Programsko rješenje filtra za detekciju rubova na slici

Gubec, Sandra

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Sandra Gubec

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Sandra Gubec

Zagreb, 2023.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru izv. prof. dr. sc. Tomislavu Stipančiću na stručnom vođenju, pristupačnosti, povjerenju te pruženoj pomoći tijekom pisanja ovog rada.

Posebno se zahvaljujem svojoj obitelji na razumijevanju, strpljenju i neizmjernoj podršci.

Također zahvaljujem svojim prijateljima i kolegama koji su mi tijekom cijelog studija bili velika pomoć i podrška.

Sandra Gubec

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 23 - 6 / 1**	
Ur.broj: **15 - 23 -**	

DIPLOMSKI ZADATAK

Student: **Sandra Gubec** JMBAG: 0035217506

Naslov rada na hrvatskom jeziku: **Programsko rješenje filtra za detekciju rubova na slici**

Naslov rada na engleskom jeziku: **Program solution of the filter for detecting edges in the image**

Opis zadatka:

Pronalaženje rubova (kontura) objekata na slikama važan je početni korak kod mnogih zadaća računalnog i strojnog vida. Između ostaloga, konture kao granice objekata na slici predstavljaju temelj za pronalaženje, lokalizaciju, segmentaciju i praćenje. Analizu rubova je moguće provoditi u prostornoj i frekvencijskoj domeni koristeći različite korake koji mogu utjecati na kvalitetu lokalizacije i detekcije rubova.

U radu je potrebno:

- objasniti značenje rubova kao nositelja informacija na slikama,
- odrediti i objasniti korake koji vode ka računalnom prepoznavanju rubova na slici,
- implementirati korake za prepoznavanje rubova u računalni program,
- usporediti više metoda za određivanje rubova u prostornoj i frekvencijskoj domeni, te
- ponuditi svoje rješenje filtera za detekciju rubova koje je sastavljeno od više koraka.

Razvijeno programsko rješenje je potrebno temeljiti na OpenCV biblioteci implementiranoj kroz Python programski jezik. Konačno rješenje je potrebno eksperimentalno evaluirati u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava. U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
28. rujna 2023.	30. studenoga 2023.	4. – 8. prosinca 2023.

Zadatak zadao: Predsjednik Povjerenstva:

Izv. prof. dr. sc. Tomislav Stipančić Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
a, a_k	m	pomoćna veličina, ne-negativni cijeli broj, amplituda
$a_0/2$	-	član za translaciju funkcije
A	m	amplituda
b	-	pomoćna veličina, ne-negativni cijeli broj
$\mathbf{b_j}$	-	izlazni signal neurona
b_k	-	pomoćna veličina
c_k	-	pomoćna veličina
c_0	-	član za translaciju funkcije
$E(u,v)$	-	jačina lokalnog ruba
$f(x,y)$	-	funkcija ulazne slike
$F(u,v)$	-	Fourierova transformacija funkcije slike
$g(x,y)$	-	funkcija filtrirane slike
$G(s,t)$	-	Gaussova funkcija
$G(u,v)$	-	konačna filtrirana funkcija
$H(u,v)$	-	transferna funkcija
$H'(u,v)$	-	Fourierova transformacija niskopropusnog filtriranja
$\mathbf{H_x^D}$	-	horizontalni gradijentni filter
$\mathbf{H_y^D}$	-	vertikalni gradijentni filter
$\nabla I(u,v)$	-	gradijent funkcije
k	rad	frekvencija
K	-	konstanta
L	rad	period
m	-	broj redaka kernela
m_f, m_b	-	srednja vrijednost
n	-	broj stupaca kernela
P	rad	period
r	m	udaljenost središta od bilo koje točke funkcije *G*
s	-	parametar sumiranja, pomak
$\mathrm{S_v}$	-	veličina korelacijskog/konvolucijskog niza
$\mathrm{S_h}$	-	veličina korelacijskog/konvolucijskog niza
t	-	parametar sumiranja, pomak
$\mathbf{T_j}$	-	prag osjetljivosti
$\mathbf{v}$	-	vektor koji definira 2D kernel
$\mathbf{w}$	-	kernel filtera
$\mathbf{w_i}$	-	težinski koeficijent
$W(s,t)$	-	funkcija filtera

x	-	koordinata slike
x_0	-	koordinata diskretnog impulsa snage
y	-	koordinata slike
y_0	-	koordinata diskretnog impulsa snage
$\boldsymbol{\delta}$	-	diskretni impuls snage
ξ	-	medijan
Φ_k	rad	faza sinusne funkcije
$\Phi(u,v)$	-	lokalna orijentacija ruba
σ	m	standardna devijacija

SAŽETAK

U doba digitalne obrade slika, primjena postupaka filtriranja ima značajnu ulogu u poboljšanju i manipulaciji slikama sa svrhom izvlačenja korisnih informacija i unaprjeđenja njihove vizualne kvalitete. Filtriranje se odvija u dvije domene, prostornoj i frekvencijskoj. Filtriranje u prostornoj domeni izravno utječe na vrijednosti piksela, dok se filtriranje u frekvencijskoj domeni temelji na Fourierovoj transformaciji. Iako različitog pristupa, obje tehnike imaju jednake zadatke – izoštravanje, zaglađivanje, smanjenje šuma te detekcija rubova na slici. Rubovi su nositelji informacija o strukturi i granicama objekata na slici. Definiraju oblike te pomažu pri prepoznavanju objekata. Detekcija rubova također se izvodi u prostornoj i frekvencijskoj domeni. U prostornoj domeni ona se temelji na upotrebi operatora gradijenta koji ističu područja nagle promjene intenziteta što ukazuje na prisutnost ruba. U frekvencijskoj domeni koriste se visokopropusni filteri kako bi se istaknule visokofrekventne komponente koje su karakteristične rubovima. Detekcija rubova značajna je u područjima medicinske dijagnostike, računalnog vida te sigurnosnih sustava.

Ključne riječi: filtriranje, prostorna domena, frekvencijska domena, rub, detekcija rubova

SUMMARY

In the era of digital image processing, the application of filtering procedures plays a significant role in improving and manipulating images to extract useful information and enhance their visual quality. Filtering occurs in two domains: spatial and frequency. Spatial domain filtering directly influences pixel values, while frequency domain filtering is based on Fourier transformation. Despite different approaches, both techniques have identical purpose – sharpening, smoothing, noise reduction, and edge detection in images. Edges are carriers of information about the structure and boundaries of objects in the image. They define shapes and help to recognize objects. Edge detection is performed in both spatial and frequency domains. In the spatial domain, it relies on the use of gradient operators that highlight areas of abrupt intensity changes, indicating the presence of an edge. In the frequency domain, high-pass filters are used to emphasize high-frequency components characteristic of edges. Edge detection is significant in areas such as medical diagnostics, computer vision, and security systems.

Key words: filtering, spatial domain, frequency domain, edge, edge detection

1. UVOD

U suvremenoj eri digitalne obrade slika, postupci filtriranja imaju ključnu ulogu u unaprjeđenju i manipulaciji slikama s ciljem ekstrakcije korisnih informacija ili poboljšanja njihove vizualne kvalitete. Filteri se uglavnom koriste za suzbijanje visokih frekvencija, što rezultira zaglađivanjem rubova slika, te niskih frekvencija, što rezultira poboljšanjem slike te detekcijom rubova [1]. Filtriranje slika predstavlja moćan alat za različite primjene uključujući primjenu kod računalnog vida, medicinske dijagnostike, obrade fotografija, videozapisa te animacija, biometrije te mnoge druge. Ono doprinosi izdvajanju relevantnih informacija sa slike te pomaže u analizi i vizualnoj obradi u različitim područjima. Dva ključna pristupa u filtriranju slika jesu filtriranje u prostornoj i frekvencijskoj domeni. Oba pristupa nude jedinstvene prednosti za poboljšanje slika, no razlikuju se po svojim osnovnim konceptima i tehnikama. Filtriranje slika u prostornoj domeni odnosi se na obradu slika izravno u njihovom prostornom obliku gdje je slika predstavljena pikselima koji su složeni u matricu. Temelji se na izvođenju operacija u susjedstvu pojedinog piksela. Filtriranje u prostornoj domeni je tradicionalna i široko korištena metoda u digitalnoj obradi slika, a često je primjenjivana za uklanjanje šuma, izoštravanje kontura, a ponajviše za poboljšanje kvalitete slika. Poboljšanje slike je proces manipulacije slike kako bi konačan rezultat bio pogodniji od originala za određenu primjenu. S druge strane, filtriranje u frekvencijskoj domeni je napredniji pristup koji temelji svoje osnove na Fourierovoj transformaciji. Fourierova transformacija prenosi sliku iz prostorne u frekvencijsku domenu gdje se analizira pomoću frekvencijskih komponenata što omogućuje precizniju manipulaciju i filtriranje slika. Rubovi su jedni od značajnijih nositelja informacija na slikama. Oni se pojavljuju na granicama između područja različite boje, intenziteta i teksture. Detekcija rubova je pristup koji se često koristi za segmentaciju slika na temelju naglih (lokalnih) promjena u intenzitetu. Modeli rubova klasificiraju se prema svojim profilima intenziteta [2].

U ovome će se radu dati detaljna analiza filtriranja slika u prostornoj i frekvencijskoj domeni popraćena primjerima te usporedbom. Objasnit će se značenje rubova kao nositelja informacija na slikama te odrediti koraci koji vode k računalnom prepoznavanju rubova na slici. Usporedit će se više metoda u prostornoj i frekvencijskoj domeni te ponuditi vlastito rješenje za detekciju rubova.

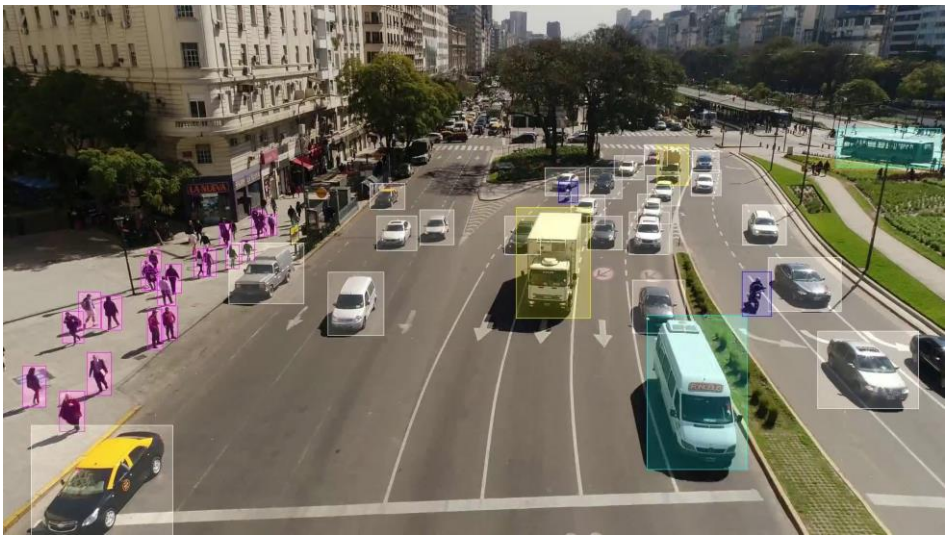
2. TEORIJSKA OSNOVA RADA

2.1. Računalni vid

Računalni vid je područje umjetne inteligencije koje omogućuje računalnim sustavima da izvlače značajne informacije iz digitalnih slika, videa i ostalih vizualnih ulaza te poduzima radnje, odnosno, daje preporuke na temelju tih informacija. Računalni vid funkcionira nalik ljudskom, no da bi naučio razlikovati objekte, odrediti njihovu udaljenost i kretanje potreban je niz podataka, kamera i algoritama kako bi umjesto mrežnice, optičkih živaca i vizualnog korteksa obavio navedene radnje u što kraćem vremenu. Moglo bi se reći kako računalni vid radi upravo inverzno od ljudskog, nastoji opisati svijet koji nas okružuje te rekonstruirati njegove karakteristike poput oblika, osvjetljenja te raspodjele boja. Za postizanje navedenog koriste se dvije ključne tehnologije: vrsta strojnog učenja zvana duboko učenje te konvolucijske neuronske mreže. Strojno učenje koristi algoritamske modele koji omogućuju računalu da samo uči o kontekstu vizualnih podataka. Potrebno je mnogo podataka kako bi računalo naučilo razlikovati objekte. Posljedično, algoritmi omogućuju stroju da uči samostalno umjesto da je specifično programirano da prepoznaje pojedine slike. Konvolucijske neuronske mreže pomažu dubokom učenju na način da slike razlamaju na piksele kojima se dodjeljuju oznake. Oznake predstavljaju matematičku operaciju konvolucije (dobivanje treće funkcije iz dvije) te iznosi predviđanja o onome što vidi. Neuronska mreža provodi konvolucije te provjerava točnost svojih predviđanja u nizu iteracija sve do trenutka kada predviđanja ne postanu istinita. U tome trenutku računalo prepoznaje slike poput čovjeka. Nalik ljudskom vidu, konvolucijske neuronske mreže prvo razaznaju oštre rubove i jednostavne oblike, a potom popunjavaju informacije ovisno o točnosti iteracija predviđanja [3]. Neki od primjera upotrebe računalnog vida u svakodnevnom životu jesu [4]:

- Optičko prepoznavanje znakova
- Provjera ispravnosti strojeva
- Maloprodaja
- Izrada 3D modela
- Medicinska dijagnostika
- Automobilska industrija
- Nadzor na cestama
- Prepoznavanje lica te otiska prsta

Optičko prepoznavanje znakova podrazumijeva primjerice čitanje rukom pisanih poštanskih brojeva na pismima te automatsko prepoznavanje registarskih pločica. O *Canny edge* metodi pričat će se u narednim poglavljima. Provjera ispravnosti stroja omogućuje brzu inspekciju dijelova za osiguranje kvalitete korištenjem stereo vida. U maloprodaji računalni vid služi kao sredstvo za prepoznavanje objekata kod automatiziranih blagajna, a u automobilskoj industriji kao pomoć pri otkrivanju prepreka. Usko povezano s automobilima, česta je primjena i osiguravanje sigurnosti vozila, odnosno, otkrivanje neočekivanih prepreka na cesti u uvjetima u kojima tehnike aktivnog vida kao što je radar ne rade. Prepoznavanje lica i otiska prsta služe za automatsku autentifikaciju, a jedan dio toga aspekta jest i prepoznavanje rubova koje će se detaljnije razraditi. Primjer upotrebe računalnog vida u prometu vidljiv je na slici 1 te u svrhu prepoznavanja na slici 2.



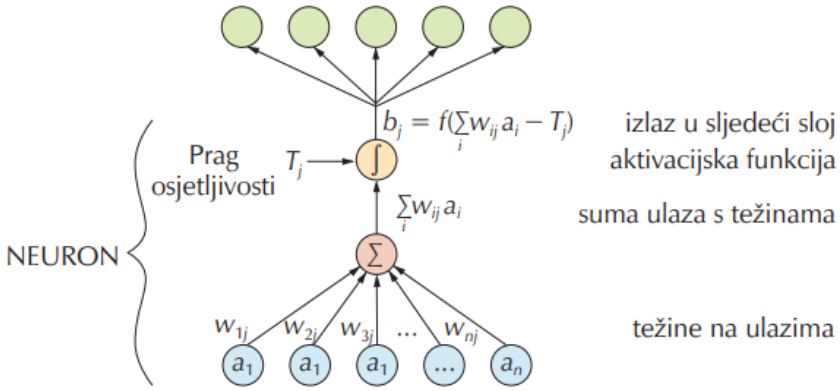
Slika 1. Primjena računalnog vida u prometu [23]

Slika 2. Primjena računalnog vida za prepoznavanje [24]

2.1.1. Neuronske mreže i duboko učenje

Neuronske mreže kao vrlo važan aspekt u području umjetne inteligencije služe kao sredstvo za adaptivno učenje parametara funkcija odlučivanja. Počeci razvoja ideje neuronskih mreža datiraju iz ranih 1940-ih kada su McCulloch i Pitts predložili modele neurona u obliku binarnih uređaja i stohastičkih algoritama s naglim promjenama stanja od 0 do 1 i obrnuto kao osnova za modeliranje neuronskih sustava. Tijekom sredine 1950-ih i ranih 1960-ih klasa takozvanih strojeva za učenje, koje je pokrenuo Rosenblatt, izazvala je velik preokret u razvoju neuronskih mreža. Razlog interesa u navedene strojeve, nazvane perceptronima, bio je razvoj matematičkih dokaza koji su pokazivali kako perceptroni, kada se treniraju s linearno odvojivim skupovima podataka za trening, odnosno skupovima koji se mogu odvojiti hiperplohom, konvergiraju prema rješenju u konačnom broju iterativnih koraka. Rješenje je poprimalo oblik parametara (koeficijenata) hiperploha koji su bili u mogućnosti ispravno odvajati klase predstavljene uzorcima skupova podataka za trening. Iako je osnovni perceptron, koji je prikazan na slici 3., naišao na prepreke u smislu neadekvatnosti za većinu zadataka prepoznavanja uzoraka i dalje se težilo njegovom poboljšanju. Kasniji rezultati razvoja algoritama za obuku višeslojne jedinice nalik na perceptron značajno su promijenile stvari. Naime, metoda zvana *backpropagation* pružala je učinkovitu tehniku za obuku višeslojne mreže. Iako navedeni algoritam nije bio u mogućnosti konvergirati rješenju u smislu dokaza za jednoslojni perceptron, uspješno je generirao rezultate koji su revolucionirali polje prepoznavanja uzoraka. Ova vrsta višeslojnog učenja naziva se dubokim učenjem, a njegova praktična implementacija povezana je s velikim skupovima podataka [2]. Neuronske mreže u kontekstu današnjice nastoje oponašati mrežu neurona koja čini ljudski mozak s namjerom da računala imaju mogućnost razumjeti i donositi odluke poput ljudi. Svaka neuronska mreža sastoji se od ulaznog, skrivenog te izlaznog sloja. Ulazni sloj poprima vrijednosti ulaznih veličina. Sinaptička operacija predstavlja množenje svakog ulaznog signala s težinskim koeficijentom, $\mathbf{w_i}$. Tako otežani ulazni signali se zbrajaju, a njihov zbroj uspoređuje se s pragom osjetljivosti neurona, $\mathbf{T_j}$. Težinski faktori analogni su dendritima biološkog neurona. Skriveni sloj zbraja otežane ulaze pomoću funkcije sumiranja te se time stvara vlastita interna aktivacija. U slučaju da je zbroj otežanih signala veći od praga osjetljivosti neurona, nelinearna funkcija f generira izlazni signal neurona iznosa $\mathbf{b_j}$. Prijenosna funkcija može biti diskontinuirana, skokomična ili kontinuirana funkcija poput sigmoide ili tangens – hiperbolne funkcije. Učenje neuronskih mreža jest iterativan postupak optimiranja vrijednosti težinskih faktora na temelju pogreške

između modelom proračunate vrijednosti i stvarne vrijednosti mjerene veličine. Podešavanje težinskih faktora radi se na temelju pravila širenja unatrag ili algoritma unatražne propagacije izlazne pogreške. Tijekom prolaza informacije kroz neuronsku mrežu generira se vrijednost koja se potom uspoređuje sa stvarnom vrijednošću. Na temelju razlike stvarne i izračunate vrijednosti korigiraju se težinski faktori. Njihovom korekcijom moguće je predviđati stvarne vrijednosti, a time se smanjuje razlika stvarnih i predviđenih vrijednosti izlaznih veličina [5].



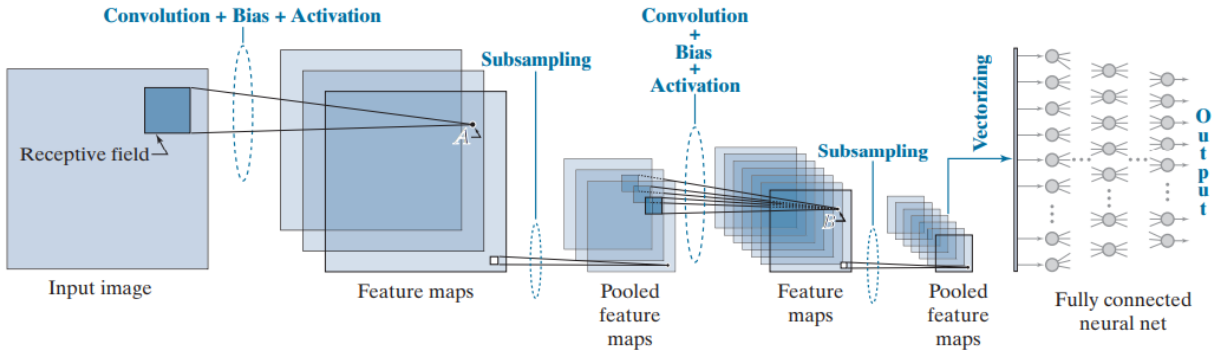
Slika 3. Osnovni perceptron [5]

2.1.2. Konvolucijske neuronske mreže

Osnovna razlika između običnih neuronskih i konvolucijskih neuronskih mreža (CNN) jest da su ulazi u običnim neuronskim mrežama vektori, a u konvolucijskim neuronskim mrežama 2D nizovi, odnosno slike. Unatoč tome, izračuni koje provode obje mreže vrlo su slični. Najprije se formira zbroj umnožaka, a potom s doda vrijednost *bias*. Nakon što se rezultat proslijedi kroz aktivacijsku funkciju, ona postaje ulaz u sljedeći sloj. Osim što se razlikuju po tome da su ulazni formati u konvolucijsku neuronsku mrežu 2D oblika naspram vektora, važna je razlika u tome što su konvolucijske neuronske mreže sposobne učiti 2D značajke izravno iz neobrađenih slikovnih podataka. Budući da ne postoje alati za sustavno projektiranje sveobuhvatnih skupova značajki za složene zadatke prepoznavanja slika, posjedovanje sustava koji može naučiti vlastite značajke slike iz neobrađenih slikovnih podataka ključna je prednost konvolucijske neuronske mreže. Još jedna od razlika jest način na koji su slojevi povezani. U običnoj neuronskoj mreži izlaz svakog neurona u sloju dovodi se izravno u ulaz svakog neurona u sljedećem sloju. Nasuprot tome, kod konvolucijske neuronske mreže u svaki ulaz sloja unosi se jedna vrijednost koja je određena konvolucijom preko prostornog susjedstva u izlazu prethodnog sloja. Još jedna od značajnih razlika jest da se 2D nizovi od jednog do drugog sloja poduzorkuju kako bi se smanjila osjetljivost na translacijske varijacije na ulazu. Osnovna

operacija koja se izvodi jest prostorna konvolucija. Ona izračunava zbroj umnožaka između piksela i skupa težina kernela te se provodi na svakoj prostornoj lokaciji na ulaznoj slici. Rezultat na svakoj lokaciji u ulazu je skalarna vrijednost. Na slici 4. s krajnje lijeve strane vidljivo je susjedstvo na jednoj lokaciji na ulaznoj slici. Ta se susjedstva nazivaju i receptivnim poljima te onda odabiru regije piksela na ulaznoj slici. Prva operacija koja se potom provodi jest konvolucija, čije se vrijednosti zatim generiraju kako se receptivno polje pomiče po slici te se formira zbroj produkata skupa težina i piksela sadržanih u receptivnom polju. Skup težina, raspoređenih u obliku receptivnog polja, naziva se kernel. Broj prostornih inkremenata za koje s receptivno polje pomiče naziva se korak. Svakoj vrijednosti konvolucije dodaje se *bias,* a potom se rezultat šalje kroz aktivacijsku funkciju kako bi se generirala jedna vrijednost. Navedena se vrijednost šalje na odgovarajuću lokaciju na ulazu sljedećeg sloja. Ponavljanje postupka za sve lokacije na ulaznoj slici rezultira 2D skupom vrijednosti koji se pohranjuje u sljedeći sloj kao 2D niz, a naziva se kartom značajki. Taj naziv sugerira kako je uloga konvolucije upravo izvlačenje značajki poput rubova, točaka, mrlja, što znači kako je konvolucija zapravo osnova prostornog filtriranja za zadatke poput izoštravanja, zamućivanja i izračunavanje rubova na slici. Na slici 4. vidljive su tri karte značajki u prvom sloju mreže. Druge dvije karte značajki generiraju se korištenjem različitih skupova težina i *biasa* za svaku kartu značajki. Budući da je svaki skup težina i *biasa* drugačiji, svaka će karta značajki sadržavati različit skup značajki iako su sve izvučene iz jednake ulazne slike. Karte značajki se zajednički nazivaju konvolucijski sloj. Dakle, konvolucijska neuronska mreža prikazana na slici 4. ima dva konvolucijska sloja. Nakon konvolucije i aktivacije dolazi do poduzorkovanja. Tijekom treniranja konvolucijske neuronske mreže s velikim bazama slika, poduzorkovanje, odnosno, udruživanje, ima dodatnu prednost smanjenja količine podataka koji se obrađuju. Udruživanje se izvršava podjelom karte značajki u skup manjih regija, koje se nazivaju udruženim susjedstvima, i zamjenom svih elemenata u jednom od susjedstva jednom vrijednošću. Udružene karte značajki zajednički se nazivaju sloj udruživanja. Za višestruke ulaze u drugi sloj, postupak je sljedeći. Generiranjem vrijednosti za prvu kartu značajki u drugom konvolucijskom sloju potreban je postupak konvolucije, dodavanje *biasa* te aktivacije. Potom se mijenja kernel i bias te se ponavlja postupak za drugu kartu značajki, koristeći jednak ulaz. Na kraju se generiraju tri vrijednosti za istu lokaciju u svakoj karti značajki sa po jednom vrijednošću koja dolazi s odgovarajuće lokacije u svakom od tri ulaza. Način na koji se tri pojedinačne vrijednosti modu spojiti u jednu jest metoda superpozicije. Izlazi konvolucijske neuronske mreže jesu 2D nizovi (filtrirane slike smanjene rezolucije), dok su ulazi u potpuno

