Segmentacija slika na točke, linije i rubove

Zaninović, Jelena

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Jelena Zaninović

SEGMENTACIJA SLIKA NA TOčKE, LINIJE I RUBOVE

Diplomski rad

Voditelj rada:
prof. dr. sc. Nela Bosner

Zagreb, ožujak, 2024

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Mojoj obitelji - za vašu beskrajnu podršku, ljubav i strpljenje.
Prijateljima i kolegama - za uljepšavanje mojih studentskih dana.
Mojoj mentorici - za Vaše razumijevanje, savjete i svu Vašu pomoć.
Mojim asistentima i profesorima - za svo znanje koje ste mi podarili.
Od srca vam hvala.

Sadržaj

Uvod

U ovom radu ćemo se baviti problemom segmentacije digitalnih slika na točke, linije i rubove. Segmentacija slike bavi se njenom podjelom na sastavne dijelove ili objekte, a detaljnost te podjele ovisi o samoj primjeni slike. Segmentacija netrivijalnih slika jedan je od najtežih zadataka kod procesiranja slika. Metode za detektiranje izoliranih točaka, linija ili rubova temelje se na otkrivanju oštre, lokalne promjene intenziteta. Pikseli ruba su pikseli kod kojih dolazi do nagle promjene intenziteta funkcije slike, a sam rub, ili bolje rečeno segment ruba, je skup povezanih piksela ruba. Takve nagle promjene funkcije slike otkrivaju se proučavanjem njene prve i druge derivacije, odnosno njihovih aproksimacija.

Ovaj rad je organiziran na sljedeći način. U početnom poglavlju ćemo dati kratki uvod u područje digitalne obrade slika uz kratke crte iz povijesti i nekoliko primjera iz primjene. Drugim i trećim poglavljem postavljamo temelj za kasnija poglavlja: u drugom poglavlju definiramo i predstavljamo model digitalne slike, zatim opisujemo odnose između piksela poput susjednosti i povezanosti; u trećem poglavlju objašnjavamo prostorno filtriranje slike i predstavljamo osnovne filtere za izglađivanje i izoštravanje, odnosno detekciju ruba.

Četvrto poglavlje se bavi problemima detekcije izoliranih točaka i pravaca. Predstavljene metode se temelje na Laplaceovom filteru i odsjecanju. U petom poglavlju predstavljamo tri algoritma za detekciju ruba: gradijentnu metodu, Marr-Hildrethin i Cannyjev algoritam. Gradijentna metoda detektira rubove na temelju aproksimacije magnitude; Marr-Hildrethin algoritam se oslanja na korištenje Laplaciana Gaussijana ili ”LoG” operatora; Cannyjev algoritam je najkompleksniji od tri algoritma i u svojim koracima koristi potiskivanje nemaksimuma i histerezno odsjecanje, a za razliku od ostalih metoda radi s gradijentnom slikom i magnitude i kuta.

Kako su algoritmi detekcije ruba obično upareni s algoritmima povezivanja segmenata ruba, u šestom poglavlju smo predstavili tri takva algoritma. Među ta tri algoritma najznačajniji je Houghov algoritam. Konačno, u sedmom poglavlju smo predstavili rezulate implementacije algoritama u programu *Octave*, s fokusom na algoritme detekcije.

Poglavlje 1

Motivacija

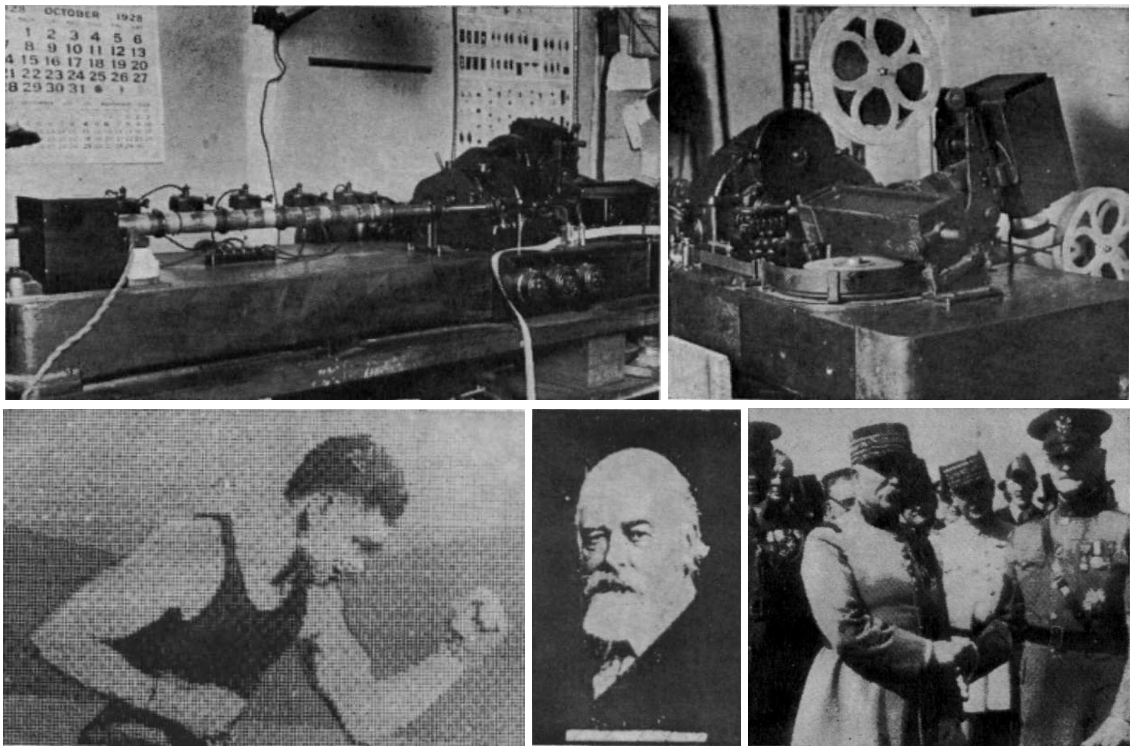
U ovom ćemo poglavlju kratko predstaviti područje digitalne obrade slika - navest ćemo osnovne crte iz povijesti područja, ilustrirati širinu njegove primjene te proći kroz radni slijed, odnosno osnovne korake obrade slike kako bismo stekli bolje razumijevanje uloge segmentacije u ukupnom procesu obrade.

1.1 Kratka povijest obrade slika

Jedna od najranijih primjena digitalnih slika bila je u novinskoj industriji kad su se prve takve slike slale podvodnim kanalima između Londona i New Yorka. U početku je slikama trebalo i po tjedan dana da stignu na svoju destinaciju, a to se vrijeme srezalo na svega nekoliko sati uvođenjem Bartlane kabelskog sustava za prijenos slika [1] u ranim 1920-tima. U početku je bilo poteškoća s odabirom i distribucijom razina intenziteta, odnosno nijansi sive boje kojima su reprezentirane fotografije. U manje od 10 godina broj nijansi je skočio s 5 na 15, a uslijedila su i druga unaprijeđenja sustava koja su poboljšala kvalitetu slika i brzinu prijenosa.

Premda su slike digitalizirane, ovaj način obrade svejedno ne smatramo *digitalnom* obradom jer u proces nisu uključena računala. Područje digitalne obrade slika usko je povezano s razvojem digitalnog računala, a budući da slike zahtijevaju dosta pohrane i računalne snage, razvoj područja će pratiti razvoj tehnologija za pohranu, prikaz i prijenos podataka. Prva računala dovoljne snage za digitalnu obradu slika javila su se u ranim 1960-tima, a pioniri tog područja su većinski radili u svemirskim programima. 1959-te je sovjetska letjelica Luna 3 na Zemlju poslala prve slike dalje strane Mjeseca. *Jet Propulsion Laboratory* je 1964.-te primio prve slike Mjeseca snimljene SAD-ovom svemirskom letjelicom

[1] Naziv je dobio po svojim izumiteljima, Harryju G. Bartholomewu i Maynardu D. McFarlaneu. Više informacija o sustavu može se pronaći u [23] i [29].

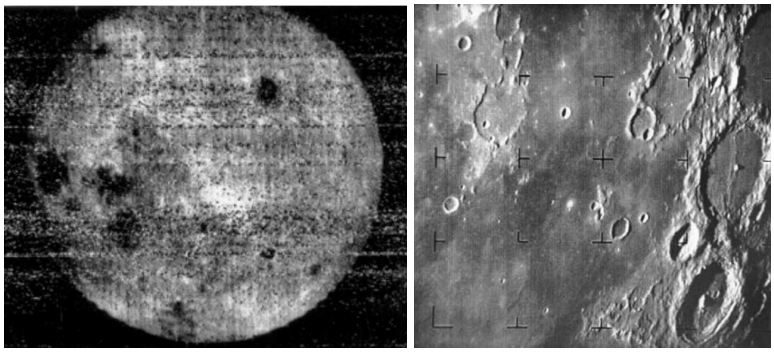


Slika 1.1: Na slikama u gornjem redu vidimo Bartlane odašiljač, kojim su fotografije kodirane za prijenos kabelima, i prijamnik, koji je prevodio električne impulse u sliku. U donjem redu su slike poslane Bartlaneovim sustavom, redom iz 1921., 1922. i 1929., s tim da je za prve dvije korišteno 5 razina intenziteta, a za treću 15. Slike su preuzete iz [23].

(*Ranger 7*) te su razvijene nove tehnike poboljšanja slika i popravka izobličenja nastalih pri snimanju. Tehnike snimanja i obrade razvijene za ovu misiju služile su kao temelj za daljnji razvoj ovog područja i za metode korištene u Mariner i Apollo misijama.

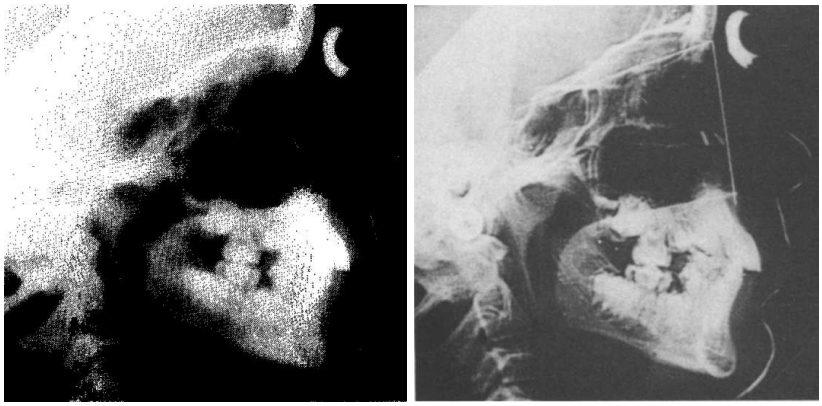
U kasnim 1960-tima i ranim 1970-tima, nove tehnike su se razvijale za primjenu u medicini, astronomiji i geološkim istraživanjima. Izum računalne tomografije[2] ili CAT/CT skenova jedan je od najvažnijih događaja za primjenu digitalne obrade slika u medicini. Osim u medicini, digitalna obrada slika koristila se za poboljšanje, restauraciju i prikaz snimki u arheologiji, biologiji, geografiji, fizici, telekomunikaciji, industriji i tako dalje, s metodama prilagođenima raznim načinima generiranja slika - od satelitskih snimki, ras-

[2]Proces u kojem prsten detektora okruži objekt koji želimo snimiti, a izvor rendgenskog značenja koji je koncentričan s detektorom kruži oko objekta, s jedne strane ispuštajući rendgenske zrake i skupljajući ih s druge. Pomoću očitanih signala se konstruiraju presjeci kroz objekt, a niz presjeka se kombinira u 3D prikaz unutrašnjosti objekta.



Slika 1.2: Slijeva je prva slika dalje strane mjeseca - 1959. ju je snimila i na Zemlju poslala sovjetska letjelica Luna 3. Zdesna vidimo prvu od 4316 slika mjeseca koje je 1964. snimila američka sonda Ranger 7. Slike su preuzete iz [2] i [35].

ponskog snimanja i termografije do ultrazvuka, magnetske rezonance, računalno generiranih slika i fotografije.



Slika 1.3: Rendgenska snimka lubanje i lica (kefalogram) prije i poslije digitalne obrade. Slike su preuzete iz [11].

Fokus obrade je prije bio na prilagodbi slika za ljudsko oko i interpretaciju, no pažnja se sve više prebacuje na prilagodbu slika za strojnu interpretaciju i obradu. Često se iz slika izvlače informacije koje su ljudima teže za interpretirati, poput statističkog momenta, koeficijenata Fourierove pretvorbe, višedimenzionalne mjere udaljenosti i slično. Strojna obrada i vid nude širok spektar primjene - nadzor proizvodnje, upravljanje robotima i vozilima, automatizitana detekcija anomalija i defekata, analiza medicinskih snimki, itd. Širenje infrastrukture, porast snage računala i smanjenje troškova računanja omogućili

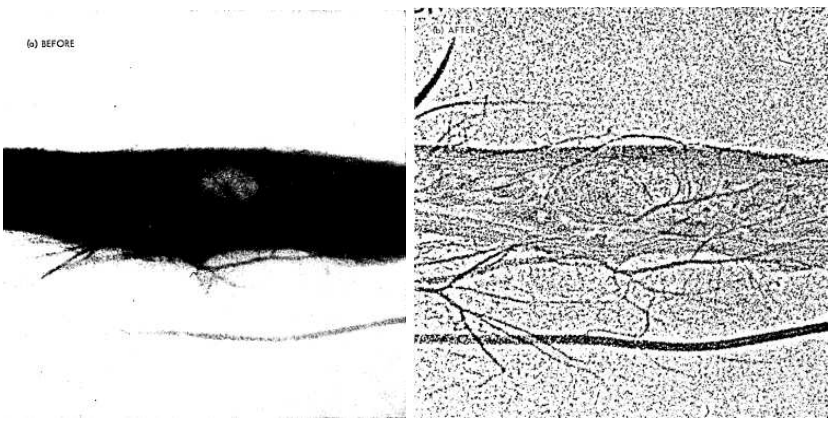
su području obrade slika kontinuiran razvoj te napredak klasičnih (manualna obrada i modifikacija) i novijih (duboko učenje) metoda.

1.2 Primjeri iz primjene

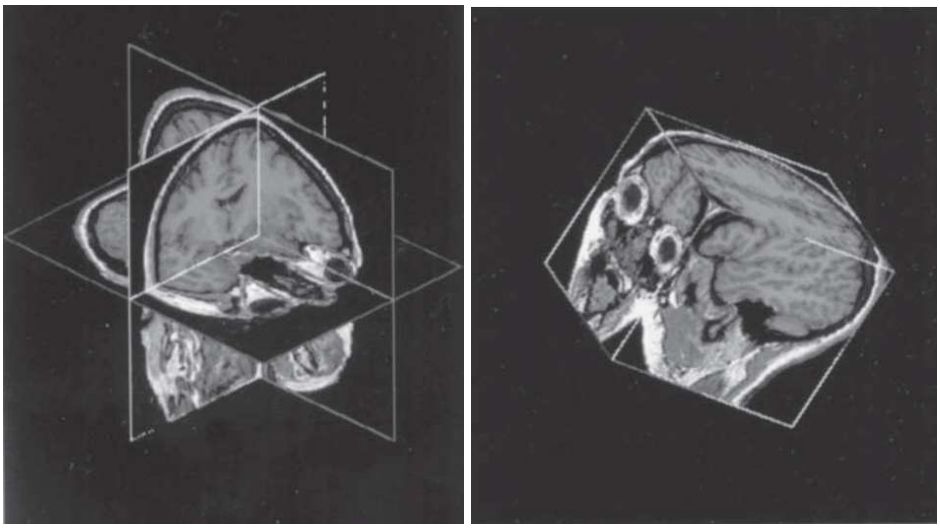
Ranije smo spomenuli da je digitalna obrada slika primjenjiva u brojnim disciplinama; u današnje digitalno doba je teško precijeniti njenu ulogu koliko u znanosti i industriji, toliko u svakodnevnom životu. U ovoj sekciji ćemo vrlo kratko navesti nekoliko primjera iz primjene, koje ćemo okvirno podijeliti po područjima na primjere iz medicine, industrije i znanosti (premda, naravno, u toj podjeli postoji dosta preklapanja).

Medicina

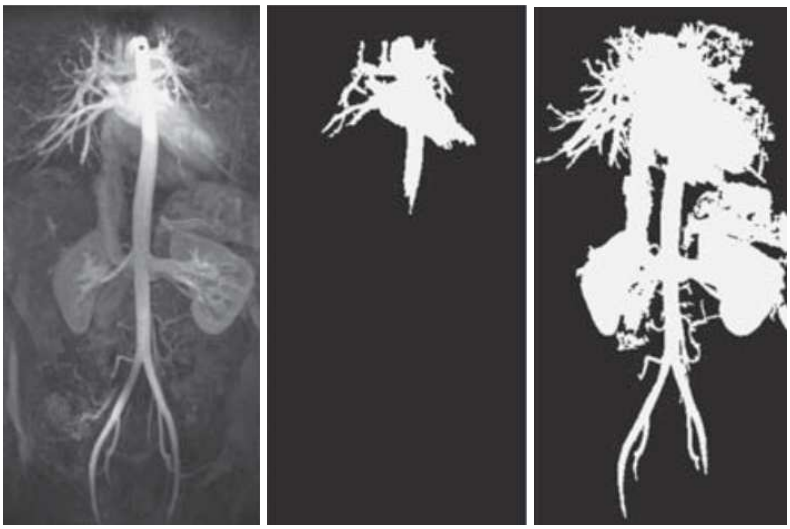
U medicini se slike generiraju raznim tehnikama: rendgenskim zračenjem, gama zračenjem, računalnom tomografijom, magnetskom rezonancom, itd. Neke od tih metoda su notorno osjetljive te su rezultantne slike često šumovite, stoga je potrebno očistiti i podesiti snimke kako bi bile jednostavnije za analizirati. Neke metode zahtijevaju rekonstruiranje konačne slike iz snimki, primjerice 3D slike iz niza 2D presjeka kroz objekt. Ponekad sa snimke želimo izdvojiti i bolje promotriti neko područje od interesa pa ga je korisno segmentirati sa slike - segmentacija nam osim toga može omogućiti analizu objekta ili objekata na slici (npr. brojnost, površina/volumen, sferičnost, raspored), što omogućuje statističku analizu, izgradnju modela, itd.



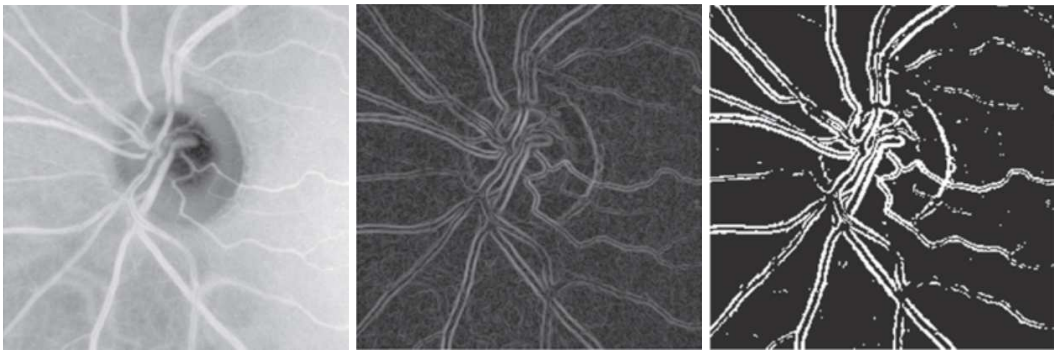
Slika 1.4: Rendgenska snimka ruke s tumorom i njena segmentacija prigušivanjem niskih frekvencija. Slike su preuzete iz [4].



Slika 1.5: Ortogonalni presjeci slike 3D volumena prikazan kao presjek ortogonalnih ravnina (lijevo) i kao kubični volumen (desno). Slike su preuzete iz [10].



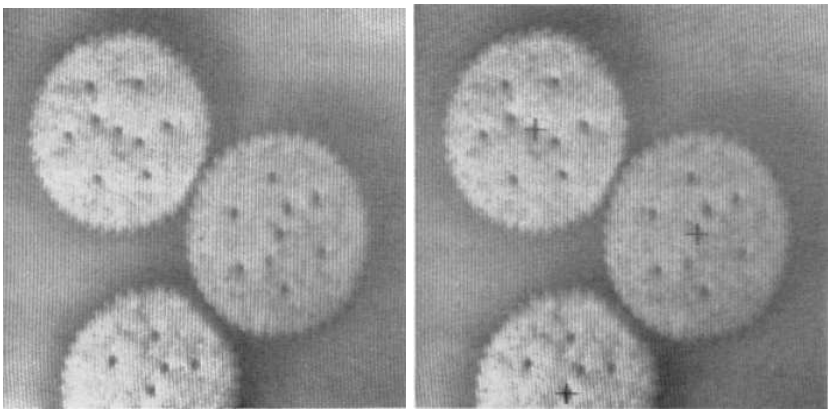
Slika 1.6: MR angiografija (metoda generiranja slika za vizualizaciju krvnih žila u tijelu) aorte i krvnih žila i njena segmentacija metodom rastućih regija. Slike su preuzete iz [10].



Slika 1.7: Žilice u mrežnici i postupak segmentacije žilica sa slike pomoću Sobelovog operatora i odsjecanja (predstavljeni kasnije u radu). Slike su preuzete iz [10].

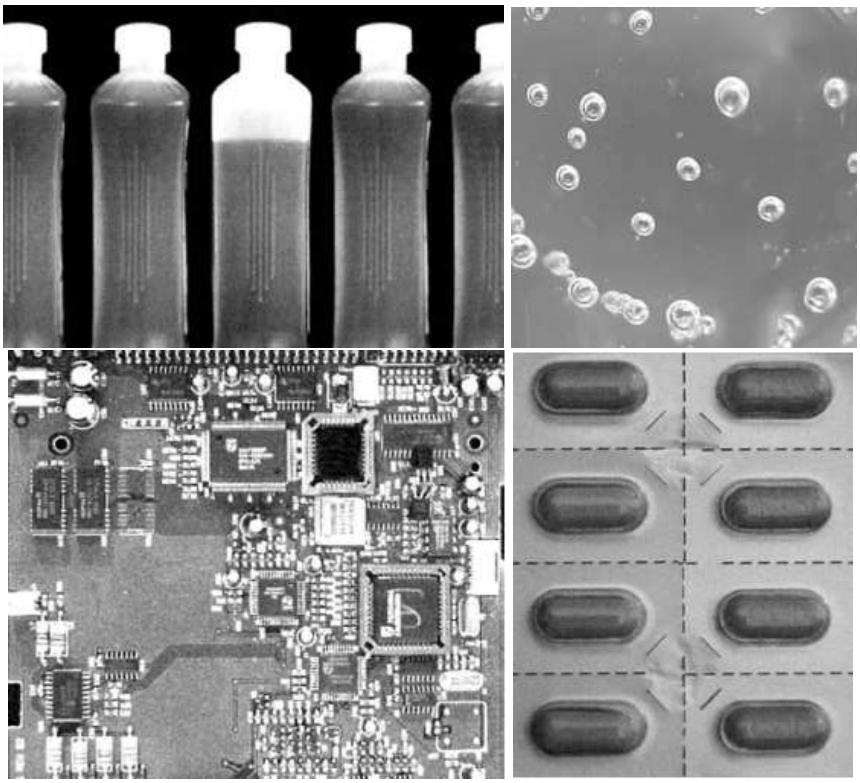
Industrija

U industriji se pomoću digitalne obrade slika može provjeriti kvaliteta proizvoda - npr. u tekstilnoj industriji se pomoću strojnog učenja može automatizirati detekcija oštećenja tkanine i drugih materijala, a u prehrambenoj industriji se može analizirati veličina, oblik ili struktura prehrambenog proizvoda (npr. mjehurići u kruhu, pukotine na keksima, klice na povrću). Ovisno o samoj industriji, načini generiranja (pa i obrade) slika drastično se razlikuju: dok mikroprocesore možemo analizirati pod običnim mikroskopom, za detekciju poroznosti lopatice turbine će nam trebati skroz drukčija metoda, poput rendgenskog zračenja.

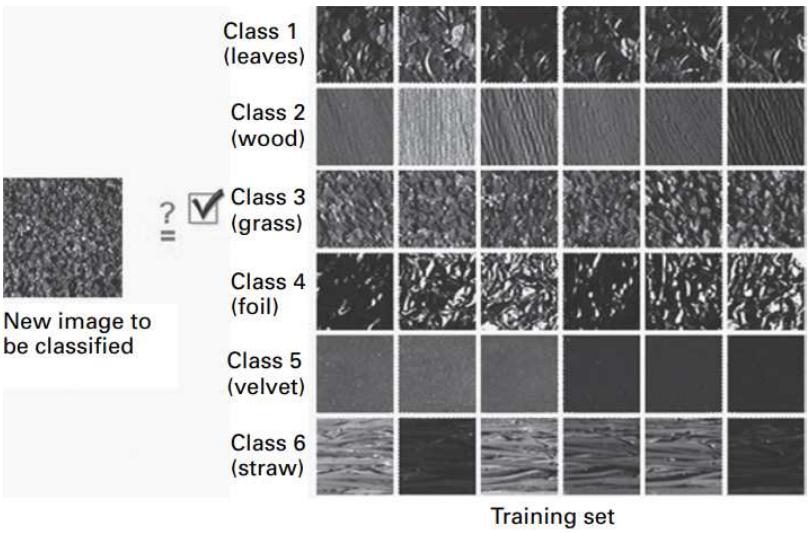


Slika 1.8: Detekcija centra keksa pomoću Houghove pretvorbe. Slike su preuzete iz [8].

Slika 1.9: Detekcija specifičnih vijaka segmentacijom po sličnosti u intenzitetu (nijansama sive) objekata. Slike su preuzete iz [10].



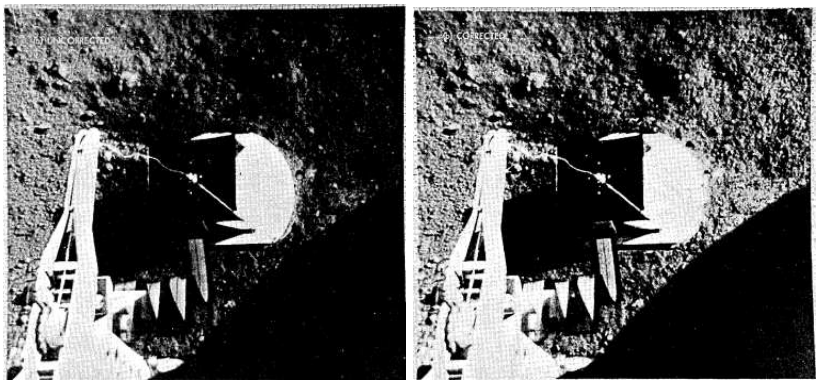
Slika 1.10: Primjeri proizvoda koji se često pregledavaju za defekte pomoću digitalne obrade. Redom su prikazane boce, prozirna plastika (s mjehurićima), upravljač pločice sklopa i paket tableta. Slike su preuzete iz [35].



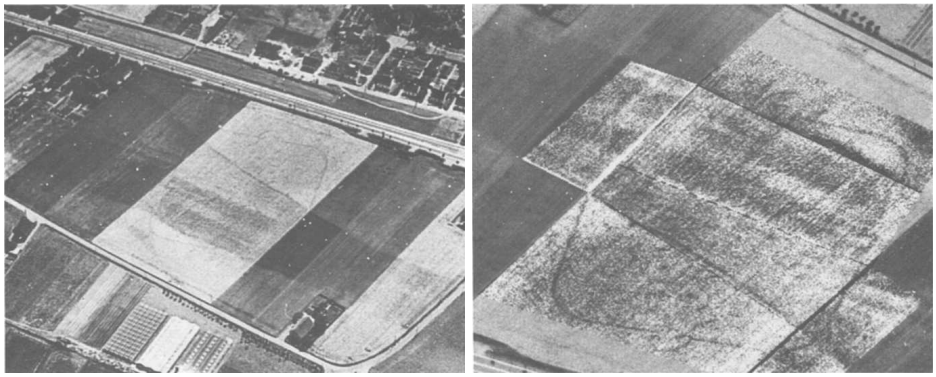
Slika 1.11: Klasifikacija teksture pomoću strojnog učenja. Slika je preuzeta iz [10].

Znanost

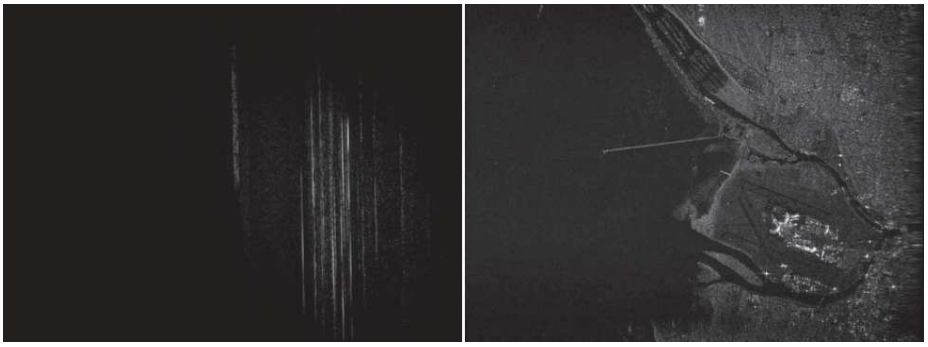
Od molekularnih biologa koji detektiraju jezgre stanica do astronoma koji klasificiraju zvijezde i galaktike, od restoracije starih povijesnih dokumenata do generiranja 3D modela za simulaciju tvari, od analize površine Mjeseca do pronalaska arheoloških lokaliteta na Zemlji, digitalna obrada igra vrijednu ulogu kako u prirodnim, tako i u humanističkim znanostima.



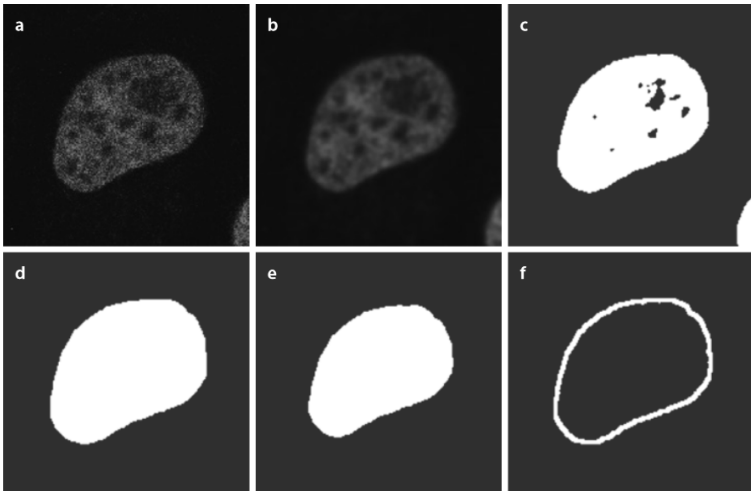
Slika 1.12: Korekcija neuniformnog odziva kamere. Slike su preuzete iz [4].



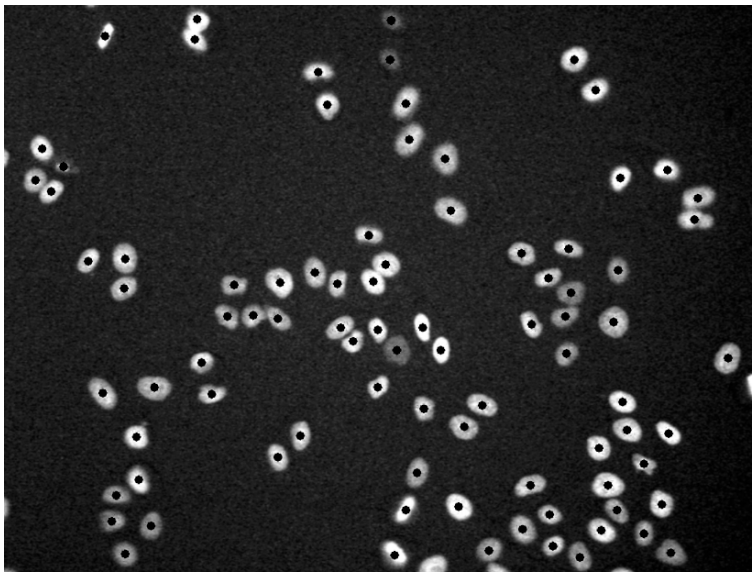
Slika 1.13: Detekcija arheološkog lokaliteta poboljšanjem zračne snimke. Slike su preuzete iz [32].



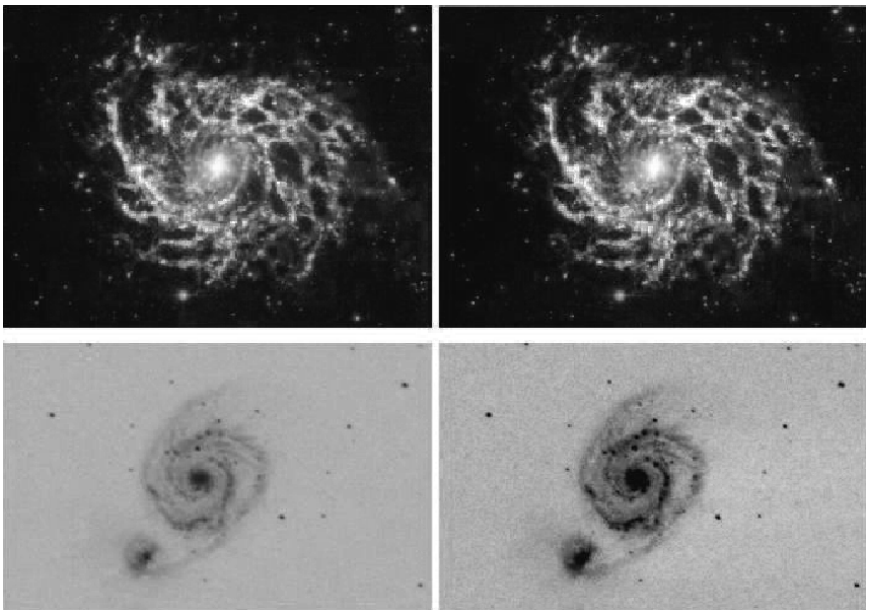
Slika 1.14: Radarska snimka komprimirana u frekvencijskoj domeni (lijevo) i njena rekonstrukcija u prostornoj domeni (desno). Slike su preuzete iz [34].



Slika 1.15: Segmentacija ruba stanice. Slike su preuzete iz [24].

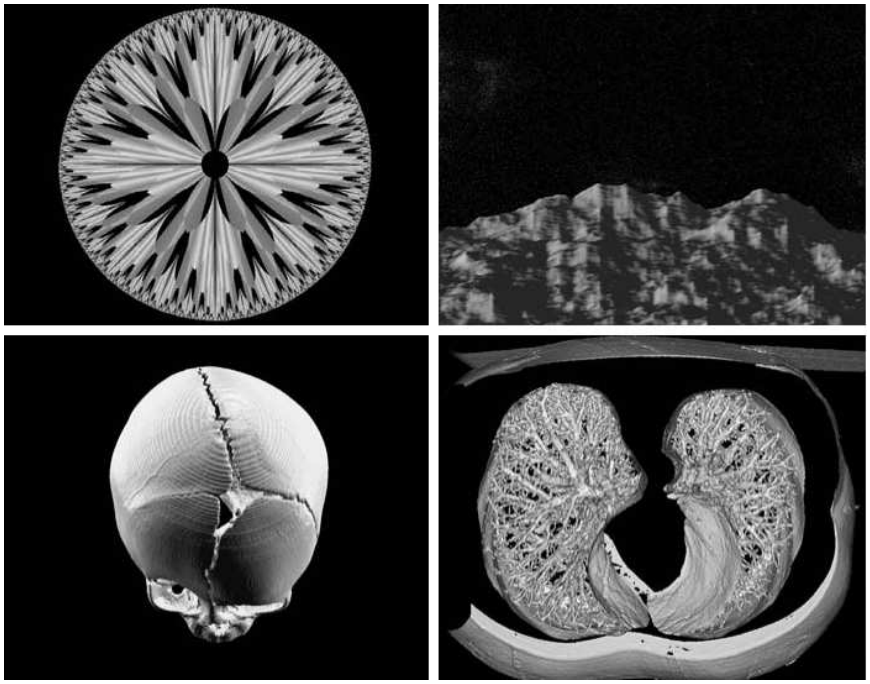


Slika 1.16: Detekcija stanica pronalaženjem njihovih jezgri pomoću LoG operatora (predstavljen kasnije u radu). Slike su preuzete iz [20].



Slika 1.17: Snimka galaktike i njene pripremne obrade za potrebu sustava za automatsku klasifikaciju. Slika je preuzeta iz [17].

Slika 1.18: Primjeri restaurirane slike i dokumenta. Slike su preuzete iz [12] i [19].



Slika 1.19: U gornjem redu su računalno generirani fraktali, a u donjem redu pripadni 3D modeli. Računalno generirane slike su korisne u simulacijama i modeliranju. Slike su preuzete iz [35].

1.3 Digitalna obrada slika

Digitalna obrada slika je procesiranje digitalnih slika pomoću digitalnog računala. Kasnije ćemo preciznije definirati digitalne slike, ali zasad je dovoljno da ih razumijemo kao diskretizacije kontinuiranih slika čija je diskretizacija rezultat metode generiranja slike i njenom pohranom na računalo. Glavni ciljevi obrade slika su poboljšanje slikovnih podataka za ljudsku i strojnu interpretaciju, kao i ekonomičnija pohrana i prijenos slika.

