

**PARALELNO PROGRAMIRANJE
-DETEKCIJA IVICA UNUTAR SLIKE-**

Dušan Lečić SV80/2021

1 – Analiza problema

Problem detekcije ivica se može definisati kao proces identifikacije prelaznih regiona između različitih oblasti u slikama. Osnovna ideja je pronaći tačke na slici gde intenzitet piksela naglo varira. Ove tačke predstavljaju granice objekata i predstavljene su kao promene u teksturi, boji ili osvetljenju. Ova operacija je veoma bitna u oblastima kao što su prepoznavanje lica, robotika, digitalna obrada slike i mnoge druge.

2 – Koncept rešenja

Posmatraćemo dva pristupa detektovanja ivica: Prewitt operator i detekcija ivica u okolini piksela.

Prewitt operator koristi kvadratne matrice veličine 3×3 piksela (ponekad i 5×5 piksela). Ove matrice se postavljaju preko svakog piksela na slici, a zatim se vrši konvolucija između matrice filtera i matrice okoline piksela. Ovaj proces se primenjuje odvojeno za horizontalne i vertikalne filtere. Rezultati konvolucije se sabiraju uzimajući apsolutne vrednosti, a pikseli se boje crno ili belo na osnovu predefinisanog praga. Kada se ovaj postupak primeni na sve piksele slike, dobija se filtrirana slika sa jasno istaknutim ivicama objekata.

Detekcija ivica u okolini piksela koristi dva operatora: P i O . Ovaj algoritam može se izvršiti u dva prolaza kroz sliku ili ali je u ovom projektu implementiran kroz jedan prolaz. U prvom prolazu, vrednosti piksela iznad praga se postavljaju na jedinicu, dok se vrednosti ispod praga postavljaju na nulu. U drugom prolazu, analiziramo okolinu svakog piksela, koju prethodno definišemo. Ako u okolini postoje jedinice, operator P se postavlja na jedinicu, inače na nulu. Slično tome, ako u okolini postoje nule, operator O se postavlja na nulu, inače na jedinicu. Razlikom operatora P i O dobijamo informaciju da li je piksel ivica objekta ili ne.

Oba pristupa imaju svoje prednosti i nedostatke. Prewitt operator je jednostavan za implementaciju i brz za izračunavanje, ali može biti osetljiv na šum i generisati tanke ivice. Detekcija ivica u okolini piksela pruža više kontrole nad okolinom piksela, ali može biti složenija za implementaciju. Konačan izbor pristupa zavisi od specifičnih zahteva problema i željenih rezultata.

3 – Programsко rešenje

Projekat se sastoji iz 2 glavna dela:

- bitmap – direktorijum koji sadrži EasyBMP biblioteku
- detector – direktorijum koji sadrži implementaciju algoritama

U implementaciji algoritma korišćena je EasyBMP biblioteka u svrhu konvertovanja slika iz bitmap formata u matricu vrednosti svakog piksela, gde je vrednost svakog piksela između 0 i 255. Iz biblioteke je korišćena BitmapRawConverter klasa, koja služi za otvaranje slike u željenom formatu. Ova klasa osigurava da će slika biti tretirana kao crno-bela slika.

U direktorijumu "detector" nalazi se "detector.h" zaglavljne fajl koji sadrži:

- konstante za Prewitt operator – matrice kojim filtriramo sliku, postoje vertikalne i horizontalne matrice od po 9, odnosno 25 elemenata
- THRESHOLD konstantu – vrednost koju koristimo kada je potrebno doneti odluku da li će piksel izlazne slike biti obojen u crno ili belo
- pixel_grid strukturu – struktura koja sadrži početnu i krajnju vrednost za širinu i dužinu slike, koristi se u metodama gde se prosleđuje veličina slike ili nekog njenog dela
- prewitt_convolve funkcija - služi za implementaciju Prewitt operatora
- edge_detection_p_and_o funkciju - služi za računanje odnosa P i O vrednosti kod detekcije ivica u okolini piksela
- Detector klasu – klasa koja sadrži sve parametre potrebne za obradu slike, kao i funkcije za pokretanje svakog serijskog i paralelnog algoritma korišćenog u projektu

