Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Klemen Kozjek

Avtomatska segmentacija belih madežev na gladkih zobnih ploskvah

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: doc. dr. Luka Šajn

Ljubljana, 2018

Zahvala

Zahvaljujem se mentorju doc. dr. Luki Šajnu za idejo, pomoč, nasvete in usmerjanje pri izdelavi magistrskega dela. Hvala asist. dr. Tanji Tomaževič, dr. dent. med. in Poloni Ivanič, dr. dent. med. za priskrbljene slike. In hvala vsem, ki so kakorkoli pripomogli k uspešno izdelanemu migistrskemu delu, in vsem, ki so me med študijem podpirali.

Klemen Kozjek, 2018

Vsem rožicam tega sveta.

"The only reason for time is so that everything doesn't happen at once."

— Albert Einstein

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
FP	false positive	lažno pozitivni
TP	true positive	resnično pozitivni
FN	false negative	lažno negativni
TN	true negative	resnično negativni
MST	minimum spanning tree	najmanjše vpeto drevo
CLAHE	contrast limited adaptive	kontrastno omejeno
	histogram equalization	prilagodljiv razteg histograma
LAD	least absolute deviation	najmanjše absolutno odstopanje
JSEG	Jimage segmentation	segmentacija s sliko J
SOM	self-organizing map	samoorganizirajoča preslikava
RI	rand index	mera rand
VOI	variation of information	razdalja skupnih informacij
GCE	global consistency error	globalna napaka doslednosti
LCE	local consistency error	lokalna napaka doslednosti
MSE	mean squared error	napaka dvokratnega povprečja
BDE	boundary displacement error	napaka meje
CCD	charge-coupled device	nabojna naprava za sliko
LED	light-emitting diode	svetleča dioda
K-MEANS	k-means clustering	razvrščanje z voditelji
FCM	fuzzy c means	mehko razvrščanje
CQ-ABC	color quantization with	barvna kvantizacija
	artificial bee colony	z umetno kolonijo čebel
SVM	support vector machine	metoda podpornih vektorjev
EM	expectation-maximization	maksimizacija pričakovanja
MRF	markov random fields	naključna Markova polja
ANN	artificial neural network	umetna nevronska mreža
STAPLE	simultaneous truth and	sočasna ocena resničnosti
	performance level estimation	in zmogljivosti
GLCM	gray-level co-occurrence matrix	matrika sočasnih pojavitev sivin

Povzetek

Naslov: Avtomatska segmentacija belih madežev na gladkih zobnih ploskvah

Vse pogosteje se za medicinske namene uporablja računalniške tehnologije, saj te predstavljajo pomemben gradnik sodobnega zdravstva. Po snetju nesnemljivega ortodontskega aparata se pogosto pojavijo začetki demineralizacije, ki se kažejo kot bele ploskve na gladkih površinah zobne sklenine. V magistrskem delu razvijemo prototip sistema za avtomatsko segmentacijo zob in belih madežev, ki lahko pripomore k bolj natančnemu in objektivnemu spremljanju zdravljenja. Pri razvoju smo uporabili različne tehnike za obdelavo slik in algoritmov za segmentacijo. Razvit sistem smo ovrednotili na podatkovni zbirki, katero smo pripravili iz izbranih slik kliničnih pregledov. Rezultati so pokazali, da sistem deluje in ima še veliko prostora za nadaljnje delo ter izboljšave.

Ključne besede

računalniški vid, obdelava slik, beli madeži, segmentacija slik, rast regij, segmentacija z grafi

Abstract

Title: Automatic segmentation of white spot lesions on smooth tooth surfaces

Computer technologies are ubiquitous and in the recent times, these technologies are penetrating more into the field of medicine, where they play a vital role in modern healthcare. In this master's thesis, we are solving a problem, which dentists typically encounter during teeth alignment treatment. At the end of the treatment, when permanent orthodontic braces are removed, the initial phase of tooth demineralization often appears as white spot lesions on the smooth surfaces of a tooth. We developed a prototype for automatic segmentation of teeth and white spot lesions, which may contribute to a more accurate and objective way of treatment monitoring. In the process of development, we used various image processing techniques and image segmentation algorithms. The developed prototype was evaluated against a database, which we built from the selected images of clinical examinations. The prototype showed promising results with a lot of potential for improvements and future work.

Keywords

computer vision, image processing, white lesions, image segmentation, region growing, segmentation with graphs

Poglavje 1

Uvod

1.1 Motivacija

Računalniške tehnologije se vse pogosteje uporablja tudi za medicinske namene, saj predstavljajo pomemben gradnik sodobnega zdravstva [1]. Računalniška področja, ki se uporabljajo za analiziranje medicinskih podatkov, so predvsem: računalniški vid, procesiranje signalov in strojno učenje. Uporaba metod je odvisna od posameznega problema, formata in količine podatkov, ki jih imamo na voljo. Avtomatski sistemi, ki operirajo nad podatki, pridobljenimi s kliničnimi pregledi, nam omogočajo razbremenitev medicinskega osebja, saj lahko sistem olajša postopek zaznave anomalij v podatkih.

V delu bomo reševali problem, s katerim se srečujejo zobozdravniki. Po snetju nesnemljivega ortodontskega aparata se pogosto pojavijo začetki demineralizacije, ki se kažejo kot bele ploskve na gladkih površinah zobne sklenine (Slika 1.1) [2]. Po snetju aparata gredo pacienti skozi zdravljenje, pri katerem morajo zobozdravniki pri vsakem pregledu oceniti velikost belega madeža in spremljati spremembe.

Naš cilj je razviti prototip sistema, ki bo iz zajetih RGB slik, pridobljenih tekom zobozdravniškega pregleda, obdelal, segmentiral območja našega zanimanja in ocenil delež prizadete zobne sklenine. Tako želimo doseči bolj objektiven sistem za spremljanje velikosti belih madežev in tako prispevati k

Slika 1.1: Primer belih madežev po snetju ortodontskega aparata

razvoju sodobnega zdravstva.

1.2 Pregled področja

Segmentacija slik je pomemben del pri procesu obdelave in analize slik, saj lahko služi kot končna rešitev problema, ali pa kot vmesni korak, ki zmanjša sliko na manjše in bolj obvladljive kose [3]. Glavni namen segmentacije je razdeliti slikovne točke (ang. *pixels*) vhodne slike na manjše regije, ki imajo podobne lastnosti znotraj posameznih regij in različne lastnosti v primerjavi z drugimi.

Pri pregledu sorodnih del smo ugotovili, da ni veliko člankov, ki so neposredno povezani s problemom, katerega rešujemo v našem delu. Zato smo pregled razširili na segmentacijo medicinskih in barvnih slik. Osnovni pregled glede različnih tehnik obdelave in segmentacije slik smo črpali iz del [3, 4], saj nudita celovit pregled nad tematiko, podajata pa tudi primere uporabe ter prednosti in slabosti posameznih metod.

V članku [5] predstavijo učinkovito segmentacijo zob s prirejeno metodo *Watershed* in morfološkimi značilnostmi zob za segmentacijo zob. V tem članku segmentirajo celotno zobovje, posneto s sprednje strani. V koraku predobdelave analizirajo sliko v RGB barvnem prostoru, kjer ugotovijo, da se razlika v intenziteti med zobno sklenino in dlesnijo najbolj izrazi v kanalu rdeče barve. Tako na podlagi razlike RGB slike in komplementa kanala rdeče barve pridobijo regijo z zobno sklenino. Nato s pomočjo metode treh rezov (ang. *trisection*) in morfološkimi značilnostmi določijo potencialno območje zanimanja. Potencialno območje razdelijo na tri dele zaradi razlike v osvetlitvi, saj imajo tkiva, ki so pravokotna na vir svetlobe, drugačen kontrast kot tista ob straneh. S tem lokalizirajo dele z različno osvetlitvijo in posledično bistveno izboljšajo rezultat.

Članek [6] ni neposredno povezan s problemom, ki ga rešujemo, vendar uporabi veliko pristopov, ki smo jih uporabili tudi pri naši implementaciji. Najprej analizirajo razlike histogramov v barvnem prostoru HSV, kjer ugotovijo, kako se histogrami ozadja in objektov zanimanja razlikujejo. Pred segmentacijo slike aplicirajo filter povprečenja (ang. *mean filter*) velikosti 3 x 3, da zmanjšajo šum. Uporabijo kanal barve (ang. *Hue*) iz barvnega prostora HSV. Ker je kanal H definiran kot krog (od 0° do 360°), je priporočljivo, da vrednosti slike premaknemo za nekaj stopinj, če je naša ciljna regija barve na prelomnici med maksimalno in minimalno vrednostjo. Za segmentacijo so uporabili maksimalno entropijo (ang. *maximum entropy*) in prilagodljivo upragovljanje (ang. *adaptive thresholding*). Na koncu uporabijo še morfološke operatorje, da zapolnijo regije ter odstranijo manjše luknje.

Postopek v delu [7] ponovno uporabi metodo *Watershed*. Večji del članka je namenjen postopku predobdelave. Za segmentacijo so uporabili kanal svetlosti (ang. *Value channel*) iz barvnega prostora HSV. Nad izbranim kanalom aplicirajo korekcijo kontrasta CLAHE (ang. *Contrast Limited Adaptive Histogram Equalization*), ki deluje tako, da ne popravi kontrasta slikovnih točk na podlagi celotne slike, temveč za izračun popravka uporabi le okno sosednjih slikovnih točk. Z metodo korekcije kontrasta dosežejo, da se prikažejo

robovi, ki so bili pred tem manj vidni. Potem uporabijo Otsu upragovljanje in Sobelov filter, ki poudari prehode med kontrasti in tako predpripravi robove bazenov, ki se bodo zapolnili v postopku segmentacije z *Watershed* algoritmom.

V članku [8] uporabijo barvni prostor HSI, ker je lažje definirati razdaljo med sosednjimi barvami za razliko od barvnega prostora RGB. Za segmentacijo uporabljajo par komponent (I, H) barvnega prostora HSI ter jih posredujejo v algoritem za gručenje *k-median*, ki je različica algoritma *k-means*. Za mero razdalje uporabljajo LAD (ang. *Least Absolute Deviation*).

V delu [9] za segmentacijo uporabijo algoritem JSEG, ki deluje tako, da vhodno sliko zmanjšajo na minimalno število barv (ang. *color quantization*) ter vsako barvo predstavijo z razredom. Potem se z drsečim oknom sprehodijo čez sliko z dodeljenimi razredi in izračunajo sliko J, nad katero aplicirajo algoritem tipa rasti regij (ang. *region growing*).

V delu [10] predstavijo postopek segmentacije, temelječ na algoritmu *Mean-shift*, ki naj bi se izkazal za boljšega kot pred tem razvite metode. V članku predstavljen postopek uporabi algoritem *Mean-shift*, ki vrne regije, ki naj bi zadostovale pogojem. Nad pridobljenimi regijami aplicirajo še dodatne pogoje, ki zavrnejo napačno segmentirane regije. Ti pogoji so specifični za problem, ki ga rešujejo.

Ker podatki o barvi in lokaciji posamezne slikovne točke včasih niso dovolj, lahko pridobimo dodatne podatke, kot so na primer značilke teksture. V članku [11] predstavijo, kako so izboljšali postopek, ki temelji na algoritmu *Mean-shift*, s podatki o teksturi. Za pridobitev le-teh so z uporabo valčkov (ang. *wavelet*) sliko transformirali ter izračunali srednjo energijsko vrednost. Potem so te energijske vrednosti gručili v štiri razrede ter jih uporabili kot novo značilko za opis slikovnih točk.

Poleg nizkonivojskih tehnik za segmentiranje se uporabljajo tudi nevronske mreže. Primer takega dela je [12], v katerem uporabijo SOM (ang. *Self-organizing map*).

Ker smo v delu implementirali tudi orodje za evalvacijo sistemov za se-

gmentiranje z referenčnimi slikami, bomo predstavili tudi pregled sorodnih del na področju analize zmogljivosti takšnih sistemov.

Ena izmed podatkovnih zbirk za evalvacijo in primerjavo algoritmov za segmentacijo je podatkovna zbirka Berkeley, ki je glede na število citiranj precej razširjena [13]. Zbirka vsebuje naravne slike, ki so barvne in sivinske. V njihovem sistemu za primerjavo in rangiranje segmentacij različnih algoritmov uporabljajo vrednost F (ang. *F-score*) ter graf s komponentama natančnost (ang. *Precision*) na ordinatni osi in priklic (ang. *Recall*) na abscisni osi.

Druga takšna zbirka slik in orodij za evalvacijo je [14]. Ta zbirka, za razliko od Berkleyeve, ne vsebuje naravnih slik, temveč računalniško generirane mozaike. Za rangiranje algoritmov po zmogljivosti uporabljajo kar 27 različnih kriterijev, ki jih razdelijo v tematske kategorije: glede na regije, slikovne točke, mere konsistentnosti in kriterije za primerjavo gruč.

V članku [15] naredijo analizo zmogljivosti algoritmov za segmentacijo, ki temeljijo na principu gručenja. Za mere uporabijo statistične mere, kot so RI (ang. *Rand Index*), VOI (ang. *Variation of Information*), GCE (ang. *Global Consistency Error*) in BDE (ang. *Boundary Displacement Error*).

V članku [16] kot glavno mero zmogljivosti binarnega klasifikatorja oziroma segmentacije na ozadje in ospredje uporabljajo Jaccardov koeficient podobnosti oziroma Jaccardov indeks.

1.3 Prispevki

Magistrsko delo zajema implementacijo prototipa sistema, ki s pomočjo avtomatske segmentacije slik in drugih tehnik procesiranja slik segmentira regije zob in prizadete dele z belimi madeži na gladkih površinah zobne sklenine. Za potrebe implementacije, testiranja in evalvacije sistema potrebujemo primerno podatkovno zbirko slik, ki je na spletu nismo našli. Zato smo, v sodelovanju z medicinsko fakulteto, ustvarili lastno podatkovno zbirko. Pri ustvarjanju podatkovne zbirke smo ročno označili območja našega zanimanja,

ki nam služijo kot reference (ang. *ground truth*) za evalvacijo. Za namen evalvacije sistema smo razvili orodje, ki nam omogoča primerjavo regij med referenčnimi oznakami ter oznakami, pridobljenimi z zunanjimi sistemi. Orodje za primerjavo segmentacij uporablja različne metrike, ki se uporabljajo za analizo zmogljivosti takšnih algoritmov. Cilj implementiranega sistema je prototip orodja, ki bi ga lahko zobozdravniki uporabljali pri spremljanju velikosti prizadetih delov zobne sklenine tekom zdravljenja, saj objektivno in numerično poda dejstva o stanju.

1.4 Pregled magistrskega dela

Magistrsko delo je razdeljeno na šest poglavij oziroma vsebinskih sklopov. Vsako poglavje vsebuje podpoglavja, ki vsebino razdelijo na manjše celovite vsebinske dele in jih natančneje predstavijo.

Prvo poglavje je uvod, kjer se predstavi motivacija za delo, pregled področja, prispevki ter struktura magistrskega dela.

V drugem poglavju je predstavljeno zajemanje in priprava podatkovne zbirke, ki smo jo ustvarili pri razvoju za namen testiranja in evalvacije končnega sistema. Poglavje predstavi protokol zajemanja slik, ročno označevanje območji našega zanimanja in lastnosti podatkovne zbirke.

Tretje poglavje predstavi teoretično ozadje področij računalniškega vida, uporabljenih v implementaciji sistema za lažje razumevanje nadaljnjih poglavij. V tem poglavju predstavimo različne pristope segmentacije, tehnike predobdelave ter mere za oceno in analizo zmogljivosti algoritmov za segmentiranje.

Četrto poglavje predstavi implementacijo sistema, ki smo ga razvili tekom magistrskega dela. Najprej predstavimo korake celotnega sistema, od začetka do konca. Nato pa v vsakem podpoglavju natančneje predstavimo posamezen korak končnega sistema.

V petem poglavju predstavimo rezultate sistema.

Šesto poglavje je zaključek, kjer predstavimo končne ugotovitve ter mo-

žnosti za izboljšave in nadaljnje delo.