Procese digitalne obrade slika možemo okvirno podijeliti na tri razine: nisku, srednju i visoku. Procesi niske razine imaju slike kao ulaz i izlaz, uglavnom koriste jednostavne operatore te se većinski bave čišćenjem šuma, pojačavanjem kontrasta, izoštravanjem slike, popravljanjem osvjetljenja i slično. Procesi srednje razine uključuju segmentaciju (particiju slike na regije ili objekte), opis tih objekata za ljudsku ili strojnu obradu i interpretaciju, te klasifikaciju individualnih objekata. Za razliku od procesa niske razine, ovaj put su ulazni podatci slike, a izlazni su ponekad slike, a ponekad atributi izvučeni iz ulaznih slika (npr. rubovi i granice, individualni objekti, itd.). Naposljetku, procesi visoke razine zahtijevaju razumijevanje i interpretaciju ulaznih podataka i može se reći da imaju kognitivnu ulogu u ukupnoj obradi i analizi slike. U ovom radu nas zanimaju pretežno procesi srednje, ali i niske razine - naime, kako bi segmentacija bila što uspješnija, sa slika je ponekad potrebno očistiti šum, pa ćemo se kratko osvrnuti i na prostorne filtere kojima se to postiže.

Različiti sustavi generiranja slike, različiti ciljevi u obradi i različite vrste slika zahtijevaju drukčije pristupe u obradi, zato će radni tok svakog zadatka izgledati drukčije. Međutim, ipak ćemo navesti okvirni radni tok ili barem moguće korake radnog toka kako bismo stekli bolje razumijevanje ukupnog procesa i uloge segmentacije u njemu. Potrebno je naglasiti da tok nije potpuno linearan - moguće je kombinirati izlaze, formirati petlje koje kombiniraju više koraka i pomoću informacija dobivenih u jednom od kasnijih koraka podesiti koeficijente ranijeg koraka.

Obradu slike možemo podijeliti na sljedeće korake:

1. **Akvizicija slike**: generiranje digitalne slike metodama poput fotografije, rendgendskog snimanja, snimanja raspona, ultrazvukom, itd. Ključni procesi su *uzorkovanje* i *kvantizacija* slike.
2. **Filtriranje i poboljšanje slike**: provodi se u prostornoj ili frekvencijskoj domeni. Cilj je očistiti šum, popraviti svojstva poput kontrasta, svjetline i oštrine ili naglasiti neke značajke slike. Ukratko, generirana slika se prilagođava za obradu i primjenu.
3. **Restauracija i rekonstrukcija**: odstranjivanje nekih vrsta šuma i oštećenja te rekonstrukcije slike ovisne o metodi akvizicije. Za razliku od prethodnog, ovaj korak

se temelji na matematičkim i vjerojatnosnim modelima degradacije umjesto na osobnoj procjeni.

4. **Obrada slike u boji**: korekcije svojstava poput zasićenosti i nijansi, posebne metode komprimiranja i segmentiranja, itd. Ovisno o problemu i algoritmu, ovaj se korak može rascjepkati ili obaviti prije ranijih koraka.
5. **Multirezolucijsko procesiranje i elementarni valovi**: elementarni valovi su osnova reprezentacije slike u različitim rezolucijama, što može biti korisno u komprimiranju slike i za konstrukciju piramidalnog[3] prikaza slike.
6. **Kompresija**: uklanjanje suvišnih informacija uz čuvanje vjernosti slike, pojeftinjenje pohrane i vremena obrade i prijenosa te skrivanje vodenog žiga.
7. **Morfološka obrada**: obrada na temelju regija i oblika regija u razne svrhe, pr. uklanjanje šuma, traženje kostura, zadebljanje, itd.
8. **Segmentacija**: particija slike na objekte i regije, izdvajanje objekata od interesa sa slike, označavanje međusobno različitih objekata na slici, itd. Rezultati segmentacije često su ulazni podatci algoritama za prepoznavanje i klasifikaciju.
9. **Reprezentacija i opis slike**: razni načini predstavljanja segmentiranih objekata i regija te njihovih svojstava, opis karakteristika čitave slike, označavanje objekata.
10. **Prepoznavanje objekata**: uspoređivanje i klasifikacija objekata na temelju definiranih deskriptora.

Obratimo pažnju na položaj segmentacije u radnom toku: nakon što se slika generira i obavi pripremna obrada (niska razina) slijedi segmentacija, kojom iz slike izdvojimo objekte od interesa (srednja razina), nakon čega ih pripremimo za analizu te konačno iz njih odabranom metodom izvučemo traženu informaciju (visoka razina). Osim toga, primijetimo da je segmentacija jedan od koraka koji na izlazu može vratiti sliku ili drukčiju vrstu podataka - u algoritmima koje ćemo mi obraditi dalje u radu, izlaz će ipak činiti samo slike, premda ne nužno u istom formatu kao ulazna (npr. sa sivom ulaznom slikom možemo dobiti crno-bijelu (”binarnu”) sliku, slici se može promijeniti veličina i sl.).

[3] Piramidalni prikaz slike je koristan u strojnom vidu - ista slika se ponavlja na svakoj razini piramide, s tim da su na nižim razinama slike viših rezolucija, a na višim razinama slike nižih rezolucija. Postoji više vrsta, npr. Gaussova piramida, piramida poduzorkovanja, itd. Detaljniji opis može se pronaći u [35].

Poglavlje 2

Osnove digitalne slike

2.1 Model digitalne slike

Definicija 2.1.1. *Neka su A, B i C proizvoljni skupovi. Slika je funkcija* $f : A \times B \to C$*, gdje je* $A \times B$ *njena prostorna domena, a C njen interval intenziteta. Ako A, B i C imaju konačne, diskretne vrijednosti, kažemo da funkcija predstavlja* ***digitalnu sliku****.*

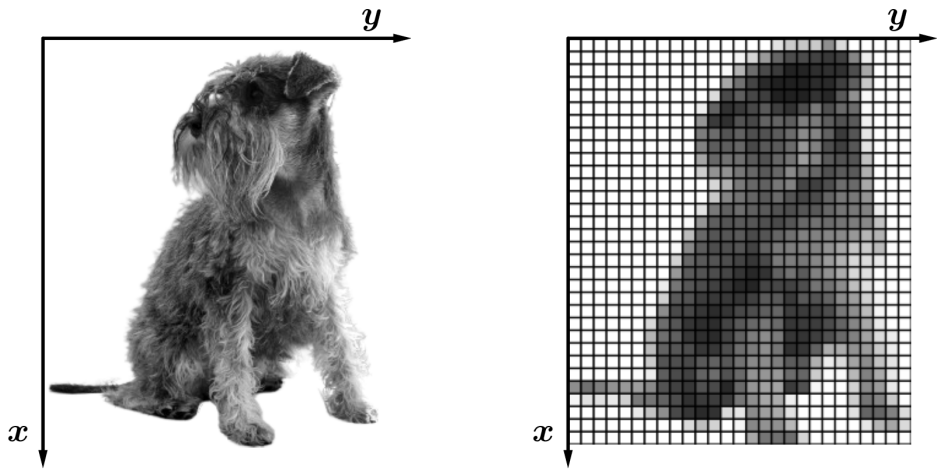
Element slike $(x, y, f(x, y))$ s prostornim koordinatama (x, y) i intenzitetom $f(x, y)$ zovemo **piksel** (skraćeno od "*picture element*"). U daljnjem ćemo se tekstu povremeno referirati na piksel po njegovim koordinatama.

Digitalna slika se dobije iz kontinuirane slike pomoću **uzorkovanja**, koje diskretizira domenu, i **kvantizacije**, koja diskretizira kodomenu. Uzorkovanje je definirano načinom na koji senzori biraju mrežu točaka koje će snimiti. Međutim, kako su intenziteti snimljenih točaka i dalje u kontinuiranom rasponu, potrebno je podijeliti raspon u konačno mnogo intervala te dodijeliti svakoj točki interval i intenzitet - taj postupak nazivamo kvantizacijom.

Spomenute procese možemo definirati i malo formalnije. Uzorkovanje možemo smatrati particijom skupa $\mathbb{R}^2$ na mrežu, gdje su koordinate centra svake ćelije elementi $\mathbb{Z}^2$. S druge strane, kvantiziranje je pridruživanje intenziteta elementima mreže, s tim da je broj intenziteta zbog hardverskih zathjeva najčešće broj oblika $L = 2^k$. Taj se broj često zove **dinamički raspon** i karakteristika je sustava za generiranje slike. Broj bitova za pohranu digitizirane slike dobijemo formulom

$$b = M \times N \times k.$$

Kad slika ima dinamički raspon od 2^k, kažemo da je slika k-bitna (česti primjeri su 8-, 16- i 32-bitne slike). Na primjer, 8-bitna slika ima raspon od 256 mogućih intenziteta, također zvanih **sive vrijednosti** (*"greyvalues"*).



(a) (b)

Slika 2.1: (a) Slika psa. Ishodište se konvencionalno smjesti u gornji lijevi kut slike kako bismo uskladili ovaj prikaz s matričnim. (b) Rezultat uzorkovanja i kvantizacije - slika je projicirana na mrežu, a broj intenziteta/sivih vrijednosti je ograničen na 20.

Prikaz digitalne slike

Neka je f digitalna slika koja je nastala uzorkovanjem s mrežom dimenzije $M \times N$, gdje je M broj redaka, a N broj stupaca mreže. Prostorne koordinate ćemo indeksirati na sljedeći način: $x = 0,\ 1,\ 2, \ldots,\ M-1$ i $y = 0,\ 1,\ 2, \ldots,\ N-1$. Sliku možemo pikazati kao matricu čija je vrijednost na mjestu (i, j) upravo intenzitet u pikselu (x_i, y_j), tj. $f(x_i, y_j)$.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

Koristeći ovaj prikaz, **granicom** slike ćemo smatrati piksele reprezentirane prvim i posljednjim retkom i stupcem matrice 2.1. Skup svih piksela koji čine granicu slike f označavat ćemo s $G(f)$.

Ovisno o informacijama koje želimo izvući i strukturi korištenih algoritama, klasičnu reprezentaciju možemo zamijeniti drugim modelima. Primjerice, sliku možemo transformirati tako da vrijednost intenziteta piksela zamijenimo njegovom udaljenosti od pozadine ili neke klase objekata - tako pretvorene slike se koriste kao ulaz u *watershed* algoritmu, skeletonizaciji, granulometriji i drugim algoritmima, iako i same slike mogu biti korisno sredstvo za analizu.

2.2 Odnosi između piksela

Susjedstvo i susjednost

Segmentacija slike zahtijeva da budemo u stanju odrediti jesu li 2 piksela dio istog objekta ili regije uspoređujući njihove intenzitete i/ili položaj u mreži. Piksele u neposrednoj okolini određenog piksela smatrat ćemo njegovim *susjedima*, a smatrat ćemo ih i *susjednima* ako imaju (približno) jednak intenzitet tom pikselu.

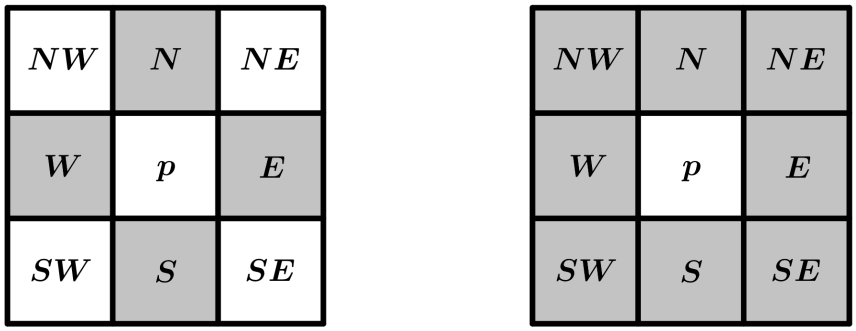
Definicija 2.2.1. *Neka je $p = (x, y)$ proizvoljan piksel slike f. Njegove horizontalne i vertikalne susjede nazivamo njegovim $4-$susjedima te uvodimo sljedeću oznaku za skup:*

$$N_4(p) := \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}. \tag{2.2}$$

Na sličan način definiramo i skup dijagonalnih susjeda piksela p:

$$N_D(p) := \{(x+1, y+1), (x-1, y+1), (x+1, y-1), (x-1, y-1)\}. \tag{2.3}$$

Elemente unije $N_8(p) := N_4(p) \cup N_D(p)$ nazivamo $8-$susjedima od p.



(a) 4-susjedstvo $N_4(p)$ (b) 8-susjedstvo $N_8(p)$

Slika 2.2: Piksel p i njegovi susjedi - potonji su označeni sivom bojom.

Skup $N_4(p)$ nazivamo *Von Neumannovim* ili 4-susjedstvom, a skup $N_8(p)$ nazivamo *Mooreovim* ili 8-susjedstvom piksela p. Radi jasnoće u referenciranju susjeda uobičajeno je koristiti tzv. geografske nazive - primjerice, piksel $(x+1, y)$ bismo nazvali "južnim" susjedom piksela (x, y).

Pri definiranju susjednosti potrebno je podijeliti dinamički raspon na manje intervale intenziteta. Naime, piksele koji su susjedi u mreži ne smatramo susjednima ako im se intenziteti previše razlikuju, odnosno ako nisu u istom intervalu V - primjerice, ako u binarnoj slici

piksel $p = (x, y)$ ima intenzitet 1, a piksel $q = (x, y-1)$ ima intenzitet 0, piksele smatramo susjedima, ali ne i susjednima.

Broj intervala ovisi o dinamičkom rasponu, o izgledu slike i cilju segmentacije. Raspon binarnih slika, sa samo 2 moguća intenziteta, prirodno dijelimo na 2 intervala, dok kod 8-bitne slike, s 256 mogućih intenziteta, imamo veću slobodu izbora broja i veličine intervala. Ako je cjelokupna slika tamnija, obično je korisnije gušće podijeliti niže intenzitete nego podijeliti čitavi raspon na jednake intervale. Također, ako je cilj izdvojiti samo bijele piksele sa slike, birat ćemo 2 intervala (V_1 = "bijeli", V_2 = "ostali") bez obzira na to koliki je dinamički raspon.

Definicija 2.2.2. *Neka su* $p = (x_1, y_1)$, $q = (x_2, y_2)$ *proizvoljni pikseli s vrijednostima u V, gdje je V podskup dinamičkog raspona* $[0, L-1]$. *Definiramo* 3 *vrste susjednosti:*

1. 4*-susjednost. Kažemo da su p i q* 4*-susjedni ako je* $q \in N_4(p)$.
2. 8*-susjednost. Kažemo da su p i q* 8*-susjedni ako je* $q \in N_8(p)$.
3. *"mješovita" ili m-susjednost. Kažemo da su p i q m-susjedni ako*

$$q \in N_4(p) \vee \{q \in N_D(p) \ \wedge \ N_4(p) \cap N_4(q) \cap V = \varnothing\}.$$

Mješovita susjednost je modifikacija 8-susjednosti uvedena s ciljem olakšavanja pronalaska jedinstvenog puta između 2 piksela. Osim navedenih, moguće je definirati i drukčije susjednosti, poput "switch" ili *s*-susjednosti koja uzima u obzir susjede samo po onoj dijagonali koju bira tzv. matrica "sklopka" (više u [10]). Ako se ne naglasi drukčije, u daljnjem tekstu se podrazumijeva korištenje 8-susjednosti, premda su definicije neovisne o odabiru susjednosti.

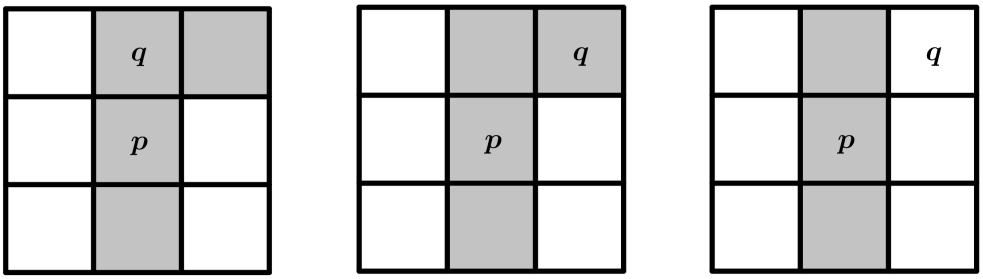
Povezanost

Određivanje susjedstva i susjednosti piksela tek je jedan korak prema segmentaciji slike. Cilj je podijeliti sliku na regije odnosno povezane komponente, pa je potrebno uvesti pojam puta i povezanosti piksela.

Definicija 2.2.3. *Neka je* $p = (x, y)$ *i* $q = (s, t)$. ***Digitalni put*** *od p do q je niz međusobno različitih piksela s koordinatama*

$$(x_0, y_0),\ (x_1, y_1), \ldots (x_n, y_n),$$

gdje je $(x_0, y_0) = (x, y)$ *i* $(x_n, y_n) = (s, t)$ *te su pikseli* (x_i, y_i) *i* (x_{i+1}, y_{i+1}) *za* $0 \leq i < n$ *susjedni. U slučaju da su početni i krajnji piksel jednaki, radi se o zatvorenom putu.*



(a) 4-susjedni p i q (b) 8-susjedni p i q (c) p i q nisu susjedni

Slika 2.3: (a) Označeni pikseli su 4-susjedni, pa su po definiciji i 8-susjedni. (b) Pikseli su samo 8-susjedni jer jer q dijagonalni susjed od p. (c) p i q imaju značajno različite intenzitete, pa ih ne smatramo susjednima.

Definicija 2.2.4. *Neka je S podskup piksela slike f. Kažemo da su $p, q \in S$* ***povezani*** *u S ako postoji put od p do q koji se sastoji od piksela iz S. Skup svih piksela koji su s pikselom p povezani u S nazivamo povezanom komponentom od S, a ako S ima samo jednu povezanu komponentu, kažemo da je skup S povezan skup.*

Definicija 2.2.5. *Neka je R podskup piksela slike f. Kažemo da je R* ***regija*** *slike ako je R povezan skup. Kažemo da su dvije regije susjedne ako je njihova unija ponovno povezan skup, inače kažemo da su disjunktne.*

Definicija 2.2.6. *Neka slika ima K disjunktnih regija R_k, $1 \leq k \leq K$, t.d. nijedna regija ne dodiruje granicu slike . Tada je $R_U = \bigcup R_K$ prednji plan slike, a njen komplement $\left(\bigcup R_K\right)^c$ pozadina.*

Neki se algoritmi temelje na promatranju granice regije s ostatkom slike. Razlikujemo unutarnju granicu, koja se čitava nalazi unutar regije, i vanjsku granicu, koja "okružuje" regiju izvana, ali ne dijeli piksele s njom.

Definicija 2.2.7. *Neka je R regija slike f. Unutarnja granica regije R je unija svih piksela iz R s bar jednim susjedom iz R^c i svih piksela R na granici slike, tj.*

$$G_u(R) := \{p \in R \mid \exists\, q \in R^c \text{ t.d. } q \in N_8(p)\} \cup \{p \in R \cup G(f)\}. \tag{2.4}$$

Vanjsku granicu regije čine pikseli iz R^c s bar jednim susjedom iz R, tj.

$$G_v(R) := \{p \in R^c \mid \exists\, q \in R \text{ t.d. } q \in N_8(p)\}. \tag{2.5}$$

Potrebno je razlikovati pojmove granice ("*border*") i ruba ("*edge*") - dok o granici možemo razmišljati kao o zatvorenom putu koji opasava regiju, definiciju ruba ćemo temeljiti na diskontinuitetima u intenzitetu u nekom danom pikselu. Granica je stoga globalni, a rub

lokalni pojam. U nekim slučajevima, primjerice u binarnim slikama, granica i rub se potpuno podudaraju.

Sad možemo formalnije definirati segmentaciju slike na regije. Označimo s $Q(R_i)$ uvjet ili skup uvjeta koje pikseli moraju zadovoljiti kako bismo ih svrstali u skup R_i (npr. intenzitet, udaljenost od pozadine, boja).

Definicija 2.2.8. *Neka je $f : A \times B \to C$ slika. Označimo s R ukupni prostor $A \times B$. Segmentacija slike je particija skupa R u n podskupova $R_1, R_2, R_3 \ldots R_n$ tako da su zadovoljeni sljedeći uvjeti:*

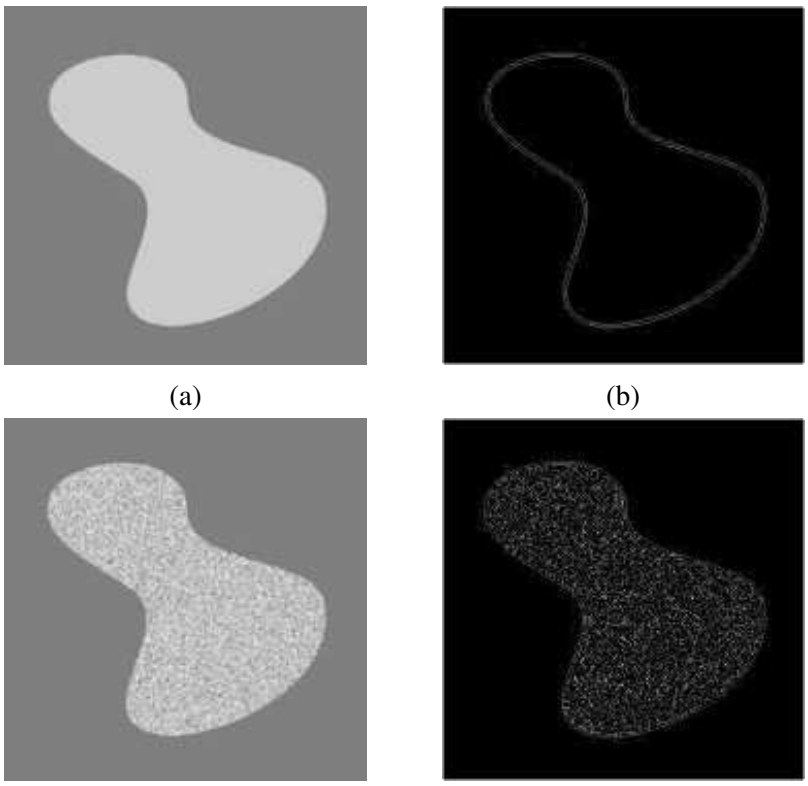
1. $\bigcup_{i=1}^{n} R_i = R$
2. R_i *je povezan* — *za* $i = 1, 2, \ldots n$
3. $R_i \cap R_j = \varnothing$ — *za sve* i *i* j, $i \neq j$
4. $Q(R_i) = \mathsf{TRUE}$ — *za* $i = 1, 2, \ldots n$
5. $Q(R_i \cup R_j) = \mathsf{FALSE}$ — *za susjedne regije* R_i *i* R_j

Dakle, segmentacija mora biti potpuna, tj. svaki piksel svrstan. Podskupovi moraju biti regije (dakle povezani), a sve regije moraju biti u parovima disjunktne. Zadnja dva zahtjeva odnose se na uvjet Q: prvi tvrdi da svi pikseli unutar regije moraju zadovoljiti dani uvjet (ili uvjete), a drugi da se susjedne regije trebaju razlikovati u smislu Q. Izraz "segmentacija" odnosi se na proces particioniranja, ali se često koristi i za konačni rezultat.

Jedan od najvećih izazova pri segmentiranju je odabir primjerenog kriterija na temelju kojeg će se pikseli dijeliti u regije. Za jednobojne slike se obično koristi sličnost intenziteta ili diskontinuitet u intenzitetu, s tim da izbor uvelike ovisi o izgledu slike. Segmentacija na temelju regija (*"region-based segmentation"*) dijeli sliku na regije koje su slične na temelju unaprijed određenih kriterija. S druge strane, segmentacija na temelju ruba (*"edge-based segmentation"*) radi uz pretpostavku da se granice regija dovoljno razlikuju međusobno i od pozadine da se možemo osloniti na nagle promjene u intenzitetu, tj. lokalne diskontinuitete kako bismo na slici izdvojili rubove i granice. S obzirom na prirodu problema koji rješavamo (detekcija točaka i linija) te vrste slika s kojima radimo (bez jakog šuma i s jasnim rubovima), pretpostavka će biti zadovoljena stoga biramo pristup na temelju ruba.

Udaljenost između piksela

Ranije smo spomenuli da je u nekim situacijama korisno koristiti prikaz slike na kojem je fokus na udaljenostima među pikselima, a ne na njihovim intenzitetima. Za konstrukciju



(a) (b)

(c) (d)

Slika 2.4: Primjeri segmentacije pomoću detekcije diskontinuiteta. (a) Obje regije imaju konstantni intenzitet te je jasno vidljiva granica svijetle regije. (b) Pretpostavka je zadovoljena i segmentacija je uspješna. (c) Unutar svijetle regije ovaj put ima puno šuma. (d) Detektor je osjetljivo reagirao na diskontinuitete uzrokovane šumom, stoga nije uspješno segmentirao sliku. U ovakvom je slučaju primjerenije segmentirati na temelju regija. Slike (a) i (c) su preuzete iz [35].

takvih prikaza, ali i za direktnu primjenu u algoritmima za pronalaženje puteva i povezivanje rubova u granice, potrebno je uvesti funkciju udaljenosti za piksele.

Definicija 2.2.9. *Neka je* $f : A \times B \to [0,\ 2^k - 1]$ *slika i neka su* $p = (x,y)$, $q = (s,t)$ *i* $z = (v,w)$ *pikseli slike* f. *Kažemo da je funkcija* $D : A \times B \to \mathbb{R}$ ***metrika*** *ako zadovoljava sljedeće uvjete:*

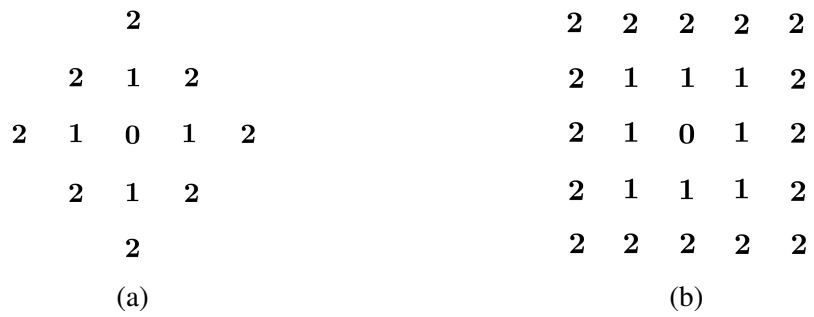
1. $D(p,q) \geq 0$
2. $D(p,q) = 0 \iff p = q$
3. $D(p,q) = D(q,p)$ *(simetrija)*

4. $D(p,z) \leq D(p,q) + D(q,z)$ *(nejednakost trokuta)*

Definicija 2.2.10. *Neka je* $f : A \times B \to [0,\ 2^k - 1]$ *slika i neka su* $p = (x,y)$ *i* $q = (s,t)$ *pikseli slike* f*. Definiramo sljedeće metrike, odnosno funkcije udaljenosti:*

1. D_4 *("taxicab") udaljenost:* $D_4(p,q) := |x-s| + |y-t|$
2. D_8 *("chessboard") udaljenost:* $D_8(p,q) := \max(|x-s|, |y-t|)$
3. D_e *ili euklidska udaljenost:* $D_e(p,q) := \sqrt{(x-s)^2 + (y-t)^2}$

Pikseli koji su prema D_4 udaljenosti udaljeni od piksela p za manje od r čine dijagonalno orijentirani kvadrat s centrom u p čiji su vrhovi udaljeni $\lfloor r \rfloor$ piksela od centra (slika 2.5a). Skup piksela q koji zadovoljavaju $D_4(p,q) = 1$ je upravo 4-susjedstvo od p. S druge strane, pikseli koji zadovoljavaju $D_8(p,q) \leq r$ bit će dio kvadrata s centrom u p poput onog na slici 2.5b, a pikseli za koje vrijedi $D_8(p,q) = 1$ su njegovi 8-susjedi.



(a) (b)

Slika 2.5: Skupovi piksela čija je udaljenost od centralnog piksela najviše 2 prema (a) D_4 metrici i (b) D_8 metrici, zajedno s njihovim udaljenostima od centra.

D_4 i D_8 udaljenosti između p i q ne ovise o putevima između tih dvaju piksela, nego su ovisne samo o njihovim prostornim koordinatama. Kad bismo koristili m-susjednost, D_m udaljenost bismo definirali kao duljinu najkraćeg m-puta između tih piksela te bi u tom slučaju ona ovisila i o intenzitetima piksela u susjedstvu i piksela duž puta koji povezuje p i q.

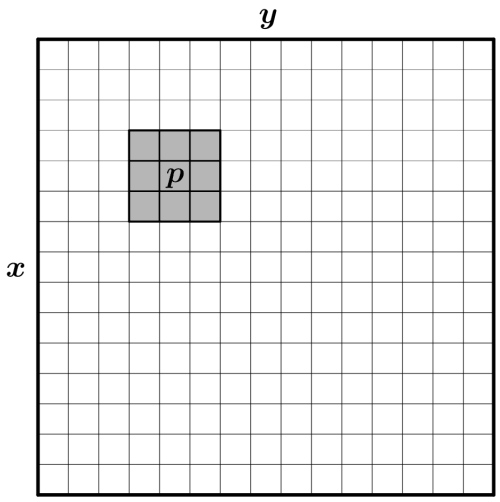
Poglavlje 3

Prostorni filteri

Prostorno filtriranje obuhvaća metode izmjene netransformirane slike djelovanjem na okolinu svakog piksela na slici. Neka je $f(x,y)$ ulazna slika, T operator na f definiran djelovanjem na okolinu točke (x,y), a $g(x,y)$ izlazna slika. Prostorni filter možemo zapisati na sljedeći način:

$$g(x,y) = T[f(x,y)]. \tag{3.1}$$

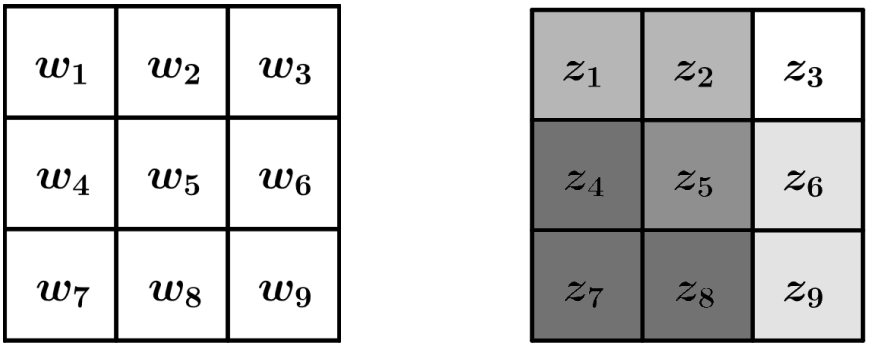
Okolina na koju T djeluje je obično pravokutnik ili kvadrat sa središtem u (x,y), znatno manji od same slike. Može se raditi i s drukčijim okolinama, npr. digitalnom aproksimacijom kruga, ali one su računski složenije te se stoga rjeđe koriste.



Slika 3.1: 3×3 okolina piksela p označena je sivom bojom. Za razliku od susjedstva, smatramo da je piksel p dio svoje okoline.

Djelovanje filtera određeno je okolinom i operatorom T - on prima piksele iz okoline, nad njima izvrši operaciju te kao rezultat izmijeni piksel u središtu odabrane okoline. Putujući po slici, on gradi procesiranu (”filtriranu”) sliku piksel po piksel. Ako je operator T linearan, kažemo da je filter **linearni prostorni filter**, inače kažemo da je **nelinearan**.

Za linearni filter koji djeluje na $m \times n$ okolinu piksela p (poput one u 3.2b) uvodimo $m \times n$ **masku** s mn koeficijenata (vidi sliku 3.2a). Kako je jednostavnije raditi s maskama neparnih dimenzija, za masku veličine $m \times n$ pretpostavljamo da vrijedi $m = 2a + 1$ i $n = 2b + 1$ za neke $a, b \in \mathbb{N}$.



(a) 3×3 maska s koeficijentima w_i (b) pikseli 3×3 okoline s intenzitetima z_i

Slika 3.2

Primjer 3.0.1. *Na slici 3.3 vidimo primjer slike filtrirane maskom veličine* 3×3*. Centralni koeficijent* $w(0,0)$ *preklapa se s centralnim pikselom* (x, y)*. Odziv filtera* $g(x, y)$ *u pikselu* (x, y) *je zbroj produkata koeficijenata filtera i intenziteta piksela pokrivenih filterom:*

$$g(x,y) = w(-1,1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(1,1)f(x+1,y+1).$$

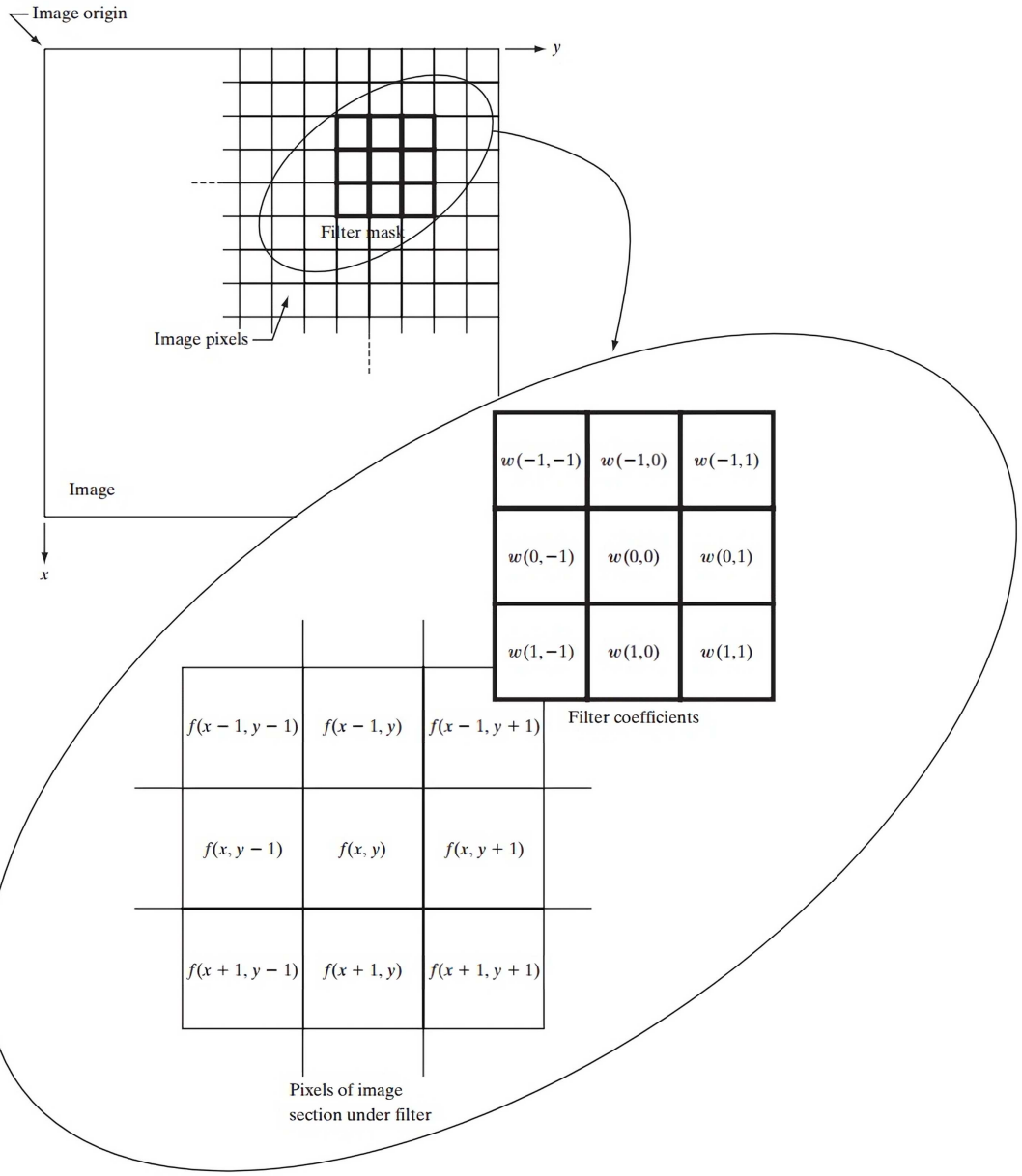
Kako bismo pojednostavnili zapis, koeficijente maske možemo označiti kao u 3.2a, a intenzitete piksela kao u 3.2b. Time dobijemo

$$g(x,y) = w_1z_1 + w_2z_2 + \cdots + w_9z_9.$$

Ukratko, linearno prostorno filtriranje slike $M \times N$ s maskom veličine $m \times n$ opisano je sljedećim izrazom:

$$R = w_1z_1 + w_2z_2 + \cdots + w_{mn}z_{mn} = \sum_{i=1}^{mn} w_iz_i = \mathbf{w}^\top \mathbf{z}, \tag{3.2}$$

gdje R predstavlja odziv filtera u p, w-ovi predstavljaju koeficijente maske, a z-ovi predstavljaju intenzitete piksela iz okoline piksela p.



Slika 3.3: Mehanika linearnog prostornog filtriranja 3×3 maskom, opisana jednadžbom iz primjera 3.0.1. U sekciji o korelaciji i konvoluciji ćemo predstaviti poopćenje te formule. Ilustracija je preuzeta iz [35].

