Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Student: Ivan Martić SV77/2020

Predmet: Paralelno programiranje

Tema projektnog zadatka: Detekcija ivica unutar slike

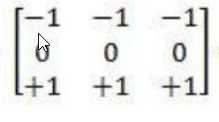
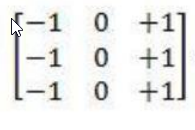
# Opis problema

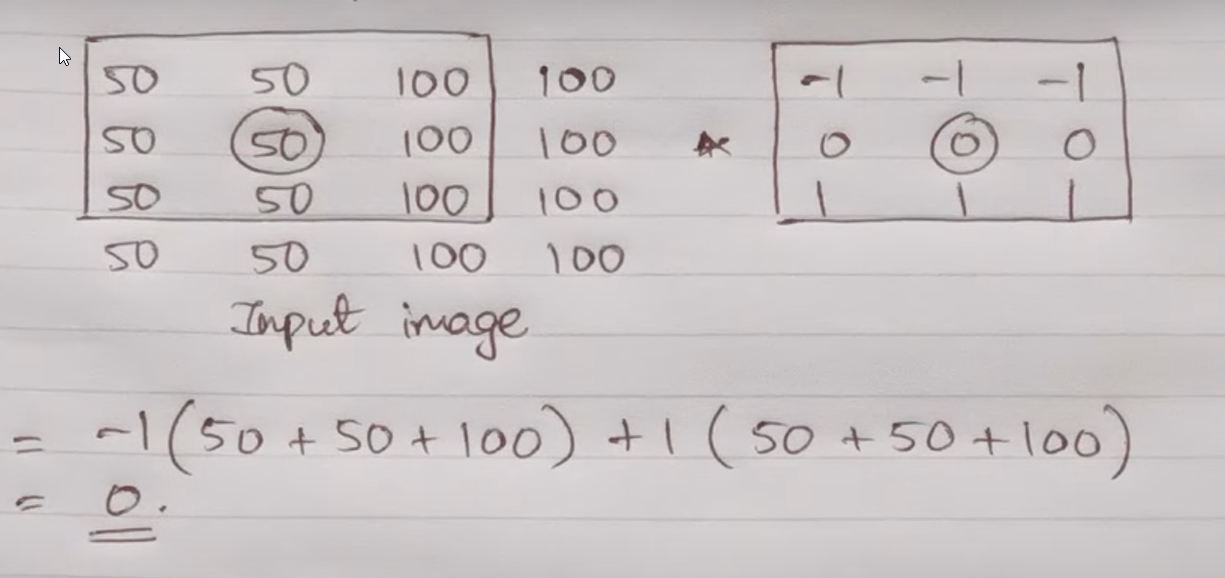
Problem koji se rešava napisanim kodom jeste detekcija ivica na slici. Slika za koju detektujemo ivice je data u obliku matrice, svaka ćelija matrice predstavlja piksel. Svaki piksel koji detektujemo da je ivica, bojimo ga belom bojom (vrednost ćelije postavimo na 0 ) dok ostale piksele bojimo crnom bojom (vrednost ćelije postavimo na 255).

# Koncept resenja

Detektovanje ivica unutar slike možemo postići imlementacijom **prewit metode** ili **metodom detekcije ivica.**

**Prewitt metoda** svodi se na množenje podmatrice nase slike koja sadrži podatke o vrednostima boja nase slike sa kvadratnom matricom dimenzija (3x3, 5x5, 7x7), koja predstavlja filter. Ulaznu sliku filtriramo operatorima za horizontalne, odnosno vertikalne ivice. Rezultujuću tacku G izlazne slike računamo zbirom apsolutnih vrednosti horizontalne Gx i vertikalne komponente Gy. Nakon toga skaliramo dobijenu vrednost na osnovu globalnog praga T=128, čime se postiže dobijanje crne slike sa belim ivicama. Dobijenu vrednost upisujemo na odgovarajući piksel izlazne slike.

