odnosno minimizuje sumu kvadratnih odstupanja između piksela slike i centara klastera. Cilj je pronaći optimalni skup centara koji minimizuju ukupno odstupanje. Na kraju, čestica sa najboljim rješenjem predstavlja skup optimalnih centara klastera. Ti centri se zatim predaju K-means algoritmu, koji dalje fino podešava njihove pozicije tako što računa prosjek svih podataka unutar svakog klastera, osiguravajući bolju preciznost konačnog rješenja.

Poboljšani K-means algoritam je predstavljen na dijagramu 4.2.




**Slika 4.2:** Poboljšani K-means algoritam

# Poglavlje 5

# Testiranja i numerički rezultati

## 5.1 Testiranje performansi PSO-SA-Levy algoritma

U ovoj sekciji biće analizirane performanse predloženog PSO-SA-Levy algoritma kroz 10 poznatih testnih funkcija prikazanih u Tabeli 5.1. Ove funkcije su standardne u analizi performansi optimizacionih algoritama, jer omogućavaju procjenu sposobnosti algoritma da pronađe globalni optimum u kompleksnim pretraživačkim prostorima [40, 41].

Testne funkcije uključuju unimodalne funkcije (F1,F2,F3) i multimodalne funkcije (F4-F10). Multimodalne funkcije imaju veći broj lokalnih minimuma, što predstavlja izazov za algoritme optimizacije jer može dovesti do zaglavljivanja u lokalnim optimumima. S druge strane, unimodalne funkcije imaju samo jedan globalni optimum, što ih čini pogodnim za analizu sposobnosti algoritma da brzo konvergira prema globalnom rješenju.

Složenost ovih funkcija zavisi i od dimenzionalnosti prostora pretrage. U ovom istraživanju korišćene su dimenzije 2 i 20 , što omogućava detaljnu analizu efikasnosti predloženog algoritma. Fokus testiranja biće na poređenju predloženog algoritma sa drugim metaheurističkim algoritmima u pogledu pronalaska globalnog optimuma (minimuma) funkcije filja i srednje vrijednosti kao i standardne devijacije nakon većeg broj pokretanja algoritama.

| **Naziv** | **Definicija** | **Dim** | **Interval** | **Minimum** | **Klasifikacija** |
|---|---|---|---|---|---|
| Sphere | $F_1(x) = \sum_{i=1}^{n} x_i^2$ | 20 | [-100,100] | 0 | Unimodalna |
| Rosenbrock | $F_2(x) = \Sigma_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | 20 | [-30,30] | 0 | Unimodalna |
| Schwefel 2.22 | $F_3(x) = \sum_{i=1}^{n}(\lvert x_i\rvert) + \prod_{i=1}^{n}(\lvert x_i\rvert)$ | 20 | [-10,10] | 0 | Unimodalna |
| Booth | $F_4(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ | 2 | [-10, 10] | 0 | Multimodalna |
| Rastrigin | $F_5(x) = \Sigma_{i=1}^{n}[x_i^2 - 10\ cos(2\pi x_i) + 10]$ | 20 | [-5.12,5.12] | 0 | Multimodalna |
| Ackley | $F_6(x) = -20\ exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$ | 20 | [-32,32] | 0 | Multimodalna |
| Griewank | $F_7(x) = \frac{1}{4000}\Sigma_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$ | 20 | [-600,600] | 0 | Multimodalna |
| Shubert | $F_8(x) = \prod_{i=1}^{n}\left(\sum_{j=1}^{5} j \cdot \cos((j+1)x_i + j)\right)$ | 2 | [-10, 10] | -186.7309 | Multimodalna |
| Six-Hump Camel-back | $F_9(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$ | 2 | [-5, 5] | -1.0316 | Multimodalna |
| Powell | $F_{10}(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$ | 4 | [-5, 5] | 0 | Multimodalna |

**Tabela 5.1:** Informacije o funkcijama

### 5.1.1 Poređenje sa prvom grupom metaheurističkih metoda (PSO, SA, GA)

Ovdje će se predloženi algoritam uporediti sa tri metaheuristička algoritma: genetičkim algoritmom (GA), klasičnim PSO algoritmom i klasičnim SA algoritmom, čiji su rezultati preuzeti iz [42], gdje su optimizacione sposobnosti tih algoritama procijenjene nad funkcijama. Biće testirane funkcije $F1$–$F8$ i $F10$, pri čemu se upoređuju minimalne vrijednosti koje su algoritmi pronašli. Globalni minimumi ovih funkcija navedeni su u tabeli 5.1, u koloni *Minimum*. Predloženi algoritam postepeno smanjuje $\omega$ sa 1,2 na 0,2 tokom vremena (4.12). Takođe, algoritam koristi parametre preuzete iz rada [42], gdje su $c_1$

i $c_2$ postavljeni na 2.0, a koji kontrolišu uticaj $P_{best}$ i $G_{best}$ na kretanje čestica u procesu optimizacije. Maksimalni broj iteracija je 5000, a veličina roja je fiksirana na 50. Rezultat je prikazan u Tabeli 5.2.

| **Funkcija** | **PSO** | **SA** | **GA** | **PSO-SA-Levy** |
|---|---|---|---|---|
| F1 | 3.6927e-37 | 1.140e-7 | 1.068e-9 | **0** |
| F2 | 0.04051 | 0.0277 | 0.0136 | **2.9501e-06** |
| F3 | 0.4123 | 1.174e-4 | 4.470e-5 | **0** |
| F4 | 0 | 0 | 1.15840e-10 | 0 |
| F5 | 0.473e-8 | 3.304e-13 | 2.0232e-9 | **0** |
| F6 | 8.8818e-16 | 1.13631e-5 | 2.467370e-6 | 8.8818e-16 |
| F7 | 0.2323 | 0.0105 | 5.508e-10 | **0** |
| F8 | -186.7309 | -186.7309 | -186.7309 | -185.6208 |
| F10 | 0 | 1.93582e-6 | 0 | 0 |