3.1 Prostorna korelacija i konvolucija

Razlikujemo dva načina linearnog filtriranja slike - filtriranje korelacijom i filtriranje konvolucijom. Korelacijom nazivamo proces pomicanja maske po slici i računanja sume produkata na svakoj lokaciji, kao što je opisano u 3.2. Konvolucija se provodi na isti način, ali se maska prije korištenja zarotira za 180°. Rotacijom filtera w korelaciju možemo pretvoriti u konvoluciju i obratno, stoga izbor metode filtriranja može biti proizvoljan, no pri opisu algoritama je potrebno precizirati izbor radi prilagodbe maske - ako se izvodi korelacija, koristi se maska w, a ako se koristi konvolucija, w trebamo zarotirati prije primjene.

Objasnimo kako funkcionira filtriranje slike. Pretpostavimo da imamo filter veličine $m \times n$. Potrebno je dopuniti sliku s barem $m-1$ redova nula s gornje i donje strane te $n-1$ stupaca nula s lijeve i desne strane. Budući da ćemo raditi s 3×3 filterima, nadodavat ćemo 2 reda i stupca sa svake strane slike prije primjene filtera - primjer nadopunjene slike prikazan je na slici 3.4b. Nadopunjene vrijednosti ne moraju nužno biti samo nule - možemo reflektirati sadržaj slike preko granice (*"reflective edge treatment"*) ili reflektirati i granicu zajedno s ostatkom slike (*"reflective double-edge treatment"*).

Korelacija je funkcija premještanja filtera - to znači da prva vrijednost korelacije odgovara premještanju filtera za nulu, druga vrijednost odgovara premještanju filtera za jedan, itd. Primjenom filtera w na sliku koja sadrži samo nule i jednu jedinicu (inače zvanu 2D diskretni jedinični impuls), na mjestu jedinice (impulsa) dobijemo kopiju w zarotiranu za 180° (što u 2D slučaju znači zrcaljenje preko obje osi). Konvolucija maske w s jediničnim impulsom vraća kopiju maske w na mjestu impulsa - naime, rotiranje maske prije filtriranja poništi kasniju rotaciju.

Izrazimo korelaciju i konvoluciju jednadžbama. Radi jednostavnosti ponovno pretpostavljamo da je maska neparnih dimenzija i za $m \times n$ masku uvodimo vrijednosti $a = (m-1)/2$ i $b = (n-1)/2$.

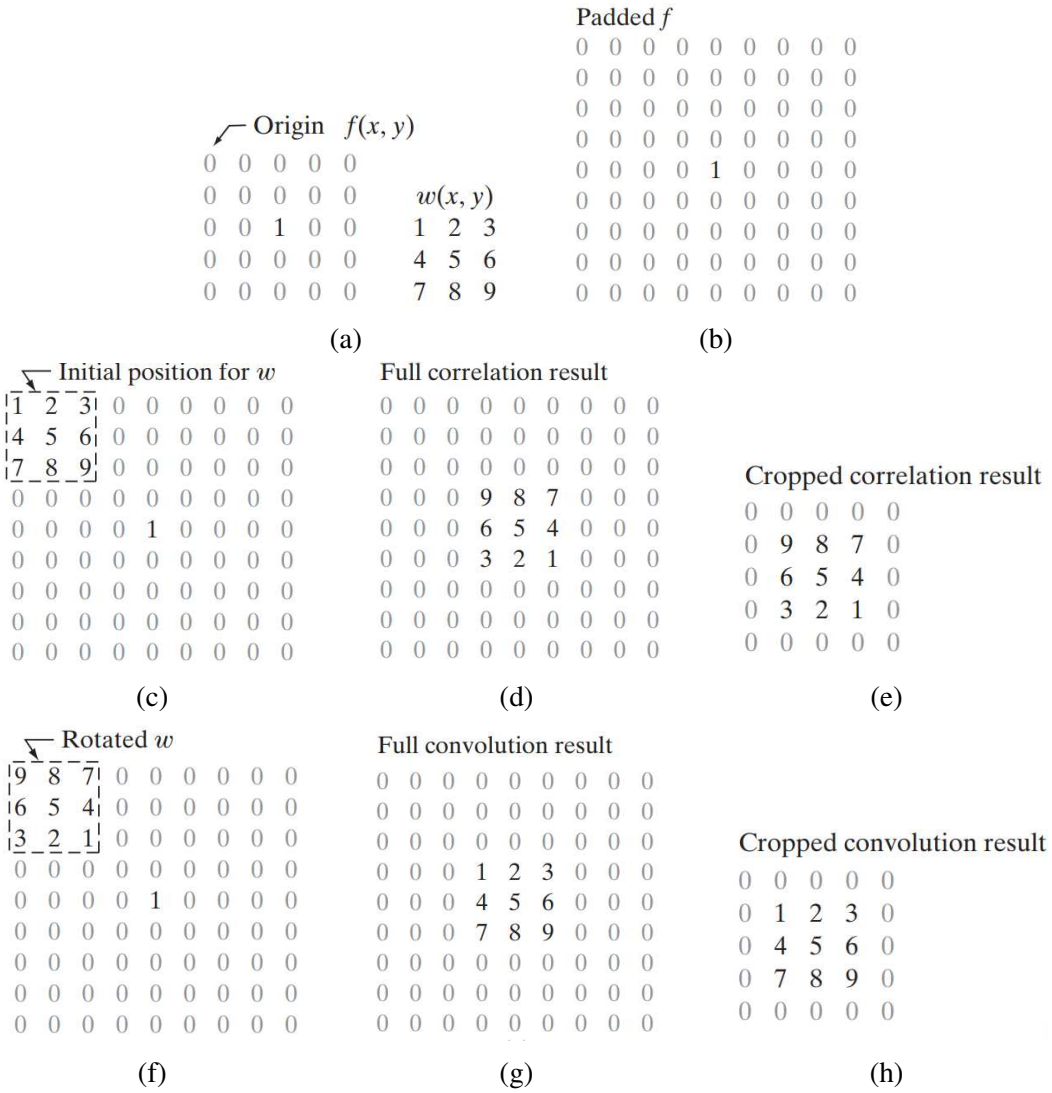
Definicija 3.1.1. *Neka je w filter, a f slika popunjena u skladu s filterom w. Korelaciju filtera $w(x,y)$ s $f(x,y)$ definiramo sljedećom formulom*

$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t). \tag{3.3}$$

Definicija 3.1.2. *Neka je w filter, a f slika popunjena u skladu s filterom w. Konvoluciju filtera $w(x,y)$ s $f(x,y)$ definiramo na sljedeći način*

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t). \tag{3.4}$$

Zrcaljenjem i pomicanjem funkcije f umjesto filtera w pojednostavljujemo zapis i usklađujemo definiciju s konvencionalnim zapisom konvolucije, a rezultat je u oba slučaja isti.



Slika 3.4: U prvom redu nadopunjujemo sliku f nulama. Na nadopunjenu sliku u (b) primijenimo korelaciju (drugi red) i konvoluciju (treći red). Ilustracije su preuzete iz [35].

3.2 Filteri za izglađivanje

Jedan od osnovnih načina uklanjanja šuma sa slike je pomoću linearnog filtera koji vraća prosjek intenziteta piksela pokrivenih maskom. Takve filtere nazivamo **niskopropusnim**[1] (*"low-pass"*), odnosno filterima usrednjavanja (*"averaging"*). Primjenom niskopropusnog filtera na slici se uklone oštri prijelazi u intenzitetu, jedno od obilježja slučajnog šuma, ali i rubova, stoga ovaj oblik uklanjanja šuma ima ponekad nezgodnu nuspojavu zamagljenja rubova. Još jedna korisna primjena je u izglađivanju lažnih kontura[2] koje nastanu kad je dinamički raspon preuzak (vidi sliku 3.6).

Najjednostavniji primjer takvog filtera je tzv. **box filter** (poznat i kao mean filter), koji računa aritmetičku sredinu piksela pod maskom. Prema centralnom graničnom teoremu slijedi da se uzastopnom primjenom box filtera može aproksimirati Gaussov filter (vidi [14]). Osim aritmetičke sredine, može se uzeti i težinska sredina (vidi 3.5b), tj. nekim se pikselima može pridodati veća vrijednost u odnosu na druge. Odabir "važnijih" piksela ovisi o željenom učinku, ali uobičajeno je pikselima u sredini maske dati veću težinu nego onima na rubu kako bi se slika izgladila uz manje zamućivanja.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

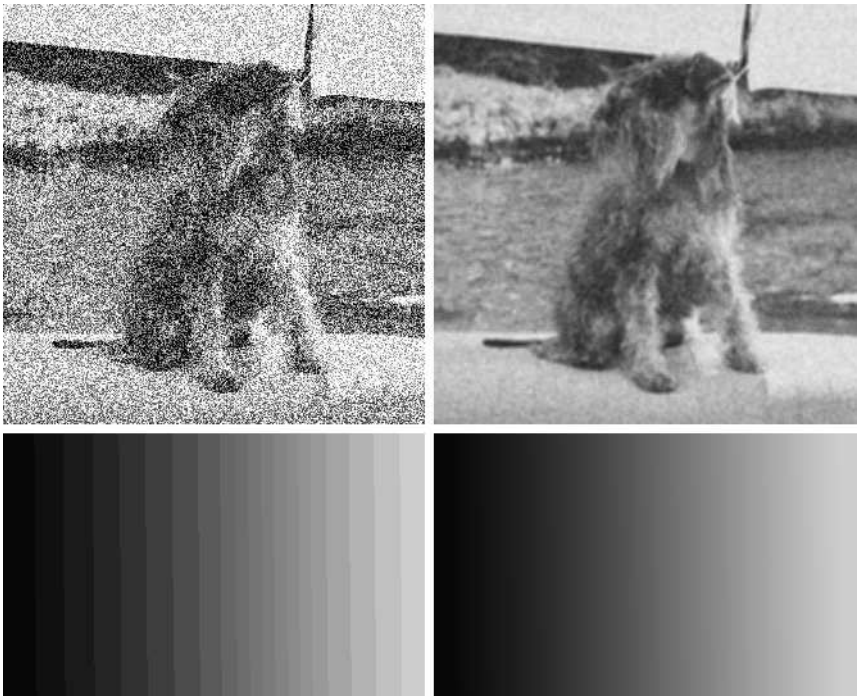
(a) standardni box filter (b) težinski box filter

Slika 3.5: Maske 2 izglađujuća linearna filtera. Ignoriramo li koeficijent pred maskom, maska (b) je najbolja aproksimacija Gaussove krivulje malim cijelim brojevima (vidi [7]).

Jedan od najčešće korištenih filtera je već spomenuti **Gaussov filter**. Gaussov filter će igrati značajnu ulogu u više algoritama za detekciju ruba koje ćemo kasnije obraditi. Nje-

[1] Niskopropusni filteri propuštaju niske frekvencije, a potiskuju visoke frekvencije. Područja konstantnog intenziteta imaju niske frekvencije, dok značajke poput rubova obično imaju visoke, stoga je konačan efekt izglađivanje i zamućivanje slike.

[2] Pojava lažnih kontura naziva se "posterizacijom".



Slika 3.6: Primjeri poboljšanja slika pomoću box filtera - na gornjoj slici je raščišćen šum, a na donjoj su ublažene lažne konture nastale zbog ograničenog broja razina intenziteta.

govo djelovanje je slično box filteru, samo što ovaj put masku konstruiramo diskretizacijom Gaussove funkcije.

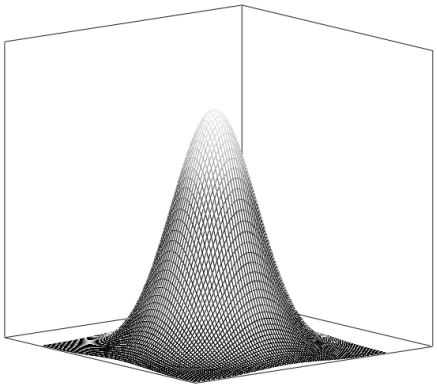
Definicija 3.2.1. ***Gaussovu funkciju*** *ili Gaussian sa standardnom devijacijom* $\sigma = (\sigma_x, \sigma_y)$ *i srednjom vrijednosti* $\mu = (\mu_x, \mu_y)$ *definiramo sljedećom formulom*

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma_x\sigma_y}}. \tag{3.5}$$

U naše svrhe je dovoljno raditi s Gaussijanom sa srednjom vrijednosti $\mu = (0,0)$, a za standardnu devijaciju ćemo birati da vrijedi $\sigma = \sigma_x = \sigma_y$. Same vrijednosti Gaussove funkcije najčešće nisu toliko važne[3] koliko je važno očuvati odnos veličina i centralnu simetriju, pa možemo izostaviti i konstantu. Formulu 3.5 sad možemo zapisati kao

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.6}$$

[3]U nekim situacijama je važno paziti na red veličine koeficijenata maske. Na primjer, jedan od načina za izdvajanje oštrijih značajki (poput ruba) sa slike je oduzimanje izglađene slike od originalne - u tom je slučaju potrebno paziti da je red veličina na obje slike isti kako bi se dobili smisleni rezultati.



(a)

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

(b)

Slika 3.7: Primjer Gaussove funkcije sa standardnom devijacijom $\sigma = 1.4$. Slijeva vidimo graf Gaussijana, a zdesna primjer maske koja oponaša funkciju, s tim da centar maske zamišljamo smještenog u vrh zvonolike krivulje.

Masku ćemo konstruirati uzorkovanjem Gaussijana, što je moguće izvesti na više načina. Na slici 3.7 slijeva možemo vidjeti primjer Gaussove funkcije sa standardnom devijacijom $\sigma = 1.4$, a zdesna jedan primjer 5×5 maske koja aproksimira oblik krivulje cijelim brojevima. Uzorkovanje u slučaju na slici je provedeno računanjem vrijednosti Gaussove funkcije u točkama rešetke $\mathbb{L}^2$, gdje je $\mathbb{L} = \{-2, -1, 0, 1, 2\}$, pa skaliranjem svih vrijednosti istim faktorom. Što je dimenzija maske veća, to će aproksimacija krivulje biti preciznija - za maske veličine 30×30 i veće su greške gotovo zanemarive. Međutim, bilo je pokušaja optimizacije koeficijenata manjih maski (npr. 3×3, 5×5). Davies[4] je definirao dva kriterija za kvalitetu maske: vjernost formi Gaussijana i izotropnost, te je na temelju tih uvjeta izračunao maske predstavljene na slici 3.8.

$$\begin{pmatrix} 1/\sqrt{2} & 1 & 1/\sqrt{2} \\ 1 & \sqrt{2} & 1 \\ 1/\sqrt{2} & 1 & 1/\sqrt{2} \end{pmatrix}$$

(a)

$$\begin{pmatrix} 0.09 & 0.25 & 0.35 & 0.25 & 0.09 \\ 0.25 & 0.71 & 1.00 & 0.71 & 0.25 \\ 0.35 & 1.00 & 1.42 & 1.00 & 0.35 \\ 0.25 & 0.71 & 1.00 & 0.71 & 0.25 \\ 0.09 & 0.25 & 0.35 & 0.25 & 0.09 \end{pmatrix}$$

(b)

Slika 3.8: Optimalne aproksimacije Gaussijana za (a) 3×3 i (b) 5×5 okoline. Slike su preuzete iz [7].

Nakon što konstruiramo prikladnu masku, izglađivanje vršimo standardnom konvolucijom

[4]E.R. Davies, *Design of optimal Gaussian operators in small neighbourhoods* ([7])

(ili korelacijom) sa slikom koju želimo izgladiti. Međutim, postoji još jedan razlog zašto je Gaussov filter posebno popularan kao izglađujući filter - on je jedini centralno simetrični 2D operator koji se može dekomponirati na dva 1D operatora (vidi [13], [7]). To svojstvo je računski isplativo[5] jer za $n \times n$ operator sreže količinu procesiranja s $O(n^2)$ na $O(n)$.

3.3 Filteri za izoštravanje

Cilj izoštravanja slike je naglasiti prijelaze u intenzitetu. Kao što su filteri za izglađivanje okvirno vođeni idejom integriranja, filteri za izoštravanje će se temeljiti na diskretiziranim (digitalnim) derivacijama.

Derivacije digitalnih funkcija ćemo uvesti kao diferencije. Zahtijevamo da aproksimacija prve derivacije zadovoljava sljedeće uvjete:

1. u područjima konstantnog intenziteta treba iznositi 0
2. poprima nenul-vrijednost na početku kosine ili stepenice
3. poprima nenul-vrijednost duž kosine

Kosinom smatramo polagani prijelaz u intenzitetu iz jedne vrijednosti u drugu, a stepenicom nagli skok u intenzitetu. Aproksimacija druge derivacije treba zadovoljiti prva dva uvjeta, ali postavljamo dodatni zahtjev da poprima nenul-vrijednost i na početku i na kraju kosina i stepenica. Također, ona mora iznositi nula duž kosine.

Neka je f realna funkcija dvije varijable. Derivaciju funkcije f po x možemo aproksimirati razvojem $f(x + \Delta x, y)$ u Taylorov red oko x i zadržavanjem samo linearnog člana.

$$f(x + \Delta x, y) = \sum_{n=0}^{\infty} \frac{1}{n!} \frac{\partial^n f(x,y)}{\partial x^n} (x + \Delta x - x)^n \approx f(x,y) + \Delta x \, \partial_x f(x,y) \tag{3.7}$$

Minimalna udaljenost duž koje se može odviti promjena je između 2 susjeda na mreži, pa možemo uzeti $\Delta x = 1$. Uvrštavanjem u jednadžbu 3.7 slijedi:

Definicija 3.3.1. *Neka je f proizvoljna slika. Prvu derivaciju od f u varijabli x definiramo kao digitalnu diferenciju*

$$\partial_x f(x,y) = f(x+1,y) - f(x,y). \tag{3.8}$$

Analogno definiramo i derivaciju po y:

$$\partial_y f(x,y) = f(x,y+1) - f(x,y). \tag{3.9}$$

[5] Ovakva dekompozicija uništi izotropnost operatora, pa je potrebno oprezno birati način implementacije.

Deriviranjem prve derivacije po x dobijemo aproksimaciju derivacije drugog reda po x:

$$\partial_x f(x+1,y) - \partial_x f(x,y) = f(x+2,y) - 2f(x+1,y) + f(x,y)$$

S obzirom da nas zanima druga derivacija oko x, pomaknut ćemo argument za 1:

$$\partial_x f(x,y) - \partial_x f(x-1,y) = f(x+1,y) - 2f(x,y) + f(x-1,y)$$

Definicija 3.3.2. *Neka je f proizvoljna slika. Drugu derivaciju od f u varijabli x definiramo kao digitalnu diferenciju*

$$\frac{\partial^2 f}{\partial x^2}(x,y) = f(x+1,y) - 2f(x,y) + f(x-1,y). \tag{3.10}$$

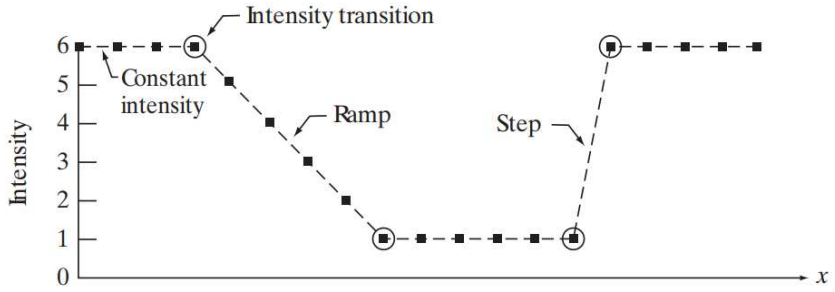
Analogno definiramo drugu derivaciju po varijabli y:

$$\frac{\partial^2 f}{\partial y^2}(x,y) = f(x,y+1) - 2f(x,y) + f(x,y-1). \tag{3.11}$$

Ovako definirana prva i druga digitalna derivacija zadovoljavaju navedene uvjete te ih smatramo valjanim aproksimacijama. Promotrimo sličnosti i razlike u njihovom ponašanju na slici 3.9. Profil intenziteta u 3.9a sadrži kosinu (”intensity ramp”), 3 područja konstantne vrijednosti i stepenicu (”intensity step”), a početak i kraj svakog navedenog prijelaza intenziteta označen je kružićem. U 3.9b su za svaku točku iz profila intenziteta zabilježeni njen intenzitet te vrijednosti prve i druge derivacije u toj točki. Konačno, treća slika 3.9c pruža vizualizaciju derivacija na temelju vrijednosti iz 3.9b.

Obratimo li pažnju na grafove u 3.9b i 3.9c, jasno je da obje derivacije zadovoljavaju prvi uvjet - na području konstantnog intenziteta su nula. Obje derivacije poprimaju nenulvrijednosti na početku kosine i stepenice, dok druga derivacija nije nula ni na kraju prijelaza. Zadovoljen je i treći uvjet - duž kosine prva derivacija nije jednaka nuli, a druga jest. Jedna značajna stvar koju možemo primijetiti je promjena znaka druge derivacije na početku i kraju prijelaza intenziteta te da crta koja u 3.9c povezuje te dvije vrijednosti prolazi kroz horizontalnu os - tu pojavu, tj. svojstvo druge derivacije nazivamo ”zero crossing” svojstvom ili svojstvom prijelaza kroz nulu i ono je temelj brojnih metoda za detekciju ruba.

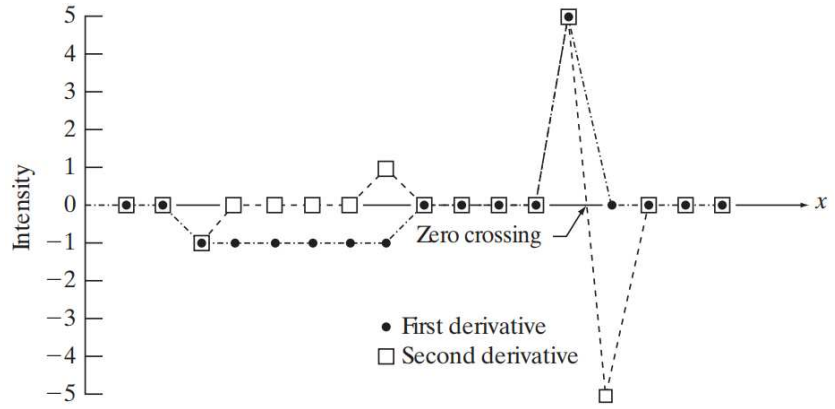
Rubovi se na digitalnim slikama javljaju kao prijelazi u obliku kosine (ili stepenice). Budući da prva derivacija nije nula duž kosine, očekujemo da će primjena prve derivacije rezultirati debljim rubovima u odzivu. S druge strane, druga derivacija vraća rub debljine 1 piksel okružen nulama, pa će njeni rubovi biti tanki. Zbog naglih skokova u intenzitetu pretpostavljamo da će druga derivacija pojačati sitne detalje i jače izoštriti sliku.



(a) Pojednostavljeni profil intenziteta. Crnim točkama su obilježeni pikseli čiji nam je intenzitet poznat, a kružićem su označena mjesta početka i kraja prijelaza u intenzitetu.

Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0		
2nd derivative	0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0		

(b) Traka u prvom redu prikazuje vrijednosti intenziteta u točkama iz (a). U drugom retku je prva derivacija, izračunata oduzimanjem intenziteta trenutne točke od intenziteta sljedeće. U trećem retku je druga derivacija, izračunata oduzimanjem intenziteta prethodne točke od sljedeće. Kako ne bismo ispali iz raspona trake, derivacije ne računamo u prvoj i posljednjoj točki.



(c) Pojednostavljeni profil intenziteta prve i druge derivacije. Crne točkice označavaju vrijednosti prve derivacije, a bijele točke vrijednosti druge. Na mjestu stepenice je vidljiv prijelaz kroz nulu (”zero crossing”), jako korisno svojstvo druge derivacije.

Slika 3.9: Ilustracija prve i druge derivacije isječka iz horizontalnog profila slike. Slike su preuzete iz [35].

Laplaceov operator

Cilj nam je konstruirati masku na bazi druge derivacije. Zahtijevamo da je maska izotropna - ne želimo da odziv ovisi o smjeru diskontinuiteta, tj. želimo dobiti isti rezultat primijenimo li filter na rotiranu sliku i rotiramo li filtriranu sliku. Najjednostavniji izotropni diferencijalni operator je Laplaceov operator.

Definicija 3.3.3. *Neka je f slika.* ***Laplacian*** *je linearni operator na f zadan s*

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{3.12}$$

Koristeći 3.10 i 3.11, dobijemo diskretnu formulaciju Laplaciana:

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y). \tag{3.13}$$

Zbog diskretizacije je operator izotropan samo za rotacije u inkrementima od 90°, ali moguće je formulu prilagoditi dodavanjem članova za dijagonalne smjerove te postići izotropnost za inkremente od 45°.

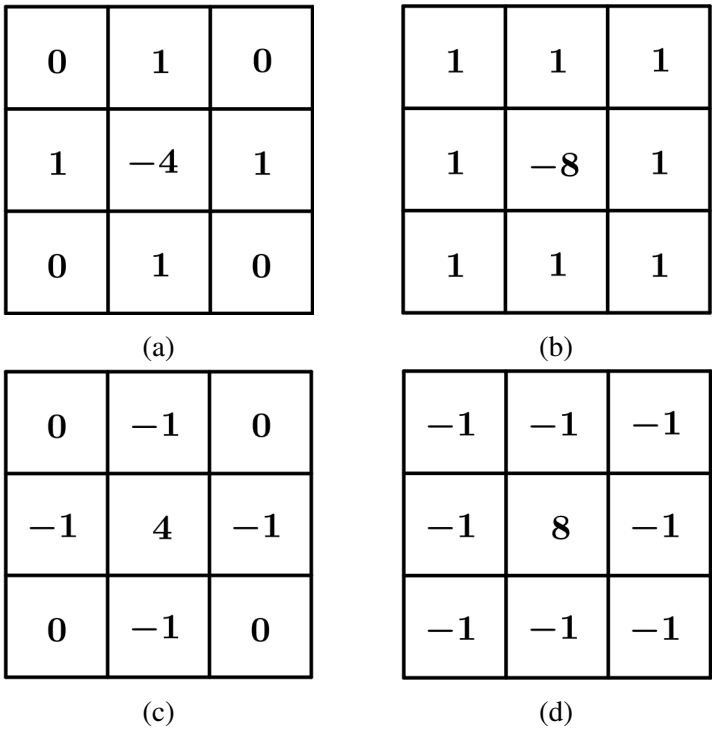
$$\begin{aligned}\nabla^2 f(x,y) = {} & f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) \\ & + f(x-1,y-1) + f(x-1,y+1) + f(x+1,y-1) \\ & + f(x+1,y+1) - 8f(x,y)\end{aligned} \tag{3.14}$$

Kao i ostali filteri za izoštravanje, Laplacian priguši dijelove slike s blažim prijelazima u intenzitetu. Moguće je obnoviti te dijelove dodavanjem ili oduzimanjem odziva od originalne slike (ovisno o implementaciji filtera, vidi sliku 3.11). Dakle, izoštravanje slike Laplacianom možemo izraziti preko sljedeće formule

$$g(x,y) = f(x,y) + c\Big[\nabla^2 f(x,y)\Big], \tag{3.15}$$

gdje je ovisno o odabiru maske $c = 1$ ili $c = -1$.

Laplacian se u praksi rijetko koristi sam jer je popraćen efektom dvostrukog ruba (*"double edge effect"*). Naime, jedan od uvjeta koji mora zadovoljiti aproksimacija druge derivacije je taj da u području konstantnog intenziteta odziv iznosi nula, stoga zbroj koeficijenata u maski mora biti nula. Konvolucija takve maske sa slikom f će rezultirati slikom čiji je zbroj intenziteta svih piksela također nula pa se zato uz pozitivne vrijednosti uvijek jave i negativne vrijednosti. Na primjer, ako filter detektira svijetlu regiju na tamnoj pozadini konstantnog intenziteta, njegov odziv na rubu regije će biti dvostruko deblji rub - dio ruba koji se preklapa s pravim rubom regije imat će pozitivne vrijednosti intenziteta, a zadebljanje negativne. Postoje načini za tretirati ovaj efekt, a neke pristupe ćemo komentirati u



Slika 3.10: (a) Maska opisana formulom 3.13. (b) Maska opisana formulom 3.14. (c) i (d) su 2 često korištene alternativne implementacije opisanih maski. Pri izoštravanju slike se u slučaju (a) i (b) Laplacianov odziv oduzme, a u slučaju (c) i (d) doda originalnoj slici.

poglavlju o detekciji linija.

Još jedan od razloga zašto se Laplacian obično ne koristi sam je to što je on kao aproksimacija druge derivacije iznimno osjetljiv na šum. Međutim, ta se osjetljivost može ublažiti kombiniramo li ga sa zaglađivanjem, koje može očistiti dio šuma sa slike. Konvolucijom Laplaciana s Gaussovom maskom dobijemo tzv. Laplacian Gaussijana (*"Laplacian of a Gaussian"*), odnosno LoG operator. Ovaj operator se zbog svojeg oblika naziva i "Mexican hat" operator.

Prisjetimo se da je 2D Gaussova funkcija sa standardnom devijacijom σ dana sljedećom formulom

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3.16}$$

Konstanta $\frac{1}{2\pi\sigma^2}$, inače prisutna u definiciji Gaussijana, obično se izostavlja jer u velikom dijelu primjena nije potrebna egzaktna vrijednost Gaussove funckije. U slučaju da nam treba egzaktna vrijednost, konstantu lako nadodamo u konačnu formulu za LoG operator.

Slika 3.11: Slika mjeseca prije i nakon izoštravanja Laplacianom. Rezultat filtriranja s 3.10b je oduzet od originalne slike. Slika na lijevoj strani je preuzeta iz [35].

Sad računski izvedimo jednadžbu za Laplacian Gaussijana:

$$\begin{aligned}\nabla^2 G(x,y) &= \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} = \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] \\ &= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{3.17}$$

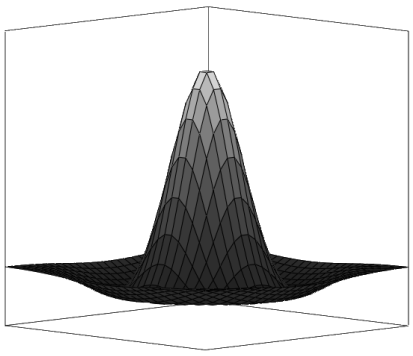
Laplaceov i LoG operator su temelj tzv. *"zero crossing"* detekcije ruba. Naime, kako u profilu intenziteta očekujemo vidjeti stepenice na mjestu gdje se javlja rub, a druga derivacija na takvim mjestima ima prijelaz kroz nulu, jedan od načina za pronaći kandidate za piksele ruba je filtrirati sliku filterom sa zero crossing svojstvom, odsjeći odziv operatora na nuli (ili dovoljno malenoj pozitivnoj vrijednosti) i zadržati točke na granici pozadine i prednjeg plana, odnosno točke prijelaza kroz nulu. O ovoj ideji ćemo reći nešto više u poglavlju o detekciji ruba.

Nelinearno izoštravanje slika

Gradijent funkcije f u pikselu p je vektor koji daje smjer i stopu najveće promjene funkcije u tom pikselu. U obradi slika se prva derivacija implementira pomoću magnitude gradijenta.

Definicija 3.3.4. *Neka je f slika.* ***Gradijent*** *slike f u pikselu (x,y) definiramo kao*

$$\nabla f := \operatorname{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.18}$$



(a)

0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

(b)

Slika 3.12: (a) negativan LoG (*"Mexican hat"*) operator. (b) 5×5 maska koja aproksimira oblik iz (a). U praksi se koristi verzija ove maske sa suprotnim predznacima koeficijenata. Maska je dobivena sličnom metodom (uzorkovanjem) kao i aproksimacija Gaussove krivulje (vidi 3.7b).

Magnituda *vektora ∇f u pikselu (x, y) predstavlja stopu promjene u smjeru gradijentnog vektora u tom pikselu. Označavamo ju s $M(x, y)$ i definiramo ju kao*

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}. \tag{3.19}$$

Slika $M(x, y)$ je iste veličine kao original i nazivamo ju **gradijentnom slikom** (skraćeno gradijent).

Komponente gradijentnog vektora su derivacije, pa su one ujedno i linearni operatori, dok magnituda nije. S druge strane, magnituda je izotropna, a parcijalne derivacije definirane u 3.18 nisu. Ponekad je računski prikladnije aproksimirati kvadrate i kvadratni korijen apsolutnom vrijednosti, tj.

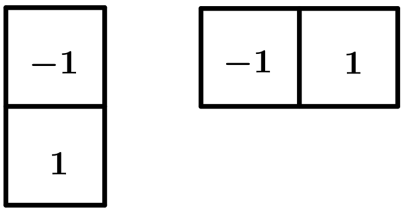
$$M(x, y) \approx |g_x| + |g_y|. \tag{3.20}$$

Ovaj izraz čuva relativne promjene u intenzitetu, premda ovako aproksimirana magnituda nažalost gubi svojstvo izotropnosti. Međutim, kako digitalni gradijenti ionako nisu izotropni (osim za ograničeni broj inkremenata koji dozvoljava maska), dovoljno je aproksimirati gradijent maskom koja je izotropna za inkremente od npr. 90°.

Uvodimo diskretnu aproksimaciju prethodnih jednadžbi i pomoću toga ćemo formulirati primjerene maske. Najjednostavnija aproksimacija parcijalnih derivacija prvog reda dana je s 3.8 i 3.9, što koristeći skraćenu notaciju iz 3.0.1 možemo zapisati kao

$$g_x = z_8 - z_5, \tag{3.21}$$

$$g_y = z_6 - z_5. \tag{3.22}$$



(a) (b)

Slika 3.13: (a) maska koja predstavlja g_x aproksimiranu s 3.21 (b) maska koja predstavlja g_y aproksimiranu s 3.22.

Navedene aproksimacije implementiramo jednodimenzionalnim maskama sa slike 3.13. U ranom razvoju digitalne obrade slika korištene su i Robertsove[6] križne diferencije (*"cross differences"*), tj.

$$g_x = z_9 - z_5, \tag{3.23}$$

$$g_y = z_8 - z_6. \tag{3.24}$$

Uvrstimo li Robertsove diferencije u definiciju magnitude 3.19, dobijemo

$$M(x,y) = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2\right]^{1/2}, \tag{3.25}$$

a ako koristimo aproksimaciju magnitude zadanu s 3.20, formula glasi

$$M(x,y) \approx |z_9 - z_5| + |z_8 - z_6|. \tag{3.26}$$

Križne diferencije možemo implementirati kao linearne filtere pomoću maski 3.14a i 3.14b - te maske nazivamo Robertsovim operatorima križnog gradijenta. Dosad smo radili s maskama neparnih dimenzija jer one imaju centar simetrije te su stoga jednostavnije za implementirati, no ove maske su parnih dimenzija. Pokušamo li prilagoditi aproksimacije derivacija tako da radimo s 3×3 maskama, prirodno je podijeliti piksele u parove koji se nalaze sa suprotnih strana u odnosu na središnji piksel i izračunati njihovu razliku. Najjednostavnija digitalna aproksimacija parcijalnih derivacija maskama veličine 3×3 je onda dana sljedećim formulama:

$$g_x = \frac{\partial f}{\partial x} = (z_7 - z_1) + (z_8 - z_2) + (z_9 - z_3) = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{3.27}$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 - z_1) + (z_6 - z_4) + (z_9 - z_7) = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \tag{3.28}$$

[6]Lawrence G. Roberts, *Machine Perception of Three-Dimensional Solids* ([30])

Razlika između trećeg i prvog reda 3×3 okoline aproksimira prvu derivaciju u x smjeru, a razlika između trećeg i prvog stupca derivaciju u y smjeru. Ove aproksimacije implementiramo filtriranjem slike maskama 3.14c i 3.14d, a te maske zovemo Prewittini[7] operatori.