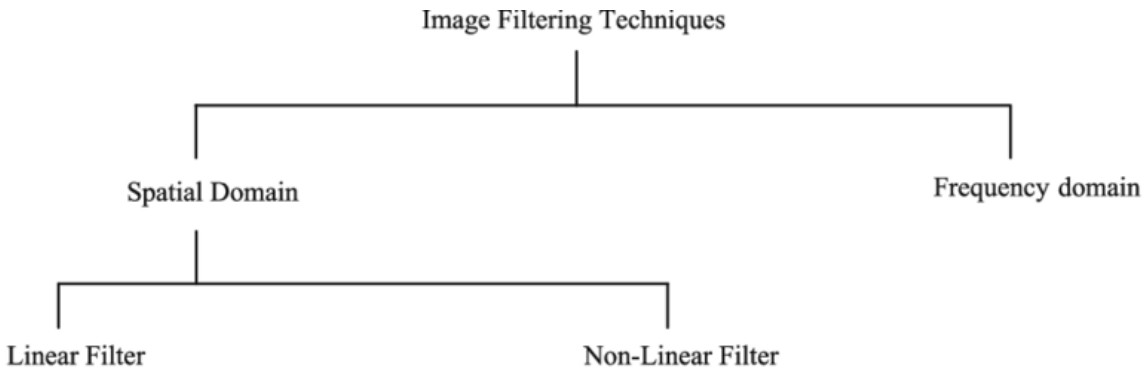
povezanu mrežu vektori. U posljednjem sloju događa se vektorizacija 2D udruženih karti značajki za što se koristi linearno indeksiranje. Svaki 2D niz u posljednjem polju se pretvara u vektor te se svi dobiveni vektori spajaju u jedan vektor. Taj se vektor potom širi kroz neuronsku mrežu. Broj izlaza u potpuno povezanoj mreži jednak je broju klasa uzoraka koji se klasificiraju. Izlaz s najvećom vrijednošću određuje klasu ulaza [2].



Slika 4. Konvolucijska neuronska mreža [2]

3. FILTRIRANJE SLIKE

Filtriranje slika je temeljna i svestrana tehnika u domeni obrade digitalne slike koja ima velik utjecaj na različite primjene poput računalnog vida te poboljšanja i manipulacije slika. U suštini filtriranje slike obuhvaća raznovrstan skup algoritama i metoda osmišljenih za izmjenu ili poboljšanje vizualnih karakteristika slike. Proces je važan za izdvajanje značajnih informacija, smanjenje šuma i poboljšanje cjelokupne kvalitete slike. Razumijevanje područja filtriranja slika zahtjeva daljnju klasifikaciju filtera u dvije primarne domene: prostornu i frekvencijsku. Osnovna podjela tehnika filtriranja slike vidljiva je na slici 5. Filtriranje u prostornoj domeni podrazumijeva izravno djelovanje na vrijednosti piksela slika što ga čini vrlo intuitivnim te široko korištenim pristupom. Uobičajeni filteri uključuju zaglađivanje, izoštravanje i otkrivanje rubova. Filtriranje u frekvencijskoj domeni objedinjuje transformaciju slike u njezine frekvencijske komponente pomoću Fourierove transformacije. To omogućuje kompleksnije manipulacije u frekvencijskoj domeni te rezultira rješavanjem problema poput periodičkog šuma ili naglašavanja određenih frekvencijskih raspona. Kao što je vidljivo na slici 5., filtriranje u prostornoj domeni obuhvaća linearno i nelinearno filtriranje. Linearni filteri čine temeljnu kategoriju tehnika filtriranja slika, a karakterizira ih načelo superpozicije. Konvolucijski filteri predstavljaju primjer linearnosti primjenjujući ponderirani zbroj vrijednosti piksela unutar lokalnog susjedstva. Linearni se filteri koriste za zadatke poput zaglađivanja i poboljšavanja značajki. U slučaju kada su nelinearni odnosi presudni, koriste se nelinearni filteri. Oni prkose načelu superpozicije čime se dopuštaju složenije transformacije. U rastućem području umjetne inteligencije i strojnog učenja, filtriranje slike služi kao korak pretprocesiranja olakšavajući izdvajanje ključnih značajki za naknadnu analizu te donošenje odluka [2].



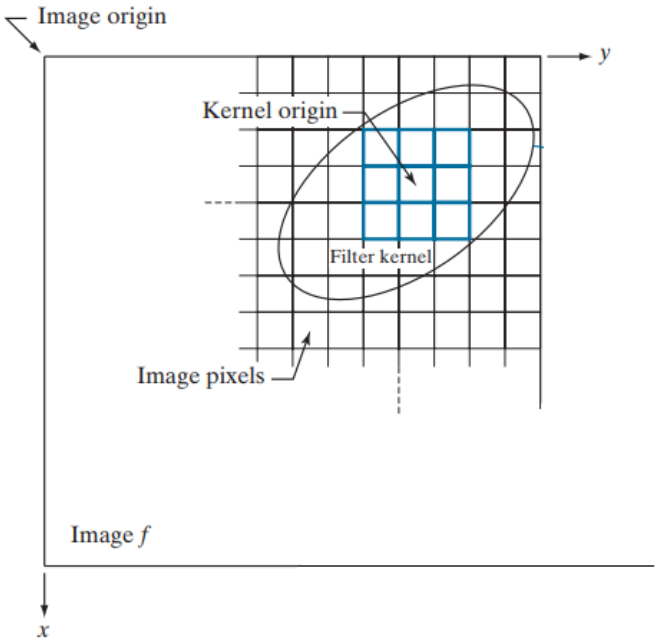
Slika 5. Osnovna podjela tehnika filtriranja [25]

3.1. Filtriranje u prostornoj domeni

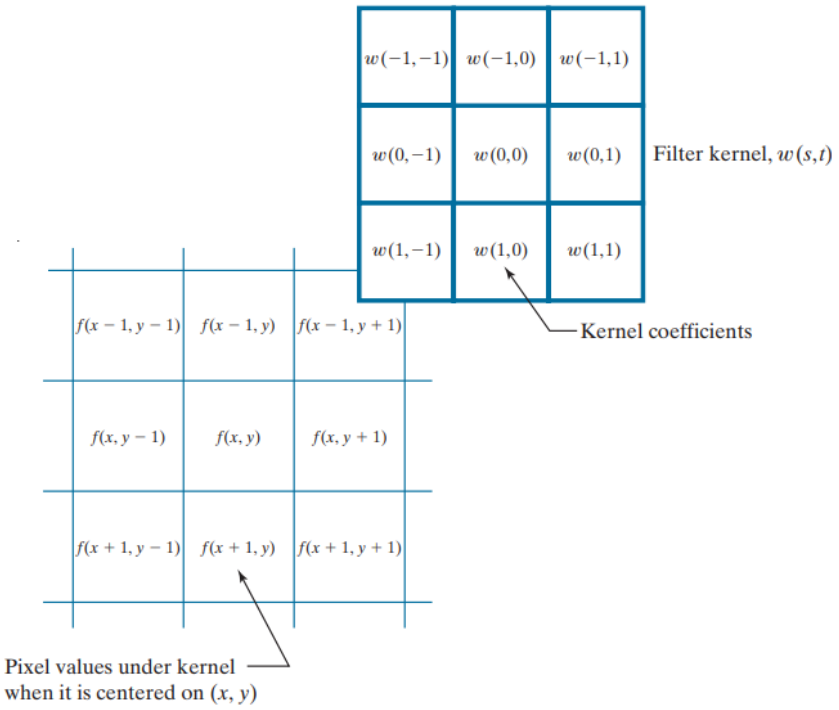
Pojam prostorne domene odnosi se na samu ravninu slike, a metode obrade slike u ovoj domeni temelje se na izravnoj manipulaciji pikselima na slici. Prostorno filtriranje obuhvaća izvođenje operacija u blizini svakog piksela na slici, odnosno zamjenu vrijednosti svakog piksela funkcijom vrijednosti piksela i njegovih susjeda. Ako je operacija izvedena na pikselima slike linerarna, tada se filter naziva linearnim prostornim filterom. U suprotnom, radi se o nelinearnom prostornom filteru [2].

3.1.1. Linearno prostorno filtriranje

Linearni prostorni filter izvodi operaciju zbroja produkata između slike, f i kernela filtera, **w**. Kernel je niz čija veličina definira područje provođenja operacije, odnosno, susjedstvo. Njegovi koeficijenti određuju prirodu filtera. Na slici 6. i slici 7. vidljiv je način rada linearnog prostornog filtriranja pomoću $3 \, x \, 3$ kernela. Pikseli su prikazani poput kvadratića radi pojednostavljenja i lakšeg razumijevanja. Treba uzeti u obzir kako je ishodište slike u gornjem lijevom kutu, a ishodište kernela u središtu. Iz navedenog je razloga potrebno postavljanje ishodišta u središte prostorno simetričnih kernela čime se pojednostavljuje pisanje izraza za linearno filtriranje.



Slika 6. Linearno prostorno filtriranje [2]



Slika 7. Koeficijenti kernela ovisno o položaju [2]

U bilo kojoj točki (x, y) slike, odgovor $g(x, y)$ filtera jest suma produkata koeficijenata kernela i piksela slike obuhvaćenih kernelom izražena jednadžbom (1):

$$g(x,y) = \mathbf{w}-1,-1)f(x-1,y-1) + \mathbf{w}(-1,0)f(-1,y) + \cdots + \mathbf{w}(1,1)f(x+1,y+1) \quad (1)$$

Kako se koordinate x i y mijenjaju, središte kernela pomiče se od piksela do piksela generirajući pritom filtriranu sliku g. Općenito, linearno prostorno filtriranje slike veličine $M\ x\ N$ s kernelom veličine $m\ x\ n$ dano je izrazom (2):

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \mathbf{w}(s,t)f(x+s,y+t) \quad (2)$$

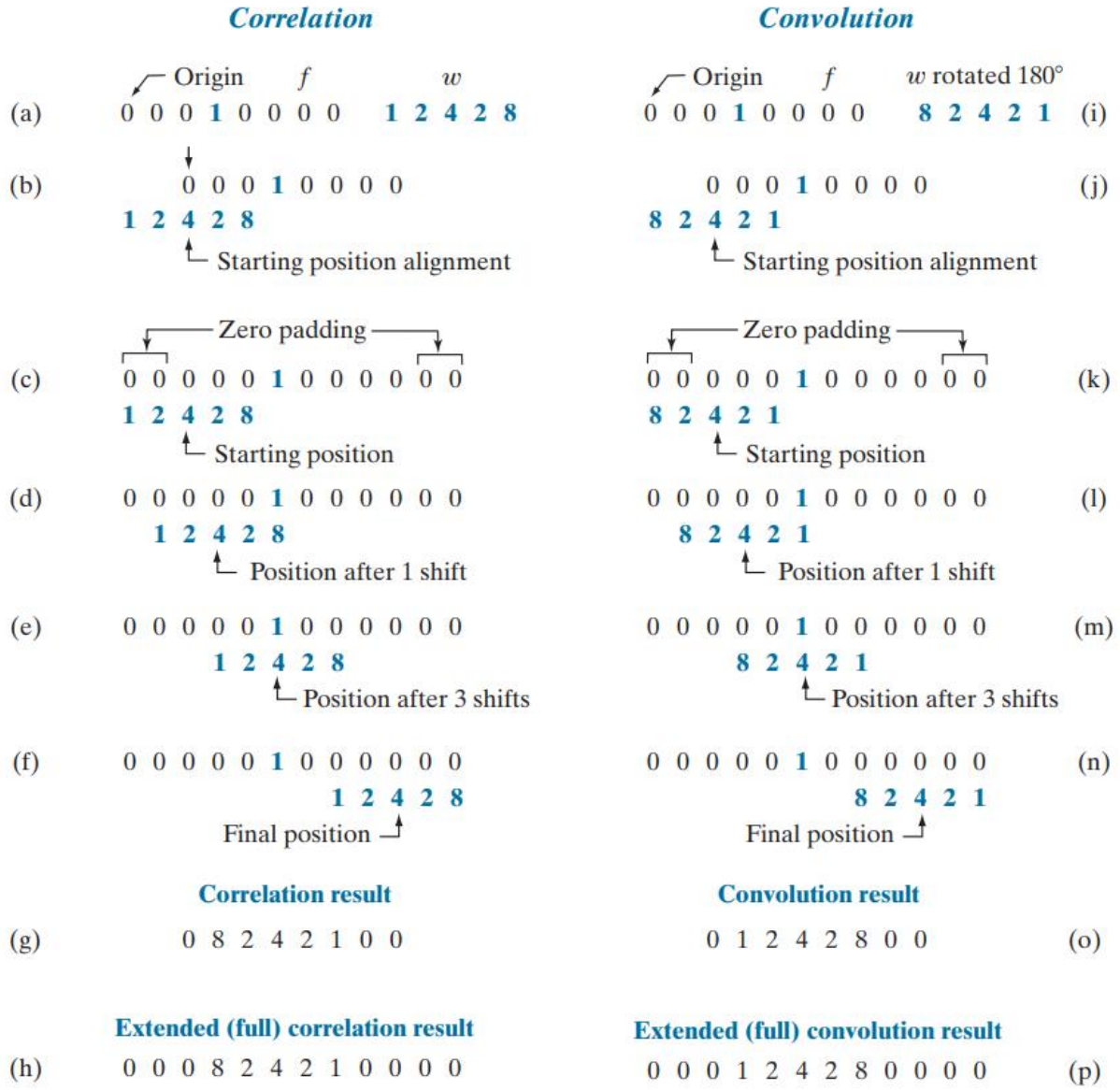
gdje su a i b pozitivni cijeli brojevi, a x i y se mijenjaju tako da ishodište kernela posjećuje svaki piksel slike samo jednom. Za fiksnu vrijednost (x, y) jednadžba (2) implementira se kao suma produkata za kernel proizvoljne neparne veličine. Spomenuta jednadžba zapravo opisuje prostornu korelaciju. Prostorna korelacija i prostorna konvolucija nalik su jedna na drugu, međutim kernel korelacije zarotiran je za 180° što znači da kada su vrijednosti kernela simetrične u odnosu na središte, korelacija i konvolucija daju isti rezultat. Kako bi se navedeno dokazalo, uzet će se 1D ilustracija čime se jednadžba (2) pojednostavljuje u izraz (3):

$$g(x) = \sum_{s=-a}^{a} \mathbf{w}(s) f(x+s) \quad (3)$$

Na slici 8. u koraku a) vidljiva je 1D funkcija, f te kernel, **w**. Za kernel veličine $1\ x\ 5$, uzima se da je $a = 2, b = 0$. U koraku b) vidljiva je početna pozicija koja se koristi za izvođenje korelacije u kojoj je **w** postavljen tako da se njegov središnji koeficijent podudara s ishodištem funkcije f. Primjećuje se kako dio kernela **w** leži izvan funkcije f pa je stoga zbrajanje u tom području nedefinirano. Taj se problem jednostavno rješava – upotpunjavanjem funkcije s odgovarajućim brojem nula sa svake strane. Općenito, ako je veličina kernela veličine $1\ x\ m$, potrebno je $(m-1)/2$ nula s obje strane funkcije kako bi se mogla odrediti početna i završna konfiguracija kernela u odnosu na funkciju. U koraku c) prikazana je pravilno postavljena funkcija. Prva vrijednost korelacije, koja je zbroj umnožaka u početnoj poziciji, izračunata kada je $x = 0$ dana je izrazom (4)

$$g(0) = \sum_{s=-2}^{2} \mathbf{w}(s) f(0+s) = 0 \quad (4)$$

Navedena se vrijednost nalazi na krajnjem lijevom mjestu rezultata korelacije u koraku g). Da bi se dobila druga vrijednost korelacije, pomiču se relativni položaji kernela i fukcije za jedno mjesto udesno, odnosno prebacuju se na idući piksel. U jednadžbu se tada uvrštava $x = 1$ te dobiveni rezultat iznosi $g(1) = 8$. Daljnjim uvrštavanjem sve većih vrijednosti, odnosno mijenjajući x, gradi se rezultat korelacije prikazan u koraku g). Dakle, potrebno je sveukupno 8 vrijednosti za potpuni pomak kernela iza funkcije tako da je središnji keoficijent u kernelu „posjeti“ svaki piksel u f. Kako bi se to ispunilo, potrebno je početi s krajnjim desnim elementom **w** koji se podudara s ishodištem f te završiti s krajnjim lijevim elementom od **w** koji se podudara s posljednjim elementom od f. U koraku h) vidljiva je proširena, odnosno, puna korelacija. Standardna korelacija dobivena je reduciranjem proširene korelacije [2].



Slika 8. Usporedba korelacije i konvolucije [2]

Zaključno, korelacija je funkcija pomaka kernela filtera u odnosu na sliku što znači da prva vrijednost korelacije odgovara nultom pomaku kernela, druga odgovara jedinici pomaka i tako dalje. Također je važno primijetiti kako korelacija kernela **w** s funkcijom koja sadrži sve nule i jednu jedinicu daje kopiju **w**, zakrenutu za 180°. Funkcija koja sadrži sve nule te jednu jedinicu naziva se diskretni jedinični impuls. Korelacija kernela s diskretnim jediničnim impulsom daje zarotiranu verziju kernela na mjestu impulsa. Desna strana slike 8. prikazuje redoslijed koraka za izvođenje konvolucije. Pri konvoluciji kernel je unaprijed zarotiran za 180°, dakle prije izvođenja operacija pomaka te zbroja produkata. Rezultat predrotiranja kernela jest točna kopija kernela na mjestu jediničnog impulsa. Upravo je konvolviranje funkcije s impulsom, čime se

dobiva kopija funkcije na mjestu impulsa, temelj teorije linearnog sustava. Diskretni impuls snage definiran je jednadžbom (5):

$$\boldsymbol{\delta}(x - x_0, y - y_0) = \begin{cases} \mathbf{A} & if\ x = x_0\ and\ y = y_0 \\ 0 & otherwise \end{cases} \tag{5}$$

Na slici 9. u srednjem redu prikazana je korelacija, a u zadnjem redu konvolucija 2D kernela sa slikom koja se sastoji od diskretnog jediničnog impulsa. Korelacija kernela **w** veličine $m\ x\ n$ sa slikom $f(x, y)$ označuje se kao $(\mathbf{w} \diamondsuit f)(x, y)$. Budući da kerneli ne ovise o (x, y) jednadžba (2) može se zapisati kao sljedeći izraz (6):

$$(\mathbf{w} \diamondsuit f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \mathbf{w}(s, t) f(x + s, y + t) \tag{6}$$

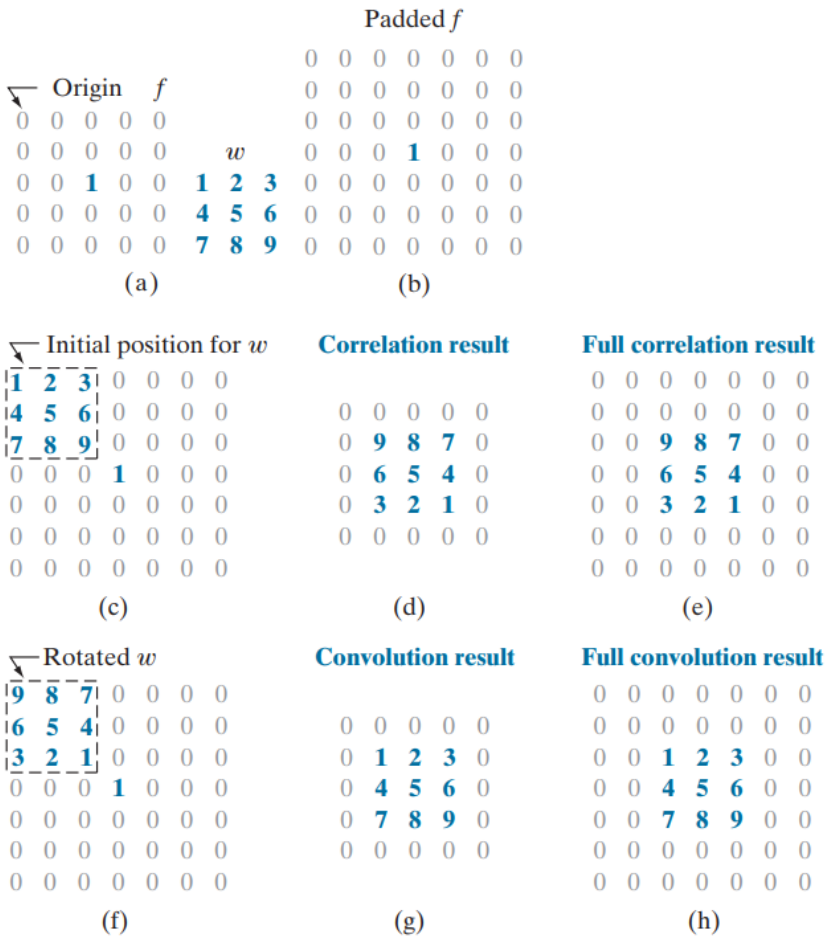
dok se konvolucija kernela može zapisati kao izraz (7):

$$(\mathbf{w} \blacklozenge f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \mathbf{w}(s, t) f(x - s, y - t) \tag{7}$$

gdje predznak minus poravnava koordinate f i **w** kada se jedna od funkcija zakrene za 180°. Navedena jednadžba implementira zbroj produkata koji se referira kao proces linearnog prostornog filtriranja, odnosno, prostorne konvolucije. Budući da je konvolucija komutativna, nije važno jesu li **w** i f zarotirane, no rotacije kernela provode se prema konvenciji. Kao i kod korelacije, jednadžba (7) procjenjuje se za sve vrijednosti varijabli pomaka x i y tako da središte kernela posjećuje svaki piksel u funkciji za koji se pretpostavlja da je ispunjen. Korelacija i konvolucija mogu se definirati na način da svaki element od **w** umjesto samo u svoje središte posjećuje svaki piksel u f što zahtijeva da početna konfiguracija bude takva da se donji desni kut kernela podudara s ishodištem slike. Slično tome, završna konfiguracija bit će s gornjim lijevim kutom kernela koji se podudara s donjim desnim kutom slike. Ukoliko su kernel i slika veličine $m\ x\ n$, odnosno, $M\ x\ N$, popuna bi se morala povećati na $(m - 1)$ element popune iznad i ispod slike te $(n - 1)$ element s lijeve i desne strane. Pod navedenim uvjetima veličina dobivenog punog korelacijskog ili konvolucijskog niza bit će veličine $S_v \times S_h$ gdje su vrijednosti S_v i S_h prikazane jednadžbama (8) i (9):

$$S_v = m + M - 1 \tag{8}$$

$$S_h = n + N - 1 \tag{9}$$



Slika 9. Korelacija i konvolucija 2D kernela [2]

Često se algoritmi prostornog filtriranja temelje na korelaciji i stoga implementiraju jednadžbu (6). Da bi se koristila jednadžba za korelaciju, koristi se **w**, a za konvoluciju koristi se **w** zarotiran za 180°. Suprotno vrijedi za algoritam koji implementira jednadžbu (7). Dakle, obje se jednadžbe mogu koristiti rotacijom kernela filtera. Valja imati na umu kako je redoslijed unesenih funkcija u korelacijski algoritam vrlo važan budući da korelacija nije komutativna ni asocijativna. Neka od temeljnih svojstava konvolucije i korelacije vidljiva su u tablici 1.