Poglavje 2

Zajem in priprava podatkov

V tem poglavju je predstavljen proces zajemanja in priprave podatkov za grajenje podatkovne zbirke. Za potrebe magistrskega dela smo potrebovali slike zob s prisotnimi belimi madeži, ki so lahko prisotni po snetju nesnemljivega ortodontskega aparata.

Primerne podatkovne baze, ki bi bila primerna za reševanje našega problema, nismo našli prosto dostopne na spletu. Zato smo v sodelovanju z medicinsko fakulteto zgradili lastno podatkovno zbirko, primerno za potrebe testiranja in evalvacije razvitega sistema.

2.1 Zajem podatkov

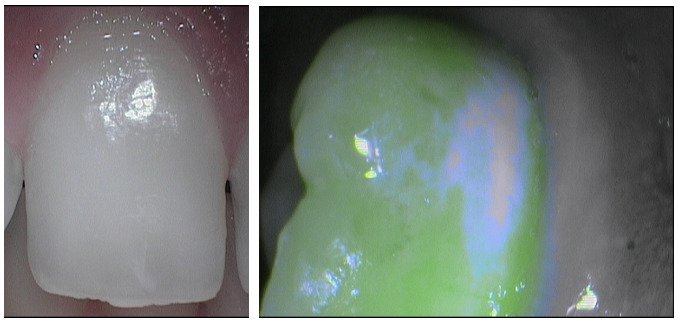
Za zajem slik smo uporabili intraoralno kamero Soprolife (Slika 2.1). Uporablja se kot pripomoček za diagnozo prevalence demineralizacije zobne sklenine. K orodju je priložen tudi programski paket – Sopro [17], ki omogoča vodenje evidence pregledov pacientov in dodajanja ter urejanje meta podatkov zajetih slik. Tako smo med kliničnimi pregledi pacientov zajeli slike zob, ki so imeli prisotne bele madeže. Nato smo slike skupaj z meta podatki, ki so bili anonimizirani, izvozili ter uporabili za gradnjo podatkovne zbirke. Meta podatki, ki se izvozijo poleg slik, so podatki o lokaciji zob, datumu nastanka slike in še nekaj ostalih podatkov. Pri zajemanju slik smo se dogovorili, da

naj bo zob, ki je predmet pregleda, vedno v središču slike. Tako ima vsaka zajeta slika v središču zob, ki ga želimo analizirati, skupaj z delom dlesni in manjšim delom sosednjih zob. S tem smo si malo poenostavili kasnejše obdelovanje slik, saj nas zanima le zob z največjo površino.

Slika 2.1: Naprava za zajemanje slik – SOPROLIFE

Vir: https://www.gerl-shop.de

Orodje ima nameščeno kamero s CCD senzorjem in 8 LED diod, od tega 4 bele in 4 modre. Tako lahko zajame sliko zob, ki so osvetljeni z belo svetlobo (običajna barvna slika) ali pa z modro svetlobo valovne dolžine $450nm$ [18]. Demineralizirani deli se ob osvetljevanju s svetlobo valovne dolžine $450nm$ obarvajo svetleje (bele barve), nedemineralizirani deli sklenine pa se obarvajo z odtenki zelene barve (Slika 2.2). Kamera zajema slike z ločljivostjo 640 x 480.



Slika 2.2: Levo: primer slike, zajete z belo svetlobo; Desno: primer slike, zajete z modro svetlobo

2.2 Priprava podatkov

V procesu kliničnih pregledov smo pregledali 10 pacientov, ki so opravili en, dva ali tri preglede. Tako smo zbrali približno 150 slik. Vse slike smo ročno pregledali ter izbrali tiste, ki so bile dovolj kakovostne za nadaljnjo obdelavo. Zajem slik dobre kakovosti s Soprolife predstavlja izziv, saj je potrebno zelo natančno nastaviti položaj kamere, roka zobozdravnika mora biti mirna, prav tako pacientova ustna votlina. Zato smo med ročnim pregledovanjem veliko slik zavrnili, saj so bile preveč zamegljene in tako neuporabne.

Izbrali smo 30 slik, ki so se zdele najbolj primerne ter jih ročno segmentirali. Za ročno segmentacijo smo uporabili orodje Gimp [19], s katerim smo za vsako sliko ustvarili še dodatni dve črno-beli sliki (Slika 2.3). Prva slika predstavlja masko zoba, ki je predmet zanimanja. Druga slika predstavlja masko belih madežev na zobu našega zanimanja. To smo storili za vseh 30 slik.

Slika 2.3: Levo: originalna slika; Center: segment zoba; Desno: segment belih madežev

Poglavje 3

Teoretično ozadje

V tem poglavju predstavimo teoretično ozadje računalniškega vida, ki se neposredno navezuje na naše delo v magistrskem delu. Ker se delo navezuje predvsem na segmentacijo slik, je to poglavje namenjeno predvsem splošnemu pregledu segmentacij slik in temam, ki so tesno povezane. V prvem podpoglavju predstavimo različne barvne prostore, ki se pogosto uporabljajo za namen segmentacije slik. V drugem podpoglavju se osredotočimo na predobdelavo. V tretjem podpoglavju predstavimo pregled različnih pristopov, ki se uporabljajo za segmentacijo slik. V četrtem predstavimo mere za oceno algoritmov za segmentacijo.

3.1 Barvni prostori

Barvni prostori so razpon in organizacija barv, ki jih lahko neka naprava uporabi ali prikaže. Obstaja zelo veliko barvnih prostorov, vsak pa je neposredno povezan s svojim namenom uporabe. V tem delu bomo predstavili samo barvne prostore, ki so povezani z našim delom.

3.1.1 RGB

RGB je barvni prostor, ki ga danes uporablja večina naprav za zajemanje in prikazovanje slik. Vsak kanal predstavlja odziv senzorja na določen filter. V

tem barvnem prostoru so tudi vse slike, ki smo jih zajeli za potrebe našega dela. Slabost RGB barvnega prostora je, da se barv med seboj ne more enostavno primerjati (npr. z evklidsko razdaljo), saj barve, ki so si numerično blizu, ne odražajo vedno podobnosti kot jih vidi človeško oko. Zato so se razvili barvni prostori, ki bolj približajo organizacijo barv, kot jih dojema človeško oko. Primera takšnih barvnih prostorov sta CIELUV in CIELAB [20, 21].

3.1.2 HSV

HSV je barvni prostor, ki služi kot alternativa barvnemu prostoru RGB. Njegova prednost je, da barve loči v komponente barv (ang. *Hue*), nasičenost (ang. *Saturation*) in svetlost (ang. *Value*). Zaradi tega se ta barvni prostor zelo pogosto uporablja v računalniškem vidu. Omogoči nam manipulacijo slikovnih točk iste barve, ki nimajo vedno enake svetlosti. V primeru barvnega prostora RGB se ob spremembi osvetlitve spremenijo vse tri komponente, v primeru barvnega prostora HSV pa se bo največja razlika opazila le v enem kanalu. Za namen preslikave med barvnim prostorom RGB in HSV smo uporabili funkcijo *cvtColor* v knjižnici OpenCV.

3.1.3 CIELAB

CIELAB je barvni prostor, ki je bil razvit z namenom, da se čim bolj približa dojemanju razdalj med barvami, kot jih dojema človeško oko. Komponenta L predstavlja svetlost (višja kot je vrednost, svetlejša je barva). Komponenta a predstavlja zeleno-rdečo barvo, komponenta b pa modro-zeleno. Ker so slike običajno v RGB barvnem prostoru, je potrebno uporabiti preslikavo, če želimo uporabljati CIELAB barvni prostor. Za namen preslikovanja med barvnimi prostori smo uporabili funkcijo *cvtColor* v knjižnici OpenCV.

3.2 Metode za predobdelavo

V procesu segmentacije slik ima zelo velik vpliv proces predobdelave. V nekaterih primerih predstavlja tud najpomembnejši korak. V procesu predobdelave lahko uporabimo širok nabor metod, ki običajno bistveno izboljšajo rezultate in pripomorejo k stabilnejšim metodam in skladnejšim rezultatom. V nadaljevanju predstavimo nekaj uporabljenih metod predobdelave.

3.2.1 Filtri

Filtriranje je široko področje računalniškega vida, ki zajema kar nekaj različnih področij. Uporabimo ga lahko za odstranjevanje šuma (glajenje), iskanje robov, detekcijo vzorcev, abstrakcijo značilnosti (ojačanje ali zadušitev) ...

Proces filtriranja zajema jedro (ang. *kernel*) ter vhodni signal, ki v našem primeru predstavlja 2D digitalno sliko z enim ali več barvnimi kanali. Sliko si lahko predstavljamo tudi kot nabor 2D matrik z diskretnimi vrednostmi. Vsaka matrika predstavlja svoj barvni kanal in vsaka vrednost matrike predstavlja nivo intenzitete na določeni lokaciji slikovne točke v barvnem kanalu. Jedro pa si lahko predstavljamo kot 2D matriko, običajno simetrične oblike in lihe dimenzije.

Filtre za filtriranje slik lahko razdelimo v dve kategoriji, nelinearne in linearne. Primer nelinearnega filtra je filter mediane (ang. *median filter*). Deluje tako, da se čez sliko zapelje drseče okno določene velikosti in na vsakem mestu slikovne točke nastavi mediano vrednosti znotraj drsečega okna. Filter je nelinearen, ker izhod filtra ni sestavljen iz linearne kombinacije vhodnih vrednosti (vrednosti znotraj drsečega okna).

Primer linearnega filtra je filter povprečenja (ang. *mean filter*), ki na vsakem mestu slikovne točke nastavi povprečno vrednost vrednosti znotraj drsečega okna. Za apliciranje linearnih filtrov se uporablja konvolucija (ang. *convolution*). Proces konvolucije si lahko predstavljamo kot uteženo vsoto intenzitet sosednjih slikovnih točk. Enačba 3.1 definira konvolucijo za primer 2D jedra nad 2D sliko, kjer je f funkcija slike, ki vrne intenziteto, h predsta-

vlja jedro, ki je preslikano v vertikalni in horizontalni smeri, k pa predstavlja dimenzijo jedra.

$$g(x,y) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) f(x-u, y-v) \tag{3.1}$$

V delu smo uporabili filtriranje za odstranjevanje šuma v slikah in iskanje intenzitetnih prehodov. Obstaja več vrst šumov. Najpogostejša sta šum *sol in poper* (ang. *salt and pepper*) in šum Gaussa. Za odstranjevanje šuma sol in poper se običajno uporabi filter mediane ali morfološki filter. Šum Gaussa pa odstranimo s filtrom povprečenja (ang. *mean filter*) ali filtrom Gaussa (ang. *Gaussian filter*).

3.2.2 Morfološki operatorji

Morfološki operatorji so zbirka enostavnih operacij, ki se izvajajo nad binarnimi slikami. Za izvedbo morfološkega operatorja potrebujemo poleg slike še strukturni element oziroma jedro (ang. *kernel*). Jedro je lahko različne velikosti in oblike. Običajno se uporablja jedro v obliki pravokotnika, križa, elipse in kroga. Dve najbolj osnovni operaciji morfološkega operatorja sta erozija (ang. *erosion*) in razširitev (ang. *dilatation*). Iz osnovnih dveh operacij lahko zgradimo kompleksnejše operacije, kot so: odpiranje (ang. *opening*), zapiranje (ang. *closing*), morfološki gradient (ang. *morphological gradient*), top hat in black hat. Z velikostjo jedra ter obliko lahko nastavljamo rezultat same operacije. Za potrebe razlage v tem delu bomo uporabili vrednost 1 kot ospredje in vrednost 0 kot ozadje.

Morfološki operatorji delujejo tako, da se z jedrom sprehodimo čez sliko. V vsakem koraku se premaknemo za eno slikovno točko v eno smer, enako kot deluje 2D konvolucija. Uporablja se jih večinoma v postopku predobdelave slik za namen čiščenja.

Erozija in razširitev

Erozija je ena izmed dveh osnovnih morfoloških operacij. Kot pove že samo ime, deluje podobno kot erozija prsti v naravi. Z njo se znebimo manjših delov slike in razdelimo večjo regijo v manjše, v primeru, da so povezane z ozkimi povezavami. Ta operacija zmanjša površine regij.

Erozija je definirana z enačbo 3.2. A predstavlja regijo ospredja, B pa strukturni element. Vrednosti izhodne slike se nastavijo na 1 samo v primeru, ko so vse slikovne točke, prekrite z jedrom, enake 1.

$$A \bigominus B = \{x : B_x \subseteq A\} \tag{3.2}$$

Razširitev je druga osnovna operacija. Deluje ravno nasprotno kot operacija erozije.

Razširitev je definirana z enačbo 3.3. A predstavlja regijo ospredja, B pa strukturni element. Vrednosti izhodne slike se nastavijo na 1, če je vsaj ena slikovna točka prekrita z jedrom enaka 1.

$$A \bigoplus B = \{x : B_x \cap A \neq 0\} \tag{3.3}$$

Odpiranje in zapiranje

Odpiranje je ena izmed operacij, ki je sestavljena iz osnovnih dveh. Odpiranje se uporablja za odstranjevanje šuma v slikah. Najprej nad vhodno sliko izvedemo erozijo, ki povzroči, da se odstranijo manjše regije, ki predstavljajo šum. Zaradi erozije se regije, ki jih želimo obdržati, tanjšajo. Da bi ohranili velikosti večjih regij, nad rezultatom erozije apliciramo še razširitev. Operacija je idempotentna tako, da večkratno apliciranje nad podatki ne spremeni končnega rezultata. Odpiranje je definirano z enačbo 3.4.

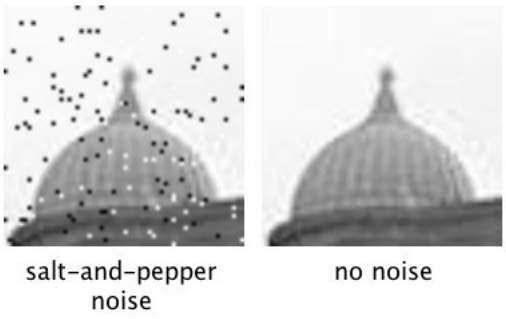
$$A \circ B = (A \bigominus B) \bigoplus B \tag{3.4}$$

Naslednja sestavljena operacija je zapiranje. Deluje nasprotno kot odpiranje. Namesto, da odstrani manjše regije, zapolni luknje v regijah. Tudi

ta operacija ohrani velikost regij in je idempotentna. Zapiranje nad vhodno sliko najprej izvede razširitev, ki povzroči, da se luknje v regijah zapolnijo in regije, ki so si med seboj dovolj blizu, zlijejo v eno. Regije se zaradi razširitve povečajo. Da bi ohranili velikost regij, nad rezultatom razširitve apliciramo še erozijo. Zapiranje je definirano z enačbo 3.5.

$$A \bullet B = (A \bigoplus B) \bigominus B \tag{3.5}$$

S kombinacijo odpiranja in zapiranja lahko sestavimo novo operacijo, ki se običajno uporablja za odstranjevanje šuma vrste *sol in poper*.



Slika 3.1: Levo: Primer šuma sol in poper; Desno: odstranjen šum

Vir: https://craftofcoding.wordpress.com

Morfološki gradient

Morfološki gradient je razlika razširitve ter erozije. Razširitev poveča površine regij, erozija jih zmanjša, rezultat je slika robov regij. Morfološki gradient je definiran z enačbo 3.6.

$$\nabla A = (A \bigoplus B) - (A \bigominus B) \tag{3.6}$$

3.2.3 Redukcija števila barv

Proces redukcije števila barv (ang. *color quantization*) je proces, s katerim želimo zmanjšati velikost barvne palete, vendar ohraniti podobnosti z izvirno sliko. Za reševanje tega problema se uporabljajo metode dveh vrst: metode gručenja (ang. *clustering methods*) in metode deljenja (ang. *splitting methods*) [22]. Redukcija števila barv se uporablja za kompresijo slik pri prenosu preko medijev, saj je človeško oko manj občutljivo na barve kot svetlost. V našem primeru smo jo uporabili za zmanjšanje iskalnega prostora.

Pri metodah deljenja se uporabi celoten barvni prostor, ki ga lahko razdelimo na enake ali različne velikosti podprostorov. Primeri metod [22]: razdelitev v enakomerno velike podprostore, rekurzivno deljenje glede na kriterij, metoda srednjega reza (ang. *median cut method*), metoda minimalne variance (ang. *minimum variance method*) ... Te metode običajno delujejo hitreje kot metode gručenja, vendar običajno ne podajo optimalne rešitve.

