Segmentacija slike temeljena na algoritmu šišmiša

Antunović, Dražen

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

Sveučilišni studij

SEGMENTACIJA SLIKE TEMELJENA NA ALGORITMU ŠIŠMIŠA

Diplomski rad

Dražen Antunović

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Dražen Antunović
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1263R, 07.10.2022.
JMBAG:	0165083913
Mentor:	izv. prof. dr. sc. Časlav Livada
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	izv. prof. dr. sc. Časlav Livada
Član Povjerenstva 2:	doc. dr. sc. Tomislav Galba
Naslov diplomskog rada:	Segmentacija slike temeljena na algoritmu šišmiša
Znanstvena grana diplomskog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu je potrebno napraviti opis algoritma šišmiša i njegovih varijanti kao i metode segmentacije slika. Poterbno je napraviti objašnjenje uporabe algoritma šišmiša i pojedinih varijanti kod segmentacije slika. Zatim treba Izraditi programsko rješenja te usporediti rezultate različith varijanti algoritma šišmiša. Tema rezervirana za: Dražen Antunović

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	15.07.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	19.09.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	19.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 19.09.2024.

Ime i prezime Pristupnika:	Dražen Antunović
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1263R, 07.10.2022.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Segmentacija slike temeljena na algoritmu šišmiša**

izrađen pod vodstvom mentora izv. prof. dr. sc. Časlav Livada

i sumentora

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

Potpis pristupnika:

SADRŽAJ

1. UVOD

Unutar ovog rada opisan je način korištenja algoritma šišmiša (engl. *bat algorithm*, BA) kod segmentacije slika. Pojedine metode segmentacije slike se mogu optimizirati, optimizacija znači pronalaženje što boljeg skupa parametara u što manjem vremenu kako bi se efektivno pronašla dobra rješenja. Te metode mogu imati veliki broj parametara ili veliki raspon vrijednosti koje parametri mogu poprimiti pa se može dogoditi da je isprobavanje redom kombinacija tih parametara u nekim slučajevima neizvedivo ili predugo traje. Cilj je bio pronaći te različite metode koje se mogu optimizirati i zatim koristiti algoritam šišmiša za optimizaciju. Utjecaj vrijednosti parametara koje on optimizira ima utjecaj i na konačnu segmentiranu sliku. Unutar ovog rada pokriven je način rada metaheurističnog algoritma točnije algoritma šišmiša, uloga takvog optimizacijskog algoritama u konkretnom problemu segmentacije slike te kvantitativni rezultati uporabe.

U drugom poglavlju opisana su postojeća rješenja uporabe algoritma šišmiša kod segmentacije slika. Zatim, u trećem poglavlju opisan je algoritam šišmiša. Specifičnije, opisan je princip rada, parametri koji se koriste, jednadžbe, pojedine varijante algoritma šišmiša te programska implementacija u programskom jeziku Python. Nakon toga je u četvrtom poglavlju opisana segmentacija slike. Točnije, opisane su vrste i metode segmentacije slike. Potom u petom poglavlju je opisana upotreba algoritma šišmiša kod segmentacije slika, opisane su varijante koje su se koristile za pojedine tipove segmentacije slike kao i metode koje te varijante koriste. U šestom poglavlju prikazani su eksperimentalni rezultati za različite varijante algoritma šišmiša kod pojedinih tipova segmentacije slike te je dan osvrt na postignute rezultate.

2. PREGLED PODRUČJA TEME

U ovom poglavlju opisana su neka od postojećih rješenja segmentacije slike korištenjem algoritma šišmiša i njegovih varijanti.

S. Yang, Q. Cheng i L. Peng su koristili algoritam šišmiša za segmentaciju crno-bijelih slika na više razina. Za funkcije cilja su koristili između-klasnu varijancu (engl. *between-class variance*) Otsuove metode i Kapurovu entropiju. Koristili su sljedeće metrike za evaluaciju sličnosti slika: Vršni omjer signala i šuma (engl. *peak signal-to-noise ratio*, PSNR) i Metodu indeksa strukturne sličnosti (engl. *structural similarity index method*, SSIM) prema [1].

J. Senthilnath i ostali su predložili nov (engl. *novel*) algoritam šišmiša u postupku grupiranja (engl. *clustering*) za različite tipove usjeva. Koristili su multi-spektralne satelitske slike za taj postupak. Korišten je bio skup za treniranje u svrhu pronalaženja optimalnih središta za grupiranje pomoću tog novog algoritma te su ta središta dalje validirana na skupu za testiranje. Usporedili su rezultate tog algoritma sa sljedećim algoritmima: genetskim algoritmom (engl. *genetic algorithm*, GA), optimizacijom roja čestica (engl. *particle swarm optimization*, PSO) i hibridnim pristupom algoritma šišmiša sa *K-meansom* prema [2].

Z. Ye i ostali su predložili modificirani kaotični algoritam šišmiša sa 2D Tsallisovom entropijom u svrhu segmentacije slika. Koristili su stvarne i infracrvene slike za testiranje te su ga usporedili sa postojećim metaheurističnim algoritmima, a to su: PSO, GA, optimizacija kolonije mrava (engl. *ant colony optimization*, ACO) i algoritam diferencijalne evolucije (engl. *differential evolution algorithm*, DE) prema [3].

X. Yue i H. Zhang su predložili hibridni algoritam šišmiša sa operacijom genetskog križanja (engl. *genetic crossover*) i pametnom inercijskom težinom (engl. *smart inertia weight*) u svrhu odabira najboljih graničnih vrijednosti (engl. *thresholds*) za segmentaciju slika. Koristili su između-klasnu varijancu Otsuove metode i Kapurovu entropiju kao funkciju cilja (engl. *objective function*). Usporedili su ga sa ostalim metaheurističnim algoritmima poput: GA, BA, gravitacijskog algoritama traženja (engl. *gravitational search algorithm*, GSA), algoritma optimizacije kitova (engl. *whale optimization algorithm*, WOA) itd. prema [4].

Q. Lu, Z. Zhang i C. Yue su koristili algoritam šišmiša kod optimizacije parametara pulsno-spregnute neuronske mreže (engl. *pulse-coupled neural network*, PCNN). PCNN ima veliki broj parametara pa je korišten algoritam šišmiša za njihovu optimizaciju, a PCNN je korištena u svrhu segmentacije slika prema [5].

3. ALGORITAM ŠIŠMIŠA

Algoritam šišmiša pripada skupini metaheurističnih algoritama temeljenih na inteligenciji roja (engl. *swarm-based*) prema [6]. Inspiriran je ponašanjem šišmiša u prirodi te ga je razvio X. S. Yang u 2010. godini.

Metaheuristični algoritmi su algoritmi koji se koriste kod optimizacije. Upotrebljavaju se u slučaju kada tradicionalne metode nisu prikladne za rješavanje određenog problema. Oni iterativno pronalaze bolja rješenja za zadanu funkciju cilja ovisno o tipu problema (maksimizacija ili minimizacija zadane funkcije) prema [7]. Inspirirani su od različitih stvari poput: inteligencije roja (npr. kolonija mrava, jato ptica), evolucije (npr. genetski algoritmi), putanja (npr. simulirano žarenje (engl. *simulated annealing*)) itd. prema [6].

3.1. Karakteristike i pseudokôd algoritma šišmiša

Šišmiši imaju sposobnost eholokacije (engl. *echolocation*). To znači da šišmiši ispuštaju veoma glasne zvučne impulse koji se zatim odbijaju od okolnih površina. Na taj način šišmiši se orijentiraju u prostoru što im omogućuje pronalaženje plijena, izbjegavanje prepreka na putu itd. Postoje brojne vrste šišmiša te ih većina koristi eholokaciju. Neki je koriste više, neki manje, međutim mikro šišmiši je koriste intenzivno. Svaki impuls koji šišmiši ispuste ima konstantnu frekvenciju koja je obično u rasponu od 25 kHz do 150 kHz ili 100 kHz, ovisno o vrsti šišmiša. Mikro šišmiši tipično emitiraju 10 do 20 zvučnih prasaka svake sekunde te se taj broj može povećati na oko 200 kada love plijen. Ti pulsovi mogu biti glasni do 110 decibela, te se njihova glasnoća smanjuje kada se približavaju meti. Tipični domet pulsova je oko nekoliko metara, ovisno o frekvenciji.

Kako bi se mogao načiniti algoritam temeljen na ponašanju šišmiša potrebno je prilagoditi karakteristike ponašanja šišmiša te prikladno uključiti funkciju cilja koju algoritam treba optimizirati.

Virtualni šišmiši korišteni u simulacijama imaju sljedeće parametre:

- x – trenutna lokacija/pozicija šišmiša, ovo predstavlja rješenje
- v – trenutna brzina šišmiša
- r – brzina emisije pulsa
- f – frekvencija emitiranih pulsova
- A – glasnoća šišmiša (emitiranog pulsa)

Šišmiši se kreću sa lokacije na lokaciju ovisno o trenutnoj brzini. Ti šišmiši mogu automatski podešavati frekvenciju emitiranih pulsova, u implementaciji to je ostvareno tako što je generiraju nasumično. Brzina emisije pulsa i glasnoća šišmiša se mijenjaju sa iteracijama jer se šišmiši kroz iteracije približavaju svojoj meti, brzina emisije pulsa se povećava, a glasnoća se smanjuje. Šišmiši imaju mogućnost istraživanja (engl. *exploration*) i iskorištavanja (engl. *exploitation*).

Pseudokôd za algoritam šišmiša prikazan je na slici 3.1.

Linija ***Kod***

```
        Inicijaliziraj populaciju šišmiša sa lokacijom x_i i brzinom v_i (i =
1:      1, 2, 3, … , n)
        Inicijaliziraj frekvencije f_i, brzine emitiranja pulsa r_i i glasnoće
2:      A_i
3:      dokle god je (t < t_max) čini
4:         Generiraj nova rješenja prilagođavanjem frekvencije
5:         Ažuriraj brzine i lokacije
6:         ako je (rand > r_i) onda
7:            Izaberi rješenje između najboljih rješenja
8:            Generiraj lokalno rješenje oko izabranog najboljeg rješenja
9:         kraj
10:        Generiraj nova rješenja letećи nasumično
11:        ako je (rand < A_i i f(x_i) < f(x_*)) onda
12:           Prihvati nova rješenja
13:           Povećaj r_i i smanji A_i
14:        kraj
15:        Rangiraj šišmiše i pronađi trenutni najbolji x_*
16:     kraj
```

Sl. 3.1. Pseudokôd algoritma šišmiša

Algoritam se izvršava dok se ne dođe do maksimalnog broja iteracija gdje t_{max} predstavlja maksimalni broj iteracija. *rand* predstavlja nasumični broj iz uniformne distribucije u rasponu [0, 1) prema [8].

3.2. Izračuni parametara

Kako bi se mogao shvatiti rad algoritma šišmiša potrebno je shvatiti kako se parametri računaju.

Izraz za generiranje frekvencije pojedinog šišmiša izgleda ovako:

$$f_i = f_{min} + (f_{max} - f_{min}) * \beta \tag{3-1}$$

gdje f_i predstavlja frekvenciju *i*-tog šišmiša, f_{min} predstavlja minimalnu vrijednost koju frekvencija može poprimiti, f_{max} predstavlja maksimalnu vrijednost koju frekvencija može poprimiti i β predstavlja nasumični broj iz uniformne distribucije u rasponu [0, 1).

Sljedeće, izrazi za računanje brzine i lokacije/pozicije pojedinog šišmiša izgledaju ovako:

$$v_i{}^{t+1} = v_i{}^{t} + (x_i{}^{t} - x_*) * f_i \tag{3-2}$$

$$x_i{}^{t+1} = x_i{}^{t} + v_i{}^{t+1} \tag{3-3}$$

gdje $v_i{}^{t+1}$ predstavlja ažuriranu brzinu *i*-tog šišmiša, $v_i{}^{t}$ predstavlja trenutnu brzinu *i*-tog šišmiša, $x_i{}^{t+1}$ predstavlja ažuriranu lokaciju *i*-tog šišmiša, $x_i{}^{t}$ predstavlja trenutnu lokaciju *i*-tog šišmiša, f_i predstavlja frekvenciju *i*-tog šišmiša i x_* predstavlja lokaciju šišmiša koji ima najbolju lokaciju.

Kôd unutar ($rand > r_i$) uvjeta predstavlja lokalno traženje. Unutar tog dijela lokalno rješenje/lokacija se računa sljedećim izrazom:

$$x_{novo} = x_{staro} + \sigma * \epsilon_t * A^{(t)} \tag{3-4}$$

gdje x_{novo} predstavlja novo rješenje, x_{staro} predstavlja staro rješenje, ϵ_t predstavlja nasumični broj iz Gaussove normalne distribucije sa sredinom 0 i varijancom 1 tj. N(0, 1), σ predstavlja faktor skaliranja i $A^{(t)}$ predstavlja prosječnu glasnoću svih šišmiša u toj iteraciji.

Unutar ($rand < A_i$ i $f(x_i) < f(x_*)$) uvjeta brzina emisije pulsa pojedinog šišmiša se računa na sljedeći način:

$$r_i{}^{t+1} = r_i{}^{0} + (1 - e^{-\gamma * t}) \tag{3-5}$$

gdje $r_i{}^{t+1}$ predstavlja ažuriranu brzinu emisije pulsa *i*-tog šišmiša, $r_i{}^{0}$ predstavlja inicijalnu brzinu emisije pulsa *i*-tog šišmiša, γ predstavlja faktor povećavanja i *t* predstavlja broj iteracije.

Glasnoća pojedinog šišmiša ažurira se na sljedeći način:

$$A_i{}^{t+1} = \alpha * A_i{}^{t} \tag{3-6}$$

gdje $A_i{}^{t+1}$ predstavlja ažuriranu glasnoću *i*-tog šišmiša, α predstavlja faktor smanjivanja i $A_i{}^{t}$ predstavlja trenutnu glasnoću *i*-tog šišmiša.

Inicijalne vrijednosti glasnoće i emisije pulsa su proizvoljne, no trebale bi biti pozitivni brojevi u rasponu (0, 1]. Ažurirane vrijednosti brzine emisije pulsa teže ka inicijalno zadanoj vrijednosti brzine emisije pulsa, sve više i više se tokom iteracija približavaju toj vrijednosti. Glasnoća

pojedinog šišmiša tokom iteracija opada te teži u 0. Ove dvije vrijednosti su ažurirane u slučaju kada je rješenje koje je našao *i*-ti šišmiš bolje u odnosu na njegovu prijašnju vrijednost prema [8].

3.3. Varijante algoritma šišmiša

Prema [8] postoje brojne varijante algoritma šišmiša. Neke od poznatijih su:

- Kaotični algoritam šišmiša (engl. *chaotic bat algorithm*, CBA) – koristi generirane nasumične brojeve od strane kaotičnih mapa u pojedinim dijelovima algoritma. Postoji više različitih varijanti ovog algoritma.
- Binarni algoritam šišmiša (engl. *binary bat algorithm*, BBA) – koristi se za rješavanje binarnih problema. Lokacije šišmiša mogu poprimiti vrijednosti 1 ili 0.
- Algoritam šišmiša sa više ciljeva (engl. *multi-objective bat algorithm*, MOBA) – koristi više funkcija cilja pri traženju rješenja. Postoje različite strategije za rad sa više ciljeva.
- Hibridni algoritam šišmiša (engl. *hybrid bat algorithm*) – algoritam šišmiša u kombinaciji sa još nekim algoritmom/algoritmima u svrhu zajedničkog rješavanja problema.

3.4. Programska implementacija

Algoritam šišmiša je implementiran u obliku objektno-orijentiranog (engl. *object-oriented*) rješenja u programskom jeziku Python. Sljedeće su prikazane komponente koje ga sačinjavaju.

3.4.1. Klasa za šišmiša

Klasa Šišmiš prikazana je na slici 3.2. Ona sadrži parametre za pojedinog šišmiša.

```
import numpy as np
import numpy.typing as npt

class Bat:
    def __init__(self, position: npt.NDArray[np.float64],
                 velocity: npt.NDArray[np.float64], frequency: float,
                 fitness: float, marginal_pulse_rate: float=1.,
                 pulse_rate: float=0., loudness: float=1.) -> None:
        self.position: npt.NDArray[np.float64] = position
        self.velocity: npt.NDArray[np.float64] = velocity
        self.frequency: float = frequency
        self.fitness: float = fitness
        self.marginal_pulse_rate: float = marginal_pulse_rate
        self.pulse_rate: float = pulse_rate
        self.loudness: float = loudness
```

Sl. 3.2. Klasa za pojedinog šišmiša

Atribut *marginal_pulse_rate* što doslovno prevedeno znači *marginalna_emisija_pulsa* predstavlja inicijalnu brzinu emisije pulsa kojoj vrijednosti brzine emisije pulsa teže tokom iteracija odnosno r_i^0 prema izrazu (3-5).