Prewittove operatore možemo modificirati tako da središnjem koeficijentu damo veću težinu, npr. 2. Tako definirane aproksimacije nazivamo Sobelovim[8] gradijentnim operatorima (vidi 3.14e i 3.14f). Središnjem koeficijentu Sobelovih maski je dana veća težina jer središnjem pikselu radi izglađivanja želimo dati veću važnost u odnosu na ostale piksele - u poglavlju o detekciji ruba ćemo vidjeti zašto je izglađivanje poželjan efekt pri radu s derivacijama. Ovako definirane aproksimacije glase:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{3.29}$$

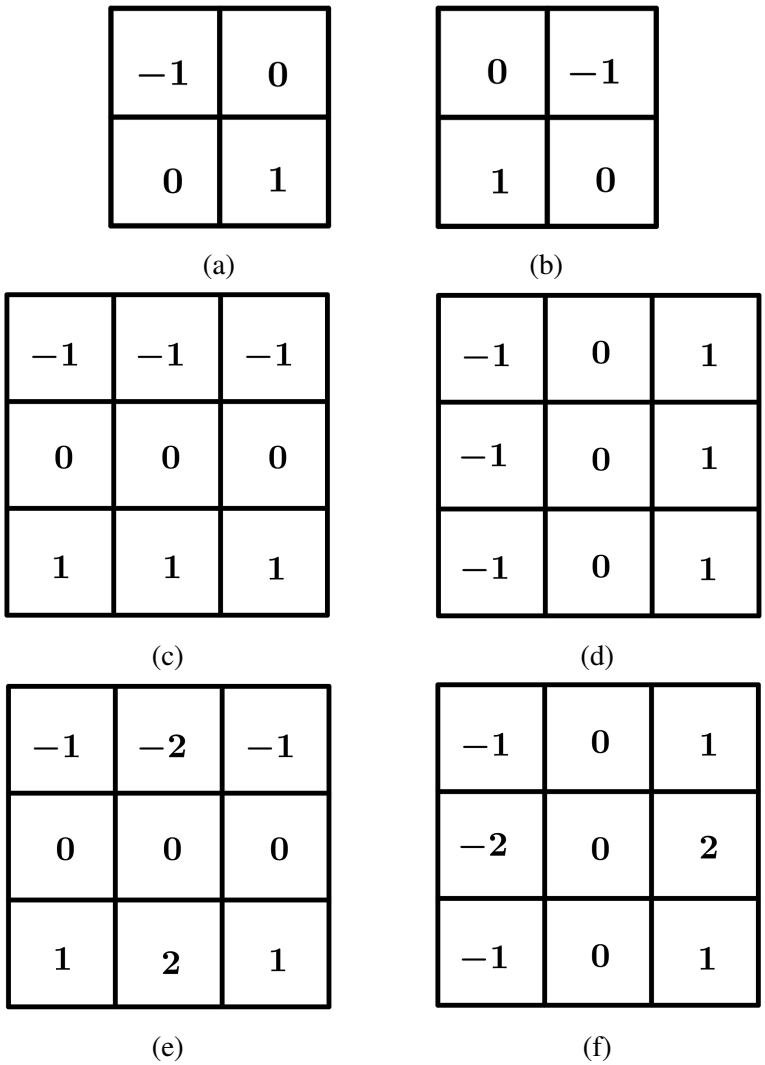
$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{3.30}$$

Uvrstimo li 3.29 i 3.30 u jednadžbu aproksimacije magnitude 3.25, dobit ćemo

$$M(x, y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|. \tag{3.31}$$

[7] Judith M.S. Prewitt, *Object Enhancement and Extraction* ([27])

[8] Irwin E. Sobel, *Camera Models and Machine Perception* ([33])

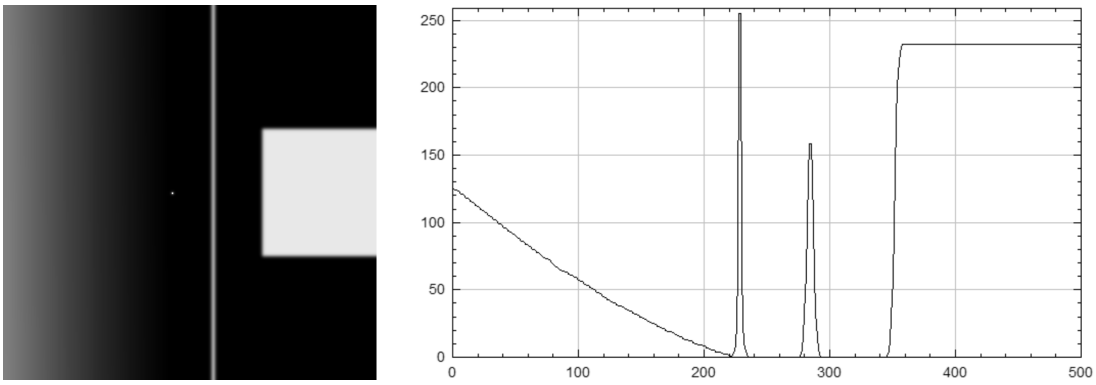


Slika 3.14: (a) i (b) Robertsovi operatori križnog gradijenta. (c) i (d) Prewittini operatori. (e) i (f) Sobelovi operatori.

Poglavlje 4

Detekcija točaka i linija

U poglavlju o filterima smo predstavili prvu i drugu derivaciju kao alate za pronalazak finih detalja i rubova na slikama. Također smo promotrili i opisali njihovo ponašanje u susretu s područjima konstantnog intenziteta i različitim prijelazima u intenzitetu. Na slici 4.1 možemo vidjeti 3 različite vrste ruba prema prijelazu u intenzitetu između objekta u prednjem planu i drugog objekta ili pozadine - kosine (*"ramp edge"*), krovove (*"roof edge"*) i stepenice (*"step edge"*). Kosine se javljaju na područjima s postupnim, a stepenice na područjima s naglim prijelazom u intenzitetu, dok se krovovi javljaju kad je objekt jako tanak ili malen. Primijetimo da izolirana točka (veličine jednog piksela) i linija (debljine jednog piksela, uz ponešto zamućivanja) uzrokuju odziv u obliku krova na profilu intenziteta. U ranijoj diskusiji smo zaključili da je druga derivacija osjetljivija na fine detalje nego prva, stoga će se metode detekcije temeljiti na aproksimaciji druge derivacije.



Slika 4.1: Slika i profil intenziteta duž horizontalne linije koja prolazi kroz područje s gradijentom, izoliranu točku, vertikalnu liniju i svijetli kvadrat. Na profilu intenziteta slijeva nadesno vidimo jednu kosinu, dva krova i jednu stepenicu.

4.1 Detekcija izoliranih točaka

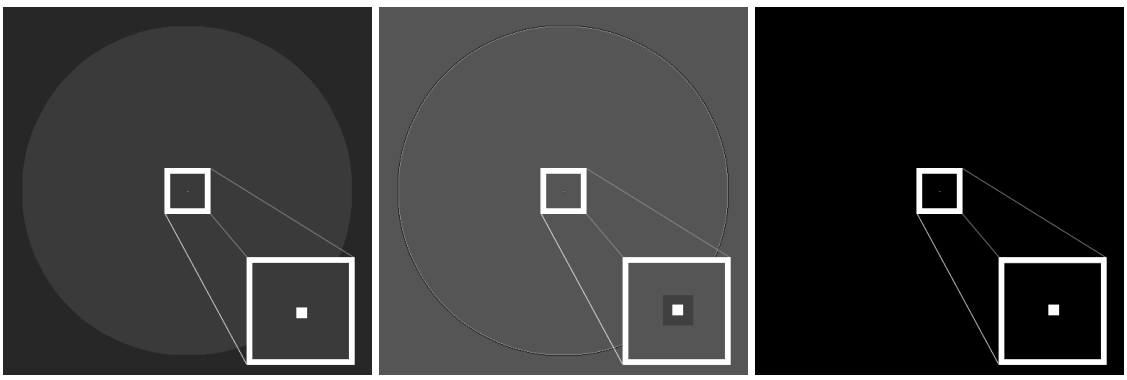
Ključna ideja u detekciji izoliranih točaka je odsjecanje (*"thresholding"*). Metoda odsjecanja se često primjenjuje na slike čiji se prednji plan i pozadina značajno razlikuju u intenzitetu. U našem slučaju će izolirana točka biti objekt u prednjem planu, dok će ostatak slike biti u pozadini. Kako je za uspjeh metode poželjna veća razlika u intenzitetu, odsjecanje nećemo primijeniti na originalnu sliku, već na sliku filtriranu Laplaceovim operatorom.

Definicija 4.1.1. *Neka je* $f : A \times B \to [0,\ 2^k - 1]$ *slika.* ***Odsjecanje*** *je particija piksela slike* f *na objektne i pozadinske piksele, pri čemu je izlazna slika* g *dana s*

$$g(x,y) = \begin{cases} 1 & za\ f(x,y) > T \\ 0 & za\ f(x,y) \leq T \end{cases} \tag{4.1}$$

Funkciju $T : X \to [0,\ 2^k - 1]$ *zovemo* ***prag****. Kad je* T *konstantna na cijeloj slici, 4.1 nazivamo globalnim odsjecanjem, inače ga nazivamo varijabilnim odsjecanjem. Ako* T *kao argument prima prostorne koordinate piksela (*$X = A \times B$*), kažemo da se radi o dinamičkom ili adaptivnom odsjecanju.*

Za detekciju izoliranih točaka koristit ćemo Laplaceov operator implementiran pomoću maske 3.13 ili 3.14, s tim da filter ne smije biti premalen u odnosu na veličinu točke. Odziv maske u danom pikselu mjeri težinsku razliku u intenzitetu piksela i njegovih susjeda, a kako očekujemo da će intenzitet izolirane točke biti dosta drukčiji od intenziteta njenih susjeda, piksele s dovoljno jakim odzivom ćemo smatrati izoliranim točkama.



(a) krug s točkom u središtu (b) Laplaceova slika (c) segmentirana slika

Slika 4.2: Detekcija izolirane točke. Na sliku se prvo primijeni Laplaceov filter pa se dobivena slika metodom odsjecanja segmentira na točku i pozadinu.

Označimo s $R(x,y)$ odziv Laplaciana u pikselu (x,y). Slika nakon odsjecanja dana je s

$$g(x,y) = \begin{cases} 1 & \text{za } |R(x,y)| > T \\ 0 & \text{za } |R(x,y)| \leq T \end{cases} \tag{4.2}$$

Sad možemo kompaktno opisati kriterij za pronalazak izolirane točke na slici.

Definicija 4.1.2. *Neka je Laplaceova maska centrirana na mjestu (x,y) u slici f i neka je T nenegativan prag. Kažemo da je detektirana točka na mjestu (x,y) ako je apsolutna vrijednost odziva maske u (x,y) veća od T, tj. ako vrijedi $|R(x,y)| \geq T$.*

Konačna slika je binarna, a izolirane točke su u prednjem planu i označene bijelom bojom. Napominjemo da ova metoda djeluje samo u specijalnim slučajevima kad je točka dovoljno malena (npr. veličine piksela) i okružena susjedima homogenog intenziteta.

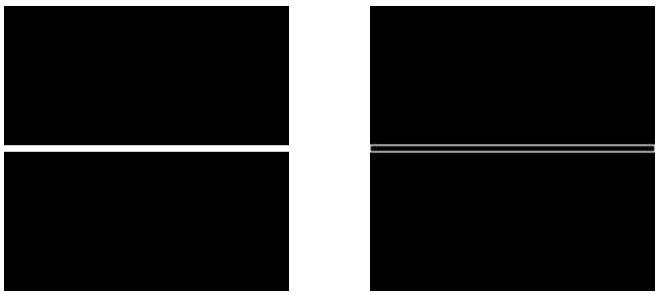
4.2 Detekcija linija

Očekujemo da će druga derivacija imati jači odziv i rezultirati tanjim linijama nego prva, stoga se ponovno odlučujemo za korištenje Laplaceovog filtera, no u ovom slučaju trebamo pripaziti na efekt dvostrukog ruba. Možemo uzeti apsolutnu vrijednost odziva, no time ćemo zadebljati linije. Želimo li izbjeći zadebljanje, možemo primijeniti odsjecanje te zadržati samo pozitivni (ili negativni) dio odziva. Na slici 4.3 su oba pristupa demonstrirana na jednostavnom primjeru - binarnoj slici s linijom debljine 1 piksel.

(a) (b) (c) (d)

Slika 4.3: (a) Horizontalna linija debljine jednog piksela. (b) Rezultat filtriranja slike (a). Vidljiv je efekt dvostrukog ruba - bijelu liniju koja predstavlja rub okružuju dvije crne linije. (c) Tretiranje dvostrukog ruba uzimanjem apsolutne vrijednosti ukupnog odziva. Dobivena linija je vidljivo deblja od originalne. (d) Tretiranje dvostrukog ruba zadržavanjem samo pozitivne vrijednosti odziva rezultira tanjom linijom.

U praksi je dosta slika šumovito, pa je korisnije odabrati pozitivni prag i time eliminirati



Slika 4.4: Slika linije debljine 6 piksela prije i nakon primjene 3×3 Laplaceovog filtera. Filter je premalen da detektira ovakvu liniju - on vidi unutrašnjost linije kao prostor konstantnog intenziteta, pa se u konačnom odzivu javlja crna crta u središtu linije (tzv. *"zero valley"*).

slučajne varijacije oko nule. Osim toga, slično kao i kod detekcije točaka, trebamo paziti da veličina filtera bude primjerena veličini linije. Deblje linije poput one sa slike 4.4 je ponekad bolje tretirati kao regije i pristupiti im drukčijom metodom.

Ponekad se na slikama s više linija većina njih pruža u jednom smjeru (npr. slike vlakana) ili nam je cilj izdvojiti samo linije specifičnog smjera (npr. dok tretiramo neke vrste šuma). Moguće je konstruirati masku tako da odziv bude najveći u pikselima kroz koje prolazi, primjerice, horizontalna linija. To se može postići pridavanjem veće težine u željenom smjeru. Međutim, potrebno je paziti da koeficijenti maske u zbroju budu 0 kako bi odziv u regijama konstantnog intenziteta bio 0 te da filter ostane valjana aproksimacija derivacije.

-1	2	-1
-1	2	-1
-1	2	-1

(a) 0° (vertikalna)

2	-1	-1
-1	2	-1
-1	-1	2

(b) 45°

-1	-1	-1
2	2	2
-1	-1	-1

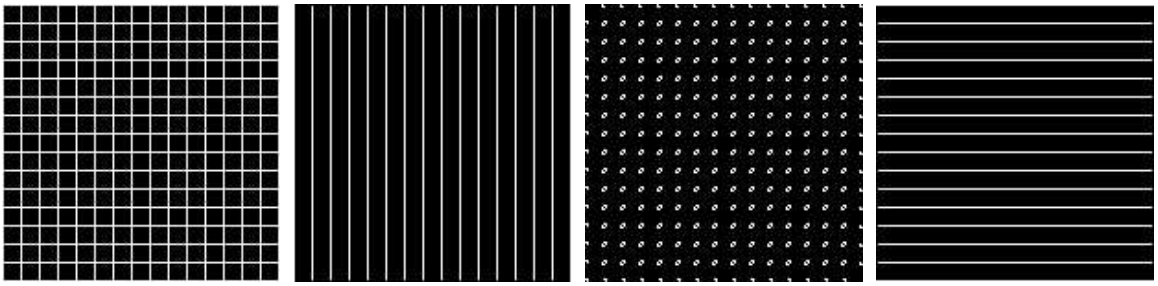
(c) 90° (horizontalna)

-1	-1	2
-1	2	-1
2	-1	-1

(d) −45°

Slika 4.5: Maske za 4 moguća smjera. Primijetimo da su nagibi uzeti u odnosu na x-os, s tim da je ishodište smješteno u gornji lijevi kut, kao u slici 2.1a.

Označimo redom s R_1, R_2, R_3 i R_4 odzive maski iz 4.5 izračunate pomoću 3.2 i filtrirajmo neku sliku svakom maskom zasebno. Ako je u danom pikselu odziv $|R_j| > |R_i|$ za sve $i \neq j$, piksel je vjerojatno dio linije u smjeru maske j. Kad nam je unaprijed zadan smjer na koji se trebamo fokusirati, onda filtriramo pripadnom maskom i na apsolutnu vrijednost njenog



(a) rešetka (b) odziv maske 4.5a (c) odziv maske 4.5b (d) odziv maske 4.5c

Slika 4.6: Koristeći maske čiji se odziv fokusira na specifični smjer, sa slike rešetke možemo izdvojiti horizontalne ili vertikalne linije. Maska 4.5b (konstruirana za detekciju dijagonala) neće biti osjetljiva na linije, ali će imati slab odziv u sjecištima.

odziva primijenimo odsjecanje.

Napomena 4.2.1. *Kako Laplaceov operator vraća i pozitivne i negativne vrijednosti, odziv operatora je potrebno skalirati radi prikaza slike. Želimo li zadržati intenzitete slike u specifičnom intervalu npr.* $[0, K]$*, koristimo sljedeće formule:*

$$f_m = f - \min(f), \tag{4.3}$$

koja vrati sliku f_m *čija je minimalna vrijednost* 0*, i nakon toga*

$$f_s = K\Big[f_m / \max(f_m)\Big], \tag{4.4}$$

koja vrati sliku f_s *čiji su intenziteti u rasponu* $[0, K]$*. K određuje maksimalnu vrijednost funkcije* f_s*, tako da za željeni interval* $[0, 1]$ *koristimo* $K = 1$*. Ako želimo dobiti k-bitnu sliku, kao K koristimo* $2^k - 1$*.* $\min(f)$ *je najmanji intenzitet na slici* f*, dok je* $\max(f)$ *najveći.*

Poglavlje 5

Detekcija rubova

5.1 Modeli rubova

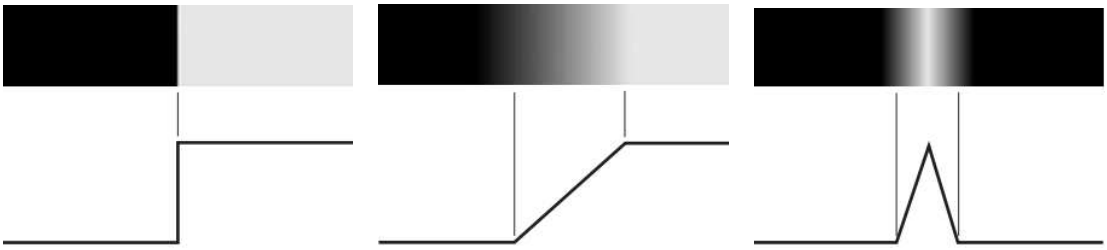
Na početku prethodnog poglavlja predstavili smo nekoliko modela ruba klasificiranih prema profilu intenziteta:

1. **stepenice** - rubovi sa stepenastim profilom intenziteta (5.1a). Prijelaz iz jednog intenziteta u drugi idealno se odvija duž jednog piksela, ali kako se zbog izglađivanja takav rub često "razmaže", kao stepenice ćemo klasificirati i kosine s jakim nagibom. Idealne stepenaste rubove nalazimo na računalno generiranom slikama za primjenu u animaciji, modeliranju čvrstih tijela i testiranju algoritama. Cannyjev detektor temeljit će se na ovom modelu ruba.

2. **kosine** - rubovi s profilom intenziteta kosine (5.1b). U praksi je dosta slika zamućeno i šumovito, s tim da zamućenost ovisi o ograničenjima mehanizma za fokusiranje (npr. leća), a šum pretežno ovisi o elektroničkim komponentama sustava za generiranje slike. Rubove na takvim slikama modeliramo kosinama, čiji je nagib obrnuto proporcionalan stupnju zamućenja ruba. U ovom slučaju su rubni pikseli svi pikseli koji su dio kosine, a segment ruba je bilo koji skup takvih točaka koje su povezane.

3. **krovovi** - rubovi s krovastim profilom intenziteta (5.1c). Širina ruba ovisi o debljini i oštrini tankog objekta. Ovakve rubove nalazimo u snimanju raspona, gdje se tanki objekti poput cijevi nalaze bliže senzoru nego njihova pozadina te zbog toga izgledaju dosta svjetlije od svoje pozadine. Ovaj model opisuje i rubove koje nalazimo pri digitalizaciji crteža i u satelitskim snimkama (npr. ceste).

Na slikama ponekad možemo naći sve tri vrste ruba. Premda zamućivanje i šum poremete idealne profile, rubovi na slikama dovoljne oštrine i s umjerenom količinom šuma nalikuju na opisane modele. Koristeći idealne modele možemo lakše matematički izraziti rubove,

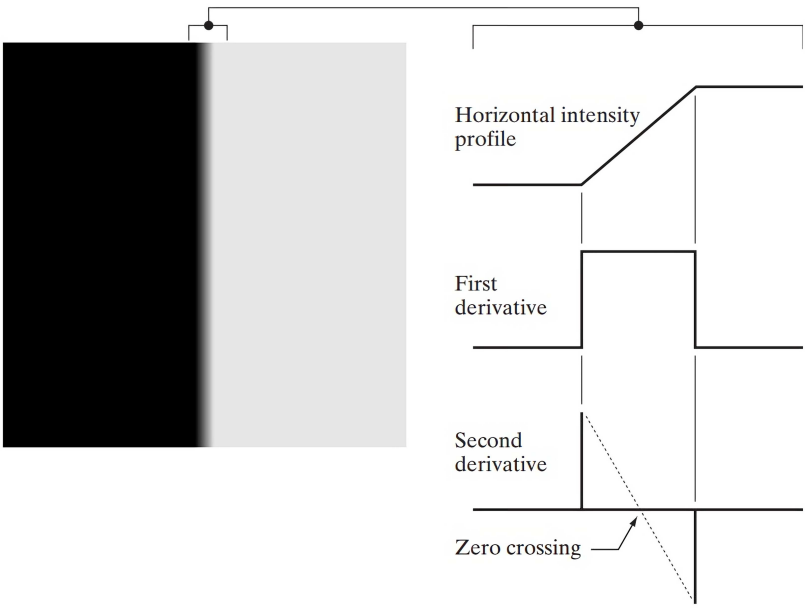


(a) (b) (c)

Slika 5.1: Model (a) stepenice, (b) kosine i (c) krova te njihovi profili intenziteta. Ilustracija je preuzeta iz [35].

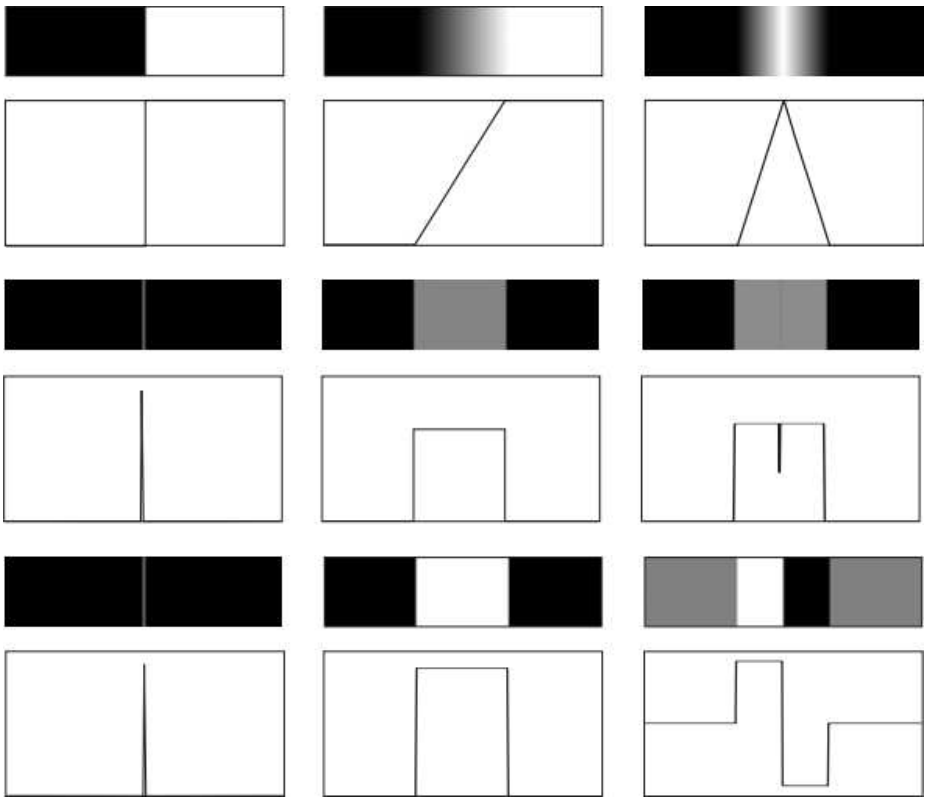
što se pokazalo dosta korisno u razvoju algoritama za obradu slike. Kvaliteta rezultata razvijenih algoritama ovisit će o tome koliko dobro modeli rubova aproksimiraju prave rubove na slici.

Pogledamo li profile intenziteta na slici 5.2, primijetit ćemo da je prva derivacija pozitivna na početku i duž kosine, a da je nula u područjima konstantnog intenziteta. Druga



Slika 5.2: Slijeva vidimo dvije regije konstantnog intenziteta razdijeljene idealnim vertikalnim kosinastim rubom. Zdesna vidimo horizontalni profil intenziteta zajedno s prvom i drugom derivacijom profila. Ilustracija je preuzeta iz [35].

derivacija je pozitivna na početku, negativna na kraju i nula duž kosine te područjima kons-

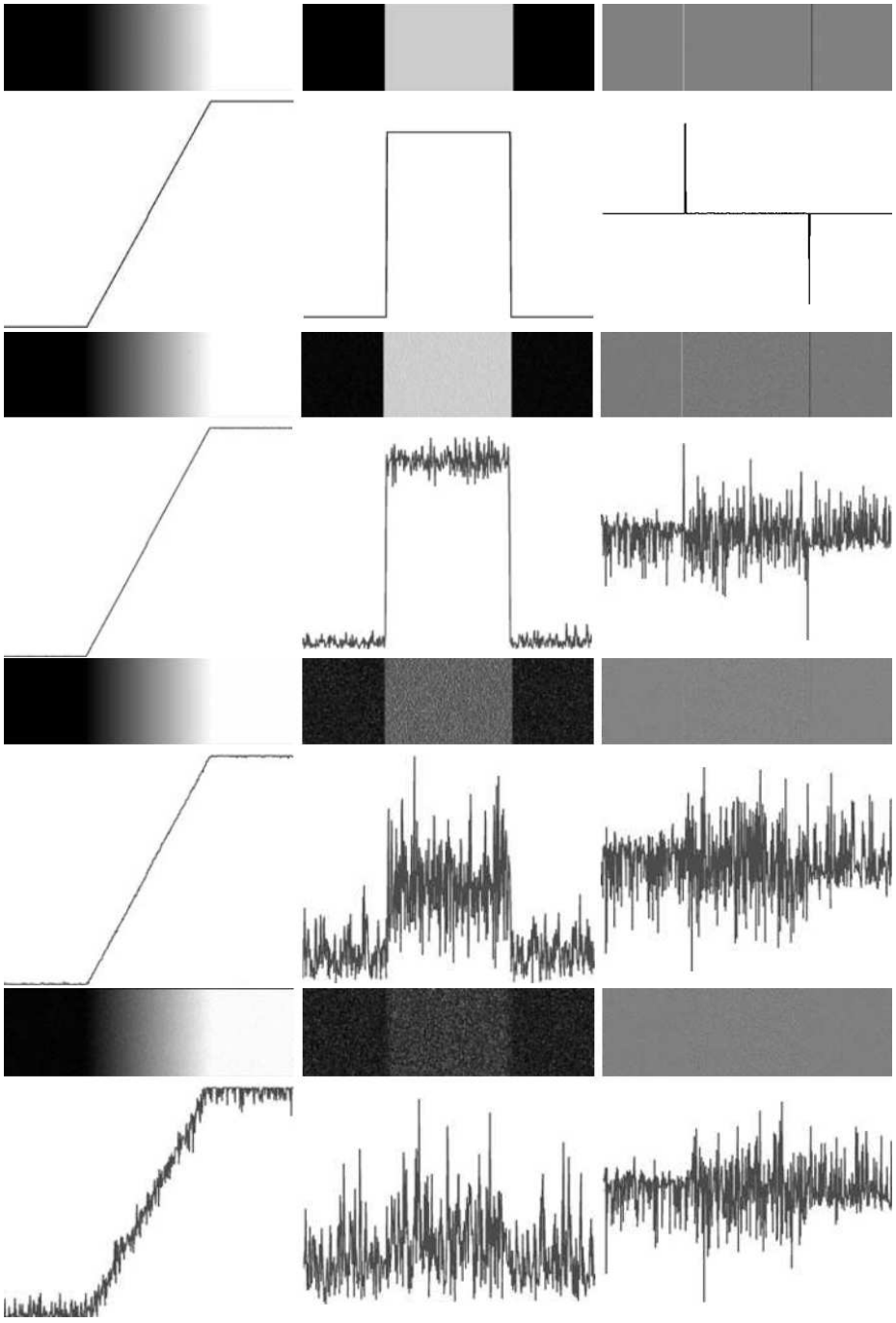


Slika 5.3: U prvom retku se nalaze slike rubova sa svojim profilima intenziteta, redom stepenica, kosina i krov. U drugom retku su slike i profili magnitude gradijenta. U zadnjem su retku prikazane slike i profili gradijentnih kuteva, svojstvo na kojem se temelji Cannyjev algoritam. Ilustracija je preuzeta iz [15].

tantnog intenziteta. U slučaju da smo imali prijelaz iz svijetlog u tamno područje, predznaci derivacija bi bili obrnuti. Presjek osi intenziteta nula i crte koja povezuje ekstreme druge derivacije zovemo **prijelaz kroz nulu** ili zero crossing druge derivacije.

Na slici 5.3 dodatno možemo vidjeti profile intenziteta za stepenicu i krov. Zaključujemo da pomoću magnitude prve derivacije možemo detektirati prisutnost ruba u pikselu slike, a predznak druge derivacije nam može otkriti nalazi li se piksel ruba na tamnoj ili svijetloj strani ruba. Ranije smo komentirali kako je druga derivacija u susretu s rubom popraćena efektom dvostrukom ruba i da njegovi prijelazi kroz nulu i ”zero valley” značajke mogu poslužiti za otkrivanje sredina debljih rubova, što je korisno kad želimo stanjiti rub.

Obratimo pažnju na sliku 5.4. U prvom stupcu vidimo kosinasti rub, u drugom stupcu njegovu prvu derivaciju, a u trećem drugu derivaciju. Prvi redak prikazuje profile inten-



Slika 5.4: U prvom stupcu se nalaze slike i profili intenziteta kosinastog ruba nakon dodavanja Gaussovog šuma sa srednjom vrijednosti $\mu = 0$ i standardnim devijacijama razina intenziteta $\sigma = 0.0, 0.1, 1.0, 10.0$. U drugom i trećem stupcu redom vidimo slike i profile intenziteta prve i druge derivacije. Ilustracija je preuzeta iz [35].

ziteta bez ikakvog šuma, a u svakom idućem retku se količina Gaussovog šuma[1] na slici pojača. Količina šuma nije značajna - na nekim slikama je jedva vidljiv golim okom.

Najgornja slika u drugom stupcu predstavlja prvu derivaciju. U područjima konstantnog intenziteta je nula i ta područja su prikazana u crnom, dok je duž rampe konstantna te je taj dio prikazan u sivom. Kako se originalnoj slici dodaje sve više šuma, primjećujemo da se profil i slika prve derivacije značajno mijenjaju te je posljednju sliku i profil u stupcu teško prepoznati kao prvu derivaciju kosinastog ruba.

Druga derivacija se mijenja još brže u odnosu na količinu šuma. Na prvoj slici trećeg stupca vidimo sliku druge derivacije - područje na kojem poprima nulu je sive boje (rezultat skaliranja), a crte koje označavaju početak i kraj kosine su redom bijele i crne boje. Na idućoj slici u stupcu je jedva vidljivo da se radi o drugoj derivaciji, dok je ona na donje dvije slike neprepoznatljiva te joj se pozitivna i negativna komponenta ne mogu očitati.

Iz ovog primjera je jasno koliko su derivacije osjetljive na šum, što predstavlja problem koji je nužno riješiti prije nego počnemo koristiti odziv derivacija za pronalazak ruba. Kao što smo komentirali u trećem poglavlju, šum možemo djelomično odstraniti izglađivanjem slike.

Naposljetku predstavimo osnovne korake detekcije ruba:

1. izglađivanje slike radi smanjenja šuma
2. detekcija točaka ruba, tj. pronalazak kandidata za rub
3. lokalizacija ruba, odnosno izgradnja ruba odabirom piksela među kandidatima za rub

5.2 Osnovna detekcija ruba

U ovoj sekciji ćemo predstaviti prvu derivaciju kao alat za detekciju ruba. U poglavlju o filterima smo definirali gradijent slike f u pikselu (x, y) na sljedeći način:

$$\nabla f := \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

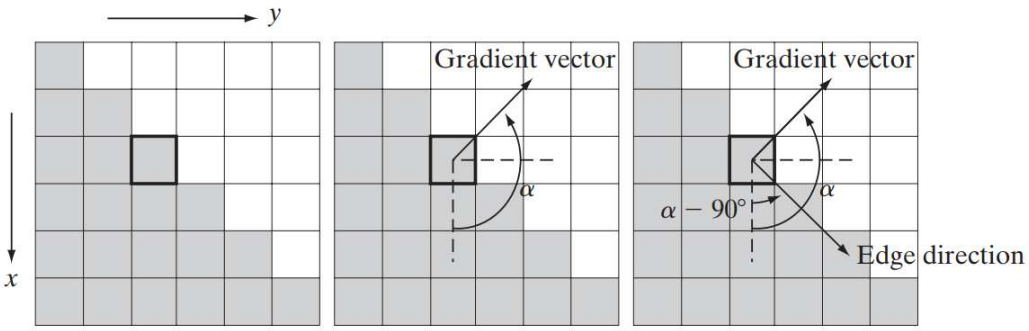
Magnitudu gradijenta smo definirali kao $M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$. Primijetimo da su g_x, g_y i $M(x, y)$ slike iste veličine kao original koje kreiramo putovanjem po svim pikselima u f. Osim magnitude, zanima nas i smjer gradijenta u danom pikselu.

[1] Gaussov šum je šum čija je funkcija gustoće vjerojatnosti Gaussova distribucija ([25]).

Definicija 5.2.1. *Neka je f slika. Neka je $p = (x, y)$ piksel slike f i neka je ∇f gradijent slike f. Smjer gradijentnog vektora u odnosu na x-os u pikselu p definiramo kao*

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{5.1}$$

$\alpha(x, y)$ je također slika veličine originalne slike f. Smjer ruba u pikselu (x, y) je ortogonalan smjeru gradijentnog vektora, tj. iznosi $\alpha(x, y) - 90°$.



Slika 5.5: Određivanje jačine i smjera ruba u pikselu pomoću gradijenta. Ilustracija je preuzeta iz [35].

Primjer 5.2.2. ***(Računanje digitalnog gradijenta):*** *Na slici 5.5 vidimo uvećani isječak slike s ravnim, dijagonalnim segmentom ruba. Kvadratići na slici predstavljaju piksele, a nas zanima jačina i smjer ruba u podebljanom pikselu koji ćemo označiti s p. Sivi pikseli imaju intenzitet 0, a bijeli pikseli imaju intenzitet 1. Centrirat ćemo 3×3 okolinu u p i izračunati parcijalnu derivaciju u x smjeru oduzimanjem gornjeg reda piksela od donjeg, a parcijalnu derivaciju u y smjeru oduzimanjem piksela lijevog stupca od piksela desnog stupca. Koristeći opisane diferencije kao aproksimacije parcijalnih derivacija, dobijemo $\frac{\partial f}{\partial x} = -2$ i $\frac{\partial f}{\partial x} = 2$, tj.*

$$\nabla f := \operatorname{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

Magnituda u tom pikselu onda iznosi $M(x, y) = 2\sqrt{2}$, a smjer gradijentnog vektora iznosi $\alpha(x, y) = \tan^{-1}(g_y/g_x) = -45° = 135°$. Konačno, na slici 5.5 vidimo da je smjer ruba okomit na smjer gradijenta i da je njegov smjer $\alpha - 90° = 45°$. Svi rubni pikseli imaju isti gradijent, pa je čitavi segment ruba u istom smjeru. Vektor gradijenta nazivamo normalom ruba, a kad ga normaliziramo tako da ga podijelimo njegovom magnitudom, rezultantni vektor nazivamo jediničnom normalom ruba.

Gradijentni operatori

Gradijent dobijemo računanjem parcijalnih derivacija u svakom pikselu na slici. Derivacije smo već ranije aproksimirali s 3.8 i 3.9, tj.

$$g_x = \partial_x f(x,y) = f(x+1,y) - f(x,y)$$
$$g_y = \partial_y f(x,y) = f(x,y+1) - f(x,y)$$

Ove jednadžbe i pripadne maske (3.13) možemo koristiti za detekciju horizontalnih i vertikalnih rubova. Kad nas zanimaju dijagonalni rubovi, potrebna nam je 2D maska - Robertsovi operatori križnog gradijenta (vidi 3.14) su jedan od ranih pokušaja konstrukcije maske prilagođene dijagonalama. Međutim, u poglavlju o filterima smo komentirali da je malo nezgodno implementirati filtere parnih dimenzija, što je motiviralo uvođenje Prewittinih, a zatim i Sobelovih maski. Sobelove maske su malo teže za implementirati nego Prewittine maske, ali su zato bolje u prigušivanju šuma, što smo prepoznali kao važno svojstvo u radu s derivacijama. Dalje ćemo se većinski služiti Sobelovim operatorima 3.14e i 3.14f, imajući na umu da imaju najjači odziv u doticaju s horizontalnim i vertikalnim rubovima.

