Detekcija i lokalizacija kontura

Bubnjar, Veronika

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Veronika Bubnjar

Zagreb, 2024. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori: Student:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing. Veronika Bubnjar

Zagreb, 2024. godina.

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

Zahvaljujem se mentoru doc. dr. sc. Tomislavu Stipančiću na pomoći i savjetima tijekom izrade ovog rada.

Također se zahvaljujem svojoj obitelji i prijateljima bez kojih ništa od ovog ne bi bilo moguće.

Veronika Bubnjar

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 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Veronika Bubnjar** JMBAG: 0035217917

Naslov rada na hrvatskom jeziku: **Detekcija i lokalizacija kontura**

Naslov rada na engleskom jeziku: **Contour detection and localization**

Opis zadatka:

Analiza kontura (rubova) objekata na slikama važan je početni korak u računalnom i strojnom vidu. Između ostaloga, konture kao granice objekata predstavljaju temelj za pronalaženje, lokalizaciju, segmentaciju i praćenje objekata. Konture su nositelji informacija jer predstavljaju promijene.

U radu je potrebno:

- objasniti značenje kontura i rubova kao nositelja informacija na slikama
- odrediti i objasniti korake koji vode ka računalnom prepoznavanju kontura objektima na slici
- implementirati korake za prepoznavanje kontura u računalni program
- usporediti više metoda za detekciju i lokalizaciju kontura
- analizirati dobre i loše strane odabranih metoda.

Razvijena programska rješenja je potrebno temeljiti na OpenCV biblioteci implementiranoj kroz Python programski jezik. Razvijene modele je potrebno eksperimentalno evaluirati koristeći vizijski sustav u sklopu Laboratorija za projektiranje izradbenih i montažnih sustava.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
9. svibnja 2024.	11. srpnja 2024.	15. – 19. srpnja 2024.
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
G	-	Gaussova funkcija
σ	-	Standardna devijacija
x	-	Os apscisa
y	-	Os ordinata
f	-	Vektor gradijenta
M	-	Magnituda gradijenta
α	°	Kut vektora gradijenta

SAŽETAK

U diplomskom radu objašnjeni su i primijenjeni koraci algoritama za detekciju rubova i kontura na slikama. Objašnjena je važnost detekcije kontura kao jednog od bitnih segmenata računalnog vida. Detaljnije su objašnjeni modeli detekcije kao što su Canny, Sobel, Roberts i Prewitt. Implementacijom koraka kroz odabrane modele se uspoređuju i analiziraju dobiveni rezultati. Razvijena programska rješenja su temeljena na OpenCV biblioteci implementirani kroz Python programski jezik

Ključne riječi:

Računalni vid, detekcija rubova, detekcija kontura, Canny, Sobel, Roberts, Prewitt

SUMMARY

In this thesis, the steps of algorithms for edge and contour detection in images are explained and applied. The importance of contour detection as one of the crucial segments of computer vision is discussed. Detection models such as Canny, Sobel, Roberts, and Prewitt are explained in detail. By implementing the steps through the chosen models, the obtained results are compared and analyzed. The entire process is carried out in Python using OpenCV.

Key words:

Computer vision, edge detection, contour detection, Canny, Sobel, Roberts, Prewitt

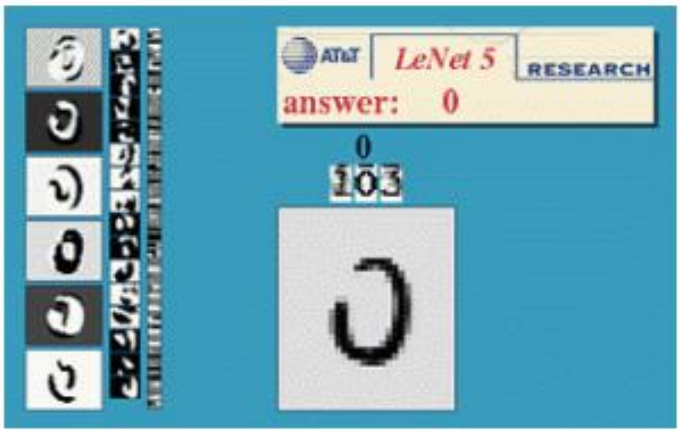
1. UVOD

Detekcija rubova i kontura je jedan od ključnih koraka u sklopu područja računalnog vida u značajnom razvoju umjetne inteligencije. Područje koje se bavi analizom i obradom vizualnih podataka pomoću računala. Kao što ljudi imaju mogućnost percepcije osjetilom vida, ideja je omogućiti računalu sposobnost. Nešto što ljudima dolazi kao prirodno, razumijevanje slika koje formiraju smišljenu informaciju, za računalni vid predstavlja puno kompleksniji problem.

Kao odgovori na zahtjeve funkcioniranja računalnog vida, razvijaju se matematičke tehnike za pronalaženje, lokalizaciju, segmentaciju i praćenje objekta na slikama.

Računalni vid se koristi u širokom spektru primjene, koje uključuju:

- Optičko prepoznavanje znakova: čitanje rukom pisanih poštanskih brojeva na pismima (Slika 1.) i automatsko prepoznavanje registarskih pločica;



Slika 1. Prepoznavanje pisama [1]

- Strojna inspekcija: brza inspekcija dijelova za osiguranje kvalitete korištenjem stereo vida sa specijaliziranim osvjetljenjem za mjerenje tolerancija na krilima aviona ili dijelovima karoserije automobila (Slika 2.) ili traženje nedostataka u čeličnim odljevcima koristeći rendgenski vid;

Slika 2. Vizijski sustav za inspekciju kontrole kvalitete [1]

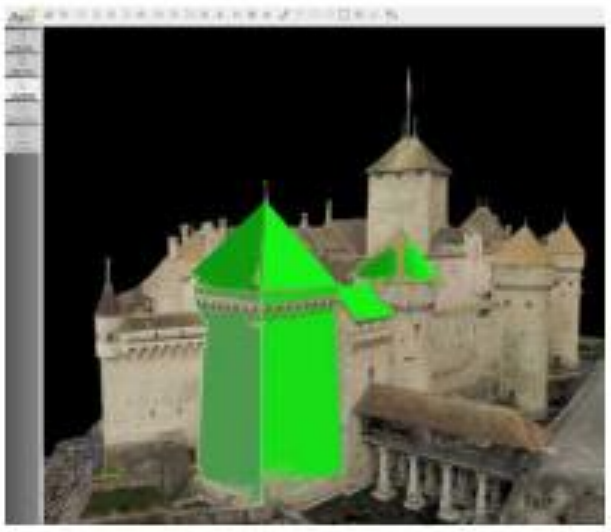
- Maloprodaja: prepoznavanje objekata za automatizirane blagajne i potpuno automatizirane trgovine;
- Logistika skladišta: autonomna dostava paketa i paleta i odabir dijelova pomoću robotskih manipulatora (Slika 3);

Slika 3. Robotski manipulator u industriji [1]