**Tabela 5.2:** Rezultati prvog poređenja

Na osnovu rezultata iz tabele, možemo izvesti sljedeće zaključke u vezi sa performansama predloženog algoritma:

1. **Unimodalne funkcije**:
   - Za funkcije $F1$ i $F3$, predloženi algoritam uspješno identifikuje globalni minimum.
   - Iako globalni minimum nije pronađen za $F2$, rezultat algoritma je značajno bolji u odnosu na druge metode i blizu je globalnog minimuma.
2. **Multimodalne funkcije**:
   - Predloženi algoritam pronalazi globalni optimum za četiri funkcije: $F4$, $F5$, $F7$ i $F10$.
   - U slučaju funkcije $F6$, algoritam se ponaša uporedivo sa PSO.
   - Međutim, za funkciju $F8$, predloženi algoritam daje lošiji rezultat u poređenju sa drugim algoritmima.

Sveukupno, predloženi algoritam pokazuje obećavajuće performanse u pronalasku globalnog minimuma nad unimodalnim i multimodalnim funkcijama.

### 5.1.2 Poređenje sa drugom grupom metaheurističkih metoda (ACO, PSO, GA, GWO, HS)

U ovom dijelu biće ispitani predloženi algoritam i algoritmi iz [43], čiji su rezultati preuzeti i iskorišćeni za poređenje. Naime, radi se o Optimizaciji kolonije mrava (ACO), PSO, GA, optimizaciji sivih vukova (GWO) i pretrazi harmonije (HS), a poređenje će biti u smislu standardne devijacije (*STD*) i

prosječne vrijednosti dobijenih minimuma (*AVG*). Biće ispitane tri unimodalne funkcije ($F1$, $F2$ i $F3$) i tri multimodalne funkcije ($F5$, $F7$ i $F9$) definisane u Tabeli 5.1. Parametri predloženog algoritma su postavljeni na sljedeći način: parametri $c_1$ i $c_2$ postavljeni su na 1.7 i 1.5. Veličina roja je fiksirana na 50, a maksimalan broj iteracija sada 1000. Svi pomenuti parametri su takođe preuzeti iz rada [43]. Rezultati su prikazani u Tabeli 5.3.

Za testiranje, svaki algoritam je pokrenut više puta (100 puta). Na osnovu rezultata svih ponavljanja, izračunate su srednje vrijednosti i standardne devijacije za svaki algoritam i funkciju. Formule koje su korišćene za izračunavanje srednje vrijednosti i standardne devijacije su:

$$\text{AVG} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\text{STD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \text{AVG})^2}$$

gdje je $N$ broj ponavljanja (100 u ovom slučaju, kao što je definisano u radu [43]), $x_i$ predstavlja minimum funkcije cijla koje je algoritam pronašao u procesu pronalska globanog minimuma za $i$-ti pokušaj, dok je AVG srednja vrijednost rezultata. Standardna devijacija (STD) mjeri varijabilnost rezultata u odnosu na srednju vrijednost.

| **Funkcija** | **ACO** | **PSO** | **GA** | **GWO** | **HS** | **PSO-SA-Levy** |
|---|---|---|---|---|---|---|
| F1 | 0.4163 / 5.0949 | 0.0438 / 0.2089 | 1.1068 / 6.7035 | 0.9263 / 5.1402 | 1.5003 / 5.9403 | **0 / 0** |
| F2 | 0.3643 / 0.5221 | 0.4051 / 0.7441 | 1.9296 / 3.4089 | 0.9222 / 3.3771 | 1.0003 / 2.0042 | **0.0290 / 0.3093** |
| F3 | **0.0.0017 / 0.0032** | 0.4565 / 1.7417 | 0.0501 / 0.3962 | 0.0814 / 1.1011 | 0.0563 / 1.0271 | 0.4270 / 3.9324 |
| F5 | 0.0790 / 1.0966 | 0.4733 / 1.4273 | 0.0616 / 0.8088 | 0.4066 / 1.7537 | 0.0237 / 0.8423 | **0 / 0** |
| F7 | 0.7343 / 4.4431 | 14.5538 / 49.1234 | 2.9883 / 21.8681 | -5.7441 / 2.4104 | 1.6723 / 14.2884 | **1.9011e-04 / 0.0019** |
| F9 | – | 0.7982 / 1.9983 | 0.0004 / 0.0010 | 0.0003 / 0.0016 | – | **-3.7692e-04 / 0.0119** |

**Tabela 5.3:** Rezultati drugog poređenje u pogledu AVG/STD

Analizom Tabele 5.3 zaključujemo da predloženi algoritam postiže jako dobre rezultate u šest funkcija ($F1$ do $F6$) u pogledu srednje vrijednosti (AVG) i standardne devijacije (STD). Ovi rezultati nadmašuju one postignute drugim algoritmima, s jednom iznimkom: funkcijom $F3$. Dobra (uglavnom niska) srednja vrijednost znači da algoritam u prosjeku daje tačne i kvalitetne rezultate. Niska standardna devijacija ukazuje na to da su rezultati algoritma konzistentni kroz različita pokretanja. To znači da algoritam

daje slične rezultate svaki put kada se izvrši, što povećava pouzdanost algoritma. Međutim, mogu se primijetiti nešto lošije performanse kako predloženog, tako i PSO algoritma u poređenju sa prethodnim eksperimentom. Ove razlike mogu se objasniti promjenom parametara $c_1$ i $c_2$, kao i manjim brojem iteracija u drugom eksperimentu.

