SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

SUSTAVI ZA DIGITALNU OBRADU SIGNALA

MATLAB DETEKCIJA RUBOVA SLIKE S WEB-KAMERE U REALNOM VREMENU

Antonija Grbavac i Ante Šerić

Split, lipanj 2024.

SADRŽAJ

1. UVOD – OPIS ZADATKA

Detekcija rubova predstavlja temeljni korak u analizi i obradi digitalnih slika. Kao tehnika koja omogućuje prepoznavanje značajnih promjena u intenzitetu slike, detekcija rubova koristi se u mnogim područjima računalnog vida, uključujući segmentaciju slike, prepoznavanje objekata i analizu scena. Identifikacija rubova omogućava izlučivanje ključnih značajki iz slika, što je ključno za razumijevanje i interpretaciju vizualnih podataka.

U proteklim desetljećima, razvijeno je mnogo algoritama za detekciju rubova, svaki sa svojim prednostima i ograničenjima. Među najpoznatijima su *Sobel, Prewitt, Canny, LoG* (*Laplacian of Gaussian*) i *Roberts* operatori. Ovi algoritmi koriste različite metode za identificiranje rubova, od jednostavnih diferencijalnih operacija do složenih tehnika filtriranja i analize gradijenata. Međutim, učinkovitost ovih algoritama može značajno varirati ovisno o karakteristikama slike, poput šuma, kontrasta i teksture.

Cilj ovog rada je evaluirati performanse ugrađenih MATLAB funkcija za detekciju rubova i usporediti ih s vlastitom implementacijom Canny detektora rubova koji sliku s web-kamere obrađuje u realnom vremenu. Canny algoritam, poznat po svojoj robusnosti i preciznosti, odabran je za dublju analizu i prilagodbu zbog svoje sposobnosti prilagodbe različitim uvjetima slike kroz parametre kao što su pragovi (engl. *threshold*) i Gaussovo filtriranje. Implementacijom i testiranjem vlastitog Canny detektora, istražit će se mogućnosti poboljšanja u odnosu na standardne metode i analizirati kako različiti parametri utječu na performanse detekcije rubova.

Istraživanje uključuje procjenu različitih metrika performansi, uključujući recall faktor, preciznost, F1 rezultat i vrijeme obrade, kako bi se dobio sveobuhvatan uvid u efikasnost i pouzdanost svakog algoritma.

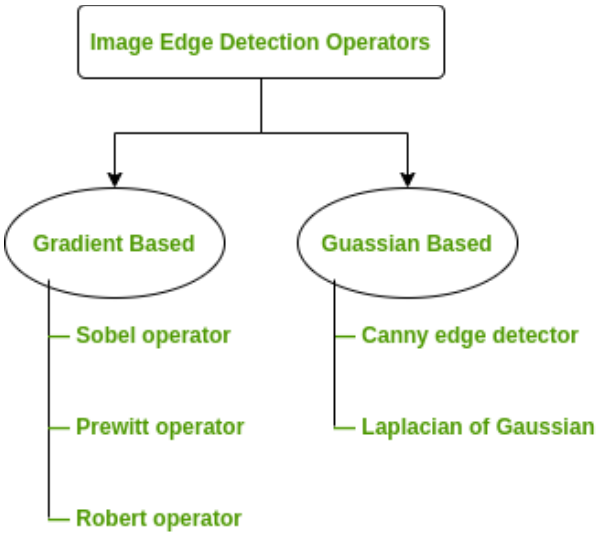
2. DETEKCIJA RUBOVA NA SLICI

Detekcija rubova je metoda segmentiranja slike putem razlikovanja i lociranja područja diskontinuiteta na slici. Široko je korištena tehnika u digitalnoj obradi slike, a primjenu pronalazi u prepoznavanju uzoraka, morfologije slike te izdvajanju značajki.

Rub može sadržavati najvažnije informacije o slici. Sastoji se od skupa piksela, usko povezanih, koji tvore granicu između dva disjunktna područja. Detektiranjem značajnih promjena svjetline piksela područja diskontinuiteta, ova metoda omogućuje jednostavno i precizno promatranje objekata na slici. [2]

Operatore za detekciju rubova na slici moguće je podijeliti u dvije skupine:

- **operatori temeljeni na gradijentu** (engl. *Gradient – based operators*) koji promatraju promjenu intenziteta na rubovima te pronalaze područja gdje je derivacija intenziteta veća od određene granične vrijednosti (poput *Sobel, Prewitt, Roberts* operatora)
- **Gaussovi operatori** (engl. *Gaussian – based operators*) koji računaju derivacije drugog reda u slici, a detekcijom nultih prijelaza pronalaze rubove (kao *Canny* detektor rubova te *LoG – Laplacian of Gaussian* operator)



Slika 1. Osnovna podjela operatora za detekciju rubova [1]

2.1. Vrste operatora za detekciju rubova

Korištenje gotovih operatora olakšava pronalazak rubova na slici, a svojom svestranošću pronalaze mnoge primjene. Svima je jednako da koriste konvolucijske maske kojima pronalaze vodoravne, okomite ili dijagonalne rubove.