- Medicinske slike: registracija preoperativnih i intraoperativnih slika ili provođenje dugoročnih studija morfologije mozga ljudi kako stare;
- Autonomna vozila: sposobna za vožnju od točke do točke između gradova (slika 4.), ali autonomni letovi;

Slika 4. Autonomno vozilo [1]

- Izrada 3D modela (fotogrametrija): potpuno automatizirana izrada 3D modela iz zračnih i fotografija dronova (Slika 5.);



Slika 5. 3D model slike snimljene iz zraka [1]

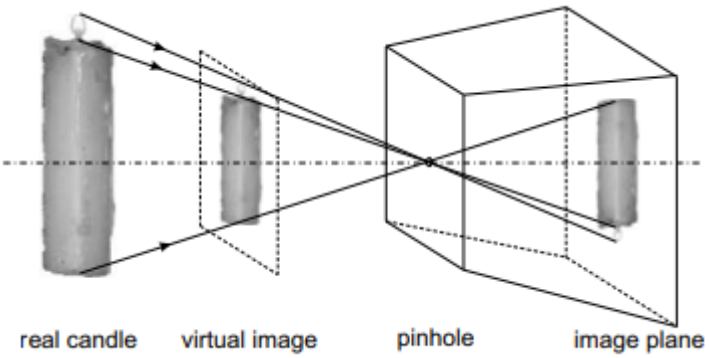
- *Match move*: spajanje računalno generirane slike (tzv. CGI) s live akcijskim snimkama praćenjem značajki u izvornom videu kako bi se procijenilo 3D kretanje kamere i oblik okoline. Takve tehnike se široko koriste u Hollywoodu, svima dostupno primijetiti u poznatim filmovima. [1]

1.1. Računalni vid

Računalnim vidom se pokušavaju opisati slike i rekonstruirati svojstva, kao što su oblik, osvjetljenje i raspodjele boja. No postoje faktori koji otežavaju ovaj problem.

Ovo pitanje pruža uvid u složeni krajolik računalnog vida. Može se odgovoriti na mnogo načina.

Gubitak informacija u prijelazu 3D → 2D je fenomen koji se događa u tipičnim uređajima za snimanje slike kao što su kamera ili oko. Njihova geometrijska svojstva stoljećima su bila aproksimirana modelom rupice (kutija s malom rupom u njoj—'camera obscura' na latinskom). Ovaj fizički model odgovara matematičkom modelu perspektivne projekcije; na slici 6. je prikazan princip. Projektivna transformacija mapira točke duž zraka, ali ne očuva kutove i kolinearnost.



Slika 6. Model rupice [2]

Glavni problem s modelom koji ima jedan dostupan pogled jest da projektivna transformacija vidi mali objekt blizu kamere na isti način kao veliki objekt udaljen od kamere. U ovom slučaju, čovjeku je potrebno 'mjerilo' za procjenu stvarne veličine objekta koju računalo nema. [2]

Tumačenje slike je problem koji ljudi nesvjesno rješavaju i to je glavni alat računalnog vida. Kada čovjek pokušava razumjeti sliku, tada se prethodno znanje i iskustvo primjenjuju na trenutnu opažanje. Ljudska sposobnost razmišljanja omogućuje predstavljanje dugogodišnjeg znanja i njegovu uporabu za rješavanje novih problema. Umjetna inteligencija radi desetljećima kako bi računalima dala sposobnost razumijevanja opažanja; iako je napredak ogroman, praktična sposobnost stroja da razumije opažanja ostaje vrlo ograničena.

S matematičko-logičkog i/ili lingvističkog gledišta, tumačenje slike može se smatrati preslikavanjem: interpretacija: podaci slike → model.

Logički model znači neki specifičan svijet u kojem opažani objekti imaju smisla. Primjeri mogu biti jezgre stanica u biološkom uzorku, rijeke na satelitskoj slici ili dijelovi u industrijskom procesu koji se provjeravaju na kvalitetu. Isto tako može postojati nekoliko tumačenja iste slike. Uvođenje interpretacije u računalni vid omogućuje nam korištenje koncepata iz matematičke logike, lingvistike kao sintakse (pravila koja opisuju ispravno oblikovane izraze) i semantike (studija značenja). Smatrajući opažanja kao primjer formalnih izraza, semantika proučava

odnose između izraza i njihovih značenja. Tumačenje slike u računalnom vidu može se razumjeti kao primjer semantike. [3]

Razvijanjem algoritma za razumijevanje slike, može se specificirati područje za koje je vezana slika, odnosno informacije koje nosi te se analiza može koristiti za složenije probleme.

Šum je inherentno prisutan u svakom mjerenju u stvarnom svijetu. Njegovo postojanje zahtijeva matematičke alate koji se mogu nositi s nesigurnošću; primjer je teorija vjerojatnosti. Otklanjanje šuma jedan je od ključnih koraka prilikom primjene računalnog modela za detekciju rubova kako bi se detektirale konture.

Previše podataka. Slike su velike, a video—sve češće predmet primjena vida—odgovarajuće veći. Tehnički napreci čine zahtjeve za procesorom i memorijom mnogo manje problematičnim nego što su nekad bili, i mnogo se može postići s proizvodima na razini potrošača. Ipak, učinkovitost u rješavanju problema i dalje je važna i mnoge primjene još uvijek nisu u stanju ostvariti izvedbu u stvarnom vremenu.

Svjetlina ovisi o zračenju (vrsta izvora svjetlosti, intenzitet i položaj), položaju promatrača, lokalnoj geometriji površine i svojstvima refleksije površine. Ona otežava lokalizaciju i segmentaciju zbog oscilacija u intenzitetu svjetline na slikama. Iz tog razloga, fizika snimanja slika obično se izbjegava u praktičnim pokušajima razumijevanja slika. Umjesto toga, traži se izravna veza između izgleda objekata u scenama i njihove interpretacije. [1]

Lokalni prozor vs. potreba za globalnim pogledom, odnosno razumijevanje „šire slike“. Uobičajeno, algoritmi za analizu slika analiziraju određeni spremnik u operativnoj memoriji (npr. piksel na slici) i njegovo lokalno susjedstvo; računalo vidi sliku kroz ključanicu; to čini vrlo teškim razumijevanje globalnog konteksta.

Segmentacija slika jedan je od najvažnijih koraka u analizi obrađenih podataka slika. Njen glavni cilj je podijeliti sliku na dijelove koji imaju jaku korelaciju s objektima ili područjima stvarnog svijeta prisutnima na slici. Možemo težiti potpunoj segmentaciji, što rezultira skupom disjunktnih područja koja jedinstveno odgovaraju objektima u ulaznoj slici, ili djelomičnoj segmentaciji, gdje područja ne odgovaraju izravno objektima na slici. Kako bi ona uopće bila moguća, razvijaju se modeli za ključni korak u segmentaciji, a to su modeli za detekciju rubova. [3]

2. DETEKCIJA RUBOVA I KONTURA

Postoji velika grupa metoda koje vrše segmentaciju temeljenu na informacijama o rubovima u slici; to je jedan od najstarijih pristupa segmentaciji i dalje je vrlo važan. Segmentacija temeljena na rubovima oslanja se na operatore za detekciju rubova. Rubovi označavaju lokacije u slici gdje postoje diskontinuiteti u nijansama sive boje, boji, teksturi, itd.. Računalni program pomoću označenih rubova konturira objekte na slici.

