PARALELNO PROGRAMIRANJE

-DETEKCIJA IVICA U OKVIRU SLIKE-

Lazar Milanović SV15/2020

1 - Analiza problema

Detekcija ivica na slici je vitalna operacija u procesiranju slika, mašinskom učenju i digitalnoj obradi slike uopšte. Neophodno je razlikovati objekte na slici jedne od drugih. To se postiže posmatranjem okolina oko piksela i primenom određenih operacija kako bi se što jasnije ekstrahovali fizički objekti sa slike.

2 - Koncept rešenja

Posmatraćemo dva pristupa rešenja ovog problema; takozvani Prewitt operator i detekciju u okolini piksela (Edge detection).

Prewitt operator se bavi filterima koji su zapravo kvadratne matrice veličine 3x3 piksela, a ređe 5x5 piksela. Matrica se preklopi preko posmatranog piksela tako da bude centrirana preko piksela. Potom se izvrši konvolucija tog filtera sa matricom koja čini okolinu piksela veličine koja odgovara veličini filtera. Ovaj proces se izvršava jednom za horizontalni i jednom za vertikalni filter. Potom se saberu apsolutne vrednosti ovih rezultata i, u zavisnosti od predefinisanog praga, piksel se oboji u crnu odnosno belu boju. Kada se ovaj proces izvrši nad svakim pikselom na slici, dobija se filtrirana slika koja sadrži jasne konture ivica objekata na slici.

Edge detection je algoritam koji posmatra okolinu određenog piksela kako bi utvrdio da li je taj piksel ivica nekog objekta ili nije. Koriste se dva operatora, P i O. Ovaj algoritam se u teoriji izvršava iz dva prolaza kroz sliku, ali programsko rešenje se može napisati tako da postoji samo jedan prolaz kroz sliku. Prvim prolazom kroz sliku svaku vrednost piksela iznad predefinisanog praga promenimo na jedinicu, svaku vrednost ispod praga promenimo na nulu. Drugim prolazom kroz sliku posmatramo okolinu piksela, koju prethodno zadajemo. Ako u okolini posmatranog piksela postoji barem jedna jedinica, postavljamo operator P na jedinicu, u suprotnom on postaje nula. Analogno tome, ako u okolini posmatranog piksela postoji barem jedna nula, postavljamo vrednost operatora O na nulu, u suprotnom on postaje jedinica. Primenom XOR operacije nad operatorima P i O dobijamo informaciju da li je piksel ivica nekog objekta ili nije.

3 - Programsko rešenje

Prilikom implementacije algoritama za detekciju slike, korišćena je biblioteka EasyBMP u svrhu konvertovanja slika iz bitmap formata u matrice vrednosti piksela u rasponu od 0 do 255. Iz biblioteke je korišćeno sledeće:

BitmapRawConverter - klasa koja služi za otvaranje bitmap slike u svrhu čitanja i pisanja. Osigurava da će slika biti tretirana kao crno-bela slika.

Što se konkretnog projekta tiče, korišćena su dva imenska prostora; image_filter i filter_test. U okviru image_filter imenskog prostora se nalazi klasa Filter koja služi za primenu algoritama na određenu sliku. U okviru filter_test imenskog prostora se nalaze jedinični testovi dveju metoda koje su srž algoritama, kao i stress test filtera nad raznim cutoff vrednostima, veličinama filtera i udaljenosti. Ključna metoda prewitt algoritma je image_filter::prewitt_convolve koja predstavlja konvoluciju matrice filtera i matrice okoline piksela na slici. Ključna metoda edge detection algoritma je image_filter::calculate_p_o koja procenjuje da li je traženi piksel ivica objekta ili nije na osnovu operatora P i O. Nad obe funkcije je izvršen manji broj jediničnih testova kako bi se osigurao njihov ispravan rad.