Metode gručenja so po navadi počasnejše, vendar proizvedejo boljše rezultate. Velik vpliv na sam rezultat ima proces inicializacije začetnih središč gruč. Primeri metod [22]: k-means, FCM, CQ-ABC ...

3.2.4 Korekcija kontrasta

Digitalne slike lahko poleg izrisa predstavimo tudi s histogrami, ki predstavljajo porazdelitev vrednosti slikovnih točk v sliki, v posameznem kanalu barvnega prostora. Zaradi različnih nastavitev naprave za zajemanje slik in pogojev osvetlitve, v katerih je bila slika zajeta, dobimo različne oblike histogramov. V primeru, ko kumulativna vsota histograma ne narašča linearno oziroma je velik del vrednosti zgoščen le v enem delu histograma, dobimo sliko z nizkim kontrastom. Problem lahko rešimo z raztegom histograma, da zavzame celotni razpon barvnega prostora in posledično poveča kontrast med barvami. Pri samem raztegu histograma je potrebno upoštevati, da bodo slike z nizko barvno globino postale popačene. Pri barvnih slikah se je potrebno odločiti, v katerem barvnem prostoru in nad katerimi kanali želimo

aplicirati sam razteg, saj to vpliva na rezultate izhodne slike. V našem delu smo poizkusili dve rešitvi za razteg histograma: razteg histograma (ang. *histogram equalization*) in CLAHE (ang. *Contrast Limited Adaptive Histogram Equalization*).

Prva metoda deluje globalno in naredi razteg histograma celotne slike. Ta metoda deluje dobro v primerih, ko je histogram vhodne slike zgoščen le v enem delu. V primerih, ko imamo več zgoščenih delov, bo ta metoda odpovedala. Zato lahko v takšnih primerih uporabimo metodo CLAHE, ki za razteg uporabi le manjši del celotne slike. Ta metoda razdeli celotno sliko na enako velike manjše kose, kjer se na vsakega aplicira prva metoda. Vsak kos celotne slike obdelamo posebej in jih na koncu združimo. Pri združevanju kosov v celotno sliko se uporabi bilinearna interpolacija, da se izognemo ostrim prehodom na robovih med posameznimi kosi [23]. V našem delu smo uporabili implementacijo metod CLAHE iz knjižnice OpenCV.

3.3 Segmentacija

Segmentacija digitalnih slik je pomemben in kritičen del računalniškega vida, analize slik, prepoznavanja vzorcev in robotike. Je eno izmed bolj zahtevnih opravil v procesiranju slik, saj posredno vpliva na končni rezultat algoritmov [3, 4]. Z vidika računalniškega vida je segmentacija proces razdelitve slik na več homogenih regij oziroma segmentov, ki se med seboj ne morejo združiti v večjo homogeno regijo. Splošno sprejeta definicija segmentacije je: če $P(\circ)$ definira homogenost slikovnih točk znotraj regije (enačba 3.7), potem je popolna segmentacija razdelitev slike I v končno število povezanih regij $\{C_1, ..., C_n\}$. Kjer $I = \bigcup_{i=1}^{n} C_i$ in $C_i \cap C_j = 0, \forall i \neq j$.

$$\begin{aligned} P(C_i) &= true \\ P(C_i \cup C_j) &= false \end{aligned} \tag{3.7}$$

V procesu segmentacije se vsaki slikovni točki določi razred, v katerega pripada. Slikovne točke z istimi razredi tvorijo segment, ki prekriva del slike.

Končni rezultat segmentacije je skupek segmentov, ki pokrivajo celotno sliko. Lastnosti slikovnih točk znotraj regije so si enake ali podobne, lastnosti slikovnih točk med različnimi regijami pa so si različne. Značilke, ki se uporabljajo pri segmentaciji, so barva, tekstura, robovi ... Nameni segmentacije slike so različni. V nekaterih primerih predstavljajo končno rešitev, v ostalih primerih se uporablja kot metoda predobdelave za zmanjšanje območij zanimanja v nadaljnjem procesu algoritma.

Pomembno je tudi vedeti, da je rešitev segmentacije odvisna od posameznikovega dojemanja, katere slikovne točke spadajo v skupno regijo in katere ne. Prav zaradi tega ne obstaja vedno analitična rešitev problema. To je verjetno tudi razlog, zakaj je v literaturi ogromno različnih rešitev. Veliko rešitev je razvitih po principu *ad hoc* za specifičen domenski problem, ki ga rešujejo [3].

Metode za segmentacijo se delijo v več skupin glede na način delovanja. V različnih literaturah delijo metode v različne skupine. Na primer v literaturi [3] delijo metode v tri skupine: pragovne metode (ang. *thresholding methods*), metode, ki temeljijo na robovih (ang. *edge-based*) in metode, ki temeljijo na regijah (ang. *region-based*). V literaturi [4] delijo metode v osem skupin:

1. pragovne metode (ang. *thresholding methods*)
2. metode regij (ang. *region-based*)
3. klasifikatorji (ang. *classifiers*)
4. gručenje (ang. *clustering*)
5. Markova naključna polja (ang. *Markov random field models*)
6. umetne nevronske mreže (ang. *artificial neural networks*)
7. deformabilni modeli (ang. *deformable models*)
8. predloge (ang. *atlas-guided approaches*)

3.3.1 Pragovne metode

Te metode spadajo med najbolj preproste metode, saj so računsko nezahtevne in zelo hitre. Pri pragovnih metodah lahko uporabimo globalne ali lokalne pragove. V praksi je globalen prag v večini primerov manj uspešen pristop. S pragovnimi metodami sliko razdelimo na ozadje in ospredje glede na vrednost intenzitete slikovnih točk in izbrani prag. Kot rezultat dobimo binarno sliko. Primer uporabe pragovne metode je segmentacija zdravih in rakavih tkiv na digitalnih slikah mamografije. Slabosti teh metod so, da ne upoštevajo nobenih drugih lastnosti, kot so robovi, teksture, prostorska povezanost slikovnih točk, in delujejo le na enokanalnih slikah (sivinske slike) [4]. Zelo so občutljive na šum, ki je lahko prisoten v slikah. V praksi se večinoma uporablja izpeljanke pragovnih metod in kot korak kompleksnejših algoritmov.

3.3.2 Metode regij

Metode regij temeljijo na širjenju povezanih delov slikovnih točk v regije glede na definiran kriterij. Lastnosti slike, ki se uporabljajo v povezavi z metodami regij, so barve/intenzitete in robovi. Slabosti osnovnih metod tega tipa so, da potrebujejo človeško interakcijo, saj za delovanje potrebujejo začetno točko (ang. *seed point*), od kjer se širjenje algoritma začne za vsako območje, ki ga želimo izluščiti. Zato se redko uporabljajo samostojno – po navadi so vmesni korak kompleksnejših algoritmov. Avtomatski algoritmi te vrste, ki ne potrebujejo človeške interakcije in postavljanje začetnih točk za vsako regijo zanimanja, so *Split-and-merge*. Lahko so občutljive na šum, kar privede do lukenj v segmentih rezultatov in po možnosti tudi do nepovezanih segmentov. Velikokrat so prisotne pri segmentaciji medicinskih slik, skupaj z algoritmom *watershed*.

3.3.3 Klasifikatorji

Klasifikatorji so metode, ki temeljijo na iskanju vzorcev v podatkih. Poskušajo razdeliti slikovne točke glede na njihove značilnosti (ang. *features*) in njihove dodeljene razrede. So nadzorovane metode (ang. *supervised methods*), saj za delovanje potrebujejo učno množico, ki ima ročno ali s pomočjo drugega algoritma že generirane segmente, ki se uporabijo kot referenca za učenje uteži klasifikatorja. Kasneje pa se uporabijo pri avtomatski segmentaciji novih slik. Primeri takšnih klasifikatorjev so kNN (ang. *k-nearest-neighbor*), Bayesov klasifikator, SVM (ang. *Support vector machine*). Za dobro delovanje segmentiranja s klasifikatorji potrebujemo dobro izbran prostor značilk (ang. *feature space*), ki dobro loči vsak razred v učni množici. Za razliko od pragovnih metod se klasifikatorje lahko aplicira na več kanalne slike. Segmentacija s klasifikatorji ni iterativna, je računsko manj zahtevna operacija. Slabost klasifikatorjev je, da potrebujemo veliko časa za pripravo učne množice in označevanje referenčnih segmentacij. Prav tako ne upoštevajo prostorske povezanosti med slikovnimi točkami, čeprav naprednejše variacije teh metod to že znajo upoštevati. Za dobro delovanje klasifikacijskih metod potrebujemo različne in številne primere, saj se v nasprotnem primeru pojavu prenasičenost (ang. *overfitting*).

3.3.4 Gručenje

Gručenje nekako predstavlja najbolj intuitiven način segmentacije slike, vsaj s stališča človeškega razumevanja problema segmentacije. Je zelo podobno klasifikatorjem, le da ne potrebuje učne množice. Zato spada pod nenadzorovane metode (ang. *unsupervised methods*). Ker ne uporablja učne množice, je implementirano iterativno in ponavlja isti korak, dokler rezultat konvergira glede na mero, ki se jo optimizira. Najpogosteje uporabljeni algoritmi za gručenje za namen segmentacije so k-means, FCM in EM (ang. *expectation-maximization*) [4]. Čeprav ne potrebuje človeške interakcije ali učne množice, potrebuje začetno stanje segmentacije oziroma začetne točke in število se-

gmentov, ki jih želimo imeti v končnem rezultatu. V standardni implementaciji ne upošteva prostorske povezanosti slikovnih točk in je občutljivo na šum.

3.3.5 Markova naključna polja

Markova naključna polja ali MRF (ang. *Markov random field*) ni metoda, ki bi se neposredno uporabljala za segmentacijo slik. Uporablja se kot dodatno orodje za modeliranje lastnosti povezanosti sosednjih slikovnih točk glede na statistični model. MRF se uporablja skupaj z metodami za gručenje. Uporaba MRF je prisotna predvsem v segmentaciji slik magnetne resonance, na primer slik mamografije [4].

3.3.6 Umetne nevronske mreže

Umetne nevronske mreže ali ANN (ang. *Artificial Neural Networks*) so paralelna mreža povezanih vozlišč, ki jemljejo navdih iz delovanja možganov in simulirajo njihovo učenje. ANN običajno vsebuje več nivojev vozlišč, ki se med seboj povezujejo. Vsaka povezava je predstavljena s težo, ki se v času učenja prilagaja glede na vhodne učne podatke in pričakovane rezultate vrednosti. Vsako vozlišče opravlja neko osnovno operacijo. ANN je pristop, ki izvira iz strojnega učenja. Uporaba ANN z medicinskimi slikami se največkrat uporablja kot klasifikator, čeprav jo lahko uporabimo tudi kot metodo gručenja [4].

3.3.7 Deformabilni modeli

Deformabilni modeli je metoda, ki temelji na modelu razmejevanja regij s pomočjo sklenjene parametrične krivulje, ki se oblikuje glede na sile znotraj krivulje in sile okoli krivulje. Delujejo tako, da moramo najprej določiti oziroma označiti približno območje našega zanimanja. Na označeni del se postavi krivulja (na začetku je lahko kar krog). Potem se začne iterativni

proces sproščanja, ki poskuša obdržati čim bolj homogeno regijo znotraj krivulje. Ta pristop segmentacije se zelo pogosto uporablja za segmentacijo medicinskih slik magnetne resonance iz področja rekonstrukcije možganske skorje [4]. Prednost teh metod je, da je rezultat podan s parametrizirano sklenjeno krivuljo z gladkimi robovi, ter neobčutljivost na šum. Slabost teh metod pa je, da je potrebno ročno določiti začetni model (krivuljo in njeno lokacijo) ter določiti primerne parametre.

3.3.8 Predloge

Segmentacija s predlogami je učinkovit pristop za segmentacijo medicinskih slik, ko imamo na voljo predlogo območja zanimanja. Konceptualno metode s predlogami delujejo podobno kot klasifikatorji. Glavna razlika je v tem, da predloge neposredno uporabljajo znanje o prostorski povezanosti, za razliko od klasifikatorjev, ki delujejo v prostoru značilnosti in običajno ne upoštevajo prostorske povezanosti slikovnih točk. Običajno segmentacija s predlogami predstavlja problem, ki se ga rešuje s poravnavo slik, saj želimo najti najboljše ujemanje naše predloge z delom regije v sliki. Poravnavo predloge z vhodno sliko dosežemo z zaporedjem linearnih in nelinearnih transformacij. Prednost pristopov s predlogami je, da se segmentacija regije in vsi strukturni elementi prenesejo, saj je predloga že segmentirana. Tako se rezultat neposredno prenese na ciljno sliko ob najdenem najboljšem ujemanju. Pristop segmentacije s predlogami se večinoma uporablja za segmentacijo slik magnetne resonance možganov. Slabost teh metod je, da so zelo občutljive na variabilnost ciljne regije. Zato je pristop s predlogami primerno uporabiti le v primerih, ko je iskana regija stabilna skozi celotno populacijo vhodnih podatkov.

3.4 Mere

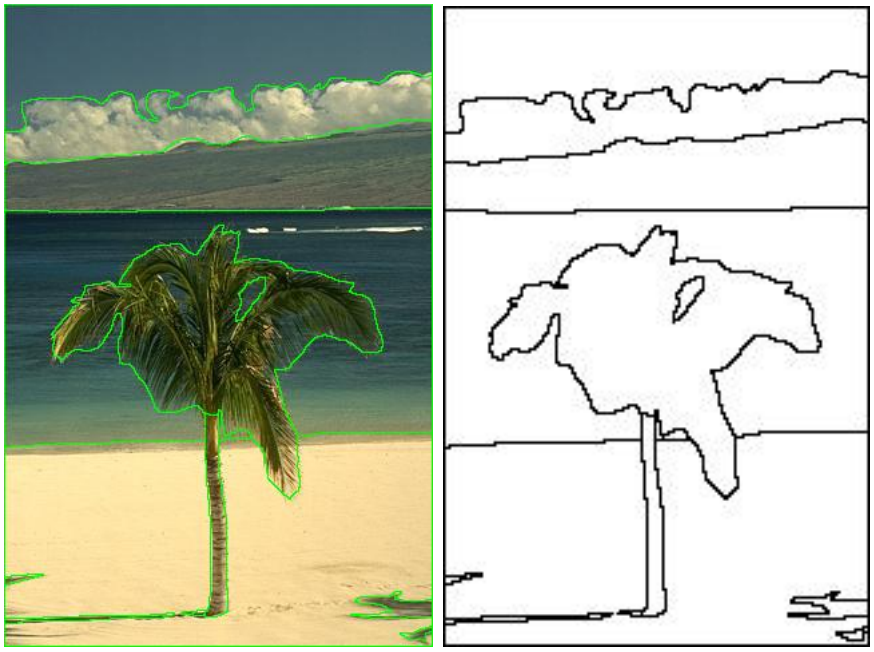
Pri razvoju in implementaciji algoritmov je zelo pomembno, da imamo na voljo ogrodje, ki nam omogoča konsistentno, ponovljivo in objektivno ocenjeva-

nje zmogljivosti algoritmov. Tako lahko enostavno preverimo, ali spremembe v algoritmu izboljšajo ali poslabšajo delovanje.

Za namen merjenja zmogljivosti smo uporabili različne mere, ki se uporabljajo za ocenjevanje segmentacij z referenčno segmentacijo (ang. *ground truth*) oziroma nadzorovano evalvacijo (ang. *supervised evaluation*). Kot zanimivost: pri nenadzorovani evalvaciji se uporablja algoritem STAPLE (ang. *Simultaneous Truth And Performance Level Estimation*), ki na podlagi probalističnih metod in rezultatov različnih ekspertnih sistemov [24] zgradi referenčno segmentacijo. Pristop z algoritmom STAPLE se velikokrat uporablja pri evalvaciji segmentacij medicinskih slik, saj za njih težko dobimo popolno referenčno segmentacijo. Zato zdravniški eksperti svojih področij ročno označijo segmente glede na njihovo presojo, iz katere se generira referenčna segmentacija. V našem primeru si nismo mogli privoščiti, da bi nam slike označil ekspert, kaj šele več ekspertov. Zato smo za namen tega dela sami označili slike glede na naše razumevanje predstavljenega problema in predpostavili, da je naša ročna segmentacija popolna referenca.