2.1.1. Sobel operator

Diskretni je operator diferenciranja. Računa aproksimaciju gradijenta funkcije intenziteta slike za detekciju rubova. Na pikselima slike, Sobelov operator proizvodi ili normalu na vektor ili odgovarajući vektor gradijenta. Koristi dvije 3 x 3 matrice ili maske (M_x *i* M_y) koje se konvoluiraju s ulaznom slikom za računanje aproksimacije vertikalne i horizontalne derivacije.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Metoda pruža jednostavan i vremenski učinkovit proračun, a potraga za glatkim rubovima jako je olakšana. Međutim, ovaj operator pokazuje veliku osjetljivost na šum te manju preciznost u detekciji rubova. Dijagonalni rubovi nisu uvijek ispravno sačuvani, a detekcija s jačim i grubim rubovima ne daje odgovarajuće rezultate.

2.1.2. Prewitt operator

Gotovo sličan Sobel operatoru. Također detektira okomite i vodoravne rubove slike. Jedan je od najboljih načina otkrivanja orijentacije i veličine slike. U tu svrhu koristi maske:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Iako predstavlja sjajnu izvedbu za detekciju okomitih i vodoravnih linija slike, dijagonalne linije nisu sačuvane, kao i kod Sobela. Prilagođavanje operatora za određenu primjenu je nemoguće, budući da je veličina koeficijenata fiksna te time nepromjenjiva.

2.1.3. Roberts operator

Ovaj operator baziran na određivanju gradijenta računa zbroj kvadrata razlike između dijagonalno susjednih piksela na slici kroz diskretno diferenciranje. Potom se vrši aproksimacija gradijenta. Kako bi postigao navedeno, koristi sljedeće dvije 2 x 2 maske

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Ovaj operator predstavlja najjednostavnije rješenje za brzu detekciju rubova i orijentacije, uz očuvanje dijagonalnih linija. Zbog svoje brzine i jednostavnosti, ali i velike osjetljivosti na šum, izrazito kaska po preciznosti za ostalim operatorima te se u primjenama, gdje je točnost ključna, ne koristi.

2.1.4. Laplacian of Gaussian (LoG)

Marr-Hildreth operator (poznatiji kao *LoG operator*) Gaussov je operator koji koristi Laplacea za izračun druge derivacije funkcije intenziteta slike. Odlično traži područja slike gdje se intenzitet naglo mijenja, što najčešće ukazuje na rub. Baziran je na tj. *zero-crossing* metodi, kada derivacija drugog reda prijeđe nulu onda ta lokacija odgovara maksimalnoj vrijednosti. Gaussov operator smanjuje šum, a Laplace detektira jake rubove. Gaussova funkcija definirana je kao

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

gdje je μ *srednja vrijednost*, a σ *standardna devijacija*. LoG operator računa se izrazom

$$LoG = \frac{\partial^2}{\partial x^2} G(x, y) + \frac{\partial^2}{\partial y^2} G(x, y)$$

LoG operator predstavlja odličan način detekcije rubova i njihovih različitih orijentacija. Precizno detektira jake rubove u svim smjerovima. No zbog osjetljivosti na šum, podložan je i detekciji tzv. „lažnih rubova“, pri čemu moguće smetnje u gradijentu zamjenjuje za rubove.

2.1.5. Canny operator

Gaussov je operator gotovo u potpunosti otporan na šum. Gotovo savršeno izdvaja značajke sa slike. Ima napredni algoritam koji se bazira na radu prethodno opisanog LoG operatora. Široko je korištena tehnika optimalne detekcije rubova. Detekcija se vrši na temelju tri kriterija: niska stopa pogreške, točno lokalizirane rubne točke te mogućnost postojanja samo jednog rubnog odziva. Zbog sjajne lokalizacije linija slike te niske osjetljivosti na šum, Canny detektor je najbolje rješenje za preciznu i sigurnu detekciju i ekstrakciju značajki i objekata sa slike. Zbog svojih naprednih mogućnosti, njegovo računanje je složeno i računalno intenzivnije u odnosu na ostale vrste operatora.

2.2. Usporedba performansi MATLAB internih funkcija za detekciju rubova

Testiranje i procjena rada algoritama za detekciju rubova uključuje nekoliko koraka kako bi se osigurala njihova točnost, učinkovitost i robusnost. U ovom radu, rezultati testiranja bazirani su na kvantitativnoj procjeni te njenim parametrima: preciznosti, *recall* faktoru, F1 rezultatu te potrebnom vremenu obrade [3].

2.2.1. Opis korištenih parametara u kvantitativnoj procjeni rada operatora

Preciznost mjeri točnost detekcije algoritma. Odgovara na pitanje: *od svih detektiranih rubova, koliko ih je zapravo stvarnih*.

$$PR = \frac{TP}{TP + NP}$$

gdje je:

- TP (točno pozitivni) – broj točno predviđenih stvarnih rubova
- NP (netočno pozitivni) – broj netočno predviđenih stvarnih rubova

Veća preciznost ukazuje na to da kada algoritam predvidi rub, to je vrlo vjerojatno točno.

