SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 800

DETEKCIJA RUBOVA NA SLIKAMA KORIŠTENJEM TRADICIONALNIH METODA I METODA DUBOKOG UČENJA

Sara Tedeško

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 800

DETEKCIJA RUBOVA NA SLIKAMA KORIŠTENJEM TRADICIONALNIH METODA I METODA DUBOKOG UČENJA

Sara Tedeško

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

DIPLOMSKI ZADATAK br. 800

Pristupnica:	**Sara Tedeško (0036523716)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Detekcija rubova na slikama korištenjem tradicionalnih metoda i metoda dubokog učenja**

Opis zadatka:

Detekcija rubova na slikama jedan je od klasičnih, ali i dalje aktualnih problema u računalnom vidu. Tradicionalne metode poput Cannyja, Sobela, Prewitta i Laplaciana često se koriste za ovu svrhu. Međutim, suvremene metode temeljene na strojnom (npr. SVM, slučajne šume) i dubokom učenju (npr. HED, PiDiNet, DexiNed) mogu poboljšati rezultate. Cilj ovog diplomskog rada je istražiti i implementirati različite metode detekcije rubova, uključujući uobičajene i suvremene algoritme, te analizirati njihove performance pomoću različitih metričkih kriterija. Naglasak će se staviti na izbor prikladnih mjera kvalitete detektiranih rubova s obzirom na specifične primjene (npr. stvaranje bojanki). Također, potrebno je istražiti tehnike naknadne obrade detekcije rubova (engl. postprocessing) kako bi se poboljšala kvaliteta detekcije. Rad uključuje izradu web aplikacije s razmatranim algoritmima i metrikama, uzimajući u obzir prikupljeni skup testnih slika iz slobodno dostupnih izvora. Aplikacija treba omogućiti ukrcavanje slike i detekciju rubova korištenjem implementiranih algoritama. Eksperimentalni rezultati trebaju biti analizirani i uspoređeni sa srodnom literaturom kako bi se identificirale najučinkovitije metode u različitim scenarijima.

Rok za predaju rada: 4. srpnja 2025.

Zahvaljujem mentoru izv. prof. dr. sc. Alanu Joviću na stručnom vodstvu te prijateljima Dariu Pavloviću, Heidi Sokolovski i Matei Mustedanagić na emotivnoj potpori za vrijeme školovanja i pri izradi ovoga rada.

Sadržaj

1. Uvod

Unatoč dugoj povijesti istraživanja, detekcija rubova i dalje je izazovno područje zbog brojnih faktora koji utječu na kvalitetu rubne mape. Izazovi uključuju osjetljivost na šum i varijacije osvjetljenja, složenost scena, razlike u teksturi, debljinu rubova, povezanost rubova te postizanje ravnoteže između detekcije što više relevantnih rubova i minimiziranja "lažnih" rubova. Dodatno, specifične primjene zahtijevaju različite karakteristike rubova. Detekcija rubova nalazi široku primjenu u raznim područjima, uključujući medicinsku dijagnostiku (npr. segmentacija organa na CT ili MRI snimkama), autonomna vozila (prepoznavanje prometnih traka i prepreka), industrijsku inspekciju (otkrivanje defekata na proizvodima) te u kreativnim industrijama gdje omogućuje razvijanje kreativnih rješenja poput generiranja linijskih crteža i bojanki iz stvarnih slika.

U kontekstu ovog rada, detekcija rubova razmatra se s posebnim naglaskom na njezinu primjenu u stvaranju bojanki. U takvoj primjeni, kvaliteta detekcije mora zadovoljiti određene estetske i funkcionalne kriterije - linije moraju biti čiste, kontinuirane i jasno definirane kako bi korisnicima omogućile ugodno iskustvo bojanja. Stoga je važno ne samo detektirati rubove, već i provesti odgovarajuću obradu kako bi rezultati bili pogodni za ovu specifičnu namjenu.

Detekcija rubova nije samo tehnički izazov, već i ključni proces koji omogućuje mnoge primjene. S obzirom na razvoj sve naprednijih metoda – od tradicionalnih operatora do modela temeljenih na strojnom učenju – istraživanje i usporedba različitih pristupa postaje sve relevantnija, osobito kada se detekcija koristi u svrhu daljnje vizualne obrade i interpretacije.

Cilj je istražiti i implementirati različite metode detekcije rubova, obuhvaćajući tradicionalne metode, metode strojnog i dubokog učenja kao i segmentacijske metode u ovu

svrhu. Potrebno je spomenuti i tehnike naknadne obrade (engl. *postprocessing*) kojima se može dodatno unaprijediti konačna kvaliteta rubne slike. Koristeći metričke kriterije, osobito one relevantne za specifične primjene poput stvaranja bojanki, potrebno je usporediti rezultate navedenih metoda. U sklopu rada razvijeno je web sučelje koje omogućuje eksperimentalnu evaluaciju na prikupljenom skupu testnih slika te prikaz rezultata detekcija raznih metoda.

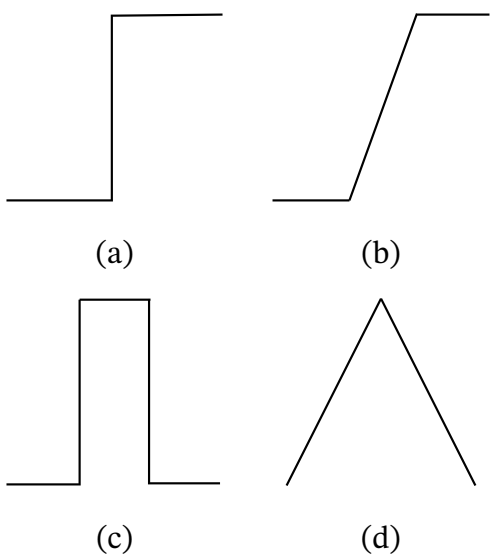
2. Teorijska podloga

2.1. Detekcija rubova

Detekcija rubova predstavlja jedan od temeljnih koraka u obradi i analizi digitalnih slika, posebno u području računalnog vida. Rubovi u slici definiraju granice između različitih objekata ili regija koje se razlikuju po intenzitetu, boji, teksturi ili nekoj drugoj vizualnoj karakteristici. Prepoznavanje tih rubova omogućuje računalnim sustavima da razumiju strukturu prikazane scene, identificiraju objekte i izdvoje važne informacije iz složenog vizualnog sadržaja.

U matematičkom smislu, rubovi odgovaraju naglim promjenama u svjetlini piksela, odnosno velikim gradijentima intenziteta. Rub na slici značajna je lokalna promjena u intenzitetu slike, obično povezana s diskontinuitetom u samom intenzitetu slike. Diskontinuiteti u intenzitetu slike mogu biti rub stepenice (engl. *step edge*), gdje intenzitet slike naglo prelazi s jedne vrijednosti na jednoj strani diskontinuiteta na drugu vrijednost na suprotnoj strani, ili linijski rub (engl. *line edg*e), gdje intenzitet slike naglo mijenja vrijednost, a zatim se vraća na početnu vrijednost unutar male udaljenosti [1]. Zbog niskofrekventnih komponenti ili zaglađivanja koje uvodi većina senzorskih uređaja, oštri diskontinuiteti, poput rubova stepenice ili linijskog ruba, rijetko stvarno postoje u slikama. Zbog toga, rubovi stepenice prebacuju se u nagibne rubove (engl. *ramp edges*), a linijski rubovi postaju krovni rubovi (engl. *roof edges*), gdje promjene intenziteta nisu trenutačne, nego se događaju unutar određenog raspona. Ovi rubovi vizualizirani su na slici 2.1.

Analiza gradijenata omogućuje detekciju mjesta na slici gdje dolazi do prijelaza između različitih regija, što se u računalnom vidu često koristi kao prvi korak u složenijim zadacima poput segmentacije, prepoznavanja objekata, praćenja objekata i mapiranja.



Slika 2.1. Tipovi rubova (a) rub stepenice (b) nagibni rub (c) linijski rub (d) krovni rub

Jedan od razloga zašto je detekcija rubova toliko važna jest njezina sposobnost smanjenja količine podataka koja se treba obraditi. Umjesto analize cijele slike, fokus se stavlja na značajne prijelaze koji često sadrže ključne informacije o strukturi prikazanih objekata. Time se povećava učinkovitost algoritama i omogućuje preciznija interpretacija vizualnih podataka.

2.2. Tradicionalne metode detekcije rubova

Tradicionalne metode detekcije rubova temelje se na analizi promjena intenziteta piksela unutar slike [2]. Većina ovih metoda koristi konvolucijske maske (engl. *kernels*) koje otkrivaju oštre prijelaze u svjetlini slike, pri čemu se ti prijelazi interpretiraju kao rubovi. Relativno su jednostavne i brze za implementaciju, no tradicionalne metode često su osjetljive na šum i ne detektiraju rubove jednako dobro u kompleksnim slikama [1]. Iako ograničene u složenijim vizualnim scenama, i dalje predstavljaju važan temelj za razumijevanje i razvoj naprednijih algoritama. Njihova jednostavnost i efikasnost čine ih pogodnima za osnovne zadatke kao i za usporedbe s metodama temeljenima na strojnome i dubokome učenju.

Tradicionalni operatori temeljeni na gradijentu, poput Sobela, Prewitta i Robertsa, u ranim istraživanjima korišteni su za detekciju rubova, ali nisu davali oštre rubove te su bili vrlo osjetljivi na šum [1].

Sobelov operator koristi dvije konvolucijske maske za detekciju horizontalnih i vertikalnih gradijenata intenziteta piksela [3, 4]. Maska je oblikovana tako da kombinira izračun gradijenta i uklanjanje šuma pomoću filtriranja. Gradijenti se računaju pomoću sljedećih matrica:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Veličina gradijenta:

$$G = \sqrt{G_x^2 + G_y^2}$$

Sobelov operator kombinira detekciju rubova s blagim filtriranjem, što ga čini otpornijim na šum od jednostavnijih operatora poput Robertsa [2].

Prewittov operator koristi slične maske kao Sobel, ali bez težinskog naglašavanja središnjeg reda ili stupca [5, 6]:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

Veličina gradijenta računa se kao:

$$G = \sqrt{G_x^2 + G_y^2}$$

Prewittova metoda daje slične rezultate kao Sobel, ali je nešto manje učinkovita pri suzbijanju šuma.

Robertsov operator temelji se na izračunu gradijenata intenziteta pomoću dijagonalnih razlika između susjednih piksela [7, 8]. Za razliku od ostalih operatora koji koriste

3×3 matrice, Robertsov koristi jednostavne konvolucijske maske veličine 2x2:

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Veličina gradijenta izračunava se prema sljedećoj formuli:

$$G = \sqrt{G_x^2 + G_y^2}$$

Robertsov operator ima prednost u vrlo brzoj implementaciji, ali je izrazito osjetljiv na šum i manje pogodan za slike niske kvalitete. Zbog korištenja samo četiri piksela, često rezultira grubim rubovima, ali se i dalje koristi u edukativne svrhe i u slučajevima kada je važna brzina izvođenja.

Laplaceov operator također ima probleme visoke vjerojatnosti detekcije lažnih rubova i značajne pogreške lokalizacije na zakrivljenim rubovima [9, 10]. Ovaj operator koristi drugi derivat slike kako bi detektirao rubove, odnosno mjesta gdje dolazi do promjene nagiba intenziteta. Budući da je vrlo osjetljiv na šum, obično se koristi u kombinaciji s Gaussovim filtriranjem — što se naziva *Laplacian of Gaussian (LoG)* [9]. Matematička formulacija operatora je:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Laplaceova metoda filtrira sliku kako bi uklonila šum, a zatim pronalazi nulte prijelaze u Laplaceovoj transformaciji slike. Omogućuje detekciju finih detalja, ali može rezultirati duplim linijama.

Međutim, algoritam koji je 1986. godine predložio John F. Canny smatra se jednom od najpreciznijih i najstabilnijih metoda među tradicionalnim pristupima za detekciju rubova na slikama koje su zasićene šumom [11]. Cannyjev cilj bio je pronaći optimalni algoritam za detekciju rubova koji smanjuje vjerojatnost detekcije lažnih rubova i daje oštre rubove [1, 11]. **Cannyev algoritam** sastoji se od više koraka [11, 12]:

- Gaussovo filtriranje slike radi uklanjanja šuma

- Izračunavanje gradijenata (magnituda i orijentacija)
- Potiskivanje ne-maksimuma (engl. *non-maximum suppression*)
- Dvoslojno pragiranje (gornji i donji prag)
- Histerezna detekcija rubova (spajanje rubova ovisno o povezanosti)

Zahvaljujući ovoj višefaznoj obradi, Cannyjev algoritam nudi izvrsnu detekciju i lokalizaciju rubova uz relativnu otpornost na šum. Ipak, odabir parametara, poput standardne devijacije Gaussovog filtra i pragova, ključno je za kvalitetu rezultata.

2.3. Metode dubokog učenja za detekciju rubova

Tradicionalne metode detekcije rubova, unatoč svojoj matematičkoj elegantnosti, često se suočavaju s ograničenjima pri radu s kompleksnim prirodnim slikama. Sve veća složenost vizualnih podataka i potreba za robusnijim performansama potaknula je razvoj pristupa temeljenog na dubokom učenju (engl. *deep learning*) koji kombinira sposobnost konvolucijskih neuronskih mreža (CNN) s domenskim znanjem o detekciji rubova.

Pristup detekciji rubova putem CNN-a

Konvolucijske neuronske mreže predstavljaju promjenu paradigme u području detekcije rubova kroz sposobnost učenja hijerarhijskih reprezentacija iz podataka. Umjesto oslanjanja na ručno definirane kernele poput operatora Sobela ili Cannya, arhitekture CNN-ova mogu automatski naučiti optimalne filtere kroz proces propagacije unatrag (engl. *backpropagation*).

Prednosti ovog pristupa uključuju [13]:

- Hijerarhijsko učenje značajki: dublji slojevi kombiniraju jednostavne rubove u složenije strukture
- Optimiranje s kraja na kraj mreže (engl. *end-to-end*): cijela mreža se optimira za specifičan zadatak detekcije rubova
- Adaptivnost na podatke: mreža može naučiti rubove specifične za određenu domenu ili vrstu slika

- Robusnost na šum: dublja arhitektura razlikuje prave rubove od lažnih

2.3.1. Holistically-Nested Edge Detection (HED)

Pristup **Holistically-Nested Edge Detection (HED)** koristi konvolucijske mreže s dubokim nadzorom za predviđanje rubova po principu “slika-u-sliku” (engl. *image-to-image*) [14]. Pojam "slika-u-sliku" odnosi se na tip predviđanja gdje je i ulaz i izlaz mreže u formatu slike iste veličine.