 **Gore navedene slike predstavljaju horizontalni, odnosno vertikalni filter.**



**Gore navedena slika prikazuje proces filtriranja slike sa vertikalnom komponentom.**

**Metoda detektovanja ivica**, prvi korak jeste postavljanje vrednosti ulazne slike odsecanjem za prag T=128 na 0 ili 255. Ovo ce rezultovati crno-belom slikom. Potom se pretrazuje okolina svake tacke, za pretrazivanje cemo koristiti dve promenljive P i O, za koje važi:

* P(i,j) = 1, ako u okolini tačke postoji tačka sa vrednošću 1
* P(i,j) = 0, ako u okolini tačke ne postoji tačka sa vrednošću 1
* O(i,j) = 0, ako u okolini tačke postoji tačka sa vrednošću 0
* O(i,j) = 1, ako u okolini tačke ne postoji tačka sa vrednošću

Nakon toga vrednost rezultujuće tacke racunamo na osnovu formule |P(i,j) – O(i,j)|

# Programsko rešenje

* void **calculate\_serial\_prewitt**

metoda vrši detekciju ivica serijiski prewitt metodom. Metoda računa piksele red po red.

* void **calculate\_row\_prewitt**

metoda računa vrednosti piksela za odredjeni red izlazne slike na osnovu reda ulazne slike.

* void **calculateG**

metoda računa G rezultujuću tačku izlazne slike na nacin opisan gore u tekstu.

* int **remove\_color**

metoda vrši skaliranje rezultujuće tačke na osnovu T=128.

* void **calculate\_parallel\_prewitt**

metoda vrši detekciju ivica paralelno tako sto svaki task računa zasebno jedan red.

* void **calculate\_serial\_edge\_detection**

metoda vrši detekciju ivica serijiski metodom detekcija ivica. Metoda računa piksele red po red.

* void **make\_image\_black\_white**

metoda svodi ulaznu sliku (sliku u boji) na crno-belu sliku.

* void **calculate\_row\_edge\_detection**

metoda računa vrednosti piksela za odredjeni red izlazne slike na osnovu reda ulazne slike.

* int **calculatePO**

metoda računa P i O na način opisan gore i svodi ga na odgovarajući opseg.

* void **calculate\_parallel\_edge\_detection**

metoda vrši detekciju ivica paralelno tako što svaki task racuna zasebno jedan red.

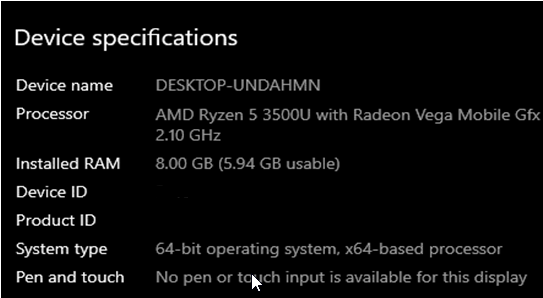
**Koriscene biblioteke su:**

**„tbb/task\_group.h“** => koriscena je za kreiranje zadataka, odnosno za samu paralelizaciju programa

**„tbb/tick\_count.h“** => koriscena je za merenje vremena, odnosno koje vreme je potrebno da se program izvrsi

# Ispitivanje

Specifikacije uređaja na kojem je testiran program:



Brzina izvršavanja (debug režim):

**Prewitt**

Img1.bmp

|  |  |  |
| --- | --- | --- |
| Dimenzija filtera | Prewitt serijski | Prewitt paralelno |
| 3x3 | 0.121787s | 0.066686 |
| 5x5 | 0.214719s | 0.0737708s |
| 7x7 | 0.388358s | 0.131549s |

img2.bmp

|  |  |  |
| --- | --- | --- |
| Dimenzija filtera | Prewitt serijski | Prewitt paralelno |
| 3x3 | 0.268637s | 0.0992872s |
| 5x5 | 0.590959s | 0.208552s |
| 7x7 | 1.11084s | 0.266421s |

img3.bmp

|  |  |  |
| --- | --- | --- |
| Dimenzija filtera | Prewitt serijski | Prewitt paralelno |
| 3x3 | 0.299514s | 0.13618s |
| 5x5 | 0.562747s | 0.161747s |
| 7x7 | 1.00715s | 0.239931s |

color.bmp

|  |  |  |
| --- | --- | --- |
| Dimenzija filtera | Prewitt serijski | Prewitt paralelno |
| 3x3 | 1.55819s | 0.492843s |
| 5x5 | 2.51051s | 0.578688s |
| 7x7 | 4.40622s | 0.967155s |