Pošto algoritam šišmiša može rješavati višedimenzionalne probleme, atributi *position* i *velocity* su polja te je veličina polja jednaka broju dimenzija problema.

3.4.2. Klasa za globalne parametre algoritma šišmiša

Sljedeće je implementirana klasa za parametre algoritma šišmiša što je prikazano na slici 3.3.

```
import numpy.typing as npt
import numpy as np

class BatAlgorithmParams:
    def __init__(self, total_bats: int, num_iterations: int,
                 dimension: int, min_position: float | list[float] |
                 npt.NDArray[np.float64], max_position: float | list[float] |
                 npt.NDArray[np.float64], min_frequency: float=0.,
                 max_frequency: float=1., sigma: float=0.1,
                 gamma: float=0.1, alpha: float=0.97) -> None:
        if(total_bats < 1):
            raise ValueError("Argument total_bats cannot be less than 1")
        if(dimension < 1):
            raise ValueError("Argument dimension cannot be less than 1")
        if(isinstance(min_position, (list, np.ndarray))):
            if(len(min_position) != dimension and len(min_position) != 1):
                raise ValueError("Argument min_position must have same size as"
                    " dimension parameter's value or have a single value")
        if(isinstance(max_position, (list, np.ndarray))):
            if(len(max_position) != dimension and len(max_position) != 1):
                raise ValueError("Argument max_position must have same size as"
                    " dimension parameter's value or have a single value")
        self.total_bats: int = total_bats
        self.num_iterations: int = num_iterations
        self.dimension: int = dimension
        self.min_position: npt.NDArray[np.float64] = np.copy(min_position)
        self.max_position: npt.NDArray[np.float64] = np.copy(max_position)
        self.min_frequency: float = min_frequency
        self.max_frequency: float = max_frequency

        self.sigma: float = sigma
        self.gamma: float = gamma
        self.alpha: float = alpha
```

Sl. 3.3. Klasa za globalne parametre algoritma šišmiša

Ovu klasu nasljeđuju klase svih varijanti algoritma šišmiša jer sve varijante koriste te parametre. Ona sadrži parametre koji nisu karakteristični za jednog šišmiša, nego predstavljaju vrijednosti koje su, može se reći, globalne.

Atribut *dimension* predstavlja broj dimenzija problema, tj. broj ulaznih vrijednosti funkcije cilja.

Atributi *min_position* i *max_position* predstavljaju granične vrijednosti za poziciju do koje se šišmiši mogu kretati.

Atributi *min_frequency* i *max_frequency* predstavljaju granične vrijednosti za frekvenciju ispuštanja pulsova šišmiša.

Pošto veličina polja za poziciju ili brzinu kod svakog šišmiša ovisi o veličini problema odnosno broju dimenzija problema, tako i veličine polja kod atributa za minimalnu i maksimalnu poziciju ovise o tome, iako mogu imati i jednu vrijednost što znači da ona vrijedi za sve dimenzije.

3.4.3. Klasa za algoritam šišmiša

Sljedeće je implementirana klasa za sam algoritam šišmiša sa svojom logikom. Ona nasljeđuje klasu *BatAlgorithmParams*. Prikaz klase za algoritam šišmiša dan je na slici 3.4.

```
from bat_algorithm.bat import Bat
import numpy as np
from typing import Callable
import numpy.typing as npt
from bat_algorithm.bat_algorithm_params import BatAlgorithmParams

class BatAlgorithm(BatAlgorithmParams):
    def __init__(self, total_bats: int, num_iterations: int,
                 dimension: int, min_position: float | list[float] |
                 npt.NDArray[np.float64], max_position: float | list[float] |
                 npt.NDArray[np.float64], min_frequency: float=0.,
                 max_frequency: float=1., sigma: float=0.1,
                 gamma: float=0.1, alpha: float=0.97) -> None:

        super().__init__(total_bats=total_bats, num_iterations=num_iterations,
                         dimension=dimension, min_position=min_position,
                         max_position=max_position, min_frequency=min_frequency,
                         max_frequency=max_frequency, sigma=sigma,
                         gamma=gamma, alpha=alpha)

    def _initialize_population(self, objective_function:
                               Callable[[npt.NDArray[np.float64]], float]
                               ) -> list[Bat]:
        bats: list[Bat] = []
```

```
        for _ in range(self.total_bats):
            position: npt.NDArray[np.float64] = np.random.uniform(
                self.min_position, self.max_position, self.dimension)
            velocity: npt.NDArray[np.float64] = np.zeros(self.dimension)
            frequency: float = 0.
            marginal_pulse_rate: float = 1.
            pulse_rate: float = 0.
            loudness: float = 1.
            fitness: float = objective_function(position)
            bats.append(Bat(
                position=position,
                velocity=velocity,
                frequency=frequency,
                fitness=fitness,
                marginal_pulse_rate=marginal_pulse_rate,
                pulse_rate=pulse_rate,
                loudness=loudness
            ))
        return bats

    def run(self, objective_function: Callable[[npt.NDArray[np.float64]], float]
            ) -> tuple[npt.NDArray[np.float64], float]:
        bats: list[Bat] = self._initialize_population(objective_function)
        t: int = 0
        best_bat: Bat = min(bats, key=lambda bat: bat.fitness)
        while(t < self.num_iterations):
            t = t + 1
            for bat in bats:
                bat.frequency = np.random.uniform(
                    self.min_frequency, self.max_frequency)
                bat.velocity = bat.velocity + (bat.position-best_bat.position) \
                    * bat.frequency
                temp_position: npt.NDArray[np.float64] = bat.position \
                    + bat.velocity
                if(np.random.rand() > bat.pulse_rate):
                    temp_position = best_bat.position + self.sigma \
                        * np.random.normal(0, 1, self.dimension) \
                        * np.mean([bat.loudness for bat in bats])
                temp_position = np.clip(
                    temp_position, self.min_position, self.max_position)
                new_fitness: float = objective_function(temp_position)
                if(np.random.rand() < bat.loudness
                   and new_fitness < bat.fitness):
                    bat.position = temp_position
                    bat.fitness = new_fitness
```

```
                    bat.pulse_rate = bat.marginal_pulse_rate \
                        * (1 - np.exp(-self.gamma*t))
                    bat.loudness = self.alpha * bat.loudness
                    if(new_fitness < best_bat.fitness):
                        best_bat = bat
        return best_bat.position, best_bat.fitness
```

Sl. 3.4. Klasa za algoritam šišmiša

Unutar metode *_initialize_population* generiraju se početne vrijednosti parametara za svakog šišmiša unutar populacije. Funkcija cilja prima lokacije pojedinog šišmiša, a vraća *fitness* vrijednost. Na kraju metoda *_initialize_population* vraća polje šišmiša sa postavljenim vrijednostima parametara.

Na početku metode *run* inicijalizira se populacija šišmiša pozivom metode *_initialize_population*. Nakon toga se pronalazi šišmiš sa najboljom vrijednošću *fitnessa* odnosno u ovom slučaju najmanjom jer se rješava minimizacijski problem. Nakon toga se vrti glavni dio algoritma šišmiša dok se ne dođe do maksimalnog broja iteracija. Unutar iteracija se za svakog šišmiša iz populacije nasumično generira frekvencija prema izrazu (3-1), *min_frequency* je uključiva, a *max_frequency* je isključiva u programu. Zatim se generira brzina prema izrazu (3-2) te se generira privremena pozicija prema izrazu (3-3). Nakon toga se provjerava je li nasumično generirani broj iz uniformne distribucije veći od brzine ispuštanja pulsa od trenutnog šišmiša. U slučaju da jest, privremena pozicija se ažurira prema izrazu (3-4) gdje x_{staro} predstavlja trenutno najbolje rješenje, a x_{novo} privremeno rješenje trenutnog šišmiša (nije nužno najbolje). Zatim se ograničavaju vrijednosti trenutne pozicije u slučaju da su prešle donju ili gornju granicu. Nakon toga se izračunava nova *fitness* vrijednost za privremeno rješenje. Zatim se provjerava je li nasumično generirani broj iz uniformne distribucije manji od glasnoće trenutnog šišmiša i je li novi *fitness* manji od postojećeg *fitnessa* trenutnog šišmiša. U slučaju da su oba uvjeta zadovoljena postavlja se privremena pozicija na stvarnu poziciju trenutnog šišmiša te se novi *fitness* postavlja na stvarni *fitness* trenutnog šišmiša. Potom se ažuriraju brzina emisije pulsa i glasnoća prema izrazima (3-5) i (3-6). Poslije toga se unutar istih tih uvjeta provjerava da li je novi *fitness* manji od *fitnessa* najboljeg šišmiša. U slučaju da jest, trenutni šišmiš postaje najbolji šišmiš. Na kraju funkcija vraća najbolje rješenje i najbolji *fitness*.

4. SEGMENTACIJA SLIKE

Segmentacija slike predstavlja podjelu slike na više dijelova ili regija. Ti dijelovi su odabrani na temelju karakteristika unutar slike poput: boje, teksture, svjetline itd. prema [9]. Segmentacija slike je proširenje klasifikacije slike prema [10]. Segmentacija slike pojednostavljuje prikaz slike i omoguće lakšu analizu za pojedine svrhe. Pikseli imaju odgovarajuću oznaku (engl. *label*) koja im je pridružena. Pikseli koji pripadaju istoj kategoriji imaju istu oznaku prema [9]. Stoga se pomoću segmentacije slike pored pridruživanja oznake objektima, oni mogu i ocrtati. Ima primjenu u raznim granama industrije kao što su medicina, satelitske snimke, agrokultura, autonomna vožnja i ostalo prema [11].

4.1. Vrste segmentacije slike

Postoje 3 vrste segmentacije slike:

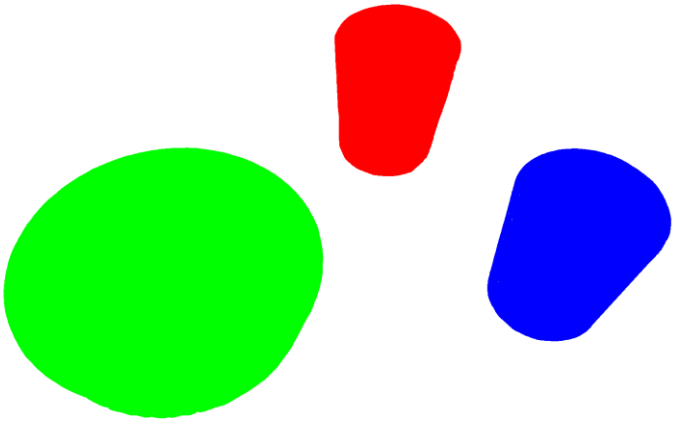
- segmentacija instance (engl. *instance segmentation*)
- semantička segmentacija (engl. *semantic segmentation*)
- panoptička segmentacija (engl. *panoptic segmentation*)

Kako bi se vizualizirala razlika između različitih vrsta segmentacija, korištena je slika 4.1.

Sl. 4.1. Testna slika za prikaz različitih tipova segmentacija

Segmentacija instance služi za označavanje objekata na slici. Svaki objekt predstavlja pojedinačnu instancu te je tako drugačije i označen naspram ostalih.

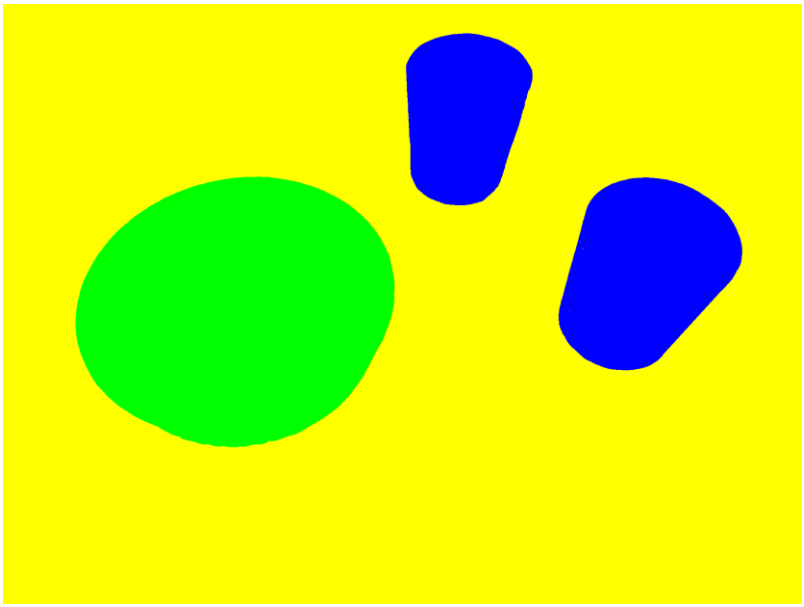
Slika 4.2. prikazuje kako bi trebala izgledati slika segmentirana pomoću segmentacije instance.



Sl. 4.2. Slika segmentirana segmentacijom instance

Kod semantičke segmentacije svakom je pikselu na slici pridružena odgovarajuća klasa. Ovdje se ne prave razlike u označavanju između različitih instanci iste klase/kategorije.

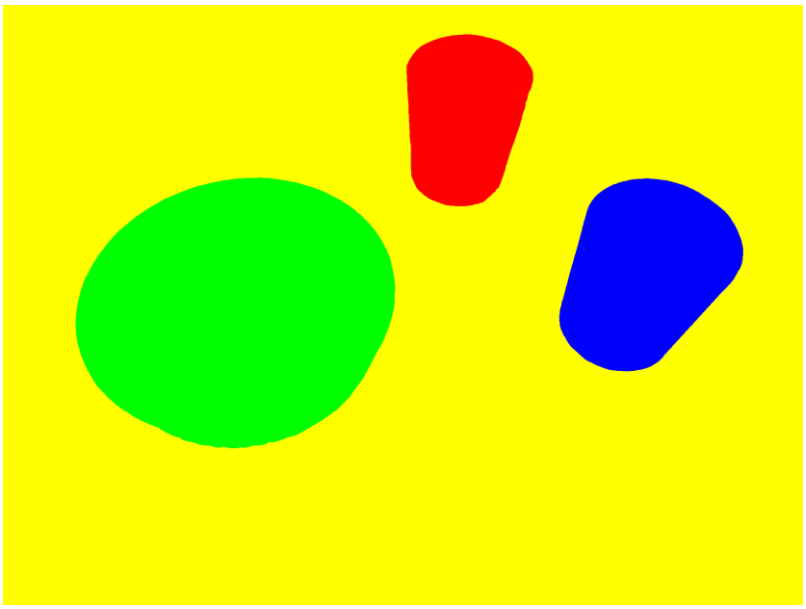
Slika 4.3. prikazuje kako bi trebala izgledati slika segmentirana pomoću semantičke segmentacije.



Sl. 4.3. Slika segmentirana semantičkom segmentacijom

Panoptička segmentacija predstavlja kombinaciju prethodne dvije segmentacije na način da je svakome pikselu pridružena odgovarajuća klasa, ali su također pojedinačne instance objekta posebno označene prema [9].

Slika 4.4. prikazuje kako bi trebala izgledati slika segmentirana pomoću panoptičke segmentacije.



Sl. 4.4. Slika segmentirana panoptičkom segmentacijom

4.2. Metode segmentacije slike

Prema [11] postoje 2 tipa metoda za segmentaciju slika, a to su:

1. Tradicionalne metode
2. Metode dubokog učenja (engl. *deep learning methods*)

Tradicionalne metode su nastale prije metoda dubokog učenja. Temeljene su na matematičkim izračunima i algoritmima koji služe za detekciju dijelova slike ovisno o sličnim karakteristikama koje su spomenute ranije prema [9]. Prvotno je segmentacija slike nastala tako što se koristila obrada slike zajedno sa optimizacijskim algoritmima. Ti algoritmi su koristili metode koje su se temeljile na lokalnim razlikama unutar slike kao što su: rast regije (engl. *region growing*) i algoritam zmija (engl. *snakes algorithm*), a mnogo kasnije su došle metode koje su se fokusirale na globalni prikaz slike kao što su: Otsuova metoda i algoritmi grupiranja prema [10]. Ovi algoritmi su računalno efektivni jer ne zahtijevaju dugo vrijeme izvođenja i poprilično su jednostavni za implementaciju prema [9].