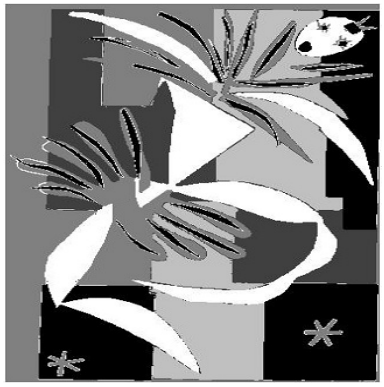
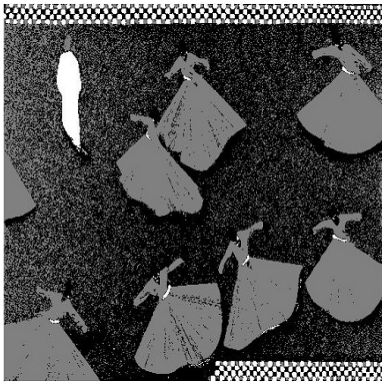
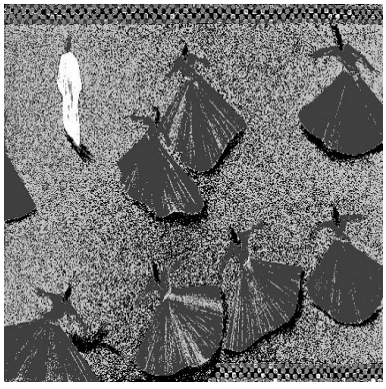
### 5.1.3 Ekesperimentalni rezultati segmentacije slike

Za naš eksperiment izabrane su umjetničke i apstraktne slike. K-means tehnika segmentacije može pomoći u izolaciji različitih regija ili objekata unutar umjetničke slike, što je korisno za analizu vizuelnih osjećaja koje izazivaju specifični elementi ili kompozicije u umjetničkom djelu. Odabrali smo skup od 10 umjetničkih slika sa WikiArt-a. WikiArt je sveobuhvatna online baza podataka koja pruža pristup velikoj kolekciji vizuelnih umjetničkih djela, uključujući slike i fotografije, iz različitih vremenskih perioda i umjetničkih pravaca [44]. Parametri predloženog algoritma su postavljeni na sljedeći način: veličina roja $Swarm\_size = 50$ i maksimalan broj iteracija $Max\_iter = 100$. Slike su segmentirane sa 3 i 5 klastera. Vizuelni rezultati su prikazani u tabeli 5.4 Ovaj eksperiment segmentacije sproveden je u Matlabu R2018b. Prikazan je u algoritmu 9, gdje su uključeni ključni dijelovi koda potrebni za razumijevanje postupka, dok je ostatak izostavljen zbog njegove obimnosti.

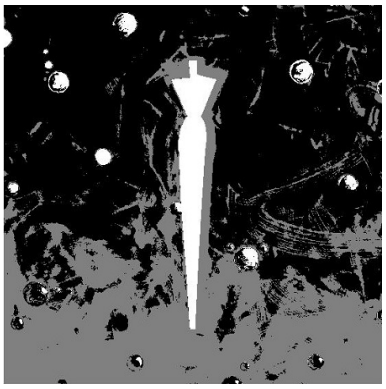
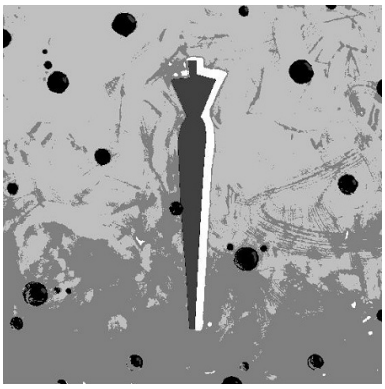
**Algoritam 9** Proces segmentacije slike pomoću K-means metode

1: **Ulaz:** Učitati sliku `im = imread('Slika.jpg')`
2: Konvertovati sliku: `im = im2double(im)`
3: Izdvojiti dimenzije slike: `[redovi, kolone, broj_kanala] = size(im)`
4: Inicijalizovati praznu matricu `imIn` za skladištenje vrijednosti piksela
5: **for** kanal $i = 1$ do $broj\_kanala$ **do**
6: Izdvojiti $i$-ti kanal slike
7: Dodati kanal u matricu `imIn(:, i)`
8: **end for**
9: Definisati broj klastera, na prmijer 3: `nc = 3`
10: Izračunati ukupni broj piksela: `L = redovi * kolone`
11: Nasumično izabrati N% piksela: `rns = (N/100) * L`
12: Nasumično permutovati piksele: `rp = randperm(L)`
13: Izvući podskup piksela: `sds = imIn(rp(1:rns), :)`
14: Postaviti parametre za PSO algoritam:
15: Definisati veličinu roja: `SwarmSize = 50`
16: Definisati maksimalan broj iteracija: `Max_iter = 100`
17: Postaviti donje granice: `Lb = zeros(1, nc * broj_kanala)`
18: Postaviti gornje granice: `Ub = ones(1, nc * broj_kanala)`
19: Postaviti dimenziju problema: `Dim = nc * broj_kanala`
20: Definisati funkciju cilja `fobj` na osnovu $X$ i $sds$
21: Pokrenuti mjerenje vremena: `tic`
22: Pokrenuti PSO-SA-Levy algoritam:
23: `clusters = PSO-SA-Levy(SwarmSize, Max_iter, Lb, Ub, Dim, fobj)`
24: Primjeniti K-means segmentaciju: `[idx, cs, egr] = kmeanseg(im, clusters)`
25: Izračunati PSNR: `[PSNR, MSE] = psnr(im, idx)`
26: Konvertovati sliku u sivu: `i1 = rgb2gray(im)`
27: Izračunati SSIM: `mssim = ssim(i1, idx)`
28: Zaustaviti mjerenje vremena: `toc`
29: Prikazati prosjek PSNR i SSIM
30: Prikazati originalnu i segmentiranu sliku