Pored zaglavlja, tu je i implementacija pomoćnih funkcija i klase koje su prethodno navedene. Evo najvažnijih funkcija:

- prewitt_convolve – funkcija koja primenjuje konvoluciju matrice filtera na matricu okoline piksela na slici, računa horizontalne i vertikalne

vrednosti piksela, i vraća vrednost koja nam govori da li piksel na izlaznom fajlu treba biti crne ili bele boje

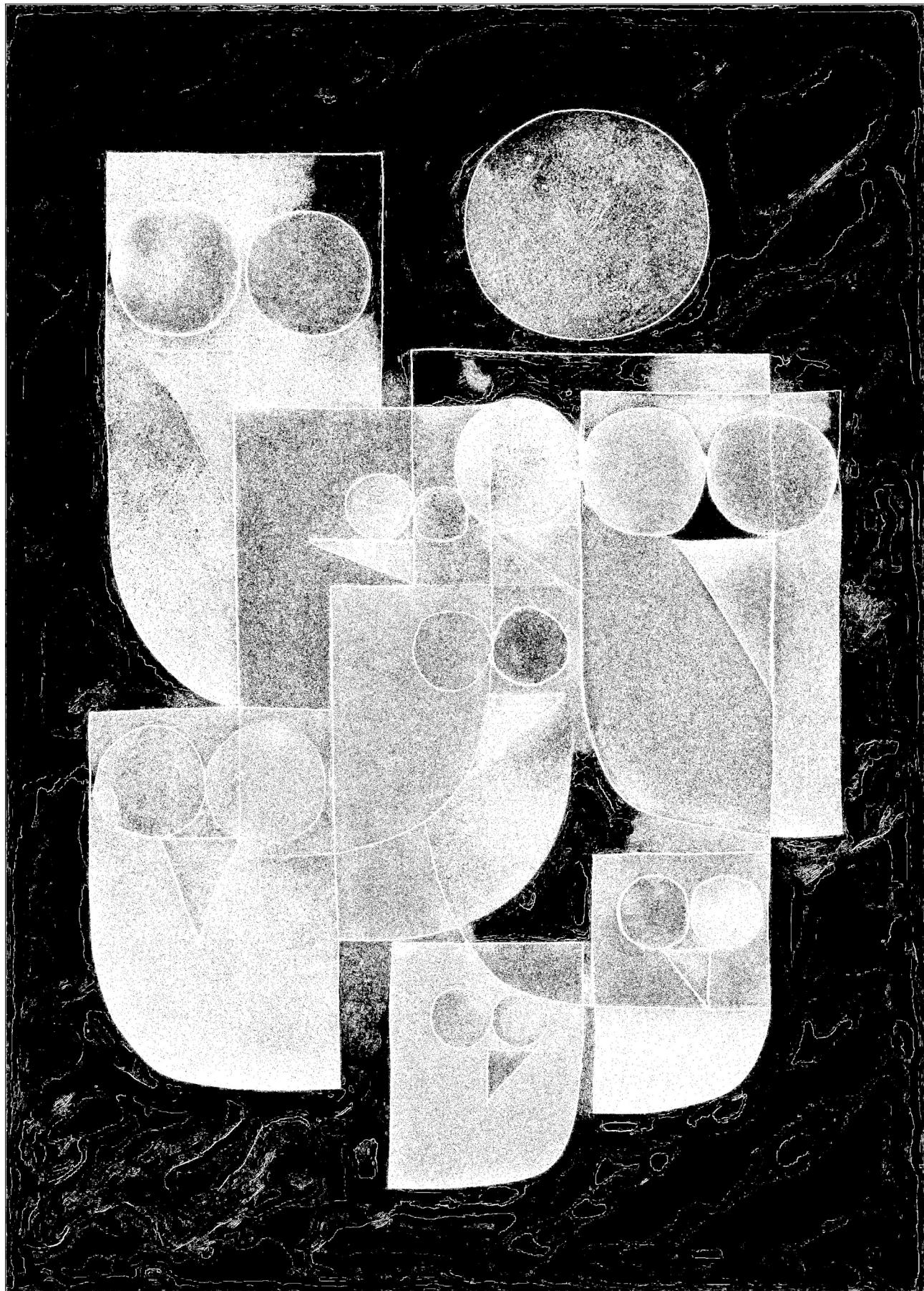
- `edge_detection_p_and_o` – funkcija koja računa p i o vrednosti, ako u okolini posmatranog piksela postoji barem jedna jedinica, postavljamo operator P na jedinicu, u suprotnom on ostaje na nuli, isto tako, ako u okolini postmatranog piksela postoji barem jedna nula, postavljamo vrednost operatora O na nulu, u suprotnom on ostaje na jedinici, na kraju oduzimanjem operatora o od operatora p dobijamo informaciju da li će piksel biti ivica ili ne
- Najvažnije funkcije u `Detector` klasi su:
 - `serial_prewitt` – koja prolazi kroz sve piksele slike, zove `prewitt_convole` i primenjuje vrednost na izlaznu sliku
 - `parallel_prewitt` – paralelna implementacija algoritma koja deli sliku na 4 manja dela sve dok ne dođe do odgovarajuće širine odnosno visine podopsega, a potom se nad svakim od manjih opsega poziva i paralelno izvršava funkcija
 - `serial_edge_detection` – koja prozlazi kroz sve piksele slike, zove `edge_detection_p_and_o` i primenjuje vrednost na izlaznu sliku
 - `parallel_edge_detection` – radi identично kao i `parallel_prewitt` samo što poziva `serial_edge_detection`
 - `run_test_nr` – koja poziva metodu, meri vreme njenog izvršavanja, i ispisuje informacije o vremenu i parametrima
 - `start_detector` – koja učitava fajlove, podešava parametre i poziva sve prethodno navedene funkcije