Tablica 1. Temeljna svojstva konvolucije i korelacije [2]

Svojstvo	Konvolucija	Korelacija
Komutativnost	$f \blacklozenge g = g \blacklozenge f$	–
Asocijativnost	$f \blacklozenge (g \blacklozenge h) = (f \blacklozenge g) \blacklozenge h$	–
Distributivnost	$f \blacklozenge (g + h) = (f \blacklozenge g) + (f \blacklozenge h)$	$f \diamondsuit (g + h) = (f \diamondsuit g) + (f \diamondsuit h)$

Na slici 10. prikazana su dva kernela koja se koriste za zaglađivanje intenziteta slike. Za filtriranje slike pomoću jednog od ovih kernela izvodi se konvolucija kernela sa slikom. Kada

se govori o filtriranju i kernelima, često se nailazi na izraz konvolucijski filter koji obično označava kernel prostornog filtera što ne mora nužno značiti da se kernel koristi za konvoluciju. Ponekad se slika filtrira sekvencijalno, odnosno u fazama koristeći različit kernel u svakoj od faza. Primjerice, slika *f* može se filtrirati kernelom $\mathbf{w_1}$, njezin rezultat kernelom $\mathbf{w_2}$, taj rezultat s trećim kernelom i tako dalje za Q stupnjeva. Zbog komutativnog svojstva konvolucije, višestupanjsko filtriranje može se izvesti u jednoj operaciji filtriranja ($\mathbf{w}\blacklozenge f$) gdje vrijedi prikazano u jednadžbi (10):

$$\mathbf{w} = \mathbf{w_1}\blacklozenge\mathbf{w_2}\blacklozenge\mathbf{w_3}\blacklozenge \ldots \blacklozenge\mathbf{w_Q} \tag{10}$$

Veličina $\mathbf{w}$ dobiva se iz veličina pojedinačnih kernela uzastopnom primjenom jednadžbi (8) i (9). Ako su svi pojedinačni kerneli veličine $m \times n$, iz ovih jednadžbi slijedi kako će veličina $\mathbf{w}$ biti jednaka slijedećim izrazima prikazanim jednadžbama (11) i (12) [2]:

$$W_v = Q \times (m-1) + m \tag{11}$$

$$W_h = Q \times (n-1) + n \tag{12}$$

$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{4.8976}\times\begin{bmatrix}0.3679 & 0.6065 & 0.3679\\ 0.6065 & 1.0000 & 0.6065\\ 0.3679 & 0.6065 & 0.3679\end{bmatrix}$$

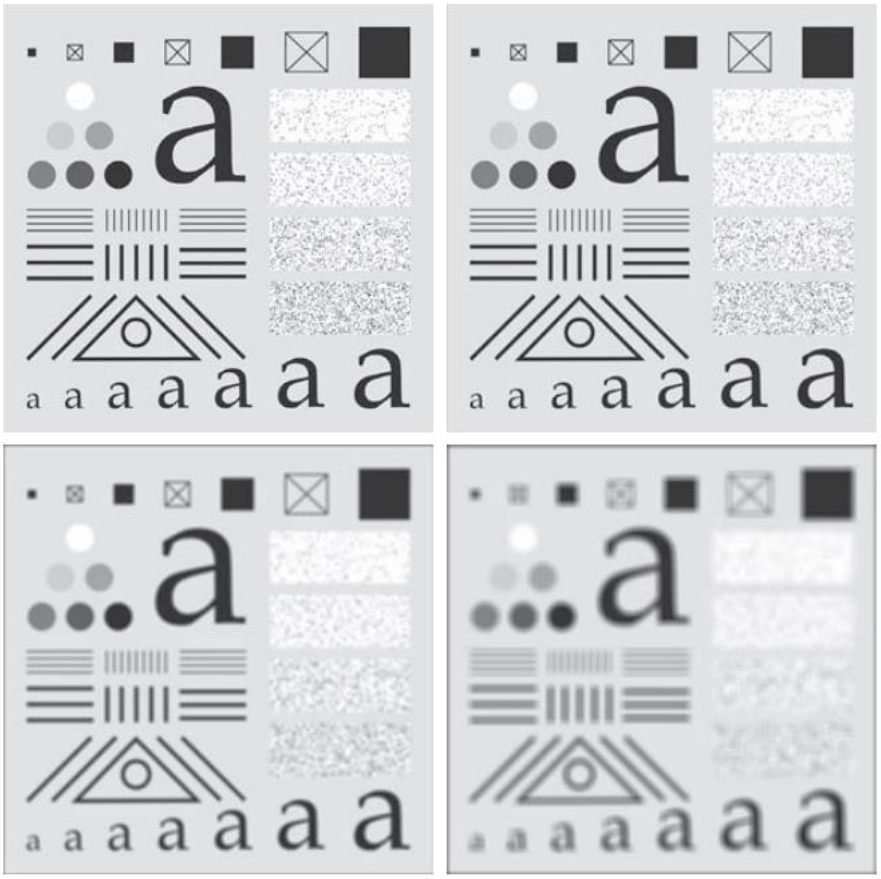
Slika 10. *Box* i Gaussian kernel [2]

3.1.1.1. Filteri zaglađivanja – niskopropusni prostorni filteri

Prostorni filteri zaglađivanja koriste se za smanjenje oštrih prijelaza intenziteta. Budući da se nasumični šum najčešće sastoji upravo od oštrih prijelaza, primjena filtera zaglađivanja jest smanjenje šuma. Zaglađivanje se koristi za smanjenje irelevantnih detalja na slici što se odnosi na područja piksela koja su mala u odnosu na veličinu kernela filtera. Druga primjena filtera jest zaglađivanje lažnih kontura koje nastaju korištenjem nedovoljnog broja razina intenziteta na slici. Filteri za zaglađivanje se koriste u kombinaciji s drugim tehnikama za poboljšanje slike poput histograma. Kao što je prethodno spomenuto, linearno prostorno filtriranje sastoji se od procesa konvolucije slike pomoću kernela filtera. Prilikom procesa konvolucije kernela koji zaglađuje sliku dolazi do zamućenja slike pri čemu se stupanj zamućenja određuje veličinom kernela i vrijednostima njegovih koeficijenata. Niskopropusni filteri temelj su u mnogim primjenama obrade te ključni za izvođenje drugih važnih filtera, primjerice visokopropusnih te

pojasno propusnih filtera. U ovome će se dijelu govoriti o niskopropusnim filterima koji se temelje na Gaussian te *box* filteru.

Najjednostavniji kernel niskopropusnih filtera koji valja spomenuti jest *box* filter čiji koeficijenti imaju istu vrijednost (najčešće 1). *Box* filter prethodno je prikazan na slici 10. s lijeve strane. Vidljivo je kako je to filter veličine $m \times n$ što znači da je to niz veličine $m \times n$ popunjen jedinicama, s konstantom normalizacije ispred njega čija je vrijednost 1 podijeljena sa zbrojem koeficijenata. Navedeno ima dvije svrhe. Prvo jest da bi prosječna vrijednost područja konstantnog intenziteta bila jednaka onom intenzitetu na filtriranoj slici. Drugo jest da normalizacija kernela na taj način sprječava uvođenje *biasa* tijekom filtriranja, odnosno, zbroj piksela u originalnoj i filtriranoj slici bit će jednaki. Budući da su u *box* kernelu svi redci i stupci jednaki, rang kernela je 1. Na slici 11. vidljiv je primjer primjene *box* filtera. Prva u nizu slika jest testni uzorak veličine 1024×1024 piksela. Preostale slike dobivene su primjenom *box* filtera na testnu sliku veličine $m \times m$, s time da veličina m poprima vrijednosti 3, 11 te 21. Za $m = 3$ (druga slika po redu) primjećuje se blago sveukupno zamućenje slike pri čemu je utjecaj na značajke slike čije su veličine usporedive s veličinom kernela veći. Te značajke uključuju tanje linije na slici i piksele šuma koji se nalaze u okvirima s desne strane slike. Filtrirana slika također ima i tanak sivi rub što je rezultat *zero-padding-a. Padding*, odnosno popuna ima ulogu u proširenju granica slike kako bi se izbjegle nedefinirane operacije kada dijelovi kernela leže izvan granice slike tijekom filtriranja. Korištenje popune prije početka filtriranja rezultira stvaranjem tamno sivog ruba na rubovima slike koji tada uključuje crne piksele u proces izračunavanja prosjeka. Korištenje kernela 11×11 (treća slika) rezultira izraženijim zamućenjem na cijeloj slici uključujući istaknutiji tamni rub. Rezultat korištenja kernela 21×21 (zadnja slika) pokazuje značajno zamućenje svih komponenti slike uključujući gubitak karakterističnog oblika pojedinih komponenti [2].



Slika 11. Utjecaj box filtera na sliku [2]

Zbog svoje jednostavnosti *box* filteri prikladni su za zaglađivanje slike budući da daju vizualno prihvatljive rezultate. Međutim, zbog svojih ograničenja često jesu loš izbor u mnogim primjenama. Primjerice, defokusirana leća često se modelira kao niskopropusni filter, no *box* filter daje lošu aproksimaciju zamućenja leće. Još jedno od ograničenja jest da *box* filter daje prednost zamućivanju duž okomitih smjerova. U primjenama koje uključuju slike s visokom razinom detalja usmjerenost ovakvih filtera na okomite smjerove često rezultira neželjenim rezultatima. Kerneli koji su u prethodno navedenim primjerima poželjniji jesu kružno simetrični odnosno izotropni, što znači da je njihovo djelovanje neovisno o orijentaciji. Pokazalo se kako su Gaussovi kerneli oblika prikazanog jednadžbom (13) jedini kružno simetrični kerneli koju su također i separabilni:

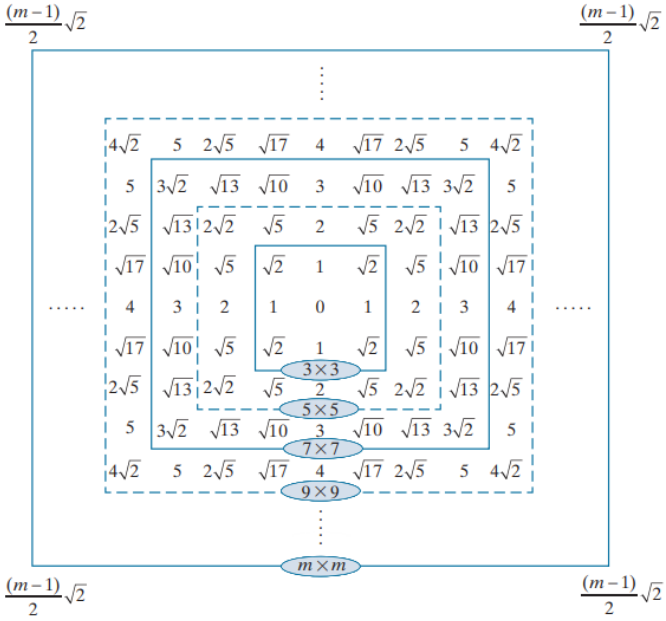
$$w(s,t) = G(s,t) = Ke^{\frac{-s^2+t^2}{2\sigma^2}} \tag{13}$$

Separabilni kerneli omogućuju filtriranje u više koraka. Prema tome, Gaussovi kerneli omogućuju jednake računalne prednosti poput *box* filtera, no imaju niz dodatnih svojstava koja

ih čine pogodnijima za obradu slike. Varijable s i t u jednadžbi jesu realni brojevi. Ako se u jednadžbu (13) zapiše da je $r = [s^2 + t^2]^{1/2}$ dobiva se jednadžba (14):

$$G(r) = Ke^{\frac{-r^2}{2\sigma^2}} \tag{14}$$

Varijabla r predstavlja udaljenost od središta do bilo koje točke na funkciji G. Na slici 12. vidljive su vrijednosti r za nekoliko veličina kernela koristeći cjelobrojne vrijednosti za s i t. Budući da se većinom radi s neparnim veličinama kernela, središta takvih kernela jesu cjelobrojne vrijednosti, a iz toga slijedi da su sve vrijednosti r^2 također cijeli brojevi.



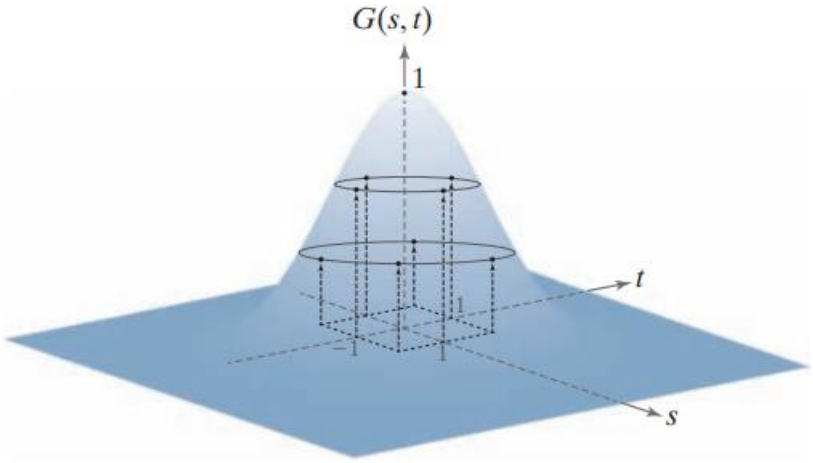
Slika 12. Vrijednosti veličine r u ovisnosti o udaljenosti [2]

Slika 13. prikazuje perspektivni dijagram Gaussove funkcije te ilustrira da su uzorci korišteni za generiranje tog kernela dobiveni određivanjem vrijednosti s i t, a potom očitavanjem vrijednosti funkcije na tim koordinatama. Navedene su vrijednosti zapravo koeficijenti kernela. Normaliziranje kernela dijeljenjem njegovih koeficijenata sa zbrojem koeficijenata dovršava se specifikacija kernela. Razlozi za normalizaciju kernela povezani su *box* filterom. Budući da su Gaussovi kerneli separabilni, moguće je uzeti uzorke duž presjeka kroz središte te na temelju njih formirati vektor v čime se dobiva 2D kernel. Navedeno je prikazano u jednadžbi (15):

$$\mathbf{w} = \mathbf{v}\mathbf{v}^{\mathbf{T}} \tag{15}$$

Separabilnost je jedno od temeljnih svojstava kružno simetričnih Gaussovih kernela. Primjer toga jest da su vrijednosti Gaussove funkcije na udaljenosti od 3σ dovoljno male da se mogu zanemariti. To znači da ukoliko je veličina kernela $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$ (oznaka $\lceil c \rceil$ koristi se za označavanje gornje granice c, odnosno najmanji cijeli broj koji nije manji od c), osigurano je

dobivanje rezultata kao da se koristila proizvoljno velik Gaussov kernel. Drugim riječima, ne dobiva se ništa korištenjem većeg kernela od navedenog. Kako se obično radi s kernelima neparnih dimenzija, koristio bi se najmanji neparan broj koji zadovoljava ovaj uvjet – za $\sigma = 7$ kernel je veličine 43×43. Druga dva temeljna svojstva Gaussovih funkcija jesu da produkt i konvolucija dviju Gaussovih funkcija također daju Gaussovu funkciju. Tablica 2. prikazuje srednju vrijednost i standardnu devijaciju umnoška i konvolucije dviju 1D Gaussovih funkcija, f i g. Rezultat konvolucije je od velike važnosti u filtriranju slika. Kao što je prethodno spomenuto, filtriranje se može izvoditi u više uzastopnih faza te se rezultat može dobiti jednom fazom filtriranja sa složenim kernelom kao i konvolucijom pojedinačnih kernela [2].



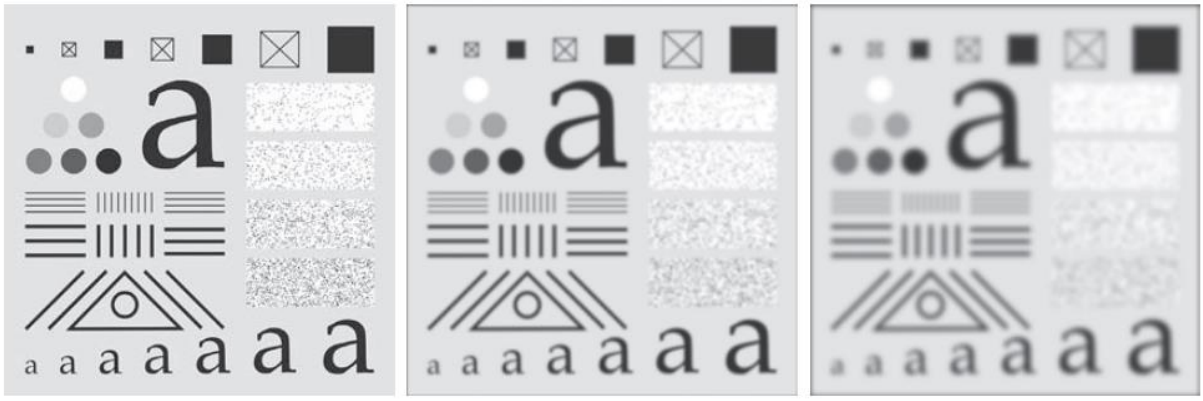
Slika 13. Prikaz Gaussove funkcije [2]

Tablica 2. Srednja vrijednost i standardna devijacija produkta i konvolucije dviju 1D Gaussovih funkcija [2]

	f	g	$f \times g$	$f \blacklozenge g$
Srednja vrijednost	m_f	m_g	$m_{f\times g} = \frac{m_f {\sigma_g}^2 + m_g {\sigma_f}^2}{{\sigma_f}^2 + {\sigma_g}^2}$	$m_{f \blacklozenge g} = m_f + m_g$
Standardna devijacija	σ_f	σ_g	$\sigma_{f\times g} = \sqrt{\frac{{\sigma_f}^2 {\sigma_g}^2}{{\sigma_f}^2 + {\sigma_g}^2}}$	$\sigma_{f \blacklozenge g} = \sqrt{{\sigma_f}^2 + {\sigma_g}^2}$

Kako bi se bolje shvatila razlika između Gaussian i *box* filtera, u prikazu Gaussian filtera koristit će se jednak primjer kao i prije. Gaussovi kerneli moraju biti veći od *box* filtera da bi se postigao isti stupanj zamućenja. Razlog tome je što kernel *box* filtera svim pikselima dodjeljuje jednaku

težinu, a vrijednosti Gaussovih koeficijenata, a time i njihov učinak, padaju kao funkcija udaljenosti od središta kernela. Kao što je prije objašnjeno, koristit će se veličina jednaka najbližem neparnom cijelom broju od $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$. Dakle, za Gaussov kernel veličine 21×21, što je veličina kernela korištena u primjeru s *box* filterom na slici 11., potrebno je uzeti vrijednost $\sigma = 3{,}5$. Na drugom primjeru slike 14. vidljiv je utjecaj navedenog niskopropusnog filtera s ovim kernelom. Uspoređujući to sa zadnjim primjerom na slici 11., može se zaključiti kako je Gaussov kernel rezultirao znatno manjim zamućenjem. Kako bi se dobio otprilike jednak stupanj zamućenja (zadnja slika u nizu) potrebno je koristiti Gaussov kernel veličine 43×43 što znači da bi tada vrijednost standardne devijacije iznosila $\sigma = 7$ [2].

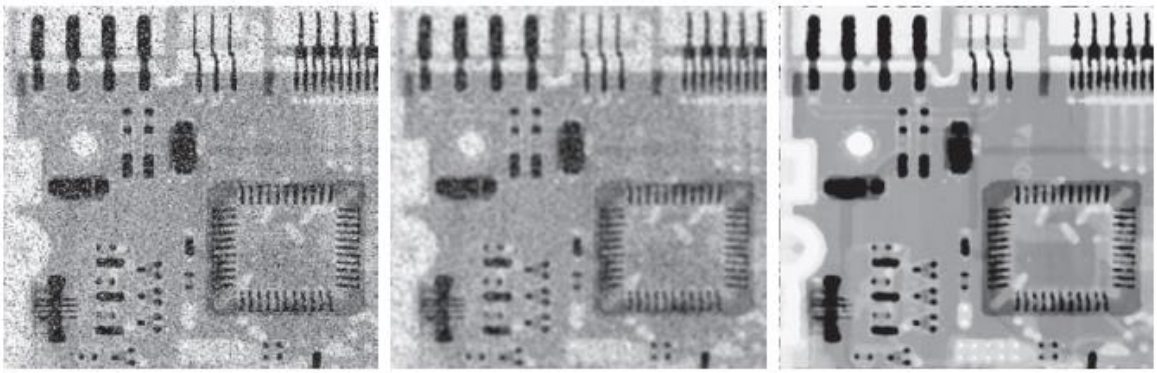


Slika 14. Utjecaj Gaussian filtera na sliku [2]

3.1.2. Nelinearno prostorno filtriranje

Svi prethodno navedeni filteri bili su linearni. To znači da je njihov odgovor na zbroj dvaju signala jednak zbroju pojedinačnih odgovora. Navedeno je ekvivalentno tvrdnji kako je svaki izlazni piksel ponderirani zbroj nekog broja ulaznih piksela. Linearne filtere lakše je sastaviti i podložni su analizi frekvencijskog odziva. Međutim, u mnogim slučajevima bolje performanse mogu se postići korištenjem nelinearne kombinacije susjednih piksela [4]. Nelinearni filteri temelje se na rangiranju piksela koji se nalaze u području obuhvaćenom filterom. Zaglađivanje se može postići zamjenom vrijednosti središnjeg piksela s vrijednošću određenom rezultatom rangiranja. Najpoznatiji filter u ovoj kategoriji jest median filter. On zamjenjuje vrijednost središnjeg piksela medijanom vrijednosti intenziteta u blizini tog piksela pri čemu je vrijednost središnjeg piksela uključena u izračun medijana. Median filteri omogućuju smanjenje buke za određene vrste nasumičnog šuma sa znatno manjim zamućenjem od linearnih filtera za zaglađivanje slične veličine. Median filter je posebice učinkovit kod prisutnosti impulsnog šuma koji se naziva i šumom soli i papra budući da se očituje kao crne i bijele točkice na slici.