Neke od metoda pronalaženja rubova te određivanja kontura su Canny, Sobel, Roberts i Prewitt koji će se detaljnije objasniti u slijedećim poglavljima.

2.1. Modeli rubova

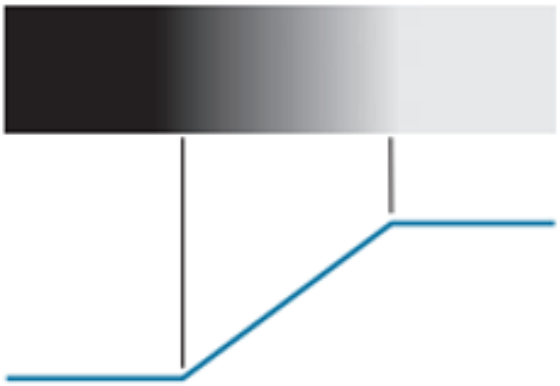
Modeli rubova klasificiraju se prema njihovim profilima intenziteta. *Step edge* (doslovni prijevod je „rub koraka“) karakterizira prijelaz između dvije razine intenziteta na slici koja je sivo skalirana (*gray scale* – ono što zovemo crno-bijeli efekt) koji se idealno događa preko udaljenosti jednog piksela.

Slika 7. prikazuje dio vertikalnog koraka ruba i horizontalni profil intenziteta kroz rub.

Slika 7. Prijelaz područja različitih intenziteta – *Step edge* [4]

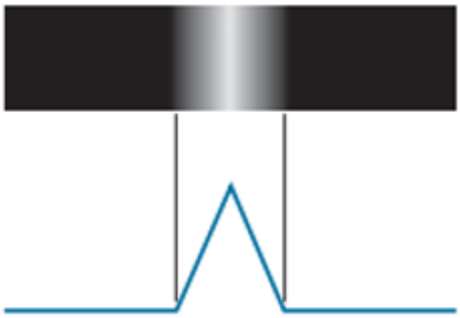
Step edges se pojavljuju, na primjer, u slikama generiranim računalom za upotrebu u područjima poput modeliranja čvrstih tijela i animacije. Ovakvi tipovi tzv. idealni rubovi mogu se pojaviti preko udaljenosti jednog piksela, pod uvjetom da se ne koristi dodatna obrada (poput zaglađivanja - *blur*) kako bi izgledali „stvarno“. Digitalni *step edge* modeli se koriste kao modeli rubova u razvoju algoritama. Na primjer, Cannyjev algoritam za detekciju rubova, koji će biti detaljnije objašnjen, izvorno je izveden koristeći ovaj model.

U praksi, digitalne slike imaju rubove koji su zamagljeni i šumoviti, pri čemu stupanj zamagljenosti prvenstveno određuju ograničenja u mehanizmu fokusiranja (npr. leće u slučaju optičkih slika), a razinu šuma prvenstveno određuju elektroničke komponente sustava za snimanje. U takvim situacijama, rubovi su više usko modelirani kao da imaju profil intenziteta rampe, poput ruba na slici 8. tzv. *ramp edge.*



Slika 8. Prijelaz područja različitih intenziteta – *ramp edge* [4]

Nagib rampe je obrnuto proporcionalan stupnju do kojeg je rub zamagljen. U ovom modelu više nemamo jednu „točku ruba“ duž profila. Umjesto toga, točka ruba sada je bilo koja točka sadržana u rampi, a segment ruba bio bi skup takvih povezanih točaka.



Slika 9. Prijelaz između dvije razine intenziteta – *roof edge* [4]

Treći tip ruba je takozvani *roof edge* (krovni rub), koji ima karakteristike prikazane na slici 9.. *Roof edges* su modeli linija kroz područje, pri čemu je baza (širina) ruba određena debljinom i oštrinom linije. U krajnjem slučaju, kada je baza široka jedan piksel, krovni rub je ništa drugo nego linija debela jedan piksel koja prolazi kroz područje na slici. Krovni rubovi se pojavljuju,

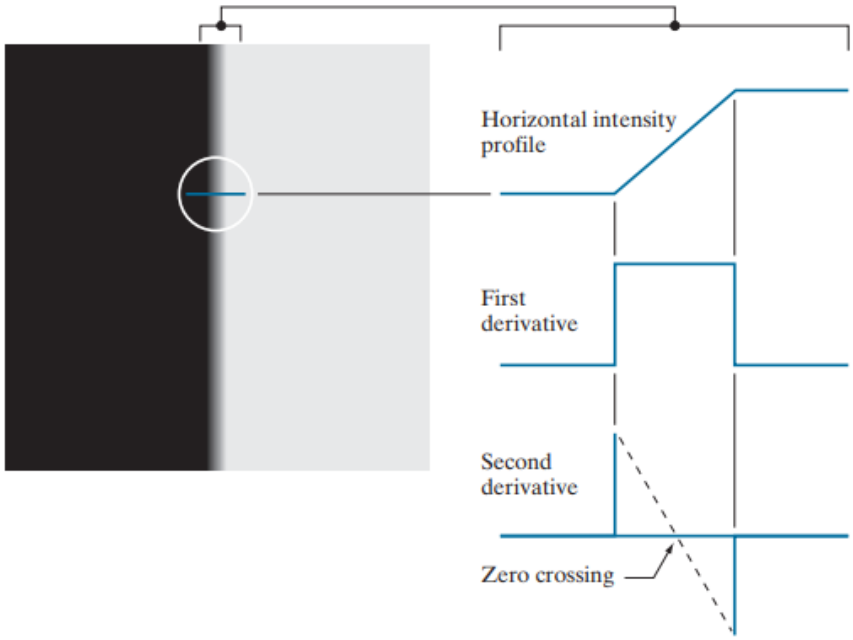
na primjer, kada su tanki objekti (poput cijevi) bliže senzoru od pozadine (poput zidova). Cijevi izgledaju svjetlije i tako stvaraju sliku sličnu modelu na slici 9.

Ostala područja u kojima se krovni rubovi rutinski pojavljuju su digitalizacija crteža i satelitske slike, gdje se tanke značajke, poput cesta, mogu modelirati ovim tipom ruba.