***Recall* faktor** određuje sposobnost algoritma da identificira sve stvarne rubove. Pokušava *među svim stvarnim rubovima otkriti koliko ih je algoritam točno predvidio kao stvarne*.

$$recall = \frac{TP}{TP + NN}$$

gdje je:

- NN (netočno negativni) – broj pozitivnih slučajeva koji su netočno detektirani kao negativni

Visoka vrijednost *recall* faktora ukazuje da algoritam može detektirati većinu točnih rubova na danoj slici.

F1 rezultat harmonijska je sredina preciznosti i *recall* faktora. Pruža jedinstvenu mjeru kao balans između ova dva parametra. Maksimalni F1 rezultat je 1 (savršena preciznost i recall), a minimalni je 0.

$$F1 = 2 \cdot \frac{PR \cdot recall}{PR + recall}$$

2.2.2. BSDS500 baza slika

Kako bi se uspješno provela analiza rada operatora uz pomoć navedenih parametara, za testiranje korištena je **BSDS500 baza slika** (engl. *Berkeley Segmentation Dataset and Benchmark 500*) [4]. Radi se o široko korištenom skupu podataka u računalnom vidu, posebno za ispitivanje i treniranje sustava za poslove detekcije i segmentacije na slikama. Sadrži 500 prirodnih slika različitih scena i objekata. Slike su raznolike, uključuju urbane, ruralne, unutarnje i vanjske prikaze.

Za svaku sliku dano je nekoliko temeljnih segmentacija (u engl. literaturi poznato kao *ground truth segmentation).* Ove su segmentacije nastale ručnim označavanjem rubova slike različitih ispitanika. Svaka segmentacija odražava ljudsku percepciju važnih granica i segmenata na slici. Ove segmentacije korištene su u izradi **temeljnih datoteka** (engl. *ground truth files*) koje ističu rubove i granice objekata svake referentne slike baze, na način kako ih ljudi percipiraju. Ove su datoteke bitne za procjenu rada algoritama detekcije rubova jer, usporedbom

rezultata obrade pojedinog operatora sa sadržajem segmentacijskih datoteka, moguće je jednostavno provesti kvantitativnu analizu rada operatora.

2.2.3. Prikaz rezultata izvođenja internih operatora u MATLAB-u

Po kratkom objašnjenju parametara koji su uzeti u obzir pri testiranju te korištene baze referentnih slika, slijedi prikaz rezultata izvođenja internih operatora u MATLAB-u. Bitno je napomenuti kako je u analizi korišteno 200 referentnih slika i odgovarajućih temeljnih datoteka opisane baze.

Tablica 1. Rezultati testiranja MATLAB internih funkcija za detekciju rubova

	Sobel	*Prewitt*	*Canny*	*LoG*	*Roberts*
recall	0,1491	0,15	0,2634	0,193	0,1738
PR	0,0651	0,0656	0,0386	0,0418	0,0967
F1 rezultat	0,0855	0,0861	0,0649	0,0657	0,117
vrijeme obrade [s]	0,0014	0,0008	0,0071	0,0048	0,0014

Roberts operator pokazao se kao dobar omjer preciznosti i vremena obrade u ovom testiranju, uz održavanje razumne vrijednosti recall-a. Upravo u *recall* parametru ističe se Canny detektor, što ukazuje veću osjetljivost na otkrivanje rubova, po cijeni duljeg vremena obrade. No, preciznost mu je u ovom testu dosta mala (moguće razloge ovakvih rezultata moguće je pronaći u poglavlju *Zaključak*). Prewitt operator je najbrži, što ga čini pogodnim za primjene gdje je brzina važna, iako žrtvuje određenu točnost.

Vizualne rezultate testiranje operatora nad tri odabrane slike iz korištene baze moguće je pronaći u *Dodatku A* na kraju ovoga rada.

Budući da je Canny operator najpoznatiji i najčešće korišten detektor rubova te s obzirom na sve njegove prednosti, u nastavku se podrobnije obrađuje kroz objašnjenje realizacije vlastite funkcije za detekciju rubova.

3. PREDLOŽENO RJEŠENJE

Canny detektor rubova koristi složeni algoritam koji se sastoji od nekoliko metoda skladno ukomponiranih za otkrivanje širokog raspona rubova na slikama. Sastoji se od **5 koraka**:

- redukcija šuma slike
- izračun gradijenta
- ne-maksimalno potiskivanje (engl. *non-maximum suppression*)
- primjena dvostruke granične vrijednosti (engl. *double threshold*)
- praćenje rubova primjenom histereze

Nakon primjene svih koraka, dobiju se rezultati kao na sljedećoj slici.

Slika 2. Rezultat obrade slike primjenom vlastitog Canny operatora

3.1. Redukcija šuma slike

Šum na slici smanjuje se primjenom Gaussovog filtra, koji izglađuje sliku piksel po piksel računanjem prosjeka vrijednosti susjednih piksela unutar definiranog kernela (veličine 3 x 3, 5 x 5, 7 x 7 itd.). Veličina kernela utječe na očekivani efekt zamućenja, manji kernel, slabije vidljivo zamućenje. Funkcija imgaussfilt primjenjuje Gaussov filter sa standardnom devijacijom σ.