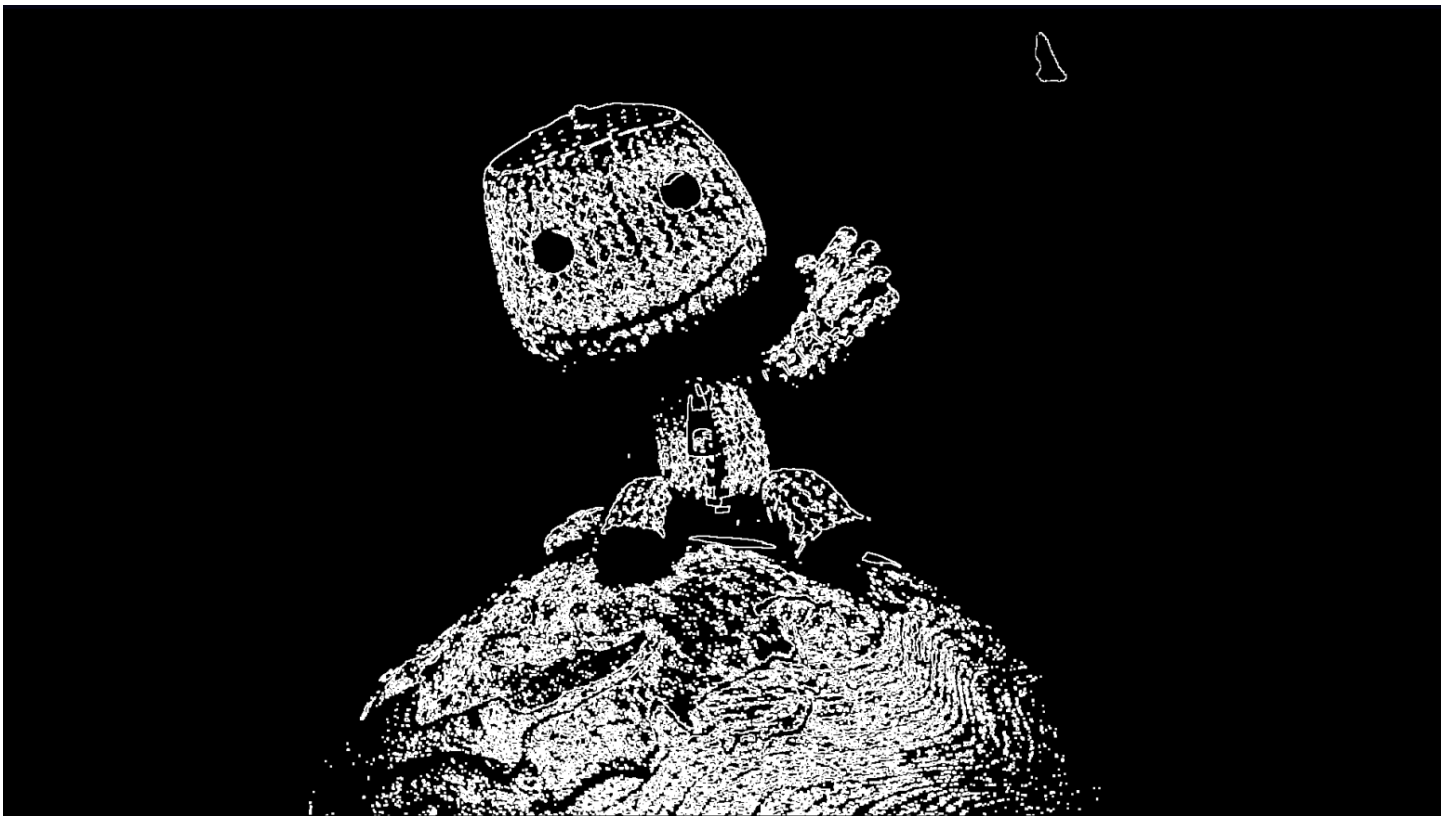
Kako su serijske implementacije ovih algoritama slične, to ih je moguće apstrahovati metodom image_filter::Filter::apply_serial u kojoj se navodi željena metoda, a filter sam prosleđuje pokazivač na funkciju koja bi trebalo da se izvrši. Rad ovih funkcija je sakriven od krajnjeg korisnika radi očuvanja integriteta algoritama. Analogno ovome, rešenje za paralelne implementacije je slično; implementacije obe funkcije su sakrivene od korisnika, a image_filter::Filter::apply_parallel vrši pozivanje odgovarajućeg paralelnog algoritma.