Medijan, ξ, skupa vrijednosti je takav da je polovica vrijednosti u skupu manja ili jednaka ξ, a polovica veća ili jednaka ξ. Kako bi se izvršilo median filtriranje prvo je potrebno sortirati vrijednosti piksela u susjedstvu, odrediti njihov medijan i tu vrijednost dodijeliti pikselu na filtriranoj slici koji odgovara središtu susjedstva. Slika 15. prikazuje rendgensku sliku tiskane ploče oštećene šumom soli i papra [2]. Kako bi se dokazala superiornost median filtera u odnosu na niskopropusno filtriranje, na drugoj slici u nizu prikazan je rezultat filtriranje slike Gaussovim filterom, a na zadnjoj slici median filterom. Gaussov filter zamutio je sliku, a učinak smanjenja šuma je bio loš. Iz rezultata filtriranja median filterom jasno je kako je njegova primjena u ovakvim slučajevima neusporedivo bolja.



Slika 15. Usporedba Gaussovog i median filtera [2]

3.1.2.1. Filteri izoštravanja – visokopropusni prostorni filteri

Izoštravanje naglašava prijelaze u intenzitetu. Upotreba izoštravanja slike kreće se od elektroničkog ispisa i medicinskog snimanja do industrijske inspekcije i autonomnog navođenja u vojnim sustavima. Filteri za izoštravanje temelje se na izvedenicama prve i druge derivacije. Valja napomenuti neke od temeljnih svojstava derivacija pa će se prema tome prvo usmjeriti pozornost na jednodimenzionalne derivacije. Zanima nas ponašanje ovih derivacija u područjima konstantnog intenziteta, na početku i kraju diskontinuiteta (diskontinuiteti koraka i rampe). Derivacije digitalne funkcije definirane su u smislu razlike. Za prvu derivaciju vrijedi:

- Mora biti jednaka nuli u područjima konstantnog intenziteta
- Mora biti različita od nule na početku koraka ili „rampe“ intenziteta
- Mora biti različita od nule duž „rampe“ intenziteta

Slično, za drugu derivaciju vrijedi:

- Mora biti jednaka nuli u područjima konstantnog intenziteta

- Mora biti različita od nule na početku i na kraju koraka ili „rampe“ intenziteta
- Mora biti nula uz „rampe“ intenziteta.

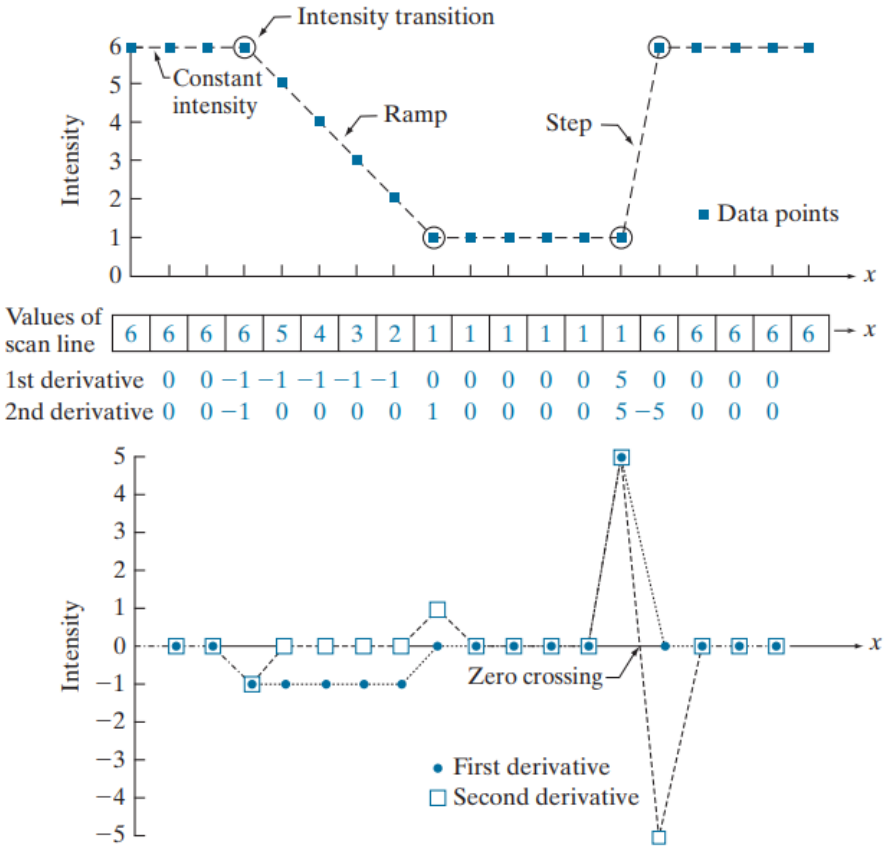
Budući da se radi o digitalnim veličinama čije su vrijednosti konačne, i najveća moguća promjena intenziteta također je konačna, a najkraća udaljenost na kojoj se ta promjena može dogoditi je između susjednih piksela. Osnovna definicija derivacije prvog reda jednodimenzionalne funkcije $f(x)$ prikazana je jednadžbom (16):

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{16}$$

U jednadžbi se upotrijebila parcijalna derivacija kako bi notacija bila dosljedna kada se razmatra slika funkcije dviju varijabli $f(x, y)$, no tada se pišu parcijalne derivacije duž dviju prostornih osi. Jasno je kako vrijedi $\frac{\partial f}{\partial x} = \frac{df}{dx}$ kada postoji samo jedna varijabla u funkciji. Isto vrijedi i za drugu derivaciju. Derivacija drugog reda $f(x)$ definirana je jednadžbom (17):

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - f(x-1) - 2f(x) \tag{17}$$

Vrijednosti označene kvadratićima na slici 16. (a) prikazuju vrijednosti intenziteta duž horizontalnog profila intenziteta. Isprekidana linija koja povezuje kvadratiće uključena je radi lakše vizualizacije. Na slici je vidljivo kako linija skeniranja sadrži tri dijela konstantnog intenziteta, rampu intenziteta te korak intenziteta. Krugovi označavaju početak ili kraj prijelaza intenziteta. Derivacije prvog i drugog reda izračunate pomoću jednadžbi (16) i (17) prikazane su ispod vrijednosti linije skeniranja te ucrtane slici 16. (b). Kada se računa prva derivacija na lokaciji x, oduzima se vrijednost funkcije na toj lokaciji od sljedeće točke kao što je naznačeno u jednadžbi (16) pa se to naziva operacija pogleda unaprijed. Slično tome, pri izračunu druge derivacije u x, koristi se prethodna i sljedeća točka kao što je vidljivo u jednadžbi (17). Na slici 16. (a) prolaskom kroz profil slijeva na desno, prvo se nailazi na područje konstantnog intenziteta. Slike 16. (b) i (c) pokazuju kako su na tom području obje derivacije jednake nuli čime je zadovoljen prvi uvjet. Zatim dolazi rampa intenziteta nakon koje slijedi korak te je vidljivo kako je derivacija prvog reda različita od nule na početku rampe i koraka. Također, druga derivacija različita je od nule na početku i na kraju i rampe i koraka čime je zadovoljen drugi uvjet. Na kraju, treći uvjet je također zadovoljen budući da je prva derivacije različita od nule, a druga je nula duž rampe. Valja imati na umu kako se predznak druge derivacije mijenja na početku i na kraju koraka ili rampe.



Slika 16. Prikaz intenziteta duž horizontalnog profila slike [2]

Rubovi u digitalnim slikama često su prijelazi intenziteta poput rampe u kojem bi slučaju prva derivacija slike rezultirala debelim rubovima jer je derivacija različita od nule duž rampe. S druge strane, druga derivacija napravila bi dvostruki rub debljine jednog piksela. Iz navedenoga se može zaključiti kako druga derivacija poboljšava detalje puno bolje od prve derivacije što je idealno svojstvo za izoštravanje slike. Također, druge derivacije zahtijevaju manje operacija za implementaciju nego prve tako da su pogodnije za korištenje [2].

3.2. Filtriranje u frekvencijskoj domeni

Filtriranje u frekvencijskoj domeni predstavlja moćan pristup obradi signala koji se koristi u brojnim područjima. Ključna ideja iza ovog pristupa leži u analizi signala u frekvencijskom području pomoću Fourierove transformacije. Francuski matematičar Jean Baptiste Joseph Fourier bavio se problemom rješavanja parcijalne diferencijalne jednadžbe za širenje topline. Doprinos po kojem je najviše zapamćen opisan je u memoarima, a kasnije objavljen 1822. u njegovoj knjizi „*O širenju topline u čvrstim tijelima*“. Njegov je rad naišao na otpor

matematičara jer je Fourier tvrdio kako se sve funkcije mogu razviti u trigonometrijski red. Prema tome, Fourierova analiza proizlazi iz ideje kako se svaka periodička funkcija može zapisati kao suma (ne nužno konačna) sinusa različitih amplituda, faza i frekvencija. Takva suma naziva se Fourierov red. Fourierove teorije razvijaju se i danas. Među njima jesu značajnije diskretna Fourierova transformacija (DFT) te brza Fourierova transformacija (FFT) koja je temeljna u multimedijskoj primjeni [6].

3.2.1. Fourierov red

Kako bi se funkcija mogla razviti u Fourierov red, potrebno je da je ona periodična. Neperiodične funkcije također se mogu razviti, no uz uvjet da ih se prvo učini periodičnima. U tome slučaju taj interval postaje period funkcije te se ponavlja beskonačno mnogo puta. Razmatranja koja će se iznijeti u daljnjoj raspravi vrijede za periodične funkcije općenitog perioda $\left[-\frac{L}{2}, \frac{L}{2}\right]$, no zbog jednostavnosti će se pretpostaviti kako je funkcija periodična na intervalu $[-\pi, \pi]$. Za funkciju $f: R \to R$ periodičnu na $[-\pi, \pi]$ žele se pronaći brojevi $a_0, a_k, \emptyset_k \in R$, za $k = 1, \dots, n$ tako da vrijedi jednadžba (18) [6]:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{n} a_k \, sin(kx + \emptyset_k), \qquad n \geq 0 \tag{18}$$

U jednadžbi a_k predstavlja amplitudu, a $\emptyset_k$ fazu za sinus funkciju frekvencije k. Član $\frac{a_0}{2}$ služi za translaciju funkcije duž y-osi. Ako se u navedenoj jednadžbi primijeni adicijska formula za sinus funkciju izražena jednadžbom (19):

$$sin(\alpha + \beta) = sin\,\alpha \, cos\,\beta + cos\,\alpha \, sin\,\beta \tag{19}$$

Fourierov red može se zapisati i poput jednadžbe (20):

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{n} (a_k \, sin(kx) \, cos\,\emptyset_k + (a_k \, cos(kx) \, sin\,\emptyset_k) \tag{20}$$

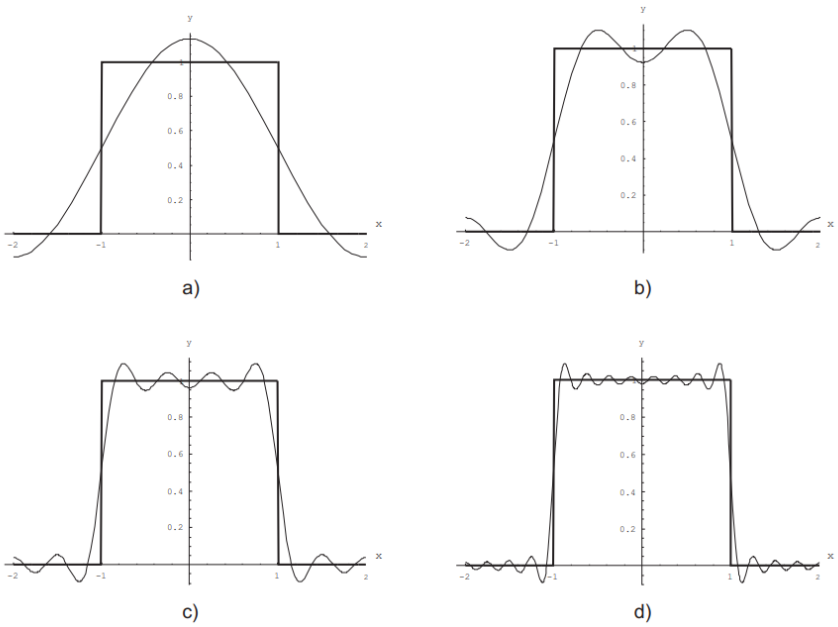
Vidljivo je kako veličine $sin\,\emptyset_k$ i $cos\,\emptyset_k$ ne ovise ni o varijabli x ni o k. Oni su koeficijenti uz $sin(kx)$ i $cos(kx)$ pa se mogu definirati novi koeficijenti $a_k, b_k \in R$ u kojima će implicitno biti uključen i $\emptyset_k$. Nakon uvedenih novih koeficijenata jednadžba prima novi oblik što je prikazano u izrazu (21):

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{n} (a_k \, sin(kx) + b_k \, cos(kx)), \quad n \geq 0 \tag{21}$$

Fourierov red se sada svodi na traženje koeficijenata a_k i b_k te više nema potrebe za eksplicitnim promatranjem faze pojedine sinus funkcije već je dovoljno pronaći koeficijente uz sinus i kosinus. Pitanje koje se sljedeće postavlja jest može li se uistinu svaku periodičku funkciju zapisati kao konačnu sumu sinusa i kosinusa različitih amplituda i frekvencija. Valja se prisjetiti kako je zbroj dviju neprekidnih funkcija također neprekidna funkcija te da je zbroj dviju m-puta derivabilnih funkcija opet m-puta derivabilna funkcija. Kako su i sinus i kosinus i neprekidne i m-puta derivabilne funkcije vrijedi da je i njihova konačna suma takva. Prema svemu navedenom slijedi zaključak kako je nemoguće razviti nederivabilnu funkciju u Fourierov red. No, kada se uzme u obzir kako veće frekvencije sinusa i kosinusa u sumi uzrokuju oštrije rubove, situacija se mijenja. Primjer toga dan je na slici 17. gdje je prikazana tzv. rectangle funkcija koja je definirana izrazom (22):

$$f(x) = \begin{cases} 1, & x \in (-1,1) \\ 0, & inače \end{cases} \tag{22}$$

Prikazana je aproksimacija rectangle funkcije na intervalu $[-2,2]$ uz pomoć Fourierova reda za a) $n = 2$, b) $n = 4$, c) $n = 8$, d) $n = 16$. Dakle, kada bi se dodavalo sve više funkcija, veličina k poprimala bi sve veću vrijednost pa bi i rezultantna krivulja sve bolje aproksimirala vrhove u kojima funkcija nije derivabilna.



Slika 17. Aproksimacija rectangle funkcije [6]

Navedeni argument iskoristit će se kako bi se jednadžba (21) pretvorila da ide u beskonačnost čime se dobiva izraz (23):

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \, sin(kx) + b_k \, cos(kx)) \tag{23}$$

Ova se suma može i zapisati pomoću Eulerove formule koja je dana izrazom (24):

$$re^{i\emptyset} = r(cos\,\emptyset + i\,sin\,\emptyset) \tag{24}$$

Jednadžbu (23) može se zapisati na nov način tako da se uvedu novi koeficijenti $c_k \in C, k \in Z$. Želimo li biti u potpunosti sigurno da će krajnji rezultat sadržavat realne koeficijente, koristi se svojstvo da je $z + \bar{z} = 2Re(z)$ gdje je $z \in C$, a veličina $\bar{z}$ označava kompleksno konjugirani broj. Kako bi se pomoću Eulerove funkcije dobio kompleksno konjugirani broj dovoljno je iskoristiti svojstvo neparnosti funkcije sinus, odnosno umjesto $\emptyset$ uzeti -$\emptyset$. Prema tome, cilj je pronaći koeficijente c_k tako da vrijedi jednadžba (25):

$$c_k e^{ikx} + \overline{c_k e^{ikx}} = a_k \, sin(kx) + b_k \, cos(kx) \tag{25}$$

Što je ekvivalentno izrazu (26):

$$\begin{aligned}(cos(kx) - i\,sin(kx))\bar{c_k} + (cos(kx) + i\,sin(kx))\,c_k \\ = a_k \, sin(kx) + b_k \, cos(kx)\end{aligned} \tag{26}$$

Nadalje vrijedi izraz (27):

$$cos(kx)(\bar{c_k} + c_k) + sin(kx)(c_k i - \bar{c_k} i) = a_k \, sin(kx) + b_k \, cos(kx) \tag{27}$$

Ako se uzme u obzir činjenica kako je $c_k i - \bar{c_k} i = -2Im(c_k)$ dobivaju se jednadžbe (28) i (29):

$$a_k = -2Im(c_k) \tag{28}$$

$$b_k = 2Re(c_k) \tag{29}$$

Pomnoži li se prva jednadžba (jednadžba (28)) s imaginarnom jedinicom, $a_{ki} = -2Im(c_k)i$ te joj se pribroji druga (jednadžba (29)) dobivaju se sljedeći izrazi (30) i (31):

$$b_k + a_k = 2Re(c_k) - 2Im(c_k)i \tag{30}$$

$$c_k = \frac{1}{2}(b_k + ia_k) \tag{31}$$

Valja primijetiti kako se koristila jednakost $Re(c_k) - Im(c_k)i = \bar{c_k}$. Uvede li se oznaka $c_{-k} = \bar{c_k}$, općenit razvoj u Fourierov red može se zapisati jednadžbom (32):

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} \tag{32}$$

Gdje je $c_{-k} = \frac{1}{2}(b_k + ia_k)$, $c_0 = \frac{1}{2}(b_k - ia_k)$. Nulti član je tada $c_0 = \bar{c_0} = \frac{a_0}{2}$

3.2.1.1. Određivanje Fourierovih koeficijenata c_k

Za periodičku funkciju $f(x)$ treba izračunati k-ti koeficijent c_k. Kreće se od jednadžbe (33) [6]:

$$f(x) = \cdots + c_{k-1}e^{(k-1)xi} + c_k e^{kxi} + c_{k+1}e^{(k+1)xi} + \cdots \tag{33}$$

Kada se navedena jednadžba podijeli s e^{kxi} za kojeg vrijedi da je uvijek različit od nule dobiva se jednadžba (34):

$$\begin{aligned} c_k &= \frac{f(x)}{e^{kxi}} - \left(\ldots + c_{k-1}e^{-xi} + c_{k+1}e^{xi}\right) + \cdots \\ &= f(x)e^{-kxi} - \sum_{j=-\infty, j\neq k}^{\infty} c_j e^{(j-k)xi} \end{aligned} \tag{34}$$

Primjenjujući činjenicu kako je c_k konstanta, cijeli se izraz može integrirati s obje strane po varijabli x na intervalu $[-\pi, \pi]$ dobivaju se sljedeći izrazi (35), (36), (37):

$$\int_{-\pi}^{\pi} c_k dx = \int_{-\pi}^{\pi} f(x)e^{-kxi}dx - \int_{-\pi}^{\pi} \sum_{j=-\infty, j\neq k}^{\infty} c_j e^{(j-k)xi}\, dx \tag{35}$$

$$c_k\pi - (-c_k\pi) = \int_{-\pi}^{\pi} f(x)e^{-kxi}dx - \int_{-\pi}^{\pi} \sum_{j=-\infty, j\neq k}^{\infty} c_j e^{(j-k)xi}\, dx \tag{36}$$

$$2\pi c_k = \int_{-\pi}^{\pi} f(x)e^{-kxi}dx - \sum_{j=-\infty, j\neq k}^{\infty} \frac{1}{(j-k)i} c_j (e^{(j-k)\pi i} - e^{-(j-k)\pi i}) \tag{37}$$

Ovi naizgled vrlo kompliciran izrazi mogu se pojednostaviti korištenjem Eulerove formule prema kojoj je $e^{(j-k)\pi i} - e^{-(j-k)\pi i} = i\sin(j-k)\pi = 0$. Cijela se suma poništava, a dijeleći jednadžbu s 2π dobiva se k-ti koeficijent Fourierova reda prikazan izrazom (38):

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-kxi}dx \tag{38}$$

Također, ova se jednadžba može napisati i da vrijedi za općenit interval, odnosno period pa je tada Fourierov red izražen preko jednadžbe (39):

$$f(x) = \sum_{k=-\infty}^{\infty} c_k\, e^{(2\pi ikx)/L} \tag{39}$$

Pri čemu su koeficijenti izraženi jednadžbom (40):

$$c_k = \frac{1}{L}\int_{-L/2}^{L/2} f(x)e^{-(2\pi kxi)/L}dx \tag{40}$$

3.2.1.2. Konvergencija Fourierovog reda

Pitanje konvergencije Fourierovog reda oduvijek je predstavljalo problem budući da su sinus i kosinus funkcije alternirajuće prirode. Najvažniji uvjet jest integrabilnost funkcije koja se želi razviti u Fourierov red. Po pitanju konvergencije navest će se teorem koji govori o konvergenciji po L^2 normi, odnosno Riesz-Fischer teorem po kojem za funkciju $f \in L^2([-\frac{P}{2}, P/2])$ s periodom P vrijedi jednadžba (41) [6]:

$$\int_{-P/2}^{P/2} |f(x)|^2 dx < \infty \tag{41}$$

Ako i samo ako je Fourierov red te funkcije L^2-konvergentan, odnosno ako vrijedi izraz (42):

$$\int_{-P/2}^{P/2} \left| f(x) - \sum_{k=-n}^{n} c_k e^{(2\pi i k x)/P} \right|^2 dx \to 0 \tag{42}$$

kada $n \to \infty$

Dakle, kada se usporedi kvadrat razlike funkcije $f(x)$ i njene konačne aproksimacije unutar određenog intervala, taj kvadrat teži prema nuli. Slični rezultati o konvergenciji Fourierovog reda mogu se dobiti i primjenom Lebesgueovog integrala. U kontekstu analize signala, L^2 integral funkcije ili signala predstavlja energiju tog signala. Ukoliko je taj integral konačan, kaže se kako signal posjeduje konačnu energiju [6].

3.2.2. Fourierova transformacija

U prethodnom se poglavlju promatralo kako razviti periodičnu funkciju perioda L definiranu na intervalu $\left[-\frac{L}{2}, \frac{L}{2}\right]$ u Fourierov red. Ukoliko se postavi da veličina L teži u beskonačnost, tada je funkcija periodična na intervalu $(-\infty, \infty)$ s beskonačno velikim periodom. No, uz tako postavljene uvjete jednadžba (40) težila bi u nulu. Zaključuje se kako takav pristup u kojemu se želi simulirati neperiodičnost s beskonačno dugim periodom nema smisla, no takvo je razmišljanje prvi korak prema transformaciji izraza (40) za k-ti Fourierov koeficijent prema formuli za Fourierovu transformaciju. Prvi korak jest da se jednadžba (40) zapiše u obliku funkcije F ovisne o varijabli k/L pa se dobije izraz (43):

$$F\left(\frac{k}{L}\right) = \int_{-L/2}^{L/2} f(x) e^{-(2\pi k x i)/L} \, dx \tag{43}$$

Tada je nova formula za Fourierov red izražena formulom (44):

$$f(x) = \sum_{k=-\infty}^{\infty} \frac{1}{L} F\left(\frac{k}{L}\right) e^{(2\pi i k x)/L} \tag{44}$$

Ukoliko se postavi tvrdnja da $L \to \infty$, vidi se kako varijabla $\frac{k}{L}$ iz diskretne prelazi u kontinuiranu varijablu s. Sada se jednadžba (43) može napisati na novi način što je izraženo jednadžbom (45):

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-2\pi s x i}\, dx \tag{45}$$

Dobivena se funkcija naziva Fourierovom transformacijom fukcije f [6].