3.2. Izračun gradijenta

Ovaj korak otkriva intenzitet i smjer ruba računanjem gradijenta slike. Rubovi odgovaraju promjeni intenziteta piksela. Da bi se otkrila, najjednostavnije je primijeniti filtre koji ističu ovu promjenu intenziteta u oba smjera: vodoravno (I_x) i okomito (I_y) . Implementiraju se konvolucijom intenziteta sa Sobel maskama K_x i K_y (navedene u potpoglavlju *2.1.1. Sobel operator*).

Potom se računa magnituda gradijenta (G_{mag}) te smjer (kut) putem relacija

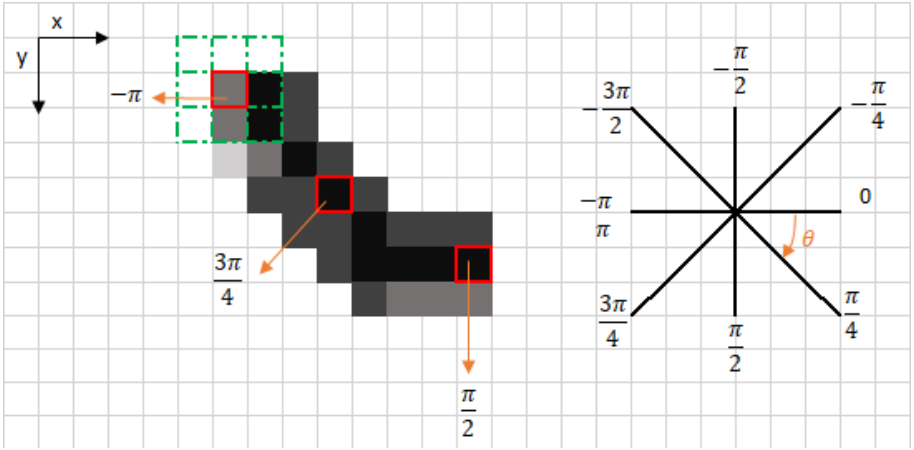
$$|G_{mag}| = \sqrt{{I_x}^2 + {I_y}^2}$$

$$\theta(x, y) = arctg\left(\frac{I_y}{I_x}\right)$$

Magnituda predstavlja jakost ruba, a kut smjer ruba. Kao rezultat dobije se niz rubova na slici, od koji su neki puniji, a neki tanji. Sljedeći korak, ne-maksimalno potiskivanje, ublažit će punije linije. Također, razina intenziteta gradijenta varira u vrijednostima između 0 i 255. Rubovi na finalnoj slici moraju imati ujednačen intenzitet (255), što odgovara intenzitetu bijelih piksela.

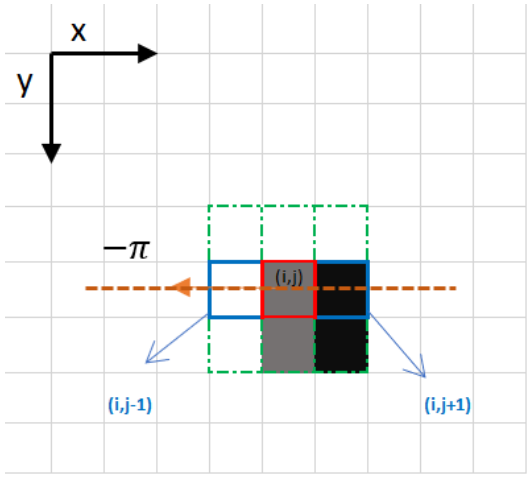
3.3. Ne-maksimalno potiskivanje

Ova se metoda koristi za redukciju debljine rubova, pri čemu se održavaju pikseli najvećeg intenziteta u smjeru gradijenta, a ostali se potiskuju. Ovaj korak idealno smanjuje rubove na širinu od jednog piksela.



Slika 3. Postupak ne-maksimalnog potiskivanja na rubne piksele [5]

Crveni okvir u gornjem lijevom kutu na prethodnoj slici predstavlja rubni piksel na kojem se trenutno primjenjuje matrica intenziteta gradijenta. Odgovarajući smjer ruba predstavljen je narančastom strelicom (kut $-\pi$ radijana).



Slika 4. Primjer djelovanja metode na jednom rubnom pikselu [5]

Na *Slici 4.* piksel (i, j) se obrađuje, a pikseli u istom smjeru označeni su plavom bojom (i, j-1) i (i, j+1). Ako je jedan od ovih susjednih piksela većeg intenziteta od promatranog, onda se zadržava samo onaj intenzivniji. U danom slučaju, piksel (i, j-1) je intenzivniji (bijele boje, vrijednosti 255). Stoga, vrijednost intenziteta trenutnog piksela (i, j) postavlja se na 0. Ako ne postoji piksel većeg intenziteta u smjeru ruba od promatranog, tada se vrijednost trenutnog piksela zadržava.

S obzirom na sve navedeno, metoda se sastoji od sljedećih koraka:

- kreiranje matrice veličine kao i izvorna matrica intenziteta gradijenta (G_{mag}) te njena inicijalizacija na vrijednost 0
- određivanje smjera ruba na temelju vrijednosti kuta (θ)
- provjera postoji li u istom smjeru piksel većeg intenziteta od trenutnog koji se obrađuje
- vraćanje slike obrađene navedenom metodom

Rezultat je sličan ulaznoj slici, s nešto tanjim rubovima. Međutim, moguće je još uvijek uočiti neke varijacije intenziteta rubnih piksela; čini se da su neki rubovi svjetliji od drugih. Zadaća je sljedeća dva koraka pokriti ovaj nedostatak.