Uobičajeno je pronaći slike koje sadrže sve tri vrste rubova. Ono što nam modeli na slici prethode tri slike omogućuju je pisanje matematičkih izraza za rubove u razvoju algoritama za obradu slika. Učinak ovih algoritama ovisit će o razlikama između stvarnih rubova i modela korištenih u razvoju algoritama. [4]

Slika 10. prikazuje sliku iz koje detalj prijelaza izdvojen desno na istoj. Slika prikazuje horizontalni intenzitetski profil. Prikazuje prve i druge derivacije intenzitetskog profila. Slijedeći profil s lijeva na desno, može se uočiti da je prva derivacija pozitivna na početku prijelaza i na točkama prijelaza, te je nula u područjima konstantnog intenziteta. Druga derivacija je pozitivna na početku prijelaza, negativna na kraju prijelaza, nula na točkama prijelaza i nula na točkama konstantnog intenziteta. Opisane derivacije bile bi obrnute za rub koji prelazi iz svijetlog u tamno. Sjecište između osi nultog intenziteta i linije koja se proteže između ekstrema drugog derivata označava točku zvanu nulti prijelaz druge derivacije.

Svrha dobivenih derivacija je ta da veličine prve derivacije mogu biti korištene za određivanje prisutnosti ruba na točki u slici, a veličina druge derivacije, nalazi li se piksel ruba na toj točki.



Slika 10. Horizontalni profil intenziteta s detaljem [4]

2.2. Koraci u primjeni algoritma

U sažetku, najčešći koraci koji se tipično izvode za detekciju rubova u računalnom programu su:

1. *Gray scale* kao što je prije spomenuto je primjena crno-bijelog efekta. Detekcija rubova u sivim slikama omogućava algoritmima da se usmjere isključivo na promjene u intenzitetu svjetlosti, što pojednostavljuje proces i smanjuje složenost koja bi bila prisutna u analizi slike u boji.
2. Zaglađivanje slike radi smanjenja šuma. Potrebu za ovim korakom rješava primjena *Gaussovog* filtera.
3. Izračunavanje gradijenta slike – način izračuna ovisi o korištenom modelu
4. *Tresholding (pragovanje)* uklanja slabe rubove koji nisu povezani s jakim rubovima i osigurava da svi relevantni rubovi budu povezani, čime se dobivaju kontinuirane rubne konture.

2.3. Gaussov filter

Primjena Gaussovog filtra je postupak u kojem se slika zaglađuje kako bi se smanjio šum i neželjene detalje prije daljnje obrade. 2D Gaussov operator zaglađivanja G(*x*, *y*) (također nazvan Gaussov filtar, ili jednostavno Gauss, dan je formulom:

$$G(x,y) = e^{-(x^2+y^2)/2\sigma^2} \tag{1.}$$

gdje su x, y koordinate slike, a σ je standardna devijacija pripadajuće distribucije vjerojatnosti.

Nekad se koristi i slijedeća formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.}$$

ili

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x^2+y^2)/2\sigma^2} \tag{3.}$$

Standardna devijacija σ je jedini parametar Gaussovog filtra—ona je proporcionalna veličini susjednih piksela na kojem filtar djeluje. Pikseli udaljeniji od centra operatora imaju manji utjecaj, a pikseli udaljeni više od 3σ od centra imaju zanemariv utjecaj.

Ono što se zapravo događa jest generiranje obično kvadratne matrice koja prolazi kroz svaku točku slike. Za svaku točku, nova vrijednost se izračunava kao prosjek vrijednosti susjednih piksela. Što je veća matrica, to je veće zaglađivanje. Primjena Gaussovog filtra rezultira zaglađenom slikom. Ova zaglađena slika ima smanjeni šum i manje detalja dok zadržava ključne informacije o slici, što olakšava daljnje korake u obradi slike, kao što je detekcija rubova.

2.4. Izračunavanje gradijenta

Gradijent na slici predstavlja vektor koji označava smjer najbržeg porasta intenziteta svjetline ili boje u svakoj točki slike. Gradijent se često koristi u računalnom vidu za detekciju rubova i kontura, gdje može ukazati na mjesta naglih promjena u intenzitetu, što su često indikacije prisutnosti objekata ili rubova na slici.

Odabrani alat za pronalaženje jačine i smjera ruba na proizvoljnoj ravnini *x* i ravnini(x,y) slike, f je gradijent, označen sa ∇f i definiran kao vektor.

$$\nabla f(x,y) \equiv \text{grad}\,[f(x,y)] \equiv \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{4.}$$

U ovom koraku nastupa razlika primijenjenih modela. Za početak ***Robertsov*** model koji se temelji na implementaciji dijagonalnih razlika. Kvadratne matrice koje su derivacije se implementiraju i filtriraju sliku u smjeru osi *x*

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

i smjeru osi *y*

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Matrice veličine 2×2 su jednostavne konceptualno, ali nisu toliko korisne za izračun smjera ruba kao matrice koje su simetrične oko svojih središta, najmanjeg reda veličine 3×3. Ovakve matrice uzimaju u obzir prirodu podataka s obiju strana središnjeg piksela te tako nose više informacija o smjeru ruba.

U ovoj takvim matricama, razlika između prvog i trećeg retka aproksimira derivaciju u smjeru x, dok razlika između prvog i trećeg stupca aproksimira derivaciju u smjeru y. Intuitivno, ovakve aproksimacije će biti preciznije od aproksimacija dobivenih koristeći Robertsov model a. Ove matrice su karakteristične za ***Prewittov*** model a u smjeru *x* osi glasi:

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

I u smjeru *y* osi

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

Blaga varijacija Prewittovog modela je kada se za srednji koeficijent uzme vrijednost 2. Mijenjanjem tog koeficijenta se zaglađuje slika. Takve matrice su karakteristične za ***Sobelov*** model.

Os *x*

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

I os *y*

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewittove matrice su jednostavnije za implementaciju od Sobelovih, ali mala razlika u računanju između njih obično nije problem. Činjenica da Sobelove matrice imaju bolje karakteristike suzbijanja šuma (zaglađivanja) čini ih poželjnijim jer, kao što je spomenuto ranije, uklanjanje šuma je važno u primjeni druge derivacije.

Bilo koje od parova matrica, odnosno bilo koji od navedenih modela može se konvoluirati sa slikom kako bi se dobili komponente gradijenta g_x i g_y na svakoj lokaciji piksela. Ova dva polja parcijalnih derivacija zatim se koriste za procjenu jačine i smjera ruba.

Dobivanje magnitude gradijenta zahtijeva izračune prema jednadžbi:

$$M(x_1 y) = \left\|\nabla_f(x,y)\right\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)} \qquad (5.)$$

Ova implementacija nije uvijek poželjna zbog računanja kvadrata i korijena, pa se često koristi pristup približne magnitude gradijenta korištenjem apsolutnih vrijednosti:

$$M(x,y) \approx |g_x| + |g_y| \tag{6.}$$

Ova jednadžba je privlačnija računalno zbog jednostavnosti, ali i dalje očuva relativne promjene u razinama intenziteta.

Primjena pojedinih modela vidljiva je u slijedećim poglavljima gdje je i detaljnije objašnjen ***Canny edge*** model koji je napredniji od prethodna tri.