Pri reševanju problema segmentacije slik se običajno pričakuje enega izmed dveh vrst rezultatov. Do prvega tipa rezultata pridemo tako, da sliko rezdelimo na več homogenih regij glede na uporabljen kriterij in uporabljene značilke. Vsaka regija predstavlja različen razred (Slika 3.2). Takšni primeri segmentacije so primerni za predobdelavo slik za detekcijo ali prepoznavanje objektov. Do drugega tipa rezultata pridemo tako, da sliko razdelimo na regijo ozadja in ospredja oziroma na dva razreda (Slika 3.3). Takšen tip segmentacije je primerljiv z binarnim klasifikatorjem. Regije ozadja in ospredja niso nujno predstavljene z eno veliko povezano regijo. Primer uporabe takšnega tipa rezultatov segmentacije je zelo pogost pri obdelavi medicinskih slik. V našem primeru kot rešitev vračamo rezultate tipa ospredje-ozadje.

Standardne mere za oceno zmogljivosti binarnih klasifikatorjev ali segmentacij, ki sliko razdelijo na ospredje in ozadje ter se največkrat uporabljajo, so natančnost, specifičnost in občutljivost. Za izračun vrednosti posameznih mer moramo najprej razdeliti podatke v štiri kategorije. To po



Slika 3.2: Primer segmentacije, kjer je rezultat večje število razredov

Vir: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

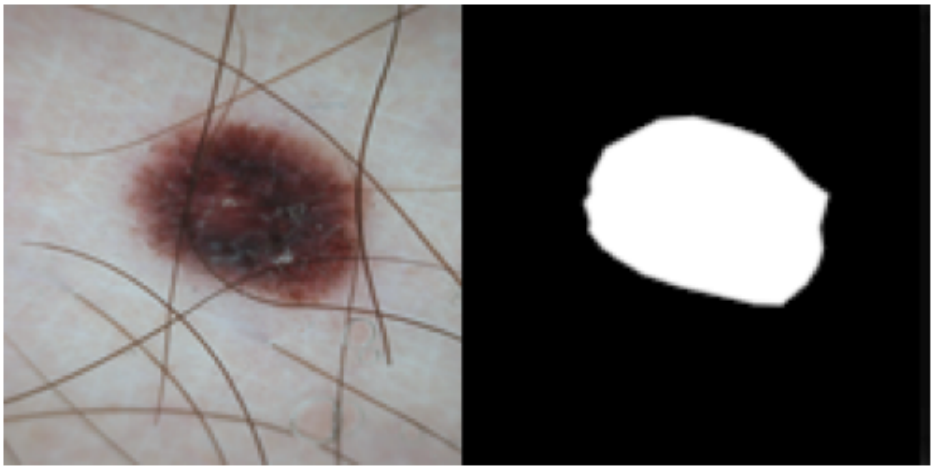
navadi storimo tako, da podatke tabeliramo v 2 x 2 tabelo 3.4.

V našem primeru, kjer želimo oceniti zmogljivost segmentacije glede na referenco, v vsak kvadrant tabele zapišemo število slikovnih točk, ki ustrezajo glede na celico tabele. V tabeli *ospredje* predstavlja segment in *ozadje* predstavlja tisto, kar ni segment.

1. TP – število slikovnih točk, ki so klasificirane kot segment v referenci in izhodu algoritma;

2. FP – število slikovnih točk, ki so klasificirane kot segment v izhodu algoritma in ne predstavljajo segmenta v referenci;

3. FN – število slikovnih točk, ki so klasificirane kot segment v referenci in ne predstavljajo segmenta v izhodu algoritma;



Slika 3.3: Primer segmentacije, kjer je rezultat ospredje in ozadje (dva razreda)

Vir: https://challenge2018.isic-archive.com/task1/

		ALGORITEM OSPREDJE	ALGORITEM OZADJE
REFERENCA	OSPREDJE	TP	FN
REFERENCA	OZADJE	FP	TN

Slika 3.4: Tabela za razdelitev podatkov

4. TN – število slikovnih točk, ki ne predstavljajo segmenta v referenci in izhodu algoritma.

Na podlagi te tabele lahko izračunamo vrednosti standardnih mer, ki v našem primeru delujejo nad slikovnimi točkami. Občutljivost, včasih tudi priklic (Enačba 3.8), je mera, ki nam pove, kolikšen delež slikovnih točk smo pravilno klasificirali. Sorodna mera, ki nam pove kolikšen delež slikovnih točk, ki ne predstavljajo segmenta, smo pravilno klasificirali, je specifičnost (Enačba 3.9). Tretja osnovna mera je natančnost (Enačba 3.10), ki nam pove, kolikšen delež klasificiranih slikovnih točk kot segment je pravilno kla-

sificiranih.

$$Se = \frac{TP}{TP + FN} \tag{3.8}$$

$$Sp = \frac{TN}{TN + FP} \tag{3.9}$$

$$Pr = \frac{TP}{TP + FP} \tag{3.10}$$

Običajno si želimo, da bi vse tri mere bile čim bližje vrednosti 1 (vrednost, ki jo zasežejo, je med 0 in 1). Da se izognemo spremljanju vseh vrednosti in primerjanj med različnimi eksperimenti, se v praksi velikokrat uporabi mero F (Enačba 3.11). F predstavlja harmonično razmerje med natančnostjo in priklicem. Tudi pri tej meri si želimo, da je vrednost čim bližje 1.

$$F1 = 2 * \frac{Pr * Se}{Pr + Se} \tag{3.11}$$

V literaturi in člankih se pojavita meri koeficient Dice in Jaccard. Obe sta tesno povezani z mero F, saj se ju lahko izračuna na podlagi mere F [25]. Katero uporabimo, je odvisno predvsem od domene problema. Pri evalvaciji segmentacije medicinskih slik se velikokrat uporabi koeficient Dice. Poleg standardnih mer se uporabljajo še [26]

1. Rand Index (RI);
2. Variation of Information (VoI);
3. Global Consistency Error (GCE);
4. Local Consistency Error (LCE);
5. Mean Squared Error (MSE);
6. Boundary Displacement Error (BDE);
7. Mean absolute surface distance;

8. razdalja Hamming in

9. razdalja Hausdorff.

Poglavje 4

Implementacija sistema

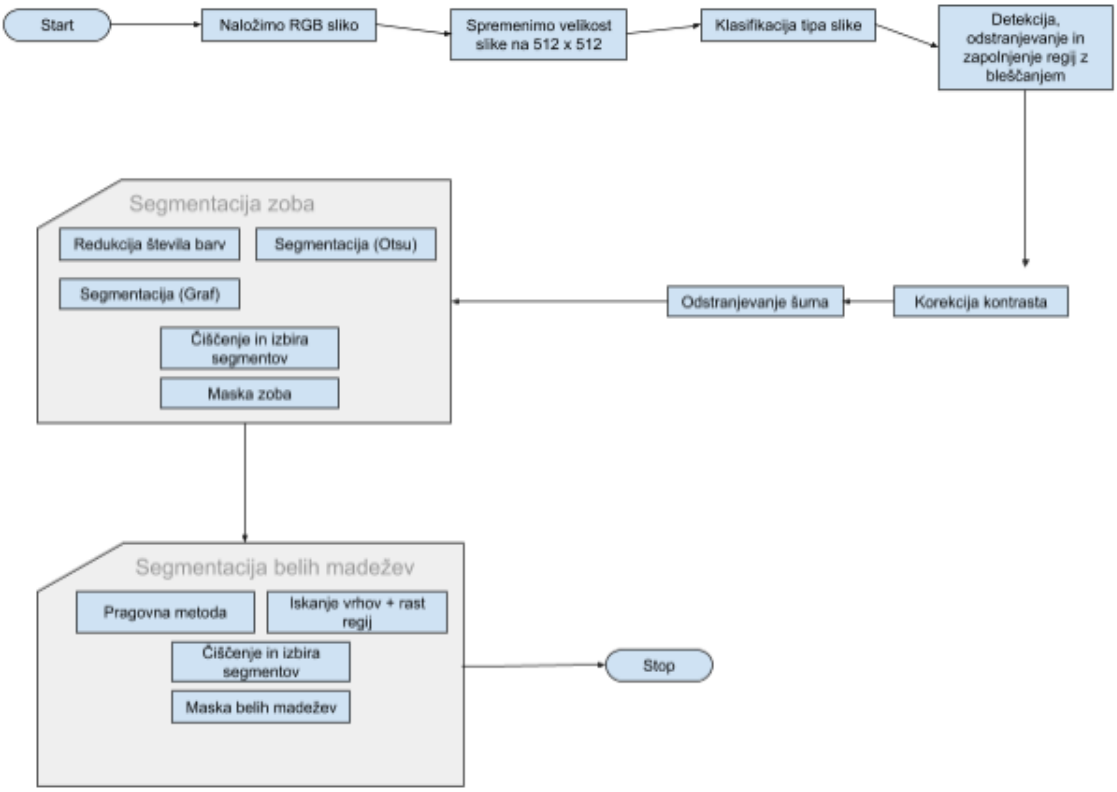
V tem poglavju predstavimo implementacijo sistema, ki smo ga razvili za avtomatsko segmentacijo zob in belih madežev. Najprej je predstavljen celoten sistem, ki je kasneje razdeljen v podpoglavja glede na tematske sklope. Rešitev je bila razvita v programskem jeziku Python z uporabo knjižnic OpenCV in SciPy.

4.1 Predobdelava

Visokonivojski pregled rešitve segmentacije zob in belih madežev je predstavljen z diagramom na sliki 4.1.

S procesom segmentacije začnemo tako, da najprej v pomnilnik naložimo vhodno sliko zoba, ki je predstavljena v barvnem prostoru RGB. Kot smo že omenili v poglavju 2, bi morale biti vse vhodne slike enake dimenzije posnete tako, da je zob v središču slike. Ker to ni držalo za vse pridobljene slike, so bile nekatere kasneje ročno obrezane. Zato smo po tem, ko smo sliko naložili v pomnilnik, spremenili njeno dimenzijo na velikost 512 x 512. To smo storili zato, da lahko uporabljamo enake parametre, ne glede na dimenzijo vhodne slike, ne da bi preveč vplivali na končni rezultat.

Kot smo že omenili v poglavju 2, smo zajeli dva tipa slik. Prvi tip so standardne barvne slike, brez posebnosti. Drug tip so zajete slike, obsijane



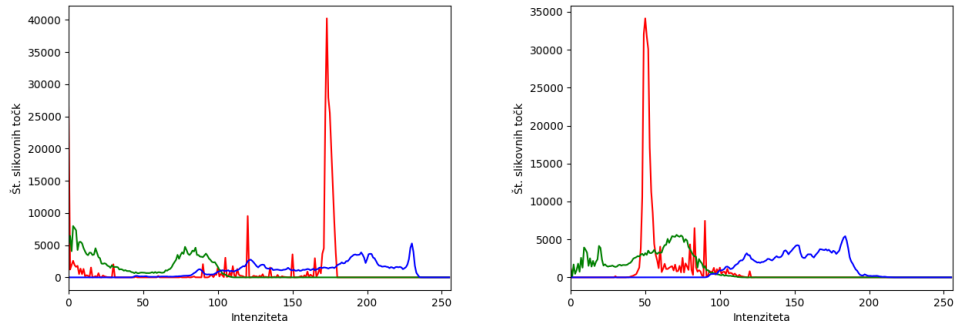
Slika 4.1: Diagram poteka segmentacije

z modro svetlobo valovne dolžine $450nm$. Da bi lahko kasneje v procesu segmentacije uporabili podatek o vrsti slike, smo implementirali preprost klasifikator, ki določi njen tip.

Klasifikator je implementiran kot prikazuje psevdokoda 1. V prvem koraku pretvorimo vhodno sliko v barvni prostor HSV, kjer uporabimo kanal barve za izračun histograma. Na podlagi empirično določenega razpona vrednosti kanala med 25 in 75 preverimo, če obstaja maksimum histograma, ki je hkrati tudi maksimum globalnega histograma. Če to drži, potem je bila vhodna slika zajeta z modro svetlobo, drugače pa ne. Na sliki 4.2 lahko vidimo primerjavo histogramov barvnega prostora HSV, normalne slike in obsijane z modro svetlobo. Kanal H je obarvan z rdečo barvo.

Algorithm 1 Psevdokoda klasifikatorja tipa slike

```
function ImageTypeClassifier(inputRgbImg)
    img ← rgb2hsv(inputRgbImg)
    hist ← calcHist(img)
    histH ← hist[0]
    localMax ← max(histH[25 : 75])
    globalMax ← max(histH)
    return globalMax > localMax
end function
```



Slika 4.2: Levo: histogram barvnega prostora HSV običajne slike; Desno: histogram barvnega prostora HSV slike pod modro svetlobo

V naslednjem koraku odpravimo bleščanje, ki se pojavlja na gladkih površinah zobne sklenine in dlesni. Za lokalizacijo regij bleščanja smo izvirali iz del [27, 28].

Postopek odstranjevanja je predstavljen s psevdokodo 2. Vhodno sliko pretvorimo v sivinsko ter jo normaliziramo tako, da vrednosti zavzamejo razpon med 0 in 1. Glede na literaturo [28] predlagajo pragovno vrednost med 0.8 do 0.9 za segmentacijo regij z bleščanjem. Glede na testiranja z različnimi vrednostmi smo izbrali pragovno vrednost 0.8, saj smo s to vrednostjo dobili najboljše rezultate.

Tako imamo sedaj masko regij, ki predstavljajo bleščanje. Trenutno stanje maske pokriva le dele slike, kjer je bleščanje najmočnejše, v večini pri-

Algorithm 2 Psevdokoda odstranjevanje bleščanja

```
1: function GlareRemoval(inputRgbImg)
2:     img ← rgb2gray(inputRgbImg)
3:     imgNorm ← normalize(img)
4:     mask ← imgNorm > 0.8
5:     kernel ← ('ELLIPSE', (5, 5))
6:     mask ← dilate(mask, kernel)
7:     mask ← selectRegions(mask)
8:     glarelessImg ← fillMaskedRegions(img, mask)
9:     return mask, glarelessImg
10: end function
```

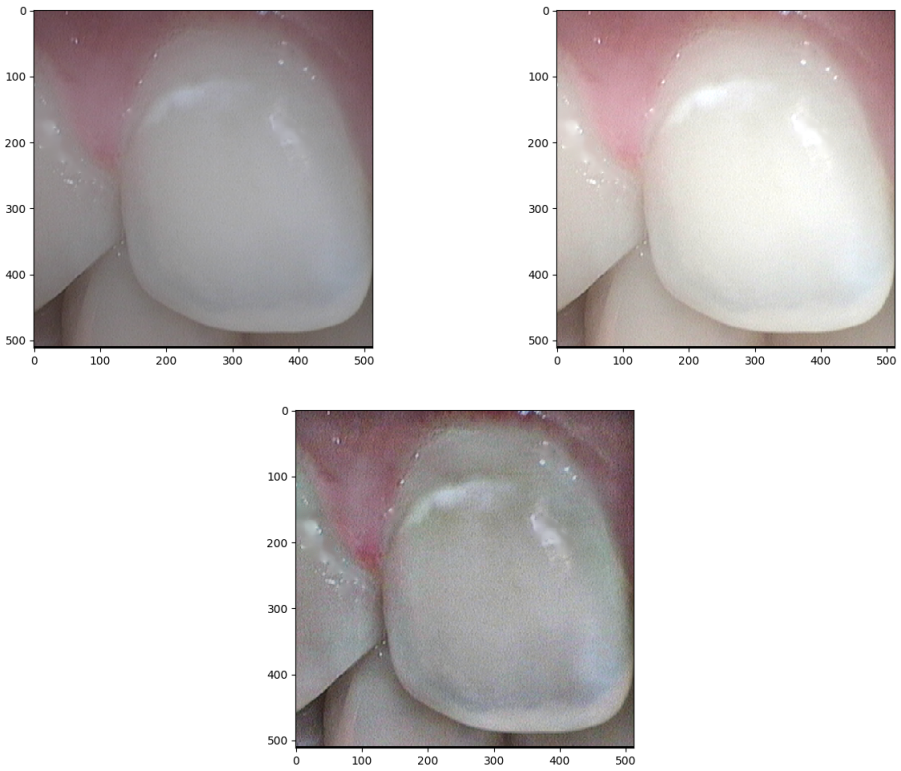
merov pa ne pokrije prehoda med sliko in bleščanjem. Če teh prehodov ne odstranimo, bomo kot rezultat dobili robove okoli regij, kjer je bilo pred tem prisotno bleščanje. Tega se lahko znebimo tako, da povečamo regije v maski. To storimo s pomočjo morfološkega operatorja za razširitev. Za strukturni element smo uporabili krog velikosti 5 x 5.