Prema [9] pod tradicionalne metode spadaju:

1. Definiranje praga (engl. *thresholding*)
2. Segmentacija temeljena na regiji (engl. *region based segmentation*)
3. Segmentacija temeljena na rubu (engl. *edge based segmentation*)
4. Grupiranje

Metode dubokog učenja su novije metode i temelje se na različitim modelima neuronskih mreža.

4.2.1. Definiranje praga

Definiranje praga je jednostavna metoda kod segmentacije slike. Kod definiranja praga slika se može podijeliti na dva ili više dijela tako što se postavi granična vrijednost/vrijednosti (prag/pragovi) za raspon intenziteta piksela. Nakon toga one vrijednosti piksela ispod granične vrijednosti pripadaju jednom segmentu dok one iznad pripadaju drugom prema [12]. Ova metoda je pogodna kada je razlika u vrijednostima piksela između klasa velika prema [10]. Ova metoda se uglavnom koristi za crno-bijele (engl. *grayscale*) slike prema [11].

Prema [12] pod metode za definiranje praga spadaju:

- Globalno definiranje praga (engl. *global thresholding*) – koristi se za segmentaciju slike na dva dijela: pozadinu i objekte. Funkcionira tako što se odabere globalna granična vrijednost za čitavu sliku prema [12]. Nakon toga su regije podijeljene na način da oni pikseli ispod granične vrijednosti pripadaju pozadini, a oni iznad objektima. Ova tehnika nije baš prikladna u slučaju promjenjivog osvjetljenja na slici prema [9].
- Ručno definiranje praga (engl. *manual thresholding*) – funkcionira na način da se pronađe srednja vrijednost za piksele unutar regije te njihova srednja vrijednost zajedno koja predstavlja graničnu vrijednost. To se ponavlja dokle god je apsolutna razlika između prijašnje granične vrijednosti i trenutne veća od neke zadane vrijednosti izvana. Inicijalna granična vrijednost je također zadana izvana prema [12].
- Adaptivno definiranje praga (engl. *adaptive thresholding*) – koristi se kada se slika želi podijeliti na pozadinu i objekte na način da se podesi granična vrijednosti lokalno za svaku manju regiju na slici na temelju sličnih karakteristika prema [9].
- Optimalno definiranje praga (engl. *optimal thresholding*) – koristi se za smanjenje pogrešne klasifikacije piksela pri segmentaciji. Bazira se na korištenju statističkih izraza i izračuna poput aritmetičke sredine i varijance prema [12].

4.2.2. Segmentacija temeljena na regiji

Segmentacija temeljena na regiji grupira povezane piksele koji imaju slične karakteristike. Pravila se zadaju te se po njima pikseli grupiraju zajedno prema [12].

Prema [12] postoji više tehnika kod segmentacije temeljene na regiji, a pod njih spadaju:

- Tehnika rasta regije (engl. *region growing technique*) – funkcionira na način da se odabere početni piksel te se gledaju njegovi susjedi. Ako oni zadovoljavaju uvjet, spadaju pod istu regiju u suprotnom spadaju pod suprotnu regiju. Nakon toga se odabiru susjedi koji su ušli u regiju te se procedura ponavlja za njih sve dok nema više susjeda za odabir.
- Dijeljenje i spajanje regije (engl. *region splitting and merging*) – funkcionira na način da se slika podijeli na više dijelova, uglavnom 4 kvadranta, te se isti postupak podijele rekurzivno nastavlja za regije sve dok zadani uvjeti nisu zadovoljeni. Nakon toga se odabere regija te se uspoređuje sa susjednim regijama, ako susjedna regija zadovoljava uvjet sličnosti spaja se sa odabranom regijom te se postupak uspoređivanja susjeda nastavlja dok više nema regija za spajanje.

4.2.3. Segmentacija temeljena na rubu

Kod ove segmentacija pronalaze se i odvajaju rubovi objekata od pozadine. Radi tako što pronalazi velike promjene u karakteristikama na slici poput boje ili intenziteta svjetlosti piksela kako bi se ocrtale granice objekata prema [9].

Dvije su kategorije tehnika za detekciju rubova, a to su:

- Tehnike temeljene na gradijentu (engl. *gradient based techniques*) – koriste prvu derivaciju kako bi detektirale ekstreme za pronalaženje rubova.
- Tehnike temeljene na Gaussu (engl. *Gaussian based techniques*) – koriste drugu derivaciju za detektiranje prijelaza preko nule (engl. *zero-crossings*) kako bi pronašle rubove.

Pod tehnike temeljene na gradijentu spadaju:

- Sobelov operator
- Prewittov operator
- Robertov operator

Ovi operatori koriste horizontalne i vertikalne maske koje su primijenjene na crno-bijelu sliku kako bi se izračunao gradijent. Nakon toga se može izračunati magnituda i orijentacija gradijenta prema [13].

Prema [13] pod tehnike temeljene na Gaussu spadaju:

- Otkrivanje rubova korištenjem Laplaceovog Gaussovog operatora (engl. *Laplacian of Gaussian edge detection*) - ova tehnika koristi Laplaceov Gaussov operator kako bi se pronašli prijelazi preko nule prema [13].
- Canny otkrivanje rubova (engl. *Canny edge detection*) – na početku tehnike crno-bijela slika se prvo ublaži (engl. *to smooth*) korištenjem Gaussovog filtera kako bi se smanjio šum. Nakon toga se računa gradijent korištenjem vertikalnog i horizontalnog Sobelovog operatora. Zatim se pronalaze magnituda i orijentacija gradijenta. Poslije toga se vrši ne-maksimalno prigušenje (engl. *non-maximum supression*). Potom se radi dvostruko definiranje praga kako bi se rubovi kategorizirali na jake rubove, na slabe rubove i na ono što nisu rubovi. Na kraju se prate rubovi pomoću histereze kako bi se povezali slabi rubovi sa jakima prema [14].

4.2.4. Grupiranje

Cilj grupiranja je grupirati piksele po sličnim karakteristikama prema [9].

Postoji više metoda za grupiranje, a pod njih prema [9] spadaju:

- *K-means* – funkcionira na način da su pikseli predstavljeni kao točke podataka u prostoru te se sličnost mjeri prema metrici za udaljenost poput Euklidske ili Mahalanobisove udaljenosti. Na početku se postavi K broj nasumičnih središta grupacija. Zatim se pikseli iterativno grupiraju unutar grupacije čije je središte najbliže tom pikselu. Nakon toga se računa srednja vrijednost grupacije koja postaje novo središte. Koraci grupiranja piksela i izračuna središta grupacija se vrše dok se ne dođe do stabilne vrijednosti ili se ne postigne maksimalni broj iteracija prema [9].
- Grupiranje prema srednjem pomaku (engl. *mean shift clustering*) – kod ove metode pikseli su također predstavljeni kao točke podataka u prostoru. Oko svake točke podataka postavljen je prozor koji se pomoče. Prozor se iterativno pomjera tako što se računa srednja vrijednost za vrijednosti unutar prozora te ona postaje središte prozora. Na taj način prozor se pomjera prema gušćoj regiji. Kada se dva prozora preklapaju onaj s više podataka je sačuvan. Proces se ponavlja dokle god se prozor pomjera. Na kraju su pikseli grupirani prema istim krajnjim prozorima. Kod ove metode se ne zadaje broj grupacija prema [15].

4.2.5. Metode dubokog učenja

Kod metoda dubokog učenja model se prvo istrenira na postojećem skupu podataka za treniranje. Tokom treninga se on validira na skupu za validaciju te se na kraju testira na testnom skupu. Obično se modeli dubokog učenja sastoje od arhitekture koder-dekoder. Koder služi za izdvajanje značajki, a dekoder za povećavanje koderovog izlaza u segmentacijsku masku prema [9].

Prema [11] neki od modela neuronskih mreža za segmentaciju slika uključuju:

- U-Net – napravljen za medicinske svrhe. Sastoji se od dva puta: sužavajućeg (engl. *contracting*) i proširujućeg (engl. *expansive*). Kod sužavajućeg puta provode se uzastopne operacije dvije 3x3 konvolucije, gdje su obje konvolucije provedene kroz funkciju ispravljajuće linearne jedinice (engl. *rectified linear unit*, ReLU). Nakon toga slijedi 2x2 maksimalno sužavanje (engl. *max pooling*) sa korakom (engl. stride) od 2 kako bi se provelo poduzorkovanje (engl. *downsampling*). Za vrijeme poduzorkovanja broj kanala značajki se udvostručuje. Na proširujućem putu provede se operacije: 2x2 gornje konvolucije (engl. *up-convolution*) gdje se događa naduzorkovanje (engl. *upsampling*), tu se broj kanala značajki smanjuje na pola. Zatim se događa spajanje sa izrezanom mapom značajki (engl. *feature map*) sa sužavajućeg puta, te se događaju dvije 3x3 konvolucije gdje nakon svake slijedi prolazak kroz ReLU. Na posljednjem sloju se provodi još i 1x1 konvolucija za mapiranje broja kanala značajki na broj klasa prema [16].
- Mask R-CNN – model za segmentaciju instance. Ovaj model se sastoji od 2 faze. Prva faza se naziva mreža prijedloga regije (engl. *region proposal network*). Unutar nje se predlažu granični okviri (engl. *bounding box*) za objekte. U drugoj fazi se paralelno sa predviđanjem klase kojoj pripada objekt i pomaka predloženog graničnog okvira, pomak se pronalazi u svrhu poboljšanja predviđanja graničnog okvira, predviđa i binarna maska za svako područje interesa prema [17].
- DeepLab – postoji više verzija DeepLab-a prema [11]. Koristi se za semantičku segmentaciju. U verziji DeepLabv3+ koristi se arhitektura koder-dekoder. U koderu se događa izrazito crno prostorno piramidalno sužavanje (engl. *atrous spatial pyramid pooling*, ASPP) kako bi izvukla kontekstualna informacija na različitim skalama prema [18].

5. KORIŠTENJE ALGORITMA ŠIŠMIŠA KOD SEGMENTACIJE SLIKA

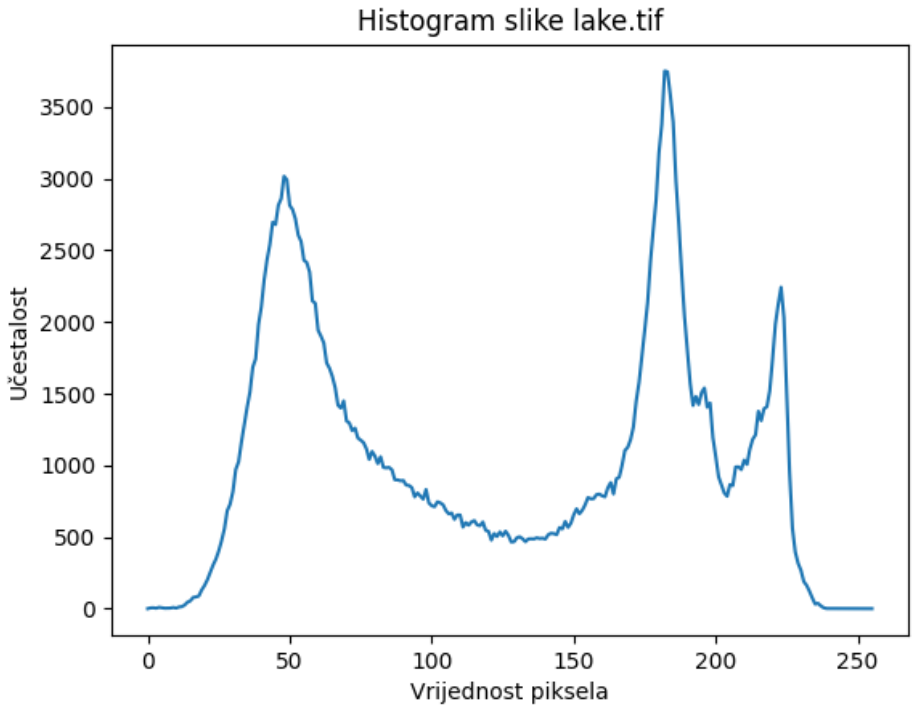
Unutar ovog poglavlja opisane su neke od mogućih upotreba algoritma šišmiša kod različitih tipova segmentacije slike. Opisane su varijante algoritma šišmiša korištene kod pojedinog načina segmentacije i korištene metode. Algoritam šišmiša u ovom radu se koristio kod segmentacije crno-bijelih slika, slika u boji te semantičke segmentacije.

Za segmentiranje crno-bijelih slika korištena je Otsuova metoda sa algoritmom šišmiša.

5.1. Otsuova metoda

Tvorac Otsuove metode je kako i ime nalaže N. Otsu te ju je predložio 1979. godine. Ova metoda koristi statistiku u svojim izračunima. Kod ove metode uzima se u obzir histogram crno-bijele slike.

Primjer histograma za sliku *lake.tif* prikazan je na slici 5.1.



Sl. 5.1. Histogram slike *lake.tif*

Iz ovoga se može vidjeti učestalost pojave piksela za pojedinu vrijednost piksela u rasponu [0, 255].

Vjerojatnost pojave piksela za određenu vrijednost dana je izrazom:

$$p_i = n_i/N \tag{5-1}$$

gdje p_i predstavlja vjerojatnost pojave piksela koji ima i vrijednost, n_i predstavlja broj piksela koji imaju i vrijednost i N predstavlja ukupan broj piksela.

Ako se slika želi razdvojiti na 2 klase, na pozadinu i objekte i neka vrijednosti piksela za pozadinu idu od 0 do vrijednosti k koja predstavlja graničnu vrijednost, a pikseli za objekte idu od $k + 1$ do L, tada se vjerojatnosti pojave piksela za pozadinu i objekte koje su potrebne u daljnjim izračunima mogu izračunati prema izrazima:

$$\omega_{POZ} = \frac{\sum_{i=0}^{k} n_i}{N} \tag{5-2}$$

$$\omega_{OBJ} = \frac{\sum_{i=k+1}^{L} n_i}{N} \tag{5-3}$$

gdje ω_{POZ} predstavlja vjerojatnost pojave piksela pozadine, ω_{OBJ} predstavlja vjerojatnost pojave piksela objekata, n_i predstavlja broj piksela koji imaju i vrijednost, N predstavlja ukupan broj piksela slike, k predstavlja graničnu vrijednost i L predstavlja maksimalnu vrijednost piksela. N, k i L predstavljaju isto u daljnjim izrazima vezanim za Otsuovu metodu pa nisu objašnjeni tamo.

Nakon toga je potrebno izračunati srednje vrijednosti piksela pozadine i objekata na način:

$$\mu_{POZ} = \frac{\sum_{i=0}^{k} i * n_i}{\sum_{i=0}^{k} n_i} \tag{5-4}$$

$$\mu_{OBJ} = \frac{\sum_{i=k+1}^{L} i * n_i}{\sum_{i=k+1}^{L} n_i} \tag{5-5}$$

gdje μ_{POZ} predstavlja srednju vrijednost piksela pozadine, μ_{OBJ} predstavlja srednju vrijednost piksela objekata, i predstavlja vrijednost piksela i n_i predstavlja broj piksela koji imaju i vrijednost.

Nakon toga se mogu izračunati varijance za obje klase prema izrazima:

$$\sigma_{POZ}^2 = \frac{\sum_{i=0}^{k} n_i * (i - \mu_{POZ})^2}{\sum_{i=0}^{k} n_i} \tag{5-6}$$

$$\sigma_{OBJ}^2 = \frac{\sum_{i=k+1}^{L} n_i * \left(i - \mu_{OBJ}\right)^2}{\sum_{i=k+1}^{L} n_i} \tag{5-7}$$

gdje σ_{POZ}^2 predstavlja varijancu vrijednosti piksela pozadine, σ_{OBJ}^2 predstavlja varijancu vrijednosti piksela objekata, *i* predstavlja vrijednost piksela, n_i predstavlja broj piksela koji imaju *i* vrijednost, μ_{POZ} i μ_{OBJ} predstavljaju srednje vrijednosti piksela pozadine i objekata slijedno.

Nakon što su izračunate varijance za klase potrebno je izračunati vrijednosti neke od metrika koje pokazuju koliko je izabrana granična vrijednost dobra za segmentaciju slike, te metrike su: unutar-klasna varijanca (engl. *within-class variance*) i između-klasna varijanca.