U nastavku su dati primeri rada ovih algoritama:

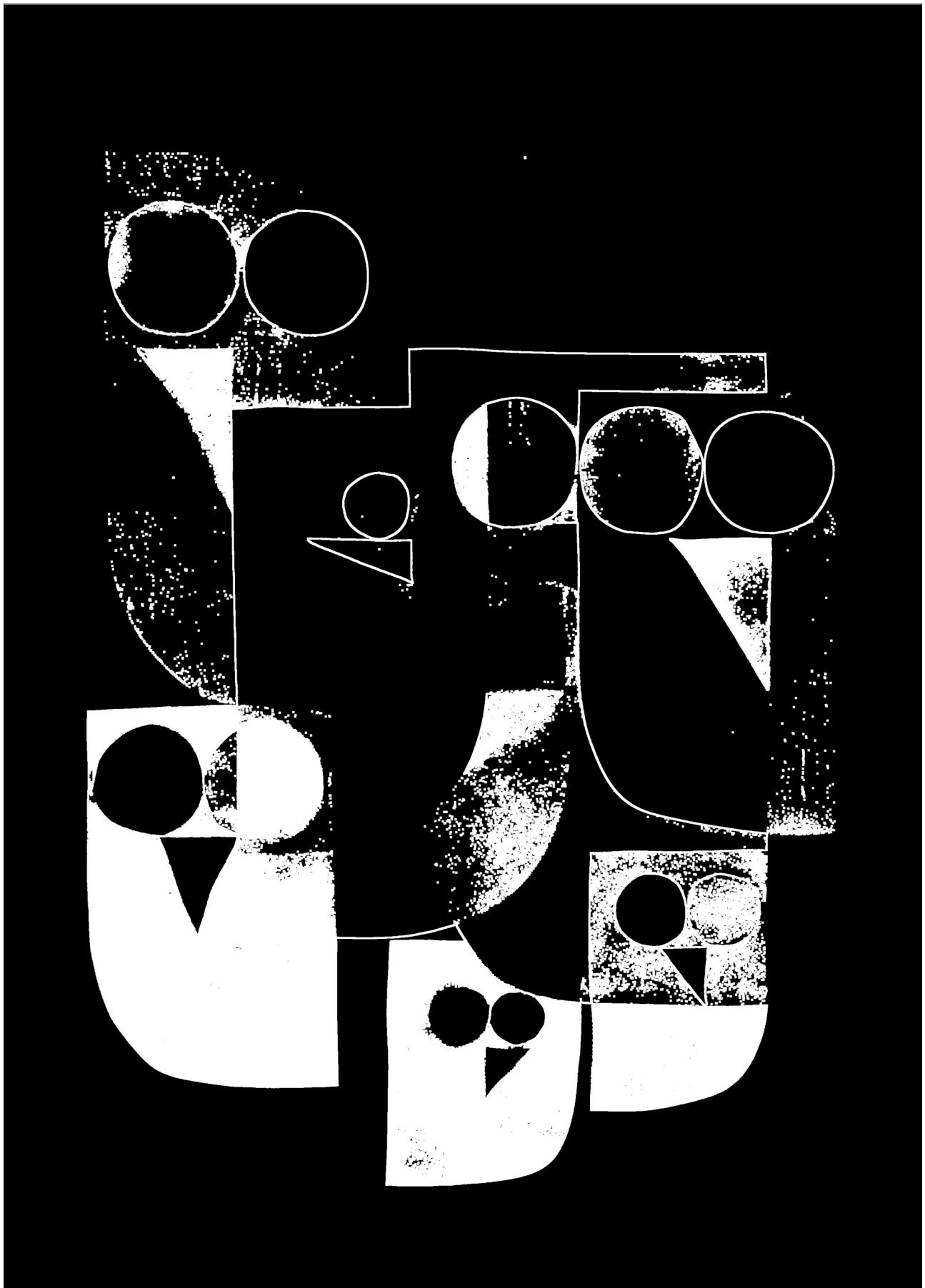
Izvorna slika



Slika dobijena koriscenjem Prewitt operatora



Slika dobijena koriscenjem detekcije ivica u okolini piksela



Izvorna slika



Slika dobijena koriscenjem Prewitt operatora



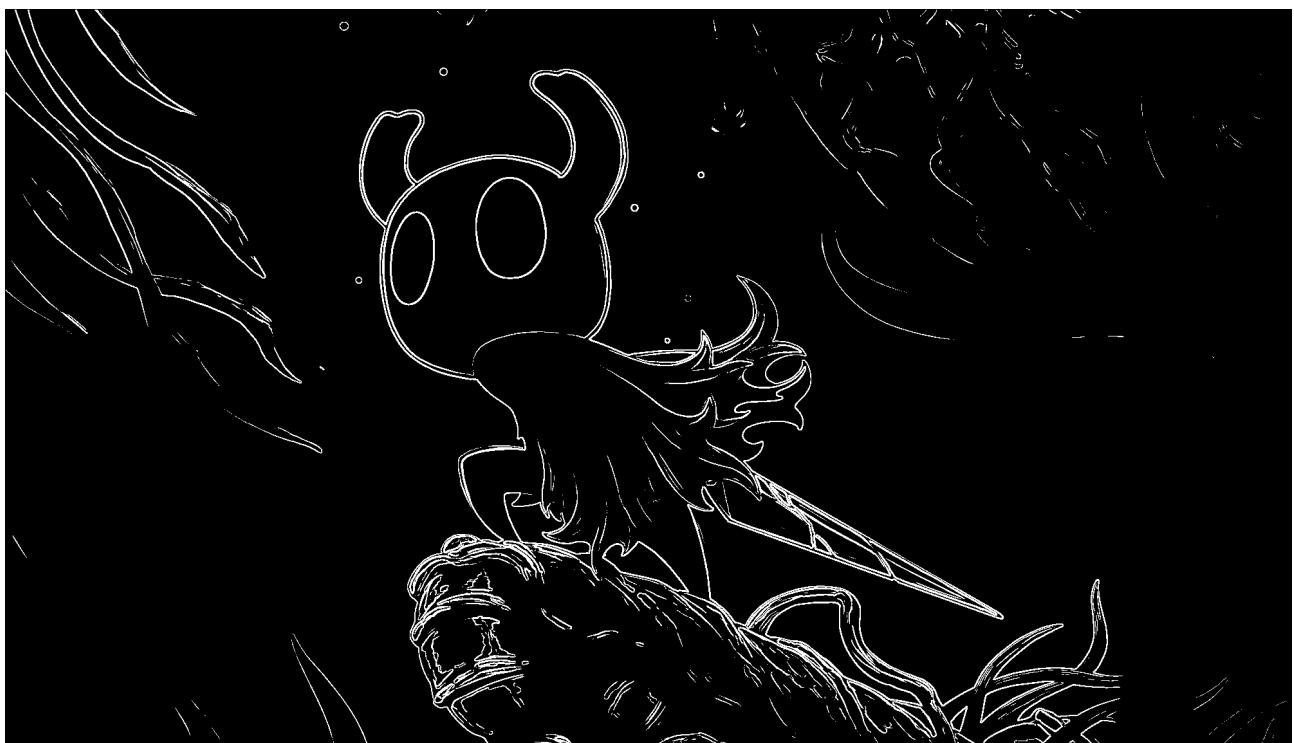
Slika dobijena koriscenjem detekcije ivica u okolini piksela



Izvorna slika



Slika dobijena koriscenjem Prewitt operatora



Slika dobijena koriscenjem detekcije ivica u okolini piksela



4 – Ispitivanje

Specifikacija računara na kom je izvršeno testiranje je:

- CPU: Ryzen 7 2700x, 8 jezgara, 3.6GHz
- RAM: 32GB
- Operativni sistem: Artix Linux
- Kernel: 6.2.11-artix1-1

Tabela 1 - Vreme(ms) potrebno za izvršenje serijskog algoritama nad različitim veličinama slike

Rezolucija slike	Prewitt: 3x3	Prewitt: 5x5	Detekcija u okolini: Udaljenost 1	Detekcija u okolini: Udaljenost 2	Detekcija u okolini: Udaljenost 3
2000x1147	109	288	92	206	370
3840x2160	400	1030	291	806	1469
2724x3816	516	1346	371	1024	1838

Iz tabele zaključujemo da je serijska implementacija Prewitt algoritma sporija u odnosu na detekciju ivica, ali to smo mogli da naslutimo s obzirom da zahteva više računskih operacija.

Tabela 2 – Vreme(ms) potrebno za izvršenje paralelnog algoritama na slici 3840x2160 piksela

Prag odsecanja	Prewitt: 3x3	Prewitt: 5x5	Detekcija u okolini: Udaljenost 1	Detekcija u okolini: Udaljenost 2	Detekcija u okolini: Udaljenost 3
100	73	151	40	98	182
200	70	151	40	98	184
300	76	158	41	104	195
400	59	164	44	109	194
500	78	160	42	104	190
600	83	167	41	106	194
700	84	165	40	105	195
800	86	159	42	104	199