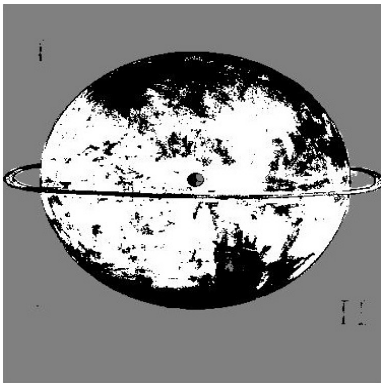
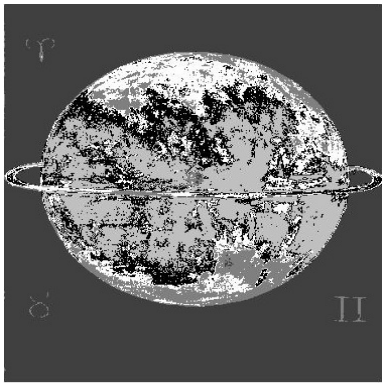
**Tabela 5.4:** Vizuelni rezultati segmentacije umjetničkih slika

| Br. | Originalna slika | Seg. slika sa 3 klastera | Seg. slika sa 5 klastera |
|---|---|---|---|
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
| 4. |  |  |  |

**5.**

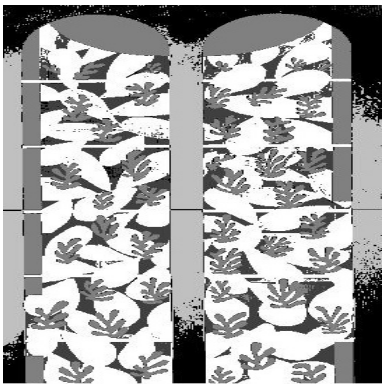




**6.**

**7.**



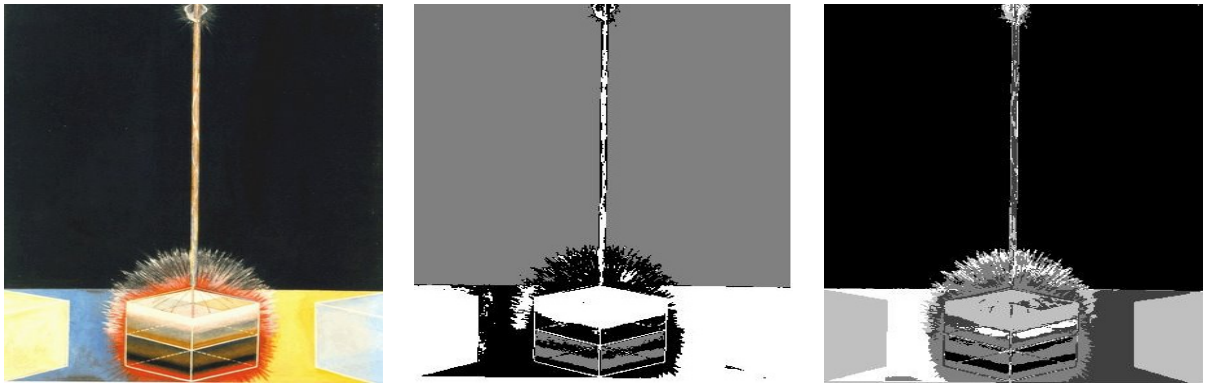


**8.**



**9.**

**10.**



Iako nije lako procijeniti kvalitet segmentacije golim okom, možemo primijetiti neke detalje segmentacije i razlike u zavisnosti od broja klastera. Ova kratka analiza fokusira se na slike segmentirane u tri klastera:

- proces segmentacije efikasno razlikuje centralni objekat od okoline, obezbjeđujući jasne granice omogućavajući nam da se fokusiramo na objekat bez smetnji od pozadine,
- detalji okoline i centralnog objekta su adekvatno očuvani,
- korišćenje tri klastera postiže ravnotežu između jednostavnosti i detalja, pojednostavljujući segmentaciju radi lakše interpretacije, dok istovremeno hvata suštinske karakteristike bez nepotrebne složenosti,
- ovaj pristup pokazuje uspješnu ravnotežu između jednostavnosti i detalja, čineći ga pogodnim za ovih 10 slika.

Dalje, slike segmentirane u pet klastera pružaju sljedeće karakteristike:

- segmentacija sa pet klastera nudi viši nivo detalja u poređenju sa verzijom sa tri klastera. Dijeljenjem slike na više klastera, algoritam identifikuje finije razlike između grupa piksela, na primjer slika 6,
- centralni objekti na svih 10 slika su jasno definisani. Njihove granice su oštrije, naglašavajući njegovu razliku u odnosu na pozadinu. Dodatni klasteri hvataju suptilne varijacije unutar objekta, poput sjenčenja ili složenih šara,

- pristup sa pet klastera jako dobro izražava složene detalje. Na primjer, na slikama 1 i 9, uhvaćeni su jako sitni složeni detalji. Ovaj nivo preciznosti može biti ključan u segmentaciji slika, pogotovo u medicinskim slikama,
- iako segmentacija sa pet klastera povećava detalje, postoji kompromis. Prekomjerna segmentacija se javlja kada manje varijacije ili šumovi stvaraju nepotrebne klastere (slika 8),
- za slike koje zahtijevaju preciznu ekstrakciju karakteristika segmentacija sa pet klastera ili više može biti korisna. Međutim, ako je jednostavnost i lakoća interpretacije ključna, verzija sa tri klastera može biti dovoljna.