3.2.2.1. Diskretna Fourierova transformacija (DFT)

Kada se dogodi situacija sa skupom kompleksnih brojeva, koja može proizaći iz uzorkovanja funkcije unutar određenog intervala, standardna Fourierova transformacija definirana za kontinuiran skup nije prikladna. U takvim se slučajevima uvodi diskretna Fourierova transformacija uz jednadžbu (46) [6]:

$$F_j = \sum_{k=0}^{n-1} f_k\, e^{\frac{2\pi i k j}{n}} \tag{46}$$

koja niz $f_0, \dots, f_{n-1}$ kompleksnih brojeva pretvara u niz $F_0, \dots, F_{n-1}$ kompleksnih brojeva s pripadajućim inverzom koji je izražen jednadžbom (47):

$$f_k = \frac{1}{n} \sum_{j=0}^{n-1} F_j\, e^{\frac{-2\pi i k j}{n}} \tag{47}$$

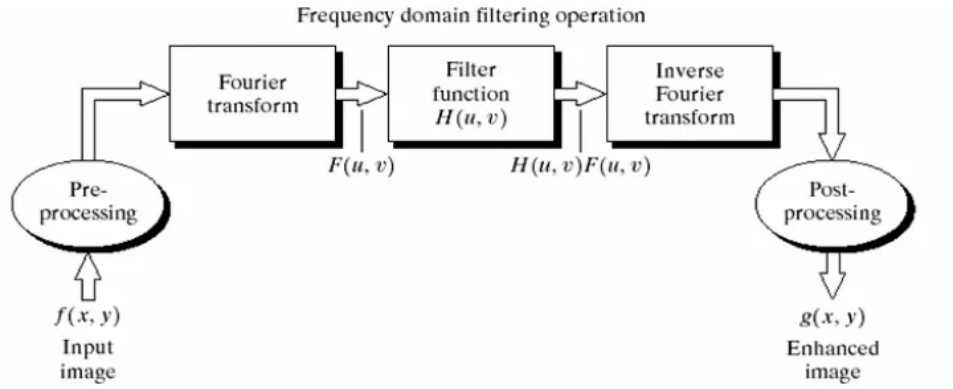
3.2.3. Koraci filtriranja u frekvencijskoj domeni

Tehnike obrade u frekvencijskoj domeni temelje se na modifikaciji Fourierove transformacije slike. Osnovna ideja pri upotrebi ove tehnike je unaprjeđenje slike manipulacijom transformacijskih koeficijenata slike poput diskretne Fourierove transformacije (DFT), diskretne valne transformacije (DWT) i diskretne kosinusne transformacije (DCT). Prednosti ovih metoda uključuju nisku složenost računanja, jednostavnost pregledavanja i manipulacije frekvencijskom kompozicijom slike te jednostavnost primjene svojstava posebnih domena transformacija. Funkcija poboljšanja slike u frekvencijskoj domeni označava se izrazom $g(x,y) = T[f(x,y)]$, gdje je $f(x,y)$ ulazna slika, a $g(x,y)$ poboljšana slika dobivena rezultatom izvođenja određene operacije T na određenoj frekvencijskoj komponenti

transformirane slike. Postupci potrebni za poboljšanje slike korištenjem tehnika u frekvencijskoj domeni uključuju pretvorbu ulazne slike u Fourierovu domenu, multiplikaciju Fourierove transformirane slike s filterom te izvršenje inverzne Fourierove transformacije slike kako bi se dobila poboljšana slika. Osnovni koraci za filtriranje u frekvencijskoj domeni glase [7]:

1. Učitavanje zadane ulazna slike $f(x, y)$ veličine $M\ x\ N$
2. Izračun $F(u, v)$, odnosno DFT slike
3. Množenje $F(u, v)$ s funkcijom filtera $H(u, v)$, npr. $G(u, v) = H(u, v)F(u, v)$
4. Izračun inverzne DFT od $G(u, v)$
5. Izdvajanje realnog dijela rezultata.

Navedeni je postupak slikovito prikazan na slici 18. Ulazna slika može se definirati kao dvodimenzionalna funkcija $f(x, y)$ gdje su x i y prostorne koordinate, a amplituda f pri bilo kojem paru koordinata (x, y) naziva se intenzitet ili siva razina slike u toj točki. Opća formula za filtriranje glasi: $G(u, v) = H(u, v)F(u, v)$, gdje je $H(u, v)$ transferna funkcija, $F(u, v)$ Fourierova transformacija funkcije slike, a $G(u, v)$ konačna filtrirana funkcija. Kod svih filtera važno je pronaći odgovarajuću funkciju $H(u, v)$ jer ona pojačava i suzbija određene frekvencijske komponente na slici.



Slika 18. Postupak filtriranja u frekvencijskoj domeni [8]

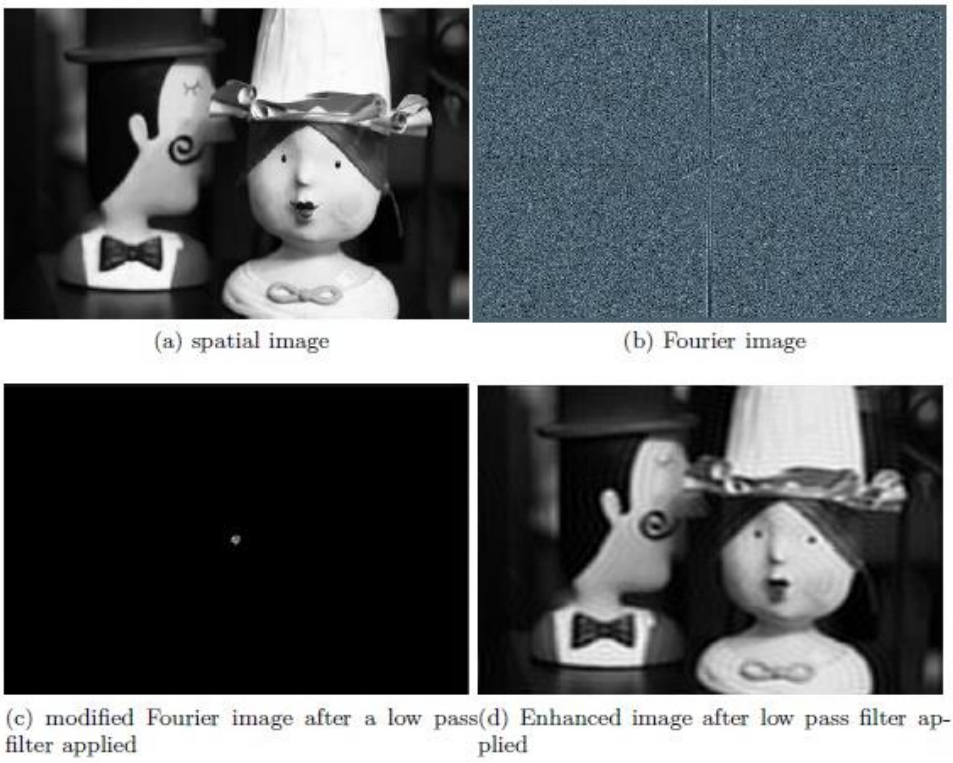
3.2.4. Klasifikacija filtriranja u frekvencijskoj domeni

Filteri frekvencijske domene koriste se za zaglađivanje i izoštravanje slike uklanjanjem komponenti visoke ili niske frekvencije. Klasifikacija filtriranja u frekvencijskoj domeni sastoji se od tri tipa filtera [9]:

- Niskopropusni filter – on uklanja visokofrekventne komponente čime se zadržavaju niskofrekventne komponente. Koristi se za zaglađivanje slike. Mehanizam niskopropusnog filtriranja u frekventnoj domeni dan je izrazom (48):

$$G(u,v) = H(u,v) \cdot F(u,v) \tag{48}$$

U frekvencijskoj domeni središte Fourierove slike jest niska frekvencija. Primjenom filtera kao što je prikazano na slici 19. c) sve frekvencije iznad granične frekvencije postavljene su na nulu. Zadnji korak prikazan na slici d) jest pretvorba iz frekvencijske domene u prostornu pomoću inverzne formule. Njezinom se primjenom jasno vidi kako su sa slike uklonjene visoke frekvencije čime se postiže efekt zaglađivanja [10].

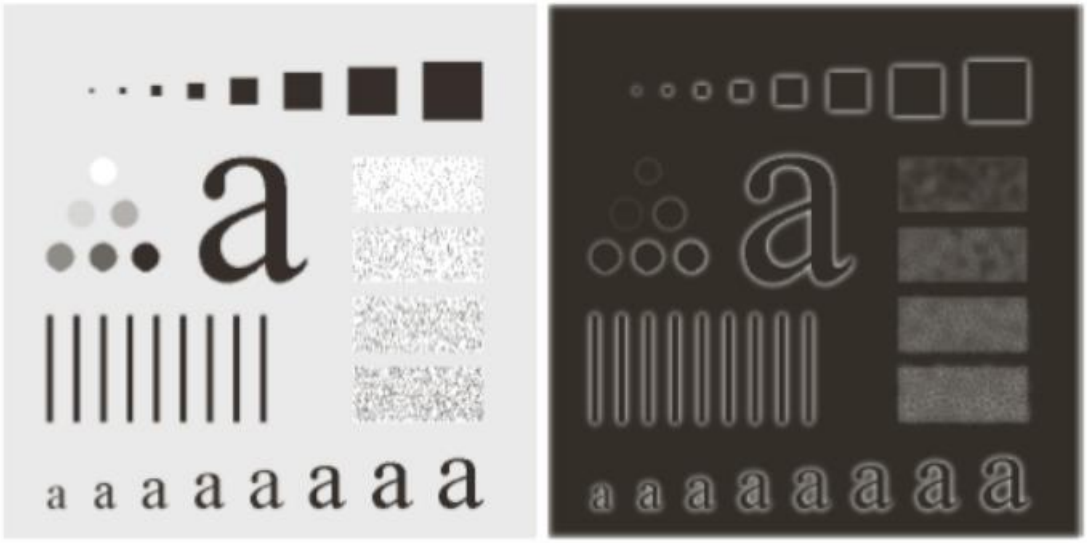


Slika 19. Primjena niskopropusnog filtera u frekvencijskoj domeni [10]

- Visokopropusni filter – on uklanja niskofrekventne komponente čime se zadržavaju visokofrekventne komponente. Koristi se za izoštravanje slike. Mehanizam visokopropusnog filtriranja u frekvencijskoj domeni dan je izrazom (49) [9]:

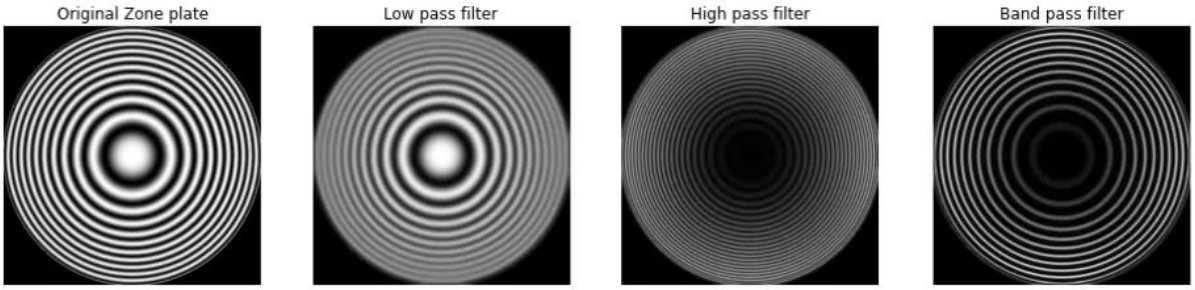
$$H(u,v) = 1 - H'(u,v) \tag{49}$$

Gdje je $H'(u,v)$ Fourierova transformacija niskopropusnog filtriranja. Primjer visokopropusnog filtriranja vidljiv je na slici 20.



Slika 20. Primjena visokopropusnog filtera u frekvencijskoj domeni [11]

- Pojasni filter uklanja vrlo visoke i vrlo niske frekvencije čime se zadržava umjeren raspon frekvencija. Filtriranje pojasnim filterom koristi se za poboljšanje rubova uz istovremeno smanjenje šumova [9]. Usporedba pojasnog filtera s prethodno spomenutima prikazana je na slici 21:



Slika 21. Usporedba raznih tipova filtera u frekvencijskoj domeni [12]

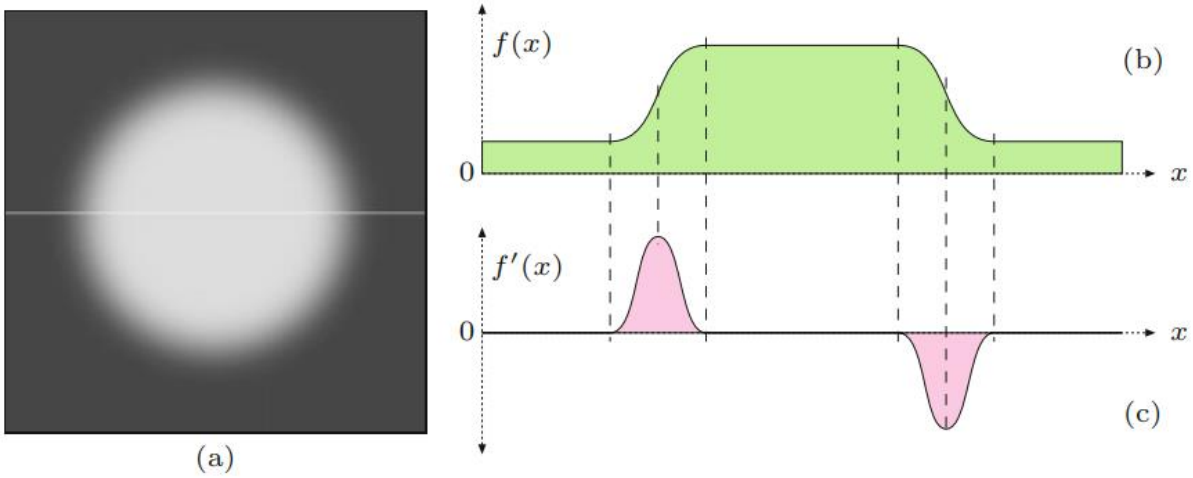
3.3. Usporedba filtriranja u prostornoj i frekvencijskoj domeni

Filtriranje u prostornoj i frekvencijskoj domeni predstavlja dva različita pristupa obradi slika. Filtriranje u prostornoj domeni podrazumijeva manipulaciju vrijednostima piksela izravno u slici, dakle bavi se sa samom ravninom slike. To se postiže primjenom kernela na svaki piksel u slici. Vrijednosti u kernelu koriste se za izračun ponderiranog prosjeka s vrijednostima

susjednih piksela. S druge strane, filtriranje u frekvencijskoj domeni obuhvaća transformaciju slike iz prostorne domene u frekvencijsku pomoću Fourierove transformacije. U frekvencijskoj domeni slika je predstavljena kao kombinacija različitih frekvencijskih komponenti. Filtriranje u prostornoj domeni je više intuitivno te lakše shvatljivo dok filtriranje u frekvencijskoj domeni može biti prilično komplicirano zbog kompleksnih matematičkih transformacija. Iz toga je razloga i skuplje jer zahtijeva više računalnih resursa. Iz navedenoga se može zaključiti kako je filtriranje u frekvencijskoj domeni sporije dok se izračuni u prostornoj domeni brže obrađuju. Jedna od glavnih prednosti frekvencijske domene jest sposobnost razdvajanja informacija na slici na temelju različitih frekvencijskih komponenti što omogućuje ciljano i specijalizirano procesiranje. Obje tehnike filtriranja koriste se u sličnim namjenama poput zaglađivanja ili izoštravanja slike, poboljšanje slike te detekcija rubova. I frekvencijska i prostorna domena imaju svoje prednosti i nedostatke, a optimalan odabir ovisi o karakteristikama slike te o specifičnim zahtjevima i ciljevima zadatka obrade slike.

4. DETEKCIJA RUBOVA

Detekcija rubova je jedan od temeljnih procesa povezan s računalnim vidom i obradom slike budući da rubovi sadrže širok spektar informacija povezanih sa sadržajem slike. Rub se pojavljuje između dva susjedna područja koja imaju različitu razinu intenziteta boje, odnosno svjetla [13]. Drugim riječima, rubovi su položaji slike gdje dolazi do promjene lokalnog intenziteta duž određene orijentacije. Što je lokalna promjena intenziteta jača, to je i veća šansa za pronalzak ruba na određenom području slike [14]. Najjednostavniji koncept za razvoj detektora rubova obuhvaća prvu derivaciju budući da je ona u matematici definirana kao količina promjene s obzirom na prostornu udaljenost. Pretpostavit će se situacija u jednoj dimenziji gdje slika sadrži jednu svijetlu regiju okruženu tamnom pozadinom. Navedeno je prikazano na slici 22 a). U tome bi slučaju profil intenziteta duž jedne linije izgledao kao 1D funkcija $f(x)$ što se vidi na slici 22 b). Računajući prvu derivaciju funkcije $f(x)$ dobiva se pozitivan skok na mjestima gdje intenzitet raste te negativan skok gdje vrijednost funkcije pada. Navedeno je vidljivo na slici 22 c).

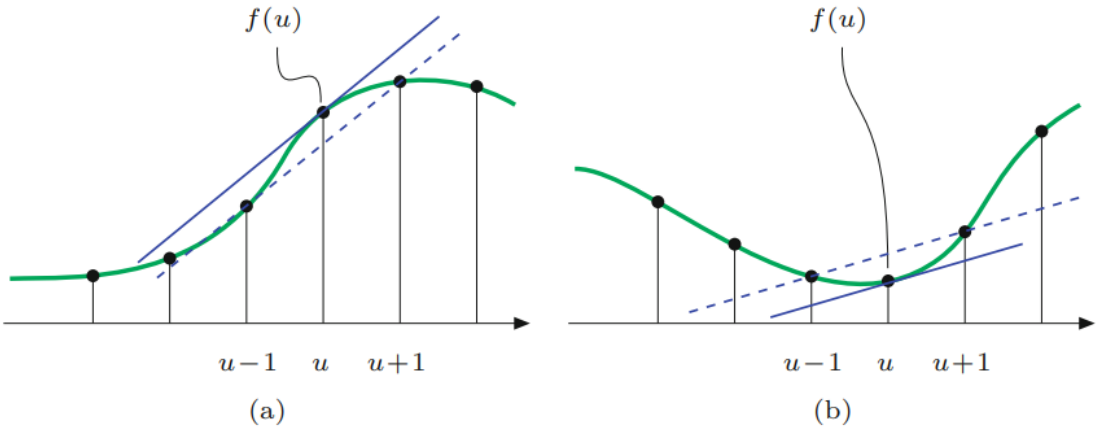


Slika 22. Prikaz prve derivacije na 1D slici [14]

Za razliku od kontinuiranih funkcija, prva derivacija nije definirana za diskretnu funkciju $f(u)$ koja zapravo predstavlja linijski profil stvarne slike pa je potrebno pronaći metodu za njezinu procjenu. Slika 23. prikazuje osnovu ideju rješenja. Prva derivacija kontinuirane funkcije na nekom položaju x može se još interpretirati kao nagib tangente te funkcije u položaju x. Metoda za grubu aproksimaciju nagiba tangente diskretne funkcije $f(u)$ jest postavljanje pravca kroz susjedne vrijednosti funkcije $f(u-1)$ i $f(u+1)$. Navedeno je prikazano jednadžbom (50):

$$\frac{df}{du}(u) \approx \frac{f(u+1)-f(u-1)}{(u+1)-(u-1)} = \frac{f(u+1)-f(u-1)}{2} \tag{50}$$

Naravno, ista metoda može se primijeniti i u okomitom smjeru za procjenu prve derivacije duž y-osi, odnosno duž stupaca slike.



Slika 23. Procjena prve derivacije diskretne funkcije [14]

Ukoliko se pak radi o višedimenzionalnim funkcijama, odnosno 2D slikama, potrebno je korištenje parcijalnih derivacija. Parcijalna derivacija jest derivacija višedimenzionalne funkcije duž jedne od njezinih koordinatnih osi. Jednadžbe (51) i (52) predstavljaju parcijalne derivacije funkcije neke 2D slike $I(u,v)$:

$$I_x = \frac{\partial I}{\partial x}(u,v) \tag{51}$$

$$I_y = \frac{\partial I}{\partial y}(u,v) \tag{52}$$

Vektor $\nabla I(u,v)$ naziva se gradijentom funkcije I u položaju (u,v). Gradijent funkcije definiran je izrazom (53):

$$\nabla I(u,v) = \begin{pmatrix} I_x(u,v) \\ I_y(u,v) \end{pmatrix} = \begin{pmatrix} \frac{\partial I}{\partial x}(u,v) \\ \frac{\partial I}{\partial y}(u,v) \end{pmatrix} \tag{53}$$

Veličina gradijenta, izražena jednadžbom (54), ostaje nepromjenjiva u odnosu na rotaciju slike pa s stoga neovisna o orijentaciji strukture slike. Navedeno je svojstvo važno kod izotropne lokalizacije rubova pa je to posljedično osnova mnogih metoda detekcije rubova.

$$|\nabla I| = \sqrt{I_x^2 + I_y^2} \tag{54}$$

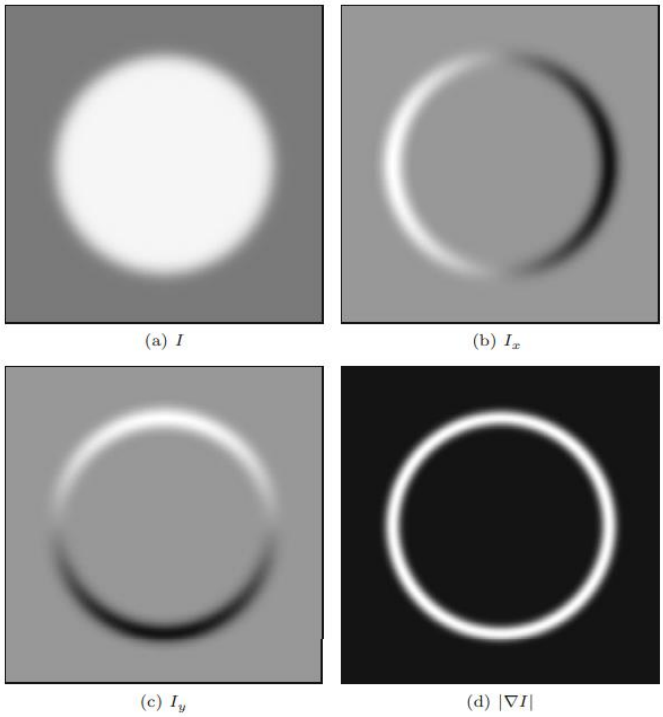
Komponente fukcije gradijenta jesu prve derivacije linija slike duž vodoravne i okomite osi. Aproksimacija prvih horizontalnih derivacija može se implementirati pomoću linearnog filtera s 1D kernelom danim izrazom (55) gdje se koeficijenti -0,5 i 0,5 primjenjuju na elemente slike $I(u-1,v)$ i $I(u+1,v)$.

$$\mathbf{H_x^D} = [-0{,}5 \quad 0 \quad 0{,}5] = 0{,}5 \cdot [-1 \quad 0 \quad 1] \tag{55}$$

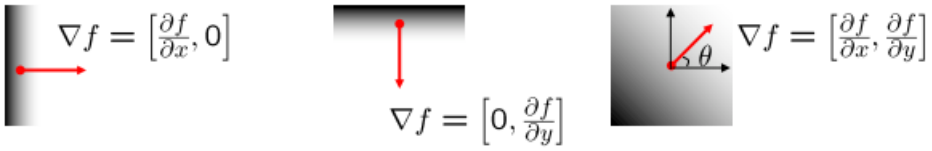
Središnji piksel je ponderiran nultim koeficijentom zbog čega se zanemaruje. Analogno tome se dobiva i vertikalna komponenta gradijenta, a prikaza je jednadžbom (56):

$$\mathbf{H_y^D} = \begin{bmatrix} -0{,}5 \\ 0 \\ 0{,}5 \end{bmatrix} = 0{,}5 \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{56}$$

Na slici 24. vidljiva je primjena gradijentnih filtera. Jasno se prepoznaje ovisnost o orijentaciji odziva filtera. Horizontalni gradijentni filter $\mathbf{H_x^D}$ najjače reagira na brze promjene duž vodoravnog smjera, odnosno na okomite rubove. Analogno tome, vertikalni gradijentni filter $\mathbf{H_y^D}$ najjače reagira na vodoravne rubove. U ravnim područjima slike, odziv filtera jednak je nuli [14]. Na slici 25. također je slikovito opisano svojstvo gradijenta koje govori da on pokazuje u smjeru najbržeg povećanja intenziteta.