Arhitektura mreže HED:

- Bočni izlazi (engl. *side outputs*): specijalizirani izlazi koji se dodaju na različite konvolucijske slojeve mreže (conv1_2, conv2_2, conv3_3, conv4_3, conv5_3) kako bi se omogućilo predviđanje rubova na različitim razinama apstrakcije
- Duboki nadzor: svaki bočni izlaz se neovisno uči za predviđanje rubne mape koristeći vlastitu funkciju gubitka
- Fuzijski sloj: kombinira predviđanja svih bočnih izlaza

Ukupna funkcija gubitka:

$$L_{\text{total}} = \sum_{m=1}^{M} \alpha_m L_{\text{side}}^{(m)} + L_{\text{fuse}}$$

gdje je $L_{\text{side}}^{(m)}$ gubitak za m-ti bočni izlaz, α_m težinski koeficijent, a L_{fuse} gubitak fuzijskog sloja.

HED je na skupu podataka za detekciju rubova BSDS500 [15] postigao F-mjeru = 0,790 s vremenom izvršavanja $\approx 0,4s$ po slici [14]. Međutim, arhitektura HED sadrži približno 15 milijuna parametara, što je značajno više od kompaktnijih pristupa poput modela PiDiNet.

2.3.2. Pixel Difference Networks (PiDiNet)

Metoda **Pixel Difference Networks (PiDiNet)**, koju su predstavili Su i suradnici, uvodi Pixel Difference Convolution (PDC) blokove koji integriraju tradicionalne gradijentne

operatore u arhitekturu CNN-ova [16][16]. PDC operatori inspirirani su Sobelovim, Prewittovim i Robertsovim kernelima, omogućavajući mreži da koristi prednosti tradicionalnih pristupa unutar duboke arhitekture.

Ova kompaktna arhitektura sadrži manje od 1 milijun parametara u osnovnoj verziji, što je značajno manje od konkurentskih modela. Unatoč svojoj kompaktnosti, PiDiNet postiže visoke performanse s F-mjerom = 0,807 rezultatom na BSDS500 skupu podataka [16].

Autori su razvili tri varijante arhitekture PiDiNet optimirane za različite scenarije primjene. Verzija PiDiNet-Tiny sadrži manje od 0,1 milijuna parametara i postiže brzinu od približno 200 slika u sekundi, što je čini pogodnom za aplikacije u realnom vremenu. Verzija PiDiNet-Small predstavlja balans između brzine izvršavanja i točnosti detekcije. Standardna verzija PiDiNeta nudi najbolju točnost među varijantama, zadržavajući pritom efikasnost u usporedbi s drugim sličnim metodama.

2.3.3. Dense Extreme Inception Network (DexiNed)

Arhitektura **Dense Extreme Inception Network (DexiNed**), koju su razvili Soria i suradnici, predstavlja pristup detekciji rubova koji se razlikuje od većine suvremenih metoda po sposobnosti učenja od početka, bez potrebe za prethodno naučenim modelima iz drugih domena poput detekcije objekata [17][17] što znači da se mreža DexiNed uči s potpuno nasumično inicijaliziranim težinama, a ne koristi već naučene parametre iz drugih zadataka.

Sastoji se od dva glavna dijela koji rade zajedno da bi postigli točnu detekciju rubova. Koder Dexi dio je koji izdvaja značajke iz slika i sastoji se od šest blokova inspiriranih mrežom Xception, što omogućava dobro prepoznavanje obrazaca na različitim razinama detalja [13]. Drugi, dekoder USNet, odgovoran je za generiranje mape rubova (engl. *edge map*) u različitim veličinama, što omogućava mreži da detektira rubove različitih veličina i složenosti unutar iste slike.

Sposobnost učenja od početka (engl. *training from scratch*) glavna je prednost metode DexiNed u odnosu na druge metode koje koriste prethodno naučene modele iz skupa podataka [17]. Ova nezavisnost od prethodno naučenih težina omogućuje mreži da nauči

značajke samo za detekciju rubova, bez utjecaja prethodnog znanja iz drugih zadataka.

Za razliku od drugih metoda koje često proizvode debele ili nejasno definirane rubove, DexiNed generira tanke, jednopikselske rubove koji su oštriji i bolje lokalizirani [17]. Ova karakteristika čini ga posebno pogodnim za aplikacije koje zahtijevaju visoku preciznost, poput stvaranja bojanki ili tehničkih crteža.

DexiNed je na skupu podataka BSDS500 postigao F-mjeru = 0,799, što je poboljšanje u odnosu na HED (0,798) [13]. Ovi rezultati potvrđuju učinkovitost ove arhitekture za praktičnu upotrebu u detekciji rubova.

2.4. Ostale metode

2.4.1. Strukturirane šume

Algoritam **strukturirane šume** (engl. *Structured Forests*), koji su razvili Dollár i Zitnick, predstavlja hibridni pristup koji kombinira strojno učenje (engl. *machine learning*) s eksplicitnim modeliranjem lokalne strukture rubova [18]. Za razliku od tradicionalnih metoda koje analiziraju pojedinačne piksele, metoda strukturiranih šuma radi s manjim dijelovima slike kako bi bolje uhvatila lokalne obrasce rubova.

Ovaj algoritam koristi strukturirane labele umjesto označavanja pojedinačnih piksela [18]. To znači da umjesto da za svaki piksel posebno odlučuje je li dio ruba, algoritam gleda cijele lokalne maske rubova odjednom. Ovaj pristup omogućava bolju detekciju povezanih struktura poput ravnih linija, T-spojeva ili Y-spojeva koje su česte u prirodnim slikama [18].

Temelj algoritma čine slučajne šume odlučivanja (engl. *random decision forests*) koje su učene na skupu podataka BSDS500 [18]. Tijekom učenja, model uči prepoznavati obrasce u dijelovima slike veličine 32×32 piksela i za svaki takav dio predviđa masku rubova veličine 16×16 . Ova struktura omogućava algoritmu da nauči složenije oblike rubova nego što bi bilo moguće analiziranjem pojedinačnih piksela.

Jedna od glavnih prednosti metode strukturiranih šuma je brzina izvršavanja u realnom vremenu, što ih čini pogodnima za video aplikacije [18] i rješenja gdje su računski resursi ograničeni. Iako je algoritam brz, on i dalje postiže visoku točnost detekcije. Na

BSDS500 skupu podataka postiže F-mjeru = 0,74, što predstavlja dobar balans između brzine i preciznosti [18].

2.4.2. Segmentacijske tehnike

Segmentacijske tehnike mogu se koristiti kao komplementarne metode za poboljšanje kvalitete detekcije rubova kroz izdvajanje regija interesa ili poboljšanje lokalizacije granica objekata. U ovom radu primijenjene su dvije klasične segmentacijske metode: algoritmi Watershed i GrabCut.

Algoritam Watershed

Algoritam Watershed temelji se na konceptu topoloških *watershed* linija iz geografije [19]. Algoritam tretira sivu sliku kao 3D topografiju gdje intenzitet svakog piksela predstavlja visinu u toj točki [20][21]. Proces segmentacije funkcionira kao postupno "poplavljavanje" dolina od označenih markera, pri čemu se "voda" iz različitih dolina ne smije pomiješati [19].

Matematički, algoritam radi na gradijentnoj slici gdje lokalni minimumi predstavljaju početne točke za "poplavljavanje" [22]. Proces se može opisati kroz tri glavna koraka. Prvo se definiraju početne točke (engl. *seeds*) za svaku regiju kroz postavljanje markera. Zatim slijedi postupno širenje regija od markera prema višim gradijentnim vrijednostima kroz poplavljavanje dolina. Konačno, granice između regija postaju segmentacijske linije.

Algoritam može uspješno identificirati objekte koji su međusobno povezani ili se dodiruju, što je teško postići tradicionalnim metodama koje koriste pragove (engl. *threshold*). Međutim, algoritam ima značajna ograničenja koja uključuju ekstremno pretjerano segmentiranje (engl. *over-segmentation*) kada se koristi bez pravilnih markera, osjetljivost na šum u originalnoj slici, i gubitak konture ciljnih regija u slikama niskog kontrasta [23][24].

Algoritam GrabCut

Algoritam GrabCut predstavlja poluautomatsku iterativnu metodu segmentacije koja kombinira teoriju grafova s Gaussovim miješanim modelima (engl. *Gaussian Mixture*

Models) (GMM) [25][26].

Ova metoda predstavlja sliku kao graf gdje čvorovi predstavljaju piksele, a algoritam traži optimalni presjek koji dijeli čvorove na objekte interesa odnosno prednji plan (engl. *foreground*) i pozadine (engl. *background*). Gaussian Mixture Models se koriste za modeliranje intenziteta piksela, gdje se svaki piksel dodjeljuje komponenti miješanja na temelju izračunatih vjerojatnosti.

Prednosti GrabCuta su visoka točnost segmentacije za dobro definirane objekte i mogućnost iterativnog poboljšanja rezultata kroz korisničku interakciju. Ograničenja uključuju ovisnost o kvaliteti inicijalizacije i sklonost konvergenciji u lokalne optimume, posebno kada objekti i pozadinke regije imaju slične karakteristike [26][27].

3. Metodologija istraživanja

3.1. Odabir i priprema skupa podataka

Uz skup podataka Berkeley Segmentation Dataset 500 (ranije spominjan kao BSDS500), često korišten skup u svrhe detekcije rubova je i NYU Depth v2 [28]. BSDS500 nudi višestruke, ručno izrađene granice za prirodne prizore, što omogućuje robusnu procjenu detekcije rubova, dok NYUDv2 koristi dubinske slike unutarnjih prostora koje često sadrže šum te nema takvu razinu višestruke ručne validacije. Uz to, BSDS500 je dizajniran upravo za zadatke detekcije rubova i segmentaciju u prirodnim scenama. Sadrži slike visoke varijabilnosti sadržaja (priroda, urbanistički prizori, teksture) s namjerom za testiranje raznih modela, dok je NYUDv2 fokusiran na semantičku segmentaciju unutarnjih prostora i sadrži mnogo ponavljajućih elemenata poput zidova i namještaja. Zbog standardne rezolucije BSDS500 omogućuje lakšu usporedivost tradicionalnih i dubokih metoda.

Skup BSDS500, preuzet s GitHub repozitorija [29], sastoji se od 500 slika, od kojih je 200 namijenjeno za učenje, 100 za validaciju i 200 za testiranje. Svaka od slika ima pet zasebnih ručno označenih segmenata, pri čemu se te oznake smatraju referentnima (engl. *ground truth*) u obliku datoteke MATLAB. One služe kao referentna oznaka za detekciju rubova i segmentaciju i kasnije se koriste u računanju metrika za svaku od metoda.

Kako priotitet ovog rada nije na vlastitom učenju modela, korišteni su prethodno naučeni modeli. Konačno, korišteni modeli naučeni su upravo na ovom skupu podataka, stoga je on korišten za eksperimentalno testiranje u ovom radu.

Za potrebe ovog rada koristio se isključivo testni skup slika, čime su rezultati usporedivi s referentnim rezultatima postojećih metoda nad skupom BSDS500 i time osigurano

pošteno uspoređivanje.

Izbor uzorka slika

Kako bi se obuhvatile raznolike razine vizualne složenosti i teksturalne varijacije, ali i što točniji rezultati mjerenja, u svrhu usporedbe odabran je cijeli skup slika za testiranje skupa BSDS500, odnosno 200 slika. Broj od 200 slika omogućuje statističku reprezentativnost različitih scena (jednostavni oblici, složene strukture, kontrastni i niskokontrastni dijelovi) bez prevelikog opterećenja na detaljnu vizualnu analizu.

Odabrane slike zadovoljavaju sljedeće kriterije:

- različit stupanj teksturne složenosti (npr. glatke površine u odnosu na bogate teksture)
- raznolikost kontrasta rubova (oštri prijelazi u odnosu na gradijente)

3.2. Implementacija metoda

Svaka od implementiranih metoda prima parametar `invert` koja ako je postavljena na `True` omogućava inverziju boja za bolje vizualno predstavljanje gdje crne linije označavaju detektirane rubove na bijeloj pozadini. U ovu svrhu razvijena je pomoćna funkcija:

```
def invert_colors(image):
    inverted_image = 255 - image
    return inverted_image
```

3.2.1. Implementacija tradicionalnih metoda

U ovom poglavlju prikazane su implementacije nekoliko tradicionalnih algoritama detekcije rubova koji služe kao referentne metode za usporedbu s modernim pristupima temeljenim na dubokom učenju. Svaka metoda ima specifične karakteristike u načinu detektiranja rubova i različite prednosti ovisno o vrsti slike. Za implementaciju tradicionalnih metoda korištena je biblioteka OpenCV.

Cannyjev algoritam

Cannyjev algoritam temelji se na nekoliko ključnih koraka: Gaussovo zamućivanje za uklanjanje šuma, izračun gradijenta intenziteta, potiskivanje ne-maksimuma i dvostruki prag za finalno odlučivanje o rubovima.

Implementacija koristi fiksne pragove:

```
result = cv2.Canny(img, 100, 200)
```

Vrijednosti 100 i 200 predstavljaju niži i viši prag za histerezno povezivanje rubova. Pikseli s gradijentom većim od 200 sigurno su rubovi, oni između 100 i 200 postaju rubovi samo ako su povezani s jačim rubovima, a oni ispod 100 se odbacuju [30].

Sobelov operator

Sobelov operator izračunava gradijent u horizontalnom i vertikalnom smjeru, omogućavajući detekciju rubova različitih orijentacija.

```
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)  # vertikalni rubovi
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)  # horizontalni rubovi
result = np.sqrt(sobelx**2 + sobely**2)             # kombinacija
```

U ovoj implementaciji koristi se kombinacija horizontalnog (1,0) i vertikalnog (0,1) gradijenta s kernelom veličine 5×5. Tip podataka `CV_64F` osigurava preciznost izračuna jer gradijent može imati i negativne vrijednosti.