0	1	1
−1	0	1
−1	−1	0

(a)

−1	−1	0
−1	0	1
0	1	1

(b)

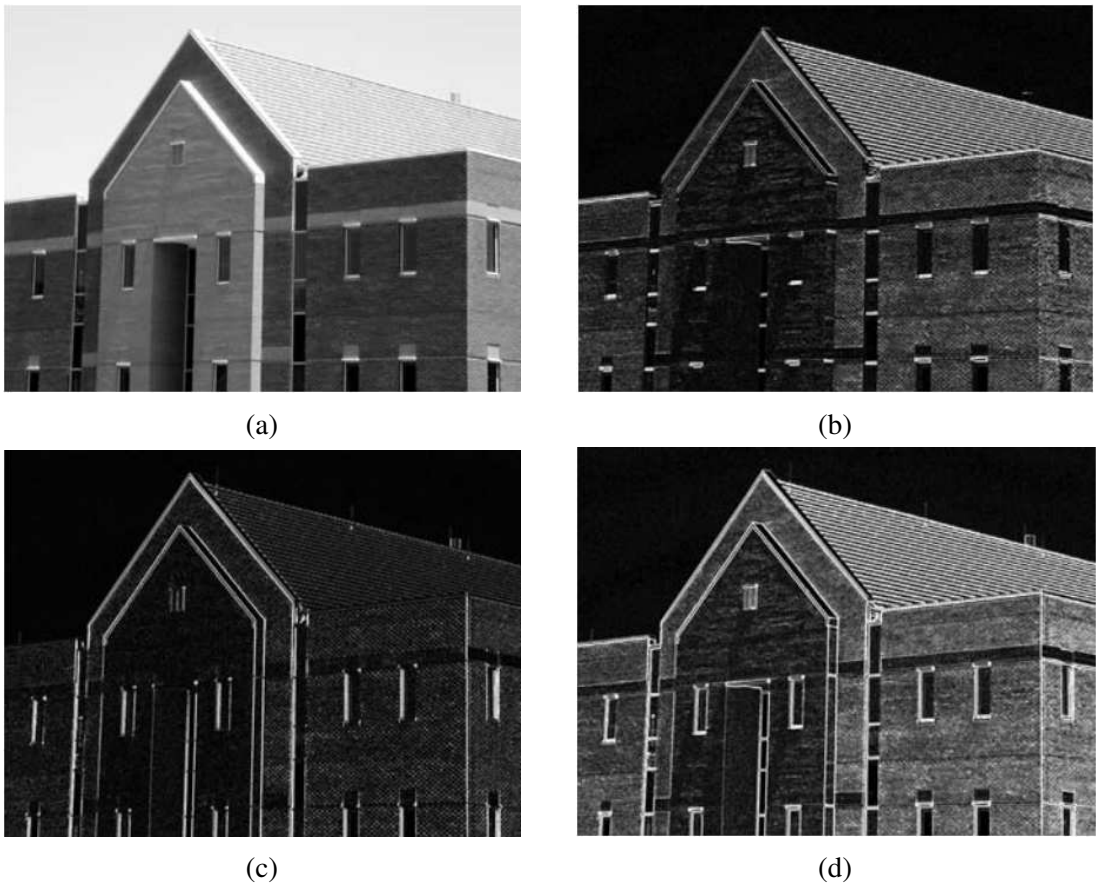
0	1	2
−1	0	1
−2	−1	0

(c)

−2	−1	0
−1	0	1
0	1	2

(d)

Slika 5.6: Dijagonalni Prewittini operatori za smjerove (a) −45° i (b) 45°. Dijagonalni Sobelovi operatori za smjerove (c) −45° i (d) 45°.



(a) (b)

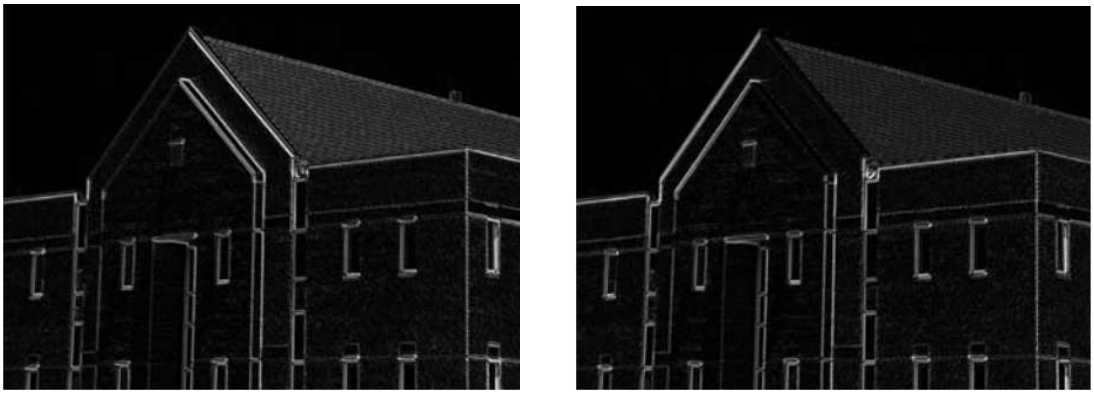
(c) (d)

Slika 5.7: (a) Slika zgrade. (b) $|g_x|$ - komponenta gradijenta u x smjeru dobivena Sobelovom maskom. (c) $|g_y|$ - komponenta gradijenta u y smjeru dobivena Sobelovom maskom. (d) Gradijentna slika $|g_x| + |g_y|$. Slike su preuzete iz [35].

Moguće je konstruirati 3×3 maske s najsnažnijim odzivom duž dijagonala, npr. maske iz 5.6. Naime, na slikama 5.7 i 5.9 se vidi da horizontalne i vertikalne Sobelove maske ne razlikuju između rubova u smjerovima $\pm 45°$, a ponekad je poželjno fokusirati se na te smjerove. Slike 5.8a i 5.8b pokazat će apsolutne vrijednosti odziva dijagonalnih Sobelovih maski. Oba filtera imaju sličan odziv u doticaju s vertikalnim i horizontalnim rubovima, ali je taj odziv svakako slabiji od odziva filtera koji su prilagođeni za ta dva smjera.

Pomoću dosad spomenutih maski računamo komponente gradijenta g_x i g_y u svakom pikselu slike f, zatim pomoću njih aproksimiramo jačinu i smjer ruba. Magnitudu gradijenta, koja određuje jačinu ruba, definirali smo na sljedeći način:

$$M(x, y) = \mathrm{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}.$$



(a) (b)

Slika 5.8: Detekcija dijagonalnih rubova pomoću Sobelovih maski (a) 5.6c i (b) 5.6d. Slike su preuzete iz [35].

Kako nije računski poželjno koristiti korijene i kvadrate, koristit ćemo ranije uvedenu aproksimaciju magnitude apsolutnim vrijednostima, tj.

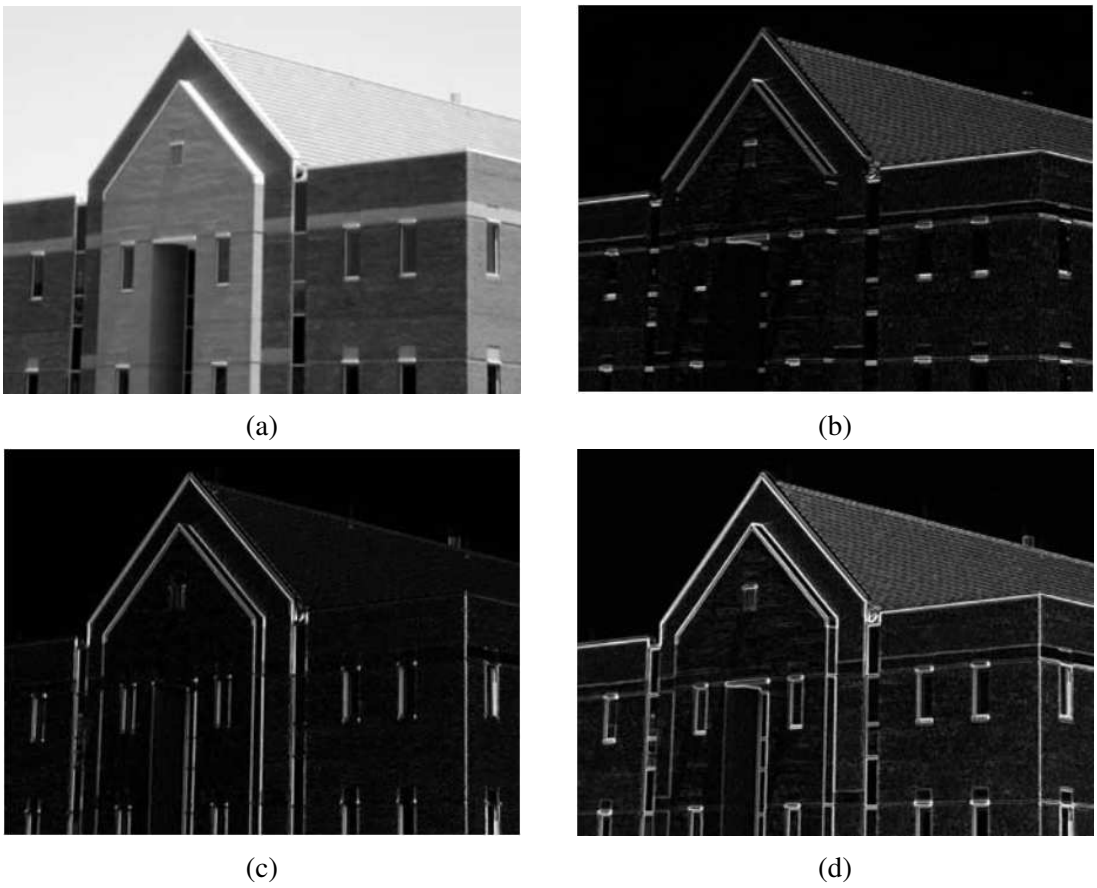
$$M(x,y) \approx |g_x| + |g_y|.$$

Ova je formula računski manje skupa od formule iz definicije i štiti relativne promjene u razinama intenziteta. Zauzvrat se odričemo svojstva izotropnosti, ali to ne predstavlja veliki problem jer su Sobelove maske (koje ćemo primarno koristiti) izotropne samo za vertikalne i horizontalne rubove, a njihov odziv je za obje formule isti.

Napomena 5.2.3. *Sobelova maska daje jednaki odziv neovisno o tome koristimo li definiciju magnitude (3.19) ili njenu aproksimaciju (3.20). Naime, lako se provjeri da za Sobelovu masku vrijedi $g_x = 0$ u susretu s vertikalnim rubovima, a $g_y = 0$ u susretu s horizontalnim rubovima. Na primjer, u potonjem slučaju bismo dobili $M(x,y) = \sqrt{g_x^2 + g_y^2} = \sqrt{g_x^2} = |g_x|$. Isto vrijedi i za Prewittinu masku.*

Primjer 5.2.4. ***(Uklanjanje suvišnih detalja):*** *Slika zgrade 5.7a je dovoljno visoke rezolucije i doprinos cigli na zidovima i krovu je dovoljno značajan da ti detalji ostanu sačuvani na gradijentnoj slici. U detekciji ruba su ovakvi detalji obično nepoželjni jer se ponašaju kao šum te ih računi s derivacijama mogu pojačati i time zakomplicirati detekciju glavnih rubova na slici. Jedan od načina za umanjiti količinu finog detalja i pojačati selektivnost detektora je izgladiti sliku. Na slici 5.9 vidimo rezultat detekcije ruba nakon filtriranja 5×5 box filterom. Odziv svake maske čuva glavni rub, a količina detalja je značajno umanjena.*

Primjer 5.2.5. *Za pojačanje selektivnosti detektora ruba možemo koristiti i metodu odsjecanja. Na slici 5.10a vidimo rezultat odsjecanja gradijentne slike 5.7d. Kao prag je*



(a) (b)

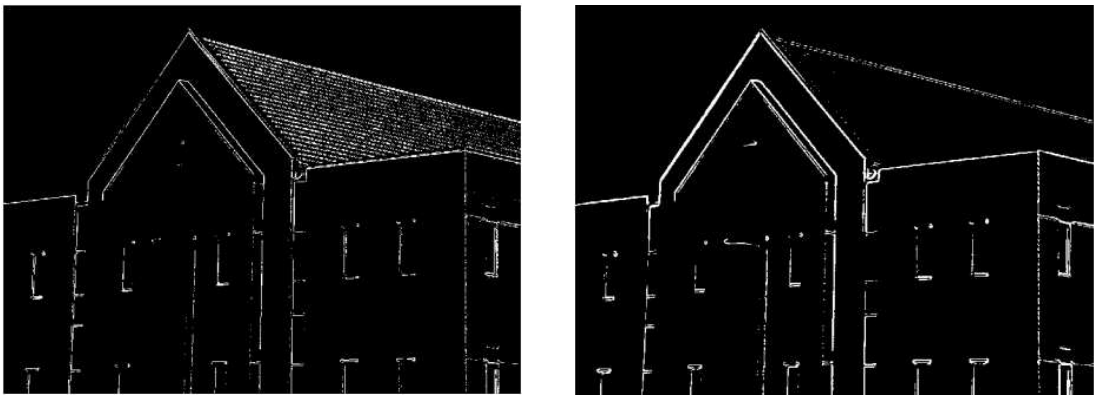
(c) (d)

Slika 5.9: (a) Originalna slika je izglađena 5×5 box filterom. Računamo (b), (c) i (d) kao i ranije (5.7). Ovaj put su na gradijentnoj slici očuvani glavni rubovi, a zbog izglađivanja je izgubljen dio sitnih detalja. Slike su preuzete iz [35].

korišteno 33% najviše vrijednosti intenziteta - pikseli slabijeg intenziteta od praga su na izlazu crne boje. Usporedimo rezultat sa slikom 5.9d: na slici nakon odsjecanja je manje rubova i rubovi su oštriji, ali su brojni rubovi isprekidani. Kako bi se očuvala povezanost, standardno je kombinirati odsjecanje s izglađivanjem, kao što smo učinili na slici 5.10b. Neki rubovi koje izglađivanje stanji će potpuno nestati sa slike nakon odsjecanja, pa trebamo pažljivo birati parametre u slučaju da ih želimo očuvati.

Zaključujemo da je u računanju gradijentnih slika dobra praksa izgladiti sliku box filterom - veličina filtera, odnosno stupanj zamućivanja ovisi o šumovitosti i količini detalja koju želimo sačuvati. Tada računamo gradijentnu sliku koristeći formulu aproksimacije magnitude apsolutnim vrijednostima - naime, ta formula je računski isplativija, a kao što smo komentirali u napomeni 5.2.3, nećemo izgubiti izotropnost ako koristimo Sobelovu masku.

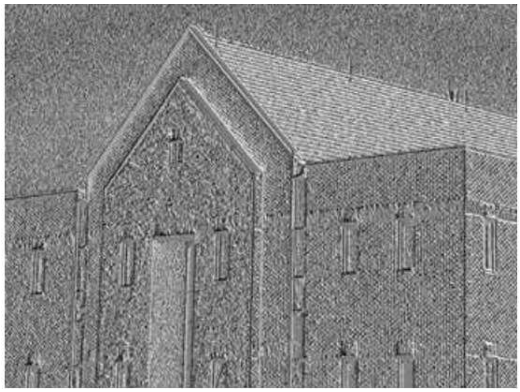
Konačno, ako u gradijentu ostane previše detalja, možemo ih ukloniti metodom odsjecanja, a prag biramo ovisno o tome koliko rubova želimo ukloniti.



(a) (b)

Slika 5.10: Rezultat odsjecanja (a) gradijentne slike originala (5.7d) i (b) gradijentne slike nakon izglađivanja (5.9d). Kao prag je korišteno 33% najvišeg intenziteta na slici. Vidimo da je kombinacija izglađivanja i odsjecanja učinkovita u segmentiranju glavnih rubova. Slike su preuzete iz [35].

Dosad smo se fokusirali na jačinu ruba, a sad se nakratko obazrimo na smjer. Na slici 5.11 vidimo primjer slike gradijentnih kuteva. Ovakve slike nam općenito nisu korisne za detekciju ruba kao gradijentne slike (tj. slike magnitude gradijenta), ali kao što ćemo vidjeti kasnije u poglavlju, kutevi će biti ključni u implementaciji Cannyjevog algoritma



Slika 5.11: Slika gradijentnih kuteva dobivena primjenom formule 5.1 na sliku zgrade 5.7a. Područja konstantnog intenziteta na slici ukazuju na to da je smjer gradijentnog vektora isti za svaki piksel u regiji. Slika je preuzeta iz [35].

detekcije ruba.

U ovom potpoglavlju smo predstavili postupak detekcije rubova na slici na temelju prve derivacije. Pokazali smo kako olakšati pronalazak dobrih kandidata za rub i kako paziti da se očuva povezanost. Sljedeće metode koje ćemo predstaviti su malo sofisticiranije i temelje se na drugoj derivaciji, a ti će se algoritmi pokazati robusnijima i učinkovitijima.

5.3 Marr-Hildrethin detektor ruba

Algoritam koji su razvili Marr[2] i Hildreth[3] jedan je od prvih uspješnih sofisticiranijih algoritama detekcije ruba. Ranije metode su koristile male operatore poput Sobelovih maski, no Marr i Hildreth su tvrdili da promjene u intenzitetu ovise o mjerilu slike, stoga je primjerenije koristiti operatore različitih veličina. Također su tvrdili da će nagle promjene u intenzitetu uzrokovati šiljke ili jarke u prvoj derivaciji ili prijelaz kroz nulu u drugoj derivaciji pa su vođeni ovim dvjema idejama razvili novi algoritam.

Prema njima, operator detekcije ruba treba biti diferencijalni operator koji može izračunati digitalnu aproksimaciju prve ili druge derivacije u svakom pikselu slike. Nadalje, operator bi se trebao moći podesiti tako da funkcionira za bilo koje mjerilo pa da veći operatori detektiraju mutnije rubove, a manji detektiraju izoštrene, finije detalje. Laplacian Gaussijana, ili LoG operator, zadovoljava oba uvjeta. Prisjetimo se, LoG operator u pikselu (x, y) zadan je sljedećom formulom:

$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Uzorkovanjem LoG funkcije i skaliranjem koeficijenata možemo dobiti masku proizvoljne veličine (poput maske 3.12b), ali je učinkovitije uzorkovati Gaussovu funkciju, tj.

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

do željene veličine (recimo $n \times n$) i dobiveni rezultat konvoluirati s Laplaceovom maskom. Međutim, potrebno je paziti da zbroj koeficijenata konačne maske bude nula kako bi operator bio valjan.

Napomena 5.3.1. *Rezultat konvolucije u pikselu (x, y) je zbroj produkata koeficijenata maske s pripadnim intenzitetima piksela iz okoline centrirane u tom pikselu. Konvoluiranjem*

[2]David Marr, *Theory of Edge Detection*
[3]Ellen C. Hildreth, *Theory of Edge Detection*

maske s čitavom slikom, svaki element maske se točno jednom množi sa svakim pikselom na slici. Kako je zbroj koeficijenata maske nula, zbroj produkata svih koeficijenata maske s intenzitetom istog piksela je također nula, a budući da ovo vrijedi za svaki piksel, konačni zbroj će također biti nula.

LoG operator dobro funkcionira jer Gaussovom funkcijom zamutimo sliku i time ublažimo intenzitet struktura na slici (uključujući šum). Za razliku od box filtera, Gaussian je glatka funkcija i u prostornoj i u frekvencijskoj domeni, pa je manje vjerojatno da će uzrokovati artefakte (poput prstenova), odnosno oštećenja koja ranije nisu bila prisutna na slici. S druge strane, Laplacian je izotropan[4] pa jednako reagira na promjene u intenzitetu u bilo kojem smjeru maske, čime izbjegavamo konstruiranje i korištenje više različitih maski kako bismo našli smjer maksimalnog odziva.

Ukratko, Marr-Hildrethin algoritam funkcionira tako da konvoluiramo filter $\nabla^2 G$ s ulaznom slikom $f(x,y)$

$$g(x,y) = [\nabla^2 G(x,y)] \star f(x,y), \tag{5.2}$$

pa pomoću prijelaza kroz nulu slike $g(x,y)$ određujemo lokacije rubova na slici $f(x,y)$. Budući da su operacije u 5.2 linearne, možemo jednadžbu zapisati kao

$$g(x,y) = \nabla^2 [G(x,y) \star f(x,y)], \tag{5.3}$$

što znači da smijemo prvo izgladiti sliku s Gaussovim filterom i onda izračunati Laplacian rezultata.

Sažmimo korake Marr-Hildrethinog algoritma:

1. filtrirati ulaznu sliku s $n \times n$ niskopropusnim Gaussovim filterom koji se dobije uzorkovanjem Gaussove funkcije

2. izračunati Laplacian dobivene slike koristeći neku od maski predstavljenih u 3.10

3. pronaći prijelaze kroz nulu slike dobivene u drugom koraku

Pri odabiru veličine Gaussovog filtera koristimo činjenicu da se oko 99.7% volumena ispod 2D Gaussiana nalazi u rasponu od $\pm 3\sigma$ oko srednje vrijednosti. Stoga koristimo $n \times n$ LoG filter, a za n uzimamo najmanji neparni prirodni broj veći ili jednak 6σ. Odabir manjeg

[4]Marr tvrdi da je izotropnost u skladu s karakteristikama ljudskog vidnog sustava - naime, receptivna polja ganglija u mrežnici su kružno simetrična s podražajnom (ekscitatornom) sredinom i inhibicijskim okruženjem. Fiziološka mjerenja sugeriraju podudaranja reakcije središnjih stanica s pozitivnim, a reakcije stanica iz okruženja s negativnim vrijednostima negativne LoG funkcije (3.12). Iscrpnije objašnjenje može se naći u [22].

n će odrezati dio LoG funkcije, s tim da je stupanj odrezanosti inverzno proporcionalan veličini maske; odabirom većeg n nećemo puno utjecati na rezultat.

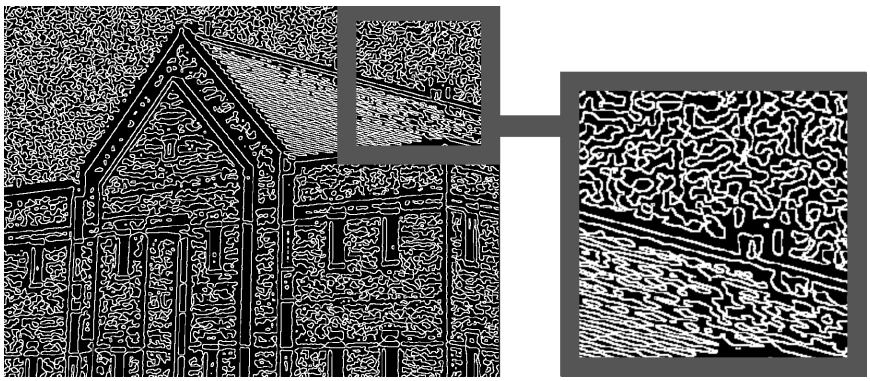
Zbog šuma i računskih grešaka, nećemo zahtijevati da piksel (x, y) zadovoljava $g(x, y) = 0$ kako bismo ga proglasili točkom prijelaza kroz nulu. Umjesto toga ćemo provjeravati 3×3 susjedstvo svakog piksela i tražiti parove suprotnih susjeda (npr. sjeverni-južni, sjeveroistočni-jugozapadni) s različitim predznacima - nađemo li barem jedan takav par, piksel proglašavamo točkom prijelaza kroz nulu. Često se zahtijeva da apsolutna vrijednost razlike intenziteta piksela u paru prijeđe neki zadani pozitivni prag kako bismo izbjegli stvaranje zatvorenih petlji na slici.

Slika 5.12: $g_P(x, y)$ slike zgrade nakon prva dva koraka Marr-Hildrethinog algoritma. Na povećanom isječku vidimo povezane komponente.

Primjer 5.3.2. *Granice povezanih komponenti činit će zatvorene petlje. Označimo s $g_L(x, y)$ LoG sliku - ona ima i pozitivne i negativne vrijednosti, a prijelazi preko nule su definirani kao ranije. Neka je $g_P(x, y)$ binarna slika čiji prednji plan čine samo pikseli pozitivnog intenziteta LoG slike. Na slici 5.12 vidimo $g_P(x, y)$ slike 5.14b. Slika sadrži povezane komponente, a svaki piksel na rubu povezane komponente je ili točka prijelaza između negativne i pozitivne vrijednosti, ili je susjedan točki prijelaza - dakle sve točke prijelaza kroz nulu su susjedne tim rubovima. Međutim, granice povezanih komponenti čine zatvoreni put jer (a) između svaka dva piksela u povezanoj komponenti postoji put i (b) pikseli koje čine granicu povezane komponente su dio povezane komponente.*

Na 5.13 vidimo prijelaze kroz nulu dobivene uz prag 0. Primijetimo da rubovi tvore zatvorene petlje, što se naziva "špageti" efektom[5].

[5]Ovaj efekt nije uvijek nepoželjan. Na primjer, u generiranju slika mozga ([31]), zatvorene petlje mogu pomoći u identifikaciji oblika 3D struktura s 2D slika, osobito pri radu s hrpom 2D slika (tzv. "slices") nastalih uslijed generiranja 3D slika.



Slika 5.13: Prijelazi kroz nulu na slici uz prag 0. Na povećanom isječku sa slike može se vidjeti spomenuti "špageti" efekt.

Primjer 5.3.3. ***(Marr-Hildrethina detekcija ruba):*** *Na slici 5.14b vidimo rezultat prvog i drugog koraka Marr-Hildrethinog algoritma uz parametre* $\sigma = 4$ *i* $n = 25$*. Slika je većim dijelom siva zbog skaliranja. Nadalje, na slici 5.14c kao prag koristimo* 4% *maksimalne vrijednosti LoG slike. Vidimo da su glavni rubovi očuvani, dok su manje zanimljive značajke poput dijelova krova i cigli filtrirani sa slike. Jedna važna posljedica korištenja prijelaza kroz nulu za detekciju ruba je ta što su rubovi debljine jedan piksel, što će olakšati kasnije stadije obrade i segmentacije, osobito povezivanja.*

Možemo uzeti u obzir promjene u intenzitetu ovisne o mjerilu korištenjem različitih vrijednosti σ, zatim zadržavanjem samo piksela ruba koji su zajednički svim rezultatima. U praksi se zbog svoje kompleksnosti ovaj pristup izbjegava ili koristi samo za odabir prihvatljivog σ za dani filter.

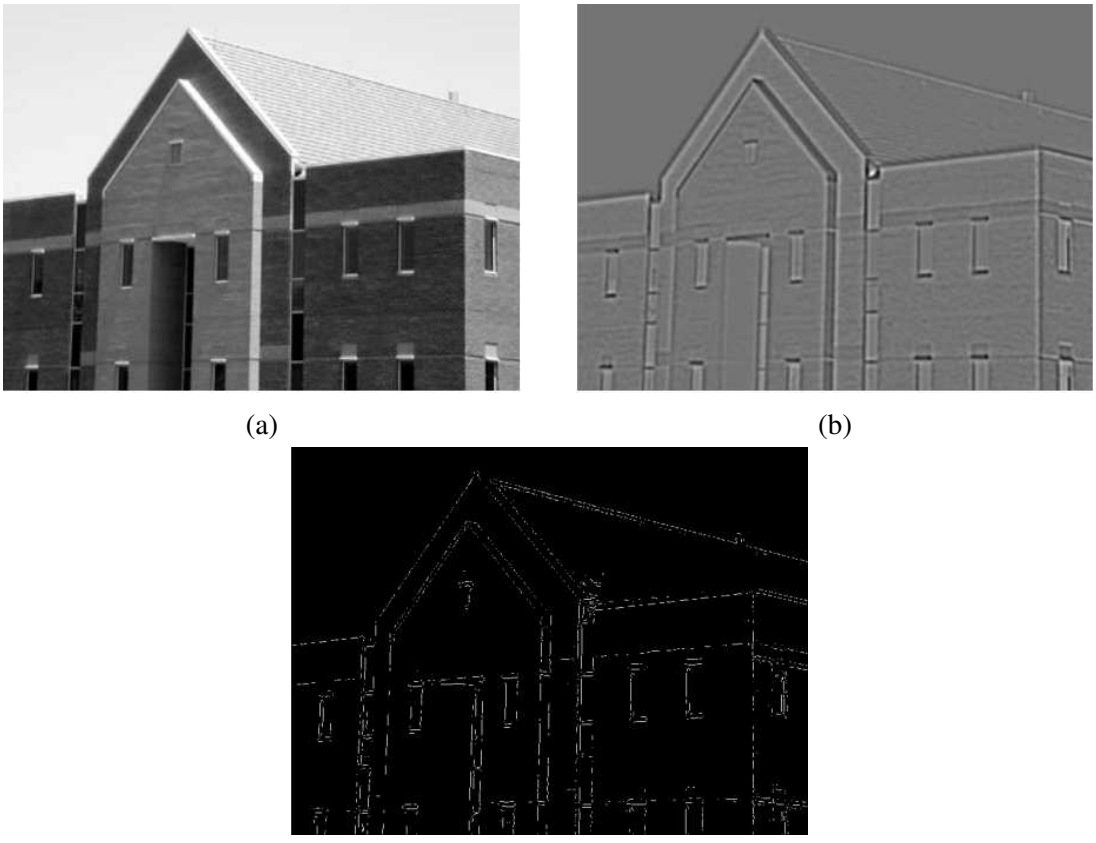
LoG slike je zahtjevan za izračunati, ali LoG se može vrlo dobro aproksimirati puno bržim operatorom **razlike Gaussijana**[6], odnosno DoG (*"difference of Gaussians"*) operatorom:

$$\mathrm{DoG}(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}, \tag{5.4}$$

gdje je $\sigma_1 > \sigma_2$[7]. Marr i Hildreth ([21]) argumentiraju da je LoG operator limes DoG operatora kako veličine dva Gaussijana teže jedna prema drugoj i da se najbolja aproksimacija postiže za omjer $K = \sigma_1 : \sigma_2 = 1.6 : 1$. Kako bismo lakše usporedili LoG i DoG operatore,

[6]Iako je Gaussian niskopropusni filter, razlika Gaussiana je visokopropusni filter. Visokopropusni filteri oslabe niske frekvencije, a visoke ne mijenjaju; rubovi sadrže dosta visokih frekvencija, pa je efekt korištenja visokopropusnog filtera naglašavanje i detekcija rubova.

[7]Marr i Hildreth ([21], [22]) tvrde da σ_1 možemo razumjeti kao inhibicijsku, a σ_2 kao ekscitatornu prostornu konstantu, s tim da su nazivi dodijeljeni na osnovi stanica koje "nose" određeni dio funkcije.



(a) (b)

(c)

Slika 5.14: (a) Slika zgrade, skalirana na $[0, 1]$. (b) Rezultat prvog i drugog koraka Marr-Hildrethinog algoritma uz $\sigma = 4$ i $\sigma = 25$. (c) Prijelazi kroz nulu na slici (b), s tim da je prag 4% najvećeg intenziteta. Slike su preuzete iz [35].

pazimo da je prostorna konstanta σ LoG operatora odabrana tako da oba operatora imaju iste prijelaze kroz nulu. Izjednačimo li 5.4 s nulom, dobit ćemo

$$\frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} = \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

Logaritmirajmo izraze s obje strane i sredimo izraze:

$$\ln\left[\frac{1}{2\pi\sigma_1^2}\right] - \frac{x^2+y^2}{2\sigma_1^2} = \ln\left[\frac{1}{2\pi\sigma_2^2}\right] - \frac{x^2+y^2}{2\sigma_2^2}$$

$$(x^2+y^2)\left[\frac{1}{2\sigma_1^2} - \frac{1}{2\sigma_2^2}\right] = \ln\left[\frac{1}{2\pi\sigma_1^2}\right] - \ln\left[\frac{1}{2\pi\sigma_2^2}\right] = \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right]$$

$\nabla^2 G(x,y)$ je jednaka nuli kad vrijedi $x^2 + y^2 = 2\sigma^2$, pa supstituirajmo izraz u prethodnu jednadžbu:

$$\sigma^2\left[\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2}\right] = \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right]$$

Konačan izbor σ za LoG operator glasi:

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2}\ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right] \tag{5.5}$$

Iako će se sad prijelazi kroz nulu operatora preklapati, njihove amplitude će se razlikovati, pa je potrebno obje funkcije skalirati tako da poprimaju istu vrijednost u ishodištu.

LoG i DoG operatori se mogu implementirati pomoću 1D konvolucija umjesto 2D konvolucija (za više detalja, vidi [35]). Za sliku veličine $M \times N$ i filter veličine $n \times n$, broj operacija zbrajanja i množenja je proporcionalan n^2MN u slučaju 2D konvolucija, a tek nMN u slučaju 1D konvolucija.

5.4 Cannyjev detektor ruba

Cannyjev detektor ruba je kompleksniji, ali učinkovitiji od dosad predstavljenih detektora ruba. Pri konstrukciji detektora, Canny[8] je postavio tri cilja:

1. dobra detekcija - potrebno je pronaći sve prave točke ruba i izbjeći lažne odzive
2. dobra lokalizacija - udaljenost između točaka ruba i središta pravog ruba treba biti što manja
3. jednostruki odziv - detektor bi trebao vratiti jedan piksel za svaki pravi piksel ruba, tj. broj lokalnih maksimuma oko pravog ruba treba biti što manji

Nije jednostavno konstruirati detektor koji zadovoljava sva tri svojstva (optimalno), ali numerička optimizacija s 1D stepenastim rubovima pokazala je da se optimalni detektor takvih rubova može aproksimirati prvom derivacijom Gaussove funkcije:

$$\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} \tag{5.6}$$

Učinkovitost prve derivacije Gaussiana je prema Cannyju oko 20% lošija od optimalnog operatora, no kako je teško uočiti razliku uspoređujući efekt oba operatora na slici i kako su

[8] John F. Canny, *A Computational Approach To Edge Detection* ([5])

derivacije Gaussiana jednostavne za računati u 2D-u, prva derivacija Gaussiana se smatra vrlo dobrim kandidatom za operator.

1D pristup se oslanja na normalu ruba, a kako nam smjer normale nije unaprijed poznat, trebali bismo primijeniti 1D detektor u svim mogućim smjerovima. Kako bismo to izbjegli, slika se prvo izgladi 2D Gaussovom funkcijom (koja je kružna) pa se računa gradijent izglađene slike te se na kraju pomoću magnitude i smjera gradijenta procijene jačina i smjer ruba u svakom pikselu.

Neka je $f(x,y)$ ulazna slika i neka je $G(x,y)$ Gaussova funkcija, tj.

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{5.7}$$

Konvoluiranjem G s f dobijemo izglađenu sliku f_s:

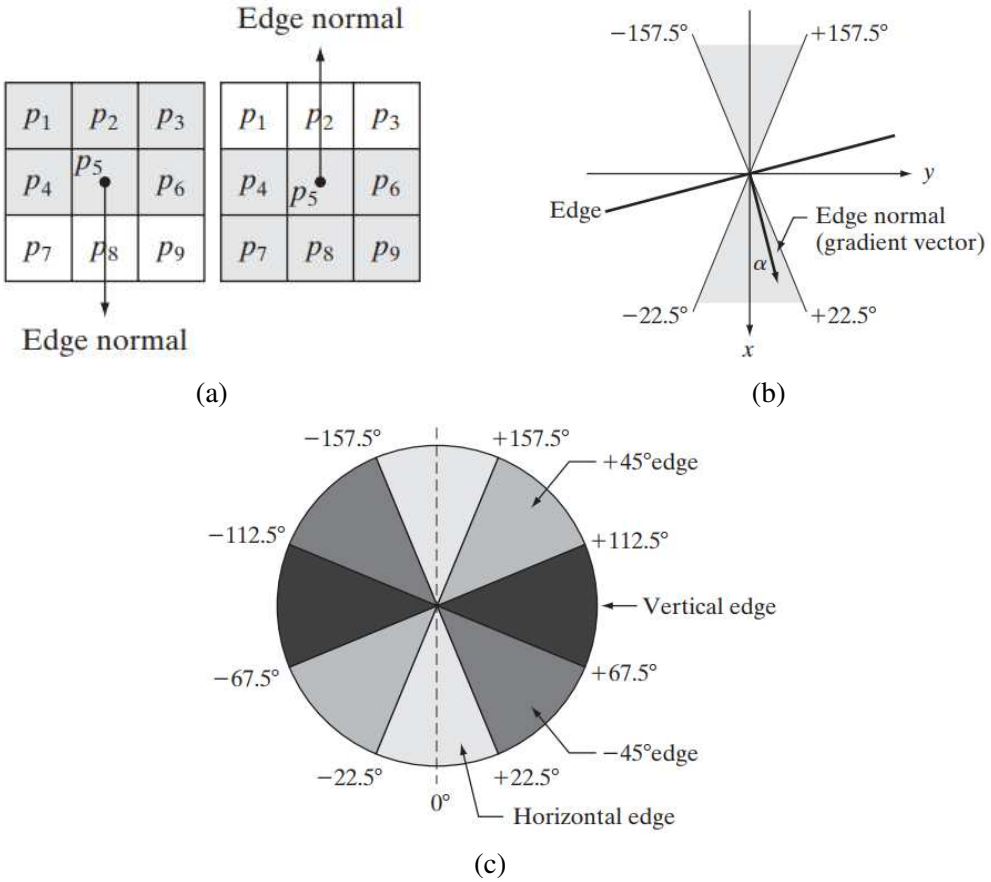
$$f_s(x,y) = G(x,y) \star f(x,y). \tag{5.8}$$

Konvoluciju implementiramo pomoću Gaussove maske veličine $n \times n$, gdje veličinu n biramo kao i u Marr-Hildrethinom algoritmu, a masku konstruiramo uzorkovanjem funkcije 5.7. Nakon izglađivanja računamo magnitudu $M(x,y)$ i smjer, tj. $\alpha(x,y)$ gradijenta, zadane s 3.19 i 5.2.1. g_x i g_y, koji su nam potrebni u računu, možemo izračunati pomoću bilo koje maske sa slike 3.14. Podsjećamo da su $M(x,y)$ i $\alpha(x,y)$ iste veličine kao slika $f(x,y)$.

Kako je $M(x,y)$ generirana pomoću gradijenta, ona obično sadrži široke grebene oko lokalnih maksimuma, pa ih je primjerice za potrebe povezivanja rubnih segmenata u rub poželjno stanjiti. Jedan od mogućih pristupa je **potiskivanje nemaksimuma** (*"nonmaxima/nonmaximal suppression"*) - ideja je pratiti vrh grebena, tražiti piksele koji se ne nalaze na samom vrhu i eliminirati ih.

Za početak ćemo specificirati nekoliko diskretnih smjerova normale ruba, odnosno gradijentnog vektora i svrstati vektore u kategorije prema smjeru. Primjerice, u radu s 3×3 okolinom možemo uvesti 4 orijentacije ruba koji prolazi kroz središte: vertikalna, horizontalna, $+45°$ i $-45°$. Kako bismo sve moguće orijentacije vektora klasificirali u neku od navedene četiri kategorije, trebamo definirati raspon smjerova za koji rub smatramo npr. horizontalnim ili vertikalnim. Smjer ruba izračunamo pomoću normale ruba. Raspon za horizontalni rub je od $-22.5°$ do $22.5°$ te od $-157.5°$ do $157.5°$, a za vertikalni rub od $67.5°$ do $112.5°$ te od $-67.5°$ do $-112.5°$ Ostali rasponi navedeni su na slici 5.15.