3. IMPLEMENTACIJA KORAKA ODABRANIH METODA U RAČUNALNI PROGRAM

Za implementaciju koraka i analizu odabrana je uslikana slika 11., a korišten je programski jezik Python i biblioteka OpenCV koja već sadrži odabrane modele koje je bilo potrebno aktivirati razvijanjem koda. Korišteni modeli uključuju Robertsov, Sobelov, Prewittov i Cannyjev algoritam.

Slika 11. Slika korištena za implementaciju koraka

3.1. Robertsov model

Robertsov model je jedan od prvih pristupa detekciji rubova u digitalnim slikama. U prethodnom poglavlju je spomenut zbog korištenja kvadratne matrice 2x2. Korištenje manjeg reda matrice rezultira jednostavnom i efikasnom primjenom.

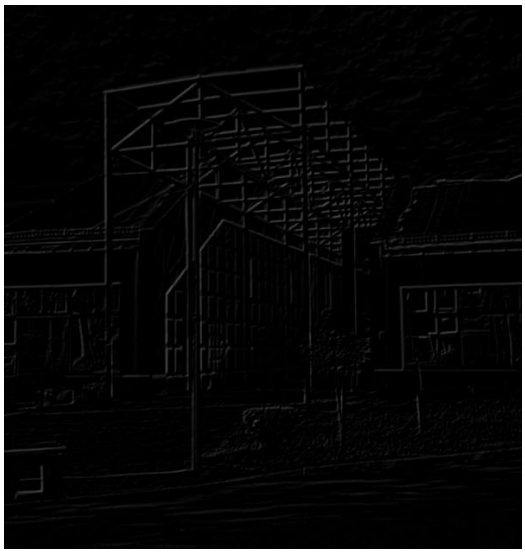
Kao prvi korak računalnog modela, za odabranu sliku, algoritam provodi sivo skaliranje koje je prikazano na slici 12. kako bi se pojednostavnila analiza.

Slika 12. Sivo skaliranje

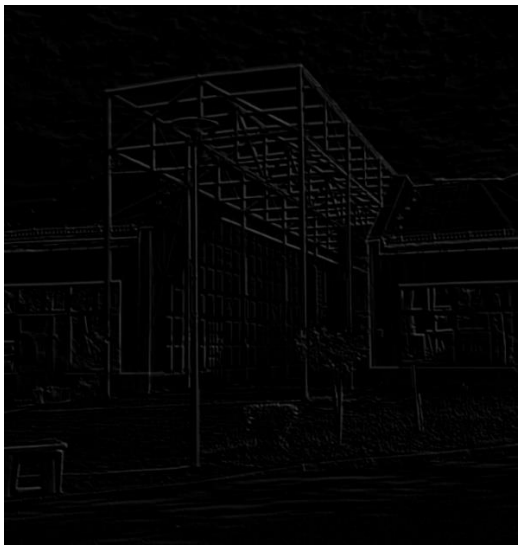
Slijedeći korak je implementacija Gaussovog filtera za redukciju šuma što rezultira zaglađenosti prikazanoj na slici 13.

Slika 13. Primjena Gaussovog filtera

Slijedeći korak je izračun gradijenta po osima te konačni zbroj. Na slici 14. je prikazana primjene matrice 2x2 po osi *x*, na slici 15. po osi *y*



Slika 14. Konvolucija matrice 2x2 po osi *x*

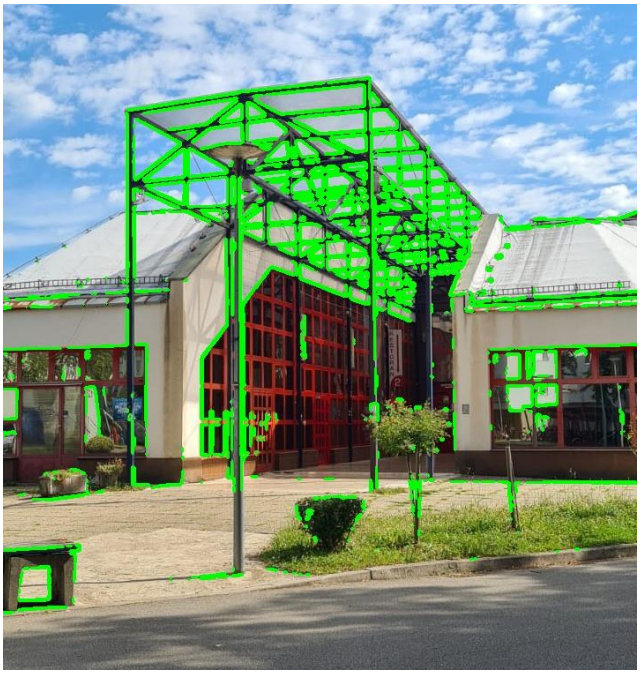


Slika 15. Konvolucija matrice 2x2 po osi *y*

Konačan rezultat magnitude gradijenta se dobije jednadžbom (6.) i rezultat je prikazan na slici 16.

Slika 16. Magnituda gradijenta (Roberts)

Konačan rezultat je prikazan na slici 17. gdje su zelenom linijom detektirane konture na slici. Golim okom se može primijetiti da nisu konturirani svi rubovi, već samo središnji dio slike gdje su nagli prijelazi intenziteta svjetline.



Slika 17. Konačni rezultat (Roberts)

3.2. Prewittov model

U Prewittovom modelu se algoritam ponaša na isti način, dakle započinje sa sivim skaliranjem, primjenom Gaussovog filtera, ali kod izračuna gradijenta se koristi kvadratna matrica 3x3 spomenuta u prethodnom poglavlju te rezultat izgleda kao prikazan na slici 18.

Slika 18. Magnituda gradijenta (Prewitt)

Konačan rezultat je prikazan na slici 19., a ono što se može primijetiti jest da je više kontura označeno, ali mnoge ne daju smišljenu cjelinu niti objekt.

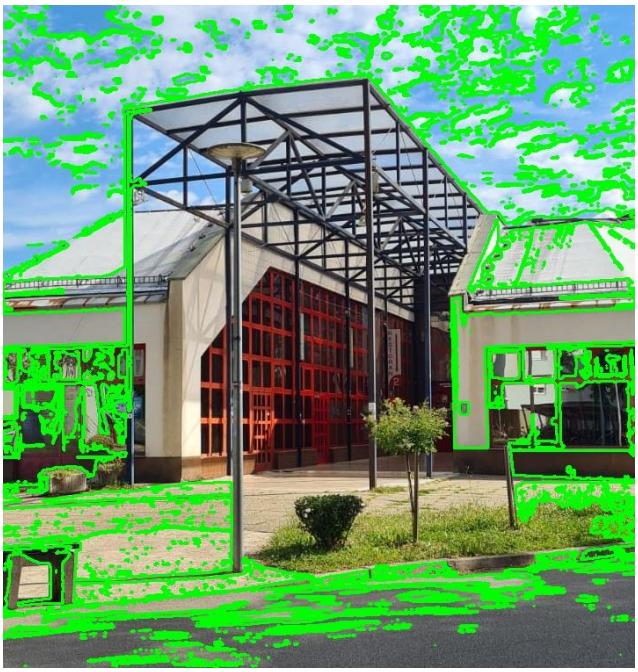
Slika 19. Konačni rezultat (Prewitt)

3.3. Sobelov model

Kao i prethodni algoritmi, koraci do računanje gradijenta su identični pa time i slike koraka. Primjenom Sobelovog modela koji konvolucijom matrice 3x3 daje i zaglađenost slike, magnituda gradijenta je prikazana na slici 20.