Serijska implementacija ovih algoritama prolazi kroz sve piksele slike u određenom opsegu širine i visine i primenjuje odgovarajuće funkcije nad njima.

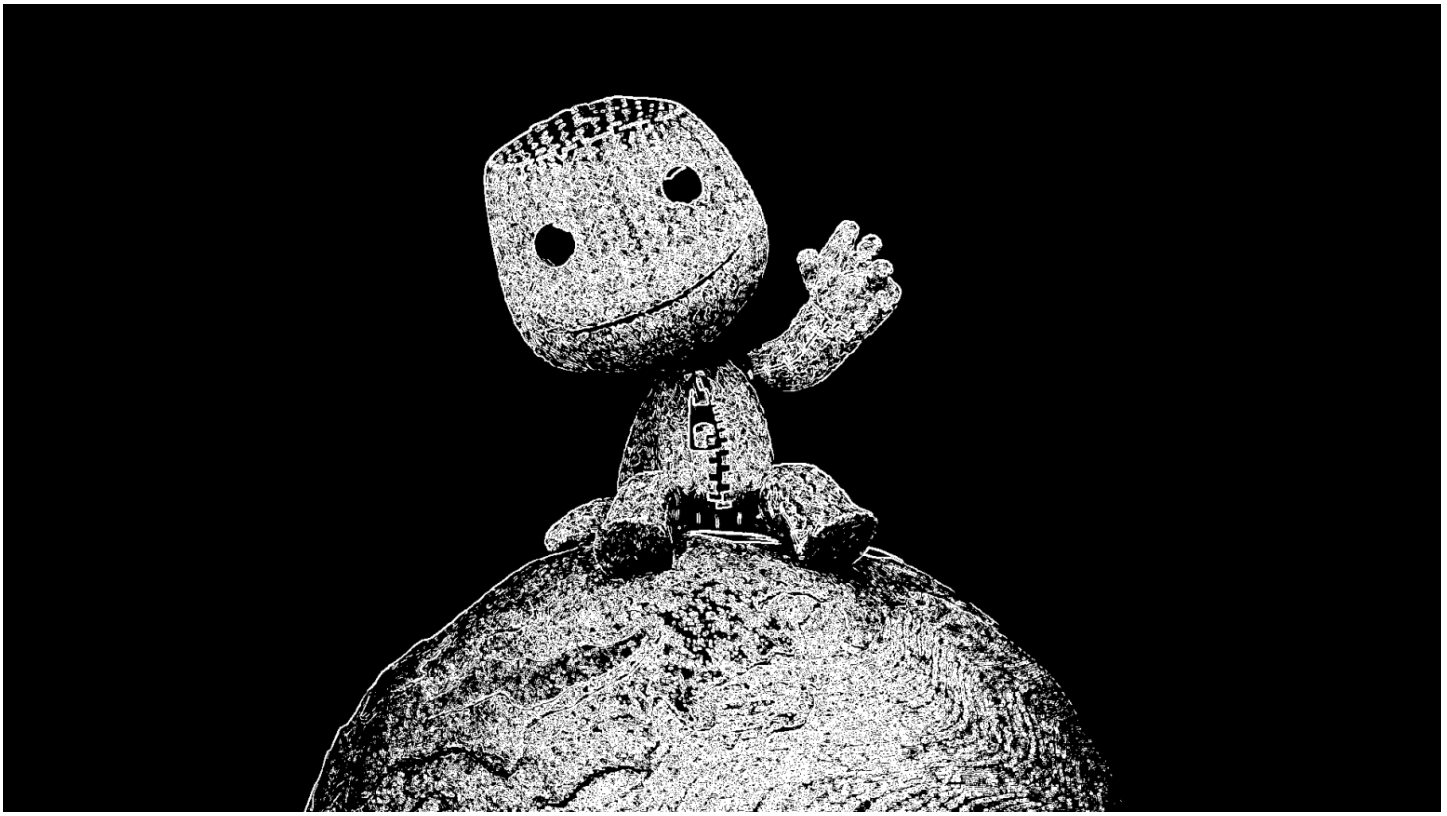
Paralelna implementacija ovih algoritama se zasniva na deljenju opsega na 4 manja dela sve dok se ne dođe do odgovarajuće širine odnosno visine podopsega. Potom se nad svakim od manjih opsega pozivaju odgovarajuće serijske funkcije i to se izvršava u paraleli.