Laplaceov operator

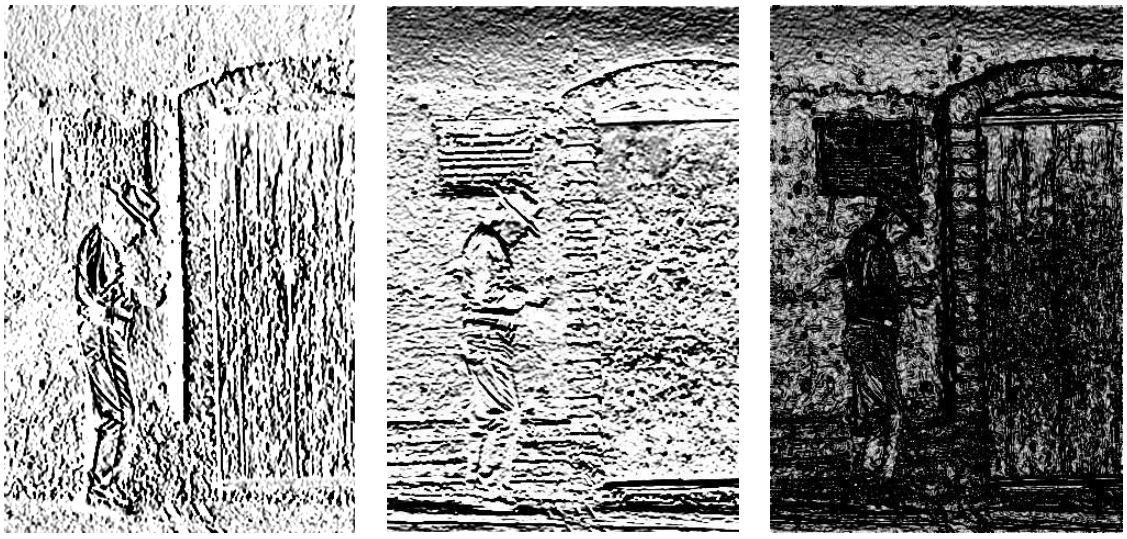
Za razliku od Sobelovog operatora, Laplaceov operator je izotropan - jednako osjetljiv na rubove u svim smjerovima.

```
result = cv2.Laplacian(img, cv2.CV_64F)
```

Laplaceov operator posebno je osjetljiv na šum u slici jer naglašava brze promjene intenziteta. Stoga se često kombinira s Gaussovim zamućivanjem u "Laplacian of Gaussian" pristupu.

Prewittov operator

Prewittov operator, slično Sobelovom operatoru, izračunava gradijent u horizontal-



x gradijent y gradijent kombinacija x i y

Slika 3.1. Primjer rezultata Sobelovog operatora kada su korišteni samo x, samo y i kombinacija oba gradijenta

nom i vertikalnom smjeru, ali koristi drugačije kernele.

```
edges = filters.prewitt(img / 255.0)
result = (edges * 255).astype(np.uint8)
```

Implementacija koristi biblioteku `skimage` koja zahtijeva normalizaciju ulazne slike na raspon [0,1]. Rezultat se zatim skalira natrag na raspon [0,255] za vizualizaciju.

Robertsov operator

Robertsov operator koristi najmanje kernele (2×2) za izračun gradijenta po dijagonalnim smjerovima [8]. Ova metoda je računski najjednostavnija, ali i najosjetljivija na šum zbog malih kernela.

```
edges = filters.roberts(img / 255.0)
result = (edges * 255).astype(np.uint8)
```

Kao i Prewittov operator, koristi `skimage` implementaciju s potrebom za skaliranjem vrijednosti.

Ograničenja tradicionalnih metoda

Tradicionalne metode nude jednostavnost implementacije i brzinu izvršavanja, što ih čini korisnima za aplikacije u realnom vremenu. Međutim, rezultati često sadrže pre-

kidne rubove i šum koje moderni algoritmi bolje rješavaju. Sve tradicionalne metode implementirane su kao jednostavne funkcije koje primaju sliku i vraćaju procesiranu verziju. Glavno ograničenje ovih implementacija je korištenje fiksnih parametara koji nisu optimirani za specifične slike [31].

3.2.2. Implementacija metoda dubokog učenja

Implementacija metode HED

U ovom radu metoda HED implementirana je koristeći OpenCV-jev Deep Neural Network (DNN) modul koji omogućava učitavanje i izvršavanje prethodno naučenih Caffe modela [14]. Implementacija se temelji na originalnom modelu HED dostupnom u službenom repozitoriju na GitHubu [32].

Predobrada (engl. *preprocessing*)

Model HED koristi specifičnu operaciju CropLayer koja nije standardno dostupna u OpenCV-ovom modulu DNN. Stoga je bilo potrebno implementirati prilagođen razred CropLayer. Ovaj razred implementira centralno obrezivanje (engl. *center cropping*) koje HED koristi za skaliranje bočnih izlaza na originalnu veličinu slike tijekom fuzijskog procesa. CropLayer automatski izračunava potrebne koordinate za obrezivanje na temelju razlike između ulaznih i ciljanih dimenzija [14].

```
cv2.dnn_registerLayer("Crop", CropLayer)
```

U glavnoj metodi stvara se *blob* koji koristi specifične srednje vrijednosti (104,00698793, 116,66876762, 122,67891434) koje odgovaraju skupu podataka ImageNet na kojem je naučen osnovni dio modela VGG-16 [14]. Parametar `swapRB=False` osigurava da kanali ostanu u redoslijedu BGR koji očekuje model Caffe, za razliku od standardnog redoslijeda RGB.

```
blob = cv2.dnn.blobFromImage(img, scalefactor=1.0, size=(W, H),
    mean=(104.00698793, 116.66876762, 122.67891434),
    swapRB=False, crop=False)
```

Implementacija glavne funkcije

Model se učitava iz dvije datoteke, `deploy.prototxt` za arhitekturu, a za naučene težišne vrijednosti učitava `hed_pretrained_bsds.caffemodel` [32].

```
def apply_hed(img, invert=True):
    ...
    net = cv2.dnn.readNetFromCaffe("models/hed_model/deploy.prototxt",
    "models/hed_model/hed_pretrained_bsds.caffemodel")
```

Glavno ograničenje implementacije je ovisnost o fiksiranim arhitekturalnim definicijama u datoteci `deploy.prototxt`, što otežava eksperimentiranje s modificiranim arhitekturama ili hiperparametrima.

Implementacija metode PiDiNed

Implementacija algoritma PiDiNet algoritma u ovom radu koristi radni okvir PyTorch s kombinacijom resursa iz različitih izvora. Prethodno naučeni model `table5_pidinet.pth` preuzet je iz repozitorija HuggingFace [33], dok se implementacija temelji na originalnim datotekama `pidinet.py`, `ops.py` i `config.py` preuzetim sa službenog repozitorija na GitHubu [34].

Konfiguracija modela

Model PiDiNet zahtijeva specifičnu konfiguraciju parametara koji definiraju arhitekturu i ponašanje algoritma:

```
class Args:
    config = 'carv4'  # konfiguracija modela
    sa = True         # prostorna pažnja uključena
    dil = True        # dilatacija uključena


args = Args()
model = pidinet(args).to(device)
```

Konfiguracija `carv4` odnosi se na standardnu varijantu arhitekture PiDiNet, odnosno onu koja je optimirana na balans između brzine i točnosti [16].

Učitavanje prethodno učenog modela

Model koristi prethodno naučeno i pohranjeno stanje (engl. *checkpoint*) modela `table5_pidinet.p`

```
checkpoint = torch.load("models/pidinet/table5_pidinet.pth",
                        map_location=device)
state_dict = checkpoint.get('state_dict', checkpoint)

# Uklanjanje 'module.' prefiksa ako je prisutan
new_state_dict = {}
for k, v in state_dict.items():
    if k.startswith('module.'):
        new_state_dict[k[7:]] = v
    else:
        new_state_dict[k] = v

model.load_state_dict(new_state_dict)
```

3.2.3. Implementacija ostalih metoda

Implementacija metode strukturiranih šuma

Implementacija u ovom radu koristi OpenCV `ximgproc` modul s prethodno učenim modelom. Prvo se provjerava format slike i provodi potrebna konverzija:

```
if img.ndim == 2 or img.shape[2] == 1:
    img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

img_float = img.astype(np.float32) / 255.0
```

Glavna detekcija provodi se kroz OpenCV implementaciju koja koristi prethodno učen model:

```
edge_detector = cv2.ximgproc.createStructuredEdgeDetection('structured_forest_model
edges = edge_detector.detectEdges(img_float)
edges = (edges * 255).astype(np.uint8)
```

Korišteni prethodno učeni model `structured_forest_model.yml` dobiven je s jav-

nog GitHub repozitorija [35] i sadrži parametre nasumičnih šuma učenih na BSDS500 skupu podataka. Ovaj model omogućava direktnu primjenu bez potrebe za dodatnim učenjem [36].

Prednosti implementacije

Glavna prednost ove implementacije je jednostavnost korištenja i brzina izvršavanja. Algoritam ne zahtijeva podršku GPU-a ili složene radne okoline strojnog učenja (engl. *deep learning frameworks*), što ga čini dostupnim na širokom spektru uređaja. Metoda postiže dobru ravnotežu između kvalitete detekcije i brzine obrade [18]. Međutim, ovisnost o prethodno učenom modelu ograničava mogućnosti prilagodbe metode specifičnim domenama ili vrstama slika.

Ova implementacija doprinosi cjelovitoj evaluaciji performansi različitih pristupa detekciji rubova, omogućavajući bolje razumijevanje prednosti i nedostataka svakog algoritma u različitim scenarijima.

Implementacija segmentacijske tehnike Watershed

Implementacija alogirtma Watershed u ovom radu koristi OpenCV biblioteku i slijedi standardni pristup segmentacije temeljene na markerima [19]. Algoritam je implementiran kao funkcija koja prima putanju slike i vraća masku granica između segmentiranih regija.

Predobrada

Implementacija počinje čitanjem slike i pretvorbom u sivu skalu, nakon čega slijedi binarizacija pomoću algoritma OTSU:

```
img = cv2.imread(image_path)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 0, 255,
                            cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

Prag u algoritmu OTSU automatski određuje optimalnu granicu za binarizaciju na temelju histograma slike [20]. Kombinacija s `THRESH_BINARY_INV` osigurava da objekti od interesa budu bijeli, a pozadina crna.

Morfološke operacije koriste se za uklanjanje šuma i definiranje sigurnih regija:

```
kernel = np.ones((3,3), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

# Sigurna pozadina
sure_bg = cv2.dilate(opening, kernel, iterations=3)

# Sigurni objekti (foreground)
dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
ret, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)
```

Primjena algoritma

Distance transform stvara sliku gdje svaki piksel ima vrijednost jednaku udaljenosti do najbliže granice objekta [22]. Threshold od 0,7 maksimalne vrijednosti osigurava da se kao sigurni *foreground* označavaju samo centralni dijelovi objekata.

```
# Nepoznata regija
unknown = cv2.subtract(sure_bg, sure_fg)

ret, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1  # pozadina postaje 1, ne 0
markers[unknown == 255] = 0  # nepoznata regija = 0

# Watershed algoritam
markers = cv2.watershed(img, markers)
```

Dodavanje 1 svim markerima osigurava da pozadinska regija nije označena kao 0, što je rezervirano za nepoznate regije.

Ova implementacija Watershed algoritma, iako tehnički ispravna, nije optimalna za detekciju rubova iz nekoliko razloga. Algoritam je dizajniran za segmentaciju objekata, ne za detekciju finijih rubnih struktura. Ovisnost o pragu OTSU čini ovaj algoritam osjetljivim na globalne varijacije osvjetljenja u slici.

Implementacija segmentacijske tehnike GrabCut

GrabCut algoritam predstavlja poluautomatsku metodu segmentacije koja kombinira teoriju grafova s *Gaussian Mixture Models* za izdvajanje objekata od interesa iz pozadine [25][27]. Algoritam je razvijen kao poboljšanje GraphCut metode s mogućnošću iterativnog poboljšanja segmentacije kroz korisničku interakciju.

Implementacija u ovom radu koristi automatsku inicijalizaciju umjesto korisničke interakcije. Slika se učitava i stvara se maska iste veličine:

```
mask = np.zeros(img.shape[:2], np.uint8)
rect = (int(w*0.1), int(h*0.1), int(w*0.8), int(h*0.8))
```

Pravokutnik se definira tako da obuhvaća centralnih 80% slike, pod pretpostavkom da se glavni objekti nalaze u središtu slike. Ova pretpostavka često nije valjana u realnim scenarijima.

GrabCut algoritam se pokreće s pet iteracija izgradnje modela kroz OpenCV funkciju:

```
cv2.grabCut(img, mask, rect, bgdModel, fgdModel, 5, cv2.GC_INIT_WITH_RECT)
```

Privremeni nizovi `bgdModel` i `fgdModel` sadrže parametre *Gaussian Mixture Models* za pozadinu i objekte od interesa. Algoritam iterativno poboljšava ove modele na temelju trenutne segmentacije.

Obrada rezultata

Nakon segmentacije, maska se pretvara u binarni format gdje vrijednosti 0 i 2 predstavljaju pozadinu, a 1 i 3 objekte od interesa:

```
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
contours, _ = cv2.findContours(mask2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(edge_map, contours, -1, 255, 1)
```

Funkcija `cv2.findContours` pronalazi granice segmentiranih regija, koje se zatim crtaju kao jednopikselske linije na izlaznoj slici. Ovaj pristup pretvara problem segmentacije u problem detekcije rubova.

Ograničenja implementacije

Automatska inicijalizacija predstavlja glavno ograničenje ove implementacije. GrabCut je dizajniran kao poluautomatska metoda koja ovisi o preciznoj korisničkoj inicijalizaciji [26]. Dodatno, algoritam pretpostavlja da objekti od interesa imaju konzistentnu distribuciju boja, što nije slučaj za kompleksne prirodne scene. Pet iteracija često nije dovoljno za konvergenciju u slučajevima gdje pozadina i objekti imaju slične karakteristike. Međutim, odabrano ograničenje od pet iteracija uvedeno je iz praktičnih razloga performansi i interaktivnosti web aplikacije. Svaka dodatna iteracija produžuje vrijeme obrade slike, što onemogućuje brzo isprobavanje više metoda u realnom vremenu.

Ovakva implementacija algoritma rezultira ocrtavanjem vanjskih rubova jednog, najčešće najvećeg i centralnog, objekta na slici kao što je vidljivo u kasnijem poglavlju usporedbe.

3.3. Tehnike naknadne obrade za metode dubokog i strojnog učenja

Tehnike naknadne obrade primijenjene su isključivo na rezultate algoritma temeljenih na dubokom učenju (HED, PiDiNet) i strojnom učenju (strukturirane šume), dok tradicionalne metode nisu podvrgnute dodatnoj obradi.

3.3.1. Razlozi za selektivnu primjenu naknadne obrade

Tradicionalne metode detekcije rubova poput Cannyja, Sobela, Laplaciana i sličnih algoritama proizvode binarne rezultate s jasno definiranim rubovima debljine jednog piksela. Ovi algoritmi koriste lokalne operatore i pragove koji već u svojoj osnovi stvaraju tanke i čiste rubove.

Nasuprot tome, metode dubokog učenja i strukturirane šume generiraju probabilističke karte rubova s varijabilnom gustoćom i intenzitetom. Modeli HED i PiDiNet proizvode kontinuirane vrijednosti koje predstavljaju vjerojatnost da se na određenoj poziciji nalazi rub [14][16]. Ovi rezultati često sadrže varijabilnu debljinu rubova ovisno o lokalnoj kompleksnosti slike te šum u obliku slabih i mutnih rubova.

3.3.2. Implementirane tehnike naknadne obrade

Sve implementirane funkcije naknadne obrade slijede sličnu petofaznu strukturu optimiranu za stvaranje čišćih rubova pogodnih za aplikacije poput bojanki ili tehničkih crteža.

Gaussovo zaglađivanje

Prvi korak uključuje blago zaglađivanje rezultata korištenjem Gaussovog filtera:

```
processed = cv2.GaussianBlur(hed_output, (3, 3), 0.4)
```

Ovo zaglađivanje uklanja visokofrekventni šum dok čuva glavne rubne strukture. Sigma vrijednost od 0,4 odabrana je empirijski kako bi se postigao optimalan balans između uklanjanja šuma i očuvanja detalja.

Morfološko zatvaranje

Zatvaranje malih prekida u rubovima postiže se morfološkim CLOSE operacijama:

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (ksize, ksize))
img = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

Eliptični kernel omogućava prirodnije spajanje prekida u odnosu na pravokutni kernel. Veličina kernela `ksize` prilagođava se razini željenog pojednostavljivanja (light, medium, heavy) pri pozivanju funkcije.

Kontrolirano zadebljanje rubova

Debljina konačnih rubova kontrolira se dilatacijom:

```
if line_thickness > 1:
    thickness_kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
              (line_thickness, line_thickness))
    processed = cv2.morphologyEx(processed, cv2.MORPH_DILATE,
    thickness_kernel)
```

Ova operacija omogućava kontrolu nad konačnim vizualnim izgledom rubova ovisno o namijenjnoj aplikaciji kao stvaranje bojanki.

Uklanjanje šuma

Konačni korak uklanja manje konture koje predstavljaju šum:

```
contours, _ = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
min_area = {'light':20, 'medium':30, 'heavy':50}[simplification]
mask = np.zeros_like(img)
for c in contours:
    if cv2.contourArea(c) >= min_area:
        cv2.drawContours(mask, [c], -1, 255, -1)
```

Minimalna površina kontura prilagođava se razini pojednostavljivanja, omogućavajući zadržavanje različitih razina detalja.

Ove tehnike naknadne obrade čine rezultate metoda dubokog učenja i strukturiranih šuma praktično primjenjivima u stvarnim scenarijima gdje se zahtijevaju čisti, konzistentni rubovi.

3.3.3. Specifične prilagodbe po algoritmima

Naknadna obrada metode HED

Rezultati metode HED često sadrže jake glavne rubove ali i slabije bočne izlaze koji mogu stvarati šum. Naknadna obrada za HED koristi konzervativniji pristup s manjim kernelima (3×3) kako bi očuvala fine detalje koje model uspješno detektira.

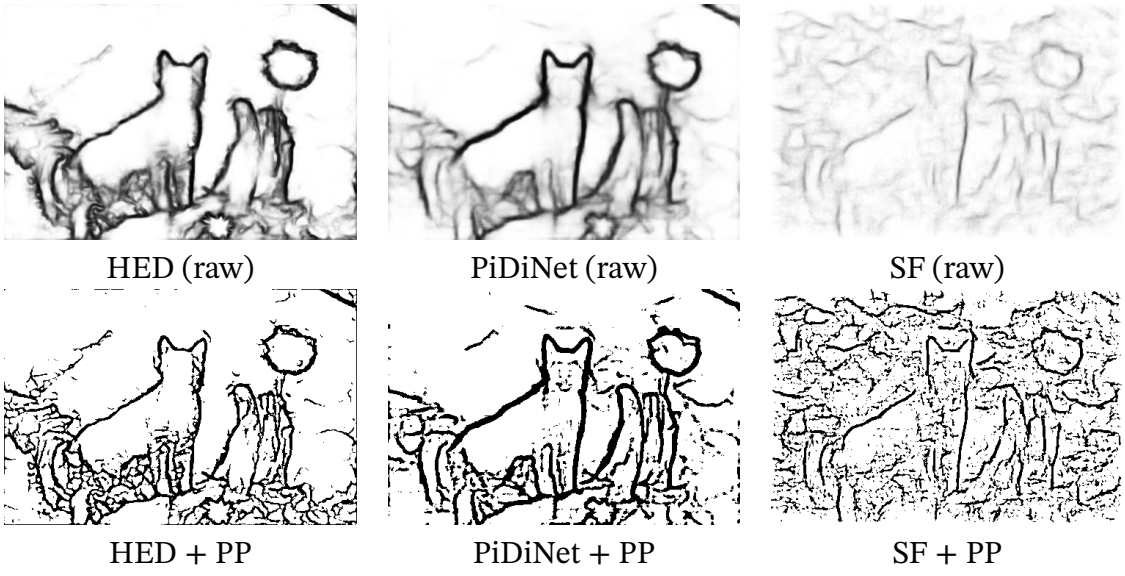
Naknadna obrada metode PiDiNet

PiDiNet općenito proizvodi čišće rezultate zbog svojih Pixel Difference Convolution blokova [16]. Naknadna obrada je prilagođena za optimizaciju rezultata za bojanje, s mogućnostima jednostavnog podešavanja debljine linija i razine detalja.

Naknadna obrada metode strukturiranih šuma

Kao najbrža od obrađenih metoda, metoda strukturiranih šuma često proizvodi nešto više šuma koji zahtijeva agresivnije čišćenje. Naknadna obrada uključuje veće kernele za zatvaranje i strože kriterije za uklanjanje malih artefakata.

Slika 3.2. Jedna od originalnih slika iz skupa BSDS500



Slika 3.3. Rezultati HED, PiDiNet i metode strukturiranih šuma, bez i s naknadnom obradom

3.4. Implementacija evaluacijskih metrika

Za objektivnu usporedbu i kvantitativnu evaluaciju kvalitete različitih algoritma detekcije rubova implementiran je skup standardnih metrika koji omogućava sveobuhvatnu analizu performansi. Metrike su dizajnirane da mjere različite aspekte kvalitete detekcije rubova, od osnovne točnosti do složenijih vizualnih karakteristika pogodnih za specifične primjene.

3.4.1. Priprema podataka za evaluaciju

Prije izračuna metrika provodi se standardizacija i provjera kompatibilnosti između predviđenih rezultata i referentnih oznaka:

```
def check_and_prepare(pred: np.ndarray, gt: np.ndarray,
tol_inversion: bool = True):
    # Provjera dimenzija
```

```
    if pred.shape != gt.shape:
        raise ValueError(f"Shape mismatch: pred {pred.shape}, gt {gt.shape}")

    # Binarizacija na {0,1}
    pred_bin = ((pred > 0) * 1).astype(np.uint8)
    gt_bin   = ((gt   > 0) * 1).astype(np.uint8)

    # Detekcija i korekcija inverzije polariteta
    if tol_inversion:
        ...
```

Ovaj korak osigurava da su svi podaci u konzistentnom formatu i polaritetu prije evaluacije [14][16] te iste veličine kako bi se točke linija obrađenih slika poklapale s točkama linija datoteka s referentnim oznakama.

3.4.2. Osnovne metrike detekcije rubova

Preciznost, odziv i F1-mjera

Temeljne metrike evaluacije implementirane su kroz funkciju `boundary_metrics` koja koristi toleranciju udaljenosti za robusniju evaluaciju:

```
def boundary_metrics(pred: np.ndarray, gt: np.ndarray, tol: int = 2):
    dt_gt = distance_transform_edt(1 - gt)
    tp = np.sum((pred == 1) & (dt_gt <= tol))
    fp = np.sum((pred == 1) & (dt_gt > tol))
    dt_pred = distance_transform_edt(1 - pred)
    fn = np.sum((gt == 1) & (dt_pred > tol))
```

Prije definiranja metrika, potrebno je objasniti osnovne pojmove klasifikacije:

- **Istinito pozitivni (engl. *True Positive* - *TP*)**: ispravno predviđeni pozitivni rezultati - stvarni rubovi koji su uspješno detektirani
- **Lažno pozitivni (engl. *False Positive* - *FP*)**: neispravno predviđeni pozitivni rezultati (greška I. vrste) - detektirani "rubovi" koji zapravo nisu rubovi

- **Lažno negativni (engl. *False Negative - FN*)**: propušteni pozitivni rezultati (greška II. vrste) - stvarni rubovi koji nisu detektirani
- **Istinito negativni (engl. *True Negative - TN*)**: ispravno predviđeni negativni rezultati - područja bez rubova koja su ispravno označena kao takva

Preciznost (engl. *precision*) mjeri koliko od detektiranih rubova stvarno jesu rubovi, definirano kao:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Visoka preciznost označava malo lažno pozitivnih detekcija, što je važno za aplikacije gdje šum može biti problematičan.

Odziv (engl. *Recall*) mjeri koliko stvarnih rubova je uspješno detektirano, definirano kao:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Visoki odziv osigurava da se ne propuštaju važni rubovi u sceni i označava malo lažno negativnih rubova.

F1-mjera predstavlja harmonijsku sredinu preciznosti i odziva:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Povećanje preciznosti može smanjiti odziv i obrnuto. F1-mjera balansira ovaj kompromis između dvije metrike, omogućavajući objektivan uvid u cjelokupnu kvalitetu algoritma [37].

3.4.3. Napredne metrike kvalitete

Prattova mjera kvalitete

Prattova mjera kvalitete (engl. *Pratt's Figure of Merit* - PFOM) kvantificira koliko su predviđeni rubovi geometrijski usklađeni sa stvarnima:

```
def pratt_figure_of_merit(pred: np.ndarray, gt: np.ndarray,
alpha: float = 1/9):
```

```
    dt = distance_transform_edt(1 - gt)
    d_i = dt[pred == 1]
    return np.sum(1.0 / (1 + alpha * (d_i ** 2))) / max(K, Kp)
```

PFOM vrijednosti blizu 1.0 označavaju odličnu lokalizaciju rubova, dok niže vrijednosti ukazuju na sistematske pomake ili duplikacije rubova. Parametar *alpha* od 1/9 je standardna vrijednost koja balansira osjetljivost na pomake [38].

Greška pomaka granica

Greška pomaka granica (engl. *Boundary Displacement Error* - BDE) mjeri prosječnu udaljenost između predviđenih i stvarnih rubova:

```
def boundary_displacement_error(pred: np.ndarray, gt: np.ndarray):
    dt_gt = distance_transform_edt(1 - gt)
    dt_pred = distance_transform_edt(1 - pred)
    d1 = dt_gt[pred == 1]
    d2 = dt_pred[gt == 1]
    return (d1.sum() + d2.sum()) / (len(d1) + len(d2) + 1e-8)
```

Manje BDE vrijednosti, bliže 0, označavaju bolju prostornu točnost algoritma. Ova metrika je posebno korisna za aplikacije gdje je precizna lokalizacija rubova kritična.

3.4.4. Metrike kontinuiranosti i zatvaranja

Rezultat kontinuiranosti

Rezultat kontinuiranosti (engl. *Continuity Score*) mjeri koliko su generirani rubovi povezani i kontinuirani:

```
def continuity_score(pred: np.ndarray, min_length: int = 50):
    num_labels, labels = cv2.connectedComponents(pred, connectivity=8)
    total_edges = pred.sum()
    continuity_count = 0
    for lab in range(1, num_labels):
        comp_size = np.sum(labels == lab)
        if comp_size >= min_length:
```

```
            continuity_count += comp_size
    return continuity_count / total_edges
```

Ova metrika je ključna za aplikacije poput stvaranja bojanki gdje su potrebni kontinuirani, neprekinuti rubovi. Viši rezultati označavaju manje fragmentiranih rubova.

Stopa zatvaranja petlji

Stopa zatvaranja petlji (engl. *Loop Closure Rate*) evaluira koliko povezanih komponenti formira zatvorene oblike:

```
def loop_closure_rate(pred: np.ndarray):
    contours, _ = cv2.findContours(pred.astype(np.uint8),
                                   cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)
    closed_loops = sum(1 for cnt in contours
                       if cv2.isContourConvex(cnt) or np.all(cnt[0] == cnt[-1]))
    return closed_loops / len(contours)
```

Ova metrika je važna za aplikacije koje zahtijevaju zatvorene regije, poput segmentacije ili bojenja unutrašnjosti objekata.

3.4.5. Cjelokupni evaluacijski proces

Glavna funkcija metrika `calculate_all_metrics` integrira sve metrike u cjelovit evaluacijski proces:

1. Učitavanje i priprema: Čitanje predviđenih rezultata i odgovarajućih *ground truth* anotacija iz BSDS500 skupa podataka

2. Normalizacija: Pretvaranje slika u standardni [0,1] format s automatskim detektiranjem inverzije polariteta

3. Binarizacija: Primjena praga (standardno 0,1) za pretvaranje probabilističkih karata u binarne rubove

4. Evaluacija: Izračun svih implementiranih metrika

5. Povrat rezultata: Strukturirana prezentacija rezultata za daljnju analizu

```
    bm = boundary_metrics(pred_clean, gt_clean, tol=1)
    pfom = pratt_figure_of_merit(pred_clean, gt_clean)
    bde = boundary_displacement_error(pred_clean, gt_clean)
    cont  = continuity_score(pred_clean, min_length=50)
    close = loop_closure_rate(pred_clean)