Izraz za izračun unutar-klasne varijance je:

$$\sigma_W^2 = \omega_{POZ} * \sigma_{POZ}^2 + \omega_{OBJ} * \sigma_{OBJ}^2 \tag{5-8}$$

gdje σ_W^2 predstavlja unutar-klasnu varijancu, ω_{POZ} predstavlja vjerojatnost pojave piksela pozadine, σ_{POZ}^2 varijancu vrijednosti piksela pozadine, ω_{OBJ} predstavlja vjerojatnost pojave piksela objekata i σ_{OBJ}^2 varijancu vrijednosti piksela objekata.

Izraz za izračun između-klasne varijance je:

$$\begin{aligned} \sigma_B^2 &= \omega_{POZ} * (\mu_{POZ} - \mu_{SLIKE})^2 + \omega_{OBJ} * \left(\mu_{OBJ} - \mu_{SLIKE}\right)^2 \\ &= \omega_{POZ} * \omega_{OBJ} * (\mu_{OBJ} - \mu_{POZ})^2 \end{aligned} \tag{5-9}$$

gdje σ_B^2 predstavlja između-klasnu varijancu, ω_{POZ} predstavlja vjerojatnost pojave piksela pozadine, ω_{OBJ} predstavlja vjerojatnost pojave piksela objekata, μ_{POZ} srednju vrijednost piksela pozadine, μ_{OBJ} srednju vrijednost piksela objekata i μ_{SLIKE} predstavlja srednju vrijednost piksela slike.

μ_{SLIKE} se računa prema izrazu:

$$\mu_{SLIKE} = \frac{\sum_{i=0}^{L} i * n_i}{N} \tag{5-10}$$

gdje *i* predstavlja vrijednost piksela i n_i predstavlja broja piksela koji imaju *i* vrijednost.

Za unutar-klasnu varijancu vrijedi što je manja njena vrijednost to je bolje odabrana granična vrijednost, dok za između-klasnu varijancu vrijedi što je veća njena vrijednost to je bolje odabrana granična vrijednost.

Otsuova metoda može se koristiti i za više graničnih vrijednosti gdje onda postoji i više od dvije klase piksela slike. Princip je isti kao u slučaju korištenja jedne granične vrijednosti pa je tako potrebno izračunati vrijednosti za: vjerojatnost pojave piksela, srednju vrijednost piksela i varijancu vrijednosti piksela (varijance je potrebno izračunati u slučaju korištenja unutar-klasne varijance kao metrike) za svaku klasu.

Izraz za računanje unutar-klasne varijance za više od dvije klase izgleda ovako:

$$\sigma_W^2 = \sum_{i=1}^{n} \omega_i * \sigma_i^2 \tag{5-11}$$

gdje σ_W^2 predstavlja unutar-klasnu varijancu, *n* predstavlja ukupan broj klasa, ω_i predstavlja vjerojatnost pojave piksela *i*-te klase i σ_i^2 predstavlja varijancu vrijednosti piksela *i*-te klase.

Izraz za računanje između-klasne varijance za više od dvije klase izgleda ovako:

$$\sigma_B^2 = \sum_{i=1}^{n} \omega_i * (\mu_i - \mu_{SLIKE})^2 \tag{5-12}$$

gdje σ_B^2 predstavlja između-klasnu varijancu, *n* predstavlja ukupan broj klasa, ω_i predstavlja vjerojatnost pojave piksela *i*-te klase, μ_{i} predstavlja srednju vrijednost piksela *i*-te klase i μ_{SLIKE} predstavlja srednju vrijednost piksela slike prema [19].

Funkcija cilja za Otsuovu metodu koja se predaje algoritmu šišmiša u ovom radu računa unutar-klasnu varijancu, a njoj se predaju pozicije šišmiša kao granične vrijednosti prema kojima se dijeli slika na segmente.

5.2. Varijante algoritma šišmiša korištene kod segmentacije slika

Unutar ovog potpoglavlja opisane su različite varijante algoritma šišmiša te kako i za koju vrstu segmentacije slike su korištene.

Za crno-bijelu segmentaciju slike korištena je Otsuova metoda sa standardnim algoritmom šišmiša i još sljedeće dvije varijante:

1. kaotični algoritam šišmiša
2. algoritam šišmiša sa inercijskom težinom (engl. *bat algorithm with inertia weight*)

Za segmentaciju obojanih slika korišten je algoritam šišmiša koji koristi *K-means* te za semantičku segmentaciju korišten je algoritam šišmiša zajedno sa konvolucijskom neuronskom mrežom (engl.

convolutional neural network, CNN) gdje se algoritam šišmiša koristi za optimizaciju hiperparametara neuronske mreže.

5.2.1. Kaotični algoritam šišmiša

Kaotični algoritam šišmiša (CBA) koristi kaotične mape (engl. *chaotic maps*) za generiranje nasumičnih vrijednosti. Kaotične mape su matematičke funkcije koje mogu proizvesti naizgled nasumičnu sekvencu slučajnih brojeva odnosno kaotičnih brojeva. Ove funkcije koriste rezultat prijašnjeg pozivanja funkcije kao parametar u trenutnom pozivu prilikom čega je potrebno postaviti inicijalnu vrijednost koja će se predati prilikom prvog poziva funkcije. Ove funkcije obično sadržavaju i kontrolne parametre. Može se dogoditi da kaotična mapa za neke raspone početne vrijednosti ili kontrolnog parametra ne daje kaotičnu sekvencu. Postoji više različitih kaotičnih mapa. Mape imaju određen broj dimenzija sa kojima rade prema [20].

U slučaju kaotičnog algoritma šišmiša korištenog u ovom radu, radi se samo s jednom dimenzijom što se tiče kaotičnih mapa (ne misli se na broj dimenzija problema za funkciju cilja). Za kaotični algoritam šišmiša korišten u ovom radu uspoređivane su 3 različite kaotične mape, a one su sljedeće:

1. logistička (engl. *logistic*) kaotična mapa
2. sinusna (engl. *sine*) kaotična mapa
3. kaotična *tent* mapa

Prema [21] izraz za logističku mapu izgleda ovako:

$$x_{i+1} = r * x_i * (1 - x_i) \tag{5-13}$$

gdje x_{i+1} predstavlja sljedeću vrijednost u sekvenci, x_i predstavlja trenutnu vrijednost i r predstavlja kontrolni parametar.

Prema [22] izraz za sinusnu mapu izgleda ovako:

$$x_{i+1} = a * sin(\pi * x_i) \tag{5-14}$$

gdje x_{i+1} predstavlja sljedeću vrijednost u sekvenci, x_i predstavlja trenutnu vrijednost i a predstavlja kontrolni parametar.

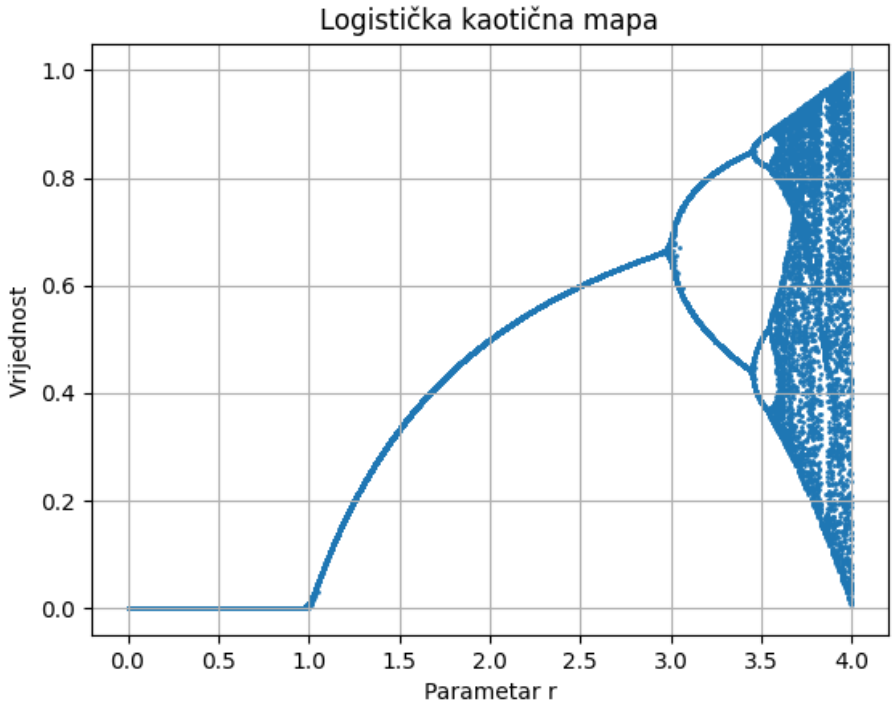
Prema [23] izraz za *tent* mapu izgleda ovako:

$$x_{i+1} = \begin{cases} \mu * x_i, & x_i < 0{,}5 \\ \mu * (1 - x_i), & x_i \geq 0{,}5 \end{cases} \tag{5-15}$$

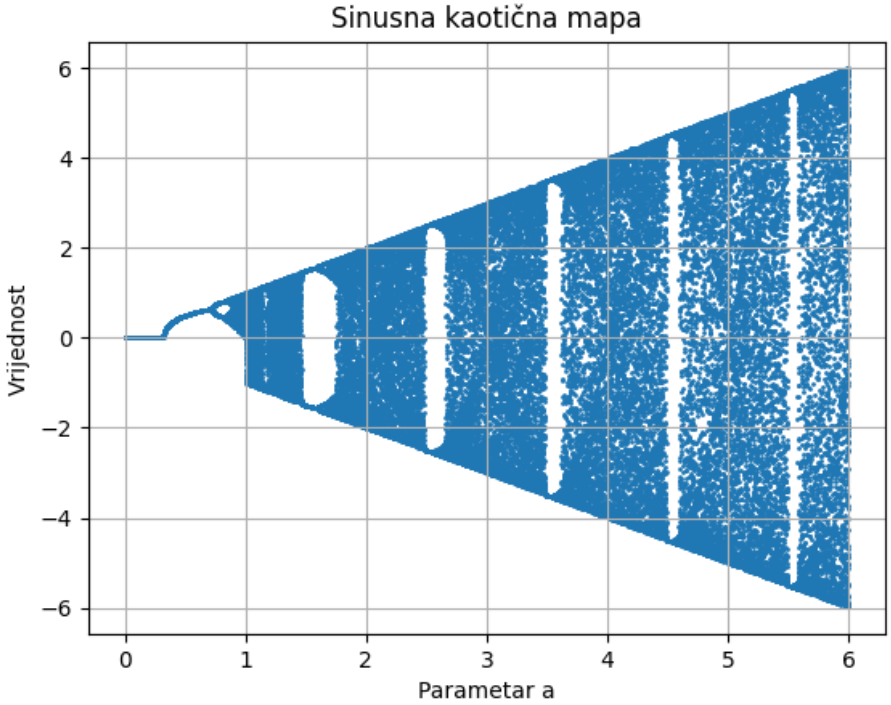
gdje x_{i+1} predstavlja sljedeću vrijednost u sekvenci, x_i predstavlja trenutnu vrijednost i μ predstavlja kontrolni parametar. Ovdje se množi sa kontrolnim parametrom minimalna vrijednost između x_i i $(1 - x_i)$.

Sljedeće su prikazani grafovi funkcija koji pokazuju kako se mijenjaju vrijednosti ovisno o promjeni kontrolnog parametra. Za svaku vrijednost kontrolnog parametra vrši se 100 iteracija funkcije sa tim parametrom te se uzima krajnji rezultat tih iteracija. Za početne vrijednosti su uzeti nasumični brojevi.

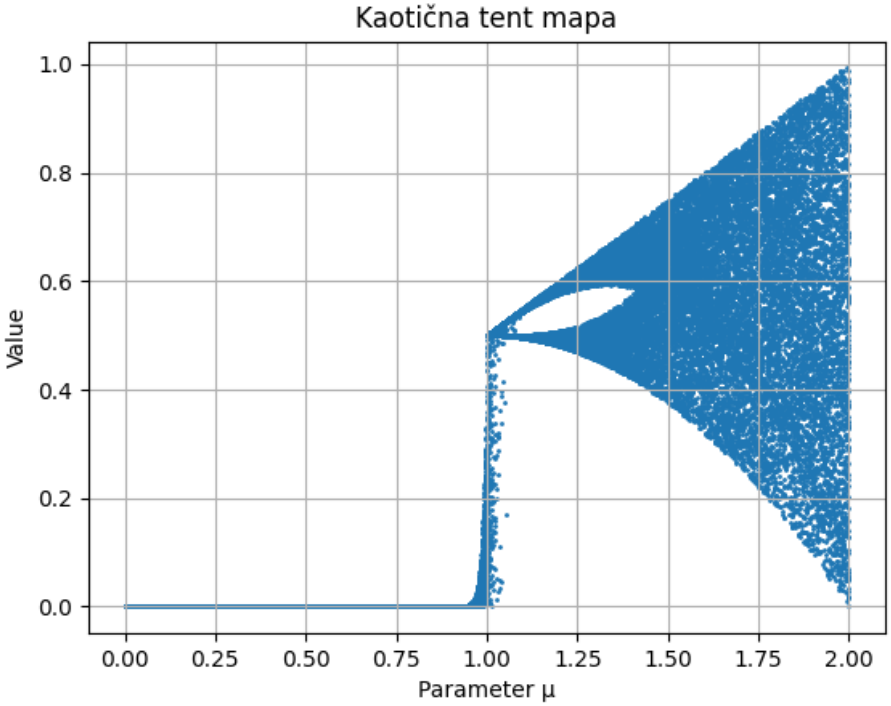
Prikaz kaotičnih mapa gdje se prikazuje ovisnost vrijednosti o kontrolnom parametru dan je na slikama 5.2., 5.3. i 5.4.



Sl. 5.2. Logistička kaotična mapa sa različitim vrijednostima kontrolnog parametra



Sl. 5.3. Sinusna kaotična mapa sa različitim vrijednostima kontrolnog parametra

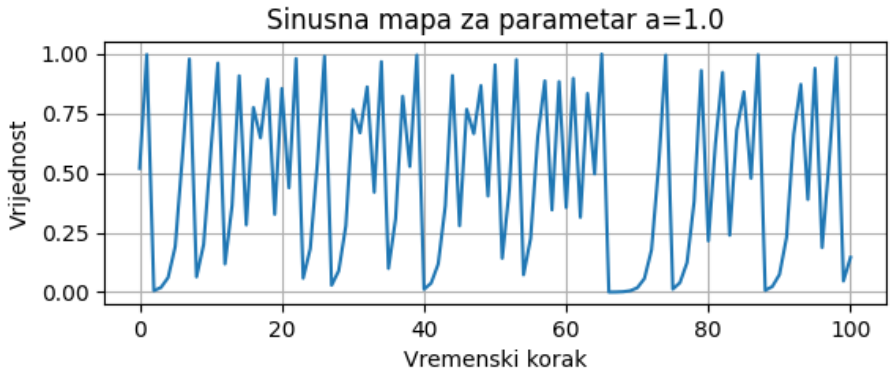


Sl. 5.4. *Tent* mapa sa različitim vrijednostima kontrolnog parametra

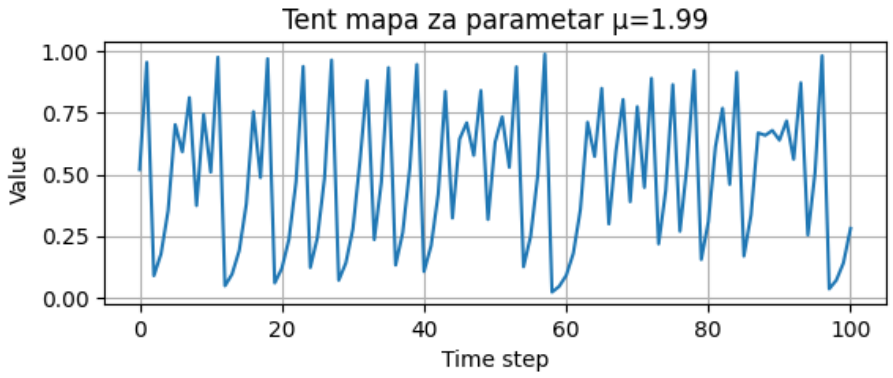
Može se vidjeti kako za određene vrijednosti kontrolnog parametra se ne generiraju kaotični brojevi. Također se mogu vidjeti prazni prostori unutar kaotičnih područja i može se vidjeti kako se vrijednosti granaju na određenim dijelovima (za sinusnu mapu se ne vide sva mjesta grananja najbolje zbog veličine slike i gustoće vrijednosti). Iznad vrijednosti 4 za kontrolni parametar kod logističke funkcije vrijednosti ogromno rastu u negativnu beskonačnost, stoga se to ne prikazuje na grafu.