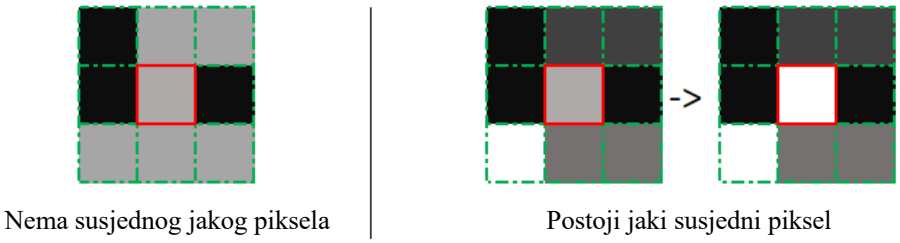
3.4. Primjena dvostruke granične vrijednosti

Korak primjene dvostrukog praga ima za cilj identificirati tri vrste piksela: *jake, slabe* i *nerelevantne*. Jaki pikseli imaju visoki intenzitet te sigurno doprinose konačnom rubu. Slabi pikseli imaju vrijednost intenziteta nedovoljnu da bi se smatrali jakima, ali ipak nedovoljno malu da bi ih se smatralo nerelevantnima. Ostali pikseli ne uzimaju se u obzir jer ne pridonose postojanju rubova.

Viša granična vrijednost (engl. *threshold*) koristi se za detekciju jakih rubova (intenziteta većeg od više granične vrijednosti). Niži se prag koristi za eliminaciju nerelevantnih piksela (intenziteta nižeg od definirane niže granične vrijednost). Svi pikseli, čiji je intenzitet po vrijednosti između oba praga, označeni su kao slabi. Sljedeći korak (primjena histereze) pomoći će u određivanju koji će se slabi rubovi naći na finalnoj slici.

3.5. Praćenje rubova primjenom histereze

Praćenje rubova histerezom osigurava zadržavanje slabih rubova povezanih s jakima, dok su ostali potisnuti. To pomaže u povezivanju rubnih segmenata i odbacivanju izoliranih slabih rubova.



Slika 5. Primjer određivanja veze slabih piksela s jakima [5]

Cijeli kod opisanog Canny algoritma moguće je pronaći u *Dodatku B* ovoga rada, a načine na koje svaki od opisanih koraka algoritma utječe na ulaznu sliku moguće je proučiti na *Slici 6.*

Slika 6. Rezultati primjene metoda Canny detektora na ulaznu sliku

4. REZULTATI TESTIRANJA PREDLOŽENOG RJEŠENJA

Predloženo rješenje kao ulaz koristi slike s web-kamere te ih obrađuje u realnom vremenu. Stoga je prvo provedeno testiranje brzine izvođenja algoritma gdje smo mjerili ukupno vrijeme obrade, prosječno vrijeme obrade po slici videa te broj prikazanih slika u sekundi (engl. *frame rate*). Za ovaj dio testa uzeti su fiksni iznosi omjera graničnih vrijednosti (`lowThresholdRatio = 0.25, highThresholdRatio = 0.275`).

Tablica 2. Testiranje brzine izvođenja predloženog algoritma koji sliku s web-kamere obrađuje u realnom vremenu

ukupno vrijeme obrade [s]	*prosječno vrijeme obrade po slici videa [s]*	*broj prikazanih slika u sekundi [FPS – engl. frames per second]*
1,9158	0,0383	26,1

Ova analiza samo okvirno prikazuje vremenske performanse sustava kada podatke obrađuje u realnom vremenu. U ovom slučaju iznosi omjera vrijednosti praga su dosta bliski, čime je detekcija rubova dosta olakšana i brzina obrade velika. Kako razlika između iznosa omjera pragova raste, tako algoritam u analizu uzima veći broj rubova, a time vremenske performanse sustava neminovno opadaju. Bitno je naglasiti kako je ovakva analiza uvjetovana hardverskom konfiguracijom (lošije performanse hardvera nužno donose niže brzine obrade algoritma), ali i rezolucijom web-kamere koja je pri testiranju korištena (niža rezolucija donosi veće brzine obrade po cijeni manje preciznosti). Za potrebe testiranja korištena je sljedeća konfiguracija: 11th Gen Intel Core i7-1165G7 @ 2.80GHz, 16 GB RAM memorije te web-kamera rezolucije 1280x720 piksela.

Nemoguće je precizno odrediti performanse i točnost sustava kada sliku sa web-kamere obrađuje u realnom vremenu. Stoga je potrebno prilagoditi algoritam testiranju na velikom skupu referentnih slika. Za tu namjenu, koristi se isti set od 200 slika iz baze BSDS500, korištenih za testiranje internih operatora detekcije rubova (značajke baze opisane u potpoglavlju *2.2.2. BSDS500 baza slika*). Parametri testiranja ostaju isti (preciznost, recall faktor, F1 rezultat i vrijeme obrade), a testiranje je provedeno za različite iznose omjera graničnih vrijednosti (`lowThresholdRatio` i `highThresholdRatio`).