Slika 20. Magnituda gradijenta (Sobel)

Konačni rezultat modela prikazan je na slici 21. te se može primijetiti da su neke konture detektirane, ali središnji dio slike nije konturiran gdje su najjasniji prijelazi intenziteta svjetline.



Slika 21. Konačni rezultat (Sobel)

3.4. Canny edge model

Prethodna tri algoritma su bila relativno jednake složenosti s razlikom u koraku računanje gradijenta. Canny edge detektor se razliku po svojoj pojačanoj složenosti algoritma. Cannyjev pristup temelji se na tri osnovna cilja:

1. Niska stopa greške – svi rubovi trebaju biti pronađeni, ali ne bi trebalo biti lažnih detekcija.
2. Dobra lokalizacija rubova – pronađeni rubovi moraju biti što bliže stvarnim rubovima. To znači da bi udaljenost između točke označene kao rub detektora i središta stvarnog ruba trebala biti minimalna.
3. Jedna točka ruba kao odgovor – detektor bi trebao vratiti samo jednu točku za svaku stvarnu točku ruba. Drugim riječima, broj lokalnih maksimuma oko stvarnog ruba trebao bi biti minimalan. To znači da detektor ne bi trebao identificirati višestruke rubne piksele gdje postoji samo jedna rubna točka.

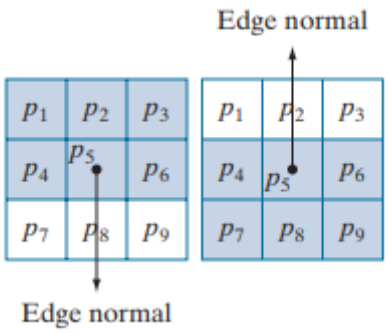
Srž Cannyjevog rada bila je u matematičkom izražavanju prethodno navedenih triju kriterija, a zatim u pokušaju pronalaska optimalnih rješenja za te formulacije. Općenito, teško je (ili je nemoguće) pronaći zatvoreni oblik rješenja koji zadovoljava sve prethodno navedene ciljeve. Međutim, korištenjem numeričke optimizacije s 1-D korakom ruba (*step edge*) koji je

„zagađen“ bijelim šumom. Bijeli šum je specifični tip šuma kojeg karakterizira prisutnost svih frekvencija s jednakom aplitudom. Došlo se do zaključka da je dobra aproksimacija optimalnog detektora koraka ruba jednaka derivaciji Gaussove funkcije koja glasi:

$$\frac{d}{dx} \cdot e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2} e^{-\frac{x^2}{2\sigma}} \tag{7.}$$

Općenito, generalizacija prethodnog rezultata na 2-D prostor uključuje prepoznavanje da se 1-D pristup i dalje primjenjuje u smjeru normalnom na rub. Budući da smjer normale nije poznat unaprijed, to bi zahtijevalo primjenu 1-D detektora ruba u svim mogućim smjerovima. Ovaj zadatak može se približno riješiti tako da se prvo izgladi slika kružnom 2-D Gaussovom funkcijom, izračuna gradijent rezultata te se zatim koristi magnituda i smjer gradijenta za procjenu jačine i smjera ruba na svakoj točki.

Slika gradijenta obično pokazuje široke grebene oko lokalnih maksimuma. Sljedeći korak je sužavanje tih grebena. Jedan od pristupa je korištenje tehnike nazvane **supresija ne-maksimuma**. Bit te tehnike je specificiranje nekoliko diskretnih orijentacija normalnih vektora ruba (gradijenta). Na primjer, u matrici 3x3 možemo definirati četiri orijentacije ruba koje prolaze kroz središnju točku matrice: horizontalnu i vertikalnu kao što je prikazano na slici 22.

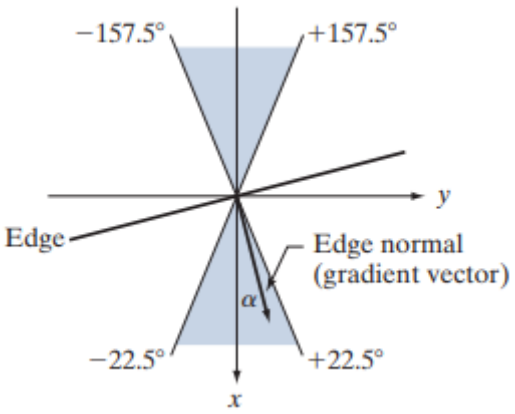


Slika 22. Smjer vektora ruba (normala) [5]

Prikazuje se situacija za dvije moguće orijentacije horizontalnog ruba. Smjer ruba se određuje jednadžbom:

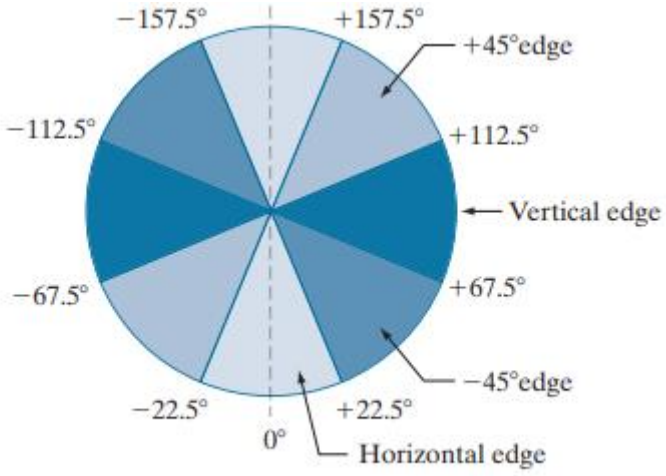
$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right] \tag{8.}$$

Ako je normala ruba u rasponu kutova od -22,5° do 22,5° ili od -157,5° do 157,5°, onda je to horizontalni rub, kao što je vidljivo na slici 23.



Slika 23. Raspon kutova za horizontalni rub [5]

Slika 24. prikazuje kutne raspone koji odgovaraju četiri razmatrane smjernice. Neka d_1, d_2, d_3, d_4, označavaju četiri osnovne smjernice rubova koji su razmatrani za matricu 3x3: horizontalnu, -45°, vertikalnu i +45°.



Slika 24. Raspon kutova za određivanje smjera [5]

Formulira se sljedeća shema algoritma za implementaciju koraka ne-maksimuma za matricu 3x3 centriranu u proizvoljnoj točki (x, y) u α glasi:

1. Pronađi smjer d_k, koji je najbliži vrijednosti α (x, y).
2. Neka K označava vrijednost $\|\nabla f_s\|$ od (x, y). Ako je K manji od vrijednosti $\|\nabla f_s\|$ kod jednog ili oba susjeda točke (x, y) duž smjera ruba d_k, neka je $g_N(x, y) = 0$ (potiskivanje); inače, neka je $g_N(x, y) = K$.