Sljedeće su prikazane promjene vrijednosti kroz iteracije uz konstantnu vrijednost kontrolnog parametra za te 3 kaotične mape što je prikazano na slikama 5.5, 5.6 i 5.7. Za logističku mapu *r* je jednako 3,99, za sinusnu mapu *a* je jednako 1, za *tent* mapu μ je jednako 1,99, a inicijalna je vrijednost za sve 3 mape jednaka 0,52.

Sl. 5.5. Generirana sekvenca logističke kaotične mape



Sl. 5.6. Generirana sekvenca sinusne kaotične mape



Sl. 5.7. Generirana sekvenca kaotične *tent* mape

Kao što se može vidjeti sekvence generirane pomoću ovih mapa djeluju nasumično te se nalaze u rasponu (0, 1).

A. H. Gandomi i X. S. Yang su razvili 4 različite verzije kaotičnog algoritma šišmiša. Unutar ovog rada korištena je verzija gdje se koristi kaotična mapa za generiranje vrijednosti frekvencije šišmiša.

Prema tome izraz (3-1) za generiranje frekvencije šišmiša je modificiran na način:

$$f_i = f_{min} + (f_{max} - f_{min}) * CM_i \tag{5-16}$$

gdje kao što je ranije spomenuto f_i predstavlja frekvenciju *i*-tog šišmiša, f_{min} predstavlja minimalnu vrijednost koju frekvencija može poprimiti, f_{max} predstavlja maksimalnu vrijednost koju frekvencija može poprimiti, a jedina razlika je što se sada koristi CM_i koji predstavlja kaotični broj u rasponu (0, 1) iz kaotične sekvence za *i*-tog šišmiša dok je se prije koristio nasumični broj iz uniformne distribucije u rasponu [0, 1). Svaki šišmiš ima svoju vrijednost iz kaotične sekvence i tu vrijednost koristi prilikom generiranja nove koja se zatim postavlja na njenu vrijednost prema [24].

5.2.2. Algoritam šišmiša s inercijskom težinom

Z. Cui, F. Li i Q. Kang su osmislili algoritam šišmiša sa inercijskom težinom gdje se primjenjuje određena inercijska težina na prethodnu brzinu prilikom ažuriranja brzine šišmiša kako bi se povećala mogućnost lokalnog traženja. Ta strategija se inače primjenjuje kod PSO algoritma.

Prema tome izraz za ažuriranje brzine šišmiša (3-2) je modificiran na način:

$$v_i{}^{t+1} = v_i{}^t * w_i{}^t + (x_i{}^t - x_*) * f_i \tag{5-17}$$

gdje kao što je već ranije spomenuto ${v_i}^{t+1}$ predstavlja ažuriranu brzinu *i*-tog šišmiša, ${v_i}^{t}$ predstavlja trenutnu brzinu *i*-tog šišmiša, ${x_i}^{t}$ predstavlja trenutnu lokaciju *i*-tog šišmiša, f_i predstavlja frekvenciju *i*-tog šišmiša, x_* predstavlja lokaciju šišmiša koji ima najbolju lokaciju, a jedina razlika je pojava parametra ${w_i}^{t}$ koji predstavlja vrijednost poziva funkcije određene inercijske težine u *t* iteraciji za *i*-tog šišmiša prema [25].

Prema [26] postoje razne funkcije inercijskih težina, no u ovom radu u eksperimentima su uspoređivane sljedeće:

1. konstantna inercijska težina
2. nasumična inercijska težina
3. linearno padajuća inercijska težina
4. inercijska težina simuliranog žarenja
5. strategija prirodne eksponencijalne inercijske težine (e1-PSO)
6. strategija prirodne eksponencijalne inercijske težine (e2-PSO)
7. logaritamski padajuća inercijska težina

Izraz za konstantnu inercijsku težinu je sljedeći:

$$w = c \tag{5-18}$$

gdje je *c* konstanta.

Izraz za nasumičnu inercijsku težinu je sljedeći:

$$w = 0{,}5 + rand/2 \tag{5-19}$$

gdje je *rand* nasumični broj u rasponu [0, 1) iz uniformne distribucije.

Izraz za linearno padajuću inercijsku težinu je sljedeći:

$$w(t) = w_{max} - \frac{w_{max} - w_{min}}{t_{max}} * \ t \tag{5-20}$$

gdje *t* predstavlja trenutnu iteraciju, t_{max} predstavlja maksimalni broj iteracija, w_{max} predstavlja gornju graničnu vrijednost, w_{min} predstavlja donju graničnu vrijednost.

Oznake u ovom izrazu predstavljaju isto i u sljedećim inercijskim težinama.

Izraz za inercijsku težinu simuliranog žarenja je sljedeći:

$$w(t) = w_{min} + (w_{max} - w_{min}) * \lambda^{t-1} \tag{5-21}$$

gdje λ predstavlja faktor koji utječe na brzinu smanjenja vrijednosti.

Izraz za strategiju prirodne eksponencijalne inercijske težine (e1-PSO) je sljedeći:

$$w(t) = w_{min} + (w_{max} - w_{min}) * \mathrm{e}^{-(\frac{10*t}{t_{max}})} \tag{5-22}$$

Izraz za strategiju prirodne eksponencijalne inercijske težine (e2-PSO) je sljedeći:

$$w(t) = w_{min} + (w_{max} - w_{min}) * \mathrm{e}^{-(\frac{4*t}{t_{max}})^2} \tag{5-23}$$

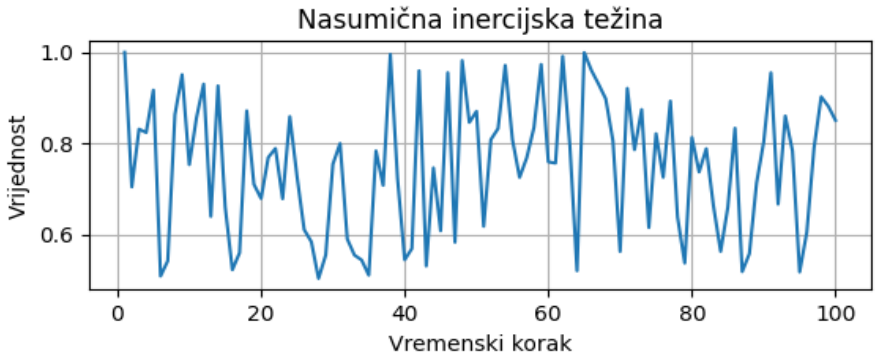
Izraz za logaritamski padajuću inercijsku težinu je sljedeći:

$$w(t) = w_{max} + (w_{min} - w_{max}) * log_{10}(a + \frac{10 * t}{t_{max}}) \tag{5-24}$$

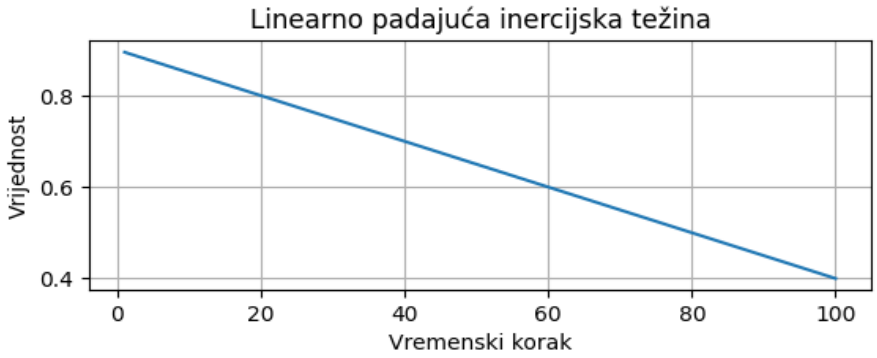
gdje *a* predstavlja faktor koji utječe na raspon vrijednosti.

Na slikama 5.8.-5.14. prikazane su funkcije inercijskih težina sa sljedećim parametrima: *c* je jednako 0,7 za konstantnu inercijsku težinu, w_{min} je jednako 0,4, a w_{max} je jednako 0,9 za inercijske težine koje koriste te parametre, λ je jednako 0,95 za inercijsku težinu simuliranog žarenja i *a* je jednako 1 za logaritamski padajuću inercijsku težinu.

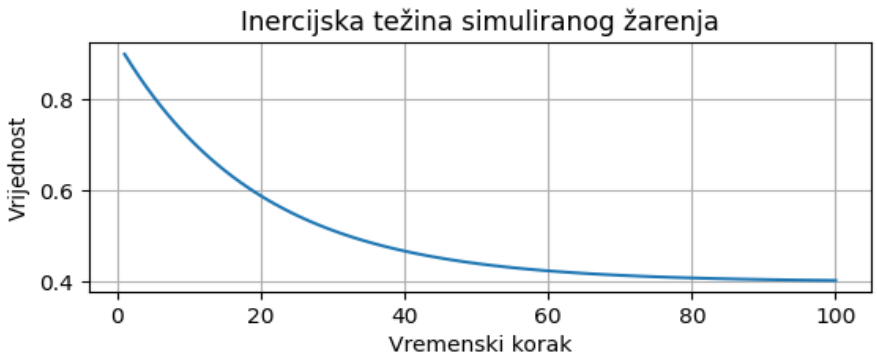
Sl. 5.8. Konstantna inercijska težina



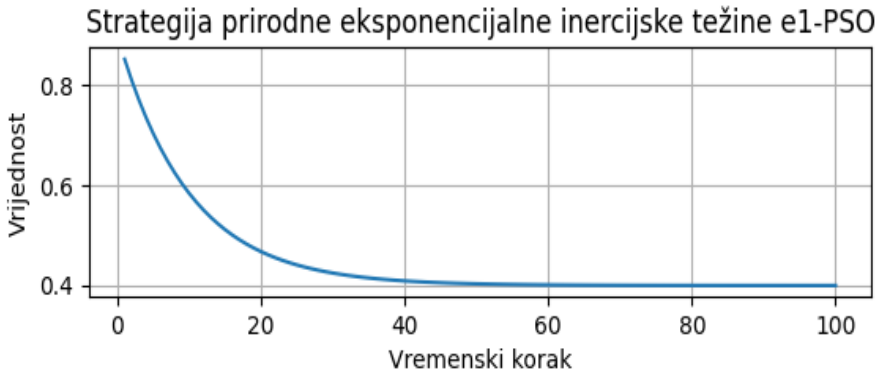
Sl. 5.9. Nasumična inercijska težina



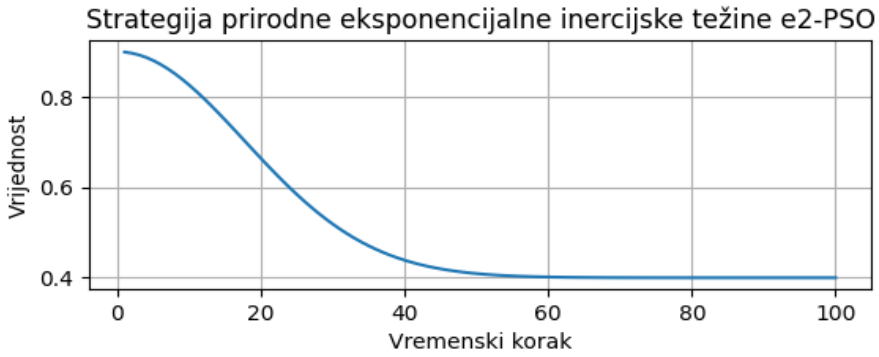
Sl. 5.10. Linearno padajuća inercijska težina



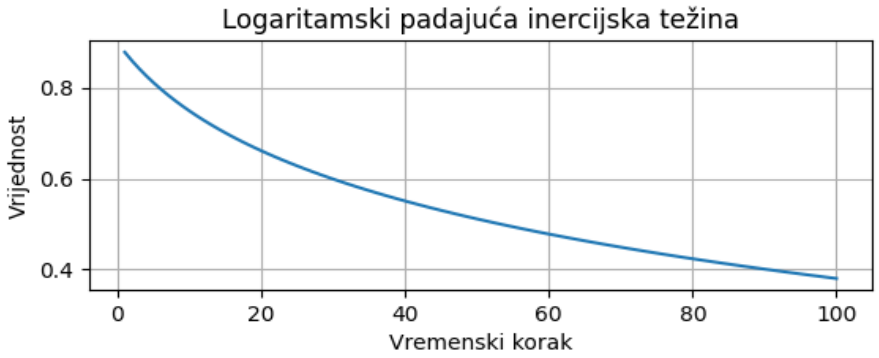
Sl. 5.11. Inercijska težina simuliranog žarenja



Sl. 5.12. Strategija prirodne eksponencijalne inercijske težine (e1-PSO)



Sl. 5.13. Strategija prirodne eksponencijalne inercijske težine (e2-PSO)



Sl. 5.14. Logaritamski padajuća inercijska težina

5.2.3. Algoritam šišmiša koji koristi *K-means*

Poboljšani algoritam šišmiša (engl. *improved bat algorithm*) koji koristi *K-means* su predložili M. Sujaritha i ostali. Unutar njega se koristi *K-means* kod inicijalizacije početnih lokacija šišmiša prema [27]. Stoga je u ovom radu implementiran algoritam šišmiša sa inicijalizacijom početnih vrijednosti lokacija šišmiša pomoću *K-meansa* gdje središta grupacija dohvaćena po završetku *K-meansa* predstavljaju početne lokacije šišmiša i dohvaćeni *fitnessi* njihove početne *fitnesse*.

Pseudokôd za *K-means* algoritam nalazi se na slici 5.15.

Linija ***Kod***

```
1:      Definiraj broj grupacija k
2:      Inicijaliziraj središta grupacija nasumično
        dokle god nije došlo do konvergencije ili maksimalni broj iteracija
3:      nije dostignut čini
           Za svaku točku podataka pronađi najbliže središte grupacije i
4:         pridruži ga toj grupaciji
5:         Za svaku grupaciju pronađi novo središte
6:      kraj
```

Sl. 5.15. Pseudokôd *K-means* algoritma

Kod *K-meansa* se iterativno pridružuju točke podataka određenoj grupaciji na temelju Euklidske udaljenosti te se ponovo računaju središta grupacija na temelju srednje vrijednosti svih točaka unutar te grupacije prema [28].

Nakon što su lokacije šišmiša inicijalizirane pomoću *K-meansa*, šišmiši nastoje pronaći bolja rješenja minimiziranjem funkcije cilja koja vraća sumu kvadratnih udaljenosti svake točke podataka od njihovih najbližih središta grupacija tj. sumu kvadriranih ostataka (engl. *sum of squared residuals*), što je prikazano izrazom:

$$y = \sum_{k=1}^{K} \sum_{\forall x_i \in C_k} ||x_i - \mu_k||^2 \tag{5-25}$$

Gdje *K* označava broj grupacija, C_k označava *k*-tu grupaciju, x_i označava podatak u prostoru i μ_k označava središte *k*-te grupacije prema [29].

5.2.4. Algoritam šišmiša za optimizaciju hiperparametara konvolucijske neuronske mreže

Postoje različite strategije za optimizaciju hiperparametara kod neuronskih mreža. Neke od poznatijih su mrežno traženje (engl. *grid search*) i nasumično traženje (engl. *random search*). Kod mrežnog traženja hiperparametri predstavljaju mrežu vrijednosti koja predstavlja sve moguće kombinacije hiperparametara koje se mogu uzeti u obzir te se isprobava svaki čvor odnosno kombinacija hiperparametara. Tijekom isprobavanja kombinacija one se bilježe te se odabire ona koja je najbolja. Ova strategija je vremenski zahtjevna jer je potrebno isprobati sve kombinacije, stoga broj isprobavanja kao i vrijeme eksponencijalno raste sa povećanjem broja hiperaparametara. Kod nasumičnog traženja kao i što samo ime govori uzimaju se nasumične kombinacije hiperparametara u određenom rasponu kako bi se pronašla najbolja kombinacija. Postoji određeni broj iteracija koje se izvode. Ova strategija može prije naći najbolju kombinaciju hiperaparametara jer se isprobavaju kombinacije sa većim razlikama vrijednosti u manjem broju iteracija prema [30].

I. Strumberger i ostali su dizajnirali model konvolucijske neuronske mreže korištenjem algoritma krijesnica (engl. *firefly algorithm*, FA) gdje se algoritam krijesnica koristi za optimizaciju hiperparametara CNN prema [31]. U ovom radu je implementiran mogući način korištenja algoritma šišmiša za optimizaciju hiperparametara CNN za semantičku segmentaciju.