Novo stanje maske je, da pokriva celotno območje, kjer je prisotno bleščanje. Kot smo omenili v poglavju 3, pragovne metode ne dajo najbolj optimalnih rešitev, saj upoštevajo le intenziteto slikovnih točk. To smo opazili tudi pri implementaciji odstranjevanja bleščanja, da na nekaterih vhodnih slikah lokaliziramo bleščanje, ki zajema večji del slike in v resnici ne predstavlja regije z bleščanjem. Zato smo postopku odstranjevanja dodali še dodaten korak, ki glede na velikost regij izloči regije, ki presegajo četrtino velikosti vhodne slike. V članku [28] regije, prekrite z masko, zapolnijo z uporabo Laplacove interpolacije. V našem primeru smo za potrebe interpolacije uporabili algoritem znotraj knjižnice OpenCV.

Funkcija, ki smo jo uporabili za zapolnitev delov s prisotnim bleščanjem, se imenuje *Inpaint*. Metode *Inpaint* delujejo tako, da zapolnijo manjkajoče slikovne točke glede na okolico. OpenCV implementira metodo, ki temelji na Navier-Stokes, in metodo, ki temelji na metodi *fast marching* [29].

V naslednjem koraku smo uporabili metode za korekcijo kontrasta. Iz analize pridobljenih slik smo ugotovili, da je pri večini primerov kontrast

med regijami, ki jih na koncu želimo dobiti kot različne segmente, zelo nizek. Zato smo z uporabo metode za razteg histograma izboljšali kontrast. Najprej smo poskusili z globalnim raztegom histograma, vendar smo ugotovili, da rezultati niso bistveno boljši od vhodne slike. Zato smo uporabili razteg histograma, ki deluje lokalno le na manjšem oknu celotne slike. Za razteg histograma z metodo CLAHE [30] smo uporabili implementacijo znotraj knjižnice OpenCV. Uporabljeno okno je velikosti 32 x 32. Na sliki 4.3 je prikazan primer vhodne slike in razteg histograma z obema metodama.



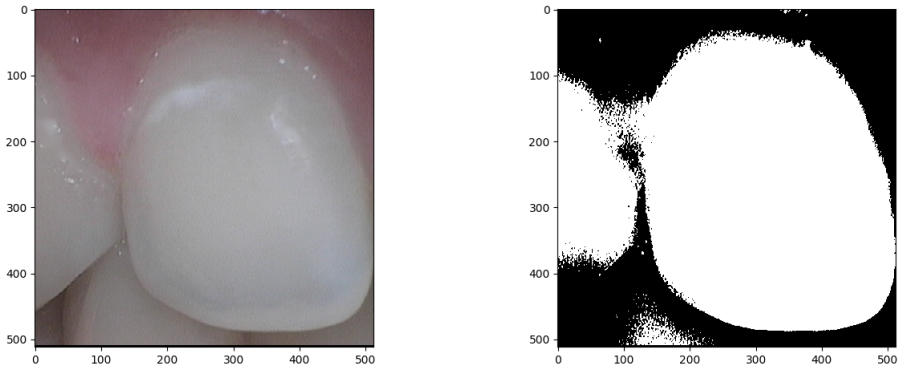
Slika 4.3: Prva: vhodna slika; Druga: globalni razteg histograma; Tretja: CLAHE

V zadnjem koraku predobdelave smo odstranili šum v slikah. Šum smo zmanjšali z uporabo Gaussovega filtra.

4.2 Segmentacija zob

Naš osnovni cilj je pridobiti segmente zob in belih madežev, da bi lahko ocenili delež prizadete površine zobne sklenine. Za ta namen smo segmentirali tako zobno sklenino kot bele madeže. Ker sta si problema različna, smo ju ločili in pripravili rešitev za vsakega posebej.

Pri segmentaciji zob smo v prvem koraku uporabili pragovno metodo Otsu [31]. Za to metodo smo se odločili, ker je hitra in enostavna. Čeprav rezultati niso najbolj natančni, nam vseeno dobro loči ospredje od ozadja (Slika 4.4). V našem primeru ospredje predstavlja zobno sklenino. Za segmentacijo z metodo Otsu smo uporabili implementacijo iz knjižnice OpenCV. Metoda predpostavlja, da vhodna slika vsebuje dva razreda slikovnih točk, ki se jih na histogramu prepozna kot dva vrhova (ang. *bi-modal histogram*), ločena z globoko dolino – ospredje in ozadje. Metoda poišče optimalen prag, ki loči slikovne točke na ospredje in ozadje. Optimalen prag se izračuna tako, da za vsako neničelno vrednost v histogramu sivinske slike (kar predstavlja maksimalno 256 različnih pragovnih vrednosti) izračuna razpršenost histograma glede na trenutno vrednost praga. Rešitev segmentacije je pragovna vrednost, pri kateri je razpršenost najmanjša.



Slika 4.4: Prva: vhodna slika; Druga: Rezultat segmentacije z metodo Otsu

Na desni sliki 4.4 opazimo, da segmentacija vsebuje nekaj šuma in segmen-

tira tudi sosednje zobe, ki nas trenutno ne zanimajo. Zato po segmentaciji sledi čiščenje, predstavljeno s psevdokodo 3. Najprej uporabimo morfološki operator erozije, da odstranimo morebitne manjše povezave med večjimi regijami segmentacije. V tem primeru smo uporabili strukturni element oblike kvadrat, dimenzije $21x21$. Potem poiščemo povezane regije znotraj slike. To storimo s klicem funkcije *connectedComponentsWithStats*, ki je del knjižnice OpenCV. Za tip povezanosti smo uporabili 4-smerno povezanost. Potem iz rezultata povezanih komponent poiščemo največjo regijo.

Glede na predpostavko, da se segmentiran zob nahaja v sredini slike in pokriva večji del slike, lahko sklepamo, da je največja povezana regija tista, ki predstavlja segmentacijo zoba. Ker smo nekaj korakov nazaj uporabili morfološki operator erozije, ki zmanjša regije, moramo sedaj uporabiti nasprotni operator, da povrnemo prvotno površino segmenta. Zato uporabimo razširitev, ki se ji reče diletacija, z istim strukturnim elementom, kot smo ga uporabili pri eroziji.

Algorithm 3 Psevdokoda čiščenja segmentacije Otsu

```
1: function CleanSegmentation(imgSegmentation)
2:     kernel ← createStructElement('RECT', (21, 21))
3:     mask ← erode(mask, kernel)
4:     connectedComponents ← connectedComponents(mask)
5:     component ← findBiggestComponent(connectedComponents)
6:     result ← createEmptyImage(imgSegmentation.shape)
7:     result ← result||component
8:     result ← dilate(result, kernel)
9:     return result
10: end function
```

V drugem koraku smo uporabili pristop segmentacije z grafi, kjer vsako vozlišče predstavlja slikovno točko ali manjšo skupino slikovnih točk, povezave med vozlišči pa predstavljajo razdaljo glede na barvo. Osnovna ideja algoritma je vzeta iz članka [32] in prilagojena za naš problem.

4.2.1 Segmentacija z grafi

V našem primeru je vhodni podatek 2D slika v poljubnem barvnem prostoru. Na koncu se je izkazalo, da so rezultati najboljši, če je vhodna slika v barvnem prostoru RGB. V osnovi je algoritem zasnovan, da deluje na enokanalnih slikah. Vendar se ga lahko enostavno razširi, da deluje tudi na več kanalnih slikah oziroma z dodatnimi značilkami. To storimo tako, da se algoritem izvede za vsak kanal posebej in na koncu vse tri rešitve združimo. Še boljša rešitev pa je, da vse kanale upoštevamo neposredno pri izračunu uteži povezave med vozlišči.

Kot pri ostalih metodah za segmentacijo, ki temeljijo na grafih, moramo tudi pri tej najprej zgraditi graf $G = (V, E)$. V našem primeru je to neusmerjen graf, kjer je vozlišče $v \in V$ in predstavlja element, ki ga želimo segmentirati. Robovi $(v_i, v_j) \in E$ definirajo prostorsko povezanost med sosednjimi vozlišči oziroma slikovnimi točkami v sliki. Pri grajenju grafa lahko uporabimo različne vrste povezanosti, kot so 4-smerno, 6-smerno ali 8-smerno. Robovi predstavljajo različnost med elementoma, ki ju rob povezuje. Za mero različnosti lahko uporabimo različne lokalne značilnosti slikovnih točk, kot so barva, intenziteta, lokacija, robovi ... Na koncu si želimo, da bi bila vozlišča, ki so si podobna, v isti regiji, vozlišča, ki so si različna, pa v različnih. Tako bi morala imeti vozlišča znotraj iste regije relativno majhne vrednosti povezav z vozlišči znotraj iste regije in relativno visoke vrednosti povezav z vozlišči med različnimi regijami.

Segmentacija z uporabo grafov je definirana kot razdelitev vozlišč V v segmentacijo S tako, da je vsaka regija R vsebovana v segmentaciji $R \in S$. Segmentacija S predstavlja povezano regijo grafa $G' = (V, E')$, kjer je $E' \subseteq E$. Z drugimi besedami, vsaka regija je predstavljena s podmnožico povezave E v grafu G.

Za rešitev problema segmentacije z grafi moramo rešiti problem najmanjšega vpetega drevesa oziroma MST (ang. *minimum spanning tree*) v neusmerjenem grafu. Rešitev MST predstavlja najmanjšo podmnožico povezav v grafu, z najmanjšo skupno ceno, s katerimi lahko obiščemo vsa vozlišča,

brez ciklov. Dva izmed znanih algoritmov za reševanje MST problema sta *Prim* [33] in *Kruskal* [34]. Oba algortima delujeta po principu požrešnih metod (ang. *greedy algorithm*), kar pomeni, da uporabljata lokalno hevristiko za odločanje v vsakem koraku za namen dosega globalne optimalne rešitve. Pri implementaciji se uporabi algoritem, ki temelji na algoritmu *Kruskal*. Glavno vlogo pri implementaciji tega algoritma igra podatkovna struktura *merge-find set* [35]. Podatkovna struktura se uporablja za spremljanje elementov, v našem primeru vozlišč, ki so razdeljeni v podmnožici brez prekrivanj. Podatkovna struktura omogoča iskanje elementov v podmnožicah in njihovo združevanje v skoraj konstantnem času [36].

Spodaj je opis algoritma, ki kot vhodne parametre sprejme graf $G = (V, E)$ z n vozlišči in m povezavami. Rešitev algoritma pa je segmentacija vozlišč V v povezane komponente $S = (C_1, ..., C_r)$. Za izračun vrednosti uteži povezav smo uporabili Evklidsko razdaljo 4.1.

$$dist(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{4.1}$$

1. Uredimo robove E v $\pi = (o_1, ..., o_m)$ v naraščajočem vrstnem redu.

2. Začnemo s segmentacijo S^0, kjer vsako vozlišče v predstavlja svojo regijo z notranjo razliko (ang. *internal difference*) 0.

3. Ponavljamo korak 4 za vse povezave $q = 1, ..., m$.

4. Generiramo novo generacijo segmentacije S^q glede na segmentacijo prejšnjega koraka S^{q-1}. Naj v_1 in v_j predstavljata vozlišči, ki ju povezuje povezava e_q. Če vozlišči nista v isti podmnožici generacije segmentacije S^{q-1} in je vrednost povezav med njima $w(v_i, v_j)$ manjša ali enaka notranji razliki, potem združimo podmnožici, katerih elementa sta v_i in v_j.

5. Končni rezultat je segmentacija generacije S^m.

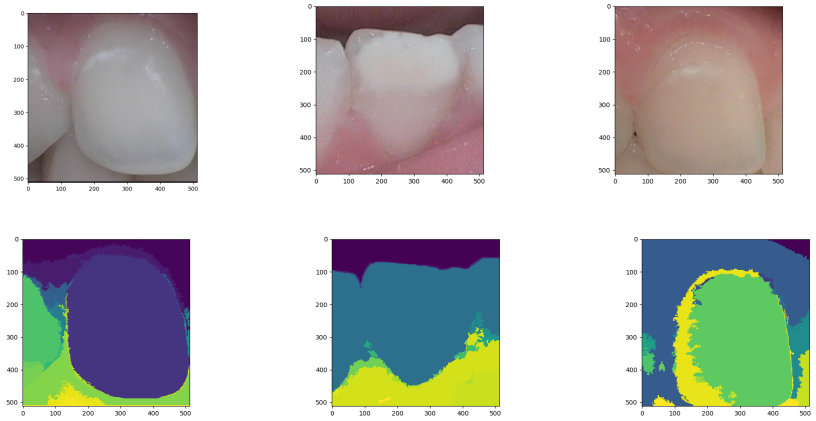
Notranja razlika regije $R_i \subseteq V$ je definirana kot največja vrednost povezave $MST(R_i, E)$ 4.2.

$$Int(R_i) = max_{e \in MST(R_i,E)} w(e) \tag{4.2}$$

Pri implementaciji posodabljanja notranje razlike komponent ni potrebno dodajati nobenih posebnosti. V prvem koraku algoritma uredimo povezave v naraščajočem vrstnem redu, kar pomeni, da se vsako novo dodano vozlišče ali podmnožica, ki se pridruži obstoječim vozliščem, povezuje z robom, ki ima trenutno največjo vrednost. Tako vedno, ko posodabljamo notranjo razliko regij, vzamemo vrednost povezave, ki je bila nazadnje dodana.

Če si dobro pogledamo 2. korak algoritma v zgornjem opisu, kjer ustvarimo prvo generacijo segmentacije S^0 in vsako vozlišče predstavlja svojo regijo, lahko opazimo, da imajo vse notranje razlike vrednost 0. V tem primeru bi algoritem na koncu vrnil rešitev, ki bi bila enaka segmentaciji S^0 oziroma regije bi zrastle samo, če se povezujejo s sosedi s povezavami vrednosti 0. Prav tako bi se isti problem pojavil pri majhnih regijah, ki zajemajo vozlišča z zelo majhno varianco značilk, po katerih jih združujemo. Za reševanje tega problema enačbi 4.2 dodamo mejno vrednost $\tau(R_i) = \frac{k}{|R_i|}$, kjer je $|R_i|$ moč množice. Vrednost k je konstanta, ki jo podamo na začetku klica algoritma. Tako lahko z vrednostjo k spreminjamo, koliko slikovnih točk lahko združimo, brez preverjanja mejne vrednosti notranje razlike, v poznejših iteracijah pa doda nekaj šuma notranji razliki. V našem primeru smo vrednost k nastavili na 1500.

Ob koncu zgoraj opisanega postopka končamo s segmentacijo S^m, ki vsebuje regije različnih velikosti. Ker v našem primeru segmentiramo zobe, ki zajemajo večji del slike, lahko upoštevamo predpostavko, da bomo imeli le regije z večjimi površinami. Zato naredimo še dodatno iteracijo, v kateri se sprehodimo čez vse robove začetnega grafa. Če vozlišči nista v isti podmnožici in ima katera od množic moč manjšo od mejne vrednosti, ju združimo. V našem primeru smo minimalno velikost regije omejili na površino 250 slikovnih točk. Primer rezultata segmentacije s tem algoritmom je prikazan na sliki



Slika 4.5: Prva vrsta: vhodna slika; Druga vrsta: Rezultat segmentacije z grafi

4.5. Barve na slikah segmentacije so naključne in predstavljajo le vrednosti različnih razredov regij.

Kot lahko razberemo iz slik 4.5, bi lahko rezultat segmentacije razdelili v tri skupine: ravno pravšnja, prevelik segment in premajhen segment. Za izbiro pravega segmenta ali segmentov si bomo pomagali s segmentacijo, pridobljeno v prvem koraku.