    return {
        'precision': round(bm['precision'], 4),
        'recall':    round(bm['recall'],    4),
        'f1':        round(bm['f1'],        4),
        'pfom':      round(pfom,            4),
        'bde':       round(bde,             4),
        'continuity':round(cont,            4),
        'closure':   round(close,           4),
        'gt_file':   os.path.basename(gt_path)
    }
```

3.4.6. Interpretacija rezultata

Kombinacija svih implementiranih metrika omogućava sveobuhvatnu evaluaciju algoritama detekcije rubova:

- F1-mjera i PFOM daju općenitu ocjenu kvalitete detekcije
- BDE kvantificira prostornu točnost
- Kontinuitet evaluira praktičnost za aplikacije koje zahtijevaju povezane rubove
- Mjera zatvorenih petlji mjeri pogodnost za segmentacijske zadatke

Ova kombinacija metrika omogućava objektivan uvid u prednosti i nedostatke različitih pristupa te pomaže pri donošenju informirane odluke o odabiru algoritma za specifične aplikacije. Implementirane metrike čine temelj za kvantitativnu usporedbu algoritma prikazanu u sljedećem poglavlju ovog rada.

4. Usporedba rezultata

Naknadna obrada značajno poboljšava vizualnu kvalitetu rezultata na račun nešto smanjene preciznosti u odnosu na originalne ground truth anotacije. Međutim, za praktične aplikacije poput stvaranja bojanki ili tehničkih crteža, ove tehnike su nezaobilazne za postizanje profesionalnih rezultata.

4.1. Eksperimentalni rezultati

4.1.1. Rezultati tradicionalnih metoda

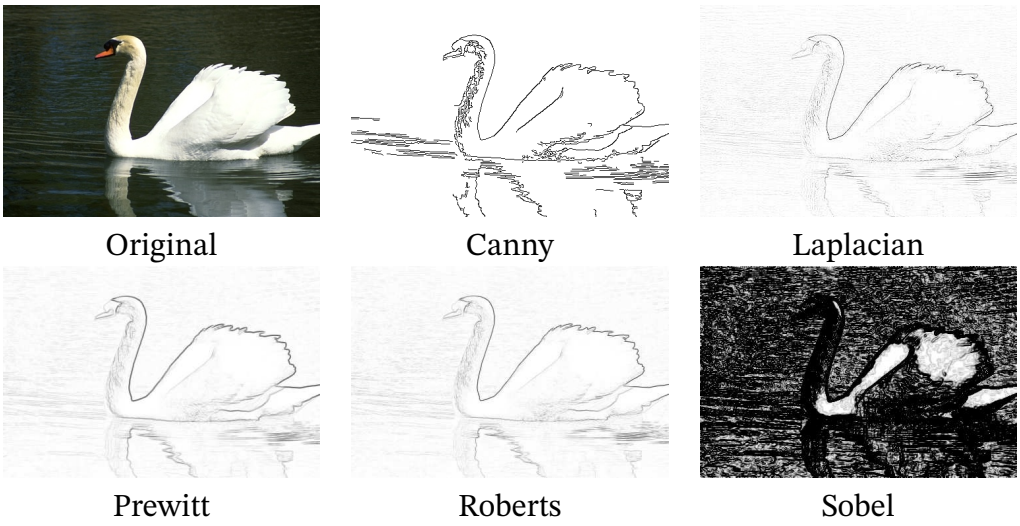
U ovoj analizi uspoređeno je pet tradicionalnih metoda detekcije rubova (Canny, Laplacian, Prewitt, Roberts, Sobel) na skupu od 200 reprezentativnih slika iz skupa BSDS500 koristeći mjere objašnjene u prethodnom poglavlju. Skup slika je odabran tako da sadrži dobru raznolikost. Sadrži slike velikog kontrasta objekta i pozadine gdje je očekivana dobra detekcija, a time i dobre mjere. Također sadrži slike manjih kontrasta za realniju procjenu uspješnosti algoritama.

Jedna od slika boljih rezultata evaluacijskih metrika tradicionalnih metoda bila je slika najjednostavnijeg sadržaja i najvećeg kontrasta.

Tablica 4.1. Rezultati evaluacijskih metrika tradicionalnih metoda nad slikom visokog kontrasta

Metoda	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
Canny	0,2240	0,6523	0,3335	0,4068	14,5702	0,7326	0,0929
Laplacian	0,3476	0,4509	0,3926	0,5796	4,3261	0,0735	0,4785
Prewitt	0,3794	0,8091	0,5166	0,6041	4,9775	0,8233	0,2489
Roberts	0,4547	0,7366	0,5623	0,6364	4,1502	0,7204	0,3280
Sobel	0,0469	1,0000	0,1511	0,1255	41,2065	0,9984	0,8844

Na ovoj slici tradicionalni operatori pokazuju vrlo različite performanse. Vizualno, Cannyjev je najbolji jer zadržava čiste i kontinuirane rubove bez pretjeranog šuma, iako



Slika 4.1. Rezultati tradicionalnih metoda na slici visokog kontrasta (slijeva udesno): Canny, Laplacian, Prewitt, Roberts i Sobel

mu F1-mjera (0,3335) signalizira umjeren kompromis između preciznosti (0,2240) i odziva (0,6523). Robertsov operator postiže najvišu F1-mjeru (0,5623) i najmanji pomak rubova (BDE 4,1502), ali pri tome gubi na vizualnoj preglednosti zbog nešto više prekinutih linija u odnosu na Cannyja. Prewittov operator pokazuje uravnotežen odziv (0,8091) i dobru prostornu točnost (BDE 4,9775), dok Laplacian visokom preciznošću (0,3476) i niskim BDE-om (4,3261) favorizira samo izražene rubove, propuštajući fine detalje. Sobelov operator ima odziv 1 što je odličan rezultat, međutim količina šuma čini sliku nekoristivom.

Sljedeća tablica pokazuje prosječne vrijednosti rezultata svake od tradicionalnih metoda kroz svih 200 slika.

Tablica 4.2. Prosječne vrijednosti metrika za 200 testnih slika

Metoda	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
Canny	0,1280	0,6792	0,2035	0,2364	23,7598	0,6491	0,2058
Laplacian	0,0832	0,6314	0,1081	0,1600	27,3101	0,5775	0,7995
Prewitt	0,1738	0,5439	0,1787	0,2341	24,9071	0,6098	0,4280
Roberts	0,1066	0,7064	0,1309	0,1742	28,5184	0,6961	0,6019
Sobel	0,0671	0,9973	0,1240	0,1677	29,5824	0,9912	0,7715

Analiza rezultata

Prosječni rezultati pokazali su jasno različite profile svakog od pet tradicionalnih operatora detekcije rubova. Sobelov operator postiže gotovo savršen prosječan odziv (0,9973), ali njegova izuzetno niska preciznost (0,0671) ukazuje na veliku količinu lažnih rubova,

što se očituje u najnižoj prosječnoj F1-mjeri (0,1240) i vrlo visokom prosječnom BDE-u (29,5824). Unatoč tome, operator Sobela prednjači u kontinuitetu (0,9912) i zatvaranju petlji (0,7715), što znači da je generirao povezane i zatvorene linije, no praktična upotrebljivost tih linija je ograničena zbog šuma.

Cannyjev operator se ističe kao uravnoteženija opcija, s drugom najvišom prosječnom F1-mjerom (0,2035) i solidnim odnosom preciznosti (0,1280) i recalla (0,6792). Njegova prosječna vrijednost BDE-a (23,7598) ukazuje na umjerenu prostornu točnost, a mjera kontinuiteta (0,6491) potvrđuje relativno dobru povezanost linija bez prekomjernog šuma.

Prewittov i Robertsov operator ostvaruju vrlo slične prosječne F1-vrijednosti (0,1787 odnosno 0,1309), pri čemu Prewittov donosi nešto viši odziv (0,5439) i kontinuitet (0,6098), a Robertsov ipak nudi nešto veću preciznost (0,1066) i najbolji prosječni PFOM (0,1742). Njihovi prosječni BDE-ovi (24,9071 i 28,5184) svrstavaju ih među najtočnije po prostornoj preciznosti.

Laplaceov operator, s prosječnom preciznošću od 0,0832 i niskim odzivom (0,6314), postiže prosječnu F1-mjeru od 0,1081. Njegov prosječni BDE (27,3101) i kontinuitet (0,5775) pokazuju da detektira samo najočitije rubove, što rezultira vrlo otvorenim i fragmentiranim linijama.

Zaključno, operatori Prewitta i Robertsa pružaju najbolji kompromis između preciznosti, odziva i prostorne točnosti, dok Cannyjev nudi najbolju vizualnu čistoću i povezanost rubova. Sobelov, iako kontinuiran i zatvoren, proizvodi previše lažnih rubova da bi bio praktičan u primjenama. Ovaj uvid potvrđuje potrebu za naprednijim pristupima dubokog učenja koji će biti obrađeni u kasnijem poglavlju.

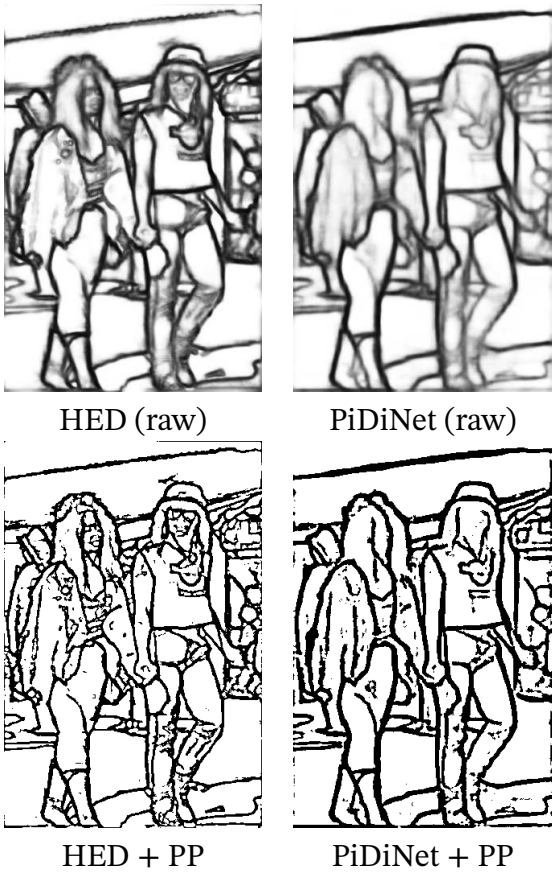
4.1.2. Rezultati metoda dubokog učenja

Metode dubokog učenja HED i PiDiNet nadmašuju tradicionalne operatore u složenim scenama zahvaljujući učenju semantičkih i kontekstualnih informacija. U nastavku uspoređujemo njihove rezultate na istih 200 testnih slika iz BSDS500 skupa na kojima su i tradicionalne metode uspoređivane, i to prije i nakon obavljanja naknadne obrade.

Ovo je primjer slike složenog sadržaja s kompliciranim objektom, mnogo linija i pro-

mjena boja te kompliciranom pozadinom.

Slika 4.2. Originalna slika kompliciranog sadržaja



Slika 4.3. Rezultati HED i PiDiNet bez i s naknadnom obradom na slici kompliciranog sadržaja

Nad slikama je obavljena i naknadna obrada te su zabilježeni rezultati metrika oboje nad slikama prije i poslije naknadne obrade.

Tablica prikazuje rezultate metoda HED i PiDiNet na slici složenog sadržaja, prije i nakon naknadne obrade. U sirovom (engl. *raw*) obliku, PiDiNet nadmašuje HED marginalno višom F1-mjerom (0,3829 naspram 0,3712) i nižim BDE-om (5,1559 naspram 5,7361), što ukazuje na bolju prostornu preciznost i lokalizaciju rubova. Naknadna

Tablica 4.3. Evaluacijske metrike za sliku kompliciranog sadržaja

Metoda	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
HED (raw)	0,2281	0,9969	0,3712	0,4801	5,7361	0,9983	0,1111
PiDiNet (raw)	0,2378	0,9816	0,3829	0,4997	5,1559	0,9988	0,0482
HED + PP	0,3103	0,9676	0,4700	0,5356	5,1891	0,9808	0,3367
PiDiNet + PP	0,3040	0,9687	0,4628	0,5425	5,4343	0,9761	0,2451

obrada značajno poboljšava preciznost obje metode, pri čemu HED ostvaruje najveći relativni porast (s 0,2281 na 0,3103). Nakon naknadne obrade, HED vodi u F1-mjeri (0,4700 naspram 0,4628), dok PiDiNet + PP ipak zadržava minimalno bolje poravnavanje rubova izraženo PFOM-om (0,5425 naspram 0,5356). Kontinuitet za obje metode ostaje visoko ($> 0,97$), što potvrđuje njihovu sposobnost generiranja koherentnih linija unatoč kompleksnom sadržaju slike. Zatvaranje petlji raste kod HED-a s naknadnom obradom (0,3367) u odnosu na *raw* oblik (0,1111), dok se kod PiDiNet povećava s 0,0482 na 0,2451, pokazujući učinak naknadne obrade u zatvaranju rubnih petlji.

Ukratko, naknadna obrada značajno poboljšava uravnoteženost preciznosti i odziva te prostornu točnost obje duboke metode na složenim slikama.

Analiza rezultata

Prosječne vrijednosti metrika za HED i PiDiNet, prikazane u tablici, jasno iskazuju prednosti i nedostatke svake metode te utjecaj naknadne obrade na 200 testnih slika.

Tablica 4.4. Prosječne vrijednosti metrika za HED i PiDiNet (s i bez naknadne obrade) na 200 testnih slika

Metoda	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
HED (raw)	0,2653	0,9074	0,4031	0,4825	11,7196	0,9851	0,2406
PiDiNet (raw)	0,2809	0,8510	0,4150	0,5294	9,0605	0,9897	0,1068
HED + PP	0,2248	0,9383	0,3556	0,3727	18,5457	0,9429	0,3423
PiDiNet + PP	0,2226	0,9265	0,3531	0,4095	14,8633	0,9382	0,3501

Bez naknadne obrade, PiDiNet nadmašuje HED u svim ključnim metrikama: postiže veću preciznost (0,2809 naspram 0,2653), nešto nižu, ali i dalje vrlo visoki prosječni odziv (0,8510 naspram 0,9074), višu F1-mjeru (0,4150 naspram 0,4031), bolji PFOM (0,5294 naspram 0,4825) i niži prosječni BDE (9,0605 naspram 11,7196). Vrijednosti za kontinuitet i zatvaranje petlja ostaju blizu maksimuma za obje metode, no PiDiNet bez naknadne obrade ostvaruje minimalno manje zatvorene rubne petlje.

Naknadna obrada donosi značajan porast preciznosti i F1-mjere za obje metode: HED se poboljšava za oko 31% u preciznosti i 23% u F1, dok PiDiNet bilježi oko 23% bolje preciznosti i 17% bolje F1. Unatoč većem apsolutnom porastu kod HED, PiDiNet s naknadnom obradom i dalje pokazuje bolje prosječne vrijednosti za preciznost (0,2226 naspram 0,2248), F1 (0,3531 naspram 0,3556) i PFOM (0,4095 naspram 0,3727). BDE zbog agresivnijeg zatvaranja rubova raste kod obje metode, pri čemu HED s naknadnom obradom ide s 11,7196 na 18,5457, a PiDiNet s 9,0605 na 14,8633. Vrijednosti kontinuiteta i zatvaranja pretlji dopuštaju donekle veće kompromise nakon naknadne obrade, no ostaju visoki (>0,94), potvrđujući sposobnost obje mreže za generiranje koherentnih i zatvorenih mapa rubova.