Algoritam šišmiša iterativno trenira neuronsku mrežu i na temelju vraćene vrijednosti funkcije cilja pronalazi bolja rješenja. Ova tehnika je prilično računalno i vremenski zahtjevna zato što je potrebno za svakog šišmiša unutar svake iteracije kao i kod inicijalizacije ponovno istrenirati

model. Algoritam optimizira hiperparametre: brzinu učenja (engl. *learning rate*) i veličinu skupine (engl. *batch size*). Veličina skupine predstavlja broj uzoraka koji su zajedno predani modelu, a brzina učenja je objašnjena nešto kasnije. Za vrijeme treniranja i validacije modela kod svake epohe prvo se trenira model na trening skupu. Zatim se provjerava je li taj model bolji od trenutnog najboljeg prema povratnoj vrijednosti funkcije gubitka (engl. *loss function*) na validacijskom skupu, ako jest ažurira se najbolji model i najmanja vrijednost funkcije gubitka. Epoha predstavlja jednu iteraciju treniranja čitavog trening skupa i validacije čitavog skupa za validaciju. Najbolja vrijednost funkcije gubitka (najmanja) vraća se kao rezultat funkcije cilja kod algoritma šišmiša, a on je nastoji minimizirati pronalazeći bolja rješenja. Broj epoha do koje se trenira može se odrediti prilikom inicijalizacije ovog algoritma.

Kod eksperimenata odabran je jedan od implementiranih modela unutar Pytorch biblioteke pod nazivom: potpuno konvolucijska neuronska mreža sa ResNet 50 okosnicom (engl. *fully-convolutional neural network with a ResNet50 backbone*). Potpuno konvolucijska neuronska mreža nema potpuno povezane (engl. *fully-connected*) slojeve nego su oni transformirani u konvolucijske slojeve prema [32]. Okosnica služi za vađenje značajki iz podataka i njihovo kodiranje prema [33]. Postoje kompleksniji modeli konvolucijskih neuronskih mreža u biblioteci Pytorch koji daju bolje rezultate treniranja u prosjeku, ali također zahtijevaju i duže vrijeme treniranja pa je zbog toga odabran postojeći model za optimizaciju kako bi se smanjilo vrijeme treniranja.

ResNet50 koristi 50 rezidualnih blokova uskog grla (engl. *bottleneck residual blocks*) koji se sastoje od 1x1 konvolucijskog sloja gdje se smanjuje broj kanala, zatim se može koristiti 3x3 konvolucija, poslije toga slijedi opet 1x1 konvolucija gdje se vraća originalni broj kanala. Nakon toga slijedi preskočna veza (engl. *skip connection*) gdje se dodaje ulaz odnosno slika prije konvolucije na izlaz iz konvolucijskih slojeva. Nakon svakog konvolucijskog sloja slijedi normalizacije skupine (engl. *batch normalization*), a ReLU slijedi nakon normalizacije skupine od prva dva konvolucijska sloja i nakon preskočne veze prema [34].

Korišteni optimizator je stohastičko gradijentno spuštanje sa momentom (engl. *stochastic gradient descent with momentum,* SGD *w/ momentum*). Gradijentno spuštanje radi na način da se iterativno pronalaze nagibi funkcije za svaki parametar odnosno pronalazi se gradijent, zatim se računa veličina koraka pomoću gradijenta i brzine učenja te se ažuriraju parametri prema veličini koraka i prethodnoj vrijednosti parametara sve dok gradijent nije blizu 0. Brzina učenja utječe na rad ovog algoritma jer ona određuje skaliranje kod veličine koraka. Stohastičko gradijentno spuštanje uzima

nasumično jedan podatak ili skupinu njih za računanje parcijalne derivacije (nagiba) kako bi se drastično smanjilo vrijeme izvršavanja i kompleksnost prema [35]. Moment omogućuje brže kretanje vektora gradijenata u pravim smjerovima što rezultira bržoj konvergenciji jer prije dolazi do globalnog minimuma.

Izraz za računanje momenta kod parametra x_1 je sljedeći:

$$v_{t+1} = \beta * v_t + (1 - \beta) * \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_1} \tag{5-26}$$

gdje v_{t+1} predstavlja ažurirani moment, v_t predstavlja trenutni moment, β predstavlja hiperparametar koji utječe na težinu momenta i $f(x_1, x_2, \dots, x_n)$ predstavlja funkciju cilja sa parametrima $x_1, x_2, \dots, x_n$.

Stoga se nova vrijednost parametra računa prema izrazu:

$$x_{1_{t+1}} = x_{1_t} - \alpha * v_{t+1} \tag{5-27}$$

gdje $x_{1_{t+1}}$ predstavlja ažuriranu vrijednost parametra x_1, x_{1_t} predstavlja trenutnu vrijednost parametra x_1, α predstavlja brzinu učenja i v_{t+1} predstavlja ažurirani moment prema [36].

Korištena funkcija gubitka je unakrsno-entropijski gubitak (engl. *cross-entropy loss*) koji računa razlike između vjerojatnosti stvarne i predviđene vrijednosti na način:

$$H(p, q) = -\sum_{x \in klase} p(x) * log_{10}\, q(x) \tag{5-28}$$

gdje $H(p, q)$ predstavlja unakrsnu entropiju, *x* predstavlja klasu, *p(x)* predstavlja stvarnu vjerojatnost za pojavu klase (1 ako jest ta klasa, 0 ako nije) i *q(x)* predstavlja predviđenu vjerojatnost za pojavu klase. Cilj je smanjiti rezultat unakrsne entropije. Kod multi-klasne unakrsne entropije predviđene vjerojatnosti za pojedinu klasu su dobivene nakon što je izlaz iz neuronske mreže proveden kroz *SoftMax* funkciju koja mapira vrijednosti izlaza iz mreže na vjerojatnosti prema [37].

6. EKSPERIMENTALNI REZULTATI I MJERENJA

U ovom poglavlju prikazani su rezultati mjerenja koliko dobro algoritam šišmiša i njegove varijante obavljaju zadatke u segmentiranju slika. Mjereni su na crno-bijelim, obojanim slikama i na semantičkoj segmentaciji.

Računate su sljedeće metrike za mjerenje sličnosti slika kod crno-bijelih i obojanih slika:

- korijen srednje kvadratne pogreške (engl. *root mean square error*, RMSE)
- vršni omjer signala i šuma (PSNR)
- metoda indeksa strukturne sličnosti (SSIM)

RMSE mjeri razlike u vrijednostima po pikselu. Što je manja njegova vrijednost to je bolje. Kada je 0 znači da su slike identične. Računa se prema izrazu:

$$RMSE = \sqrt{MSE} \tag{6-1}$$

MSE predstavlja srednju kvadratnu pogrešku i računa se prema izrazu:

$$MSE = \frac{1}{M * N} * \sum_{i=1}^{M} \sum_{j=1}^{N} [I(i,j) - K(i,j)]^2 \tag{6-2}$$

gdje *I* predstavlja originalnu sliku, *K* predstavlja modificiranu sliku, *M* predstavlja broj redaka, a *N* predstavlja broj stupaca slike koji su jednaki za obje slike.

PSNR računa omjer najveće moguće vrijednosti signala i vrijednosti šuma koja utječe na taj signal. Izražava se u decibelima. Računa se prema izrazu:

$$PSNR = 10 * log_{10}(\frac{R^2}{MSE}) \tag{6-3}$$

gdje *R* predstavlja maksimalnu moguću vrijednost signala u ovom slučaju je to vrijednost piksela, a *MSE* predstavlja srednju kvadratnu pogrešku. Što je veća vrijednost PSNR-a to je bolje.

SSIM računa pad kvalitete slike zbog posljedica procesiranja kao što je npr. kompresija. Mjeri razliku na slikama koja se može uvidjeti očima prema [38]. SSIM uzima u obzir osvjetljenje, kontrast i strukturu. Vrijednosti za SSIM se nalaze u rasponu [-1, 1] gdje 1 znači da su slike vrlo slične ili iste, a -1 da su slike vrlo različite. Izraz za SSIM kod dvaju slika je sljedeći:

$$SSIM(x,y) = \frac{(2 * \mu_x * \mu_y + C_1) * (2 * \sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) * (\sigma_x^2 + \sigma_x^2 + C_2)} \tag{6-4}$$

gdje x predstavlja prvu sliku, y predstavlja drugu sliku, μ_x predstavlja srednju vrijednost od x, μ_y predstavlja srednju vrijednost od y, σ_x predstavlja standardnu devijaciju od x, σ_y predstavlja standardnu devijaciju od y, σ_{xy} predstavlja kovarijancu od x i y i na kraju C_1 i C_2 se računaju prema izrazima:

$$C_1 = (k_1 * L)^2 \tag{6-5}$$

$$C_2 = (k_2 * L)^2 \tag{6-6}$$

gdje k_1 i k_2 predstavljaju konstante, a L predstavlja dinamički raspon vrijednosti piksela (obično 255) prema [39].

Kod semantičke segmentacije evaluiran je model neuronske mreže istreniran pomoću algoritma šišmiša te su prikazane segmentirane slike pomoću njega.

6.1. Segmentacija crno-bijelih slika

Za segmentaciju crno-bijelih slika korištena je Otsuova metoda sa funkcijom cilja unutar-klasnom varijancom (što je manja njena vrijednost to je bolje). Mjerene su 3 varijante algoritma šišmiša za pronalazak najboljih graničnih vrijednosti (pragova), a to su: standardni, kaotični i algoritam šišmiša sa inercijskom težinom. Granične vrijednosti su intenziteti piksela. Prvo su uspoređivane različite kaotične mape kod CBA za odabir najbolje. Zatim su uspoređivane inercijske težine kod BA sa inercijskom težinom u istu svrhu. Nakon toga su uspoređivane sve tri varijante BA.

Slike koje su korištene za mjerenja u ovom potpoglavlju su *walkbridge.tif*, *lake.tif* i *peppers_gray.tif*, a one su prikazane na slici 6.1.

Sl. 6.1. Testne slike za segmentaciju crno-bijelih slika [40]

Parametri koji su jednaki za sve tri varijante BA imaju iste sljedeće vrijednosti:

- broj šišmiša: 50
- broj iteracija: 10
- minimalna pozicija: 0
- maksimalna pozicija: 255
- minimalna frekvencija: 0
- maksimalna frekvencija: 1
- σ: 0,1
- γ: 0,1
- α: 0,97

Svaki je algoritam kod mjerenja u ovom potpoglavlju pokrenut 100 puta gdje je svaki put drugačija sekvenca nasumičnosti te je uzeto srednje vrijeme trajanja i srednji rezultat *fitnessa*.

Za odabir odgovarajuće kaotične mape kod kaotičnog algoritma šišmiša provedeno je testiranje za 3 različite mape, a to su: logistička, sinusna i *tent* mapa. Za logističku mapu *r* je jednako 3,99, za sinusnu mapu *a* je jednako 1 i za *tent* mapu μ je jednako 1,99. Odabrana slika je *walkbridge.tif*, a broj dimenzija odnosno broj graničnih vrijednosti je postavljen na 5.

Rezultati za *fitness* vrijednosti su prikazani u tablici 6.1.

Tablica 6.1. Rezultati *fitnessa* kod mjerenja CBA sa različitim kaotičnim mapama

Kaotična mapa	***Fitness* vrijednost**
Logistička	142,2846
Sinusna	141,6938
Tent	135,0029

Na kraju je odabrana *tent* mapa za daljnja mjerenja zbog najmanje vrijednosti *fitnessa*.

Za odabir odgovarajuće funkcije inercijske težine kod algoritma šišmiša sa inercijskom težinom provedeno je testiranje za 7 različitih funkcija, a to su: konstantna, nasumična, linearno padajuća, simuliranog žarenja, strategija prirodne eksponencijalne inercijske težine (e1-PSO), strategija prirodne eksponencijalne inercijske težine (e2-PSO) i logaritamski padajuća.

Za funkcije koje imaju *w_min* i *w_max*, ti parametri su postavljeni na 0,4 i 0,9 slijedno, za konstantnu težinu parametar *c* je postavljen na 0,7, za težinu simuliranog žarenja parametar λ je postavljen na 0,95 i za logaritamski padajuću težinu parametar *a* je postavljen na 1.

Odabrana slika je *walkbridge.tif*, a odabrani broj dimenzija je postavljen na 5. Rezultati za *fitness* vrijednosti su prikazani u tablici 6.2.

Tablica 6.2. Rezultati *fitnessa* kod mjerenja algoritma šišmiša sa inercijskom težinom sa različitim inercijskim težinama

Inercijska težina	***Fitness* vrijednost**
Konstantna	125,7634
Nasumična	125,7503
Linearno padajuća	126,4733
Simuliranog žarenja	127,3716
e1-PSO	135,7815
e2-PSO	133,2138
Logaritamski padajuća	129,9589

Na kraju je odabrana nasumična inercijska težina za daljnja mjerenja zbog najmanje vrijednosti *fitnessa*.

Sljedeće su prikazana provedena mjerenja za sve 3 varijante algoritma šišmiša korištenjem sve 3 slike sa 3 različita broja dimenzija. Korišteni brojevi dimenzija su: 3, 5 i 7. U tablici 6.3. su prikazane *fitness* vrijednosti i trajanja izvođenja kod svake varijante BA. Korištene su skraćenice: BA za algoritam šišmiša, CBA za kaotični algoritam šišmiša i BA w/ IW za algoritam šišmiša sa inercijskom težinom.

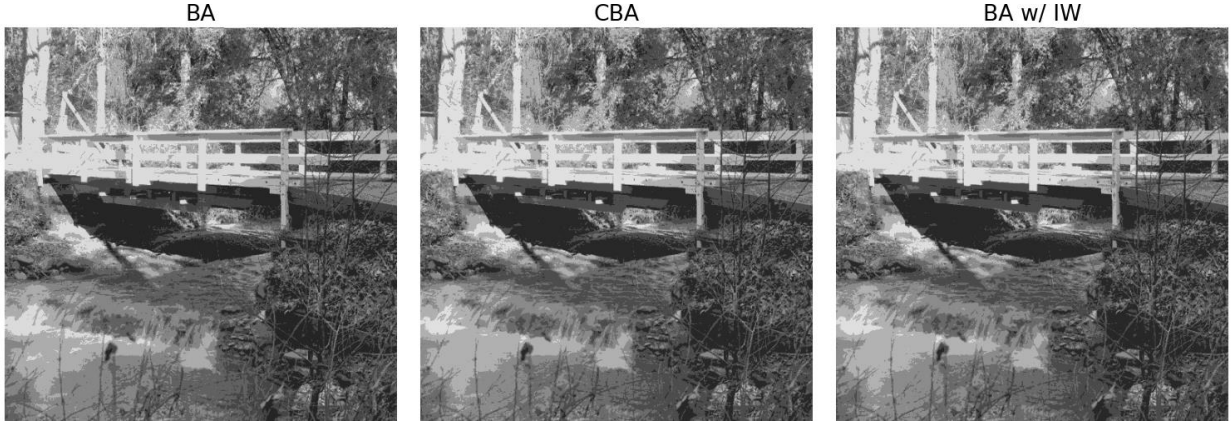
Tablica 6.3. Rezultati *fitnessa* i vremena kod mjerenja različitih varijanti algoritma šišmiša za segmentaciju crno-bijelih slika

Slika	**Broj dimenzija**	**Algoritam**	***Fitness* vrijednost**	**Vrijeme (s)**
walkbridge.tif	7	BA	87,8048	2,9585
		CBA	87,6802	2,9724
		BA w/ IW	79,2389	3,0115
	5	BA	136,3392	2,387
		CBA	135,0029	2,3754
		BA w/ IW	125,7503	2,3944
	3	BA	272,7681	1,7544
		CBA	271,475	1,7466
		BA w/ IW	264,8956	1,7571
lake.tif	7	BA	66,6162	2,6591
		CBA	68,202	2,6531
		BA w/ IW	60,645	2,6891
	5	BA	101,5814	2,0999
		CBA	101,0154	2,1013
		BA w/ IW	92,5648	2,1188

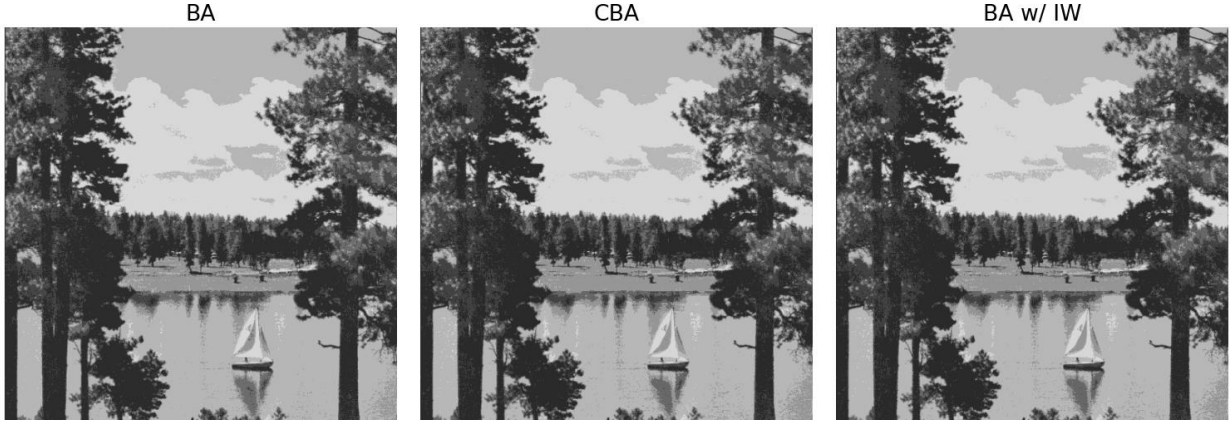
	3	BA	201,1456	1,5107
		CBA	201,243	1,5231
		BA w/ IW	195,7123	1,5078
peppers_gray.tif	7	BA	72,6178	2,5383
		CBA	71,3373	2,5305
		BA w/ IW	63,9784	2,5566
	5	BA	113,6788	1,9981
		CBA	112,9913	2,0022
		BA w/ IW	106,5832	2.0088
	3	BA	209,4903	1,4294
		CBA	207,0053	1,4342
		BA w/ IW	200,0864	1,4241

BA w/ IW ima najbolje rezultate za *fitness* vrijednost, zatim CBA i na posljetku BA.