Označimo s d_1, d_2, d_3 i d_4 4 osnovna smjera ruba za 3×3 okolinu: redom horizontalni, $-45°$, vertikalni i $45°$. Neka je $g_N(x,y)$ slika potisnutih nemaksimuma. Potiskivanje mak-



Slika 5.15: (a) Dvije različite orijentacije rubova istog smjera koje moramo imati na umu pri definiranju raspona. (b) Primjer ruba koji ćemo klasificirati kao horizontalni prema rasponu kuta njegove normale. (c) Podjela kuteva na ranije spomenute 4 kategorije ruba. Rasponi koji pripadaju istoj kategoriji su obojeni istom nijansom sive.Slike su preuzete iz [35].

smimuma za 3×3 okolinu centriranu u svakom pikselu (x, y) u $\alpha(x, y)$ funkcionira na sljedeći način:

1. Odredimo smjer d_k kojem je $\alpha(x, y)$ najbliži.

2. Ako je vrijednost magnitude $M(x, y)$ manja od barem jednog susjeda duž d_k, postavljamo $g_N(x, y) = 0$ (potiskivanje); inače, neka je $g_N(x, y) = M(x, y)$.

3. Odsjecanje s ciljem redukcije lažnih piksela ruba na slici $g_N(x, y)$.

Cannyjev algoritam koristi **histerezno odsjecanje**[9], metodu koja kombinira niski prag, koji će biti slabo selektivan i visoki prag, koji će biti jako selektivan. Potrebno je biti oprezan pri odabiru pragova: ako je niži prag prenizak, na slici se jave lažni rubovi, a ako je viši prag previsok, neki pravi rubovi nestanu. Cannyjev omjer visokog naspram niskog praga je oko $2:1$ ili $3:1$. Opisanu metodu odsjecanja možemo zamisliti kao da stvaramo dvije nove slike:

$$g_{NH}(x,y) = g_N(x,y) \geq T_H$$
$$g_{NL}(x,y) = g_N(x,y) \geq T_L.$$

Na početku su njihove vrijednosti postavljene na nulu, a nakon primjene praga će $g_{NH}(x,y)$ imati manje nenul-piksela nego $g_{NL}(x,y)$. Očito su svi nenul-pikseli iz $g_{NH}(x,y)$ sadržani u $g_{NL}(x,y)$. Na kraju oduzmimo iz $g_{NL}(x,y)$ sve nenul-piksele iz $g_{NH}(x,y)$:

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$

Nenul-piksele u $g_{NH}(x,y)$ i $g_{NL}(x,y)$ nazivamo redom "jakim" i "slabim" pikselima.

Sve jake piksele u $g_{NH}(x,y)$ smatramo valjanim rubnim pikselima pa ćemo ih tako i označiti. Ovisno o vrijednosti T_H, rubovi u $g_{NH}(x,y)$ obično imaju rupe, ali dulje je rubove moguće dobiti sljedećim postupkom:

1. Pronaći sljedeći neposjećeni piksel p u $g_{NH}(x,y)$
2. Označiti sve slabe piksele ruba iz $g_{NL}(x,y)$ koji su $8-$povezani s p kao valjane piksele ruba.
3. Ako su svi nenul-pikseli u $g_{NH}(x,y)$ posjećeni, nastaviti s idućim korakom, inače se vratiti u prvi.
4. Postaviti sve neoznačene piksele u $g_{NL}(x,y)$ na nulu.

Na kraju rezultat formiramo dodavanjem svih nenul-piksela iz slike $g_{NL}(x,y)$ slici $g_{NH}(x,y)$.

Napomena 5.4.1. *Iako je okvirnu ideju histereze lakše ilustrirati pomoću dvije slike, u praksi se može implementirati direktno tijekom potiskivanja nemaksimuma, a prag se može primijeniti direktno na $g_N(x,y)$ stvaranjem liste jakih piksela i slabih piksela s kojima su povezani.*

Ukratko, Cannyjev algoritam za detekciju ruba glasi:

1. Izgladimo ulaznu sliku Gaussovim filterom.

[9] Od grč. hysteréō - zakašnjavam. Metoda se naziva i dvostrukim odsjecanjem.

2. Izračunamo gradijentne slike magnitude i kuta.
3. Potisnemo/izbrišemo nemaksimume s gradijentne slike magnitude.
4. Pomoću histereznog odsjecanja i analize povezanosti detektiramo i povežemo rubove.

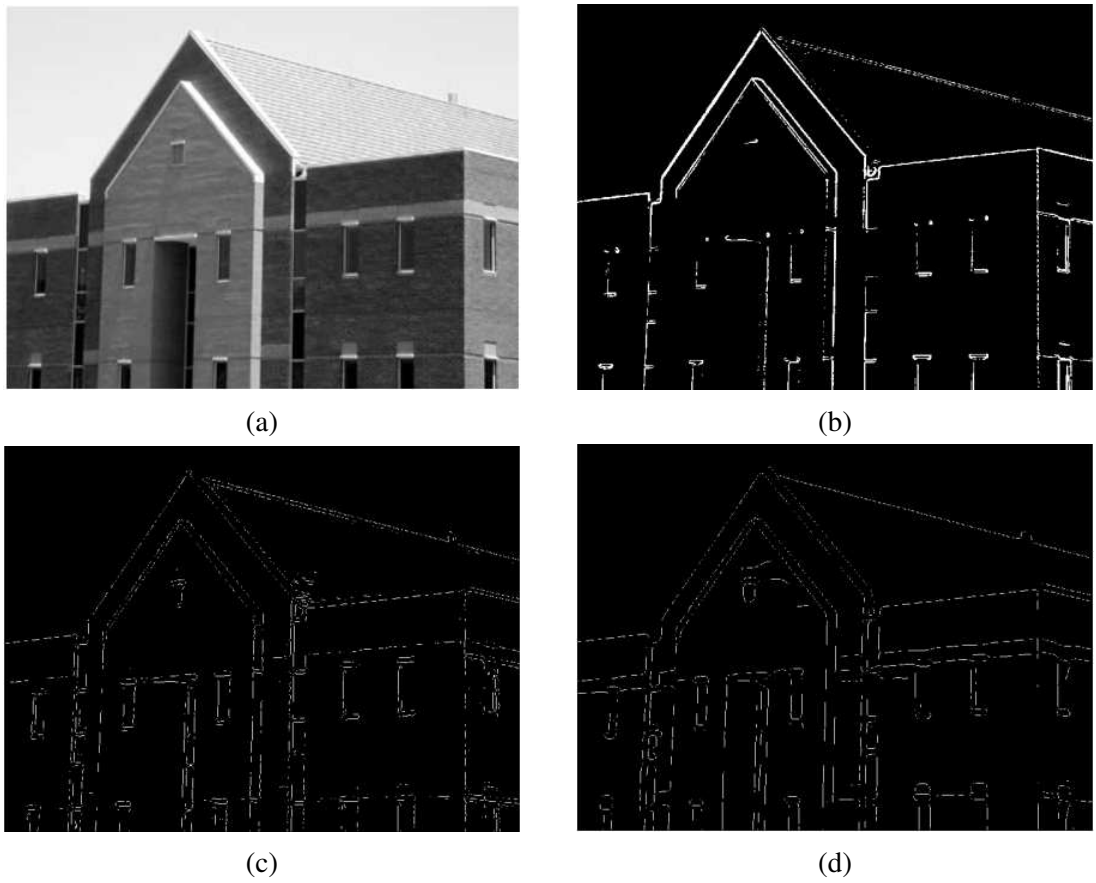
Premda su rubovi nakon potiskivanja maksimuma tanji od netaknutih rubova dobivenih gradijentom, svejedno nam mogu ostati rubovi deblji od 1 piksela. Kako bi se to izbjeglo, obično se nakon četvrtog koraka jednom primijeni algoritam stanjivanja ruba[10].

Glede implementacije, podsjetimo se da se 2D Gaussova funkcija može rastaviti u produkt dviju 1D Gaussovih funkcija, pa se prvi korak Cannyjevog algoritma može formulirati kao 1D konvolucije koje djeluju na redove (stupce) slike jedan po jedan, zatim djeluju na stupce (redove) rezultata. Nadalje, koristeći aproksimacije u 3.8 i 3.9, možemo implementirati izračun gradijenta u drugom koraku pomoću 1D konvolucija.

Primjer 5.4.2. ***(Cannyjeva detekcija ruba):*** *Na slici 5.16 možemo usporediti rezultate prijašnjih metoda s Cannyjevim detektorom. Na 5.16b vidimo rezultat gradijenta s odsjecanjem, a na 5.16c rezultat Marr-Hildrethinog detektora, s tim da su parametri birani tako da se se umanji doprinos manje važnih značajki poput rubova i krova. Slika zgrade je skalirana tako da je njen interval intenziteta* $[0, 1]$*, pa su za generiranje 5.16d korišteni niži prag* $T_L = 0.04$ *i viši prag* $T_H = 0.1$ *(omjer je oko* $2.5 : 1$*),* $\sigma = 4$ *i* 25×25 *maska. Cannyjeva detekcija ruba je bolje očuvala glavne rubove (npr. betonska traka na fasadi je očuvana s obje strane), a bolje potisnula manje važne značajke na slici (npr. na krovu se ne vidi nijedan crijep). Sami rubovi su ravniji i tanji.*

Primjer 5.4.3. *Usporedimo algoritme na još jednom primjeru. Na slici 5.17a nalazi se CT snimka glave, a cilj je segmentirati rub vanjske granice mozga (prikazan sivo na snimci), granicu leđne moždine i mozga (dio ispod nosa) te vanjsku granicu same glave. Slika 5.17b je dobivena izglađivanjem originalne slike* 5×5 *filterom pa odsjecanjem gradijentne slike uz prag od* 15% *maksimalnog intenziteta. Rezultat Marr-Hildrethinog algoritma 5.17c je dobiven uz parametre* $T = 0.002$*,* $\sigma = 3$ *i* 19×19 *masku. Slika 5.17d je rezultat Cannyjevog algoritma uz pragove* $T_L = 0.05$ *i* $T_H = 0.15$ *(omjer* $3 : 1$*),* $\sigma = 2$ *i* 13×13 *masku. Uspoređujući rezultate, primjećujemo da vrijedi sve što smo prepoznali u primjeru 5.4.2 - pravi rubovi su dobro očuvani i većinski neprekinuti, a rubovi koji su nastali zbog teksture sive tvari mozga su uspješno eliminirani.*

[10] Takvi se algoritmi najčešće implementiraju pomoću morfoloških transformacija, pristupom koji se fokusira na geometrijske strukture i značajke poput ruba, kostura i konveksne ljuske elemenata slike. Stanjivanje ruba se primjerice može postići erozijom ili oduzimanjem "hit-and-miss" pretvorbe slike od slike rubova. Više detalja o spomenutim morfološkim transformacijama nalazi se u 9. poglavlju [35]

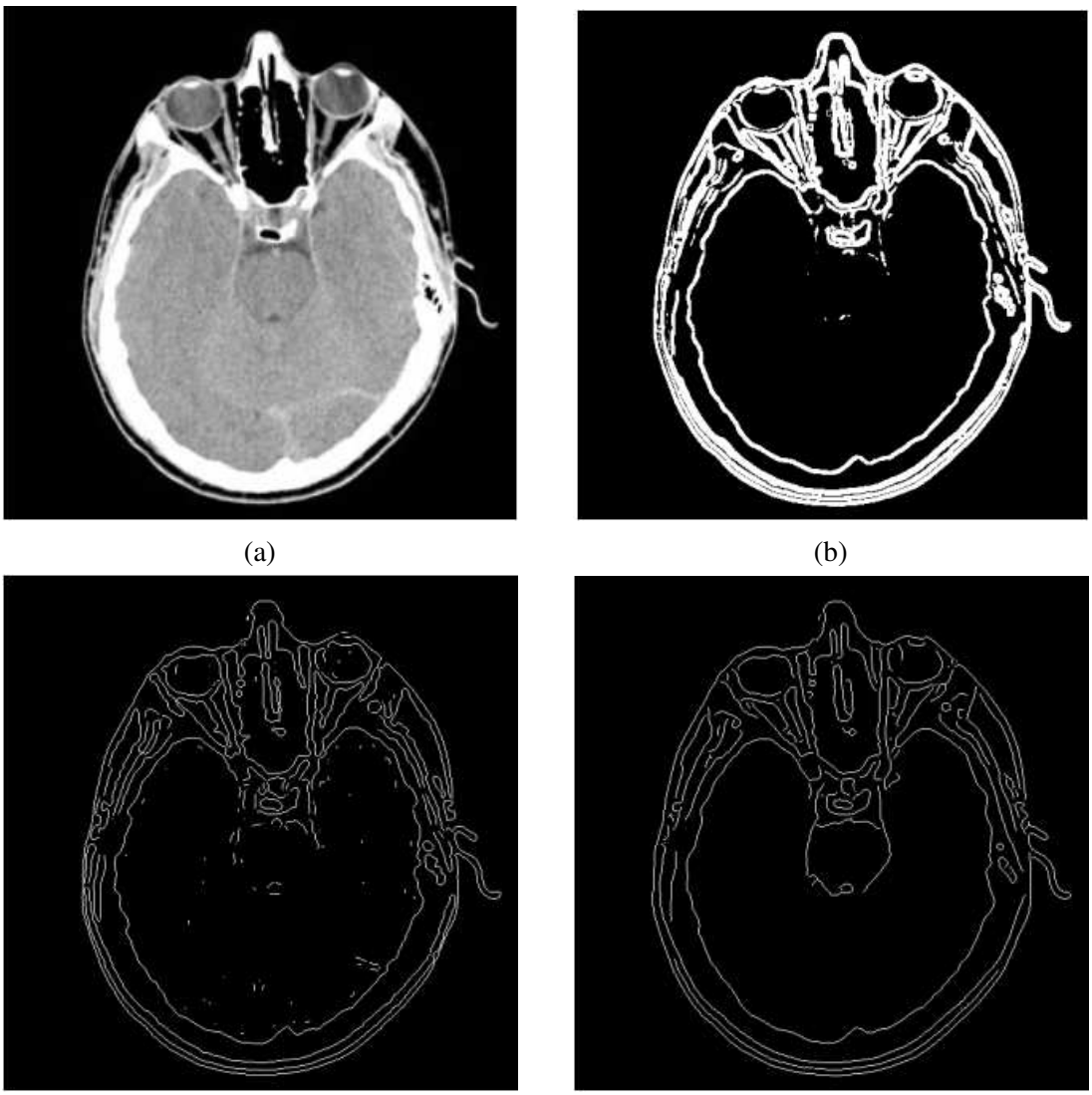


(a) (b)

(c) (d)

Slika 5.16: (a) Slika zgrade, skalirana na $[0, 1]$. (b) Odsječeni gradijent izglađene slike. (c) Rezultat Marr-Hildrethinog algoritma. (d) Rezultat Cannyjevog algoritma. Slike su preuzete iz [35].

Cannyjev algoritam daje bolje rezultate od dosad predstavljenih algoritama, ali je zato teži za implementirati i s duljim vremenom izvršavanja. Iz potonjeg razloga se izbjegava korištenje Cannyja kad je uz precizost važna i brzina (npr. u industriji), no ako je prioritet kvaliteta rubova, on je najčešći izbor detektora.



(a) (b)

(c) (d)

Slika 5.17: (a) CT snimka ljudske glave. (b) Odsječeni gradijent izglađene slike. (c) Rezultat Marr-Hildrethinog algoritma. (d) Rezultat Cannyjevog algoritma. Slike su preuzete iz [35].

Poglavlje 6

Povezivanje rubova i detekcija granice

Algoritmi detekcije ruba u idealnom slučaju vraćaju isključivo piksele koji se nalaze na rubovima, no u praksi je rezultat često poremećen pojavama koje uzrokuju nagle skokove u intenzitetu poput šuma i neuniformnog osvjetljenja, stoga su obično upareni s algoritmima koji povezuju piksele ruba u smislene rubove i granice.

Postoje tri osnovna pristupa povezivanju rubova: lokano, regionalno i globalno procesiranje. Prvi pristup se temelji na uspoređivanju karakteristika piksela iz malih okolina (unaprijed pronađenih) piksela ruba, drugi pristup zahtijeva prijašnje znanje o regijama slike i pikselima granice, a treći pristup koristi čitavu sliku ruba odjednom i oslanja se na njena globalna svojstva.

6.1 Lokalno procesiranje

Ideja lokalnog procesiranja je odabrati kriterij sličnosti piksela, zatim svakom pikselu ruba (prepoznatom kao takvim nekom od tehnika iz prethodnog poglavlja) analizirati karakteristike piksela neke njegove male okoline i konačno sve slične piksele povezati u rub. Dva najčešće korištena kriterija sličnosti piksela ruba su magnituda i smjer gradijentnog vektora u pikselu. Koristit ćemo definiciju magnitude (3.19) umjesto aproksimacije.

Definicija 6.1.1. *Neka je S_{xy} skup koordinata okoline sa središtem u pikselu (x,y) na slici f. Kažemo da je piksel ruba s koordinatama (s,t) sličan pikselu (x,y) po magnitudi ako za odabrani prag $E > 0$ vrijedi*

$$|M(s,t) - M(x,y)| \leq E. \tag{6.1}$$

Definicija 6.1.2. *Neka je S_{xy} skup koordinata okoline sa središtem u pikselu (x,y) na slici f. Kažemo da je piksel ruba s koordinatama (s,t) sličan pikselu (x,y) po kutu ako za*

odabrani prag $A > 0$ *vrijedi*

$$|\alpha(s,t) - \alpha(x,y)| \leq A. \tag{6.2}$$

Prisjetimo se da je smjer ruba u (x,y) okomit na smjer gradijentnog vektora u tom pikselu.

Smatramo da je piksel $(s,t) \in S_{xy}$ sličan pikselu (x,y) ako su zadovoljena oba kriterija, odnosno ako su slični i po magnitudi i po kutu. Potrebno je ponoviti provjeru u svakom pikselu na slici i pritom čitavo vrijeme pratiti koji su pikseli dosad povezani, što lako ostvarimo dodjelom različitih intenziteta različitim grupama piksela. Ovaj postupak je računski skup, ali moguće ga je pojednostavniti na sljedeći način:

1. Izračunamo magnitudu $M(x,y)$ i gradijent $\alpha(x,y)$ za ulaznu sliku $f(x,y)$.
2. Kreiramo binarnu sliku g čija je vrijednost u bilo kojem pikselu dana s

$$g(x,y) = \begin{cases} 1 & \text{ako } \{M(x,y) > T_M\} \wedge \{\alpha(x,y) = A \pm T_A\} \\ 0 & \text{inače} \end{cases}$$

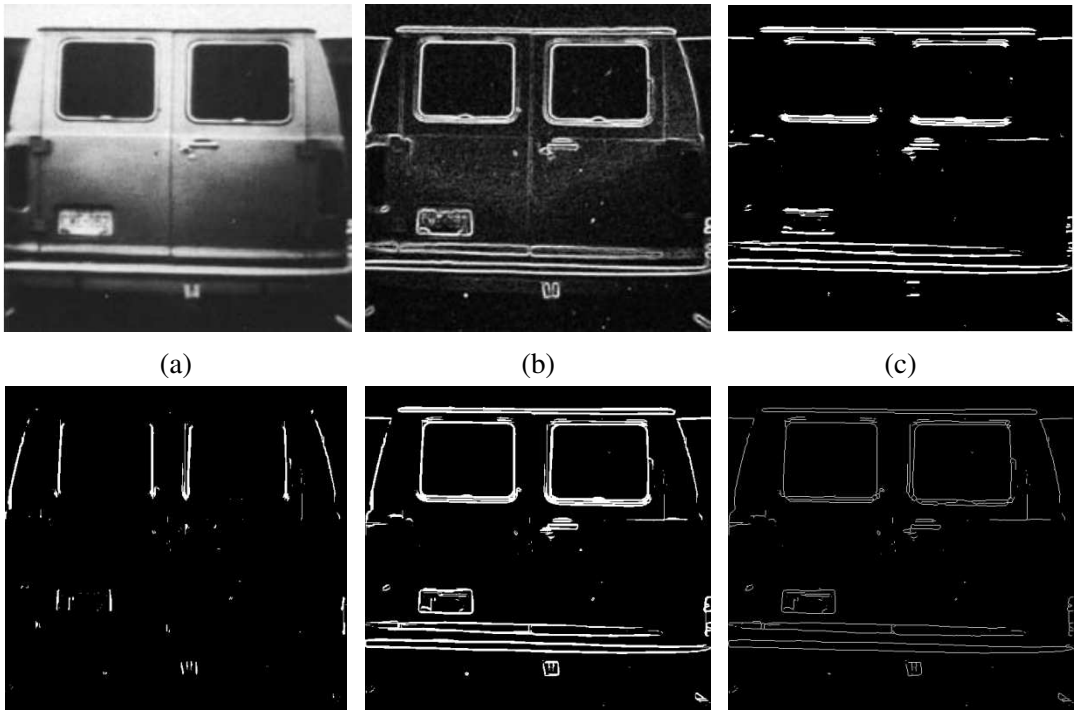
   gdje je T_M prag, A odabrani kut i $[A - T_A, A + T_A]$ raspon prihvatljivih smjerova oko A.
3. Pregledamo redove slike g i popunimo (naštimamo na 1) sve procjepe (niz nula koji je sa svake strane omeđen jedinicama) koje ne prelaze odabranu debljinu K.
4. Za detekciju procjepa u drugim smjerovima (npr. θ), rotiramo zadnje popunjeni g za kut θ, primijenimo proceduru iz trećeg koraka, pa rezultat rotiramo za $-\theta$.

Najčešće se izvršava samo horizontalno i vertikalno povezivanje rubova jer daje povoljne rezultate, a rotacija slike je skupa operacija. Kad je potrebno povezivanje u više smjerova, obično se treći i četvrti korak zamijene radijalnim pregledom i ispravkom ([35]).

Primjer 6.1.3. ***(Lokalno procesiranje):*** *Na slici 6.1 vidimo stražnji dio vozila, a cilj nam je pronaći pravokutnike koji su dobri kandidati za registracijsku tablicu vozila. Takve pravokutnike možemo segmentirati tako da detektiramo snažne horizontalne i vertikalne rubove. Slika 6.1b je gradijentna slika* $M(x,y)$*, a slike 6.1c i 6.1d su rezultat trećeg i četvrtog koraka algoritma uz sljedeće parametre:* T_M *je* 30% *maksimalne vrijednosti gradijenta,* $A = 90°$*,* $T_A = 45°$ *i* $\theta = -90°$*, a rupe s* 25 *ili manje piksela, dakle oko* 5% *širine slike, popunjavamo.*

Kako su kutevi tablica (i stražnjih prozora) na ovom automobilu zaobljeni, bilo je potrebno koristiti širok raspon dozvoljenih kuteva. Na slici 6.1e vidimo rezultat operacije logičkog ILI[1]*, a na slici 6.1f vidimo rezultat morfološkog stanjivanja prethodne slike.*

[1]Često korištena operacija za stapanje dvije slike iste veličine u jednu. U slučaju dvije binarne slike, putujući po prostornim koordinatama provjeravamo je li barem jedan piksel s tim koordinatama na bilo kojoj od dvije slike u prednjem planu: ako jest, stavljamo ga u prednji plan, inače ga ostavljamo u pozadini.



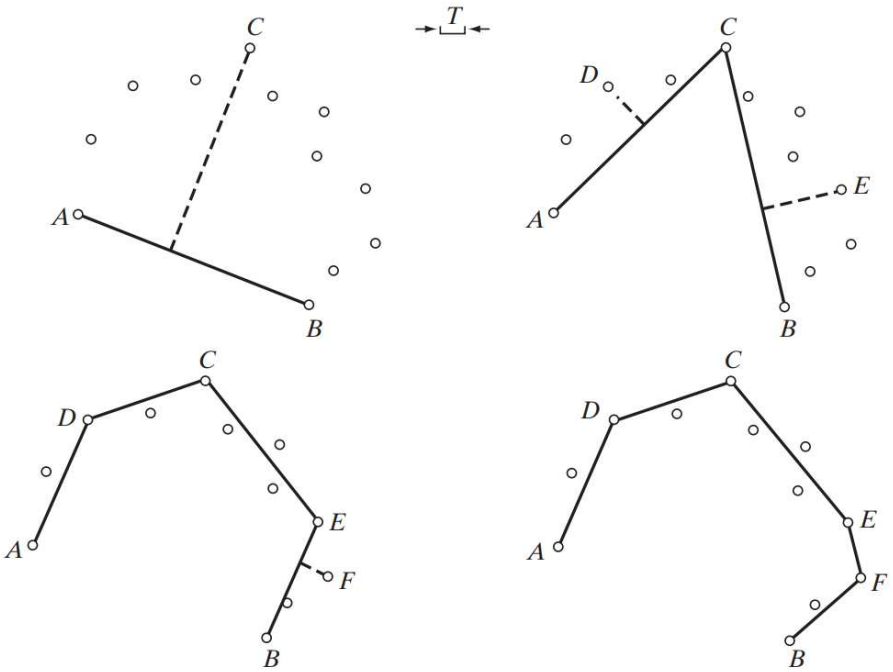
Slika 6.1: (a) Stražnji dio vozila. Slika je dimenzija 534×566. (b) Gradijentna slika. (c) Horizontalno povezani pikseli ruba. (d) Vertikalno povezani pikseli ruba. (e) Slika nastala stapanjem slika (c) i (d). (f) Rezultat stanjivanja slike (e). Slike su preuzete iz [35].

6.2 Regionalno procesiranje

Ponekad je pri obradi slika jednostavno odrediti regije na slici pa pri povezivanju rubova možemo pretpostaviti da znamo koji piksel pripada kojoj regiji (pa i njenom rubu) te se fokusirati na aproksimaciju granica regija. Jedan od mogućih pristupa je aproksimacija granice 2D krivuljom koju konstruiramo pomoću poznatih piksela. Cilj je razviti metodu koja se brzo izvršava, a koja ipak dobro aproksimira važna svojstva granice poput izbočenja i udubljenja. Poligonalne aproksimacije pružaju jednostavnu reprezentaciju granice i daju dovoljno informacija o zanimljivim svojstvima i obliku, pa će se algoritam temeljiti na pronalasku takve reprezentacije.

Primjer 6.2.1. *Na slici 6.2 imamo primjer nekoliko točaka koje predstavljaju otvorenu*[2] *krivulju čije su početna i krajnja točka redom označene s A i B. Ove dvije točke ćemo*

[2]Krivulja koja nema istu početnu i završnu točku. Zatvorenoj krivulji se početna i krajnja točka preklapaju.



Slika 6.2: Nekoliko iteracija predstavljenog algoritma. Pri vrhu ilustracije se nalazi veličina korištenog praga T. Ilustracija je preuzeta iz [35].

smatrati vrhovima mnogokuta. Računamo parametre segmenta koji prolazi kroz A i B, pa među preostalim točakama tražimo točku koja je najudaljenija[3] *od dobivenog segmenta. Prijeđe li ta udaljenost unaprijed definirani prag T, točku označavamo s C i proglašavamo vrhom. Tada računamo parametre segmenta koji povezuje A i C i segmenta koji povezuje C i B. U oba slučaja ponovno tražimo točku najveće udaljenosti od segmenta: prelazi li udaljenost prag T, točku proglašavamo novim vrhom, inače ne deklariramo novi vrh za taj segment. Iterativno ponavljamo postupak dok god ima točaka čija je udaljenost od pripadnog segmenta veća od T. Na zadnjoj slici vidimo konačnu aproksimaciju krivulje kroz dane točke.*

Kao što vidimo, potrebno je odabrati dva početna piksela i potrebno je uvesti uređaj nad pikselima (npr. obrnuto od smjera kazaljke na satu). Kad proizvoljni skup piksela u 2D prostoru ne formira povezani put, nije uvijek očito pripadaju li graničnom segmentu (otvorena krivulja) ili granici (zatvorena krivulja). Pretpostavimo li da su točke uređene, možemo procijeniti radi li se o otvorenoj ili zatvorenoj krivulji tako da izračunamo udaljenosti između piksela. Velika udaljenost između dva uzastopna piksela u uređaju u odnosu na udaljenosti između ostalih piksela uzimamo kao znak da je krivulja otvorena. U tom

[3] U slučaju da su dvije točke jednako udaljene od segmenta, svejedno je koju biramo.

slučaju koristimo krajnje piksele za započeti proces. Ako su udaljenosti između piksela većinski uniformne, onda je krivulja najvjerojatnije zatvorena. U tom slučaju možemo birati početne piksele na više načina - možemo odabrati najlijeviji i najdesniji piksel, možemo odabrati dva ekstrema[4], itd.

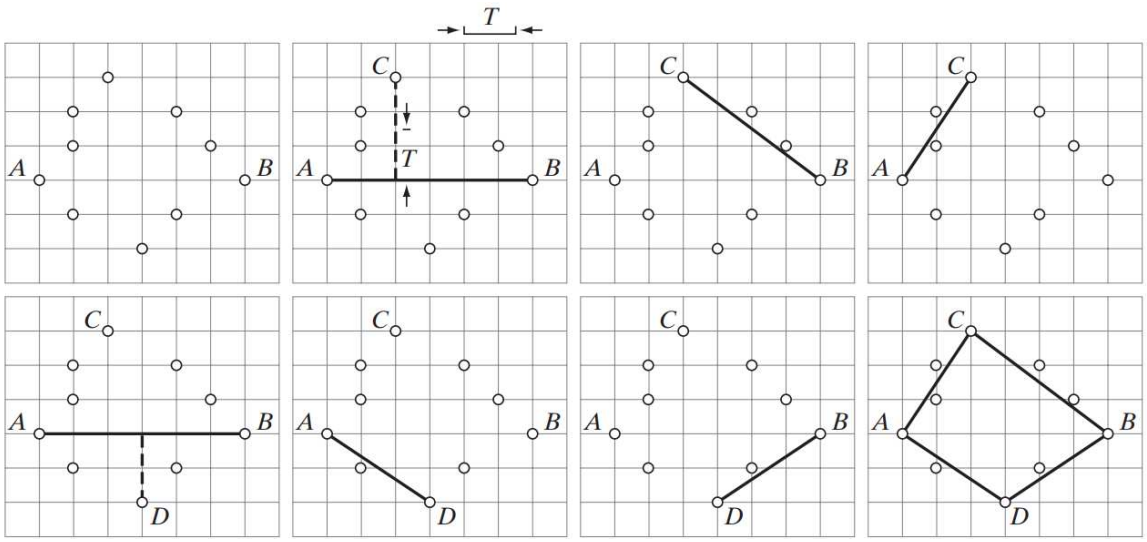
Algoritam za pronalazak poligonalne krivulje za aproksimaciju otvorene ili zatvorene krivulje glasi:

1. Neka je P niz uređenih, međusobno različitih piksela intenziteta 1 na binarnoj slici. Odabiremo dva početna piksela, A i B, kao početne vrhove poligona.
2. Odabiremo prag T i uvodimo 2 prazna stoga, TEMP i FINAL.
3. Ako pikseli u P pripadaju zatvorenoj krivulji, stavljamo A u TEMP, a B u TEMP i FINAL. Ako pikseli pripadaju otvorenoj krivulji, stavljamo A u TEMP i B u FINAL.
4. Računamo parametre segmenta iz posljednjeg vrha u FINAL stogu i posljednjeg vrha u TEMP stogu.
5. Računamo udaljenosti od segmenta iz četvrtog koraka do svih piksela u P koji se po uređaju nalaze između vrhova u četvrtom koraku. Biramo piksel $V_{\max}$ s maksimalnom udaljenosti $D_{\max}$.
6. Ako je $D_{\max} > T$, stavljamo $V_{\max}$ na TEMP stog kao novi vrh. Ponavljamo četvrti korak.
7. Inače mičemo zadnji vrh iz TEMP-a i ubacujemo ga kao zadnji vrh u FINAL.
8. Ako TEMP nije prazan, vraćamo se u četvrti korak.

Vrhove iz FINAL stoga koristimo za formirati krivulju. Ovaj algoritam je jednostavan za implementirati i obično daje prihvatljive rezultate. Demonstrirajmo algoritam na nekoliko primjera.

Primjer 6.2.2. *Na slici 6.3 vidimo skup točaka P. Pretpostavljamo da točke pripadaju zatvorenoj krivulji, da su uređene u smjeru kazaljke na satu i kao početne točke biramo najlijeviju i najdesniju, redom A i B. Prag T je otprilike veličine* 1.5 *jedinice subdivizije (radimo s uniformnom rešetkom). Prva točka u nizu je A. Iduća točka koju označavamo je C jer je ona zadovoljila šesti korak algoritma, pa nju smatramo novim vrhom i dodajemo ju na stog* TEMP. *Jedna od točaka ispod segmenta AB je također zadovoljila uvjet šestog koraka, ali je zbog uređaja prvo uočena točka C, a ta točka će biti detektirana tek u kasnijem koraku (točka D).*

[4]Ekstremima smatramo krajnje točke/piksele dijametra granice, gdje dijametar definiramo kao najveću udaljenost između 2 točke/piksela granice.



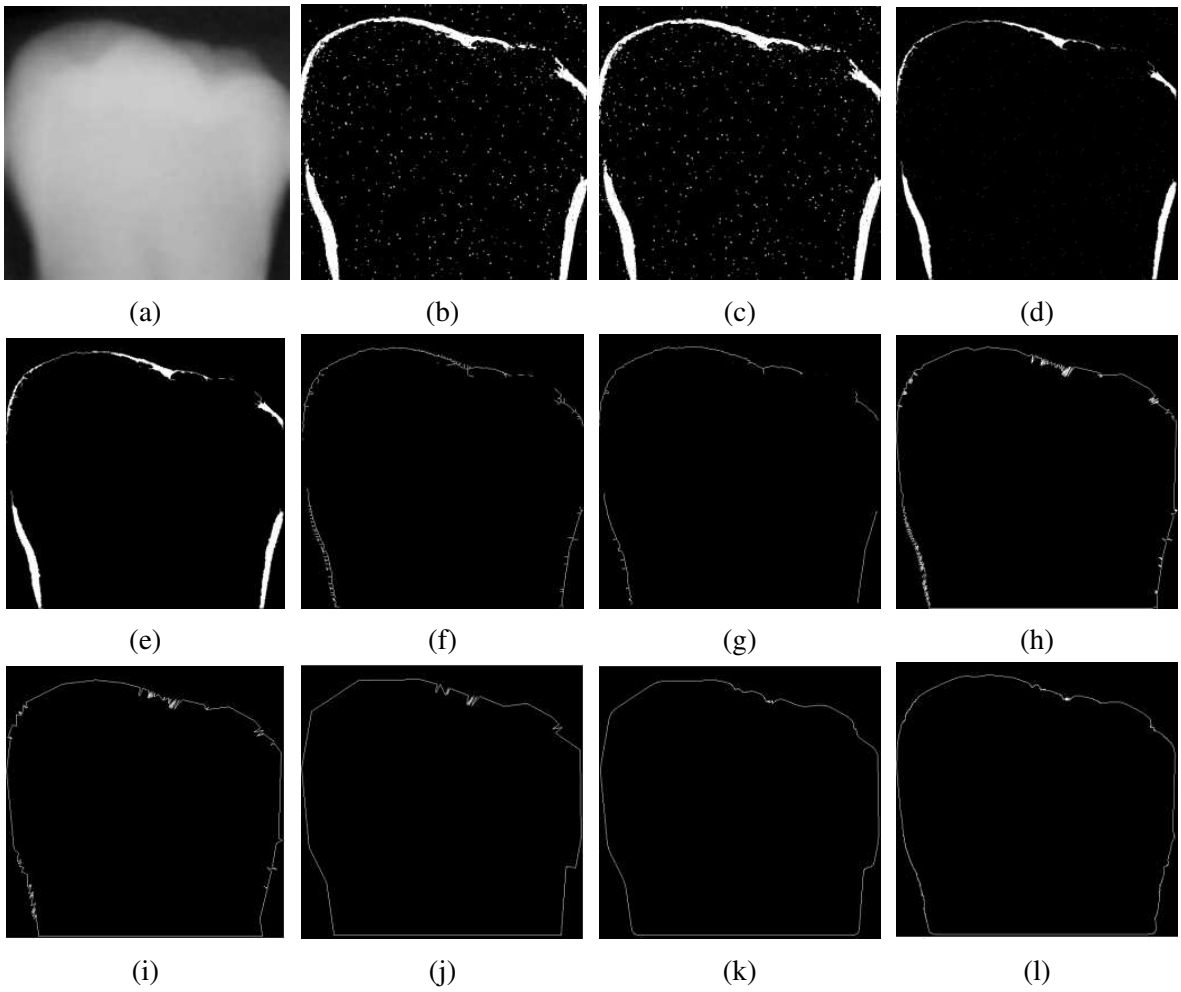
Slika 6.3: Na prvoj slici nalazimo skup točaka P s početnim točkama A i B, koje smo odabrali jer imaju redom najmanju i najveću vrijednost koordinate y. Putujući iz A u smjeru kazaljke na satu, redom prva točka maksimalne udaljenosti od A i B koja prelazi prag je točka C, pa je ona označena kao vrh Iduća točka koja zadovoljava uvjet da postane vrh je označena s D. Na konačnoj slici vidimo vrhove povezane u mnogokut. Ilustracija je preuzeta iz [35].