Zanimljivo je kako na slici nižih kontrasta HED + PP ocrtava male detalje u zidovima dok ih PiDiNet + PP ne detektira.

Original HED + PP PiDiNet + PP

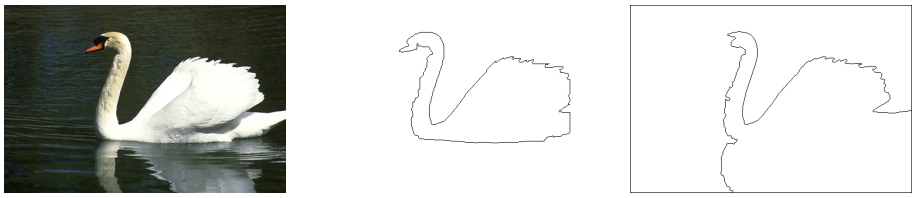
Slika 4.4. Rezultati HED i PiDiNet bez i s naknadne obrade na slici nižeg kontrasta

U konačnici, PiDiNet + PP pruža najbolji ukupni balans između preciznosti, odziva, prostorne točnosti (nizak BDE) i kvalitete lokalizacije (PFOM), dok HED + PP nudi snažan kontinuitet i vrlo stabilne rezultate zahvaljujući izvorno visokom odzivu.

4.1.3. Rezultati segmentacijskih metoda

U ovoj sekciji uspoređujemo dvije segmentacijske tehnike, GrabCut i Watershed. Primijenjene su na pet odabranih testnih slika iz sskupa BSDS500 kao i metode do sada.

Dobre vizualne rezultate stvaraju nad slikom koja ima visoki kontrast između objekta i pozadine.



Original GrabCut Watershed

Slika 4.5. Rezultati segmentacijskih tehnika na slici visokog kontrasta

Analiza rezultata

Prosječne vrijednosti evaluacijskih metrika jasno pokazuju da GrabCut u prosjeku detektira sve rubove (odziv=1.00) dok Watershed pokriva samo oko 11% stvarnih rubova, što rezultira vrlo niskom F1-mjerom (0,1103).

Tablica 4.5. Prosječne vrijednosti metrika za GrabCut i Watershed na 200 testnih slika

Tehnika	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
GrabCut	0,0824	0,9221	0,1119	0,1567	31,1868	0,9941	0,6138
Watershed	0,1138	0,1255	0,1130	0,1630	42,2793	0,9973	0,0021

Međutim, GrabCut žrtvuje preciznost zbog brojnih lažnih rubova (preciznost=0,0824), što dovodi do visokog prosječnog BDE-a (31,19). S druge strane, Watershed, unatoč skromnom odzivu, postiže nešto bolju preciznost (0,1138) i niži PFOM (0,1630), no njegov prosječni BDE od 42,28 ukazuje na lošu prostornu točnost. Vrijednost kontinuiteta od 0,99 za GrabCut ukazuje na potpuno povezane linije, dok je Watershed gotovo jednak s 0,9973, ali Watershed ne zatvara rubne petlje u prosjeku (0,21%), dok GrabCut ima prosječno zatvaranje petlji od oko 61%.

Ovi rezultati sugeriraju da GrabCut dobro zadržava kontinuitet i pokriva većinu rubova (odziv 0,9221), dok Watershed bilježi samo mali dio rubova (odziv 0,1255) i praktički ne zatvara rubne petlje (0,21% naspram 61% kod GrabCut-a).

4.2. Kvantitativna analiza svih metoda

U ovom radu uspoređene su četiri klase metoda za detekciju rubova i segmentaciju:

- Tradicionalne metode (Canny, Laplacian, Prewitt, Roberts, Sobel)
- Metode dubokog učenja (HED, PiDiNet)

- Segmentacijske tehnike (GrabCut, Watershed)
- Metoda strojnog učenja strukturirane šume

Analiza se temelji na sedam evaluacijskih metrika: preciznost, odziv, F1-mjera, PFOM, BDE, kontinuitet i vrijednost zatvorene petlje.

Tablica 4.6. sažima prosječne rezultate svih testiranih metoda na 200 testnih slika.

Tablica 4.6. Komparativna analiza prosječnih vrijednosti metrika

Metoda	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
Canny	0,1280	0,6792	0,2035	0,2364	23,7598	0,6491	0,2058
Laplacian	0,0832	0,6314	0,1081	0,1600	27,3101	0,5775	0,7995
Prewitt	0,1738	0,5439	0,1787	0,2341	24,9071	0,6098	0,4280
Roberts	0,1066	0,7064	0,1309	0,1742	28,5184	0,6961	0,6019
Sobel	0,0671	0,9973	0,1240	0,1677	29,5824	0,9912	0,7715
HED (raw)	0,2653	0,9074	0,4031	0,4825	11,7196	0,9851	0,2406
PiDiNet (raw)	0,2809	0,8510	0,4150	0,5294	9,0605	0,9897	0,1068
HED + PP	0,2248	0,9383	0,3556	0,3727	18,5457	0,9429	0,3423
PiDiNet + PP	0,2226	0,9265	0,3531	0,4095	14,8633	0,9382	0,3501
GrabCut	0,0824	0,9221	0,1119	0,1567	31,1868	0,9941	0,6138
Watershed	0,1138	0,1255	0,1130	0,1630	42,2793	0,9973	0,0021
Structured Forest + PP	0,1939	0,8030	0,3041	0,3328	19,2326	0,7567	0,4114

Rezultati pokazuju da PiDiNet s naknadnom obradom ostvaruje najbolji omjer preciznosti, odziva, F1-mjere i PFOM-a, uz zadržanu prostornu preciznost (nizak BDE) i visoki kontinuitet linija. Među tradicionalnim operatorima, Robertsov i Prewittov ostvaruju umjereno dobre F1-vrijednosti, no pate od kompromisa između preciznosti i odziva. Sobelov je najpouzdaniji u odzivu, ali proizvodi previše šuma. Segmentacijske metode ne mogu dosegnuti performanse detekcije rubova. GrabCut bilježi savršen odziv, ali s vrlo niskom preciznošću, dok Watershed zadržava čistoću linija uz zanemarivanje većine rubova. Metoda strukturirane šume uz naknadnu obradu nude konkurentan F1 i PFOM, no ne doseže razinu metoda dubokog učenja.

Za primjer vizualne usporedbe slika višeg i nižeg kontrasta koristimo 4.6. i 4.7. koje prikazuju rezultate slika procesiranih tradicionalnim i modernim metodama.

4.2.1. Analiza varijabilnosti rezultata putem standardne devijacije

Standardna devijacija pruža važan uvid u konzistentnost performansi različitih metoda detekcije rubova kroz testni skup od 200 slika. Analiza varijabilnosti otkriva značajne razlike u stabilnosti algoritma koje mogu utjecati na njihovu praktičnu primjenu.

Tradicionalne metode pokazuju različite obrasce varijabilnosti. Algoritam Canny demonstrira umjerenu konzistentnost s standardnom devijacijom od 0,0816 za preciznost i 0,1447 za odziv, što ukazuje na predvidljivu performansu kroz različite tipove slika. Sobelov operator, unatoč visokim prosječnim vrijednostima za kontinuitet, pokazuje izuzetno nisku varijabilnost (std=0,0119), što potvrđuje njegovu stabilnost u generiranju povezanih rubova. Nasuprot tome, Prewittov i Robertsov operator pokazuju veću varijabilnost u preciznosti (std=0,1670 i 0,1126), što sugerira osjetljivost na karakteristike slika.

Metode dubokog učenja pokazuju značajno manju varijabilnost u ključnim metrikama. HED postiže standardnu devijaciju od samo 0,0763 za preciznost i 0,0908 za odziv, što ukazuje na robusnost neuronske mreže. PiDiNet je još konzistentniji s standardnom devijacijom od 0,0613 za preciznost, ali nešto lošijom za odziv, 0,1335. Ova niska varijabilnost pruža predvidljive rezultate neovisno o sadržaju slike i objašnjava zašto su duboke metode preferirane u praktičnim primjenama.

Strukturirane šume pokazuju zanimljiv kontrast između sirove metode i varijante s naknadnom obradom. Osnovna verzija strukturiranih šuma demonstrira iznimno visoku varijabilnost u preciznosti (std=0,2573), što je među najgorim rezultatima u cijelom testnom skupu. Ova visoka nesigurnost, zajedno s velikim odzivom varijabilnosti (std=0,3326), ukazuje na to da algoritam vrlo nepredvidljivo reagira na različite tipove slika. Međutim, verzija s naknadnom obradom značajno poboljšava konzistentnost, smanjujući standardnu devijaciju preciznosti na 0,0829 i odziva na 0,0959. Ova dramatična stabilizacija (smanjenje varijabilnosti za preko 70%) pokazuje važnost naknadne obrade za praktičnu primjenu strukturiranih šuma, čineći ih konkurentnima dubokim metodama u pogledu predvidljivosti.

Naknadna obrada generalno povećava varijabilnost kod dubokih metoda zbog dodat-

nih transformacija. HED s naknadnom obradom pokazuje povećanje standardne devijacije s 0,0763 na 0,0796 za preciznost, dok PiDiNet bilježi porast s 0,0613 na 0,0667. Ovo povećanje varijabilnosti je kompromis za poboljšanje prosječnih performansi. Suprotno tome, strukturirane šume drastično smanjuju varijabilnost naknadnom obradom.

Segmentacijske metode pokazuju ekstremne razlike u konzistentnosti. Watershed segmentacija demonstrira relativno nisku varijabilnost (std=0,0766 za preciznost), ali GrabCut pokazuje nestabilnost (std=0,1090 za preciznost i 0,2361 za odziv), što objašnjava nekonzistentnost u praktičnoj primjeni.

Tablica 4.7. Standardne devijacije metrika za sve metode

Metoda	Preciznost	Odziv	F1	PFOM	BDE	CS	LC
Canny	0,0816	0,1447	0,1049	0,1157	14,1892	0,0915	0,0657
Laplacian	0,0686	0,4278	0,0562	0,0715	13,6520	0,4875	0,1732
Prewitt	0,1670	0,3786	0,1252	0,1382	14,8666	0,3534	0,1584
Roberts	0,1126	0,3923	0,0897	0,0914	14,3523	0,4123	0,1759
Sobel	0,0326	0,0069	0,0557	0,0665	13,4906	0,0119	0,0856
HED (raw)	0,0763	0,0908	0,0914	0,1155	9,1809	0,0112	0,0869
PiDiNet (raw)	0,0613	0,1335	0,0724	0,0939	7,0327	0,0088	0,0601
HED + PP	0,0796	0,0572	0,1019	0,1090	11,3175	0,0239	0,0533
PiDiNet + PP	0,0667	0,0806	0,0862	0,1042	9,3566	0,0256	0,0474
Structured Forest (raw)	0,2573	0,3326	0,1666	0,2006	14,4269	0,0837	0,1035
Structured Forest + PP	0,0829	0,0959	0,1063	0,1152	11,6779	0,0810	0,0217
GrabCut	0,1090	0,2361	0,0741	0,0819	13,3852	0,0253	0,1282
Watershed	0,0766	0,0965	0,0749	0,0871	17,8879	0,0082	0,0212

Ukupno, najkonzistentnije metode su PiDiNet (raw) i HED (raw), što potvrđuje stabilnost dubokih neuronskih mreža. Strukturirane šume s naknadnom obradom također pokazuju dobru konzistentnost, dok sirova varijanta pokazuje najveću varijabilnost među svim testiranim metodama. Tradicionalne metode pokazuju veću varijabilnost, osobito u odzivu, dok segmentacijske tehnike variraju značajno u konzistentnosti ovisno o pristupu.

4.2.2. Usporedba brzine izvođenja metoda

Za potrebe praktične primjene algoritama detekcije rubova i segmentacije, osim analize točnosti i stabilnosti, ključno je razumjeti i njihovu brzinu izvođenja.

Tradicionalne metode (Canny, Sobel, Laplacian, Prewitt, Roberts) pokazale su iznimno niske prosječne vremenske zahtjeve, svi ispod 0,006 sekundi po slici. Najbrži

je bio Laplaceov operator (prosjek 0,0023s), dok su Prewittov i Robertsov nešto sporiji (0,0058 s i 0,0055 s). Standardna devijacija je pritom minimalna, što sugerira vrlo konzistentno izvođenje ovih metoda čak i na različitim slikama. Ovakva efikasnost dolazi iz jednostavnosti konvolucijskih operacija koje su inherentno idealne za primjene u realnom vremenu i obradu velikih skupova.

Segmentacijska metoda Watershed također pokazuje vrlo brzu izvedbu (0,0078 s), dok GrabCut značajno odskače s prosječnim vremenom od 1,34 sekunde i izrazito velikom varijabilnosti izvođenja (std=0,62 s). Ova velika varijabilnost potječe od iterativne prirode GrabCut algoritma i složenosti odvajanja prednjeg plana koji ovisi o sadržaju svake slike.