Sedaj potrebujemo način, da izberemo en ali več segmentov, ki predstavljajo segmentacijo zoba. Kot smo že omenili pri segmentaciji z metodo Otsu, je rešitev v večini primerov manjša kot bi si želeli. Zato masko segmentacije z metodo Otsu uporabimo kot izhodišče za izbiro potencialnih regij segmentacije, pridobljene z grafi. Za lažjo razlago označimo segmentacijo, pridobljeno z metodo Otsu kot S_0, in segmentacijo, pridobljeno z grafi, kot S_1.

V prvem koraku poiščemo vse kandidate regij K. Kandidati so vse regije v S_1, ki jih prekriva S_0. Potem za vsako regijo $k \in K$ izračunamo, kolikšen del regije ne pripada S_0, $\forall k \in K; |k \setminus S_0|$. V naslednjem koraku dodamo vsak $k \in K$, ki je popolnoma prekrit s S_0, $|k \setminus S_0| = 0$ ali delno prekrit do praga τ, $|k \setminus S_0| \leq \tau$, v množico regij končne rešitve R. S tem pristopom pokrijemo primere tipa prvega in tretjega stolpca slike 4.5. V primeru, ko

je množica $|R| = 0$, se vzame kandidata, ki ima največjo skupno regijo s S_0, $k \in K; max(|k \cap S_0|)$. Tako na koncu množico rešitev R spremenimo v binarno masko, ki predstavlja segmentacijo zoba S_z.

Kot smo omenili na začetku poglavja 4, smo vsem slikam spremenili dimenzijo na 512 x 512. To pri gradnji grafa in njeni segmentaciji predstavlja 512^2 vozlišč, ki jih je potrebno obiskati in jim dodeliti razred. Da bi iskalni prostor zmanjšali, smo najprej poskusili z manjšimi dimenzijami slik, brez da bi to imelo večji vpliv na končni rezultat. Tako smo dimenzijo slik za primer segmentacije z grafi zmanjšali na 256 x 256. To je meja, pri kateri so rezultati ostali skoraj nespremenjeni. Pri gradnji začetnega grafa smo tudi zmanjšali povezanost iz osem na štiri.

Prav tako pa smo s postopkom redukcije barv zmanjšali paleto različnih barv, kar nam je omogočilo, da slikovne točke slike, ki imajo isto barvo in so si sosednje, združimo že v procesu gradnje začetnega grafa. Pri redukciji števila barv moramo določiti število dovoljenih različnih barv. Če je izbrano število preveliko, potem ne bo velike spremembe v primerjavi z vhodno sliko. V primeru, ko pa izberemo premajhno število, se nam bodo začele regije združevati, kar pa ni naš prvotni namen. Prav tako smo z eksperimentiranjem ugotovili, da je najboljše, če se uporabi isti barvni prostor za redukcijo števila barv, kot se ga uporablja kasneje za segmentacijo. Slike, tik pred vhodom v segmentacijo z grafi, vsebujejo od 15000 do 26000 različnih barv. Z redukcijo števila barv smo jih zmanjšali na 200.

Rezultati pohitritve časa izvajanja so predstavljeni v poglavju 5.

4.3 Segmentacija belih medežev

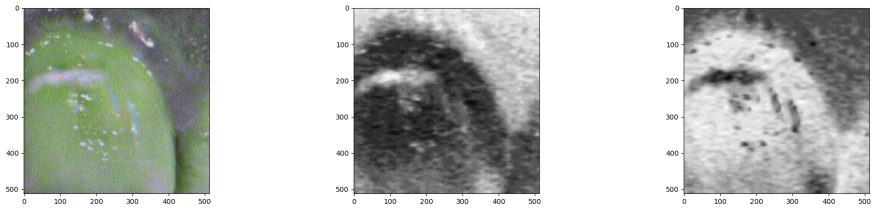
Kot smo omenili v poglavju 2, smo zajemali dve vrsti slik. Zato smo tudi v korakih predobdelave klasificirali slike glede na njihovo vrsto. Ta podatek uporabimo pri segmentaciji belih madežev, saj glede na vrsto slike prilagodimo postopek. Prav tako v procesu odstranjevanja bleščanja vrnemo sliko brez bleščanja in masko z regijami, kjer se je bleščanje nahajalo. V primeru

segmentacije belih madežev uporabimo to masko za zavrnitev FP kandidatov regij z belimi madeži. V postopku segmentacije belih madežev uporabimo tudi masko segmentacije zoba, s katero izločimo vse kandidate regij belih madežev, ki se ne nahajajo znotraj segmenta zoba.

Za segmentacijo belih madežev smo hoteli uporabiti isti algoritem kot za segmentacijo zob, vendar je problem, saj algoritem ni dovolj občutljiv na manjše intenzitetne spremembe. Prav tako smo poskusili različne metode za detekcijo robov, da bi z njimi dobili oris madežev, vendar zaradi prevelike prisotnosti šuma oziroma nizke kakovosti slik nismo uspeli pridobiti rezultatov, s katerimi bi lahko uspešno segmentirali območja našega zanimanja. Poskusili smo tudi z metodami klasificiranja in gručenja, na podlagi GLCM (ang. *Gray-Level Co-Occurrence Matrix*) z Haralickovimi značilkami (ang. *Haralick features*) [37] in različnimi barvnimi prostori, vendar rezultati niso bili preveč uspešni.

Med analizo slik smo ugotovili, da so beli madeži vedno svetlejše barve kot okolica, vendar se ta ista barva, katere je beli madež, lahko pojavi tudi na drugih delih istega zoba, kar predstavlja problem. Pri analizi slik, zajetih z modro svetlobo, smo ugotovili, da lahko zelo enostavno segmentiramo kandidate regij z belimi madeži glede na intenziteto slikovnih pik v kanalu A ali B barvnega prostora CIELAB. Kot primer si lahko pogledamo sliko 4.6. Na sliki kanala A in B lahko vidimo, da beli madež popolnoma izstopa v intenziteti od okolice, ki predstavlja zdravo sklenino. Tako smo za pridobitev kandidatov regij segmentacije belih madežev te vrste slik lahko uporabili pristop z upragovljenjem.

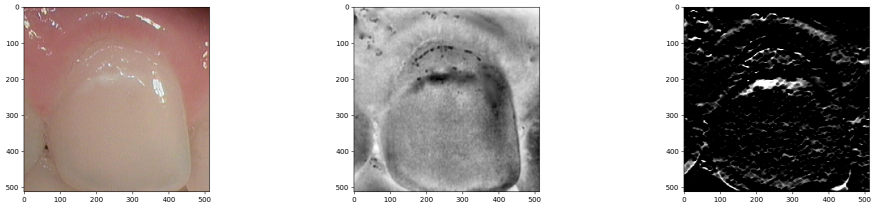
Za pridobitev kandidatov regij belih madežev na slikah, zajetih brez modre svetlobe, smo implementirali svoj algoritem, ki temelji na rasti regij. Za to vrsto algoritma smo se odločili, ker nam omogoča segmentacijo glede na lokalne značilke z globalnimi pravili. Kot smo že zgoraj omenili, je del z belimi madeži vedno svetlejše barve kot okolica sklenine, kar pomeni, če uspemo najti vsaj eno točko znotraj belega madeža in uporabimo segmentacijo rasti regij, lahko te regije segmentiramo glede na podobnost intenzitete sosednjih



Slika 4.6: Levo: vhodna slika; Sredina: kanal A; Desno: kanal B

slikovnih točk.

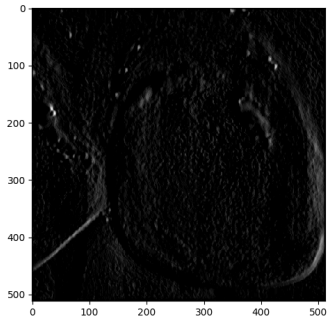
Zato je prvi korak algoritma iskanje začetnih točk (ang. *seed points*), ki se bodo nahajale znotraj belih madežev. Za iskanje kandidatov točk belih madežev si želimo uporabiti sliko, na kateri se najbolje vidijo ostri prehodi med sklenino in belimi madeži. Z eksperimentiranjem smo prišli do ugotovitve, da se prehodi dobro vidijo na sivinski sliki. Primer take slike je sredinska slika 4.7 (slika ima invertirane intenzitete).



Slika 4.7: Levo: vhodna slika; Sredina: sivinska slika vhodne slike; Desno: razlika intenzitet po filtriranju

Za pridobitev približne lokacije belega madeža oziroma večjih prehodov med intenzitetami, smo uporabili filtriranje z linearnim filtrom velikosti $9x9$ 4.3. Ideja izhaja iz pristopov za iskanje robov. Pri iskanju robov se uporablja na primer Sobelov filter, ki je velikosti $3x3$. Težava takšnega filtra je, da zazna premike intenzitete, ne dobimo pa informacije o tem, kako velik je prehod. To pomeni, da bi kot rezultat dobili veliko črt in točk (Slika 4.8), iz katerih bi zelo težko ugotovili, kateri robovi predstavljajo prehod med regijami in kateri

prehodi so le posledica šuma. Postopek iskanja začetnih točk je predstavljen s psevdokodo 4.



Slika 4.8: Primer rezultata filtriranja s filtrom Sobel

S filtrom izračunamo intenzitetne razlike v smeri y. Prav tako bi lahko uporabili isti filter, obrnjen za 90 stopinj, za iskanje prehodov v smeri x, vendar v našem primeru to ni potrebno, ker želimo pridobiti le lokacije z visoko razliko intenzitet.

$$kernel = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.3}$$

Po filtriranju s filtrom 4.3 dobimo sliko, kot je prikazana na tretji sliki 4.7. Večja kot je razlika v intenziteti na vhodni sliki, višja je intenziteta na izhodu.

Ker je naše območje zanimanja le znotraj segmenta zoba, odstranimo vse dele slike, ki se ne nahajajo v segmentu zoba. Ta korak bi lahko na-

redili že prej, vendar je potem implementacija manipulacije slike bistveno kompleksnejša, saj bi morali nekako izpustiti slikovne točke, ki se pojavljajo na robovih našega zanimanja, ali pa bi dobili močan odziv filtra na robovih zob. Pred apliciranjem maske zoba za odstranitev delov slike, ki ne pripadajo zobu, te zmanjšamo z erozijo, da odstranimo še odzive filtra na robovih zoba.

Algorithm 4 Psevdokoda iskanja začetnih točk

```
1: function FindSeedPoints(img, toothMask, windowSize, th)
2:     seeds ← []
3:     grayImg ← (255 − rgb2gray(img))
4:     kernel ← [...] // Matrika 4.3
5:     candidates ← filter2D(grayImg, kernel)
6:     kernel ← createStructElement('ELLIPSE', (5, 5))
7:     candidates = erode(gradient, kernel)
8:     candidates = medianBlur(candidates, 11)
9:     toothMaskErode ← erode(segments, kernel)
10:    candidates = bitwiseAnd(candidates, toothMaskErode)
11:    peakCutValue = max(candidates) ∗ 0.95
12:    peaks = candidates[candidates > peakCutValue]
13:    components = connectedComponents(peaks)
14:    for component in components do
15:        minValXY ← findMinValueCoordinates(component)
16:        patch ← getPatchWithCenterAndSize(img, minValXY, windowSize)
17:        minValXYPatch ← findClosestMinToPoint(patch, minValXY, th)
18:        seeds ← minValXYPatch
19:    end for
20:    return seeds
21: end function
```

Na sliki rezultata filtriranja poiščemo vse slikovne točke, ki predstavljajo zgornjih 5% intenzitete. Sedaj imamo sliko, ki je sestavljena iz manjših regij, ki se pojavijo na robovih belih madežev. Ker za algoritem rasti regij potrebujemo le točko, moramo iz vsake regije dobiti vsaj eno lokacijo slikovne točke, ki bo predstavljala eno izmed začetnih točk algoritma. To storimo tako, da s pomočjo OpenCV funkcije *connectedComponents* ločeno obravnavamo

vsako povezano regijo. Znotraj vsake regije poiščemo naključno slikovno točko $minValXY$ z najmanjšo vrednostjo in si zapomnimo njene koordinate. Znotraj okna velikosti $windowSize$ s centrom na koordinatah $minValXY$ poiščemo najmanjšo vrednost slikovne točke. V našem primeru smo uporabili okno velikosti 100. To storimo tako, da se iz koordinat $minValXY$ premikamo v vse smeri, katerih vrednost intenzitete je manjša ali enaka. V primeru, da je takih točk več, izberemo točko z najmanjšo evklidsko razdaljo do $minValXY$. Preden vrnemo koordinate novega minimuma, preverimo še, da je vrednost manjša od vrednosti na lokaciji $minValXY$ in pod pragom th.

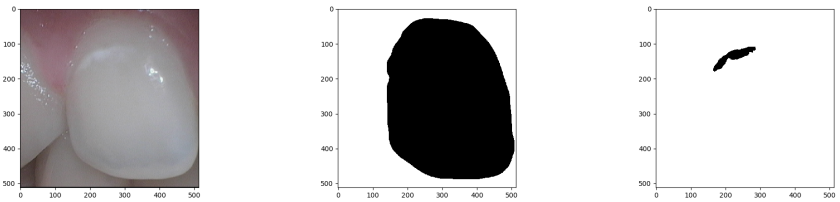
Algorithm 5 Psevdokoda rasti regij

```
1:  function RegionGrowing(img, seed, th)
2:      visited ← []
3:      result ← []
4:      toVisit ← [seed]
5:      while toVisit not empty do
6:          point ← toVisit.pop()
7:          if isValidPosition(point) then
8:              if visited[point] then
9:                  continue
10:             end if
11:             visited[point] ← true
12:             pixelValue ← img[point]
13:             if pixelValue < th then
14:                 result[point] ← 1
15:                 toVisit ← append(toVisit, point + [1, 0])
16:                 toVisit ← append(toVisit, point + [−1, 0])
17:                 toVisit ← append(toVisit, point + [0, 1])
18:                 toVisit ← append(toVisit, point + [0, −1])
19:             end if
20:         end if
21:     end while
22:     return result
23: end function
```

Za vsako slikovno točko v seznamu poženemo rast regij. Postopek rasti regij je predstavljen v 5. Premakniti se poskušamo v štiri smeri in dodamo slikovno točko v regijo, če zadostuje pogoju. Ko regije nehajo rasti, jih dodamo v rešitev regij s kandidati.

Sedaj imamo kandidate belih madežev za slike zajete z modro svetlobo in kot navadne slike. Sledi čiščenje slike kandidatov. Tako kot pri čiščenju segmentacij zob, tudi pri tem uporabimo morfološke operatorje za čiščenje slike, kot to storimo v psevdokodi 3, le da tukaj ne vrnemo le največje regije, ampak vse.

Po čiščenju kandidatov moramo odstraniti še kandidate, ki se ne nahajajo na zobu ali pa so del bleščanja. Odstranjevanje FP kandidatov belega madeža je predstavljeno z 6. Funkcija sprejme masko zoba, kandidatov belih madežev, bleščanja in vrednost t, s katero lahko nastavimo, kolikšen delež kandidata mora biti del bleščanja, da ga zavrnemo. Najprej pripravimo rezultat, v katerega bomo dodajali kandidate, ki predstavljajo bele madeže. Tako maski kandidatov belih madežev in maski z regijami bleščanja odstranimo vse regije, ki niso del zoba. Nato se še sprehodimo skozi vse kandidate belih madežev ter preverimo, če se prekrivajo z regijami bleščanja. V primeru prekrivanja preverimo, če delež prekrivanja presega mejno vrednost t. V primeru, da jo preseže, take regije ne dodamo v končno rešitev regij belih madežev.



Slika 4.9: Primer rezultata segmentacije

Na koncu kot rezultat vrnemo masko zoba in belih madežev (Slika 4.9).