Primeri rada ovih algoritama:

Izvorna slika



Edge detection algoritam



Prewitt operator algoritam

4 - Ispitivanje

Specifikacija računara na kom je izvršeno testiranje je sledeća:

Procesor: Intel i5-9400F, 6 jezgara, overclockovan na 4.1GHz
RAM: 16GB
Operativni sistem: Arch Linux
Kernel: 5.17.9-arch1-1

Tabela 1 - Vreme (ms) neophodno za serijsko izvršenje algoritama nad različitim veličinama slike

Veličina slike	Edge - Distance: 1	Edge - Distance: 2	Edge - Distance: 3	Prewitt - 3x3	Prewitt - 5x5
Ogromna slika	355	874	1706	528	1336
Normalna slika	76	189	375	105	263

Možemo primetiti da je za serijsku implementaciju ova dva algoritma, Prewitt operator sporiji za nijansu, što je i za očekivati s obzirom da zahteva da se odradi više računskih operacija od edge detection algoritma.

Tabela 2 - Vreme (ms) neophodno za paralelno izvršenje na ogromnoj slici u zavisnosti od veličine cutoffa

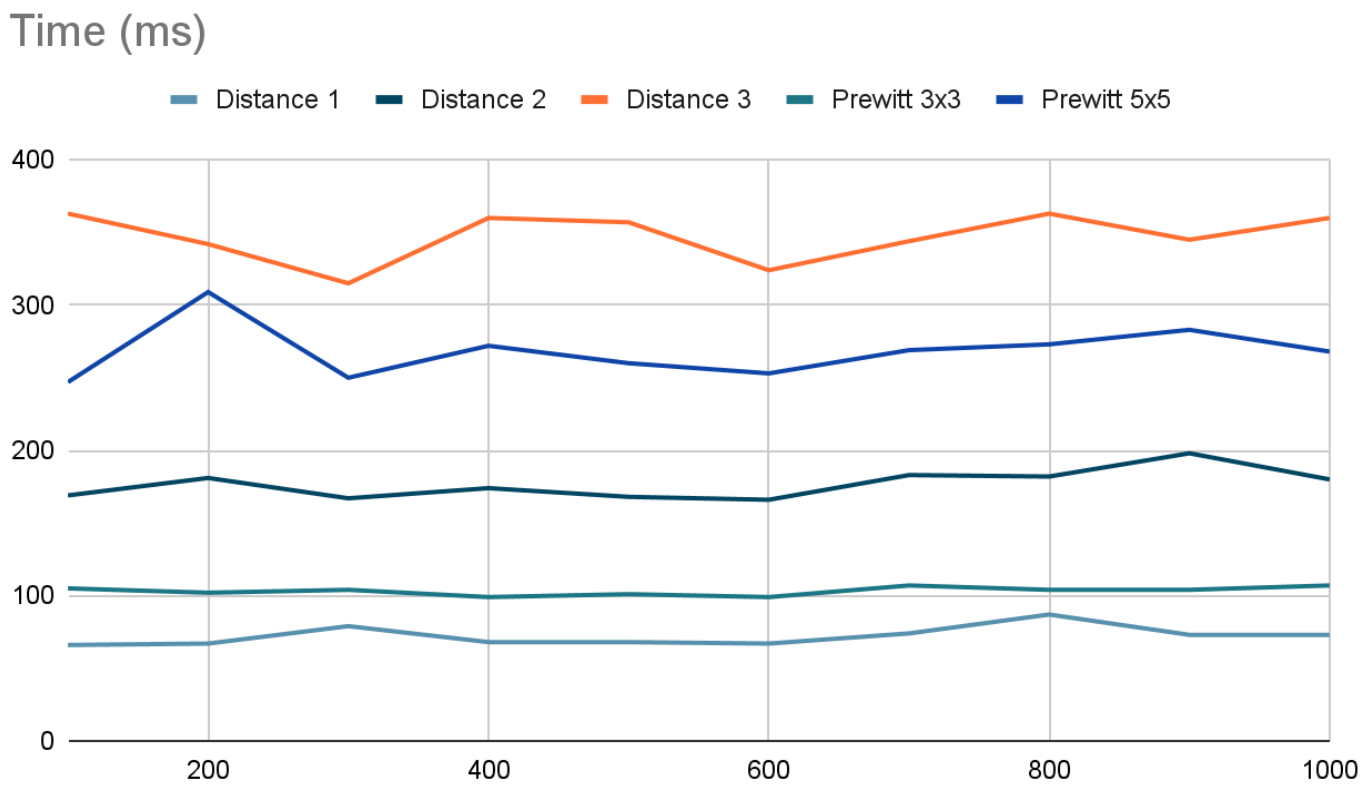
Cutoff (u pikselima)	Edge - Distance: 1	Edge - Distance: 2	Edge - Distance: 3	Prewitt - 3x3	Prewitt - 5x5
100	66	169	363	105	247
200	67	181	342	102	309
300	79	167	315	104	250
400	68	174	360	99	272
500	68	168	357	101	260
600	67	166	324	99	253
700	74	183	344	107	269
800	87	182	363	104	273
900	73	198	345	104	283
1000	73	180	360	107	268