Slika 24. Parcijalne derivacije 2D funkcije [14]



Slika 25. Svojstva gradijenta [15]

Kao što je prethodno spomenuto, lokalni gradijent funkcije slike temelj je mnogih klasičnih operatora za detekciju rubova. Razlikuju se uglavnom po vrsti filtera koji se koristi za procjenu komponenti gradijenta i načinu na koji se te komponente biraju [14]. U sljedećim će se poglavljima obraditi neke od istaknutijih metoda za detekciju rubova na slici uključujući one koji rade u prostornoj i frekvencijskoj domeni.

4.1. Filteri za detekciju rubova u prostornoj domeni

Detekcija rubova u prostornoj domeni temelji se na gradijentu slike koji zapravo predstavlja brzinu promjene vrijednosti intenziteta duž prostornih koordinata slike. Gradijent se može izračunati pomoću tehnika kao što su Sobelov, Prewittov ili Robertsov operator. Navedene tehnike obavljaju konvoluciju slike s malim kernelom te izračunavaju gradijent u vodoravnom i okomitom smjeru. Veličina gradijenta potom se računa kao korijen zbroja kvadrata vodoravnog i okomitog gradijenta. Ova veličina predstavlja jačinu ruba, a smjer gradijenta predstavlja orijentaciju ruba [16]. U daljnjim će se poglavljima obraditi neki od istaknutijih operatora za detekciju rubova u prostornoj domeni.

4.1.1. Prewitt i Sobel operatori

Prewitt i Sobel operatori koriste linearne filtere koji se protežu preko tri susjedna retka, odnosno stupca kako bi se suprotstavili osjetljivosti na šum jednostavnijih operatora gradijenta. Prewitt operator koristi kernele filtera dane izrazom (57):

$$\mathbf{H_x^P} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \mathbf{H_y^P} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{57}$$

Koji izračunavaju prosječne komponente gradijenta kroz tri susjedna retka ili stupca. $\mathbf{H_x^P}$ izvodi *box* zaglađivanje preko tri linije prije izračunavanja x gradijenta. Analogno tome, $\mathbf{H_y^P}$ izvršava zaglađivanje preko tri stupca prije izračunavanja gradijenta y. Zbog svojstva komutativnosti

linearne konvolucije o kojem se pričalo u poglavlju 3.1.1., može se reći i obrnuto, da se zaglađivanje primjenjuje nakon izračuna gradijenta.

Filteri koji uključuju Sobel operator gotovo su identični, međutim dio koji odrađuje zaglađivanje dodjeljuje veću težinu trenutnoj središnjoj liniji, odnosno stupcu. Navedeno je vidljivo u izrazu (58):

$$\mathbf{H_x^S} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \mathbf{H_y^S} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{58}$$

Procjene za komponente lokalnog gradijenta dobivene su iz rezultata filtera odgovarajućim skaliranjem. Za Prewitt operator to je iskazano izrazom (59), a za Sobel izrazom (60):

$$\nabla I(u,v) \approx \frac{1}{6} \cdot \begin{pmatrix} (I * \mathbf{H_x^P})(u,v) \\ (I * \mathbf{H_y^P})(u,v) \end{pmatrix} \tag{59}$$

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{pmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{pmatrix} \tag{60}$$

Skalirani rezultati filtera se potom zapisuju izrazom (61)(neovisno radi li se o Prewitt ili Sobel operatoru):

$$I_x = I * \mathbf{H_x} \qquad I_y = I * \mathbf{H_y} \tag{61}$$

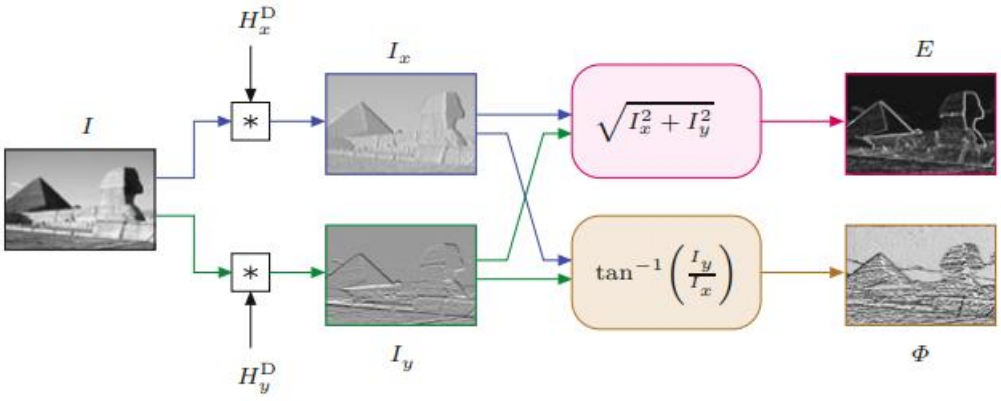
U oba je slučaja jačina lokalnog ruba izražena preko veličine gradijenta koja je iskazana izrazom (62)):

$$E(u,v) = \sqrt{I_x^2(u,v) + I_y^2(u,v)} \tag{62}$$

Lokalna orijentacija ruba izražena preko kuta dana je izrazom (63):

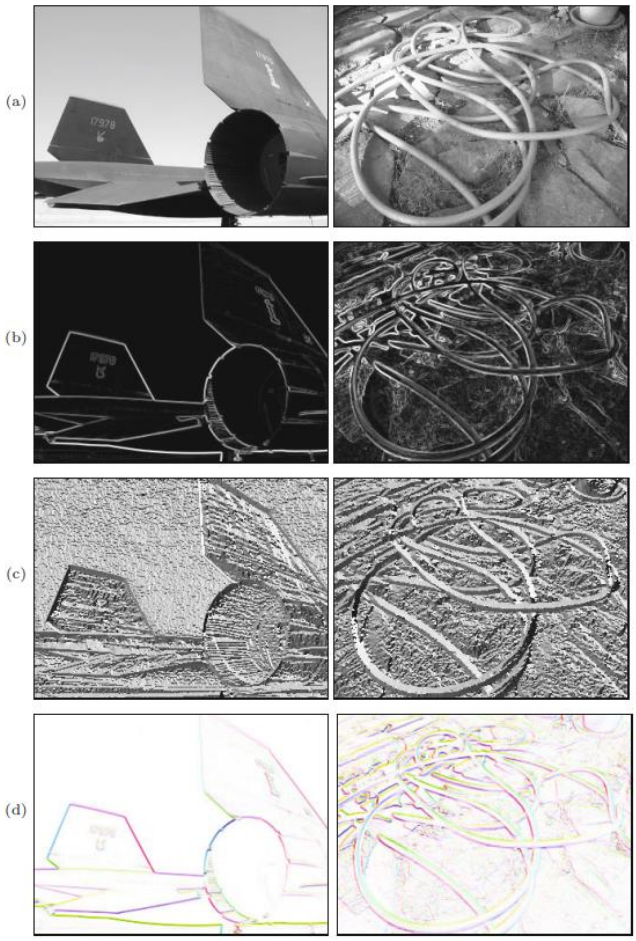
$$\Phi(u,v) = tan^{-1}\left(\frac{I_y(u,v)}{I_x(u,v)}\right) = Arctan(I_x(u,v), I_y(u,v)) \tag{63}$$

Čitav proces izdvajanja veličine i orijentacije ruba slikovito je prikazan na slici 26. Najprije se izvorna slika neovisno konvolvira s dva gradijentna filtera $\mathbf{H_x}$ i $\mathbf{H_y}$, a potom se kao rezultati filtera izračunavaju jačina ruba E i orijentacija Φ [14].



Slika 26. Proces ekstrakcije ruba korištenjem gradijenta [14]

Na slici 27. vidljiva je jačina ruba i orijentacija na dvije testne slike dobivene Sobelovim operatorom koristeći izraz (58).



Slika 27. Jačina ruba i orijentacija dobivene Sobelovim operatorom [14]

Na slici 27. a) dana je izvorna slika. Slike b) i c) pokazuju jačinu rubova te njihovu orijentaciju dok slike pod d) prikazuju orijentacijske kutove kodirane kao nijanse boja, uz jačinu rubova

koja kontrolira zasićenost boje. Procjena orijentacije ruba temeljena na Prewittovim i Sobelovim filterima relativno je netočna pa su predložene promjene Sobelovog filtera kako bi se minimizirala greška, a prikazane su jednadžbom (64):

$$\mathbf{H_x^{S'}} = \frac{1}{32}\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \qquad \mathbf{H_y^{S'}} = \frac{1}{32}\begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{64}$$

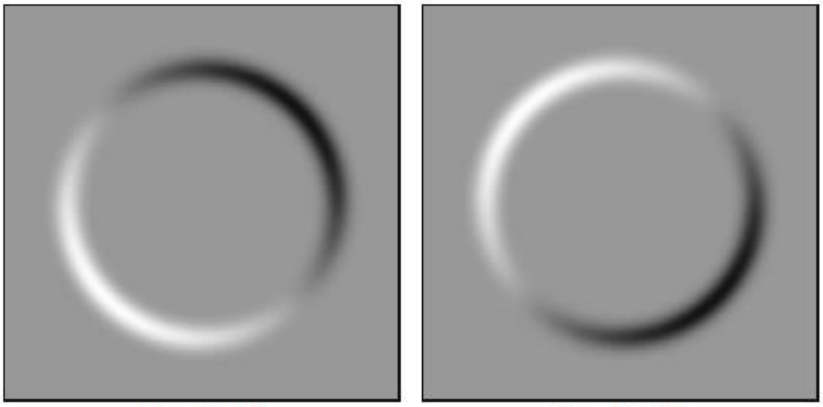
Navedeni se operatori često koriste zbog dobrih rezultata i jednostavne implementacije [14].

4.1.2. Roberts operator

Roberts operator jest jedan od najstarijih detektora rubova. Koristi dva iznimno mala filtera veličine $2x2$ za procjenu smjera gradijenta duž dijagonala slike koji su dani izrazom (65):

$$\mathbf{H_1^R} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \mathbf{H_2^R} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \tag{65}$$

Ovi filteri prirodno reagiraju na dijagonalne rubove, no nisu visoko selektivni prema orijentaciji. Drugim riječima, oba filtera pokazuju snažne rezultate u širokom rasponu kutova što je vidljivo na slici 28. Lokalna jačina ruba izračunava se mjerenjem duljine rezultantnog 2D vektora, slično izračunu gradijenta, no sa zakrenutim komponentama za 45° [14].

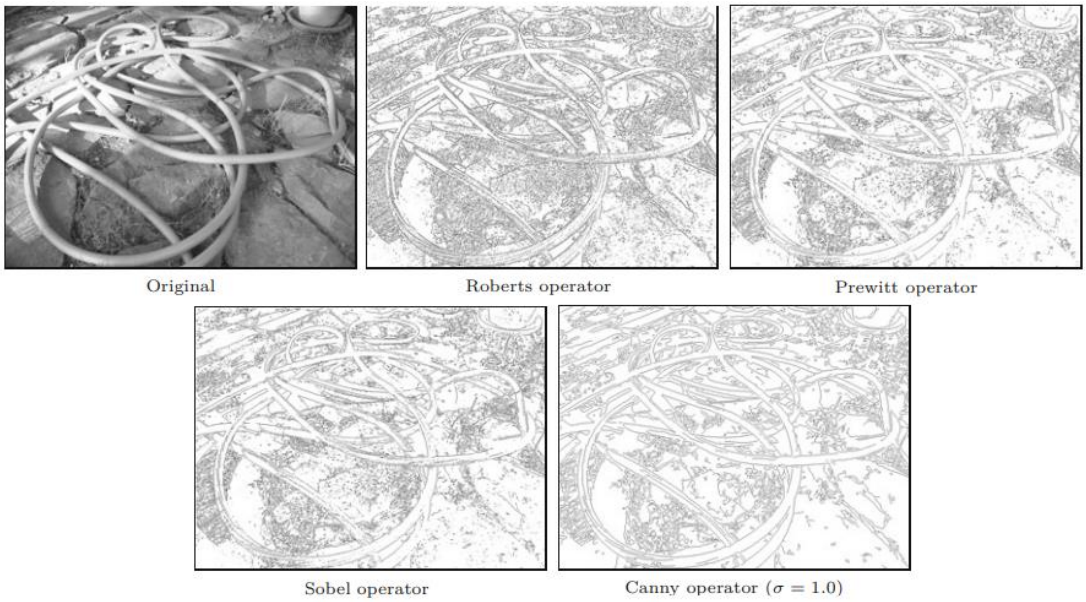


Slika 28. Komponente dijagonalnog gradijenta dobivenog Robertsovim operatorima [14]

4.1.3. Canny Edge detektor

Canny edge detektor jedna je od popularnijih i široko korištenih tehnika detekcije rubova kojoj je cilj identificirati i izdvojiti rubove objekata unutar slike. Razvio ga je John F. Canny 1986. godine te od tada postaje temeljni alat u računalnom vidu i analizi slike [17]. Metoda nastoji doseći tri glavna cilja: a) smanjiti broj lažnih rubnih točaka, b) postići dobru lokalizaciju rubova i c) dati samo jednu oznaku na svakom rubu. Navedena se svojstva obično ne postižu

jednostavnim rubnim operatorima. U suštini Cannyjev filter je gradijentna metoda, no upotrebljava nulte prijelaze drugih derivacija za preciznu lokalizaciju rubova. Potpuno implementiran, Canny detektor koristi skup relativno velikih, usmjerenih filtera na više rezolucija slike i spaja pojedinačne rezultate u zajedničku mapu rubova. Uobičajeno je koristiti samo jednostruku implementaciju algoritma s podesivim parametrom zaglađivanja σ koji je ipak bolji od većine jednostavnih rubnih operatora. Osim toga, algoritam uz binarnu mapu rubova daje i povezane lance rubnih piksela što uvelike olakšava sljedeće korake obrade. Na slici 29. vidljiva je usporedba prethodno spomenutih operatora s Canny operatorom. Jedan od važnijih kriterija za kvalitetno prepoznavanje jest količina „nereda“, odnosno nebitnih rubnih elemenata i povezanost dominantnih rubova. Robertsov operator ne reagira najbolje na male rubne strukture zbog male veličine filtera. Prewittov i Sobelov operator daju vrlo slične rezultate. Rubna mapa koju daje Cannyjev operator znatno je čišća od jednostavnijih operatora uz relativno malu vrijednost σ [14].



Slika 29. Usporedba raznih operatora za detekciju rubova [14]

Osnovni koraci *Canny edge detection* algoritma jesu [17]:

1. Pretvorba u sive tonove – prvi korak u Canny algoritmu za otkrivanje rubova jest pretvorba željene slike u sive tonove. Slike u sivim tonovima imaju kanal koji predstavlja intenzitet svakog piksela što pojednostavljuje proces otkrivanja rubova i smanjuje računsku složenost. Slike u boji predstavljene su tri kanala (crveni, zeleni,

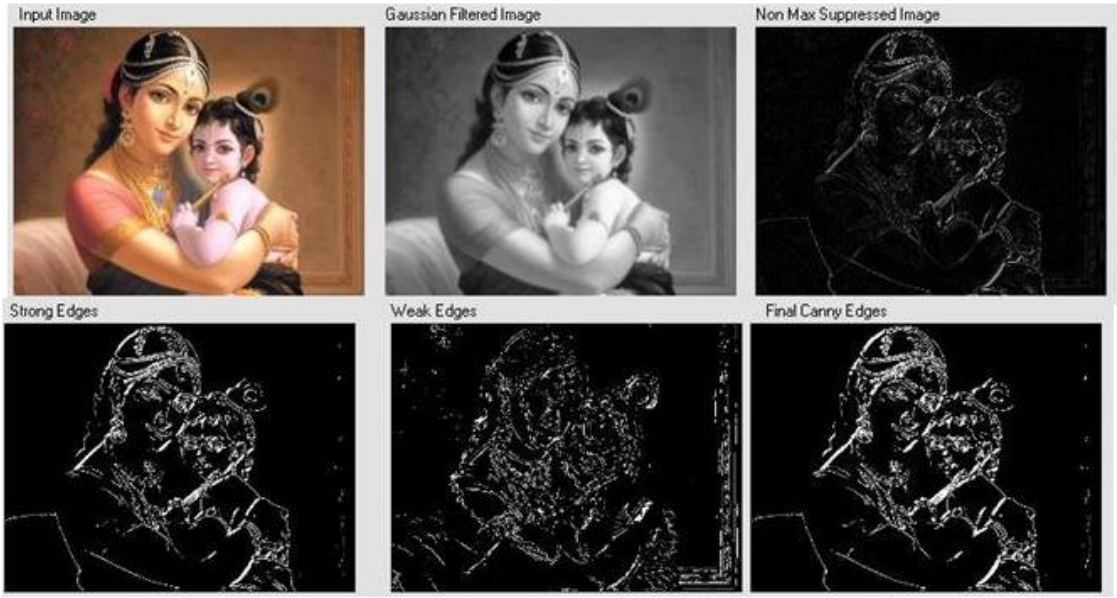
plavi - RGB). Pretvorba u sive tonove postiže se ponderiranjem sume RGB kanala za izračun vrijednosti intenziteta sivih tonova za svaki piksel.

2. Smanjenje šuma – U svrhu smanjenja šuma (buke) na slici primjenjuje se Gaussov filter. Pomoću operacije zaglađivanja dolazi do redukcije šuma. Šum može izazvati lažne rubove što smanjuje uspješnost točnost pronalaska pravih rubova. Gaussov filter zaglađuje sliku konvolvirajući je s Gaussovom jezgrom čime se smanjuje visokofrekventni šum uz očuvanje oštrine rubova. Gaussov se filter primjenjuje na svaki piksel na slici pomicanjem kernela preko slike i uzimanjem ponderiranog prosjeka intenziteta susjednih piksela. To znači da uzima u obzir svjetlinu okolnih piksela i daje veću važnost onima koji su bliže. Dakle, ukoliko je kernel veći, više piksela ulazi u izračun što rezultira jačim učinkom zamućenja na slici.
3. Izračun gradijenta – nakon smanjenja šuma, potrebno je izračunati gradijent zaglađene slike. Gradijent mjeri koliko se brzo mijenja intenzitet na lokaciji svakog piksela. Algoritam obično koristi Sobelov operator za određivanje veličine gradijenta i orijentacije za svaki piksel. Veličina gradijenta označava jačinu promjene intenziteta, a orijentacija određuje smjer najveće promjene. Izrazi povezani sa Sobelovim operatorom, veličinom i orijentacijom gradijenta dani su u poglavlju 4.1.1.
4. Suzbijanje ne-maksimalnih vrijednosti (*Non-maximum suppression*) – nakon određene veličine i orijentacije gradijenta na svakom pikselu, prelazi se na korak suzbijanja ne-maksimalnih vrijednosti. Navedeni korak učinkovito stanjuje rubove čime se omogućava čišći prikaz stvarnih rubova na slici. Postupak funkcionira na način da se ispituje veličina i orijentacija gradijenta pojedinog piksela te se uspoređuje sa susjednim pikselima duž smjera gradijenta. Ukoliko je veličina gradijenta središnjeg piksela najveća među susjedima, pretpostavlja se kako je taj piksel dio ruba te se zadržava. U slučaju da nije najveći, on se potiskuje postavljanjem njegovog intenziteta na nulu te se uklanja iz razmatranja kao rubnog piksela.
5. Dvostruki prag (*Double thresholding*) – ovaj korak izvršava kategorizaciju rubova u tri temeljne kategorije: jaki rubovi, slabi rubovi i ne-rubovi. U tu svrhu koriste se visoki i niski prag:
    - Pikseli s veličinama gradijenta iznad visokog praga smatraju se jakim rubovima što ukazuje na značajne promjene intenziteta

- Pikseli s veličinama gradijenta između niskog i visokog praga klasificiraju se kao slabi rubovi. Oni mogu predstavljat stvarne rubove ili šum pa ih je potrebno dodatno provjeriti
- Pikseli s veličinama gradijenta ispod niskog praga smatraju se ne-rubova te se posljedično odbacuju.

6. Praćenje rubova histerezom – zadnji korak Cannyjevog algoritma za detekciju rubova obuhvaća praćenje rubova histerezom. Navedeni korak ima cilj povezati slabe rubove koji su najvjerojatnije dijelovi pravih rubova, s jakim rubovima. Počevši od svakog piksela s jakim rubom, algoritam prati rub uzimajući u obzir susjedne piksele sa slabim rubom koji su povezani. Ako je slabi rubni piksel povezan s jakim rubnim pikselom, on se također smatra dijelom ruba te se zadržava. Čitav se proces ponavlja sve dok se više slabi rubovi ne povezuju. Ovo rezultira kontinuiranim i dobro definiranim rubovima.

Na slici 30. dan je prikaz slike kroz koju je proveden *Canny edge* algoritam.



Slika 30. Primjer Canny edge detektora na slici [18]

Cannyjev algoritam za detekciju rubova nudi sljedeće prednosti u odnosu na druge tehnike detekcije rubova [17]:

- Precizna lokalizacija rubova – pruža kvalitetno otkrivanje rubova zahvaljujući koraku suzbijanja ne-maksimalnih vrijednosti što rezultira zadržavanju samo najvažnijih rubova

- Niska stopa pogreške – korištenjem dvostrukog praga i histereze smanjuje se vjerojatnost lažnih rubova
- Pojedinačni odgovor na rubove – svaki rub na slici predstavljen je samo jednim odgovorom u izlazu čime se izbjegavaju dvostruka otkrivanja rubova
- Otporan na šum – upotrebom Gaussovog zaglađivanja u početku smanjuje se osjetljivost na šum.