Na kraju će biti detektirana 4 *vrha, koja su u zadnjoj slici povezana ravnim crtama kako bi formirali mnogokut koji aproksimira dane točke ruba. Primijetimo da su vrhovi označeni u obrnutom smjeru od onog koji je korišten u algoritmu. Da je krivulja bila otvorena, ne bi se dogodila promjena smjera, što je rezultat načina na koji se stogovi inicijaliziraju u slučaju otvorene i zatvorene krivulje.*

Primjer 6.2.3. *Slika 6.4 ilustrira primjenu algoritma na rendgenskoj snimci zuba. Slika je veličine* 550×556 *i intenziteti su skalirani na interval* $[0, 1]$*. Cilj nam je izvući granicu zuba (što nalazi primjenu u forenzici). Gradijentna slika (6.4b) je dobivena pomoću Sobelove maske s pragom* $T = 0.1$*, što je* 10% *najvišeg intenziteta. Ovakve su snimke obično šumovite, pa je za početak potrebno ublažiti šum. Slika je binarna pa je šum moguće ukloniti morfološkim pretvorbama. U ovom slučaju je šum uklonjen koristeći tehniku filtriranjem većinom ("majority filtering") - piksel stavimo u prednji plan ako je barem* 5 *piksela iz njegove* 3×3 *okoline u prednjem planu. Iako je šum smanjen, dio se i dalje zadržao na slici i uklonit ćemo ga morfološkim smanjivanjem*[5]*. Na slici 6.4e vidimo sliku*

[5]Jedna od često korištenih morfoloških metoda za uklanjanje šuma je tzv. "morfološko otvaranje", koje

očišćenu od šuma i preostaje nam samo debela granica, koju možemo stanjiti računanjem morfološkog kostura. Nakon toga se nekom metodom odbacivanja ("pruning") uklanjaju izbočine i time je pripremna obrada gotova.



Slika 6.4: (a) Rendgenska snimka ljudskog zuba veličine 550×566. (b) Gradijentna slika. Slike (c)-(g) predstavljaju proces čišćenja šuma sa slike. Slike (h)-(j) su rezultat korištenja različitih pragova u algoritmu, a slike (k) i (l) su redom izglađene verzije slika (h) i (i). Slike su preuzete iz [35].

Sad ćemo piksele na slici 6.4g aproksimirati mnogokutom. Na idućih nekoliko slika vidimo

uklanja male objekte iz prednjeg plana i prebacuje ih u pozadinu. Više detalja o morfološkim pretvorbama smanjivanja poput otvaranja i erozije može se pronaći u [26].

rezultate primjene algoritma s pragovima $T = 3$ (0.5% duljine), $T = 6$ (1% duljine) i $T = 12$ (2% duljine). Slike 6.4h i 6.4i su solidne aproksimacije, dok je 6.4j nešto lošija. Na rezultatima vidimo potrebu za izglađivanjem ruba, pa ćemo konvoluirati sliku s izglađujućom maskom, čime dobijemo slike 6.4k i 6.4l. Vidimo da je na slici 6.4k desna strana zuba skroz iskrivljena, dok je na slici 6.4l rub znatno izglađen s prilično dobro očuvanim oblikom zuba.

6.3 Globalno procesiranje pomoću Houghove pretvorbe

Prijašnja dva pristupa su primjenjiva kad imamo neke informacije o pripadnosti piksela granicama ili regijama, međutim često trebamo raditi sa slikama ruba bez ikakvog znanja o regijama i položaju objekata od interesa. U tom su slučaju svi pikseli kandidati za povezivanje i trebaju biti prihvaćeni ili izbačeni na temelju unaprijed odabranih globalnih svojstava.

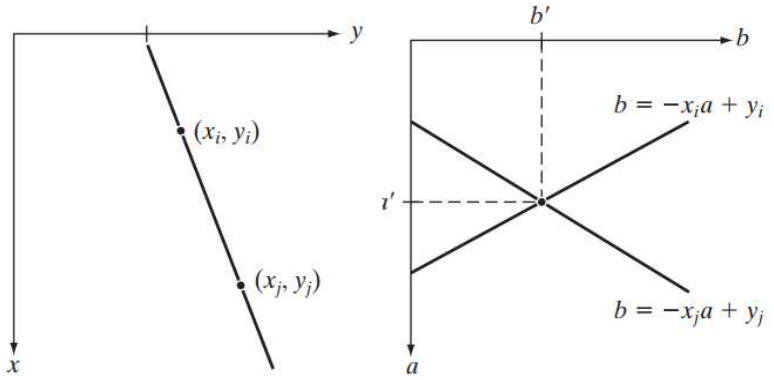
Neka slika ima n piksela i pretpostavimo da želimo pronaći podskupove tih piksela koji leže na ravnim linijama. Jedan jednostavniji pristup bio bi izračunati linije određene svakim parom piksela i tražiti podskupove piksela koji su blizu određenih linija. Ovaj pristup zahtijeva pronalazak $\approx n^2$ linija i obavljanja $\approx n^3$ usporedbi svakog piksela sa svakom linijom, što je dosta računski zahtjevno.

Hough[6] je osmislio pristup koji je po njemu nazvan **Houghova pretvorba**. Ovim pristupom je moguće izolirati značajke specifičnih oblika. Klasična pretvorba se obično koristi za oblike poput crta, krugova, elipsa i sličnih krivulja, dok se generalizirana Houghova pretvorba može primijeniti na oblike koje je teško opisati analitički. Fokusirat ćemo se na klasičnu pretvorbu jer je jednostavnija i ima široki spektar primjena - naime, u industriji i medicini ima dosta objekata čija se granica može parametrizirati regularnom krivuljom. Još jedna prednost Houghove pretvorbe je to što nije toliko osjetljiva na šum i procjepe.

Neka je (x_i, y_i) točka u xy-ravnini - opća jednadžba pravca kroz tu točku glasi $y_i = ax_i + b$. Premda beskonačno mnogo pravaca prolazi kroz (x, y), sve moraju zadovoljiti ovu jednadžbu za neke a i b. Napišemo li tu jednadžbu kao $b = -x_i a + y_i$ i razmotrimo li ab-ravninu (parametarski prostor), dobit ćemo jednadžbu jedinstvenog pravca za odabrani par (x_i, y_i). Nadalje, točka (x_j, y_j) ima pripadni pravac u parametarskom prostoru i, ako nisu paralelni, ovaj pravac siječe pravac koji pripada (x_i, y_i) u nekoj točki (a', b'). Ta točka opisuje pravac koji u xy-ravnini sadrži i (x_i, y_i) i (x_j, y_j), s tim da je a' nagib, a b' odsječak na y-osi pravca.

[6] Paul V.C. Hough, *"Methods and Means for Recognizing Complex Patterns"* ([16])

Sve točke duž tog pravca imaju pripadne pravce u *ab*-ravnini koji se sijeku u (a', b').

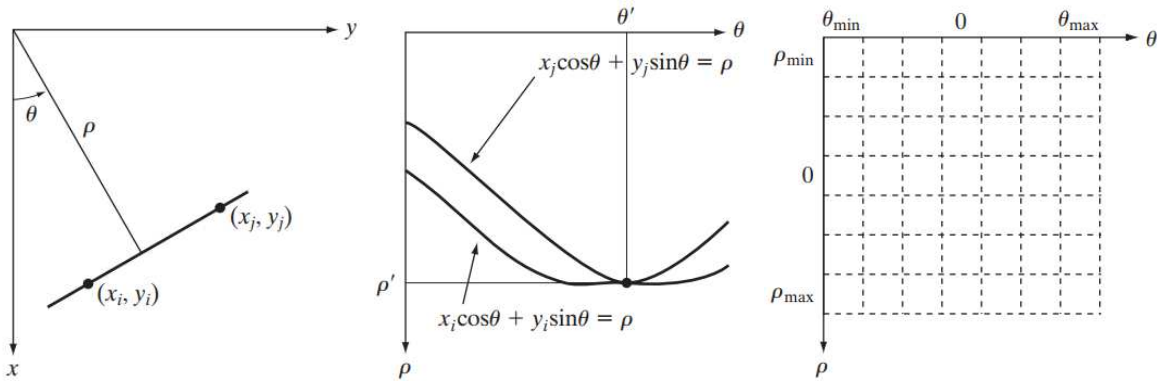


(a) *xy*-ravnina. (b) Parametarski prostor.

Slika 6.5: Slike su preuzete iz [35].

Pravci u parametarskom prostoru koji pripadaju svim točkama (x_k, y_k) u *xy*-ravnini bi se mogli nacrtati, a glavne pravce te ravnine možemo pronaći tako da identificiramo točke presjeka puno pravaca u parametarskom prostoru. Premda je to u teoriji moguće, praktično je to teže izvesti jer a teži prema beskonačnosti što je pravac vertikalniji. Taj problem možemo zaobići korištenjem sljedeće jednadžbe pravca:

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



(a) (b) (c)

Slika 6.6: (a) (ρ, θ) parametrizacija pravca u *xy*-ravnini. (b) Sinusoidalne krivulje u $\rho\theta$-ravnini. Presjek (ρ', θ') pripada pravcu koji prolazi kroz točke (x_i, y_i) i (x_j, y_j) u *xy*-ravnini. (c) Podjela $\rho\theta$-ravnine na akumulacijske ćelije. Slike su preuzete iz [35].

Slika 6.6a ilustrira geometrijsku reprezentaciju parametara ρ i θ. Za horizontalni pravac vrijedi $\theta = 0°$ i ρ je njegov pozitivni odsječak na x osi. S druge strane, vertikalni pravac ima $\theta = 90°$ s ρ kao pozitivnim odsječkom, ili $\theta = -90°$ s ρ kao negativnim odsječkom. Svaka sinusoidalna krivulja na slici 6.6b predstavlja familiju pravaca koji prolaze kroz specifičnu točku (x_k, y_k) u xy-ravnini. Točka presjeka (ρ', θ') na slici 6.6c pripada pravcu koji prolazi kroz (x_i, yi) i (x_j, y_j) na slici 6.6a.

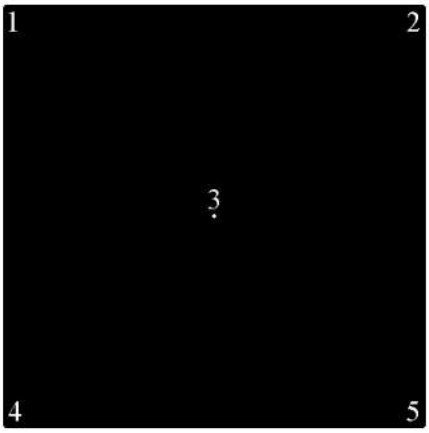
Razlog zašto je Houghova pretvorba računski efikasna je taj što se parametarski prostor $\rho\theta$ može podijeliti na tzv. akumulacijske ćelije (*"accumulator cells"*) kao na 6.6c, gdje su $(\rho_{\min}, \rho_{\max})$ i $(\theta_{\min}, \theta_{\max})$ očekivani rasponi vrijednosti parametara $-90° \leq \theta \leq 90°$ i $-D \leq \rho \leq D$, gdje je s D označena najveća udaljenost između suprotnih kuteva slike. Ćelija na položaju (i, j) s vrijednosti $A(i, j)$ pripada kvadratiću na položaju (ρ_j, θ_j) u parametarskom prostoru. Na početku je vrijednost svake ćelije postavljena na 0. Tada za svaku nepozadinsku točku (x_k, y_k) u xy-ravnini postavimo vrijednost θ na svaku od dozvoljenih vrijednosti subdivizije na θ-osi i izračunamo pripadni ρ pomoću $\rho = x_k \cos\theta + y_k \sin\theta$. Dobivene ρ zaokružimo na najveću dozvoljenu vrijednost ćelije duž ρ-osi. Ako izbor θ_p rezultira rješenjem ρ_p, onda postavimo $A(p, q) = A(p, q) + 1$. Ako na kraju ovog postupka dobijemo $A(i, j) = P$, što znači da se P točaka u xy-ravnini nalazi na pravcu $x\cos\theta_j + y\sin\theta_j = \rho_i$. Gustoća subdivizije u $\rho\theta$-ravnini određuje točnost kolinearnosti točaka. Broj izračuna u opisanoj metodi je linearan u odnosu na broj nepozadinskih piksela u xy-ravnini.

Primjer 6.3.1. ***(Houghova pretvorba):*** *Na slici 6.7a imamo sliku veličine* 101×101 *s* 5 *označenih točaka, a na slici 6.7b vidimo te iste točke preslikane na $\rho\theta$-ravninu sa subdivizijom veličine* 1 *piksel za os ρ i* $1°$ *za os θ. Vrijedi $\theta \in [-90°, 90°]$ i $\rho \in [-\sqrt{2}D, \sqrt{2}D]$, a D je u ovom slučaju udaljenost između suprotnih kuteva kvadratne slike. Kao što vidimo na slici 6.7b, svaka krivulja ima drukčiji sinusoidalni oblik. Točke A i B prikazuju kako je Houghova pretvorba u stanju detektirati kolinearnost. Točka A predstavlja presjek krivulja koje pripadaju točkama* 1,3 *i* 5 *na slici u xy-ravnini. Položaj točke A u ravnini ukazuje na to da te tri točke leže na istom pravcu koji prolazi kroz ishodište ($\rho = 0$) i pod nagibom od* $45°$. *Na sličan način krivulja koja siječe točku B u parametarskom prostoru ukazuje na to da su točke* 2, 3 *i* 4 *na pravcu pod nagibom od* $-45°$ *čija je udaljenost od ishodišta $\rho = 71$ (polovica dijagonalne udaljenosti od ishodišta do suprotnog kuta, zaokružena na najmanji cijeli broj).*

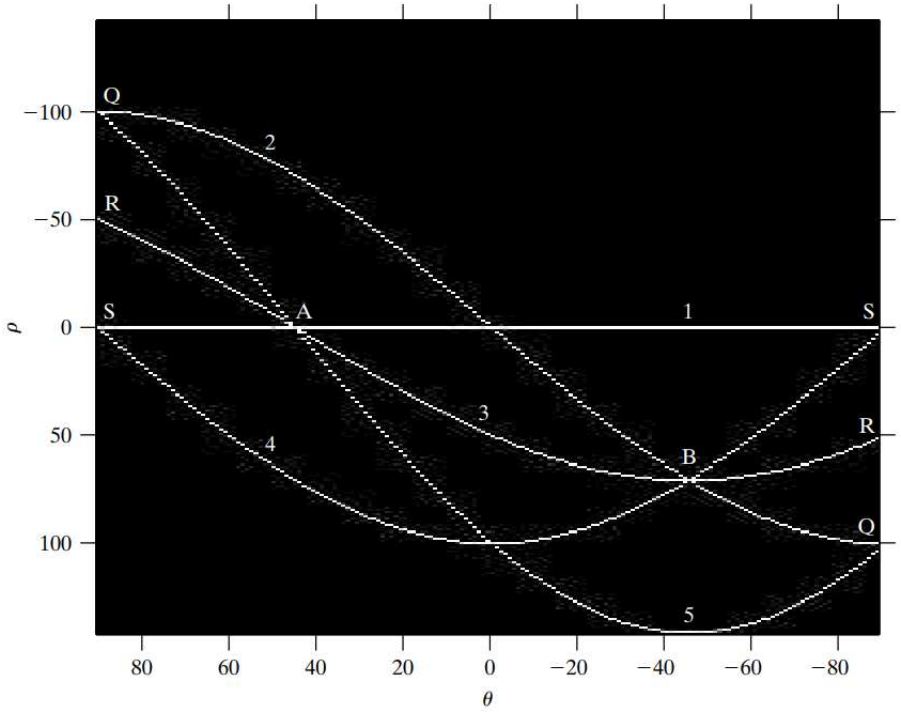
Iako je dosad fokus bio na pravcima, Houghovu pretvorbu možemo primijeniti na bilo koju funkciju oblika $g(\mathbf{v}, \mathbf{c}) = 0$, gdje $\mathbf{v}$ predstavlja vektor koordinata, a $\mathbf{c}$ predstavlja vektor koeficijenata. Ovim postupkom možemo detektirati točke koje se npr. nalaze na krugu

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2.$$

U ovom slučaju imamo 3 parametra, pa dobijemo 3D parametarski prostor s kubičnim ćelijama i akumulacijskim ćelijama oblika $A(i, j, k)$. Ovaj put povećavamo c_1 i c_2, računamo



(a) Binarna slika veličine 101 × 101 s 5 točaka u prednjem planu - po jedna u sva četiri kuta slike i jedna u središtu slike.



(b) Pripadni parametarski prostor. Jedinica subdivizije u θ-smjeru je 1°, a u ρ-smjeru je 1 piksel.

Slika 6.7: Slike su preuzete iz [35].

c_3 koji zadovoljava gornju jednadžbu i ažuriramo akumulacijsku ćeliju vezanu uz trojku (c_1, c_2, c_3). Složenost Houghove pretvorbe ovisi o broju koordinata i koeficijenata u danoj funkcionalnoj reprezentaciji. Daljnja poopćenja Houghove pretvorbe su moguća za detekciju krivulja bez jednostavnih analitičkih reprezentacija, kao i primjena pretvorbe na slike u sivim tonovima.

Opišimo kako funkcionira algoritam za povezivanje ruba na bazi Houghove pretvorbe:

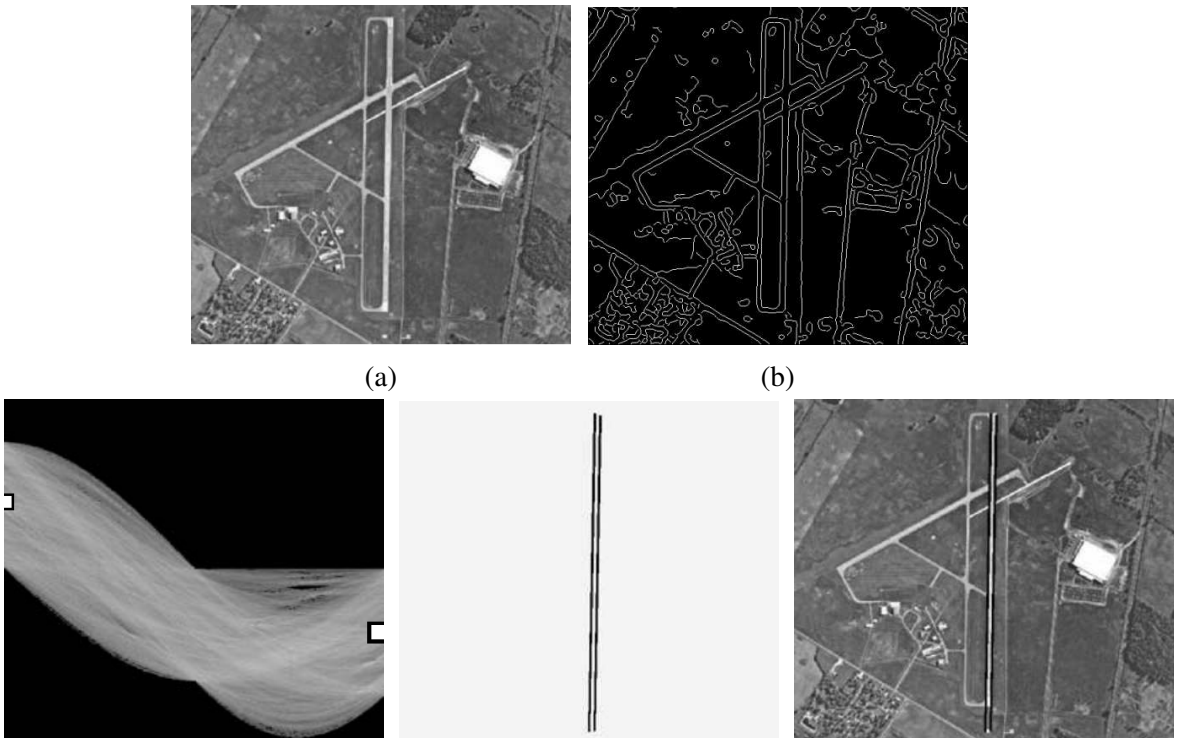
1. Dobiti binarnu sliku rubova bilo kojom od prije diskutiranih metoda.
2. Specificirati subdiviziju u $\rho\theta$-ravnine.
3. Među akumulacijskim ćelijama pronaći one s visokom koncentracijom piksela.
4. Pregledati veze između piksela u danoj ćeliji (provjeriti neprekidnost).

Neprekidnost u ovom slučaju pokažemo računanjem udaljenosti između nepovezanih piksela koji pripadaju danoj akumulacijskoj ćeliji. Procjep u pravcu kojem pripada dana ćelija je "pokrpan" ako je duljina procjepa manja od odabranog praga.

Primjer 6.3.2. *Na slici 6.8a vidimo zračnu snimku aerodroma. Cilj ovog primjera je koristiti Houghovu pretvorbu za izvući* 2 *ruba glavne staze, što bi moglo biti korisno u autonomnoj navigaciji letjelica. Prvi korak je dobiti sliku ruba. Nju ćemo dobiti pomoću Cannyjevog algoritma (slika 6.8b) sa sljedećim parametrima:* $T_L = 0.05$, $T_H = 0.15$, $\sigma = 2$ *i maskom veličine* 13×13. *Slika 6.8c prikazuje Houghov parametarski prostor dobiven koristeći inkremente od* 1° *za* θ *i* 1 *piksela za* ρ.

Željena staza je pod nagibom od 1° *u odnosu na sjever, pa ćemo odabrati ćelije koje pripadaju smjeru* $\pm 90^\circ$ *i sadrže najveći zbroj jer su tražene staze najduže linije u tom smjeru. Te ćelije su istaknute na slici 6.8c malim, bijelim kvadratićima. Slika 6.8d prikazuje linije koje pripadaju spomenutim akumulacijskim ćelijama, a na slici 6.8e vidimo originalnu sliku kojoj su dodane te linije - one očito pripadaju stazi koju smo tražili. Linije su dobivene povezivanjem svih procjepa koji ne prelaze* 20% *širine slike (*≈ 100 *piksela).*

Jedine informacije koje su nam trebale da riješimo ovaj problem je orijentacija staze i položaj promatrača u odnosu na stazu. Na primjer, vozilo koje autonomno vozi bi znalo da će sjeverno-orijentirana cesta po kojoj vozi biti vertikalna ako vozilo putuje prema sjeveru - orijentacije staza i smjer puta bili bi mu dostupni preko GPS sustava. Te informacije možemo koristiti i za izračunati udaljenosti između vozila i staze, te dozvoliti procjenu parametara poput očekivane duljine linija u odnosu na veličinu slike.



(a) (b)

(c) (d) (e)

Slika 6.8: (a) Slika aerodroma veličine 502 × 564. (b) Slika ruba dobivena Cannyjevim algoritmom. (c) Houghov parametarski prostor s 2 istaknuta kvadratića. (d) Linije na slici pripadaju točkama koje pokrivaju kvadratići na prethodnoj slici. (e) Originalna slika s dodanim linijama iz slike (d). Slike su preuzete iz [35].

Poglavlje 7

Implementacija

Predstavljene algoritme za detekciju točaka, linija i rubova smo implementirali u programskom paketu *Octave*. U ovom poglavlju ćemo opisati neke funkcije iz paketa koje su korištene u algoritmima, prezentirati rezultate algoritama na različitim slikama, prokomentirati parametre te međusobno usporediti učinkovitost algoritama na nekim oglednim primjerima. Većina slika iz primjera je preuzeta s web stranice [28], koja nudi bogatu zbirku slika za testiranje i treniranje algoritama za obradu slika. Osim ako je naglašena korištena *Octave* funkcija, svi su algoritmi samostalno implementirani.

7.1 Funkcije za prostorno filtriranje slike

Funkcija fspecial

Neke standardne linearne prostorne filtere možemo implementirati pomoću funkcije fspecial, koja je dio *Octaveovog* Image paketa ([6]). Funkcija može generirati i neke od filtera koje nismo obradili (recimo disk filter, motion filter), ali mi ćemo navesti samo tipove koji su korišteni za generiranje primjera kasnije u poglavlju. Filtere dijelimo na izglađujuće filtere i izoštravajuće/detektorske filtere.

Izglađujući filteri:

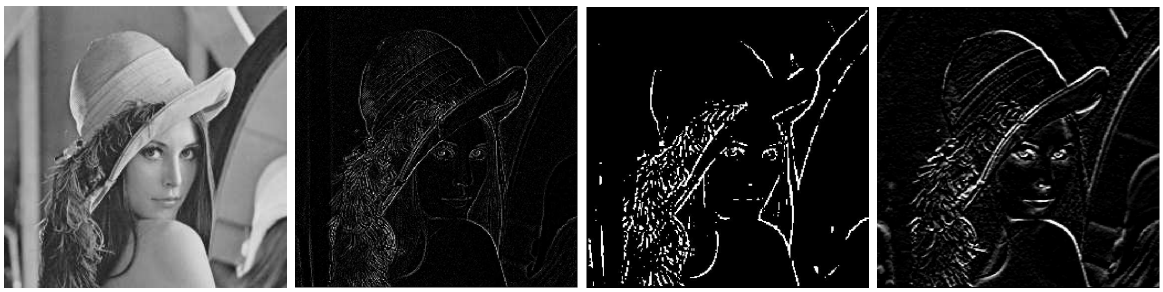
- **average** - unosimo $\mathsf{fspecial}('\mathsf{average}', [\mathsf{r}\ \ \mathsf{c}])$. Funkcija vraća box filter veličine $\mathsf{r} \times \mathsf{c}$.
- **gaussian** - unosimo $\mathsf{fspecial}('\mathsf{gaussian}', [\mathsf{r}\ \ \mathsf{c}], \mathsf{sig})$. Funkcija vraća Gaussov filter veličine $\mathsf{r} \times \mathsf{c}$ sa standardnom devijacijom sig.

Izoštravajući/detektorski filteri:

- **laplacian** - unosimo fspecial('laplacian', alpha). Funkcija vraća 3×3 poopćeni Laplacian filter čiji je oblik određen s alpha. Za alpha = 0.0 ćemo dobiti 3.10a.
- **log** - unosimo fspecial('log', [r c], sig). Funkcija vraća LoG filter veličine r × c sa standardnom devijacijom sig.
- **sobel** - unosimo fspecial('sobel'). Funkcija vraća 3×3 Sobelov filter definiran kao negativna verzija 3.14e.

(a) Slika kamermana. (b) 'average'; r = c = 10 (c) 'gaussian'; r, c = 25; σ = 4

Slika 7.1: Učinci dvaju predstavljenih filtera za izglađivanje slike i parametri koji su korišteni pri filtriranju. Slika kamermana pripada paketu standardnih test slika sa [28].



(a) Slika žene. (b) 'laplacian'; α = 0 (c) 'log'; r, c = 5; σ = 1 (d) 'sobel'

Slika 7.2: Učinci tri predstavljena filtera za izoštravanje slika i detekciju ruba, i parametri koji su korišteni pri filtriranju. Originalna slika pripada paketu standardnih test slika sa [28].

Funkcija imfilter

U trećem poglavlju smo objasnili kako funkcionira filtriranje slika. Predstavili smo dvije metode filtriranja - korelaciju i konvoluciju - i objasnili da je potrebno proširiti sliku kako bi filter uspješno prošao kroz svaki piksel. Funkcija imfilter ([6]) dozvoljava odabir načina filtriranja i nadopune slike, a dodatno možemo odabrati veličinu rezultata - može vratiti nadopunjenu ("padded") sliku ili sliku veličine originala.

U našem slučaju nemamo koristi od nadopunjene slike, pa ćemo uvijek tražiti rezultat veličine originala. Što se tiče izbora metode, koristit ćemo isključivo konvoluciju (koju odaberemo upisom 'conv' u argument) umjesto korelacije (koju odaberemo s 'corr'). Kao što smo ranije komentirali, rezultati dviju metoda se ne razlikuju previše, a izbor metode ovisi o konstrukciji algoritma. Za kraj komentirajmo još metode nadopune (iste metode koristi i *Octave* funkcija padarray ([6]), koju možemo koristiti ako nam treba samo nadopuna).

Metode nadopune:

- P - odabere se neki skalar P (obično 0) i svakom se pikselu dodijeli taj intenzitet
- 'replicate' - ponavljanje intenziteta granice
- 'symmetric' - zrcaljenje intenziteta preko vanjske granice
- 'circular' - slika se tretira kao faza periodične funkcije

(a) P = 0 (b) 'replicate' (c) 'symmetric' (d) 'circular'

Slika 7.3: Efekti različitih vrsta nadopune slike prije filtriranja. Vrsta filtriranja može imati utjecaj na rub gradijentne slike (npr. lažni rubovi).

7.2 Detekcija točaka

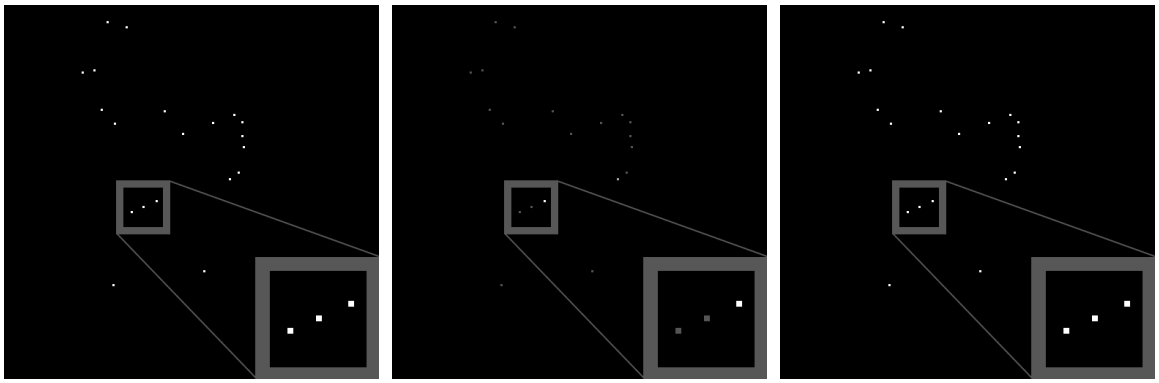
Algoritam za detekciju izoliranih točaka je prilično jednostavan za implementaciju. Potrebno je filtrirati sliku Laplaceovom maskom kako bismo pronašli piksele u kojima se

dogodio nagli skok u intenzitetu, zatim odabrati prag kojim ćemo odsjeći slabe odzive (po apsolutnoj vrijednosti) jer pretpostavljamo da će točka uzrokovati nagli skok/pad u drugoj derivaciji.

Na početku učitamo sliku i konstruiramo masku 3.10b. Učitana se slika filtrira pomoću funkcije imfilter. Prag T možemo slobodno odabrati, ali pazimo da je dovoljno visok - možemo ručno odabrati neki intenzitet I ili ga odabrati pomoću apsolutno najveće vrijednosti na Laplaceovoj slici, npr. postaviti T na $n\%$ te vrijednosti. Kad odaberemo prag, prođemo jednom kroz sve piksele i postavimo na nulu intenzitet svih piksela čija je apsolutna vrijednost odziva na Laplaceovoj slici manja od praga.

Ilustrirat ćemo algoritam na sljedeća dva primjera.

Primjer 7.2.1. *Slika 7.4a je binarna slika s* 20 *točaka veličine jedan piksel. Jako je mali dio slike u prednjem planu, stoga su područja konstantnog intenziteta (koja čine većinu slike) dosta tamna nakon primjene Laplaciana (7.4b). Budući da su u prednjem planu samo točke, potrebno je koristiti niži prag, recimo* 10% *vrijednosti maksimalnog odziva na Laplaceovoj slici. Algoritam je uz taj prag uspio detektirati sve točke, pa dobijemo sliku koja je identična prvoj.*



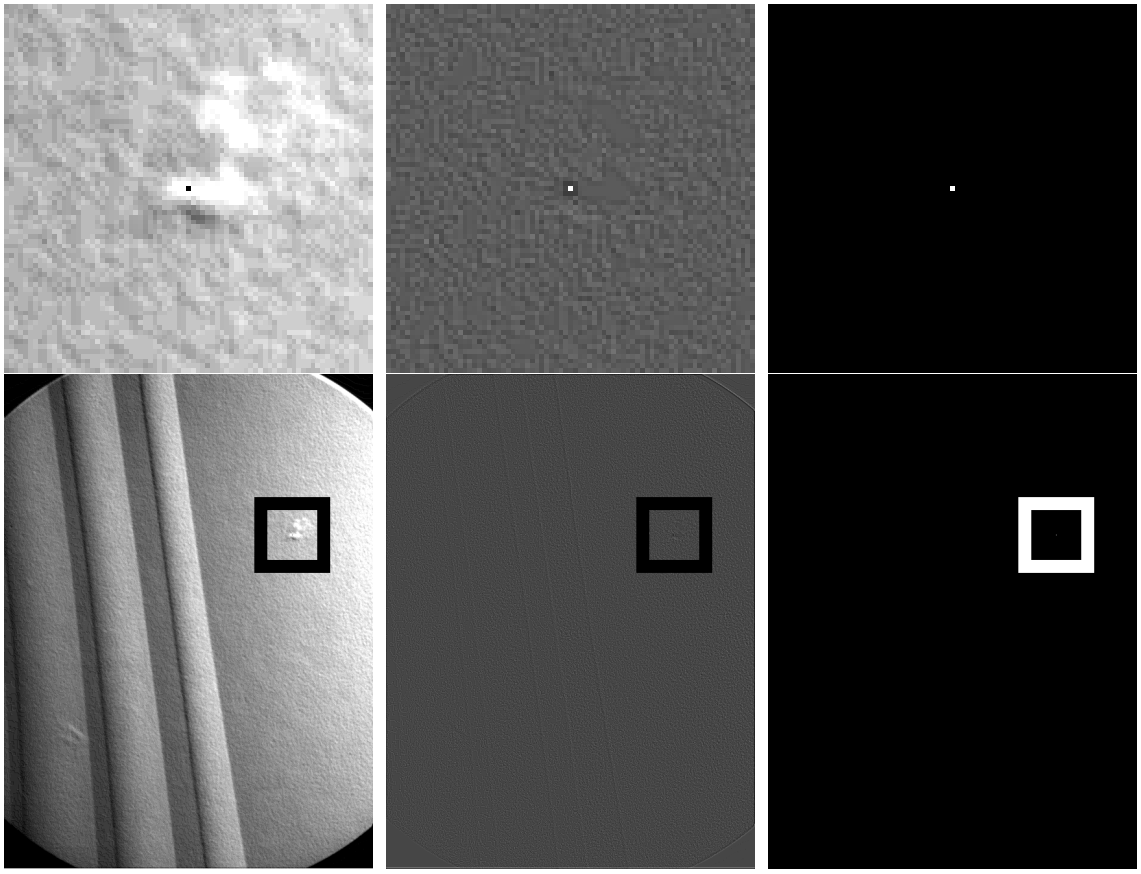
(a) Slika s 20 točkica. (b) Laplaceova slika. (c) Detektirane točke.

Slika 7.4: Detekcija točaka s binarne slike.

Primjer 7.2.2. *Obratimo pažnju na sliku 7.5 - u donjem retku imamo sliku u različitim fazama algoritma na kojoj je kvadratom označeno područje s oštećenjem, a u gornjem retku imamo uvećanu verziju tog područja. U prvom stupcu imamo sliku u sivim tonovima na kojoj je prikazana rendgenska snimka lopatice turbine s malenom poroznosti unutar označenog područja. Ta rupica je veličine jednog piksela i cilj nam je izdvojiti je sa slike.*

U drugom stupcu vidimo rezultat filtriranja slike Laplaceovim filterom. Sama točka je dosta svjetlija u odnosu na svoju širu okolinu, a njeni susjedi su znatno tamniji, što je očekivano kod odziva na temelju ranijih komentara o djelovanju Laplaciana (). Ovaj put je kao prag korišteno 90% *vrijednosti apsolutno najvećeg odziva sa Laplaceove slike. U zadnjem stupcu vidimo rezultat nakon odsjecanja - dobijemo potpuno crnu sliku s jednom bijelom točkicom u prednjem planu.*

Na slici 7.6 možemo vidjeti dva primjera loše segmentacije zbog lošeg izbora praga. Na prvoj slici je prag izračunat na isti način kao i u prethodnom primjeru, a na desnoj slici je prag povećan, ali je i dalje dovoljno nizak da uhvati odzive nastale zbog šuma.

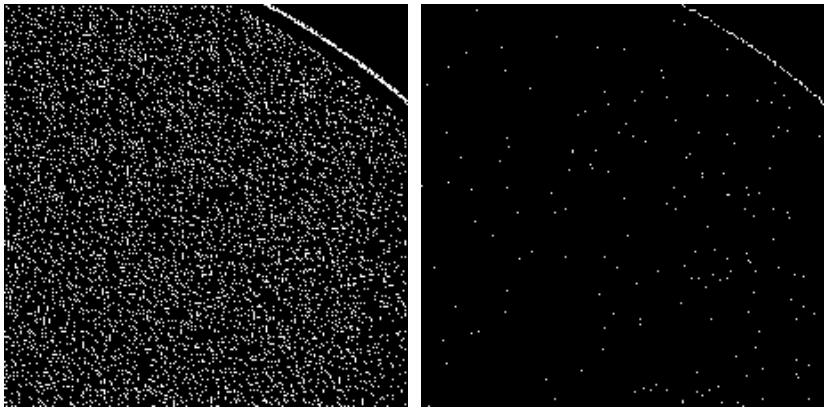


(a) Slika s malom poroznosti u označenom području.

(b) Rezultat Laplaceovog filtriranja slike (a).

(c) Segmentirana slika s 1 pikselom u prednjem planu.

Slika 7.5: Postupak segmentacije malene rupe sa snimke lopatice turbine. Originalna snimka lopatice turbine je preuzeta s [28].