Kroz veći broj sličnih testova utvrđeno je da je idealan cutoff za ove algoritme 600 piksela, odnosno, u ovom konkretnom slučaju, deljenje slike na 16 jednakih delova pre paralelizacije. Ubrzanje koje se postiže paralelizacijom na ovaj način iznosi približno 5 puta, što je očekivano ubrzanje na 6 jezgara.

Tabela 3 - Vreme (ms) neophodno za paralelno izvršenje na normalnoj slici u zavisnosti od veličine cutoffa

Cutoff (u pikselima)	Edge - Distance: 1	Edge - Distance: 2	Edge - Distance: 3	Prewitt - 3x3	Prewitt - 5x5
100	14	32	64	19	47
200	15	34	69	18	47
300	14	38	72	20	50
400	16	37	73	20	53
500	16	36	72	20	51
600	19	49	97	27	67
700	19	49	97	27	68
800	19	49	99	27	67
900	19	49	98	27	68
1000	19	49	98	27	67

Testiranjem na manjoj slici, dobijamo ponovo očekivano ubrzanje od približno 5 puta. Idealan cutoff je za nijansu manji u ovom slučaju i iznosi 500px, što ima smisla obzirom da je slika manjih dimenzija. Ovaj cutoff ponovo odgovara deljenju slike u 16 jednakih delova.



Vreme paralelnog izvršavanja u zavisnosti od cutoffa

5 - Analiza rezultata

Paralelizacijom serijskih rešenja se dobilo očekivano ubrzanje od približno 5 puta na 6 jezgara. Ovo ubrzanje je na primeru jedne slike trivijalno, jer je vreme izvršenja svega deo sekunde. Međutim, ako je neophodno obraditi veći broj slika, paralelizacija tada ima smisla i sačuvaće mnoge sekunde, ako ne i sate.

Primećujemo da je edge detection sa udaljenošću 3 najsporiji algoritam. To i ima smisla obzirom da se posmatra okolna matrica veličine 7x7. Povećanjem matrica bilo prewitt operatora ili udaljenosti od tačke se vreme izvršavanja povećava 2 - 2.5 puta. Za normalnu veličinu slike i udaljenost 1 kod edge detection algoritma je najbrže vreme izvršavanja, što je očekivano.

Prewitt operator veličine 3x3 je idealan za većinu slika. Kod određenih slika sa nejasnim ivicama je bolje koristiti operator veličine 5x5, međutim, u svim drugim slučajevima, korišćenje Prewitt operatora veličine 5x5 rezultira u gotovo beloj slici.

Edge detection algoritam sa udaljenošću 1 je idealan za većinu slika. Povećavanje distance rezultira u “pikselizovanijoj” slici, slično kao kod metoda za peglanje slike i nije pogodno za korišćenje.