Zaključno, Cannyjev algoritam široko je korištena i moćna metoda pronalaska rubova na slici. Njegov višefazni proces osigurava točnu i preciznu lokalizaciju rubova, niske stope pogrešaka te otpornost na šum. Kontinuiran razvoj i integracija Cannyjevog algoritma u različite primjene i tehnologije omogućuje napredak u poljima računalnog vida, rezultirajući još boljom interpretacijom vizualnih informacija [17].

4.2. Filteri za detekciju rubova u frekvencijskoj domeni

U frekvencijskoj domeni detekcija rubova često se postiže korištenjem visokopropusnih filtera. Visokopropusni filteri, kao što je i spomenuto u poglavlju 3.2.4. omogućavaju prolazak visokofrekventnih komponenti dok istodobno smanjuju ili potiskuju komponente niskih frekvencija. U kontekstu detekcije rubova, visokofrekventne komponente odnose se na brze promjene u intenzitetu koje su karakteristične upravo za rubove. U narednom će se poglavlju dati uvid u detekciju rubova koristeći visokopropusni filter [19].

4.2.1. Visokopropusni filter za detekciju rubova

Primjena visokopropusnog filtera iz frekvencijske domene u svrhu pronalaska rubova na slici uključuje primjenu Fourierove transformacije slike s frekvencijskim odzivom filtera. Potom je potrebno primijeniti visokofrekventni filter na transformiranu sliku čime se blokiraju sve niske frekvencije te ostavljaju komponente visokih frekvencija. Nakon što se napravi inverzna Fourierova transformacija, vidljive su razne rubne značajke na slici [19]. Slika 31. prikazuje primjer primjene visokofrekventnog filtera za detekciju rubova na slici.

Slika 31. Primjena visokofrekventnog filtera za detekciju rubova [19]

4.2.2. Canny edge detection u frekvencijskoj domeni

Canny edge filter obično se implementira u prostornoj domeni pomoću metoda temeljenih na gradijentu. Međutim, moguće je provesti detekciju rubova i u frekvencijskoj domeni koristeći Fourierovu transformaciju. Osnovna ideja je pretvorba slike u frekvencijsku domenu, identifikacija i modifikacija frekvencijske komponente te transformacija slike natrag u prostornu domenu. Dakle, koraci *Canny edge* detekcije rubova u frekvencijskoj domeni glase:

1. Primjerna Fourierove transformacije – pretvorba u frekvencijsku domenu tehnikama poput brze Fourierove transformacije
2. Filtriranje u frekvencijskoj domeni – u frekvencijskoj domeni rubovi odgovaraju visokofrekventnim komponentama pa se treba upotrijebiti filter koji će istaknuti, odnosno potisnuti određene komponente.
3. Inverzna Fourierova transformacija – vraćanje slike u prostornu domenu; rezultat transformacije dat će sliku s poboljšanim i istaknutim rubovima
4. Naknadna obrada slike – nakon povratka u prostornu domenu, moguće je primijeniti metode poput suzbijanja ne-maksimalnih vrijednosti i histereze s ciljem poboljšanja konačnog rezultata detekcije

Iako je tehnički sve navedeno izvedivo, implementacija *Canny edge* detektora rubova u frekvencijskoj domeni nije toliko primjenjivana koliko u prostornoj domeni. Metode u

frekvencijskoj domeni mogu biti računalno vrlo zahtjevne, zahtjevnije od onih u prostornoj domeni, a *Canny edge* detektor u prostornoj domeni je široko prihvaćen jer pruža visoku učinkovitost i djelotvornost. Iako implementacija može biti vrlo kompleksna, sve ovisi o karakteristikama slike te željenim rezultatima detekcije.

4.3. Usporedba filtera za detekciju rubova prostorne i frekvencijske domene

Detekcija rubova u prostornoj i frekvencijskoj domeni pruža dva različita pristupa u identifikaciji rubova na slici. Detekcija u prostornoj domeni uključuje analizu promjene intenziteta izravno na slici. Uobičajeni operatori koji se koriste jesu Prewitt, Sobel i Roberts operator. Česta je upotreba i Canny operatora koji predstavlja višefazno rješenje za detekciju rubova koje omogućuje precizan i pouzdan pronalazak rubova slike. Ovi pristupi omogućuju učinkovit i intuitivan pristup budući da se radi izravno s vrijednostima piksela. Negativna strana ovih tehnika jest da mogu bit osjetljive na šum te neefikasne za globalne informacije (osim Canny operatora koji je napravljen upravo kako bi izbjegao navedene probleme). S druge strane, detekcija u frekvencijskoj domeni uključuje korištenje Fourierove transformacije za analizu frekvencijskih komponenti slike. Visokopropusni filteri primjenjuju se kako bi istaknuli visokofrekvencijske komponente koje odgovaraju rubovima. Prednosti takvih metoda jesu učinkovito uklanjanje šuma te efikasnije shvaćanje globalnih informacija o strukturi slike, no zahtijeva više računalne snage te može rezultirati artefaktima. Izbor između navedenih tehnika ovisi o specifičnim zahtjevima te karakteristikama slika. Često se koristi i kombinacija tehnika iz prostorne i frekvencijske domene kako bi se postigli najbolji rezultati.

5. PROGRAMSKA APLIKACIJA

U radu je bilo potrebno ponuditi i vlastito rješenje filtera za detekciju rubova na slici. Nakon pretrage brojnih filtera i operatora u obje domene, odlučeno je ukomponirati nekoliko metoda. Programski kod pisan je u Pythonu uz potrebne biblioteke. Korišten kod vidljiv je u prilozima, a kroz njega će se proći korak po korak sa svim potrebnim objašnjenjima popraćenim primjerima.

5.1. Programski jezik Python

Python je interpretirani, interaktivni te objektivno orijentirani programski jezik koji uključuje module, iznimke, dinamičko povezivanje te dinamičke tipove podataka visoke razine i klase. Podržava razne tipove programiranja i izvan objektnog, primjerice proceduralno i funkcionalno programiranje. Posjeduje veliku snagu s intuitivnom sintaksom te posjeduje sučelja za mnoge sistemske pozive i biblioteke, a proširiv je u C ili C++. Budući da Python nudi jednostavnost, fleksibilnost, dosljednost te pristup raznim bibliotekama i okvirima za umjetnu inteligenciju i strojno učenje, smatra se najprikladnijim za projekte temeljene na umjetnog inteligenciji. Softverske biblioteke su unaprijed napisani kodovi koji se upotrebljavaju kod rješavanja uobičajenih programskih zadataka [20]. U nastavku će biti navedene biblioteke korištene pri stvaranju programske aplikacije te neke njihove značajke.

5.1.1. NumPy

NumPy je biblioteka Pythona koja sadrži višedimenzionalne objekte u nizovima i paket integrirajućih alata za implementaciju Pythona. U osnovi je kombinacija C-a i Pythona koji se upotrebljava za tradicionalno MATLAB programiranje gdje se podaci u obliku brojeva tretiraju poput nizova za višedimenzionalne funkcije i operacije preuređivanja. NumPy znatno olakšava rad s velikim nizovima i matricama korištenjem različitih matematičkih funkcija. Pruža funkcije koje omogućuju izvođenje osnovnih, ali i naprednijih matematičkih i statističkih funkcija na višedimenzionalnim nizovima i matricama uz malo redaka koda. Glavni nedostatak je što se samo jedna vrsta podataka može pohraniti u svaki stupac [21].

5.1.2. OpenCV

OpenCV (engl. Open Source Computer Vision) je biblioteka softvera za računalni vid i strojno učenje otvorenog koda. Napravljen je s namjerom da osigura zajedničku infrastrukturu za aplikacije računalnog vida i ubrza korištenje strojne percepcije u komercijalnim proizvodima.

Biblioteka ima više od 2500 optimiziranih algoritama što uključuje opsežan skup klasičnih, ali i suvremenijih algoritama računalnog vida i strojnog učenja. Algoritmi se mogu koristiti za identifikaciju objekata i lica, klasifikaciju radnji, praćenje pokreta i objekata i slično [22].

5.2. Izrada programske aplikacije

Kao što je prethodno spomenuto, programski kod pisan je u Pythonu uz biblioteke NumPy i OpenCV. Prvi korak u izradi aplikacije jest upravo učitavanje navedenih biblioteka. Naredba cv2 predstavlja OpenCV te služi za obradu slika, dok se NumPy koristi za numeričke operacije. Postupak učitavanja biblioteka vidljiv je na slici 32.

```
import cv2
import numpy as np
```

Slika 32. Učitavanje potrebnih biblioteka

Idući je korak definiranje funkcije. Funkcija za detekciju rubova nazvana je custom_edge_detection te ona uzima putanju slikovne datoteke. Slika se učitava naredbom cv2.imread(). Definiranje funkcije prikazano je slikom 33.

```
def custom_edge_detection(image_path):
    # Step 1: Read the image
    original_image = cv2.imread(image_path)
```

Slika 33. Definiranje funkcije

Nakon definiranja funkcije, dohvaća se visina i širina odabrane slike. Potom se postavlja željena visina slike, ovome slučaju 360 piksela, uz naglasak da se zadržava omjer slike. Veličina slike je proizvoljno odabrana, a razlog ovog koraka jest kako bi svaka slika kroz koju se želi provući filter odgovarala zaslonu, odnosno „stala“ u njega, a da se pritom ne izobliči. Kada se izvorna slika dovede u željeni dimenzijski omjer, poprima sive tonove što je i prvi korak u stvaranju filtera za detekciju rubova. Svi prethodno navedeni koraci vidljivi su na slici 34.

```
screen_height, screen_width, _ = original_image.shape
target_height = 360
aspect_ratio = target_height / screen_height
original_image_resized = cv2.resize(original_image, (int(screen_width * aspect_ratio), target_height))
grayscale_image = cv2.cvtColor(original_image_resized, cv2.COLOR_BGR2GRAY)
```

Slika 34. Postavljanje željene dimenzije

Potom dolazi dio koji izvodi prilagođenu detekciju rubova na temelju gradijenta pomoću Sobelovog operatora. Prvo se primjenjuje Sobel operator u smjeru x-osi slike u sivim tonovima. Parametar $ksize = 3$ predstavlja veličinu Sobel kernela. Analogno tome se primjenjuje Sobel operator u smjeru y-osi. Ove operacije služe za izračun gradijenta intenziteta slike u x i y smjeru. Koristeći Euklidsku normu izračunava se veličina gradijenta na svakom pikselu. Pojam veličine gradijenta iznesen je u poglavlju 4. gdje je dana teorijska osnova te formule vezane uz nju. Slijedi izdvajanje vrijednosti veličine gradijenta na raspon [0,255] kako bi se osiguralo da su sve vrijednosti unutar raspona za 8-bitnu sliku. Na kraju se izdvojena veličina gradijenta pretvara u cijeli broj što je neophodno za ispravan prikaz slike. Navedeno je vidljivo na slici 35.

```
sobel_x = cv2.Sobel(grayscale_image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(grayscale_image, cv2.CV_64F, 0, 1, ksize=3)
gradient_magnitude = np.sqrt(sobel_x ** 2 + sobel_y ** 2)
enhanced_edges = np.clip(gradient_magnitude, 0, 255).astype(np.uint8)
```

Slika 35. Izračun veličine gradijenta pomoću Sobel operatora

Sljedeći dio koda služi za poboljšanje rubova koristeći binarni prag (engl. *threshold*) . Naredba v2.threshold(enhanced_edges, 60, 255, cv2.THRESH_BINARY) primjenjuje binarni prag na poboljšanu sliku uz uvjet da pikseli s vrijednostima većim od 60 preuzimaju vrijednost 255, što znači da su bijeli, a pikseli s vrijednostima manjim ili jednakim 60 postavljeni su na 0 što znači da su crni. Potrebno je ponovno promijeniti veličinu slike zadržavajući prethodno izračunat omjer širine i visine. Navedeno je prikazano na slici 36.

```
_, thresholded_edges = cv2.threshold(enhanced_edges, 60, 255, cv2.THRESH_BINARY)
thresholded_edges_resized = cv2.resize(thresholded_edges, (int(screen_width * aspect_ratio), target_height))
```

Slika 36. Poboljšanje rubova slike koristeći threshold

Zadnji dio koda služi za stvaranje i sastavljanje prozora koji će prikazivati originalnu sliku promijenjene veličine i sliku s detektiranim rubovima jednu pored druge. U kodu je vidljivo

kako se prvo odabire lijeva polovica prozora te se za nju odabire izvorna slika, dok je filtrirana slika predviđena za desnu polovicu prozora. Navedeno je vidljivo na slici 37.

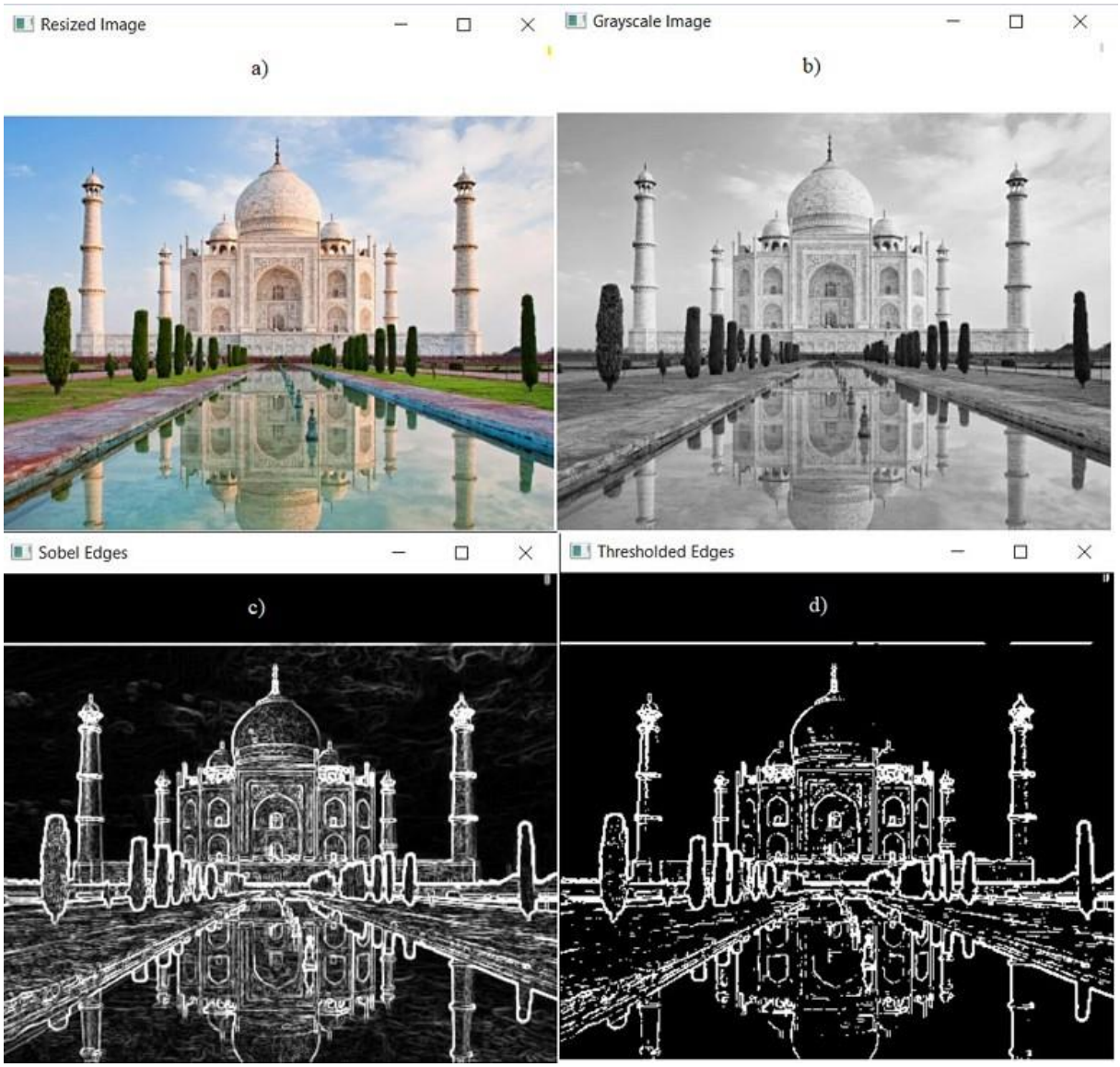
```
canvas = np.zeros((target_height, 2 * int(screen_width * aspect_ratio), 3), dtype=np.uint8)
canvas[:, :int(screen_width * aspect_ratio)] = original_image_resized
canvas[:, int(screen_width * aspect_ratio):] = cv2.cvtColor(thresholded_edges_resized, cv2.COLOR_GRAY2BGR)
cv2.imshow('Custom Edge Detection', canvas)
```

Slika 37. Stvaranje prozora za izvornu i filtriranu sliku

Na samome kraju pišu se linije koda odgovorne za rukovanje prikazom prozora. Ukratko, kod učitava Sliku 1, izvodi prilagođenu detekciju rubova te prikazuje originalnu i filtriranu sliku u jednom prozoru. Kako bi se cijeli proces filtriranja lakše vizualizirao, na slici 38. dan je dijagram toka za navedeni proces.

Slika 38. Dijagram toka filtriranja slike

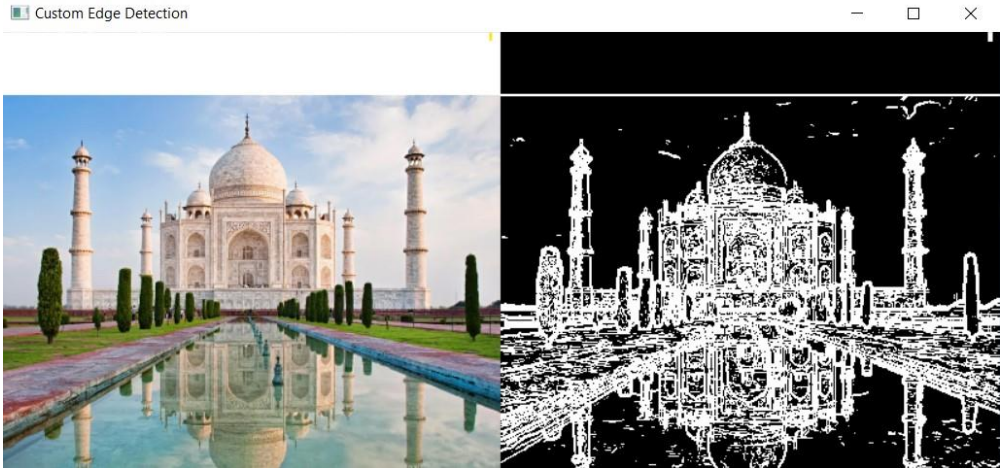
Na slici 39. slikovno su prikazani koraci dani u dijagramu toka. Slika a) prikazuje promjenu veličine slike. Kao što je prethodno navedeno, ona je proizvoljno odabrana tako da visina slike iznosi 360 piksela. Na slici b) vidljiva je pretvorba slike u sive tonove. Potom slijedi primjena Sobelovog operatora vidljiva na slici c), a konačan rezultat filtriranja slike nakon primjene *thresholda* vidljiv je na slici d).



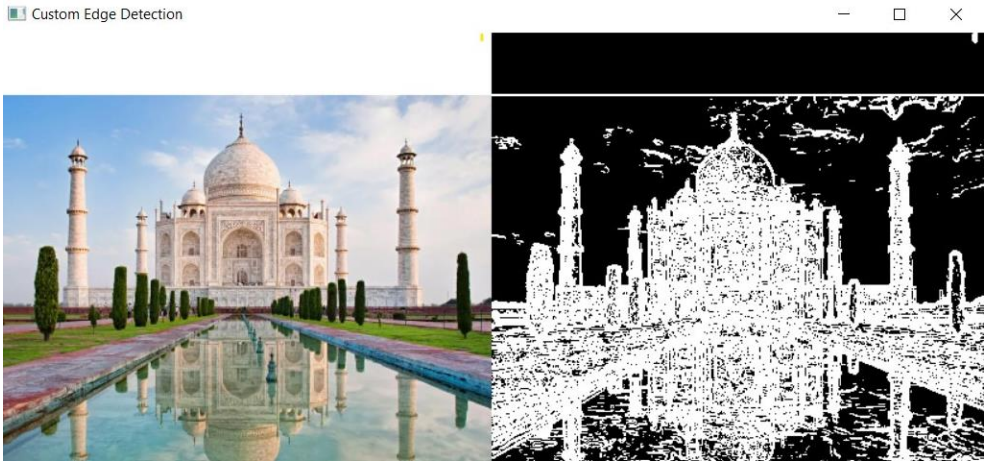
Slika 39. Slikovni prikaz dijagrama toka

Budući da kod sadrži opciju *threshold*, koja ima ključnu ulogu u detekciji rubova jer određuje koji intenziteti piksela se smatraju rubom, moguće je upravljati njegovom vrijednosti i na taj način dobiti različite rezultate pri provođenju koda. Slika 40. prikazuje threshold veličine 60. Ukoliko se vrijednost *thresholda* postavi prenisko, širok raspon intenziteta piksela smatrat će

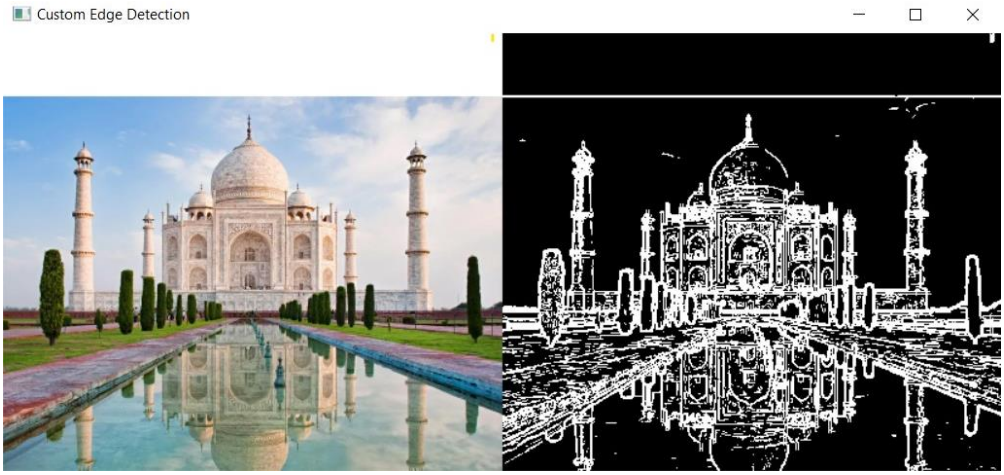
se rubom što rezultira time da se i šumovi detektiraju kao rubovi. Na slici 41. vidljivo je kako se prag postavljen prenisko (vrijednost 30) te rubovi nisu jasno detektirani. Na slici 42. vrijednost je postavljena na 90 što rezultira većom preciznošću i čistoćom rubova.