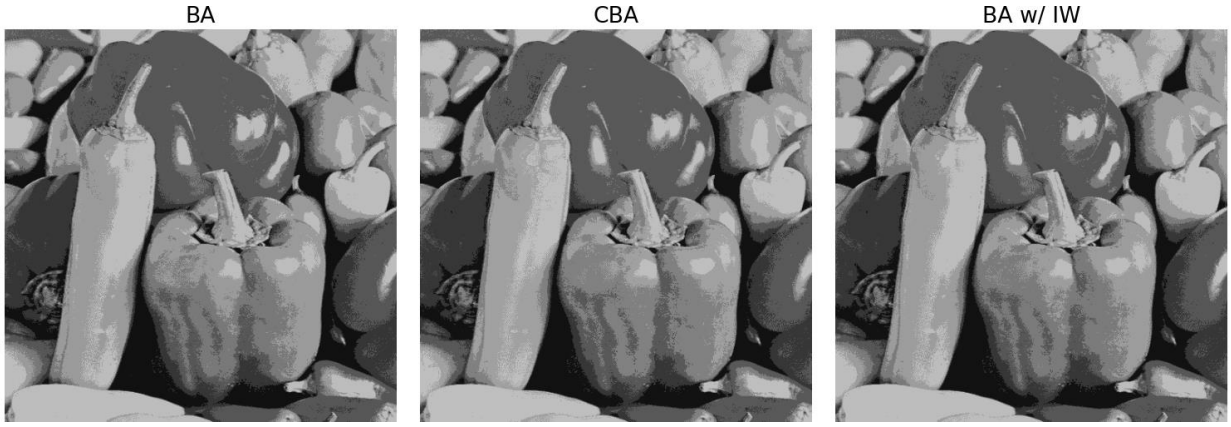
Prikazane su segmentirane slike sa 5 graničnih vrijednosti koje daju najmanju *fitness* vrijednost u 100 iteracija. Slijeva na desno su prikazane za standardni, kaotični i za algoritam šišmiša sa inercijskom težinom na slikama 6.2., 6.3. i 6.4.



Sl. 6.2. Slika *walkbridge.tif* segmentirana pomoću 3 varijante algoritma šišmiša na 6 dijelova



Sl. 6.3. Slika *lake.tif* segmentirana pomoću 3 varijante algoritma šišmiša na 6 dijelova



Sl. 6.4. Slika *peppers_gray.tif* segmentirana pomoću 3 varijante algoritma šišmiša na 6 dijelova

Sljedeće su u tablici 6.4. prikazana mjerenja za segmentirane slike koristeći metrike sličnosti slika i također je prikazana minimalna vrijednost *fitnessa* u 100 iteracija.

Tablica 6.4. Rezultati najmanjeg *fitnessa* i metrika kod mjerenja različitih varijanti algoritma šišmiša za segmentaciju crno-bijelih slika

Slika	**Algoritam**	***Fitness* vrijednost**	**Metrika**	**Rezultat metrike**
walkbridge.tif	BA	119,0939	RMSE	10,929
			PSNR	27,3592 dB
			SSIM	0,8369
	CBA	116,3608	RMSE	10,7967
			PSNR	27,465 dB
			SSIM	0,8398
	BA w/ IW	115,4229	RMSE	10,769
			PSNR	27,4873 dB
			SSIM	0,8394
lake.tif	BA	83,743	RMSE	9,1663
			PSNR	28,8869 dB
			SSIM	0,8288
	CBA	82,607	RMSE	9,1107
			PSNR	28,9398 dB
			SSIM	0,8293
	BA w/ IW	82,5248	RMSE	9,0947
			PSNR	28,955 dB
			SSIM	0,8302
peppers_gray.tif	BA	93,5474	RMSE	9,694
			PSNR	28,4007 dB
			SSIM	0,7643

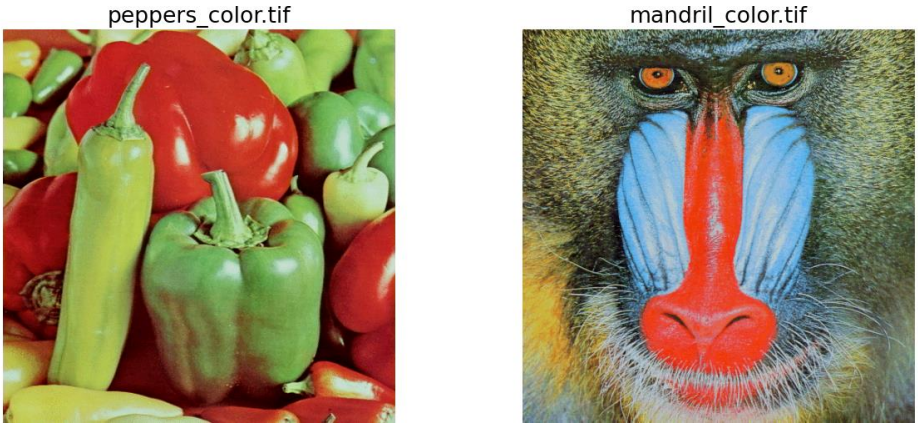
	CBA	94,4664	RMSE	9,7414
			PSNR	28,3583 dB
			SSIM	0,7627
	BA w/ IW	92,0973	RMSE	9,6138
			PSNR	28,4729 dB
			SSIM	0,7639

Po rezultatima metrika se može zaključiti da je algoritam šišmiša sa inercijskom težinom najbolje odradio segmentaciju, zatim kaotični algoritam šišmiša i na kraju standardni algoritam šišmiša. Također su veće šanse da se dogodi da je najmanja vrijednost za *fitness* od jednog algoritma manja od drugog kada su srednje vrijednosti blizu jedna drugoj, a da srednja vrijednost tog jednog nije manja od drugog. Može se vidjeti kako za sliku *peppers_gray.tif* najmanja vrijednost za *fitness* od BA je manja od CBA u tablici 6.4., ali nije i srednja vrijednost manja u tablici 6.3. za broj dimenzija 5.

6.2. Segmentacija obojanih slika

Za segmentaciju obojanih slika provedena su mjerenja kojima se uspoređivao algoritam šišmiša koji koristi *K-means* inicijalizaciju sa običnim *K-means* algoritmom koji se vrtio više puta sa nasumičnim početnim vrijednostima središta grupacija iz skupa podataka. Funkcija cilja je suma kvadrata razlika udaljenosti podataka od središta njihovih grupacija u prostoru (što je manja njena vrijednost to je bolje). Podaci sadrže informacije o BGR (engl. *blue, green, red*) vrijednosti piksela tj. intenzitet plave, zelene i crvene boje pa to onda čini 3D prostor. Mjerenja su provedena za 2 različite slike sa 3 različita broja dimenzija odnosno broja središta grupacija za *K-means*.

Slike koje su korištene za mjerenja u ovom potpoglavlju su *peppers_color.tif* i *mandril_color.tif* te su one prikazane na slici 6.5.



Sl. 6.5. Testne slike za segmentaciju obojanih slika [40]

Vrijednosti parametara za algoritam šišmiša koji koristi *K-means* su sljedeće:

- broj šišmiša: 20
- broj iteracija: 3
- minimalna pozicija: 0
- maksimalna pozicija: 255
- minimalna frekvencija: 0
- maksimalna frekvencija: 1
- σ: 0,1
- γ: 0,1
- α: 0,97
- maksimalni broj iteracija *K-meansa*: 6
- tolerancija: 5e-8

Za *K-means* algoritam koji se vrtio više puta vrijednosti parametara su sljedeće:

- maksimalni broj iteracija *K-meansa*: 12
- broj pokušaja/vrtnji: 15
- tolerancija: 5e-8

Svaki je algoritam u ovom potpoglavlju pokrenut 40 puta (za *K-means* algoritam koji se vrtio više puta misli se na odvojeno pokretanje više vrtnji 40 puta) gdje je svaki put različita sekvenca nasumičnosti te je uzeto srednje vrijeme trajanja i srednji rezultat *fitnessa*.

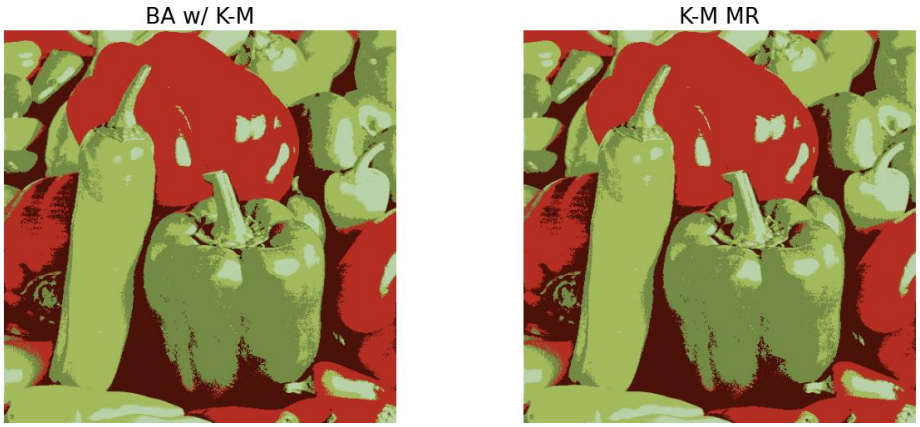
Mjerenja su prikazana u tablici 6.5. Korištene su skraćenice: BA w/ K-M za algoritam šišmiša koji koristi *K-means* i K-M MR za obični *K-means* algoritam koji se vrtio više puta.

Tablica 6.5. Rezultati *fitnessa* i vremena kod mjerenja algoritma šišmiša sa *K-meansom* i običnog *K-means* algoritma kod segmentacije obojanih slika

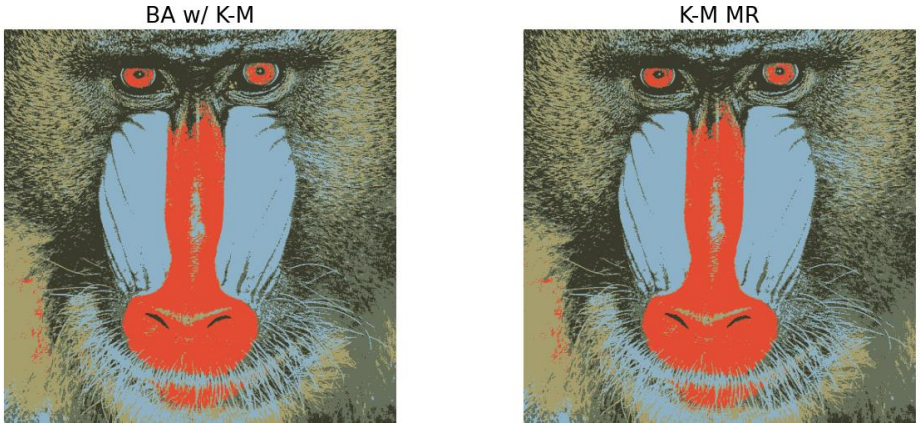
Slika	**Broj dimenzija**	**Algoritam**	***Fitness* vrijednost**	**Vrijeme (s)**
peppers_color.tif	7	BA w/ K-M	203086164,2975	20,1554
		K-M MR	202240018,7699	16,5083
	5	BA w/ K-M	313354105,6832	16,1727
		K-M MR	313211062,9216	12,1541
	3	BA w/ K-M	631881406,1881	11,885
		K-M MR	631882149,2006	7,7921
mandril_color.tif	7	BA w/ K-M	303090412,6027	20,5457
		K-M MR	301089591,6736	19,1578
	5	BA w/ K-M	419777160,4361	16,7031
		K-M MR	419415877,8577	15,2913
	3	BA w/ K-M	776934908,1492	12.7618
		K-M MR	776911373,2227	11,1938

Iz rezultata se može vidjeti da ipak standardni *K-means* algoritam koji se vrtio više puta kraće se izvršava i većinom daje bolji *fitness* rezultat.

Prikazane su segmentirane slike sa 5 grupacija sa središtima koja daju najmanju *fitness* vrijednost u 40 iteracija. Slijeva na desno su prikazane za algoritam šišmiša koji koristi *K-means* inicijalizaciju i za obični *K-means* algoritam koji se vrtio više puta na slikama 6.6. i 6.7.



Sl. 6.6. Slika *peppers_color.tif* segmentirana pomoću 2 algoritma na 5 dijelova



Sl. 6.7. Slika *mandril_color.tif* segmentirana pomoću 2 algoritma na 5 dijelova

Sljedeće su u tablici 6.6. prikazana mjerenja za segmentirane slike koristeći metrike sličnosti slika i također je prikazana minimalna vrijednost *fitnessa* u 40 iteracija.

Tablica 6.6. Rezultati najmanjeg *fitnessa* i metrika kod mjerenja algoritma šišmiša sa *K-meansom* i običnog *K-means* algoritma kod segmentacije obojanih slika

Slika	Metoda	*Fitness* vrijednost	Metrika	Rezultat metrike
peppers_color.tif	BA w/ K-M	313174343,3871	RMSE	19,9586
			PSNR	22,1282 dB
			SSIM	0,5722
	K-M MR	313174343,3871	RMSE	19,9586
			PSNR	22,1282 dB
			SSIM	0,5722
mandril_color.tif	BA w/ K-M	419347074,2458	RMSE	23,0937
			PSNR	20,861 dB
			SSIM	0,6692
	K-M MR	419338667,0388	RMSE	23,0933
			PSNR	20,8611 dB
			SSIM	0,6698

Iz ovoga se može vidjeti kako su rezultati metrika za obje metode isti za *peppers_color.tif* sliku jer imaju istu minimalnu vrijednost za *fitness* no kod *mandril_color.tif* slike obični *K-means* algoritam ima bolju (manju) minimalnu vrijednost za *fitness* i bolje rezultate metrika.

6.3. Semantička segmentacija

U ovom potpoglavlju prikazani su rezultati treniranja CNN-a sa algoritmom šišmiša (CNN-BA).

Prilikom pretprocesiranja slike su sažete na veličinu 256x256, primijenjena je normalizacija sa vrijednostima: srednja vrijednost: [0,485, 0,456, 0,406] i standardna devijacija: [0,229, 0,224, 0,225] te je vanjski rub od objekata spojen sa pozadinom.

Vrijednosti parametara za CNN-BA su sljedeće:

- broj šišmiša: 4
- broj iteracija: 3
- minimalna pozicija: [4, 0,001] (prvi broj predstavlja minimalnu veličinu skupine, a drugi minimalnu brzinu učenja)
- maksimalna pozicija: [32, 0,1]
- minimalna frekvencija: 0
- maksimalna frekvencija: 1
- σ: 0,1
- γ: 0,1

- α: 0,97
- maksimalni broj epoha: 30
- parametar kod momenta od stohastičnog gradijentnog spuštanja: 0,9

Model je istreniran, validiran i testiran koristeći Google Colab na T4 grafičkoj kartici. Model je treniran, validiran i testiran na PASCAL VOC (engl. *Visual Object Classes*) 2007 setu podataka. Nakon što je provedeno treniranje modela i nakon što je odabran najbolji model, na tom modelu se izvršilo testiranje na testnom skupu. Prilikom testiranja izračunat je srednji presjek kroz uniju (engl. *mean intersection over union*, *Mean* IoU).