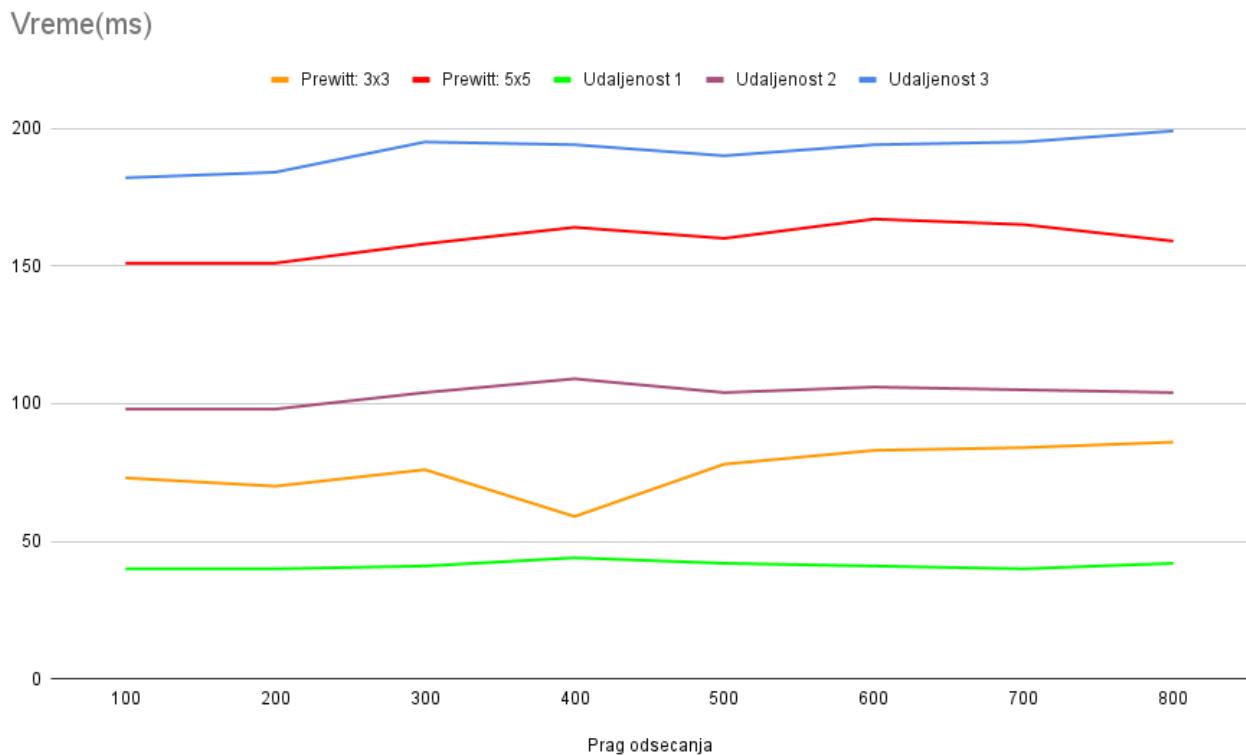
Tabela 3- Vreme(ms) potrebno za izvršenje paralelnog algoritama na slici 2000x1147 piksela

Prag odsecanja	Prewitt: 3x3	Prewitt: 5x5	Detekcija u okolini: Udaljenost 1	Detekcija u okolini: Udaljenost 2	Detekcija u okolini: Udaljenost 3
100	24	54	16	29	52
200	23	58	17	36	50
300	29	64	22	48	48
400	30	63	19	28	49
500	29	61	19	28	50
600	52	97	22	86	135
700	52	96	21	70	134
800	52	99	39	54	97