Slika 7.6: Isječci iz primjera loše segmentacije izolirane točke na slici 7.5: na lijevoj slici je korišten prag of 5% apsolutno najveće vrijednosti sa Laplaceove slike, dok je na desnoj slici prag povišen na 10% te vrijednosti.

7.3 Detekcija linija

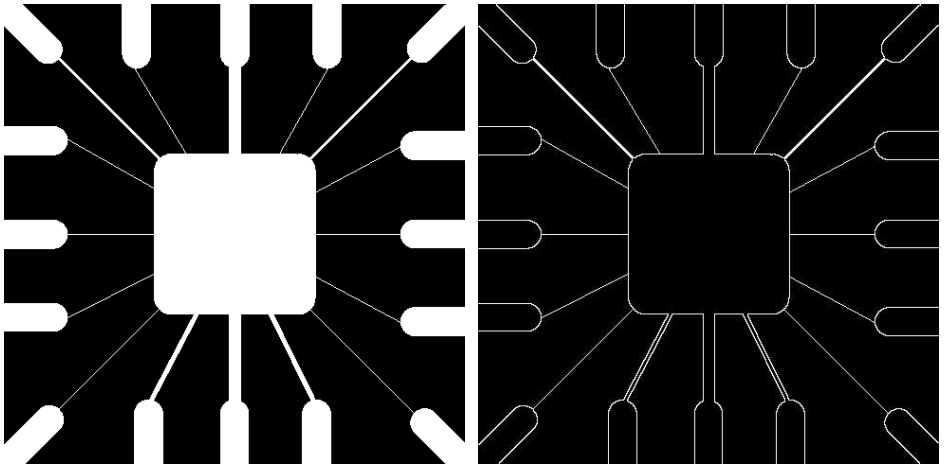
U četvrtom smo poglavlju spomenuli dvije metode detekcije linija na slici: prva metoda se oslanja na korištenje Laplaceovog filtera, a druga metoda pomoću filtera predstavljenih na slici 4.5 pokušava sa slike izdvojiti linije specifičnog smjera. Demonstrirat ćemo obje metode na slici predloška za povezivanje žica.

Slično kao i kod segmentacije izolirane točke, ideja je iskoristiti Laplaceov filter za pronalazak naglih skokova u intenzitetu. Filtriranje obavimo na isti način kao u primjeru s izoliranim točkama. Nakon toga je potrebno tretirati efekt dvostrukog ruba, što ćemo uraditi tako da zadržimo samo pozitivni dio odziva sa slike. Moguće je zadržati apsolutnu vrijednost ukupnog odziva, ali tad dolazi do zadebljanja linija, što je često nepoželjno svojstvo.

Algoritme detekcije linija ćemo demonstrirati na binarnoj slici, ali napominjemo da bismo u slučaju grayscale slike (slike u sivim tonovima) poput one u primjeru 7.2.2 samo pažljivije odabrali prag T. Umjesto da odsječemo sve negativne vrijednosti sa slike, odsjekli bismo sve vrijednosti čiji je odziv manji od neke pozitivne vrijednosti.

Primjer 7.3.1. *Slika 7.7a je binarna slika predloška za povezivanje žica s linijama koje se pružaju u* 8 *različitih smjerova. Slika je filtrirana Laplaceovim filterom, nakon čega su s dobivene slike odsječeni svi negativni odzivi (dakle* $T = 0$*), čime je dobiven rezultat na slici 7.7b. Sve su linije uspješno detektirane, s tim da na mjestima zadebljanih linija možemo primijetiti ranije spomenuti "zero valley" efekt (4.4). Naime, veličina filtera bi trebala biti*

prilagođena debljini linija, a korišteni 3×3 *filter je prilagođen za tanke linije debljine* 1 *piksel, stoga njegov odziv u susretu s linijama debljine* 3 *piksela nije iznenađujuć.*

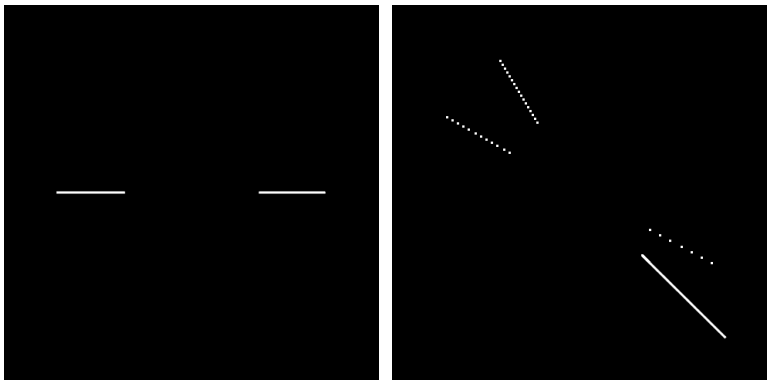


(a) Predložak za povezivanje žica s linijama u raznim smjerovima.

(b) Linije detektirane pomoću Laplaceovog filtera.

Slika 7.7: Postupak segmentacije linija sa binarne slike. Slika predloška za povezivanje žica je preuzeta s [28].

Primjer 7.3.2. *Na istom primjeru ćemo demonstrirati detekciju linija pomoću filtera sa slike 4.5. Tražimo linije s najjačim odzivom nakon filtriranja: filtriramo li maskom za detekciju horizontalnih linija, najjači odziv će imati upravo horizontalne linije. Zbog toga ćemo malo povisiti prag u odnosu na prethodni primjer - primjeri sa slike 7.8 generirani su tako da su nakon filtriranja odsječeni svi odzivi osim maksimalnog.*



Slika 7.8: Sa slike 7.7a su segmentirane redom horizontalne linije i linije pod nagibom od 45°. Odzivi su podebljani radi bolje vidljivosti.

7.4 Detekcija ruba

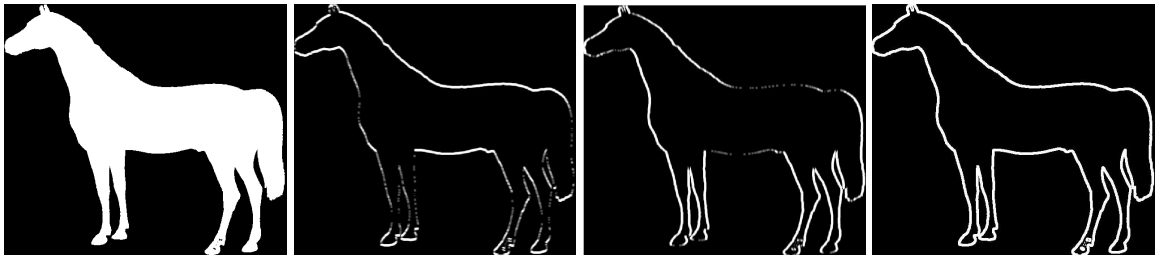
U ovoj ćemo sekciji promotriti rezultate dosad predstavljenih algoritama za detekciju ruba. Za svaki algoritam ćemo objasniti kako je implementiran, komentirati korištene parametre te demonstrirati kako radi na binarnim i grayscale slikama. Naposljetku ćemo međusobno usporediti algoritme i njihovu učinkovitost u detekciji ruba.

Ranije nam veličine slika nisu bile relevantne - više smo se fokusirali na veličine značajki koje tražimo, tj. točaka i linija. U ovoj ćemo sekciji u svakom primjeru navesti veličinu slike jer ona utječe na odabir veličine maske za izglađivanje - npr. efekt 5×5 maske neće biti jednako jak i očit kod slike dimenzija 128×128 i kod slike dimenzija 1024×1024. Također napominjemo da su intenziteti slika skalirani na interval $[0, 1]$.

Gradijentni detektor

Gradijentni detektor je najjednostavniji među predstavljenim detektorima ruba. Algoritam se sastoji od nekoliko koraka: slika se izgladi, zatim se izračuna aproksimacija magnitude gradijenta pa se dio vrijednosti s gradijentne slike odsječe. Premda je moguće pronaći rub bez izglađivanja i odsjecanja, očekujemo da će ti dodatni koraci kod kompleksnijih slika ublažiti odziv gradijenta na šum i sitne detalje.

Primjer 7.4.1. *Na slici 7.9a vidimo binarnu sliku konja dimenzija* 632×562 *piksela. Slika je jasno podijeljena na dvije regije različitog intenziteta, a rubovi na slici nisu mutni. Na ovakvoj slici nema potrebe za izglađivanjem i odsjecanjem jer nema šuma i ne postoje sitni detalji koje bismo htjeli ukloniti iz segmentacije.*



(a) Binarna slika konja. (b) Odziv $|g_x|$. (c) Odziv $|g_y|$. (d) Segmentirana slika.

Slika 7.9: Segmentacija binarne slike konja gradijentnom metodom. Slika konja je preuzeta s [28].

Nakon što je slika učitana, slika je zasebno filtrirana Sobelovim maskama 3.29 i 3.30 pomoću funkcije imfilter*, čime dobijemo slike 7.9b i 7.9c. Magnitudu (tj. gradijentnu*

sliku) izračunamo zbrajanjem apsolutnih vrijednosti odziva dvaju navedenih filtera, čime dobijemo konačnu segmentaciju 7.9d.

Primjer 7.4.2. *Slika 7.10a prikazuje sliku kuće u sivim tonovima dimenzija* 512×512 *piksela. Na slici možemo vidjeti detalje poput crjepova i cigli koje bi gradijentni detektor mogao prepoznati, a koje mi želimo eliminirati s konačne slike. Slika je nakon učitavanja izglađena* 5×5 *box filterom pomoću funkcije* imfilter*; veličina maske filtera je odabrana tako da ne zamuti sliku toliko da poremeti glavne rubove, ali da ipak izgladi područja s detaljima. Nakon toga su na isti način kao u primjeru 7.4.1 izračunate komponente gradijenta* g_x *i* g_y *te je izračunata gradijentna slika 7.10e. Međutim, na gradijetnoj slici primjećujemo da su se na slici zadržali neki sitni detalji (npr. cigle na dimnjaku), pa su metodom odsjecanja uklonjeni odzivi slabiji od* 5.5% *najveće vrijednosti na gradijentnoj slici te je dobivena konačna segmentacija 7.10f. Primjećujemo da su rubovi i u ovom i u prethodnom primjeru malo deblji (u oba slučaja je većina rubova debljine preko* 4 *piksela), što obično nije poželjno svojstvo.*

(a) Slika kuće. (b) Izglađena slika. (c) Odziv $|g_x|$.

(d) Odziv $|g_y|$. (e) Gradijentna slika. (f) Segmentirana slika.

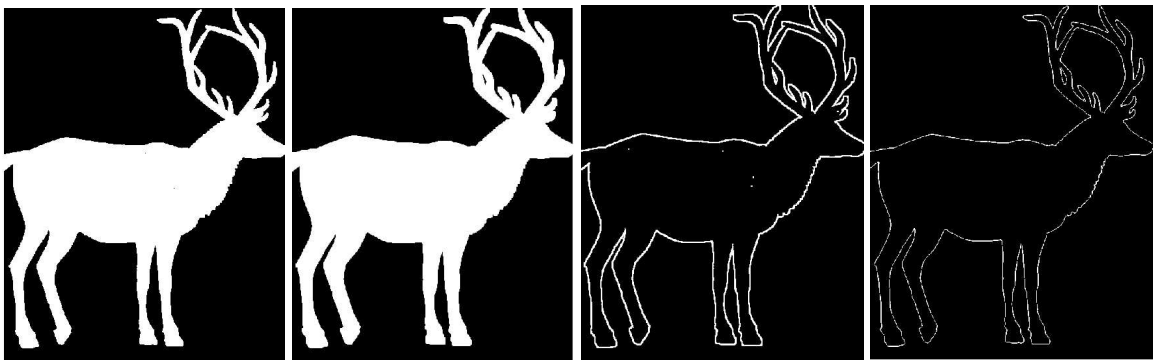
Slika 7.10: Segmentacija slike kuće gradijentnom metodom. Slika kuće je preuzeta s [28].

Marr-Hildrethin detektor

Kao "zero crossing" detektor (tj. detektor prijelaza kroz nulu), Marr-Hildrethin algoritam identificira točke ruba tražeći točke prijelaza kroz nulu druge derivacije. Algoritam se oslanja na LoG operator, s tim da se prvo izvrši radnja izglađivanja Gaussovim filterom, a tek onda primijeni Laplaceov filter. Na dobivenoj se slici točke prijelaza kroz nulu identificiraju analizom 8-susjedstva svakog piksela: traži se par piksela suprotnih predznaka čija apsolutna razlika prelazi neki pozitivni prag.

Primjer 7.4.3. *Započet ćemo binarnom slikom kao i kod gradijentnog detektora. Slika 7.11a je dimenzija* 550×691 *piksela. Kod binarne slike bez šuma nije potrebno prejako izgladiti sliku, pa ćemo primijeniti Gaussov filter s maskom veličine* 5×5 *i* $\sigma = 1.0$*. Na slici 7.11b vidimo rezultat izglađivanja - slika se nije puno promijenila, a na nekim mjestima se čini da je došlo da blagog zadebljanja (npr. kod rogova). Izglađenu sliku filtriramo Laplacianom, nakon čega slijedi detekcija točaka prijelaza kroz nulu.*

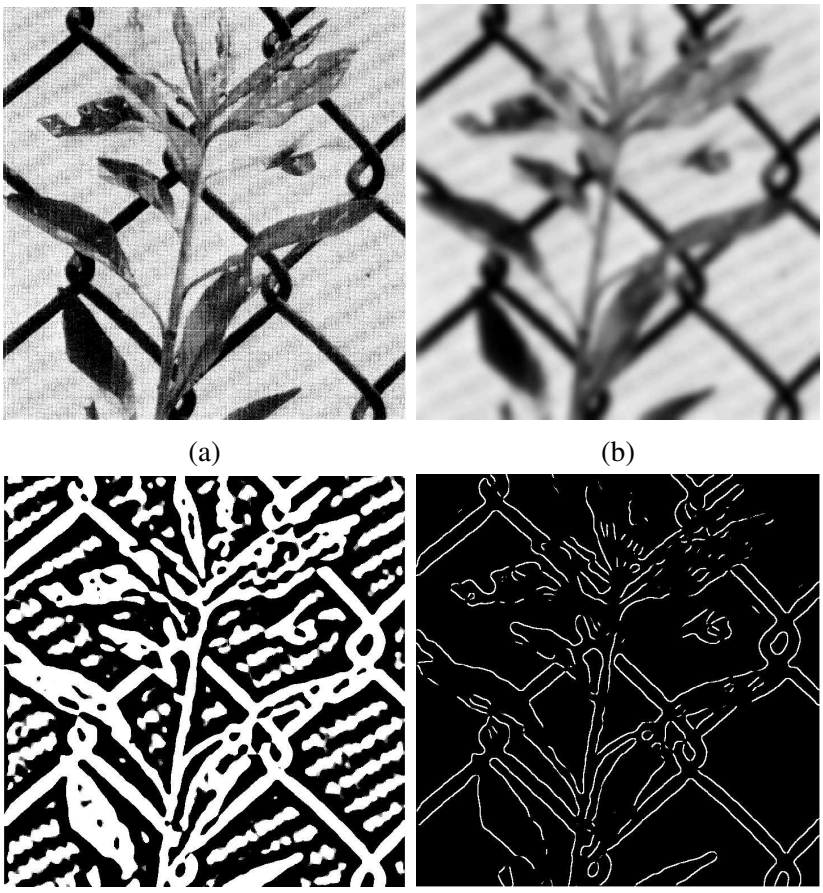
Prijelaze kroz nulu detektiramo provjerom svakog piksela i njegove okoline, a kako bismo mogli provjeriti piksele granice, potrebno je nadopuniti sliku funkcijom padarray *(npr. simetričnom nadopunom). U nadopunjenoj slici putujemo po pikselima nenadopunjene slike i u svakom pikselu vršimo provjeru suprotnih parova susjeda - čim nađemo prvi par suprotnih predznaka čija apsolutna razlika prelazi prag, piksel označavamo kao točku prijelaza kroz nulu, odnosno točku ruba. Prag je u ovom slučaju odabran kao* 75% *najveće vrijednosti na slici prije detekcije točaka ruba.*



(a) (b) (c) (d)

Slika 7.11: Segmentacija binarne slike jelena Marr-Hildrethinim detektorom. Sliku (a) prvo izgladimo Gaussovim filterom (vidi (b)), a zatim na tu sliku primijenimo Laplaceov filter (vidi (c)). Detekcijom točaka prijelaza kroz nulu na dobivenoj slici dobijemo sliku ruba (d). Slika jelena je preuzeta s [28].

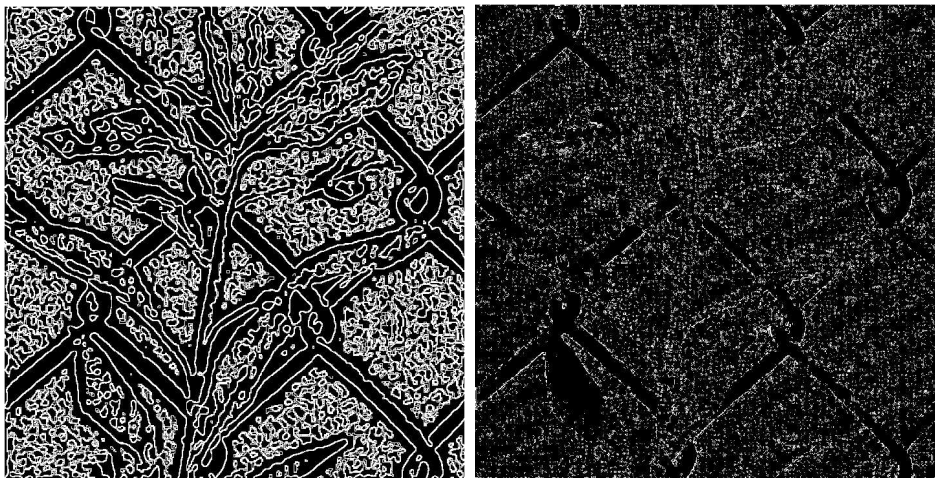
Primjer 7.4.4. *Sad ćemo pokazati djelovanje Marr-Hildrethinog algoritma na slici u sivim tonovima. Kao primjer ćemo koristiti šumovitu sliku biljke i žičane ograde 7.12 koja je korištena u članku u kojem je algoritam predstavljen. Slika ima dosta šuma, pa ćemo je jače izgladiti - koristit ćemo Gaussovu masku veličine* 49×49 *i* $\sigma = 7.0$*. Na slici 7.12b vidimo da je dio šuma očišćen, premda je dio još uvijek vidljiv u pozadini žičane ograde (npr. tamne dijagonalne linije). Nakon izglađivanja primjenjujemo Laplaceov filter i dobijemo sliku 7.12c, na kojoj ćemo onda uz prag* $T = 12\%$ *maksimalne vrijednosti slike tražiti prijelaze kroz nulu. Slika 7.12d prikazuje konačnu segmentaciju. Žice su dosta dobro uhvaćene i većina biljke je detektirana, s tim da su neki svijetli dijelovi izgubljeni.*



(a) (b)

(c) (d)

Slika 7.12: Segmentacija 730×749 slike biljke i ograde Marr-Hildrethinim detektorom. Originalna slika je preuzeta iz [21].



Slika 7.13: Primjeri loših segmentacija. Na lijevoj slici vidimo primjer korištenja preniskog praga pri detekciji točaka ruba (zatvorene konture), a na desnoj vidimo rezultat nedovoljnog izglađivanja slike (odziv je poremećen šumom). Prejakim izglađivanjem i previsokim pragom se gube rubovi u odzivu.

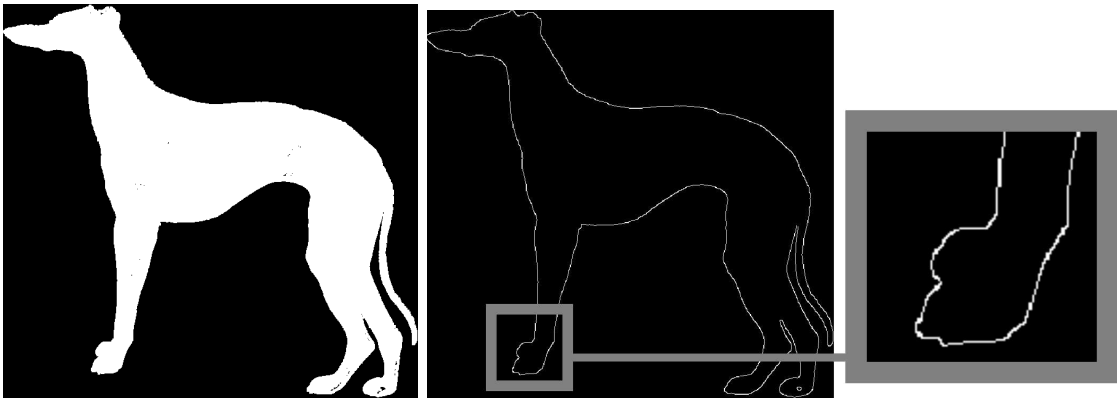
Cannyjev detektor

Na kraju se osvrnimo na Cannyjev detektor ruba. Cannyjev detektor je najkompleksniji, ali i najučinkovitiji od predstavljenih algoritama, što pokazuje činjenica da je i danas u širokoj uporabi, gotovo 40 godina nakon svog izuma. Algoritam je implementiran u *Octaveu* funkcijom edge ([6]), koja kao argumente prima sliku, 2D vektor koji kao vrijednosti ima gornji i donji histerezni prag te standardnu devijaciju σ izglađujućeg filtera. U slučaju da se umjesto vektora pragova kao argument zada samo jedan od pragova (tj. skalar), algoritam taj prag prihvati kao gornji prag, a kao donji prag koristi 40% vrijednosti gornjeg praga. Zadana vrijednost standardne devijacije σ je $\sqrt{2}$.

U sljedećim primjerima ćemo demonstrirati rad Cannyjevog algoritma na različitim slikama te navesti uz koje smo parametre dobili dane rezultate.

Primjer 7.4.5. *Započnimo s binarnom slikom psa 7.14a. Kao što vidimo na slici 7.14b, Cannyjev algoritam je bez problema detektirao rub. Zadane vrijednosti praga i standardne devijacije su u segmentaciji zadržale crne rupe (prisutne u stražnjim šapama psa), no odabirom gornjeg praga T_H = 0.7, rubovi oko tih rupa mogu se djelomično ukloniti sa slike. Dobiveni rub je tanak - čitav je debljine jednog piksela.*

Primjer 7.4.6. *Demonstrirajmo rad Cannyjevog algoritma na jednoj slici u sivim tonovima. Na slici 7.15a se nalazi amfora s puno detalja, s tim da su detalji na gornjem (originalnom) dijelu amfore naglašeniji nego oni na donjem (restauriranom) dijelu. Poigramo*



(a) (b)

Slika 7.14: Segmentacija binarne slike psa pomoću Cannyjevog detektora ruba. Na uvećanom dijelu možemo vidjeti glatkoću i tankoću ruba. Slika psa je preuzeta s [28].

li se s parametrima, možemo podesiti rezultat tako da se na slici ruba prikaže rub amfore, ali se očuvaju i detalji na njenoj gornjoj polovici. Slika 7.15b je dobivena uz gornji prag 0.175 *i* σ = 2.0*. Blago povećanje* σ *je rezultiralo ublažavanjem slabije vidljivih detalja.*

(a) (b)

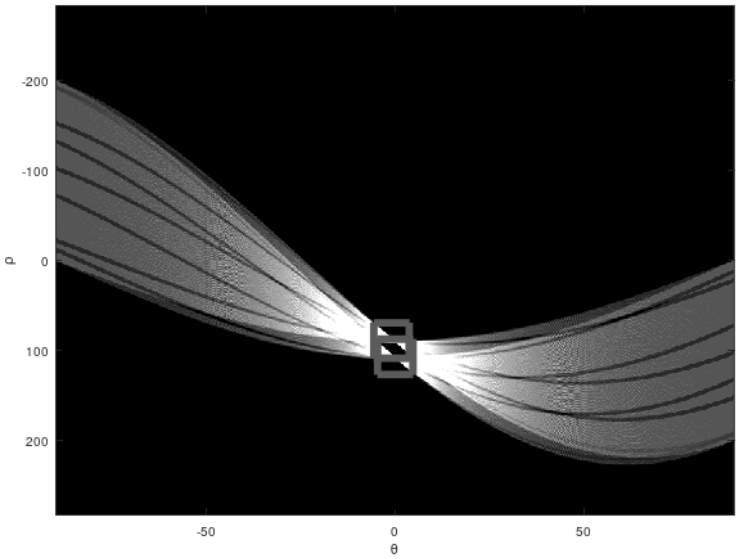
Slika 7.15: Segmentacija slike amfore pomoću Cannyjevog detektora ruba. Slika amfore je preuzeta s [28].

7.5 Houghova pretvorba

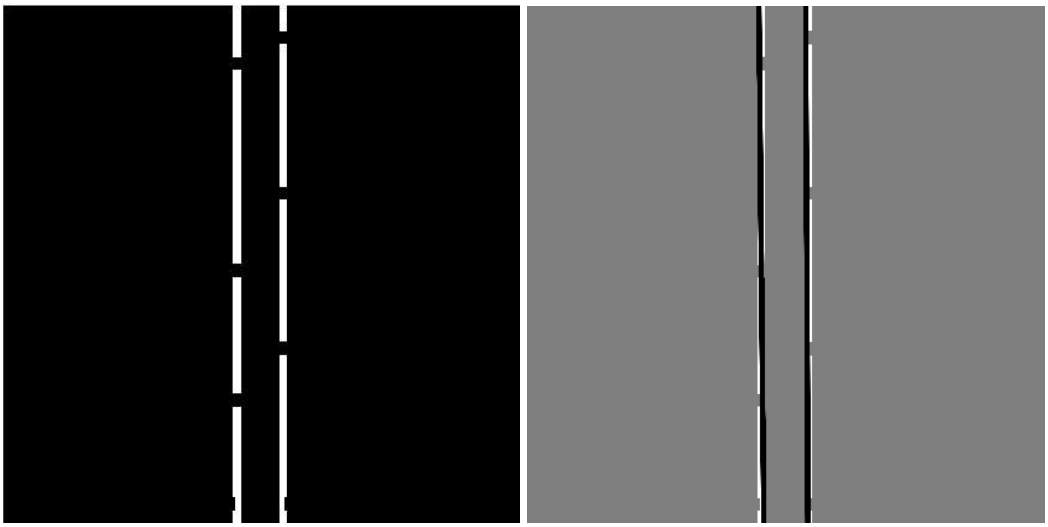
Među tri opisana algoritma povezivanja ruba, Houghova pretvorba je najznačajniji i najrobusniji algoritam. Generalizirana verzija algoritma se danas koristi u području računalnog vida, npr. kod detekcije predmeta specifičnog oblika (poput prometnih znakova, kovanica i sl.). Demonstrirat ćemo na jednom primjeru klasičnu Houghovu pretvorbu za linije.

Primjer 7.5.1. *U Octaveu je klasična Houghova pretvorba implementirana pomoću funkcije* hough *([6]), koja kao argument prima binarnu sliku f i vraća prikaz slike u parametarskom prostoru $\rho\theta$. Na slici 7.16 vidimo transformaciju dviju paralelnih isprekidanih linija (prikazanih kasnije na slici 7.17a). U istom paketu funkcija nalazi se i funkcija* houghpeaks*, koja kao argument prima Houghovu transformaciju slike, a vraća akumulacijske ćelije s najvećim vrijednostima. Moguće je odabrati broj ćelija koje će funkcija vratiti. Na slici transformacije vidimo* 2 *takve ćelije označene tamnosivim kvadratićima.*

Konačno, paket sadrži i funkciju houghlines *koja na danoj slici izvlači segmente linija iz Houghove pretvorbe. Moguće je odabrati minimalnu duljinu linija koje će funkcija razmatrati, kao i maksimalnu veličinu procjepa koje će popuniti. Rezultat koji je funkcija* houghlines *dala uz maksimalnu veličinu procjepa* 10 *piksela može se vidjeti na slici 7.17b.*



Slika 7.16: Houghova transformacija dviju isprekidanih linija. Na grafu možemo očitati da se radi o dvjema vertikalnim linijama.



(a) (b)

Slika 7.17: (a) Slika dviju vertikalnih, isprekidanih linija. (b) Slika istih linija na koju su dodane dvije crne linije koje je prepoznala funkcija houghlines. Pozadina slike je promijenjena u sivu boju radi bolje vidljivosti povezanih linija.

Bibliografija

[1] *History of USGS Astrogeology image processing software*, https://isis.astrogeology.usgs.gov/documents/IsisHistory/IsisHistory.html, Accessed: 2024-02-25.

[2] *Luna 3*, https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1959-008A, Accessed: 2024-02-25.

[3] *Ranger 7*, https://science.nasa.gov/mission/ranger-7, Accessed: 2024-02-25.

[4] Fred C. Billingsley, *Applications of digital image processing.*, Applied optics **9 2** (1970), 289–99.

[5] John Canny, *A computational approach to edge detection*, IEEE Transactions on pattern analysis and machine intelligence (1986), br. 6, 679–698.

[6] Octave Forge Community, *Octave Image Processing Package*, https://octave.sourceforge.io/image/overview.html.

[7] Edward Roy Davies, *Design of optimal Gaussian operators in small neighbourhoods*, Image Vis. Comput. **5** (1987), 199–205.

[8] E.R. Davies, *Image Processing For The Food Industry*, Series In Machine Perception And Artificial Intelligence, World Scientific Publishing Company, 2000, ISBN 9789814494410.

[9] _____, *Computer and Machine Vision: Theory, Algorithms, Practicalities*, Elsevier Science, 2012, ISBN 9780123869081.

[10] Geoff Dougherty, *Digital image processing for medical applications*, Cambridge University Press, 2009.

[11] W. Döler, N. Steinhöfel i A. Jäger, *Digital image processing techniques for cephalometric analysis*, Computers in Biology and Medicine **21** (1991), br. 1, 23–33, ISSN 0010-4825.

[12] Reza Farrahi Moghaddam i Mohamed Cheriet, *RSLDI: Restoration of single-sided low-quality document images*, Pattern Recognition **42** (2009), br. 12, 3355–3364, ISSN 0031-3203, New Frontiers in Handwriting Recognition.

[13] Robert Fisher, S. Perkins, A. Walker i E. Wolfart, *Hypermedia Image Processing Reference (HIPR)*, Artificial Intelligence - AI (1996).

[14] Pascal Getreuer, *A Survey of Gaussian Convolution Algorithms*, Image Processing On Line **3** (2013), 286–310.

[15] R.C. Gonzalez i R.E. Woods, *Instructor's Manual for Digital Image Processing*, Addison-Wesley, 1993, ISBN 9780201569445.

[16] Paul Hough, *Method and means for recognizing complex patterns.*, 1962.

[17] Jo Jenkinson, Artyom M. Grigoryan, Mehdi Hajinoroozi, Raquel Díaz-Hernández, Hayde Peregrina-Barreto, Ariel E. Ortiz Esquivel, Leopoldo Altamirano Robles i Vahram Chavushyan, *Machine learning and image processing in astronomy with sparse data sets*, 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2014), 200–203.

[18] Reinhard Klette i Azriel Rosenfeld, *Digital geometry: Geometric methods for digital picture analysis*, Morgan Kaufmann, 2004.

[19] Edwin Lechuga-S., Juan Carlos Valdiviezo-N. i Gonzalo Urcid, *Multispectral image restoration of historical documents based on LAAMs and mathematical morphology*, Optics & Photonics - Optical Engineering + Applications, 2014.

[20] Fuhai Li, Zheng Yin, Guangxu Jin, Hong Zhao i Stephen T.C. Wong, *Chapter 17: Bioimage Informatics for Systems Pharmacology*, PLoS Computational Biology **9** (2013), br. 4, ISSN 1553-734X.

[21] D. Marr i E.C. Hildreth, *Theory of Edge Detection*, AI memo, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1979.

[22] D. Marr i S. Ullman, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, The MIT Press, MIT Press, 2010, ISBN 9780262288989.

[23] M.D. McFarlane, *Digital pictures fifty years ago*, Proceedings of the IEEE **60** (1972), br. 7, 768–770.

[24] K. Miura i N. Sladoje, *Bioimage Data Analysis Workflows – Advanced Components and Methods*, Learning Materials in Biosciences, Springer International Publishing, 2022, ISBN 9783030763930.

[25] M.S. Nixon i A.S. Aguado, *Feature Extraction and Image Processing*, Academic, 2008, ISBN 9780123725387.

[26] J. Ohser i K. Schladitz, *3D Images of Materials Structures: Processing and Analysis*, Wiley, 2009, ISBN 9783527628315.

[27] Judith MS Prewitt et al., *Object enhancement and extraction*, Picture processing and Psychopictorics **10** (1970), br. 1, 15–19.

[28] Richard E. Woods Rafael C. Gonzalez i Steven L. Eddins, *ImageProcessingPlace.com*, `https://www.imageprocessingplace.com/root_files_V3/image_databases.html`, Accessed: 2024-02-14.

[29] Marius Rensen, *The Bartlane System (Coded System)*, `http://www.hffax.de/history/html/bartlane.html`, Accessed: 2024-02-24.

[30] L.G. Roberts, *Machine Perception of Three-dimensional Solids*, Its Technical report, M.I.T. Lincoln Laboratory, 1963.

[31] Chris Rorden, Roger Newman-Norlund, Chris Drake, Daniel R. Glen, Julius Fridriksson, Taylor Hanayik i Paul A. Taylor, *Improving 3D Edge Detection for Visual Inspection of MRI Coregistration and Alignment*, bioRxiv (2022).

[32] Irwin Scollar, *Image processing via computer in aerial archaeology*, Computers and the Humanities **11** (1977), 347–351.

[33] I.E. Sobel, *Camera Models and Machine Perception*, Memo (Stanford artificial intelligence Laboratory), Stanford University, 1970.

[34] B.C. Wang, *Digital Signal Processing Techniques and Applications in Radar Image Processing*, Information and Communication Technology Series, Wiley, 2008, ISBN 9780470180921.

[35] Richard E. Woods i Rafael C. Gonzalez, *Digital image processing*, Pearson Education Ltd., 2008.

[36] Richard E. Woods, Rafael C. Gonzalez i Steven L. Eddins, *Digital Image Processing Using MATLAB*, Gatesmark Publishing, 2020, ISBN 9780982085417.

Sažetak

Segmentacija slike bavi se njenom podjelom na sastavne dijelove ili objekte, a detaljnost te podjele ovisi o samoj primjeni slike. Segmentacija netrivijalnih slika je jedan od najizazovnijih zadataka kod procesiranja slika. Metode za detektiranje izoliranih točaka, linija ili rubova temelje se na otkrivanju oštre, lokalne promjene intenziteta. Pikseli ruba su pikseli kod kojih dolazi do nagle promjene intenziteta funkcije slike, a sam rub, ili bolje rečeno segment ruba, je skup povezanih piksela ruba. Takve nagle promjene funkcije slike otkrivaju se proučavanjem njene prve i druge derivacije, odnosno njihovih aproksimacija. U radu su obrađene osnovne metode za detektiranje izoliranih točaka i linija. Posebna pažnja se posvećuje detekciji rubova jer takvi algoritmi imaju najširu primjenu i vrijednu ulogu u analizi slika. Opisano je nekoliko metoda detekcije ruba, među kojima je najznačajniji Cannyjev algoritam. Kako algoritmi detekcije ponekad vraćaju nepovezane rubove, često su popraćeni algoritmima povezivanja rubnih segmenata, pa je predstavljeno i nekoliko metoda povezivanja. Na kraju rada su prikazani rezultati obrađenih algoritama detekcije na konkretnim primjerima, implementiranim u softverskom paketu *Octave*.

Summary

Segmentation subdivides an image into its constituent regions or objects, and the level of detail to which the subdivision is carried depends on the application of the image. Segmentation of nontrivial images is one of the most challenging tasks in image processing. Methods of detecting isolated points, lines and edges are based on detecting a sharp, local change in intensity. Edge pixels are pixels at which the intensity of the image function changes abruptly, and an edge, or more precisely an edge segment, is a set of connected edge pixels. Such abrupt changes in intensity in the image function are detected using its first and second derivatives, i.e. their approximations. This thesis describes basic methods of detecting isolated points and lines. Special attention is afforded to edge detection, as edge detection algorithms have a wide range of application and play a valuable role in image analysis. Several edge detection algorithms are described, the most significant of which is the Canny detector. Seeing how edge detection algorithms sometimes produce broken edges, they are often accompanied by edge linking algorithms, so several such methods are described as well. At the end of the thesis, we showcase the results of the described detection algorithms on concrete examples, implemented in *Octave*.

Životopis

Rođena sam 22. kolovoza 1998. u Šibeniku, gdje sam pohađala i završila Osnovnu školu Vidici i Gimnaziju Antuna Vrančića. 2016. započinjem preddiplomski studij matematike na Prirodoslovno-matematičkom fakultetu u Zagrebu, a 2020. stječem titulu sveučilišnog prvostupnika matematike i upisujem diplomski smjer Primijenjena matematika. Tijekom studiranja sam bila aktivni član studentske udruge PRIMUS i Studentskog zbora PMF-a, a sudjelovala sam u organizaciji Dana karijera na PMF-u *WorkIn' Science* i karijernog događaja *Meet the Mathematicians*. U srpnju 2021. sudjelovala sam u Lautenschläger ljetnoj školi na institutu za molekularnu biologiju EMBL Heidelberg, a u jesen 2023. sam radila na institutu za industrijsku matematiku Fraunhofer ITWM u Kaiserslauternu u sklopu Erasmus+ prakse.