Za dublju analizu kvaliteta segmentacije, koristiće se neke od uvedenih nenadgledanih metrika u 2.2.2. Koristiće se PSNR (2.37) i SSIM (2.36). Pored ove dvije metrike, takođe će biti uključeno i vrijeme izvršavanja algoritma za segmentaciju slike. Na ovaj način možemo dobiti dovoljno informacija o kvalitetu segmentacije i efikasnosti algoritma. Takođe, pored predloženog algoritma, na istim metrikama i slikama biće testirani algoritmi kao što su klasični K-means, ABC, DE, MRFO i GWO, a rezultati će poslužiti za komparativnu analizu ovih tehnika. Za potrebe ovog eksperimenta veličina roja postavljena je na 50, a maksimalni broj iteracija na 100, za sve algoritme. Numerički rezultati segmentacije prethodnih 10 slika, segmentiranih u 5 klastera su prikazani u tabeli 5.5. U tabeli su prikazane usrednjene vrijednosti razmatranih metrika za tih 10 slika.

| Br. | Metrika | K-means | ABC | DE | MRFO | GWO | PSO-SA-Levy |
|---|---|---|---|---|---|---|---|
| | SSIM | 0.0633 | 0.4930 | 0.5053 | 0.6054 | 0.4017 | **0.6651** |
| 1. | PSNR | 11.5108 | 38.1174 | 37.8762 | 39.9964 | 36.7129 | **41.6772** |
| | Vrijeme | **2.2064** | 71.2679 | 84.1081 | 131.5800 | 88.7403 | 71.1411 |
| | SSIM | 0.0201 | 0.6466 | 0.6291 | 0.6680 | 0.6137 | **0.6997** |
| 2. | PSNR | 4.0617 | 38.9159 | 38.9268 | 39.4521 | 38.1846 | **40.0415** |
| | Vrijeme | **2.2950** | 69.5274 | 54.7090 | 84.5768 | 45.0236 | 56.7109 |
| | SSIM | 0.0628 | 0.4227 | 0.5833 | 0.4795 | 0.4707 | **0.6296** |
| 3. | PSNR | 9.3707 | 37.121 | 39.0324 | 40.0009 | 37.6741 | **40.6039** |
| | Vrijeme | **2.4272** | 49.6591 | 58.4034 | 112.4827 | 55.5091 | 55.2683 |
| | SSIM | 0.0129 | 0.7632 | 0.6387 | **0.8141** | 0.6434 | 0.7142 |
| 4. | PSNR | 3.1803 | 42.0345 | 38.7451 | **42.1827** | 38.511 | 40.1567 |
| | Vrijeme | **1.8872** | 71.3299 | 69.5034 | 122.1715 | 64.4046 | 69.9305 |
| | SSIM | 0.0491 | 0.486 | 0.4566 | 0.5457 | **0.7604** | 0.6064 |
| 5. | PSNR | 11.1571 | 36.6601 | 38.0025 | 39.0443 | **43.643** | 40.748 |
| | Vrijeme | **2.1292** | 93.8923 | 83.7928 | 160.2068 | 85.3705 | 91.4205 |
| | SSIM | 0.023997 | 0.54006 | 0.71297 | 0.7004 | 0.68305 | **0.73834** |
| 6. | PSNR | 3.8232 | 37.6914 | 39.7779 | 40.1161 | 39.9686 | **41.0798** |
| | Vrijeme | **3.8232** | 50.4465 | 52.0248 | 100.6183 | 55.9244 | 51.8483 |
| | SSIM | 0.0430 | 0.6177 | **0.6220** | 0.6076 | 0.5809 | 0.6208 |
| 7. | PSNR | 5.2764 | 39.274 | **39.4851** | 38.8162 | 38.4765 | 39.2986 |
| | Vrijeme | **2.2494** | 49.6443 | 44.7004 | 83.0360 | 48.6709 | 46.8384 |
| | SSIM | 0.0144 | 0.6202 | 0.6245 | 0.7140 | 0.7422 | **0.7853** |
| 8. | PSNR | 4.0753 | 39.3794 | 39.1965 | 40.6602 | 41.0039 | **42.4878** |
| | Vrijeme | **3.4763** | 61.1867 | 71.2213 | 161.7264 | 59.0116 | 84.6686 |
| | SSIM | 0.0079 | 0.4594 | 0.5041 | 0.5991 | 0.6434 | **0.70069** |
| 9. | PSNR | 4.689 | 36.2033 | 37.1738 | 38.9074 | 39.3578 | **40.4328** |
| | Vrijeme | **2.457** | 61.096 | 51.9512 | 116.3460 | 50.1277 | 77.9008 |
| | SSIM | 0.0515 | 0.8049 | 0.3903 | 0.3918 | 0.3910 | **0.8193** |
| 10. | PSNR | 7.9131 | 42.623 | 35.5922 | 35.6136 | 35.6119 | **43.4239** |
| | Vrijeme | **2.3263** | 53.2134 | 54.9673 | 94.7974 | 52.0987 | 74.4038 |

**Tabela 5.5:** Usrednjene vrijednosti metrika kvaliteta segmentacije slike zasnovane na K-means grupisanju i različitim metaheurističim metodama(ABC, DE, MRFO, GWO, PSO-SA-Levy)

Posmatrajući numeričke rezultate u tabeli 5.5, možemo zaključiti da predloženi algoritam pokazuje dobre performanse, posebno na slikama 1, 2, 3, 6, 8, 9 i 10 u smislu PSNR-a i znatno viših SSIM vrijednosti. Veća vrijednost PSNR-a označava bolju očuvanost detalja slike i smanjen nivo šuma tokom

segmentacije. S druge strane, veća vrijednost SSIM-a ukazuje na superiorno očuvanje ivica, tekstura i cjelokupne strukture slike. Na primjer, na slici 1, PSO-SA-Levy je postigao SSIM od 0.6651 i PSNR od 41.6772, što pokazuje njegovu sposobnost da održi kvalitet slike. Dodatno, segmentacija pomoću PSO-SA-Levy je izvršena za 71.1411 sekundi, što predstavlja najbolje vrijeme nakon K-means algoritma. Na slici 3, sa SSIM-om od 0.6296, fino definisani detalji poput perja fazana koji su jasno vidljivi. Na slici 10, postignut je najbolji SSIM (0.8193) i PSNR (43.4249), znatno bolji od većine drugih algoritama koji su završili u lokalnom optimumu sa SSIM-om oko 0.39 i PSNR-om oko 35 dB.