Algorithm 6 Psevdokoda odstranjevanje FP kandidatov

```
1:  function RemoveFPLesionCandidates(maskTooth, maskLesion, maskGlare, t)
2:      result = createEmptyImage(maskTooth.shape)
3:      maskLesion ← bitwiseAnd(maskTooth, maskLesion)
4:      maskGlare ← bitwiseAnd(maskTooth, maskGlare)
5:      lesionRegions ← connectedComponents(maskLesion)
6:      glareRegions ← connectedComponents(maskLesion)
7:      for lr in lesionRegions do
8:          shouldAdd ← true
9:          for gr in glareRegions do
10:             overlap ← bitwiseAnd(lr, gr)
11:             if area(overlap)/area(gr) >= t then
12:                 shouldAdd ← false
13:             end if
14:         end for
15:         if shouldAdd then
16:             result ← bitwiseOr(result, lr)
17:         end if
18:     end for
19:     return result
20: end function
```

Poglavje 5

Rezultati

V tem poglavju predstavimo rezultate, pridobljene z implementiranim sistemom. Za testiranje smo uporabili zbirko 30 slik zob z belimi madeži. Od tega je 10 slik zajetih z modro svetlobo in 20 navadnih slik. Testiranje smo izvedli na računalniku z operacijskim sistemom Ubuntu 16.04 s procesorjem Intel i7 4700HQ @ 3.40 GHz in 16 GB pomnilnika.

Glavni cilj dela je bila implementacija prototipa sistema za avtomatsko ocenjevanje prizadete površine zobne sklenine z belimi madeži. Ker smo problem razdelili na segmentacijo zob in belih madežev, bomo ločeno predstavili tudi rezultate.

5.0.1 Zmogljivost segmentacije zob

Kot smo omenili v poglavju 4, smo najprej segmentirali z metodo Otsu. Ta metoda nam je zelo dobro vračala regije, ki se nahajajo na delu slike, kjer imamo zanimanje. To lahko tudi potrdimo s spodnjo tabelo 5.1. Tabela prikazuje različne mere zmogljivosti segmentacije z metodo Otsu. Prvi stolpec prikazuje zaporedno številko primera slike, potem pa si sledijo mera F, natančnost, občutljivost in specifičnost. Zadnja vrstica predstavlja povprečje vsakega stolpca. Kot lahko vidimo, je specifičnost skoraj vedno nad 0.90, kar pomeni, da se naša segmentacija skoraj v celoti nahaja znotraj referenčne regije. Če interpretiramo po formuli specifičnosti, potem to pomeni, da skoraj

Tabela 5.1: Rezultati segmentacije zoba z metodo Otsu

Slika	F1	Prec	Sen	Spec	Slika	F1	Prec	Sen	Spec
1	0.61	1.00	0.44	1.00	16	0.55	1.00	0.38	1.00
2	0.64	1.00	0.47	1.00	17	0.55	1.00	0.38	1.00
3	0.70	0.93	0.57	0.95	18	0.54	1.00	0.37	1.00
4	0.42	0.43	0.41	0.62	19	0.51	1.00	0.35	1.00
5	0.34	0.39	0.30	0.68	20	0.49	1.00	0.32	1.00
6	0.64	1.00	0.47	1.00	21	0.52	1.00	0.36	1.00
7	0.69	0.91	0.56	0.94	22	0.48	1.00	0.32	1.00
8	0.43	1.00	0.27	1.00	23	0.45	1.00	0.29	1.00
9	0.43	1.00	0.28	1.00	24	0.60	1.00	0.43	1.00
10	0.51	0.78	0.38	0.92	25	0.64	1.00	0.47	1.00
11	0.40	1.00	0.25	1.00	26	0.62	1.00	0.45	1.00
12	0.44	1.00	0.28	1.00	27	0.62	1.00	0.45	1.00
13	0.48	1.00	0.31	1.00	28	0.60	1.00	0.42	1.00
14	0.73	0.82	0.66	0.84	29	0.63	1.00	0.46	1.00
15	0.59	1.00	0.42	1.00	30	0.55	0.95	0.40	0.96
					Avg	0.55	0.94	0.40	0.96

nikoli ne vrnemo ozadja kot del regije zoba. V poglavju 3 smo omenili, da so pragovne metode zelo hitre, kar lahko potrdimo tudi v našem primeru. Proces segmentacije zoba z metodo Otsu v povprečju traja $2s$ ($2.06s \pm 0.54s$). Kot proces segmentacije je všteto vse od prvega koraka, ko spremenimo velikost vhodne slike na dimenzijo 512 x 512, pa vse do končne maske, ki predstavlja našo segmentacijo. Iz tabele lahko razberemo, da je zelo visoka tudi natančnost. Razlog je podoben kot v primeru specifičnosti. Ker je regija skoraj vedno znotraj zoba, vendar premajhna, je potem zelo velika tudi natančnost, saj klasificiramo skoraj vse slikovne točke kot del zoba. S tem je občutljivost bistveno manjša, saj je prepoznana regija zoba premajhna.

V tabeli 5.2 so rezultati segmentacije z metodo grafov. Tabela je zasta-

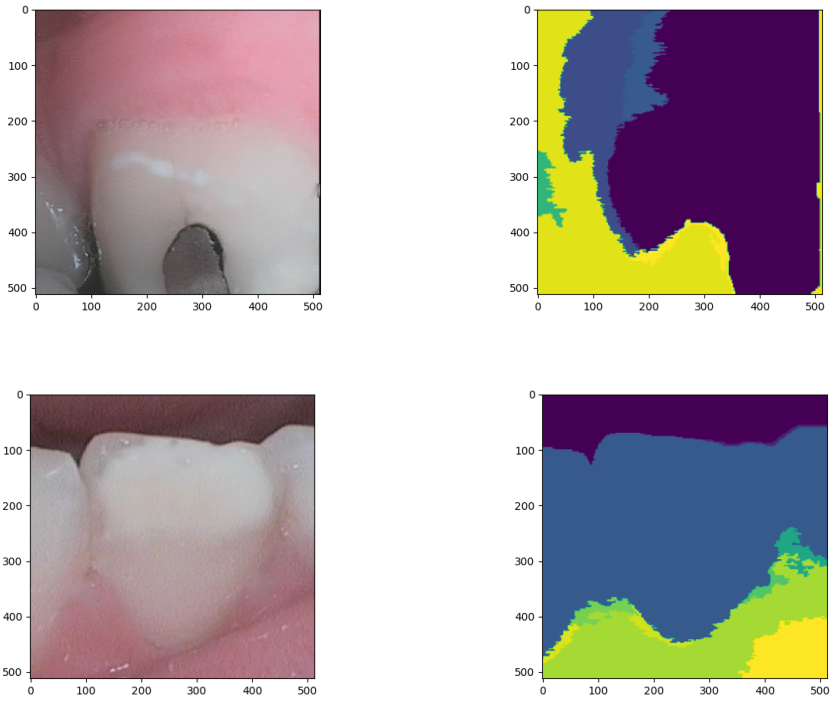
Tabela 5.2: Rezultati segmentacije zoba z metodo grafov

Slika	F1	Prec	Sen	Spec	Slika	F1	Prec	Sen	Spec
1	0.98	1.00	0.96	1.00	16	0.96	0.99	0.94	0.99
2	0.94	0.99	0.89	0.99	17	0.94	0.96	0.91	0.96
3	0.90	0.94	0.86	0.93	18	0.97	0.99	0.95	0.99
4	0.64	0.53	0.81	0.51	19	0.90	0.97	0.84	0.97
5	0.65	0.53	0.83	0.51	20	0.91	0.91	0.90	0.93
6	0.94	0.99	0.88	0.99	21	0.93	0.98	0.88	0.99
7	0.89	0.92	0.86	0.92	22	0.91	0.98	0.85	0.99
8	0.91	0.99	0.84	0.99	23	0.93	1.00	0.87	1.00
9	0.85	0.98	0.74	0.99	24	0.96	0.98	0.95	0.98
10	0.75	0.64	0.89	0.61	25	0.95	0.93	0.97	0.90
11	0.88	1.00	0.79	1.00	26	0.96	1.00	0.92	1.00
12	0.89	1.00	0.80	1.00	27	0.95	1.00	0.91	1.00
13	0.95	0.98	0.92	0.99	28	0.96	0.98	0.94	0.98
14	0.79	0.67	0.96	0.45	29	0.95	0.94	0.96	0.91
15	0.96	0.95	0.98	0.95	30	0.90	0.93	0.89	0.91
					Avg.	0.90	0.92	0.89	0.91

vljena enako kot zgornja 5.1. Iz nje lahko vidimo, da se rezultati bistveno izboljšajo. Do največje spremembe pride pri meri občutljivosti, ki se spremeni za 0.49. To pomeni, da s to metodo pravilno klasificiramo veliko več površine zoba kot s prejšnjo. Preiskovalni prostor je velik, preiskati pa ga je potrebno v celoti, kar zahteva veliko časa. To je razvidno tudi iz meritev časa izvajanja. Segmentacija z grafi potrebuje skoraj $5-$krat ($9.64s \pm 0.77s$) več časa kot prejšnja metoda. Izmerjen čas v tem primeru predstavlja algoritem z optimizacijami glede preiskovalnega prostora, katere smo omenili v poglavju 4.

Čeprav sta se vrednosti natančnosti in specifičnosti zmanjšali, je segmentacija z grafi boljša metoda. Padec vrednosti specifičnosti in natančnosti

pripisujemo temu, da pri tej metodi pokrijemo večji del zoba. V praksi to pomeni, da se verjetno na robovih regije zoba odločimo, da določena regija pripada segmentaciji, ki ima večjo vsebovanost v ozadju kot ospredju in obratno. Ta problem bi morda lahko rešili tako, da v koraku izbiranja regij, ki pripadajo segmentaciji zoba in niso popolnoma vsebovane v segmentaciji Otsu, te še dodatno obdelamo.



Slika 5.1: Primera manj uspešnih segmentacij

Za primer si poglejmo sliko 5.1. V prvi vrstici je slika z zaporedno številko 4, v drugi vrstici je slika s številko 14. Prvi stolpec predstavlja sliko pred vhodom v segmentacijo z grafi, drugi stolpec predstavlja rezultat segmentacije z grafi, kjer je vsaka regija označena z razredom. V poglavju 4, kjer opišemo, kako izbiramo kandidate regij za končno segmentacijo, vidimo, da sta oba primera posebna. Nobena regija segmentacije ni popolnoma vsebovana samo znotraj površine zoba. To pomeni, da vzamemo regijo, ki ima največji presek

Tabela 5.3: Čas izvajanja segmentacije z grafi

Velikost slike	F1	Čas[s]
512^2	0.90	24.58 ± 1.88
256^2	0.90	9.64 ± 0.77
128^2	0.37	6.16 ± 0.68
64^2	0.31	5.34 ± 0.74

s segmentacijo z metodo Otsu. V obeh primerih je problem premajhne granularnosti regij. V prvem primeru je videti, kot da je problem v izbrani meri za izračun podobnosti oziroma različnosti med vozlišči grafa, saj algoritem zazna, da sta si zob in dlesen zelo podobna. Glede na obliko regij, ki jih vidimo v sliki segmentacije, lahko sklepamo, da je v tem primeru imela zelo velik vpliv neenakomerna osvetlitev objekta. Iz slike segmentacije si lahko predstavljamo, da vidimo tri navpične pasove (rumena, modra, vijolična), ki potekajo prav tako kot svetlost slikovnih točk na sliki – iz temne regije na levi strani, do presvetle regije na desni strani. Iz stališča algoritma bi ta problem lahko rešili tako, da bi integrirali tudi znanje o segmentaciji dlesni. S tem bi algoritmu dodali kompleksnost, kar bi po vsej verjetnosti pripeljalo do novih napak in prispevalo k manj robustni segmentaciji. Boljša rešitev bi bila, da bi bile slike zajete v bolj konsistentnem okolju, v tem primeru pri boljši osvetlitvi.

V tabeli 5.3 je prikazano, kako se spreminja čas izvajanja v različnih velikostih preiskovalnega prostora in kako to vpliva na zmogljivost. Kot lahko vidimo se čas izvajanja skoraj 3-krat zmanjša iz velikosti 512^2 na 256^2, kar nam je prihranilo ogromno časa pri razvoju in testiranju. Vrednost F1 je ostala skoraj nespremenjena. Za eksperimenta z velikostjo 128^2 in 64^2 smo morali zmanjšati parameter k, saj bi drugače kot rešitev dobili samo eno regijo. Poskusili smo z različnimi vrednostmi in na koncu pristali na vrednosti 200. Kot vidimo, se je vrednost F1 močno znižala.

V poglavju 4 smo omenili, da pred segmentacijo zmanjšamo paleto barv

na 200 in pri gradnji grafa združimo sosede, ki jih povezuje povezava z vrednostjo 0. Pri merjenju časa izvajanja segmentacije smo ugotovili, da je za zmanjšanje palete potreben čas tako velik, da nimamo nobene koristi iz potencialne pohitritve. Čas izvajanja se poveča za $\approx 3s$.

5.0.2 Zmogljivost segmentacije belih madežev

V tabeli 5.4 se nahajajo rezultati segmentacije belih madežev z metodo rasti regij. Pri segmentaciji smo upoštevali podatek o tipu slike, rešitev segmentacije zob in lokalizacije območij bleščanja. Kot smo že omenili v poglavju 4, smo segmentacijo belih madežev razdelili glede na tip slike. Tip slike je v tabeli 5.4 predstavljen v stolpcu m. Če je vrednost stolpca m enaka ne, potem je slika običajna, sicer je zajeta z modro svetlobo.

Iz tabele lahko razberemo, da je zmogljivost bistveno boljša na slikah, zajetih z modro svetlobo, kot na navadnih slikah. V povprečju je vrednost F višja za 0.34. Da si bolje predstavljamo, si poglejmo sliko grafa priklica in natančnosti 5.2. Rdeče točke predstavljajo navadne slike, modre točke predstavljajo slike, zajete z modro svetlobo. Kot je razvidno iz grafa, vrednost priklica pri segmentaciji belih madežev navadnih slik zajema celoten možen razpon od 0 do 1. To pomeni, da v polovici primerov najdemo večji delež belih madežev, v ostali polovici pa jih ne najdemo. Do takšnega nihanja lahko pride zaradi več razlogov. Prvi razlog je, da smo sami označili slike s predpostavko, da vse, kar je *bele barve*, predstavlja bel madež. Tako je lahko možno, da naša referenčna segmentacija ne odraža popolnega stanja zobne površine in prisotnih belih madežev. Drugi razlog je lahko v tem, da zaradi slabše kakovosti slik ne moremo zaznati vseh lokacij belih madežev, kar bomo pokazali na primeru spodaj. Tretji razlog je, da je v nekaterih primerih bleščanje prisotno nad regijo belih madežev, kar nam predstavlja izziv, saj trenutno nimamo postopka, s katerim bi lahko ta pojav rešili.