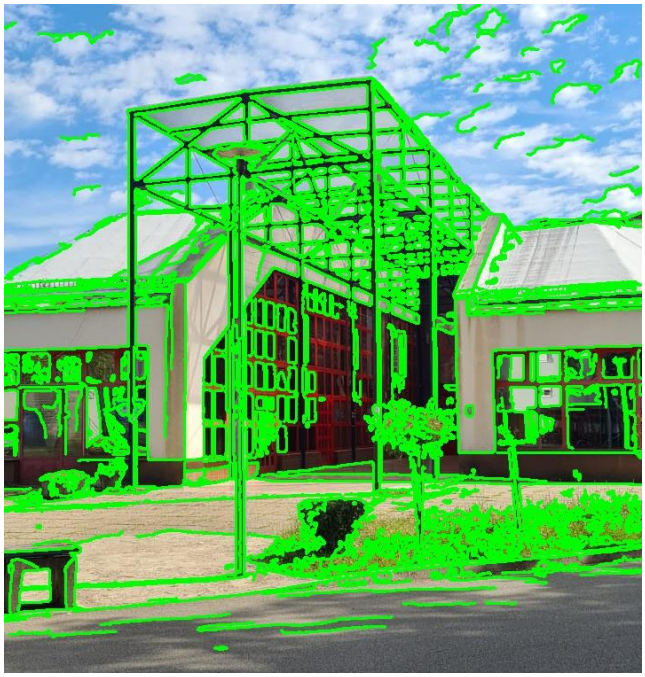
Ponovljeno za sve vrijednosti x i y, ovaj postupak rezultira slikom $g_N(x, y)$ potisnutog ne-maksimuma koja je kao $f_s(x,y)$.

Konačni korak je pragiranje (*tresholding*) gdje se koristi zaostalo pragiranje (*hysteresis tresholding*) koje je eksperimentalno dokazano Canny metodom da je najbolje postignuto ako

se u korištenju visokog i niskog praga postave omjeri u rasponu od 2:1 do 3:1. Problematika pragiranja je ako se prag postavi prenisko, bit će lažnih rubova, a ako se postavi previsoko, valjani rubovi će biti zanemareni. [5]

Slika 25. Magnituda gradijenta (Canny edge)

Konačni rezultat je prikazan na slici 26. i mogu se primijetiti konture detaljnije određenih objekta.

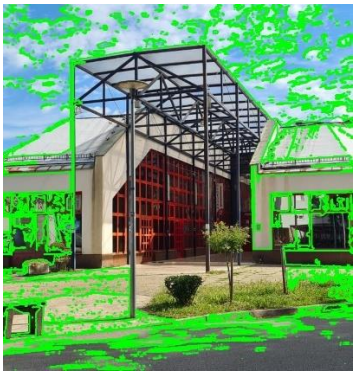
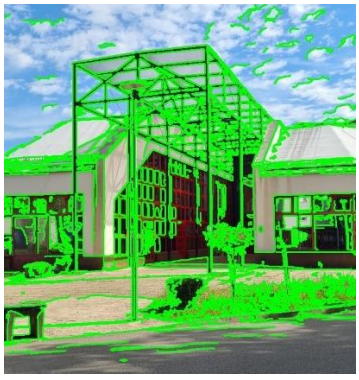


Slika 26. Konačni rezultat (Canny edge)

4. USPOREDBA METODA

Usporedbom rezultata sva četiri modela je vidljiva u tablici 1.

Tablica 1. Usporedba rezultata modela

 **Roberts**	 **Prewitt**
 **Sobel**	 **Canny edge**

S obzirom da je konvolucija Robertsovim modelom napravljena s kvadratnom matricom 2x2, golim okom je vidljiva najmanja detekcija kontura. Ako usporedimo Prewittov i Sobelov model kod kojih se koristi kvadratna matrica 3x3 s razlikom kod Sobela gdje matrica dodatno zaglađuje uklanjajući šumove. Iako su dva modela slična, računanje je tek ponešto kompleksnije kod Sobela, ali u ovom slučaju nije dobiven bolji rezultat. Može se primijetiti kako je središnji dio u potpunosti lišen kontura.

Vidno je kako su se Canny edge metodom najpreciznije ocrtale konture. Nije savršeno, ali su se konturirale cjeline i reducirali lažni rubovi koji se mogu primijetiti na prethodnim metodama, posebno kod Sobela gdje se ocrtavaju oblaci, ali ne kao cjeline.

Usporedbom algoritama, Robertsov model drži prvo mjesto po jednostavnosti i brzini, odnosno računalnoj učinkovitosti. S obzirom na matricu 2x2 i jednostavnost, nije pouzdan za

složene slike kao što je bila korištena slika, a to je zbog osjetljivosti na šum zbog čega posljedično pati detekcija rubova.

Prewitt je s matricom3x3 također jednostavan za implementaciju, ali isto velika osjetljivost na šum i sklonost detekciji lažnih rubova.

Sobel je nešto zahtjevniji u računanju, ali kao što je viđeno, zaglađivanje, odnosno redukcija šuma zbog koje ima prednost u odnosu na Prewitta, u ovom slučaju također zbog osjetljivosti detektira lažne rubove.

Canny edge je osmišljen kao optimalno rješenje za detekciju rubova u prisutnosti bijelog šuma. Detektira sve važne rubove uz minimalan broj lažnih. Rubovi su također dobro lokalizirani što znači da su blizu stvarnim. Koristi zaostalo pragiranje zbog čega se bolje detektiraju rubovi. Primjenjiv je na složenim slikama kao što je bila zadana slika. Sve te karakteristike nose sa sobom složenost i zahtijevaju više računalnih resursa i vremena zbog više koraka.

Odabir prikladne metode ovisi o zahtjevima. Gledajući na preciznost, od navedena četiri modela, prvi izbor bi bio Canny edge detetction.

5. ZAKLJUČAK

Usporedbom i analizom metoda detekcija rubova u ovom radu prikazani su ključni dijelovi računalnog vida i vizijskih sustava. Pokazana je važnost detekcije u obradi slika, ali i bitnost za razumijevanja i interpretaciju vizualnih informacija. Precizna detekcija omogućuje točniju analizu i razumijevanje slike što je ključno za napredne vizijske sustave koji se koriste u autonomnim vozilima, medicinskim slikama, industrijskoj inspekciji i ostalim područjima. Integracija s naprednijim tehnikama kao na primjer konvolucijskim mrežama i dubokim učenjem se omogućuje da sustavi nauče složenije značajke iz podataka i poboljšaju točnost prepoznavanja i klasifikacije objekta. Opisane metode su uspješne, ali novi izazovi zahtijevaju nova rješenja do kojih bi se moglo doći kombinacijom s naprednijim tehnikama vizijskih sustava pa bi rezultati bili još precizniji.