Iako predloženi algoritam nije postigao najbolje rezultate na svim slikama, možemo primijetiti da su postignute vrlo visoke vrijednosti SSIM-a i PSNR-a te da je PSNR u skoro svim slikama veći od 40 dB, što ukazuje na to da je kvalitet slike veoma dobar i da segmentirana slika izgleda veoma slično originalu.

Posmatrajući u cjelini, K-means algoritam uz PSO-SA-Levy predstavlja uravnoteženo rješenje za segmentaciju slika, nudeći zadovoljavajući kvalitet segmentacije uz umjerenu računarsku složenost.

U prethodnom eksperimentu veličina $K$ je postavljena na 50, odnosno Algoritam simuliranog kaljenja se primjenjivao nakon 50 iteracija stagnacije GBest veličine. Ova vrijednost je određena eksperimentalnom analizom, gdje je isključen uticaj Levy leta i na taj način jasno vidjeli koliki uticaj ima SA algoritam. Jedan kratak primjer u tabeli 5.6 ilustruje perfomranse segmentacije Slike 5. u 3 klastera. U ovom primjeru takođe je uključena metrika FSIM (2.38).

| **K** | **PSNR** | **SSIM** | **FSIM** |
|---|---|---|---|
| 50 | 45.7334 | 0.8239 | 0.93 |
| 150 | 43.0238 | 0.69 | 0.9469 |
| 250 | 43.0238 | 0.69 | 0.9469 |
| 350 | 43.0238 | 0.69 | 0.9469 |

**Tabela 5.6:** PSNR, SSIM, FSIM vrijednost metrika za različite vrijednosti parametra $K$ (50, 150, 250, 350)

Tabela 5.6 jasno pokazuje da su performanse PSNR i SSIM znatno veće za vrijednost parametra $K = 50$. Metrika FSIM je malo manja, nego što je to prilikom ostalih vrijednosti $K$, međutim globalno posmatrajući, rezultati su sasvim zadovoljavajući i izbalansirani prilikom utvrđene vrijednosti $K = 50$.

Zatim, veličina koja kontroliše dužinu Levy koraka jeste stepen $\beta$ koji može varirati između 1 i 2. U prethodnim eksperimentima je postavljen na vrijednost 1.35, odnosno Levy korak je malo uvećan i povećana je raznolikost pretrage. Ovi parametri su utvrđeni kao globalno optimalni, eksperimentalnim putem. Što se tiče ostalih bitnih parametara, postavljeni su na sljedeći način: $c_1 = 2.5$, $c_2 = 2.5$, faktor hlađenja $\alpha = 0.9$ i početna temperatura $T = 10$.

U nastavku je prikazan još jedan veoma zanimljiv eksperiment koji pokazuje kako na segmentaciju slike utiče dužina Levy koraka. Proizvoljno je odabrana Slika 5. iz prethonog eksperimenta. i Za ovu sliku će biti prikazane performanse segmentacije, i to u kontekstu sljedećih metrika: PSNR, SSIM i FSIM.

| $\beta$ | **PSNR** | **SSIM** | **FSIM** |
|---|---|---|---|
| 1.1 | 38.0692 | 0.4538 | 0.9359 |
| 1.2 | 41.7324 | 0.7114 | 0.9178 |
| 1.3 | 41.6633 | 0.7082 | 0.9279 |
| 1.4 | 38.0025 | 0.4566 | 0.9356 |
| 1.5 | 40.1723 | 0.5630 | 0.9382 |
| 1.6 | 37.4494 | 0.4176 | 0.9374 |
| 1.7 | 36.6439 | 0.4847 | 0.8858 |
| 1.8 | 40.9382 | 0.6521 | 0.9368 |
| 1.9 | 37.6101 | 0.5062 | 0.9035 |

**Tabela 5.7:** Performanse segmentacije Slike 5. u zavisnosti od stepena $\beta$ u Levy distribuciji

U ovom eksperimentu dodatno je razmatrana i metrika FSIM. Na osnovu rezultata iz tabele 5.7 može se izvesti nekoliko zaključaka o uticaju Levy distribucije:

- parametar $\beta$ ima veliki uticaj na performanse. Na primjer, za vrijednosti $\beta$ 1.2,1.3 postignute su bolje performanse u pogledu PSNR i SSIM u odnosu na one koji se nalaze u Tabeli 5.5, za istu sliku,
- potrebno je pažljivo odabrati vrijednost parametra $\beta$ kako bi se postigao balans u postizanju optimalnih vrijednosti metrika segmentacije. Na primjer za većinu vrijednosti (izuzev 1.3 i 1.5) se može primijetiti da PSNR i SSIM imaju bolje, odnosno lošije performanse u odnosu na FSIM metriku. Različite metričke performanse mogu preferirati različite veličine koraka. Neki balans je postignut pri vrijednostima 1.3 i 1.5.

# Poglavlje 6

# Zaključak

U radu su predstavljeni rezultati istraživanja segmentacije pomoću klasterizacije zasnovane na poboljšanim K-means algoritmom. I pored aktuelnih metoda zasnovanih na vrlo naprednim algoritmima kao što su metodi pomoću praga, regiona, ivica, i neuralnih mreza, segmentacija slike zasnovana na K-means klasterizaciji je numerički efikasna i ne zavisi od prethodnog obučavanja/treniranja. Klasični K-means algoritam je osjetljiv na početnu inicijalizaciju centara klastera i često se suočava sa problemom zarobljavanja u lokalnim minimumima. Stoga, kao dodatak ovoj jednostavnoj metodi, ispitana je upotreba metaheurističkih algoritama kako bi se unaprijedile njegove performanse.