Metode temeljene na dubokom učenju (HED, PiDiNet, kao i njihove verzije s naknadnom obradom) su red veličine sporije u odnosu na tradicionalne metode: HED prosječno zahtijeva 0,73 sekunde, a PiDiNet 0,92 sekunde po slici. Naknadna obrada minimalno povećava vrijeme izvođenja kod obje metode. Standardna devijacija vremena je mala za metodu HED (0,04 s) i PiDiNet (0,03 s), što ukazuje na stabilnu izvedbu bez obzira na kompleksnost ulaznih slika. Ovo vrijeme je posljedica izvođenja složenih mrežnih arhitektura na CPU-u (ili manje optimiziranom GPU-u), ali je još uvijek prihvatljivo za mnoge manje analize.

Strukturirane šume su sporije od svih ostalih metoda (osim GrabCuta), s prosječnim vremenom oko 1,18 sekundi po slici, te najvećom standardnom devijacijom među metodama slične složenosti (0,11 s). Naknadna obrada strukturiranih šuma blago smanjuje vrijeme izvođenja i varijabilnost. Sporost ovih metoda proizlazi iz karaktera njihove predikcije (klasifikacija svake slike putem stabala).

Tablica 4.8. Prosječna vremena izvođenja i standardne devijacije za sve metode na 200 testnih slika

Metoda	Prosječno vrijeme (s)	Std. devijacija (s)
Canny	0,0044	0,0030
Sobel	0,0047	0,0003
Laplacian	0,0023	0,0003
Prewitt	0,0058	0,0005
Roberts	0,0055	0,0004
HED	0,7266	0,0408
PiDiNet	0,9233	0,0327
HED + PP	0,7557	0,0280
PiDiNet + PP	0,9319	0,0554
Structured Forest	1,1833	0,1083
Structured Forest + PP	1,1580	0,0482
Watershed	0,0078	0,0011
GrabCut	1,3376	0,6189

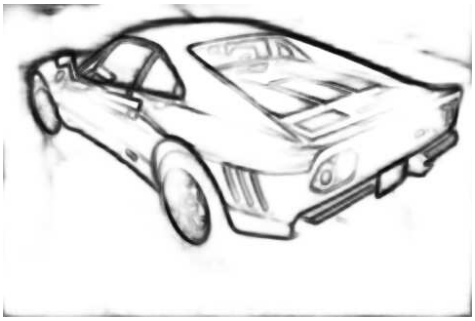
Tradicionalne metode predstavljaju optimalan izbor za primjene koje zahtijevaju obradu u realnom vremenu zbog svoje brzine ispod 0,006 sekundi po slici. Metode dubokoga učenja, HED i PiDiNet, pružaju značajno bolju preciznost uz umjeren kompromis u brzini od približno 0,7-0,9 sekundi po slici, što ih čini prikladnima za manje obrade gdje je kvaliteta prioritet. Strukturirane šume i GrabCut pokazuju najlošije performanse po pitanju brzine, pri čemu je GrabCut dodatno opterećen velikom nestabilnošću izvođenja koja može varirati od slike do slike, što predstavlja značajan izazov za produkcijske sustave koji zahtijevaju predvidljivo ponašanje.

4.2.3. Usporedba s literaturom

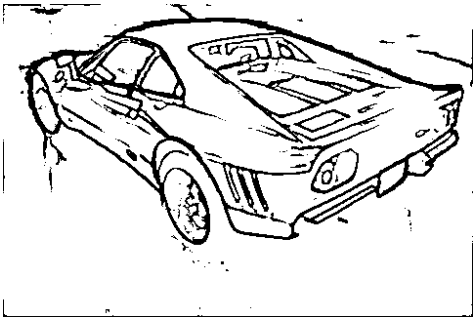
Dobiveni rezultati potvrđuju zaključke iz prethodnih radova da metode dubokog učenja nadmašuju tradicionalne operatore u složenim scenama [1]. Konkretno, PiDiNet je u radu [16] iskazan kao metoda s visokom PFOM i niskim BDE, dok je HED prepoznat po izvanrednoj kontinuiranosti rubova [14]. Segmentacijske tehnike kao što je GrabCut često se koriste za inicijalnu masku objekta, ali ne daju detaljnu detekciju rubova [39]. Metoda strukturiranih šuma pokazuje dobru ravnotežu između brzine i kvalitete rubne

karte, u skladu s rezultatima iz [18]. Evaluacija na BSDS500 skupu [37] omogućava poštenu usporedbu s ostalom literaturom [38].

Original

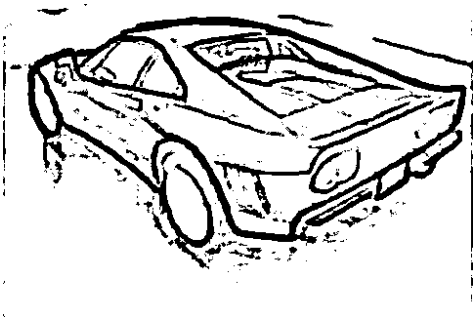


HED

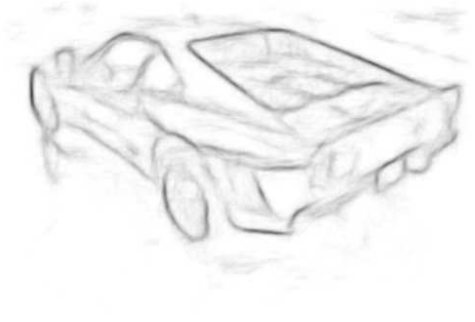


HED + PP

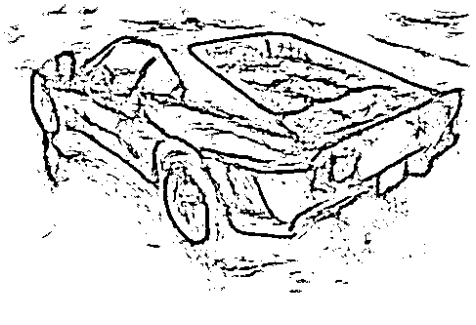
PiDiNet



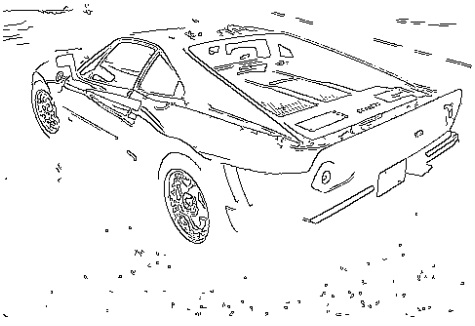
PiDiNet + PP



strukturirane šume



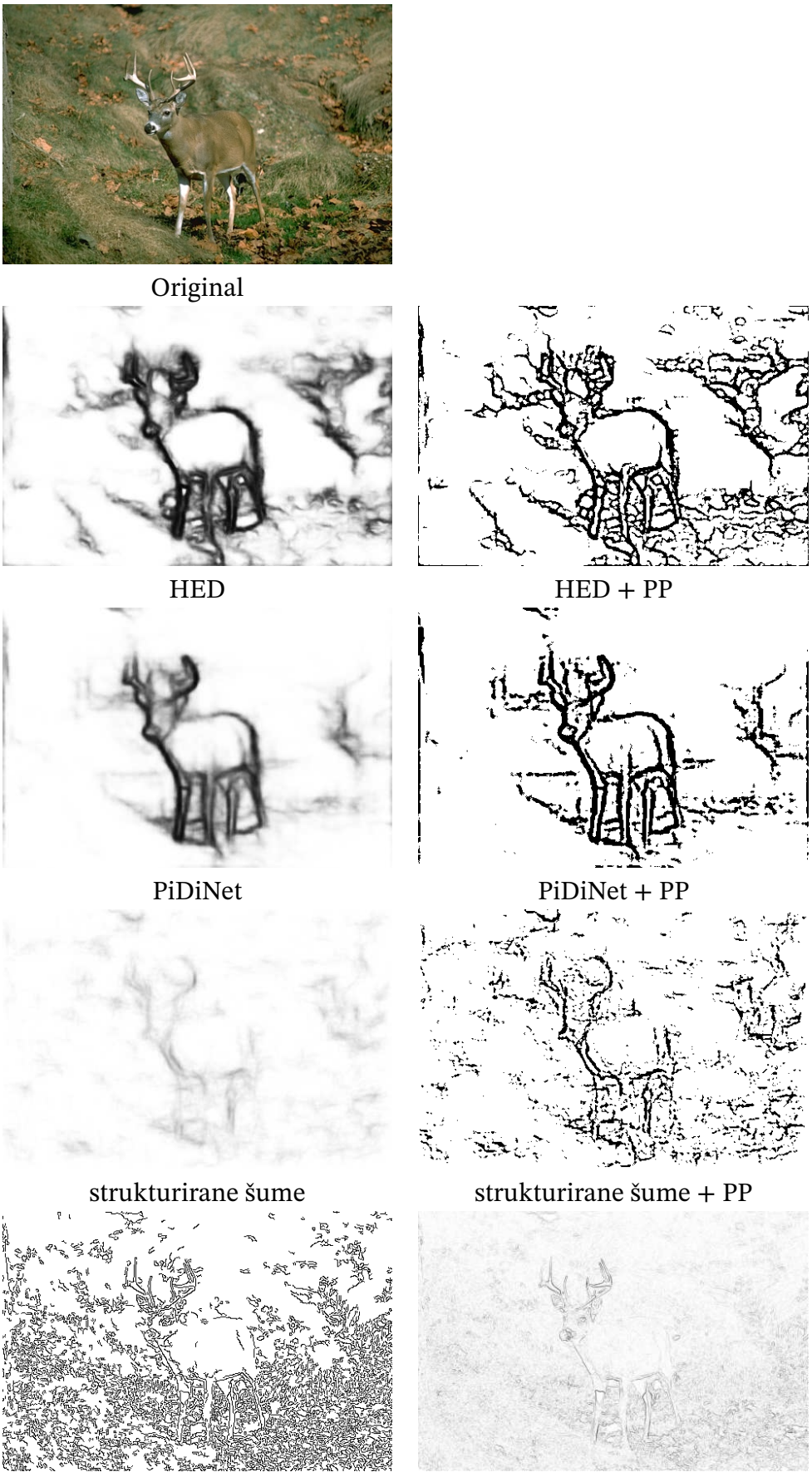
strukturirane šume + PP



Canny

Roberts

Slika 4.6. Usporedba tradicionalnih i modernih metoda na slici višeg kontrasta



Slika 4.7. Usporedba tradicionalnih i modernih metoda na slici niskog kontrasta

5. Implementacija web aplikacije

Ovo poglavlje opisuje arhitekturu, tehnologije i funkcionalnosti web aplikacije za detekciju rubova izrađene u svrhu testiranja alogiratama za ovaj rad.

5.1. Arhitektura aplikacije

Aplikacija se sastoji od dvije međusobno povezane komponente, poslužiteljske strane (engl. *backend*), realizirane pomoću radnog okvira Flask u programskom jeziku Python, te klijentske strane (engl. *frontend*), implementirane u Reactu i JavaScriptu. U središtu *backenda* nalazi se skripta `app.py` koja postavlja server Flask, omogućuje CORS i definira četiri glavne mape za pohranu podataka:

- `testImages/` za predefinirane testne uzorke
- `uploads/` za slike koje korisnik učita
- `processed/` za rezultate obrađenih slika razvrstanih po kategorijama
- `groundTruth/` za *ground-truth* oznake potrebne za izračun metrika

Logika detekcije rubova nalazi se u modulu `edge_detection_methods.py`, u kojem su implementirane tradicionalne metode (Canny, Sobel, Laplacian, Prewitt i Roberts), pristup temeljen na strojnom učenju (strukturirane šume), metode dubokog učenja (HED, PiDiNet) te segmentacijski algoritmi (Watershed i GrabCut), zajedno s odgovarajućom naknadnom obradom za HED, PiDiNet i strukturirane šume.

Evaluacija kvalitete rubova provodi se preko modula `metrics.py`, u kojem funkcija `calculate_all_metrics()` za svaku obrađenu sliku međusobno uspoređuje izlaze i *ground-truth* oznake kako bi vratila rezultate metrika.

Frontend aplikacije zasnovan je na biblioteci React, a njegova je glavna točka ulaska komponenta `App.jsx` koja upravlja stanjem i koordinira komunikaciju s *backendom*. Komponenta `UploadForm.jsx` omogućuje korisniku da učita vlastitu sliku ili pokrene testiranje na predefiniranim slikama pritiskom na odgovarajuće gumbe, dok komponenta `ImageDisplay.jsx` prikazuje originalne i obrađene slike te omogućuje korisniku podešavanje praga i pokretanje izračuna metrika pritiskom na gumb *Calculate Metrics*.

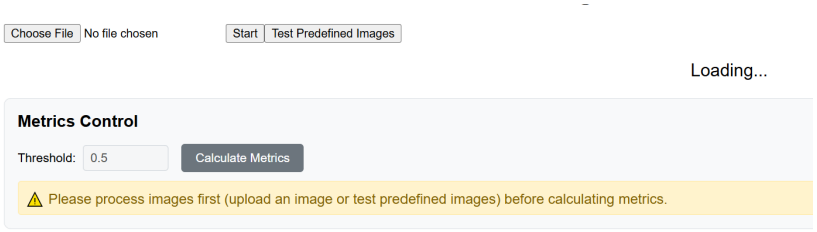
Svi podaci razmjenjuju se putem HTTP zahtjeva na rute `GET /test`, `POST /upload` i `POST /calculate_metrics`, pri čemu se JSON objektima prenose informacije o nazivima i putanjama do slika te parametrima poput vrijednosti praga.

5.1.1. Funkcionalnosti

Aplikacija omogućuje korisniku dvije osnovne interakcije odabira slika za procesiranje: pokretanje obrade skupine predefiniranih testnih slika te učitavanje i obradu vlastitih slika.

Predefinirane slike

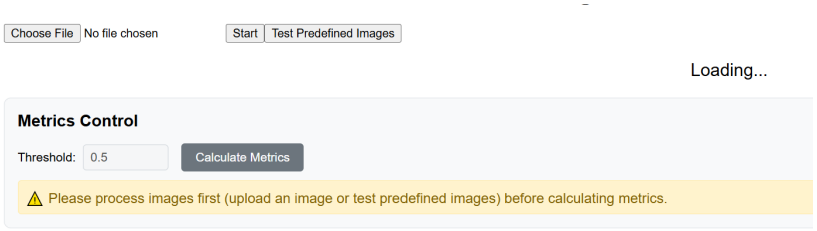
U prvom slučaju, korisnik pritiskom na gumb *Test Predefined Images* inicira slanje GET zahtjeva na rutu `/test`, nakon čega se na sučelju pojavljuje indikacija učitavanja („Loading...“) sve dok *backend* ne obradi sve slike.



Slika 5.1. Prikaz učitavanja obrade predefiniranih slika

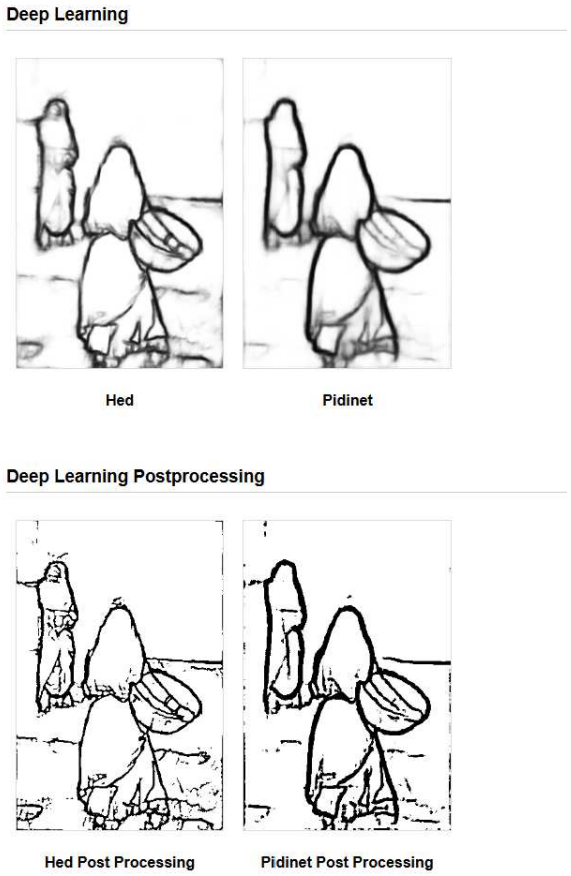
Druga funkcionalnost omogućuje korisniku da vlastitu sliku prenese preko obrasca u komponenti `UploadForm.jsx`. Nakon odabira datoteke i klika na gumb *Start*, aplikacija šalje POST zahtjev na rutu `/upload`, sprema datoteku u mapu `uploads/` te paralelno pokreće sve definirane metode detekcije rubova.

Kada su slike obrađene, komponenta `ImageDisplay.jsx` prikazuje originalnu i obra-



Slika 5.2. Prikaz učitavanja vlastite slike i klik na gumb Start

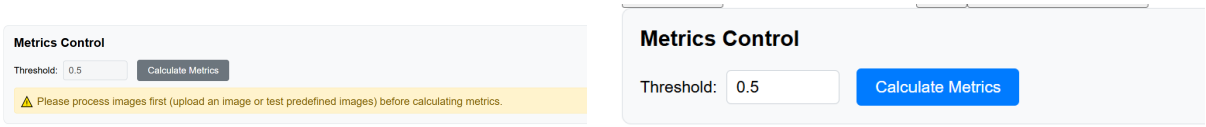
đenu sliku, organizirano po kategorijama (Traditional, Deep Learning, Segmentation itd.). Svaka podkategorija i naziv metode istaknuti su naslovom iznad slike.



Slika 5.3. Prikaz učitanih slika po kategorijama

Nadalje, aplikacija omogućuje izračun kvantitativnih metrika za procijenjenu kvalitetu detekcije rubova. U `ImageDisplay.jsx` nalazi se kontrola za prag (threshold) s početnom vrijednošću 0,5 te gumb *Calculate Metrics*.

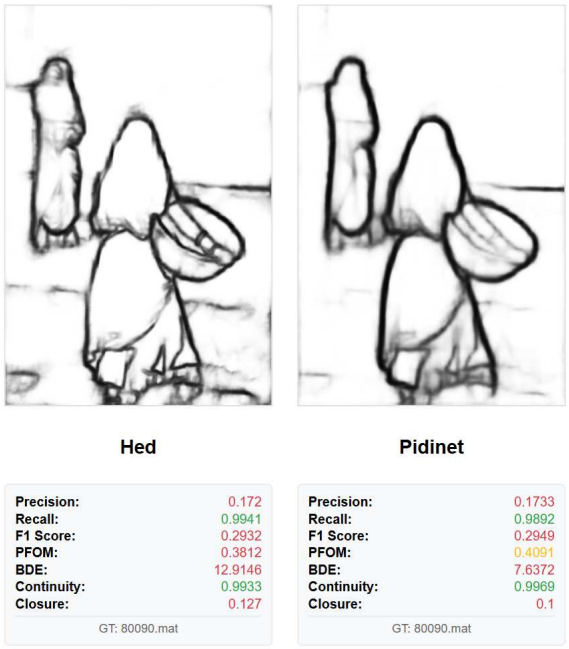
Nakon klika na gumb, šalje se POST zahtjev na `/calculate_metrics` i prikazuje se ponovno indikacija učitavanja. Kada *backend* vrati rezultate, ispod svake obrađene



Izračun metrika nije omogućen Izračun metrika je omogućen

Slika 5.4. Izračun metrika

slike automatski se ispisuju ključne mjere: preciznost, odziv, F1-mjera, PFOM, BDE, kontinuitet i zatvaranje petlji. Ako učitana slika nije iz testnoj skupa BSDS500, za nju neće biti moguće izračunavanje metrika.



Slika 5.5. Prikaz izračunatih metrika

6. Zaključak i budući rad

U ovom radu provedena je sveobuhvatna usporedba tradicionalnih operatora (Canny, Laplacian, Prewitt, Roberts, Sobel), metoda dubokog učenja (HED, PiDiNet), segmentacijskih tehnika (GrabCut, Watershed) te strojnog učenja (metoda strukturih šuma) na slikama iz BSDS500 skupa. Rezultati jasno pokazuju kako duboke metode, osobito PiDiNet s naknadnom obradom, nadmašuju tradicionalne i segmentacijske pristupe u smislu F1-mjere, PFOM-a i prostorne preciznosti (BDE), dok tradicionalni operatori i segmentacijske tehnike, služe kao vrijedna referentna točka za razumijevanje jačine i slabosti svake metode.

6.0.1. Poboljšanje rezultata

Prije primjene svih metoda preporučuje se primijeniti odgovarajuću predobradu slika, uključujući tehnike smanjenja šuma i izjednačavanja histograma, kako bi se poboljšala kontrastna razlika između objekata i pozadine. Za Watershed segmentaciju predlaže se ugradnja gradijentne magnitude kao ulazne karte umjesto binarne maske, korištenje lokalnog adaptivnog praga umjesto globalne OTSU-ove binarizacije, implementaciju hijerarhijskog pristupa na više skala radi bolje detekcije objekata različitih veličina te dodatno morfološko stanjivanje granica kako bi se dobile jednopikselske, kontinuirane linije.

U kontekstu GrabCut tehnike, korisniku bi trebalo omogućiti interaktivno označavanje područja objekta i pozadine, a sustav bi se mogao dodatno poboljšati automatskom inicijalizacijom. Povećanje broja iteracija uz postavku kriterija konvergencije poboljšalo bi prilagodbu GMM modela složenim scenama. Kombinacija GrabCut segmentacije s tradicionalnim detektorima rubova, primjerice Cannyjem, mogla bi rezultirati boljim rezultatom koji koristi prednosti oba pristupa.

6.0.2. Budući rad

Za budući rad preporučuje se istražiti ove i druge napredne pretprocesne korake, kao što su lokalno prilagođeno smanjenje šuma i korekcija osvjetljenja, te eksperimentirati s novim dubokim arhitekturama. Razvoj sustava koji simultano obavlja detekciju rubova i semantičku segmentaciju, evaluacija na raznolikim skupovima podataka te automatsko podešavanje hiperparametara optimizacijskim tehnikama poput Bayesovskog optimiziranja doprinijeli bi robusnijim i primjenjivijim rješenjima u stvarnim scenarijima.