Slika 40. Filtrirana slika uz *threshold* 60 [26]



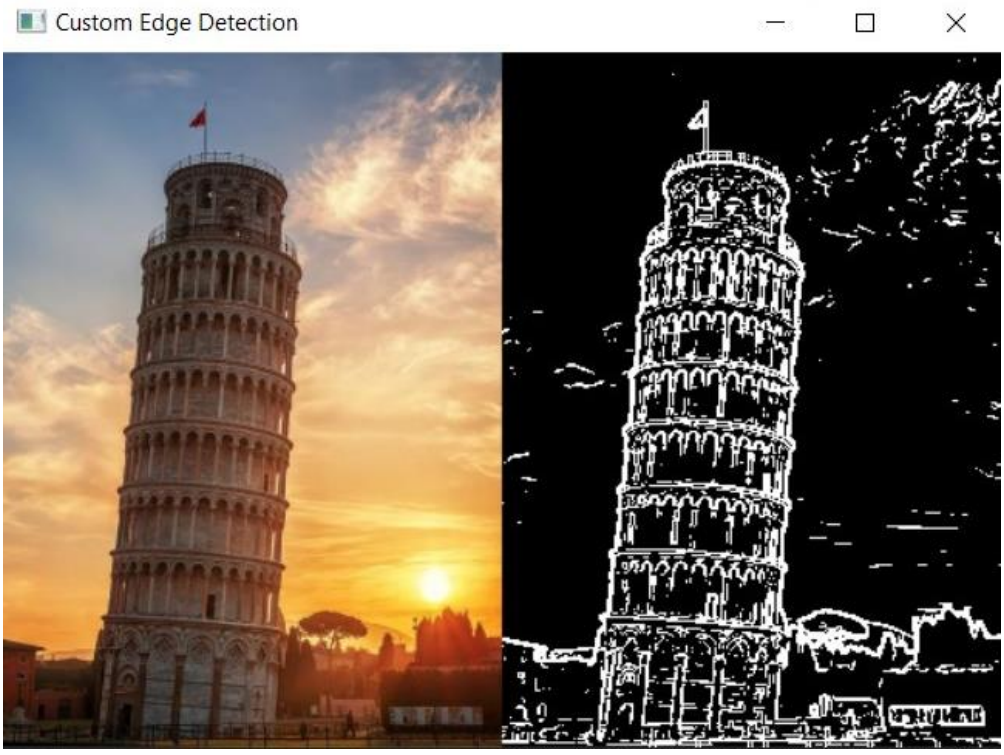
Slika 41. Filtrirana slika uz *threshold* 30 [26]



Slika 42. Filtrirana slika uz *threshold* 90 [26]

Još nekoliko primjera primjene vlastitog filtera na preuzete slike vidljivo je na slikama 43., 44. i 45.

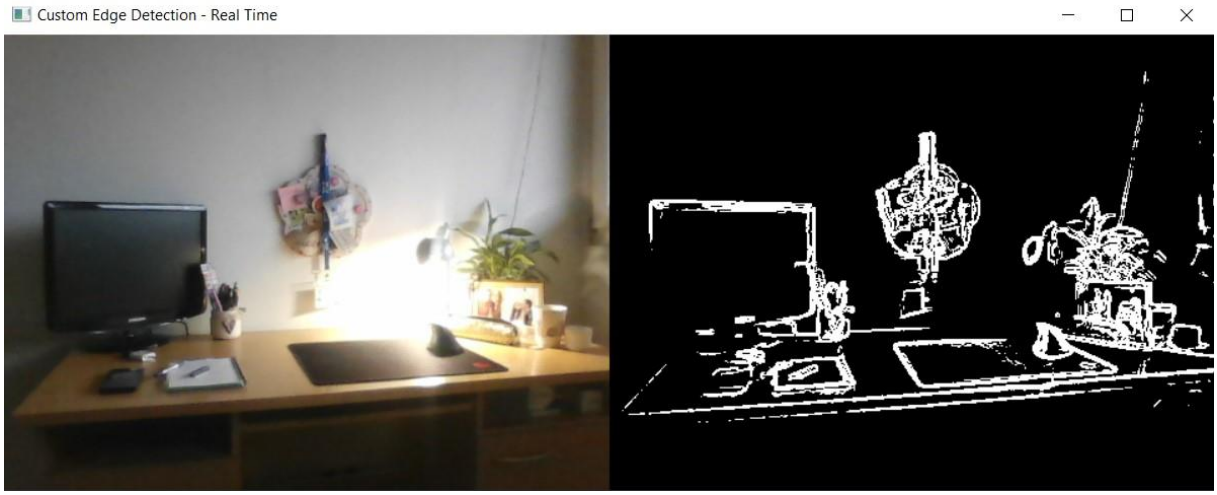
Slika 43. Detekcija rubova - Eiffelov toranj [27]



Slika 44. Detekcija rubova - Kosi toranj u Pisi [28]

Slika 45. Detekcija rubova - Machu Picchu [29]

Osim učitavanja slike u kod, moguća je modifikacija koda za dobivanje filtrirane slike u stvarnom vremenu. Promjene u kodu vidljive su u prilogu. U ovome je radu korištena kamera od laptopa, a primjer filtrirane slike u stvarnom vremenu vidljiv je na slici 46.

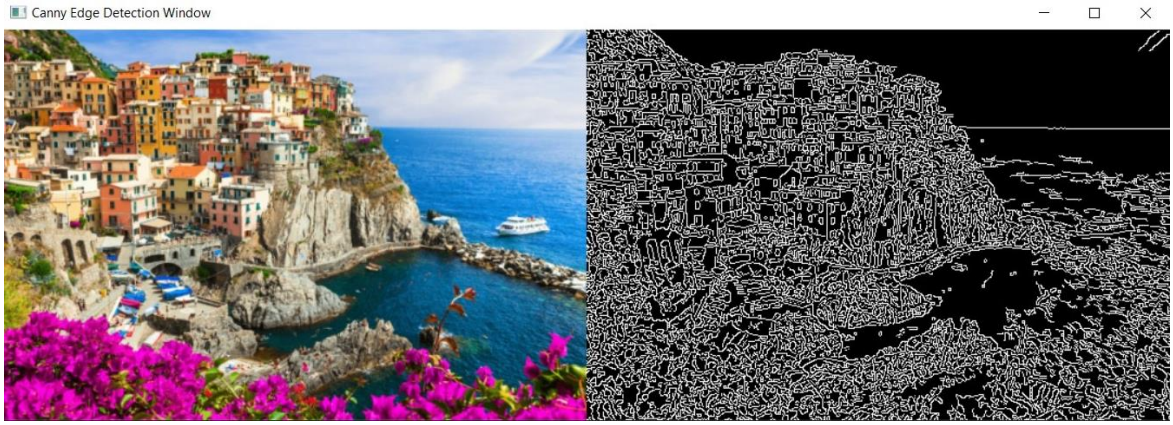


Slika 46. Primjer filtrirane slike u stvarnom vremenu

Kako bi se moglo prosuditi o preciznosti i uspješnosti vlastito osmišljenog filtera, na jednoj slici usporedit će se više metoda za detekciju rubova. Za ove potrebe, uz vlastiti *Custom Edge detection* filter, za usporedbu će se koristiti i Sobelov operator te *Canny edge detection* filter. O navedenim filterima se pričalo u prethodnim poglavljima gdje su dane njihove karakteristike, prednosti i nedostaci. Prema vlastitoj procjeni, Sobel operator trebao bi davati najslabije rezultate, potom *Custom Edge detection* filter te *Canny Edge detection* filter s najboljim rezultatima kao najnaprednija metoda pronalaska rubova na slici. Na slici 47. vidljiv je primjer upotrebe Sobelovog operatora, na slici 48. vlastitog filtera te na slici 49. *Canny edge detection* filtera.

Slika 47. Sobelov operator za detekciju rubova [30]

Slika 48. Custom Edge detection za detekciju rubova [30]



Slika 49. Canny edge detection filter za detekciju rubova [30]

6. KRITIČKI OSVRT

Kao što je i prethodno pretpostavljeno, na slikama je jasno vidljivo kako Sobelov operator daje najgore rezultate u detekciji rubova dok *Canny edge detection* filter daje najbolje rezultate. Budući da je vlastito izrađen filter inspiriran Sobelovim operatorom, uz dodatna poboljšanja, bilo je očekivano kako će u detekciji biti uspješniji od samog Sobelovog operatora, no ne toliko precizan poput Canny-jevog filtera. Poput Sobela, *Custom edge detection* algoritam izračunava gradijente pomoću operatora *cv.Sobel* u oba smjera *x* i *y* te se veličina gradijenta računa kao korijen zbroja kvadriranih derivacija. Ono što ga razlikuje od Sobela jest upravo to da Sobel sam po sebi koristi apsolutnu vrijednost derivacija te ne koristi metodu *threshold* koja je vrlo značajna pri klasifikaciji piksela koji se smatraju rubom. Upravo je iz tog razloga vidljivo kako slika 47. sadržava veliku količinu šuma zbog čega rubovi nisu precizni i čisti što rezultira dojmom kako je slika nejasna i mutna. Upotrebom *thresholda* te odabirom njegove odgovarajuće vrijednosti, reducira se opseg intenziteta piksela koji se smatraju rubom zbog čega su rubovi na slici 48. puno jasniji i vizualno bolje prepoznatljivi. Postoje i neke sličnosti s Canny-jevim filterom, no on koristi više koraka koji uključuju zaglađivanje Gaussovim filterom, izračun gradijenta, metode *non-maximum suppression* i *double-threshold* te praćenje rubova pomoću histereze. U vlastitom se filteru preskaču koraci zaglađivanja Gaussovim filterom te potiskivanja ne-maksimalnih vrijednosti. Osim toga, Canny koristi dvostruki prag (*double-threshold*) što je sofisticiranije od upotrebe pojedinačnog praga koji je korišten u vlastitom algoritmu. Upravo je iz prethodno navedenih razloga vidljivo kako su na slici 49. rubovi puno tanji i precizniji u usporedbi s prethodne dvije slike. Po pitanju koji je filter najbolji, ne postoji samo jedan ispravan odgovor. Najbolji način detekcije rubova ovisi o specifičnim zahtjevima pojedinog zadatka. Sobel je jedan od osnovnih i široko korištenih metoda za detekciju rubova. Računski je manje zahtjevan, no proizvodi dosta buke i šumova. *Custom edge detection* filter također je jednostavan i brz, nadopunjava Sobelove nedostatke zbog čega je rezultat detekcije puno jasniji i čišći. Ipak, za neke zahtjevnije detekcije nedostaje mu složenosti. Canny-jeva metoda je robusnija na buku i pruža najbolju lokalizaciju rubova zbog čega je algoritam zahtjevniji i sporiji. Iz svega navedenog može se zaključiti kako najbolji način detekcije rubova ovisi o kompromisima između jednostavnosti, brzine, otpornosti na buku i lokalizacije rubova potrebnih za specifičnu upotrebu. Sobelov operator često se koristi u situacijama gdje su jednostavnost i brzina ključne. Često se primjenjuje kod detekcije rubova u stvarnom vremenu poput obrade videozapisa. Canny se široko koristi kada je potrebna detekcija

rubova visoke kvalitete s dobrom lokalizacijom. Prikladan je za zadatke kada je preciznost važnija od brzine. Često se koristi u računalnom vidu i kod obrade slika. Vlastiti algoritam idealan je za situacije gdje se traži ravnoteža između jednostavnosti, brzine i učinkovitosti detekcije. Ono što je prednost vlastitog algoritma jest mogućnost podešavanja vrijednosti *thresholda* čime se upravlja intenzitetom piksela koji se smatraju rubom. Naravno, postoji i puno prostora za poboljšanje algoritma, a u nastavku će biti navedeni neki od prijedloga. Za početak, primjena Gaussovog filtera u svrhu zaglađivanja slike što rezultira smanjenjem buke. Još jedna od mogućnosti bila bi isprobavanje varijacija poput korištenja apsolutnih vrijednosti Sobel derivacija, različit izračun veličine gradijenta. Moguća je i kombinacija raznih operatora za detekciju rubova. Primjerice, uz Sobelov operator uključiti i Prewitt ili Roberts operator za moguće poboljšanje ukupne detekcije rubova. Može se iskoristiti i histereza kojom se postavljaju pragovi gdje se rubovi visokog praga smatraju snažnim rubovima, a rubovi niskog praga smatraju se slabim rubovima. Slabi rubovi uključuju se u konačni rezultat samo ako su povezani sa snažnim rubovima. Ovisno o složenosti slika, moguće je i pristupiti detekciji rubova temeljenoj na strojnom učenju. Konvolucijske neuronske mreže mogu se istrenirati da nauče i prilagode se specifičnim značajkama slika. Eksperimentiranje i iterativni razvoj ključni su za usavršavanje i poboljšanje performansi uz specifičnost i ciljeve koje se žele postići detekcijom rubova.

ZAKLJUČAK

Rubovi na slikama imaju ključnu ulogu u razumijevanju i interpretaciji vizualnih informacija. Oni predstavljaju mjesta nagle promjene intenziteta boje ili svjetlosti zbog čega označavaju granice između različitih objekata, tekstura ili regija. Detekcija rubova značajna je u područjima segmentacije slike, prepoznavanja objekata, obrade i kompresije slike te u računalnom vidu i robotici. Odabir algoritma za detekciju rubova ima krucijalnu ulogu u otkrivanju značajnih struktura. Vlastito osmišljen filter, *Custom Edge detection* filter, predstavlja prilagođen pristup detekciji rubova, pružajući ravnotežu između jednostavnosti, brzine i učinkovitosti detekcije. Navedeni algoritam koristi Sobel operator te metodu *threshold*. Metoda *threshold* ima značajnu ulogu u detekciji rubova jer određuje koji intenziteti piksela se smatraju rubom pa je moguće upravljati njegovom vrijednosti i na taj način dobiti optimalan rezultat detekcije. Ova kombinacija ima cilj smanjenja buke, hvatanja informacija o gradijentu te poboljšanja vidljivosti rubova. Algoritam je pogodan za situacije koje zahtijevaju brzu i jednostavnu detekciju rubova, poput obrade videozapisa u stvarnom vremenu. Njegova prilagodljivost i jednostavnost čine ga pogodnim i za situacije u kojima su računalni resursi ograničeni, a nužno je brzo donošenje odluka na temelju informacija o rubovima. Unatoč navedenim prednostima, zahtjevnije detekcije traže složenije algoritme poput *Canny edge* algoritma. On je prikladan za zadatke kada je preciznost važnija od brzine. Ipak, postoje brojne tehnike za daljnje usavršavanje vlastitog filtera koje ga mogu približiti mogućnostima *Canny edge* filtera. Neke od njih jesu primjena Gaussovog filtera, kombinacija raznih operatora, dvostruko pragiranje, upotreba histereze. Značaj vlastitog algoritma nije samo u sposobnosti suočavanja s jedinstvenim zadacima već i u njegovoj prilagodljivosti i potencijalu za usavršavanje. Budući da tehnologija svakodnevno napreduje, a samim time i zahtjevi postaju izazovniji, potraga za optimalnim algoritmom za detekciju rubova je neprekidna, no uvijek će biti vođena specifičnim potrebama zadataka obrade slika.

LITERATURA

[1] Medium. URL: https://medium.com/@shashikadilhani97/digital-image-processing-filters-832ec6d18a73 (Pristupljeno 9.9.2023.)

[2] Gonzalez, R.C., Woods, R. E. (2018) *Digital Image Processing*. Fourth edition. New York: Pearson

[3] IBM. URL: https://www.ibm.com/topics/computer-vision (Pristupljeno 10.9.2023.)

[4] Szeliski, R. (2010) *Computer Vision: Algorithms and Applications*. Washington. The University of Washington

[5] Bolf, N. (2019) Umjetne neuronske mreže. *Osvježimo znanje*, Kem. Ind 68 (5-6), 219-220

[6] Poljak, S. (2011) *Fourierov red i Fourierova transformacija.* Diplomski rad. Osijek: Sveučilište J.J.Strossmayera u Osijeku

[7] *Introduction Fourier transform and the frequency domain* (2021) URL: https://gyansanchay.csjmu.ac.in/wp-content/uploads/2021/11/introduction-to-fourier-transform-and-frequence-domain.pdf (Pristupljeno 16.9.2023)

[8] Medium. URL: https://medium.com/@gokcenazakyol/what-are-filtering-in-frequency-domain-and-fourier-transform-image-processing-5-6f4cace43c91#:~:text=Low%2Dpass%2C%20Band%2Dpass,through%20is%20called%20the%20passband (Pristupljeno 17.9.2023.)

[9] Geeksforgeeks. URL:https://www.geeksforgeeks.org/frequency-domain-filters-and-its-types/ (Pristupljeno: 17.9.2023.)

[10] Binus. URL:https://socs.binus.ac.id/2017/03/20/image-enhancement-using-frequency-domain-filtering/ (Pristupljeno: 18.9.2023.)

[11] Stipančić, T. (2023) Podloge za predavanje iz kolegija „Vizijski sustavi“. Zagreb, Fakultet strojarstva i brodogradnje

[12] Medium. URL: https://medium.com/computational-photography/lowpass-highpass-bandreject-and-bandpass-filters-in-image-processing-cf4341bfe61b (Pristupljeno: 23.9.2023.)

[13] Khalid Bukhari, S. U., Brad, R., Bala-Zamfirescu, C. (2014) Fast Edge Detection Algorithm fod Embedded Systems. URL: https://sic.ici.ro/wp-content/uploads/2014/06/SIC_2014-2-Art4.pdf (Pristupljeno: 30.9.2023.)

[14] Burger, W., Burge, M. J. (2016) *Digital Image Processing: An Algorithmic Introduction Using* Java. Second Edition. Washington: Springer

[15] Computer vision: image gradients. URL: https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Lectures/amod/AMOD_2020/20201102-03%20-%20ETHZ%20-%20Image%20Gradients.pdf (Pristupljeno: 3.10.2023.)

[16] Medium. URL: https://numustafa.medium.com/basics-of-image-processing-edge-detection-132284152481#:~:text=Spatial%20edge%20detection%20is%20based,operator%2C%20and%20the%20Roberts%20operator (Pristupljeno: 7.10.2023.)

[17] Educative. URL: https://www.educative.io/answers/what-is-canny-edge-detection (Pristupljeno: 8.10.2023.)

[18] Code Project. URL: https://www.codeproject.com/Articles/93642/Canny-Edge-Detection-in-C (Pristupljeno: 14.10.2023.)

[19] Github. URL: https://akshaysin.github.io/fourier_transform.html (Pristupljeno: 15.10.2023.)

[20] Python documentation. URL: https://docs.python.org/3/faq/general.html#general-information (Pristupljeno: 28.10.2023.)

[21] Educba. URL: https://www.educba.com/what-is-numpy/ (Pristupljeno: 28.10.2023.)

[22] OpenCV. URL: https://opencv.org/about/ (Pristupljeno: 28.10.2023.)

[23] Augumented Atartups. URL:https://www.augmentedstartups.com/blog/enhancing-traffic-management-with-computer-vision-applications-and-benefits (Pristupljeno: 9.9.2023.)

[24] Digital Ocean. URL: https://www.digitalocean.com/community/tutorials/how-to-detect-and-extract-faces-from-an-image-with-opencv-and-python (Pristupljeno: 9.9.2023.)

[25] Research Gate. URL: https://www.researchgate.net/figure/Classification-of-image-filters_fig1_328619526 (Pristupljeno: 16.10.2023.)

[26] Yale News. URL: https://news.yale.edu/2023/07/26/dna-analysis-offers-new-insights-diverse-community-machu-picchu (Pristupljeno: 10.11.2023.)

[27] National Geographic Kids. URL: https://kids.nationalgeographic.com/history/article/eiffel-tower (Pristupljeno: 10.11.2023.)

[28] Housing. URL: https://housing.com/news/list-of-iconic-buildings-in-the-world/ (Pristupljeno: 10.11.2023.)

[29] Smithsonian. URL: https://www.smithsonianmag.com/travel/eight-secrets-taj-mahal-180962168/ (Pristupljeno: 10.11.2023.)

[30] Through Eternety Tours. URL: https://www.throughetternity.com/en/blog/travel-tips/cinque-terre-italy-guide-italian-riviera.html (Pristupljeno: 11.11.2023.)

PRILOZI

I. Python kod

- *Custom edge detection* s opcijom učitavanja slike u kod

```python
import cv2
import numpy as np

def custom_edge_detection(image_path):
    # Step 1: Read the image
    original_image = cv2.imread(image_path)

    # Get screen dimensions
    screen_height, screen_width, _ = original_image.shape

    # Resize the original image to have a consistent height
    target_height = 360
    aspect_ratio = target_height / screen_height
    original_image_resized = cv2.resize(original_image, (int(screen_width * aspect_ratio), target_height))

    # Step 2: Convert the original image to grayscale
    grayscale_image = cv2.cvtColor(original_image_resized, cv2.COLOR_BGR2GRAY)

    # Step 3: Apply a custom gradient-based edge enhancement
    sobel_x = cv2.Sobel(grayscale_image, cv2.CV_64F, 1, 0, ksize=3)
    sobel_y = cv2.Sobel(grayscale_image, cv2.CV_64F, 0, 1, ksize=3)
    gradient_magnitude = np.sqrt(sobel_x ** 2 + sobel_y ** 2)
    enhanced_edges = np.clip(gradient_magnitude, 0, 255).astype(np.uint8)

    # Step 4: Apply a custom thresholding with an increased threshold value
    _, thresholded_edges = cv2.threshold(enhanced_edges, 150, 255, cv2.THRESH_BINARY)

    # Resize the final result to have a consistent height
    thresholded_edges_resized = cv2.resize(thresholded_edges, (int(screen_width * aspect_ratio), target_height))

    # Create an empty canvas to arrange images
    canvas = np.zeros((target_height, 2 * int(screen_width * aspect_ratio), 3), dtype=np.uint8)

    # Display the original image in the center
    canvas[:, :int(screen_width * aspect_ratio)] = original_image_resized

    # Display the enhanced edges in the center
    canvas[:, int(screen_width * aspect_ratio):] = cv2.cvtColor(thresholded_edges_resized, cv2.COLOR_GRAY2BGR)

    # Display the images
    cv2.imshow('Custom Edge Detection', canvas)

    # Wait for a key press and close the window
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Example usage
image_path = 'Slika4.jpg'
custom_edge_detection(image_path)
```

- *Custom edge detection* s opcijom filtriranja u stvarnom vremenu

```
import cv2
import numpy as np

def custom_edge_detection_real_time():
    # Open the camera
    cap = cv2.VideoCapture(0)

    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()

        # Get screen dimensions
        screen_height, screen_width, _ = frame.shape

        # Resize the frame for consistent processing
        target_height = 360
        aspect_ratio = target_height / screen_height
        frame_resized = cv2.resize(frame, (int(screen_width * aspect_ratio), target_height))

        # Convert the frame to grayscale
        grayscale_frame = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2GRAY)

        # Apply custom gradient-based edge enhancement
        sobel_x = cv2.Sobel(grayscale_frame, cv2.CV_64F, 1, 0, ksize=3)
        sobel_y = cv2.Sobel(grayscale_frame, cv2.CV_64F, 0, 1, ksize=3)
        gradient_magnitude = np.sqrt(sobel_x ** 2 + sobel_y ** 2)
        enhanced_edges = np.clip(gradient_magnitude, 0, 255).astype(np.uint8)

        # Apply custom thresholding with an increased threshold value
        _, thresholded_edges = cv2.threshold(enhanced_edges, 100, 255, cv2.THRESH_BINARY)

        # Resize the final result for display
        thresholded_edges_resized = cv2.resize(thresholded_edges, (int(screen_width * aspect_ratio), target_height))

        # Create an empty canvas to arrange images
        canvas = np.zeros((target_height, 2 * int(screen_width * aspect_ratio), 3), dtype=np.uint8)

        # Display the original frame in the center
        canvas[:, :int(screen_width * aspect_ratio)] = frame_resized

        # Display the enhanced edges in the center
        canvas[:, int(screen_width * aspect_ratio):] = cv2.cvtColor(thresholded_edges_resized, cv2.COLOR_GRAY2BGR)

        # Display the images
        cv2.imshow('Custom Edge Detection - Real Time', canvas)

        # Break the loop if 'q' key is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release the camera and close the window
    cap.release()
    cv2.destroyAllWindows()

# Example usage
custom_edge_detection_real_time()
```