Pokazano je kako se metaheuristički metod PSO-SA-Levy primjenjuje za optimalnu inicijalizaciju centroida, pružajući efikasan pristup za optimizaciju centara klastera i predstavljajući dobru alternativu slučajnoj inicijalizaciji. Ovaj model je koristio prednosti čestične optimizacije (PSO), koja omogućava brzo istraživanje prostora rješenja, dok SA algoritam osigurava da se izbjegne zarobljavanje u lokalnim minimumima kroz probabilistički mehanizam prihvatanja lošijih rješenja. Levy letovi su dodatno pojačali sposobnost pretraživanja prostora rješenja, što je značajno poboljšalo kvalitet segmentacije.

Takođe, u ovom radu pokazana je važnost nasumičnih šetnji u metaheurističkim algoritmima. Nasumične šetnje su često implementirane u ovakvim algoritmima jer omogućavaju bolji balans između istraživanja i eksploatacije prostora rješenja. U našem radu, Levy let je predstavljen kao sofisticiranija verzija nasumične šetnje, a interesantno je dalje istražiti zamjenu funkcija za generisanje slučajnih sekvenci (`rand()`), korišćenih u mnogim algoritmima, sa Levy letom, jer bi to možda moglo dodatno unaprijediti performanse optimizacije. Takođe, interesantna bi bila implementacija Levy leta u generisanje susjeda unutar SA algoritma, što bi moglo donijeti nove mogućnosti u rješavanju kompleksnih problema optimizacije, jer bi i SA algoritam imao mogućnost pretrage još šireg i raznolikog prostora.

Uprkos prednostima PSO-SA-Levy hibridnog algoritma, treba napomenuti da ovaj model koristi veliki broj parametara, što predstavlja manu. Parametri za PSO (veličina populacije, faktoor inercije, koeficijenti učenja, itd.), SA (početna temperatura, faktor hlađenja, itd.) i Levy let ($\beta$) moraju biti pažljivo odabrani kako bi se postigle optimalne performanse. Međutim, ova karakteristika može biti i prednost jer omogućava dodatno prilagođavanje modela za različite tipove problema. Povećan broj parametara pruža veću fleksibilnost u podešavanju algoritma, ali istovremeno zahtijeva dublju analizu i eksperimentalno istraživanje kako bi se pronašle optimalne vrijednosti.

Kroz analizu eksperimentalnih rezultata jasno je da parametar Levy leta, specifično eksponent $\beta$, ima značajan uticaj na konačnu tačnost segmentacije slike. Povećanjem $\beta$ u određenom rasponu postižemo bolju ravnotežu između istraživanja i eksploatacije prostora rješenja, čime se osigurava brža konvergencija i preciznije centriranje klastera. Međutim, prevelika vrijednost $\beta$ može smanjiti stabilnost rješenja, dok premala vrijednost može ograničiti istraživački kapacitet algoritma. Optimalna vrijednost $\beta$ mora biti prilagođena specifičnostima problema i strukturi podataka. Takođe, jako je bitno utvrditi i parametar $K$ koji kontoliše uticaj SA algortima, jer je to jedan od bitnih faktora za stabilnost predloženog algoritma.

Analiza performansi PSNR, SSIM, i FSIM metrika pokazala je da kombinacija ovih tehnika u hibridnom algoritmu značajno poboljšava kvalitet segmentacije u poređenju sa standardnim K-means algoritmom. Uvedene promjene u inicijalizaciji i optimizaciji centara klastera omogućile su postizanje boljih vizuelnih i kvantitativnih rezultata, te su pokazale značajna unapređenja u smislu tačnosti, stabilnosti i brzine konvergencije.

Zaključno, hibridizacija PSO, SA i Levy leta pokazala se kao moćna tehnika za popravljanje peroformansi K-means metode segmentaciju slike i optimizaciju klaster centara, nadmašujući tradicionalne metode poput K-means algoritma, kao i bolje peroformanse od standardnih metaheurističkih algoritama. Ovaj pristup otvara nove mogućnosti za dalja istraživanja u primjeni metaheurističkih algoritama na složene probleme optimizacije, a može poslužiti i kao motivacija za unapređenje ostalih metoda segmentacije.

# Literatura

[1] R.M. Haralick and L.G. Shapiro, “Image Segmentation Techniques ”, *Computer Vision, Graphics and Image Processing*, Vol. 29, No. 1, pp. 100-132, 1985.

[2] M. W. Khan “A Survey: Image Segmentation Techniques, ”*International Journal of Future Computer and Communication*, Vol. 3, No. 2, 2014.

[3] P. Dhankhar and N. Sahu, “A review and research of edge detection techniques for image segmentation, ”*International Journal of Computer Science and Mobile Computing*, pp. 86-92, 2013.

[4] H. Li,H. He, and Y. Wen, “Dynamic particle swarm optimization and K-means clustering algorithm for image segmentation,”*Optik*, pp. 4817–4822, 2015.

[5] R. C. Hrosik, E. Tuba, E. Dolićanin, R. Jovanović, and M. Tuba, “Brain Image Segmentation Based on Firefly Algorithm Combined with K-means Clustering, ”*Studies in Informatics and Control*, vol. 28 no.2, 2019.

[6] S. Kapoor, I. Zeya, C. Singhal, and S. J. Nanda, “Grey Wolf Optimizer based automatic clustering algorithm for satellite image segmentation,”*Procedia Computer Science*, vol. 115, pp. 415–422, 2017.

[7] D. Zhu, L. Xie, , and C. Zhou, “K-means segmentation of underwater image based on improved Manta Ray algorithm, ”*Computational Intelligence and Neuroscience*, pp. 1–26, 2022.

[8] L. Idoumghar, M. Melkemi, R. Schott, and M. I. Aouad, , “Hybrid PSO-SA type algorithms for multimodal function optimization and reducing energy consumption in embedded systems, ”*Applied Computational Intelligence and Soft Computing*, pp. 1–12, 2011.

[9] R. Jensi and G. W. Jiji, “An enhanced particle swarm optimization with Levy flight for global optimization,”*Applied Soft Computing*, vol. 43, pp. 248–261, 2016.