Tablica 2.Usporedba performansi izrađenog Canny operatora za različite iznose omjera graničnih vrijednosti

`lowThresholdRatio / highThresholdRatio`	0,02 / 0,1	0,05 / 0,15	0,25 / 0,275
recall	0,2887	0,2675	0,2081
PR	0,0356	0,0429	0,0633
F1 rezultat	0,0608	0,0703	0,0899
vrijeme obrade [s]	0,0431	0,0468	0,0332

Verzija programa s omjerima praga 0,25/0,275 daje najviši F1 rezultat te preciznost, sugerirajući da nudi najbolji balans između otkrivanja pravih rezultata i smanjivanja lažno pozitivnih rezultata, s najkraćim vremenom obrade. Verzija s 0,02/0,1 omjerima osigurava najveći recall faktor, što može biti korisno u primjenama gdje je detekcija što većeg broja rubova krucijalna, nauštrb preciznosti i vremena obrade. Omjeri praga 0,05/0,15 nude kompromis s dobrim vrijednostima svih triju parametara, no zahtjeva dulje vrijeme obrade od ostalih inačica.

5. ZAKLJUČAK

U ovom radu istraživalo se područje detekcije rubova, ključne tehnike u računalnom vidu i obradi slike. Detekcija rubova služi kao temeljni alat za identifikaciju granica objekata unutar slika, čime olakšava zadatke poput segmentacije slike, prepoznavanja uzoraka i analize scena. Fokus je stavljen na ocjenjivanje performansi internih algoritama MATLAB-a, ali i vlastitog algoritma za detekciju rubova koristeći različite parametre, uključujući recall faktor, preciznost, F1 rezultat i vrijeme obrade.

U početku je testirano nekoliko ugrađenih operatora za detekciju rubova dostupnih u MATLAB-u, uključujući *Sobel, Prewitt, Canny, LoG* i *Roberts*. Rezultati su pokazali da su metričke performanse ovih operatora bile niže od očekivanih. Na primjer, Canny operator, unatoč svojoj robusnosti, postigao je recall od 0.2643, preciznost od 0.0386 i F1 rezultat od 0.0649. Ovi rezultati mogu se pripisati nekoliko čimbenika, uključujući složenosti skupa podataka, varijacijama u kvaliteti slike i fiksnim parametrima internih funkcija koje možda nisu optimizirane za specifični skup podataka. Ova otkrića naglašavaju potrebu za podešavanjem parametara i adaptivnim metodama za poboljšanje performansi detekcije rubova.

U drugom dijelu rada, opisana je implementacija vlastitog Canny detektora rubova. Navedeni pristup uključivao je sljedećih pet koraka: redukciju šuma, izračun gradijenta, ne-maksimalno potiskivanje, primjenu dvostruke granične vrijednosti te praćenje rubova histerezom.

Vlastiti Canny detektor rubova rigorozno je testiran s različitim omjerima pragova kako bi se razumio njihov utjecaj na performanse. Rezultati su pokazali da niži omjeri pragova (npr. 0,02/0,1) rezultiraju višim recall-om, ali nižom preciznošću, ističući kompromis između osjetljivosti i točnosti. Viši omjeri pragova (npr. 0,25/0,275) pružili su uravnotežene performanse s višim F1 rezultatom i kraćim vremenom obrade, pokazujući važnost podešavanja parametara za optimalnu detekciju rubova.

Unatoč poboljšanjima postignutim vlastitom implementacijom, postoji prostora za daljnje unapređenje. Vlastiti Canny detektor, iako nudi bolju kontrolu nad parametrima, i dalje se suočava s izazovima u rukovanju šumom i varijabilnim teksturama slika. Budući rad mogao bi istražiti adaptivne tehnike određivanja praga i pristupe strojnog učenja za dinamičko podešavanje

parametara na temelju sadržaja slike. Dodatno, integracija sofisticiranijih metoda za redukciju šuma i tehnika za poboljšanje rubova mogla bi dodatno unaprijediti proces detekcije rubova.

6. LITERATURA

[1] „Image Edge Detection Operators in Digital Image Processing“, s Interneta, https://www.geeksforgeeks.org/image-edge-detection-operators-in-digital-image-processing/?ref=header_search , 20. lipnja 2024.

[2] Baareh, A.K.M. i dr.: „Performance Evaluation of Edge Detection Using Sobel, Homogeneity and Prewitt Algorithms“, Journal of Software Engineering and Applications, pp. 537-551. 2018.

[3] Khaire, P.A., Thakur, N.V.: „A Fuzzy Set Approach for Edge Detection“, International Journal of Image Processing (IJIP), Volume 6, 2012.

[4] Ashwath, B. „Berkeley Segmentation Dataset 500 (BSDS500)“, s Interneta, https://www.kaggle.com/datasets/balraj98/berkeley-segmentation-dataset-500-bsds500, 16. lipnja 2024.

[5] Sahir, S.: „Canny Edge Detection Step by Step in Python - Computer Vision“, s Interneta, [https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123, 19. lipnja 2024.

[6] Liang, J: „Canny Edge Detection“, s Interneta, https://justin-liang.com/tutorials/canny/, 15. lipnja 2024.