IoU se računa tako što se podijeli presjek dvaju područja sa njihovom unijom. Vrijednost IoU-a predstavlja udio i što je veća ta vrijednost to je bolje. Izraz za IoU je sljedeći:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{6-7}$$

gdje A predstavlja prvo područje, a B predstavlja drugo područje. Unija pored presjeka uključuje vrijednosti gdje je prvo područje, ali nije i drugo i obrnuto. Kod strojnog učenja ta područja su temeljna istina i predviđene vrijednosti prema [41].

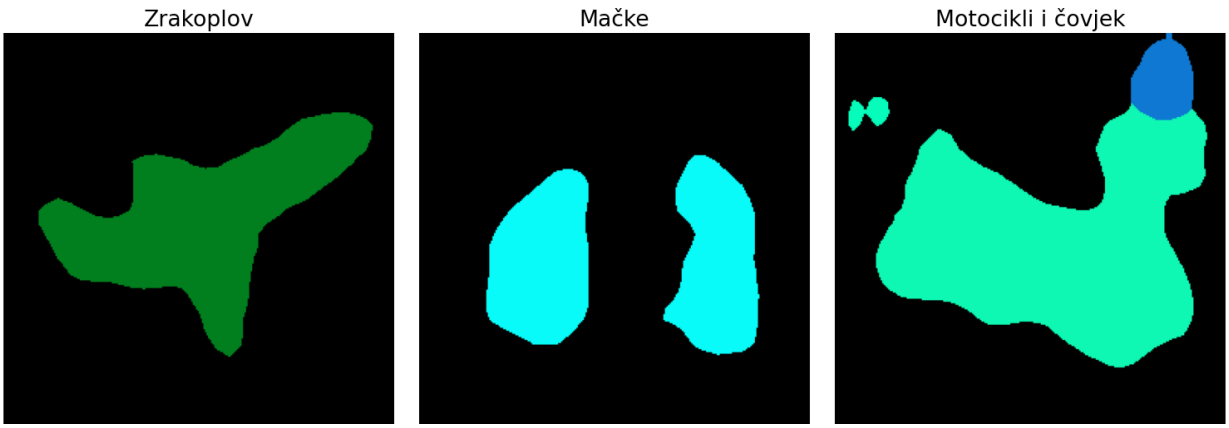
Kada je potrebno izračunati IoU za više klasa računaju se presjeci kroz uniju za sve klase i uzima se njihova srednja vrijednost prema [42].

Izračunata vrijednost za srednji IoU je 0,4453 ili 44,53 %.

Ukupno vrijeme izvršavanja CNN-BA je oko 2 h.

Na slici 6.8. su prikazane slike koje su korištene kao primjer rezultata segmentacije, a na slici 6.9. su one segmentirane.

Sl. 6.8. Testne slike za semantičku segmentaciju [43]



Sl. 6.9. Segmentirane testne slike pomoću CNN-BA

Na slici 6.10. mogu se vidjeti boje kojima su obojane pojedine stavke na slikama.

Sl. 6.10. Boje za stavke na slikama na kojima je izvršena semantička segmentacija

Može se vidjeti kako je model uspješno odradio segmentaciju za te slike.

7. ZAKLJUČAK

Iz ovog rada se može zaključiti kako se metaheuristični algoritmi poput algoritma šišmiša mogu koristiti u svrhe segmentacije slika. Također se može vidjeti kako algoritam šišmiša može koristiti različite strategije poput kaotičnih mapa ili inercijskih težina i prilagoditi ih za moguće poboljšanje svoga rada i davanja boljih rezultata. U radu su objašnjene metode koje algoritam šišmiša koristi i što optimizira kako bi se pomoću njega izvršili različiti tipovi segmentacije slike.

Algoritam šišmiša uspješno radi sa Otsuovom metodom za segmentaciju crno-bijelih slika, a njegova varijanta sa inercijskom težinom prema mjerenjima u ovom radu daje najbolje rezultate. Zatim, varijanta algoritma šišmiša koja koristi *K-means* algoritam uspješno obavlja segmentaciju obojanih slika, iako tradicionalna metoda daje bolje rezultate mjerenja u ovom radu. Na posljetku algoritam šišmiša uspješno radi kao optimizator hiperparametara kod konvolucijske neuronske mreže koja je korištena u svrhu semantičke segmentacije. Za korišteni model vrijednost srednjeg IoU-a od 44,53 % je prilično dobra.

LITERATURA

[1] S. Yang, Q. Chen i L. Peng, »Bat algorithm for multilevel image thresholding based on Otsu and Kapur's entropy«, *Journal of Physics: Conference Series*, br. 1, sv. 1982, p. 012076, srpanj 2021.

[2] J. Senthilnath, S. Kulkarni, J. A. Benediktsson i X. S. Yang, »A Novel Approach for Multispectral Satellite Image Classification Based on the Bat Algorithm«, *IEEE Geoscience and Remote Sensing Letters*, br. 4, sv. 13, pp. 599-603, travanj 2016.

[3] Z. Ye, J. Yang, M. Wang, X. Zong, L. Yan i W. Liu, »2D Tsallis Entropy for Image Segmentation Based on Modified Chaotic Bat Algorithm«, *Entropy*, br. 4, sv. 20, br. čl. 239, ožujak 2018.

[4] X. Yue i H. Zhang, »Modified hybrid bat algorithm with genetic crossover operation and smart inertia weight for multilevel image segmentation«, *Applied Soft Computing*, sv. 90, br. čl. 106157, svibanj 2020.

[5] Q. Lu, Z. Zhang i C. Yue, »Image segmentation based on bat algorithm and pulse coupled neural network«, *3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, pp. 1-4, Xiamen, 2019.

[6] A. Khamis i Y. Wang, »Trajectory-based Algorithms — AI Search Algorithms for Smart Mobility« [Mrežno]
Dostupno na: https://smartmobilityalgorithms.github.io/book/content/TrajectoryAlgorithms/index.html
[Pokušaj pristupa: 10. srpanj 2024.]

[7] »What is metaheuristic? | Autoblocks Glossary« [Mrežno]
Dostupno na: https://www.autoblocks.ai/glossary/metaheuristic
[Pokušaj pristupa: 10. srpanj 2024.]

[8] X. S. Yang, Nature-Inspired Optimization Algorithms: Second Edition, Elsevier, London, 2020.

[9] N. Buhl, »Guide to Image Segmentation in Computer Vision: Best Practices | Encord« [Mrežno]
Dostupno na: https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/
[Pokušaj pristupa: 10. srpanj 2024.]

[10] H. Bandyopadhyay, »Image Segmentation: Deep Learning vs Traditional [Guide]« [Mrežno]
Dostupno na: https://www.v7labs.com/blog/image-segmentation-guide
[Pokušaj pristupa: 10. srpanj 2024.]

[11] »Image segmentation detailed overview [Updated 2024] | SuperAnnotate« [Mrežno]
Dostupno na: https://www.superannotate.com/blog/image-segmentation-for-machine-learning
[Pokušaj pristupa: 10. srpanj 2024.]

[12] M. Tyagi, »Image Segmentation Explained | Built In« [Mrežno]
Dostupno na: https://builtin.com/machine-learning/image-segmentation
[Pokušaj pristupa: 10. srpanj 2024.]

[13] Z. Hussain i D. Agarwal, »A Comparative Analysis of Edge Detection Techniques Used in Flame Image Processing«, *International Journal of Advance Research In Science And Engineering IJARSE*, br. 1, sv. 4, pp. 1335-1343, ožujak 2015.

[14] Samina, »What is Canny edge detection?« [Mrežno]
Dostupno na: https://www.educative.io/answers/what-is-canny-edge-detection
[Pokušaj pristupa: 10. srpanj 2024.]

[15] G. Tanner, »Mean Shift« [Mrežno]
Dostupno na: https://ml-explained.com/blog/mean-shift-explained
[Pokušaj pristupa: 10. srpanj 2024.]

[16] O. Ronneberger, P. Fischer i T. Brox, »U-Net: Convolutional Networks for Biomedical Image Segmentation«, *18th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Minhen, 2015.

[17] K. He, Gkioxari, G., P. Dollár i R. Girshick, »Mask R-CNN«, *IEEE International Conference on Computer Vision (ICCV)*, pp. 2980-2988, Venecija, 2017.

[18] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff i H. Adam, »Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation«, *15th European Conference on Computer Vision (ECCV)*, pp. 833-851, Minhen, 2018.

[19] N. Otsu, »A threshold selection method from gray-level histograms«, *IEEE Transactions on Systems, Man, and Cybernetics*, br. 1, sv. 9, pp. 62-66, siječanj 1975.

[20] R. B. Naik i U. Singh, »A Review on Applications of Chaotic Maps in Pseudo-Random Number Generators and Encryption«, *Annals of Data Science*, br. 1, sv. 11, pp. 25-50, veljača 2024.

[21] L. Bradley, »Logistic Equation - Chaos & Fractals« [Mrežno]
Dostupno na: https://www.stsci.edu/~lbradley/seminar/logdiffeqn.html
[Pokušaj pristupa: 10. srpanj 2024.]

[22] C. Li, K. Qian, S. He, H. Li i W. Feng, »Dynamics and Optimization Control of a Robust Chaotic Map«, *IEEE Access*, sv. 7, pp. 160072-160081, listopad 2019.

[23] M. A. Khan i V. Jeoti, »Modified chaotic tent map with improved robust region«, *IEEE 11th Malaysia International Conference on Communications (MICC)*, pp. 496-499, Kuala Lumpur, 2013.

[24] A. H. Gandomi i X. S. Yang, »Chaotic bat algorithm«, *Journal of Computational Science*, br. 2, sv. 5, pp. 224-232, ožujak 2014.

[25] Z. Cui, F. Li i Q. Kang, »Bat algorithm with inertia weight«, *Chinese Automation Congress (CAC)*, pp. 792-796, Wuhan, 2015.

[26] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon i A. Abraham, »Inertia Weight strategies in Particle Swarm Optimization«, *Third World Congress on Nature and Biologically Inspired Computing*, pp. 633-640, Salamanca, 2011.

[27] M. Sujaritha, M. Kavitha, S. Shunmugapriya, R. S. Vikram, C. Somasundaram i R. Yogeshwaran, »Multispectral Satellite Image Segmentation Using Improved Bat Algorithm«, *International Conference on Advanced Computing Technologies and Applications (ICACTA)*, pp. 1-6, Coimbatore, 2022.

[28] C. Piech i A. Y. Ng, »CS221« [Mrežno]
Dostupno na: https://stanford.edu/~cpiech/cs221/handouts/kmeans.html
[Pokušaj pristupa: 10. srpanj 2024.]

[29] M. Saraswat, »Practical Guide to Clustering Algorithms & Evaluation in R Tutorials & Notes | Machine Learning | HackerEarth« [Mrežno]
Dostupno na: https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/clustering-algorithms-evaluation-r/tutorial/
[Pokušaj pristupa: 10. srpanj 2024.]

[30] D. Senapati, »Grid Search vs Random Search. In this article, we will focus on two… | by Deepak Senapati | Medium« [Mrežno]
Dostupno na: https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318
[Pokušaj pristupa: 10. srpanj 2024.]

[31] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko i M. Tuba, »Designing Convolutional Neural Network Architecture by the Firefly Algorithm«, *International Young Engineers Forum (YEF-ECE)*, pp. 59-65, Costa da Caparica, 2019.

[32] E. Shelhamer, J. Long i T. Darrell, »Fully Convolutional Networks for Semantic Segmentation«, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, br. 4, sv. 39, pp. 640-651, travanj 2017.

[33] M. Shroff, »Know your Neural Network architecture more by understanding these terms | by Megha Shroff | Medium« [Mrežno]
Dostupno na: https://medium.com/@shroffmegha6695/know-your-neural-network-architecture-more-by-understanding-these-terms-67faf4ea0efb
[Pokušaj pristupa: 10. srpanj 2024.]

[34] P. Potrimba, »What is ResNet-50?« [Mrežno]
Dostupno na: https://blog.roboflow.com/what-is-resnet-50/
[Pokušaj pristupa: 10. srpanj 2024.]

[35] A. Srinivasan, »Stochastic Gradient Descent — Clearly Explained !! | by Aishwarya V Srinivasan | Towards Data Science« [Mrežno]
Dostupno na: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31
[Pokušaj pristupa: 10. srpanj 2024.]

[36] V. Bushaev, »Stochastic Gradient Descent with momentum | by Vitaly Bushaev | Towards Data Science,« [Mrežno]
Dostupno na: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d
[Pokušaj pristupa: 10. srpanj 2024.]

[37] S. Oladele, »Machine Learning Cross-Entropy Loss Functions« [Mrežno]
Dostupno na: https://encord.com/blog/an-introduction-to-cross-entropy-loss-functions/
[Pokušaj pristupa: 10. srpanj 2024.]

[38] N. Ekhtiari, »Comparing ground truth with predictions using image similarity measures · UP42« [Mrežno]
Dostupno na: https://up42.com/blog/image-similarity-measures
[Pokušaj pristupa: 10. srpanj 2024.]

[39] P. Datta, »All about Structural Similarity Index (SSIM): Theory + Code in PyTorch | by Pranjal Datta | SRM MIC | Medium« [Mrežno]
Dostupno na: https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e
[Pokušaj pristupa: 10. srpanj 2024.]

[40] »Image Databases« [Mrežno]
Dostupno na:
https://www.imageprocessingplace.com/root_files_V3/image_databases.htm
[Pokušaj pristupa: 10. srpanj 2024.]

[41] D. Shah, »Intersection over Union (IoU): Definition, Calculation, Code« [Mrežno]
Dostupno na: https://www.v7labs.com/blog/intersection-over-union-guide
[Pokušaj pristupa: 10. srpanj 2024.]

[42] »Mean Intersection over Union (IoU) - OECD.AI« [Mrežno]
Dostupno na: https://oecd.ai/en/catalogue/metrics/mean-intersection-over-union-%28iou%29
[Pokušaj pristupa: 10. srpanj 2024.]

[43] M. Everingham, V. L., C. K. I. and Williams, J. Winn i A. Zisserman, »The {PASCAL} {V}isual {O}bject {C}lasses {C}hallenge 2007 {(VOC2007)} {R}esults« [Mrežno]
Dostupno na: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html
[Pokušaj pristupa: 10. srpanj 2024.]

SAŽETAK

U ovom radu prvo je objašnjen koncept algoritma šišmiša kao i način njegova rada. Zatim su objašnjene vrste i metode segmentacije slike. Potom slijedi opis različitih varijanti algoritma šišmiša koje su korištene u svrhu segmentacije slika kao i različite metode koje su korištene sa tim varijantama te njihov međusobni rad zajedno. Kod varijanti algoritma šišmiša pojavljuju se pojmovi poput kaosa, inercijskih težina, dok je kod metoda objašnjena Otsuova metoda, *K-means* algoritam te rad sa neuronskim mrežama. Kod rada sa neuronskim mrežama pojašnjeni su pojedini pojmovi iz dubokog učenja poput: stohastičkog gradijentnog spuštanja sa momentom, gubitak unakrsne entropije, srednji IoU itd. Poslije su izvršena različita mjerenja za pojedine varijante algoritma šišmiša ovisno o tipu segmentacije. Na kraju su prikazani i raspravljeni rezultati.

Ključne riječi: Algoritam šišmiša, Metaheuristika, Računalni vid, Segmentacija slike

ABSTRACT

This paper firstly explains the concept of the bat algorithm and its working mechanism. After that types and methods of image segmentation are explained. Following this, different variants of bat algorithm are described, along with different methods employed with these variants and their functionality altogether. These variants include terms such as chaos and inertia weights, while the methods involve explanation of Otsu's method, K-means algorithm and working with neural networks. When working with neural networks some of the terms from deep learning are clarified including: stochastic gradient descent with momentum, cross-entropy loss, mean IoU and so on. Later, different types of measurements are conducted for specific variants depending on the type of segmentation. In the end results are presented and discussed.

Keywords: Bat algorithm, Computer vision, Image segmentation, Metaheuristics