[10] D. Lazarević, M. Mišić and B. Ćirković, “Postojeće tehnike za segmentaciju slike recent image segmentation techniques ”*ZBORNIK abstrakata/Festival kvaliteta 2014*, 2014.

[11] H. Mittal, AC. Pandey, M. Saraswat, S. Kumar, R. Pal and G. Modwel, “A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets, ” *Multimed Tools Appl* , vol.81, no. 24, 2022;

[12] R.C. González, R.E. Woods and B.R. Masters, "Digital Image Processing, Third Edition, "*Journal of Biomedical Optics*, pp. 738-776, 2009.

[13] J.S. Weszka, "A Survey of Threshold Selection Techniques ", *Computer Graphics and Image Processing*, vol. 7, no. 2, pp. 259-265, 1978.

[14] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms ", *IEEE Trans. Systems, Man, and Cybernetics*, vol. SMC-9, no. 1, pp. 62-66, 1979.

[15] P-S. Liao, T-S Chen, and P-C. Chung, "A fast algorithm for multilevel thresholding ", *Journal of Information Science and Engineering* vol. 17, no. 5, pp. 713-727, 2001.

[16] N. Ikonomatakis, K. N. Plataniotis, M. Zervakis and A. N. Venetsanopoulos, "Region growing and region merging image segmentation,"*Proceedings of 13th International Conference on Digital Signal Processing*, vol. 1, pp. 299-302, IEEE, 1997.

[17] R. Adams and L. Bischof, "Seeded region growing ". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641-647, 1994.

[18] A. Mehnert and P. Jackway, "An improved seeded region growing algorithm "*Pattern Recognition Letters*, vol. 18, no. 10, pp. 1065-1071, 1997.

[19] S. Y. Chen, W. C. Lin and C. T. Chen, "Split-and-merge image segmentation based on localized feature analysis and statistical tests ", *CVGIP: Graphical Models and Image Processing*, vol. 53, no.5, pp. 457-475, 1991.

[20] I. Đurović, "Digitalna obrada slike, "*Elektrotehnički fakultet, Elektrotehnički fakultet, Univerzitet Crne Gore*, pp. 212-260, 2006.

[21] M. Wang and D. Li, "An automatic segmentation method for lung tumor based on improved region growing algorithm, "*Diagnostics*, vol. 12, no.12, 2022.

[22] H. Sekiguchi, K. Sano and T. Yokoyama, "Interactive 3-dimensional segmentation method based on region growing method,"*Systems and Computers in Japan*, vol. 25, no. 1, pp. 88-97, 1994.

[23] A. M. Cross, S. J. Dury and D. C. Mason, "Segmentation of remotely-sensed images by a split-and-merge process, "*International Journal of Remote Sensing*, vol. 9, no.8, pp. 1329-1345, 1988.

[24] M. Popovic, "Digitalna obrada slike," *Akademska misao*, 2006.

[25] J. Long, E. Shelhamer and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation "*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.3431-3440, 2015.

[26] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation ", *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18.* Springer International Publishing, pp. 234-241, 2015.

[27] K. He, G. Gkioxari, P. Dollár, R. Girshick, "Mask r-cnn, "*In Proceedings of the IEEE international conference on computer vision*, pp. 2961-2969, 2017.

[28] G. Coleman and H.C. Andrews, "Image segmentation by clustering, "*Proceedings of the IEEE, 67*, pp. 773-785, 1979.

[29] T. Rahman and M. S. Islam, "Image segmentation based on fuzzy C means clustering algorithm and morphological reconstruction ,"*In 2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD) IEEE*, 2021.

[30] C. Rosenberger, S. Chabrier, H. Laurent and B. Emile, "Unsupervised and supervised image segmentation evaluation, "*In Advances in image and video segmentation,* pp. 365-393, 2006.

[31] V. Tomar, M. Bansal and P. Singh, "Metaheuristic Algorithms for Optimization: A Brief Review, "*Engineering Proceedings*, 2023.

[32] S. Katoch, SS. Chauhan and V. Kumar, "A review on genetic algorithm: past, present, and future, "*Multimed Tools Appl*, 2021.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization,"*Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4, pp. 1942-1948, 1995.

[34] J. R. Martins and A. Ning, "Engineering design optimization, "*Cambridge University Press*, 2021.

[35] B.Hajek, "Cooling Schedules for Optimal Annealing, "*Mathematics of Operations Research,* vol. 13, no.2, 1988.

[36] F. Lin, C. Kao and C. Hsu, "Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems,"*IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, vol. 23, no. 6, 1993.

[37] X. S. Yang,"Nature-inspired optimization algorithms,"*Academic Press*, 2020.

[38] Y. Shi and R. Eberhart, "A modified particle swarm optimizer,"in *IEEE International Conference on Evolutionary Computation Proceedings*, IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA, 1998.

[39] H. Haklı and H. Uğuz, "A novel particle swarm optimization algorithm with Levy flight,", *Applied Soft Computing*2014.

[40] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization,"*KanGAL report*2005005(2005), 2005.

[41] M. Molga and C. Smutnicki, "Test functions for optimization needs,", *Test Functions for Optimization Needs*, vol. 101, p. 48, 2005.

[42] M. Joshi, M. Gyanchandani and D. Rajesh Wadhvani, "Analysis Of Genetic Algorithm, Particle Swarm Optimization and Simulated Annealing On Benchmark Functions,"*2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1152-1157, Erode, India, 2021.

[43] A. Kumar and S. Bawa, "A comparative review of meta-heuristic approaches to optimize the SLA violation costs for dynamic execution of cloud services, "*Soft Computing 24.6* pp. 3909-3922, 2020.

[44] X. Chen, P. Miao and Q. Bu, "Image Segmentation Algorithm Based on Particle Swarm Optimization with K-means Optimization,"*2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 156-159, 2019,