[7] Git projekt „canny_edge_detector“, s Interneta, https://github.com/FienSoP/canny_edge_detector/blob/master/canny_edge_detector.py, 19. lipnja 2024.

[8] Git projekt „CannyEdgeDetector“, s Interneta, https://github.com/JustinLiang/ComputerVisionProjects/blob/master/CannyEdgeDetector/CannyEdgeDetector.m, 19. lipnja 2024.

[9] Kumar, A.: „Basic of Image Processing to Live Edge Detection“, s Interneta, https://www.mathworks.com/matlabcentral/fileexchange/75490-basic-of-image-processing-to-live-edge-detection?s_tid=prof_contriblnk , 19. lipnja 2024.

[10] „Matlab | Edge Detection of an image without using in-built function“, s Interneta, https://www.geeksforgeeks.org/matlab-edge-detection-of-an-image-without-using-in-built-function/?ref=ml_lbp, 17. lipnja 2024.

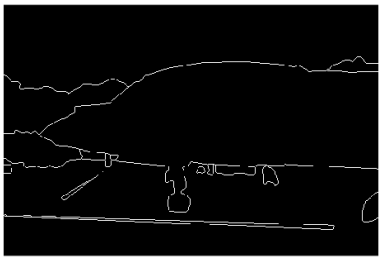
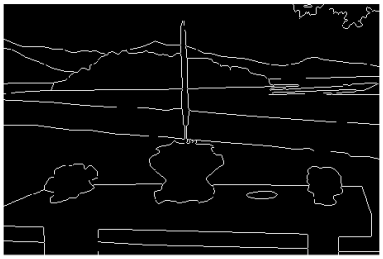
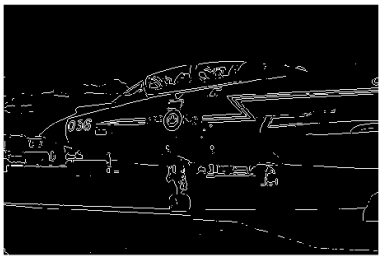
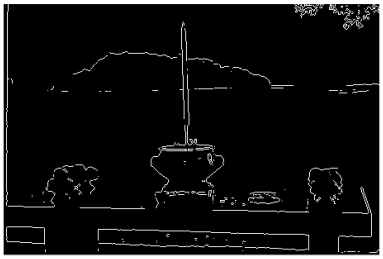
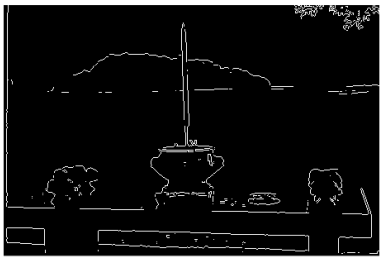
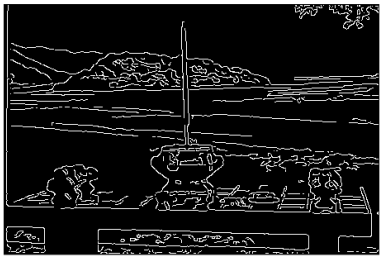
[11] Shihab, A.: „Comparative study among Sobel, Prewitt and Canny Edge Detection operators used in image processing“, Journal of Theoretical and Applied Information Technology, Vol. 96, No, 19, 2018.

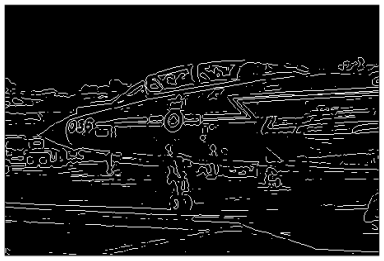
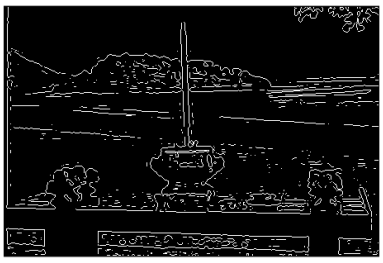
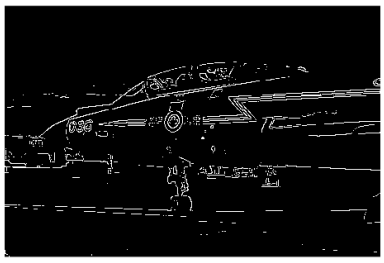
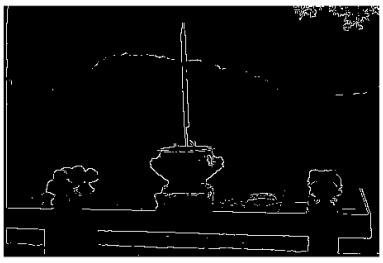
[12] Sahoo, T., Pine, S.„Design and Simulation Of Various Edge Detection Techniques Using Matlab Simulink“, International Conference on Signal Processing, Communication, Power and Embedded Systems (SCOPES), 2016.