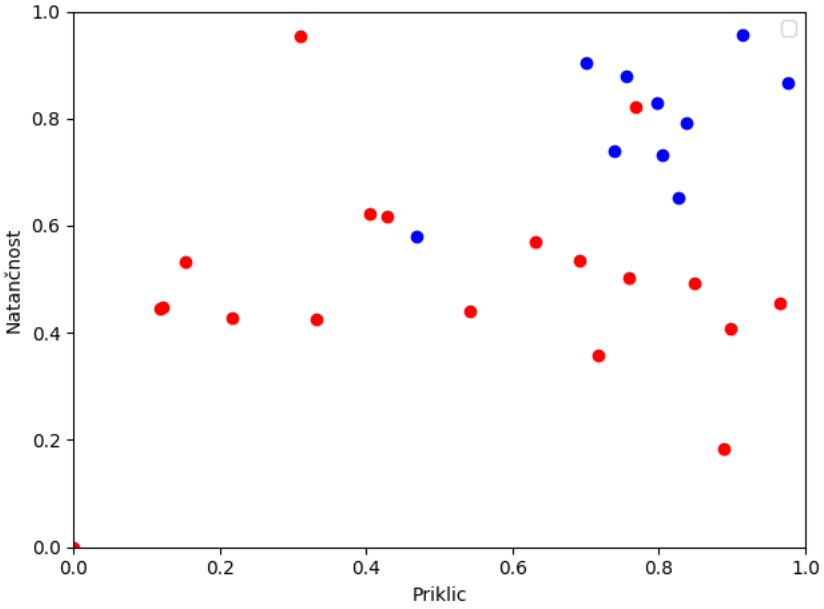
Na osi natačnosti je segmentacija belih madežev na navadnih slikah omejena predvsem na vrednosti pod 0.60. To na končni rešitvi vidimo kot območja belih madežev, ki so v postopku rasti regij preveč zrastla. Pri tem je

Tabela 5.4: Rezultati segmentacije belih madežev

Slika	F1	Prec	Sen	Spec	m	Slika	F1	Prec	Sen	Spec	m
1	0.79	0.77	0.82	1.00	ne	16	0.60	0.69	0.53	0.99	ne
2	0.92	0.98	0.87	1.00	da	17	0.60	0.76	0.50	0.99	ne
3	0.81	0.84	0.79	0.99	da	18	0.48	0.72	0.36	0.99	ne
4	0.19	0.12	0.45	0.90	ne	19	0.47	0.31	0.95	0.92	ne
5	0.19	0.12	0.45	0.92	ne	20	0.60	0.63	0.57	0.97	ne
6	0.93	0.91	0.96	1.00	da	21	0.49	0.54	0.44	0.98	ne
7	0.81	0.80	0.83	0.99	da	22	0.00	0.00	0.00	0.96	ne
8	0.24	0.15	0.53	0.89	ne	23	0.52	0.47	0.58	0.95	ne
9	0.49	0.41	0.62	0.95	ne	24	0.79	0.70	0.90	0.98	da
10	0.62	0.85	0.49	0.99	ne	25	0.74	0.74	0.74	1.00	da
11	0.30	0.89	0.18	0.99	ne	26	0.73	0.83	0.65	1.00	da
12	0.62	0.97	0.46	1.00	ne	27	0.77	0.80	0.73	0.98	da
13	0.51	0.43	0.62	0.93	ne	28	0.81	0.75	0.88	0.98	da
14	0.29	0.22	0.43	0.94	ne	29	0.54	0.58	0.58	0.97	da
15	0.37	0.33	0.43	0.98	ne	30	0.54	0.58	0.58	0.97	ne
						Avg.	0.56	0.60	0.60	0.97	

ponovno potrebno vzeti v zakup, da naša referenčna segmentacija ni popolna ter da intenziteta okoli belih madežev niha od madeža do madeža in od zoba do zoba.

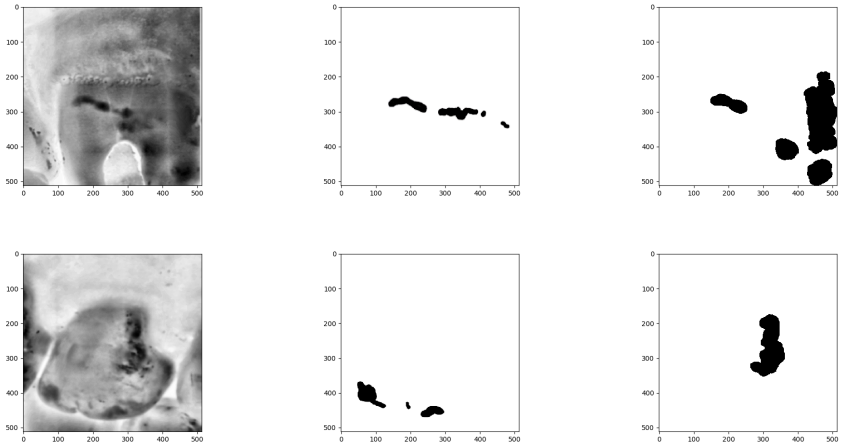
Pri segmentaciji belih madežev se na slikah, zajetih z modro svetlobo, rešitve v večini primerov nahajajo v zgornjem desnem kotu grafa 5.2, kjer predstavljajo najboljšo zmogljivost. Dobra lastnost teh slik je, da se demineralizirano območje zoba obarva izrazito drugačne barve, kar nam omogoča hitro, enostavno in robustno segmentacijo s konsistentnimi rezultati.



Slika 5.2: Zmogljivost segmentacije belih madežev na grafu priklica in natančnosti

Na sliki 5.3 imamo dva primera slabše segmentacije belega madeža. V prvi vrstici je slika z zaporedno številko 4, v drugi vrstici je slika s številko 22. Prvi stolpec predstavlja sivinsko sliko pred segmentacijo, drugi stolpec predstavlja referenčno segmentacijo, tretji stolpec pa je izhod segmentacije.

V prvi vrstici imamo primer, pri katerem smo zaznali več belih madežev kot jih imamo označenih v referenčni segmentaciji. Na sivinski sliki lahko



Slika 5.3: Primera manj uspešnih segmentacij

vidimo, da se v večini ujema rešitev segmentacije glede na sivinsko sliko. V tem primeru bi potrebovali mnenje strokovnjaka, da nam pove, katera segmentacija je bolj pravilna. Če si pogledamo barvno sliko 5.1, ugotovimo, da je zelo težko določiti, ali se v spodnjem desnem kotu dejansko nahajata bela madeža ali ne.

V drugi vrstici imamo primer, pri katerem smo zgrešili manjše madeže na spodnjem delu zoba in detektirali bel madež na območju, ki ga v referenčni segmentaciji ni. Območje, ki smo ga detektirali in ni prisotno v referenčni segmentaciji, predstavlja podobno težavo kot prejšnji primer. Iz barvne slike zoba je zelo težko določiti, ali gre za bel madež ali le za bleščanje. Najverjetneje gre za kombinacijo obeh. Težavo pri detektiranju belih madežev na spodnjem robu zoba pa gre pripisati temu, da sta madeža manjše površine in se nahajata neposredno ob robu zoba. Ker je odziv filtra, s katerim iščemo kandidate belih madežev, zelo visok na robovih zoba, nek delež kandidatov porežemo, kar pomeni, da bo bil kandidat v tem koraku odstranjen, če je to manjše območje tik ob robu zoba.

V tabeli 5.5 sta predstavljena rezultata povprečnega potrebnega časa za segmentacijo belih madežev glede na tip slike. Iz tabele je razvidno, da je

Tabela 5.5: Čas izvajanja segmentacije belih madežev glede na tip slike

Vrsta slike	F1	Čas[s]
z modro svetlobo	0.78	1.42 ± 0.45
običajna	0.44	12.12 ± 4.93

segmentacija slik z modro svetlobo skoraj za 10-krat hitrejša kot metoda rasti regij in hkrati vrača tudi bistveno boljše rezultate. Pri segmentaciji z rastjo regij se veliko časa porabi pri sami rasti, saj je potrebno za vsako začetno točko pognati algoritem, ki se širi od začetne točke skozi celotno sliko. Po potrebnem času za segmentacijo se lahko primerja s segmentacijo zob z metodo grafov.

Poglavje 6

Zaključek

V delu smo implementirali prototip sistema za avtomatsko segmentacijo slik v RGB barvnem prostoru, zajetih z intraoralno kamero Soprolife. Glavni cilj dela je bil razviti prototip sistema, ki avtomatsko oceni delež prizadete površine zobne sklenine z belimi madeži. Tak sistem bi se lahko uporabljal za spremljanje zdravljenja belih madežev. Za namen razvoja takega sistema smo implementirali segmentacijska algoritma za pridobivanje regij zob in belih madežev. Ker na spletu ne obstaja prosto dostopna podatkovna zbirka, primerna za reševanje tega problema, smo zgradili lastno podatkovno zbirko, katero smo tudi ročno označili, da je primerna za testiranje. Rezultate sistema smo ocenili in predstavili.

Kot smo ugotovili iz rezultatov evalvacije sistema, imamo še veliko prostora za izboljšave. Ena izmed pomembnejših stvari, ki jo je potrebno izboljšati, če želimo sistem še naprej razvijati, je povečanje podatkovne zbirke. Tako bi lahko poleg nizkonivojskih pristopov uporabili in eksperimentirali z bolj naprednejšimi metodami, ki za dobro delovanje potrebujejo čim večje število primerov, na primer globoke nevronske mreže. Poleg večjega števila podatkov bi bilo potrebno izboljšati tudi kakovost in ločljivost slik. V trenutni zbirki je večina slik slabše kakovosti, kar otežuje procesiranje in posledično prinaša slabše rezultate. Pri tem si lahko zastavimo vprašanje, ali je orodje Soprolife sploh primerno za zajemanje takšnih slik. Iz lastnih izkušenj vemo,

da je zajemanje kakovostnih slik z orodjem Soprolife izziv, zato bi predlagali, da bi se v primeru nadaljnjega razvoja razmislilo o alternativah, s katerimi bi lahko enostavneje zajemali slike belih madežev pod bolj nadzorovanimi pogoji in dosegli večjo kakovost slik.

Iz stališča segmentacije zob bi bilo potrebno izboljšati robustnost algoritma, da bi bil manj občutljiv na neenakomerno osvetlitev in znal razdeliti regijo v primeru prekrivanja večjega števila zob ali dlesni. Pri segmentaciji belih madežev, kot smo ugotovili iz poglavja 5, dobimo najboljše rezultate v primerih, ko je slika zajeta z modro svetlobo. V primeru nadaljnega razvoja in izboljševanja rešitve bi predlagali, da se za segmentacijo belih madežev uporablja te vrste slik. V primeru, da je mogoče zajeti dve popolnoma enaki sliki z različnima osvetljavama, bi bilo to še bolje, saj bi potem lahko kombinirali značilke obeh slik.

V nadaljevanju, v primeru povečanja števila primerov v podatkovni zbirki, bi lahko v trenutno implementiranem sistemu poskušali poiskati optimum parametrov, ki so v trenutni rešitvi nastavljeni glede na naša opažanja in testiranja. Tega bi se lahko na primer lotili z metodami linearnega programiranja.

Literatura

[1] Google cloud at rsna: engaging with the medical imaging community, https://www.blog.google/products/google-cloud/google-cloud-rsna-engaging-medical-imaging-community/, (Accessed on 08/01/2018).

[2] T. Tomaževič, R. Kosem, Ocena protokola za remineralizacijo belih madežev na zobnih ploskvah po snetju nesnemnega ortodontskega aparata: specialistična naloga.
URL https://books.google.si/books?id=9c59oAEACAAJ

[3] M. Sonka, V. Hlavac, R. Boyle, Image processing, analysis, and machine vision, Cengage Learning, 2014.

[4] D. L. Pham, C. Xu, J. L. Prince, Current methods in medical image segmentation, Annual review of biomedical engineering 2 (1) (2000) 315–337.

[5] S. D. Na, G. Lee, J. H. Lee, M. N. Kim, Individual tooth region segmentation using modified watershed algorithm with morphological characteristic, Bio-medical materials and engineering 24 (6) (2014) 3303–3309.

[6] Y. Wang, Y. Dai, J. Xue, B. Liu, C. Ma, Y. Gao, Research of segmentation method on color image of lingwu long jujubes based on the maximum entropy, EURASIP Journal on Image and Video Processing 2017 (1) (2017) 34. doi:10.1186/s13640-017-0182-5.
URL https://doi.org/10.1186/s13640-017-0182-5

[7] D. J. Bora, A. K. Gupta, A new efficient color image segmentation approach based on combination of histogram equalization with watershed algorithm, International Journal of Computer Sciences and Engineering 4 (6) (2016) 156–167.

[8] P. Taler, K. Sabo, Color image segmentation based on intensity and hue clustering-a comparison of ls and lad approaches, Croatian Operational Research Review 5 (2) (2014) 375–385.

[9] L. C. Lulio, V. L. Belini, M. L. Tronco, A. J. V. Porto, Jseg algorithm and statistical image segmentation techniques for quantization of fruits.

[10] C. K. Chama, S. Mukhopadhyay, P. K. Biswas, A. K. Dhara, M. K. Madaiah, N. Khandelwal, Automated lung field segmentation in ct images using mean shift clustering and geometrical features, in: Medical Imaging 2013: Computer-Aided Diagnosis, Vol. 8670, International Society for Optics and Photonics, 2013, p. 867032.

[11] A. Şengür, Y. Guo, Color texture image segmentation based on neutrosophic set and wavelet transformation, Computer Vision and Image Understanding 115 (2011) 1134–1144.

[12] T. Eckhard, E. M. Valero, J. L. Nieves, Labial teeth and gingiva color image segmentation for gingival health-state assessment, in: Conference on Colour in Graphics, Imaging, and Vision, Vol. 2012, Society for Imaging Science and Technology, 2012, pp. 102–107.

[13] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: Proc. 8th Int'l Conf. Computer Vision, Vol. 2, 2001, pp. 416–423.

[14] S. Mikes, M. Haindl, G. Scarpa, R. Gaetano, Benchmarking of remote sensing segmentation methods, Selected Topics in Applied Earth Obser-

vations and Remote Sensing, IEEE Journal of 8 (5) (2015) 2240–2248. doi:10.1109/JSTARS.2015.2416656.

[15] B. Sathya, R. Manavalan, Image segmentation by clustering methods: performance analysis, International Journal of Computer Applications 29 (11).

[16] F. Ge, S. Wang, T. Liu, New benchmark for image segmentation evaluation, Journal of Electronic Imaging 16 (3) (2007) 033011.

[17] Acteon — n°1 for dental equipment — dental products — equipment, imaging, pharma, https://www.acteongroup.com/en/, (Accessed on 09/12/2018).

[18] K. Angelino, P. Shah, D. A. Edlund, M. Mohit, G. Yauney, Clinical validation and assessment of a modular fluorescent imaging system and algorithm for rapid detection and quantification of dental plaque, BMC oral health 17 (1) (2017) 162.

[19] Gimp - gnu image manipulation program, https://www.gimp.org/, (Accessed on 08/06/2018).

[20] Cielab color space - wikipedia, https://en.wikipedia.org/wiki/CIELAB_color_space, (Accessed on 08/06/2018).

[21] Cieluv - wikipedia, https://en.wikipedia.org/wiki/CIELUV, (Accessed on 08/06/2018).

[22] C. Ozturk, E. Hancer, D. Karaboga, Color image quantization: a short review and an application with artificial bee colony algorithm, Informatica 25 (3) (2014) 485–503.

[23] Opencv: Histograms - 2: Histogram equalization, https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html, (Accessed on 08/15/2018).

[24] S. K. Warfield, K. H. Zou, W. M. Wells, Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation, IEEE transactions on medical imaging 23 (7) (2004) 903–921.

[25] Sørensen–dice coefficient - wikipedia, https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient, (Accessed on 08/26/2018).

[26] M. Beneš, B. Zitová, Performance evaluation of image segmentation algorithms on microscopic image data, Journal of microscopy 257 (1) (2015) 65–85.

[27] H. Lange, Automatic glare removal in reflectance imagery of the uterine cervix, in: Medical Imaging 2005: Image Processing, Vol. 5747, International Society for Optics and Photonics, 2005, pp. 2183–2193.

[28] D. K. Srivastava, Efficient fruit defect detection and glare removal algorithm by anisotropic diffusion and 2d gabor filter, International Journal of Engineering Science & Advanced Technology 2 (2) (2012) 352–357.

[29] Opencv: Computational photography, https://docs.opencv.org/3.4.1/d1/d0d/group__photo.html#gaedd30dfa0214fec4c88138b51d678085, (Accessed on 07/22/2018).

[30] K. Zuiderveld, Graphics gems iv, Academic Press Professional, Inc., San Diego, CA, USA, 1994, Ch. Contrast Limited Adaptive Histogram Equalization, pp. 474–485.
URL http://dl.acm.org/citation.cfm?id=180895.180940

[31] N. Otsu, A threshold selection method from gray-level histograms, IEEE transactions on systems, man, and cybernetics 9 (1) (1979) 62–66.

[32] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation, International journal of computer vision 59 (2) (2004) 167–181.

[33] R. C. Prim, Shortest connection networks and some generalizations, Bell system technical journal 36 (6) (1957) 1389–1401.

[34] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings of the American Mathematical society 7 (1) (1956) 48–50.

[35] Disjoint-set data structure - wikipedia, `https://en.wikipedia.org/wiki/Disjoint-set_data_structure`, (Accessed on 09/06/2018).

[36] Disjoint-set data structure - wikipedia, `https://en.wikipedia.org/wiki/Disjoint-set_data_structure`, (Accessed on 08/22/2018).

[37] R. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification SMC-3 (1973) 610–621.

Slike

Tabele

Koda