Specifično za primjenu u stvaranju bojanki, potrebno je razviti dodatne tehnike koje bi osigurale optimalne rezultate za ovu domenu. Ključno je implementirati algoritme za automatsko zatvaranje prekidnih linija koji bi koristili kontekstualne informacije za prepoznavanje gdje bi se trebale spojiti bliske, ali nepovezane rubne komponente. Razvoj inteligentnog filtriranja šuma koji razlikuje između relevantnih detalja i nepotrebnih artefakata omogućio bi čišće bojanke.

Važno je također kreirati sustav za prilagodbu složenosti obzirom na ciljanu skupinu korisnika - jednostavnije linije za mlađu djecu, detaljnije za starije. Implementacija algoritma za grupiranje i organizaciju objekata na stranici pomogla bi u stvaranju estetski privlačnih kompozicija, dok bi automatska detekcija i označavanje različitih regija boja omogućila stvaranje vodiča za bojanje.

Literatura

[1] R. V. Ramana, T. V. Rathnam, i A. S. Reddy, "A review on edge detection algorithms in digital image processing applications", *International Journal on Recent and Innovation Trends in Computing and Communication*, sv. 5, br. 10, str. 69–75, listopad 2017.

[2] R. C. Gonzalez i R. E. Woods, *Digital Image Processing*, 3. izd. Upper Saddle River, NJ: Prentice Hall, 2008.

[3] I. Sobel i G. Feldman, "A 3x3 isotropic gradient operator for image processing", *Stanford Artificial Intelligence Project*, 1968., presented at a talk at the Stanford Artificial Intelligence Project.

[4] "Sobel and scharr derivatives", https://docs.opencv.org/4.x/d2/d2c/tutorial_sobel_derivatives.html, OpenCV, 2023., openCV dokumentacija za Sobel operator.

[5] J. M. S. Prewitt, "Object enhancement and extraction", *Picture processing and Psychopictorics*, sv. 10, br. 1, str. 15–19, 1970.

[6] "Edge operators", https://scikit-image.org/docs/stable/api/skimage.filters.html, Scikit-image, 2023., scikit-image dokumentacija za edge operatore.

[7] L. G. Roberts, "Machine perception of three-dimensional solids", doktorska disertacija, Massachusetts Institute of Technology, 1963.

[8] "Roberts cross-gradient operator", https://scikit-image.org/docs/stable/api/skimage.filters.html, Scikit-image, 2023., scikit-image dokumentacija za Roberts operator.

[9] D. Marr i E. Hildreth, “Theory of edge detection”, *Proceedings of the Royal Society of London. Series B. Biological Sciences*, sv. 207, br. 1167, str. 187–217, 1980.

[10] “Laplace operator”, https://docs.opencv.org/4.x/d5/db5/tutorial_laplace_operator.html, OpenCV, 2023., openCV dokumentacija za Laplacian operator.

[11] J. Canny, “A computational approach to edge detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, sv. 8, br. 6, str. 679–698, 1986.

[12] H. Agrawal, M. Gupta, A. Birodkar, i S. Panwar, “Canny edge detection: A comprehensive review”, *International Journal of Trend in Research and Development*, sv. 6, br. 2, str. 253–256, 2019., comprehensive review of Canny edge detection algorithm steps.

[13] X. Soria i D. Poma, “Towards a robust cnn model for edge detection”, u *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020., str. 215–224.

[14] S. Xie i Z. Tu, “Holistically-nested edge detection”, *International Journal of Computer Vision*, sv. 125, br. 1-3, str. 3–18, 2017.

[15] P. Arbelaez, M. Maire, C. Fowlkes, i J. Malik, “The berkeley segmentation dataset and benchmark”, https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/, 2011., službeni BSDS500 dataset za evaluaciju detekcije rubova.

[16] Z. Su, J. Zhang, W. Luo, K.-Y. K. Wong, i L. Zhang, “Pixel difference networks for efficient edge detection”, u *IEEE International Conference on Computer Vision (ICCV)*, 2021., str. 10 350–10 359.

[17] X. Soria i D. Poma, “Dense extreme inception network for edge detection”, *Pattern Recognition*, sv. 136, str. 109234, 2023.

[18] P. Dollar i C. L. Zitnick, “Fast edge detection using structured forests”, u *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013.

[19] “Image segmentation with watershed algorithm”, OpenCV, 2025., dostupno na: https://docs.opencv.org/4.x/d3/db4/tutorial$_p y_w atershed.html$.

[20] “Watershed segmentation - an overview”, ScienceDirect, dostupno na: https://www.sciencedirect.com/topics/computer-science/watershed-segmentation.

[21] “Watershed segmentation”, Scikit-image, dostupno na: https://scikit-image.org/docs/0.25.x/auto$_e$$xamples/segmentation/plot_watershed.html$.

[22] J. Angulo i D. Jeulin, “Stochastic watershed segmentation”, *Proceedings of International Symposium on Mathematical Morphology*, 2007.

[23] Q. Guo, L. Li, X. Li, Q. Bai, Z. Ma, Y. Li, i Y. Wang, “A method of blasted rock image segmentation based on improved watershed algorithm”, *Scientific Reports*, sv. 12, 2022.

[24] “Watershed segmentation algorithm in image processing”, Aegis Softtech, 2025., dostupno na: https://www.aegissofttech.com/articles/watershed-algorithm-and-limitations.html.

[25] “Image segmentation using grabcut”, Scaler, 2023., dostupno na: https://www.scaler.com/topics/image-segmentation-using-grabcut/.

[26] X. Zhang, Q. Li, i X. Xu, “Automatic grabcut color image segmentation based on em algorithm”, u *International Conference on Computer Science and Application Engineering*. Atlantis Press, 2018., dostupno na: https://www.atlantis-press.com/article/25839526.pdf.

[27] C. Göring, “Semantic segmentation using grabcut”, 2012., dostupno na: https://pub.inf-cv.uni-jena.de/pdf/goering2012semantic.pdf.

[28] S. Rakshit, “Nyu depth v2”, https://www.kaggle.com/datasets/soumikrakshit/nyu-depth-v2, 2024., accessed: September 2025.

[29] “BSDS500 dataset repository”, https://github.com/BIDS/BSDS500, BIDS - Berkeley Image Dataset and Benchmark, 2016., accessed: 2025-09-05.

[30] “Canny edge detection”, https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html, OpenCV, 2023., openCV dokumentacija za Canny edge detection.

[31] “Image gradients”, https://docs.opencv.org/4.x/d5/d0f/tutorial_py_gradients.html, OpenCV, 2023., openCV općenita dokumentacija za detekciju rubova.

[32] S. Xie, “Hed: Holistically-nested edge detection”, https://github.com/s9xie/hed, 2015., gitHub repository.

[33] licyk, “Pidinet pretrained model weights”, https://huggingface.co/licyk/controlnet_v1.1_annotator, 2023., huggingFace repozitorij s pretreniranim modelima.

[34] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikäinen, i L. Liu, “Pidinet: Pixel difference networks for efficient edge detection”, https://github.com/hellozhuo/pidinet, 2021., gitHub repozitorij s implementacijom.

[35] “Pretrained model for structured edge detection”, https://github.com/opencv/opencv_extra/blob/master/testdata/cv/ximgproc/model.yml.gz, OpenCV, 2023., pretreniran model za implementaciju strukturiranih šuma u OpenCV.

[36] “Structured forests for fast edge detection”, https://docs.opencv.org/4.x/d0/da5/tutorial_ximgproc_prediction.html, OpenCV, 2023., openCV dokumentacija za Structured Forests.

[37] D. Martin, C. Fowlkes, D. Tal, i J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics”, u *Proceedings IEEE International Conference on Computer Vision*, sv. 2, 2001., str. 416–423.

[38] W. K. Pratt, *Digital image processing*. John Wiley & Sons, 1978., referentna definicija Pratt’s Figure of Merit.

[39] C. Rother, V. Kolmogorov, i A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts”, u *ACM SIGGRAPH*, 2004., str. 309–314.

Sažetak

Detekcija rubova na slikama korištenjem tradicionalnih metoda i metoda dubokog učenja

Sara Tedeško

U ovom radu provedena je detaljna analiza i usporedba različitih tipova metoda za detekciju rubova i segmentaciju slika, uključujući tradicionalne operatore (Canny, Laplacian, Prewitt, Roberts, Sobel), metode strojnog učenja (Structured Forest), metode dubokog učenja (HED, PiDiNet) te segmentacijske tehnike (GrabCut, Watershed). Metode su evaluirane na skupu od 200 slika iz skupa podataka BSDS500 koristeći sedam metrika (preciznost, odziv, F1, PFOM, BDE, mjera kontinuiteta i mjera zatvaranja petlji). Rezultati pokazuju superiornost dubokih modela s naknadnom obradom, pri čemu PiDiNet + PP postiže najbolji balans između preciznosti, prostorne točnosti i povezanosti rubova. Tradicionalne i segmentacijske tehnike koriste se za referentne usporedbe, otkrivajući ograničenja u kompleksnim scenama. Rad također predlaže unapređenja segmentacijskih pristupa kroz gradijentni ulaz, hijerarhijsku obradu i morfološka naknadna obrada te interaktivnu i adaptivnu inicijalizaciju za GrabCut. Za budući rad preporučuje se istraživanje naprednih pretprocesiranja, eksperimentiranje s novim dubokim arhitekturama i razvoj end-to-end sustava za simultanu detekciju rubova i semantičku segmentaciju.

Ključne riječi: detekcija rubova; web aplikacija; evaluacijske metrike; duboko učenje; strojno učenje; naknadna obrada; BSDS500; Canny; HED; PiDiNet

Abstract

Edge Detection in images using traditional and deep learning methods

Sara Tedeško

In this thesis, a comprehensive analysis and comparison of different types of edge detection and image segmentation methods was conducted, including traditional operators (Canny, Laplacian, Prewitt, Roberts, Sobel), machine learning approaches (Structured Forest), deep learning techniques (HED, PiDiNet), and segmentation algorithms (GrabCut, Watershed). The methods were evaluated on 200 images from the BSDS500 dataset using seven metrics (precision, recall, F1, PFOM, BDE, Continuity Score, and Loop Closure Rate). The results demonstrate the superiority of deep models with postprocessing, with PiDiNet + PP achieving the best balance between precision, spatial accuracy, and edge continuity. Traditional and segmentation techniques serve as reference baselines, revealing their limitations in complex scenes. The thesis also proposes improvements to segmentation methods through gradient-based inputs, hierarchical processing, and morphological postprocessing, as well as interactive and adaptive initialization for GrabCut. Future work is recommended to explore advanced preprocessing techniques, experiment with new deep architectures, and develop end-to-end systems for simultaneous edge detection and semantic segmentation.

Keywords: edge detection; web application; evaluation metrics; deep learning; machine learning; postprocessing; BSDS500; Canny; HED; PiDiNet