LITERATURA

[1] Barron J. L., Fleet D. J., and Beauchemin S. S. Performance of optical flow techniques. International Journal of Computer Vision, 1994. to je za 930

[2] Hager G. and Belhumeur P.:Computer Vision and Pattern Recognition, 1996. isto 930

[3] Milan Sonka, Vaclav Hlavac, and Roger Boyle: Image Processing, Analysis, and Machine Vision, 2015. 930

[4] Richard Szeliski: Computer Vision: Algorithms and Applications 2nd Edition, 2021

[5] Rafael C. Gonzalez, Richard E. Woods: Digital Image Processing, 2018.

PRILOZI

I. Algoritmi za detekciju rubova i kontura

```
import cv2
import numpy as np

def run_canny(input_image_path):
    # Read the input image
    image = cv2.imread(input_image_path)
    if image is None:
        raise ValueError("Could not open or find the image")

    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise and improve contour detection
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Perform edge detection using Canny
    edges = cv2.Canny(blurred, 50, 150)

    # Find contours in the edges image
        contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Draw the contours on the original image
    cv2.drawContours(image, contours, -1, (0, 255, 0), 2)  # Green color and thickness 2

    # Save the output image
    cv2.imwrite("Canny/1_gray_" + input_image_path, gray)
```

```
    cv2.imwrite("Canny/2_blurred_" + input_image_path, blurred)
    cv2.imwrite("Canny/3_edges_" + input_image_path, edges)
    cv2.imwrite("Canny/4_contours_" + input_image_path, image)

def run_prewitt(input_image_path):
    # Read the input image
    image = cv2.imread(input_image_path)
    if image is None:
        raise ValueError("Could not open or find the image")

    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Apply Prewitt edge detection
    kernelx = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
    kernely = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
    grad_x = cv2.filter2D(blurred, cv2.CV_64F, kernelx)
    grad_y = cv2.filter2D(blurred, cv2.CV_64F, kernely)
    prewitt = cv2.magnitude(grad_x, grad_y)
    # Convert the magnitude image to 8-bit
    prewitt_8u = np.uint8(prewitt)

    # Apply thresholding to get binary image
    _, thresh = cv2.threshold(prewitt_8u, 50, 255, cv2.THRESH_BINARY)

    # Find contours in the thresholded image
```

```
        contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Draw the contours on the original image
    cv2.drawContours(image, contours, -1, (0, 255, 0), 2)  # Green color and thickness 2

    # Save the output image
    cv2.imwrite("Prewitt/1_gray_" + input_image_path, gray)
    cv2.imwrite("Prewitt/2_blurred_" + input_image_path, blurred)
    cv2.imwrite("Prewitt/3_grad_x_" + input_image_path, grad_x)
    cv2.imwrite("Prewitt/4_grad_y_" + input_image_path, grad_y)
        cv2.imwrite("Prewitt/5_prewitt_apply_gradijenti_i_zbrajanje_" + input_image_path, prewitt)
      cv2.imwrite("Prewitt/6_prewitt_8u_apply_gradijenti_i_zbrajanje_" + input_image_path, prewitt_8u)
    cv2.imwrite("Prewitt/7_contours_" + input_image_path, image)

def run_sobel(input_image_path):
    # Read the input image
    image = cv2.imread(input_image_path)
    if image is None:
        raise ValueError("Could not open or find the image")

    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Apply Sobel edge detection
```

```
    grad_x = cv2.Sobel(blurred, cv2.CV_64F, 1, 0, ksize=3)
    grad_y = cv2.Sobel(blurred, cv2.CV_64F, 0, 1, ksize=3)
    sobel = cv2.magnitude(grad_x, grad_y)

    # Convert the magnitude image to 8-bit
    sobel_8u = np.uint8(sobel)

    # Apply thresholding to get binary image
    _, thresh = cv2.threshold(sobel_8u, 50, 255, cv2.THRESH_BINARY)

    # Find contours in the thresholded image
        contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Draw the contours on the original image
    cv2.drawContours(image, contours, -1, (0, 255, 0), 2)  # Green color and thickness 2

    # Save the output image
    cv2.imwrite("Sobel/1_gray_" + input_image_path, gray)
    cv2.imwrite("Sobel/2_blurred_" + input_image_path, blurred)
    cv2.imwrite("Sobel/3_grad_x_" + input_image_path, grad_x)
    cv2.imwrite("Sobel/4_grad_y_" + input_image_path, grad_y)
    cv2.imwrite("Sobel/5_sobel_apply_gradijenti_i_zbrajanje_" + input_image_path, sobel)
        cv2.imwrite("Sobel/6_sobel_8u_apply_gradijenti_i_zbrajanje_" + input_image_path, sobel_8u)
    cv2.imwrite("Sobel/7_contours_" + input_image_path, image)

def run_roberts_cross(input_image_path):
    # Read the input image
    image = cv2.imread(input_image_path)
```

```
    if image is None:
        raise ValueError("Could not open or find the image")

    # Convert to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Apply Gaussian blur to reduce noise
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Define Roberts cross kernels
    kernelx = np.array([[1, 0], [0, -1]], dtype=int)
    kernely = np.array([[0, 1], [-1, 0]], dtype=int)

    # Apply Roberts cross operator
    grad_x = cv2.filter2D(blurred, cv2.CV_64F, kernelx)
    grad_y = cv2.filter2D(blurred, cv2.CV_64F, kernely)
    roberts = cv2.magnitude(grad_x, grad_y)

    # Convert the magnitude image to 8-bit
    roberts_8u = np.uint8(roberts)

    # Apply thresholding to get a binary image
    _, thresh = cv2.threshold(roberts_8u, 50, 255, cv2.THRESH_BINARY)
    # Find contours in the thresholded image
            contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # Draw the contours on the original image
    cv2.drawContours(image, contours, -1, (0, 255, 0), 2)  # Green color and thickness 2
```

```
    # Save the output image
    cv2.imwrite("Roberts/1_gray_" + input_image_path, gray)
    cv2.imwrite("Roberts/2_blurred_" + input_image_path, blurred)
    cv2.imwrite("Roberts/3_grad_x_" + input_image_path, grad_x)
    cv2.imwrite("Roberts/4_grad_y_" + input_image_path, grad_y)
        cv2.imwrite("Roberts/5_roberts_apply_gradijenti_i_zbrajanje_" + input_image_path, roberts)
      cv2.imwrite("Roberts/6_roberts_8u_apply_gradijenti_i_zbrajanje_" + input_image_path, roberts_8u)
    cv2.imwrite("Roberts/7_tresholding_" + input_image_path, thresh)
    cv2.imwrite("Roberts/8_contours_" + input_image_path, image)


def main():
    input_image = "sava_rest.jpeg"

    # Save away each step in the image processing for comparison
    run_sobel(input_image)
    run_roberts_cross(input_image)
    run_prewitt(input_image)
    run_canny(input_image)


if __name__ == "__main__":
    main()
```