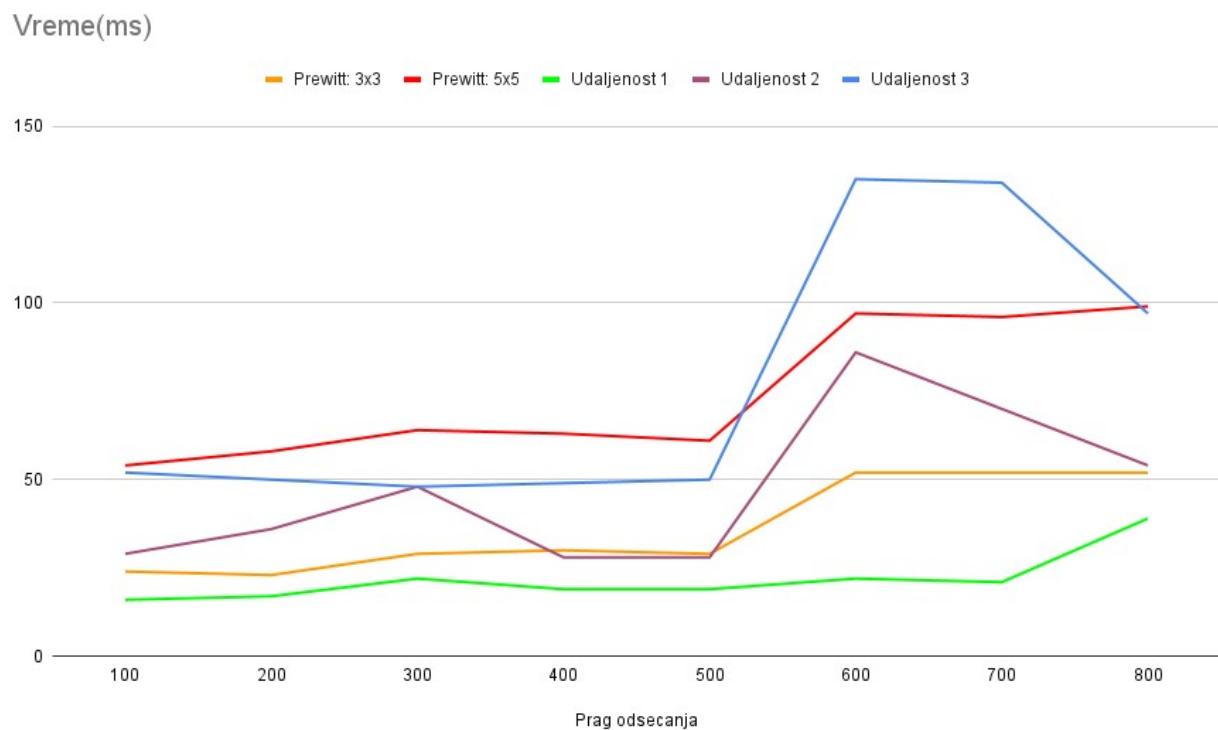
Detaljnim testiranjem dobijamo da je najbolji prag odsecanja 400. Ubrzanje koje se postiže paralelizacijom iznosi približno 7 puta, sto je očekivano ubrzanje na 8 jezgara.

U nastavku je dati grafici izvršavanja u zavisnosti od praga odsecanja za ove dve slike:

Vreme paralelnog izvršavanja u zavisnosti od praga odsecanja za sliku 3840x2160



Vreme paralelnog izvršavanja u zavisnosti od praga odsecanja za sliku 2000x1147



5 – Analiza rezultata

Paralelizacija serijskih rešenja donosi očekivano ubrzanje od približno 7 puta na 8 jezgara. Iako ovo ubrzanje može biti zanemarljivo za obradu jedne slike koja se izvršava za svega nekoliko sekundi, postaje korisno kada je potrebno obraditi veći broj slika. U takvim slučajevima, paralelizacija može uštedeti mnogo vremena.

Primetili smo da je algoritam za detekciju ivica u okolini sa udaljenošću 3 najsporiji. Povećanje matrice, bilo da se radi o Prewitt operatoru ili algoritmu za detekciju ivica u okolini, dovodi do povećanja vremena izvršavanja od 2 do 2.5 puta. Za normalnu veličinu slike i udaljenost 1 kod algoritma za detekciju ivica u okolini piksela, postiže se najbrže vreme izvršavanja, što je očekivano.

Analizom dobijamo da je Prewitt operator veličine 3×3 je idealan za većinu slika. Kod određenih slika sa nejasnim ivicama je bolje koristiti Prewitt operator veličine 5×5 , međutim, u svim drugim slučajevima, korišćenje Prewitt operatora veličine 5×5 rezultira u skoro potpuno beloj slici.