Popis oznaka i kratica

engl.	engleski jezik
LoG	Laplacian of Gaussian operator
FPS	frames per second (slika u sekundi)

DODATAK A – vizualni rezultati testiranja internih operatora

Testne slike			
Ručno označeni rubovi (engl. *ground truth files*)			
Sobel			
Prewitt			
Canny			

LoG			
Roberts			

DODATAK B – pregled programskog koda

```
% Inicijalizacija kamere
cam = webcam;
% cam.Resolution='640x480'; %mijenjanje rezolucije kamere

% Inicijalizacija varijabli frameCount i totalProcessingTime
frameCount = 0;
totalProcessingTime = 0;

% Parametri
sigma = 1;
% kernel_size = 5;
weak_pixel = 75;
strong_pixel = 255;
lowThresholdRatio = 0.02;  %omjere pragova moguće mijenjati
highThresholdRatio = 0.1;

% Kreiranje figure za prikaz originalnog videa i inačice s detektiranim
% rubovima
figureHandle = figure;
% Postavljanje veličine figure na cijeli zaslon
set(figureHandle, 'Units', 'normalized', 'OuterPosition', [0 0 1 1]);

%Cijeli program se izvršava u 50 slika, moguće mijenjati trajanje programa
while frameCount < 50
    % Dohvaćanje slike (frame) s web kamere
    input_image = snapshot(cam);
    frameCount = frameCount + 1;

    % ##### Početak mjerenja vremena obrade #####
    tic;

    % Pretvorba ulazne slike u crno-bijelu
    grayImage = rgb2gray(input_image);

    % Primjena Gaussovog filtera; stvara zaglađenu sliku, otklanja šum
    img = imgaussfilt(grayImage, sigma);
    img = double(img);

    % Maske Sobel operatora (za x i y smjer slike)
    Kx = [-1 0 1; -2 0 2; -1 0 1];
    Ky = [1 2 1; 0 0 0; -1 -2 -1];

    % Aproksimacije gradijenata
    Ix = imfilter(img, double(Kx), 'conv', 'replicate');
    Iy = imfilter(img, double(Ky), 'conv', 'replicate');

    % Izračun amplitude gradijenta i smjera ruba
    G_mag = sqrt(Ix.^2 + Iy.^2);
    G_mag = G_mag / max(G_mag(:)) * 255; % normalizacija
    angle = atan2(Iy, Ix)*180/pi;
```

```
    [height, width] = size(G_mag);

    % Ne-maksimalno potiskivanje
    output = zeros(height, width);
    angle(angle < 0) = angle(angle < 0) + 180;

    for i = 2:height-1
        for j = 2:width-1
            q = 255;
            r = 255;

            % Kut 0
            if (0 <= angle(i, j) && angle(i, j) < 22.5) || (157.5 <= angle(i,
j) && angle(i, j) <= 180)
                q = G_mag(i, j + 1);
                r = G_mag(i, j - 1);
            % Kut 45
            elseif (22.5 <= angle(i, j) && angle(i, j) < 67.5)
                q = G_mag(i + 1, j - 1);
                r = G_mag(i - 1, j + 1);
            % Kut 90
            elseif (67.5 <= angle(i, j) && angle(i, j) < 112.5)
                q = G_mag(i + 1, j);
                r = G_mag(i - 1, j);
            % Kut 135
            elseif (112.5 <= angle(i, j) && angle(i, j) < 157.5)
                q = G_mag(i - 1, j - 1);
                r = G_mag(i + 1, j + 1);
            end

            if (G_mag(i, j) >= q) && (G_mag(i, j) >= r)
                output(i, j) = G_mag(i, j);
            else
                output(i, j) = 0;
            end
        end
    end

    % Primjena dvostruke granične vrijednosti
    highThreshold = max(output(:)) * highThresholdRatio;
    lowThreshold = highThreshold * lowThresholdRatio;

    res = zeros(height, width);

    strong = strong_pixel;
    weak = weak_pixel;

    strong_indices = find(output >= highThreshold);
    weak_indices = find((output < highThreshold) & (output >= lowThreshold));

    res(strong_indices) = strong;
    res(weak_indices) = weak;
```

```
% Praćenje rubova primjenom histereze
    for i = 2:height-1
        for j = 2:width-1
            if (res(i, j) == weak)
                if any(any(res(i-1:i+1, j-1:j+1) == strong))
                    res(i, j) = strong;
                else
                    res(i, j) = 0;
                end
            end
        end
    end

    % Prekid mjerenja vremena obrade
    processingTime = toc;
    totalProcessingTime = totalProcessingTime + processingTime;

    %Prikaz originalnog videa i inačice s detektiranim rubovima u jednom
    %prozoru
    subplot(1, 2, 1);
    imshow(input_image);
    title('Originalni video');

    subplot(1, 2, 2);
    imshow(res);
    title('Video s detektiranim rubovima');

    %Pričekati da se zaslon osvježi
    drawnow;
end

clear cam;

% Računanje prosječnog vremena obrade po slici videa
averageProcessingTime = totalProcessingTime / frameCount;

% Prikaz metrika performansi
fprintf('Ukupno vrijeme obrade: %.4f seconds\n', totalProcessingTime);
fprintf('Prosjecno vrijeme obrade po slici: %.4f seconds\n',
averageProcessingTime);
fprintf('Broj prikazanih slika (frame rate): %.2f FPS\n', 1 /
averageProcessingTime);

disp('Program uspješno završen!');
```