**Detekcija ivica**

|  |  |  |
| --- | --- | --- |
| Naziv slike | Detekcija ivica serijski | Detekcija ivica paralelno |
| img1.bmp | 0.115114s | 0.0450574s |
| img1.bmp | 0.313284s | 0.124339s |
| img1.bmp | 0.42628s | 0.142717s |
| img2.bmp | 0.25923s | 0.0727783s |
| img2.bmp | 0.594955s | 0.1309342s |
| img2.bmp | 0.831942s | 0.191195s |
| Img3.bmp | 0.244598s | 0.0873501s |
| Img3.bmp | 0.865361s | 0.200085s |
| Img3.bmp | 1.1852s | 0.332212s |
| color.bmp | 1.15017s | 0.320732s |
| color.bmp | 3.10518s | 0.825293s |
| color.bmp | 4.84391s | 1.47748s |

**Dimenzije slika**

|  |  |
| --- | --- |
| Naziv slike | Dimenzije |
| img1.bmp | 1601 x 664 |
| img2.bmp | 1920 x 1080 |
| Img3.bmp | 1920 x 1280 |
| color.bmp | 3888 x 2592 |

**Grafici metode prewitt**

**Grafici metode detekcije ivica**

# Analaiza rezultata

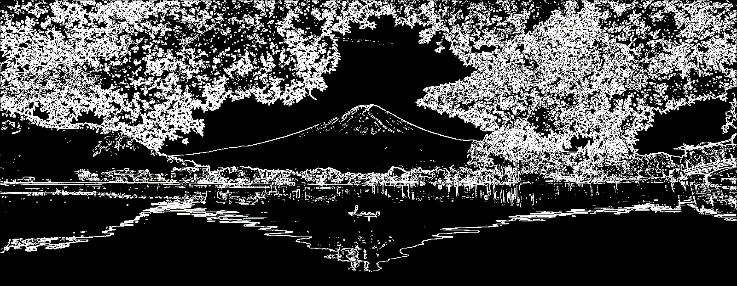
Dakle jasno možemo primetiti da će algoritmi sprorije raditi na vićim nego na manjim slikama. Kao što možemo vidieti iz priloženih tabela, veličina okoline u algoritmu koji koristi metodu dektcija ivica i veličina Prewitt operatora u drugom algoritm znajčajno utiču na brzinu izvršavanja.

Povećanjem veličine prewitt operatora koji se uzima u obzir prilikom obrade prewitt algoritmom, može se videti da se vreme obrade povećava. Na prva dva grafika možemo videti kako se ubrzanje i vreme menjaju u zavisnosti od velicine prewitt operatora i same slike. Takođe može se videti da veličina slike drastično utiče na ubrzanje, gde je za veće slike ubrzanje izraženije, dok kod manjih nije baš toliko izraženo.

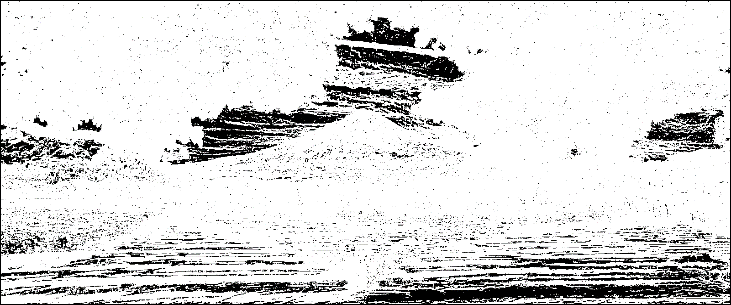
Slično se odnosi i na drugi algoritam.

Sto se tiče kvaliteta rezultata, neki od primera će biti prikazani u nastavku.

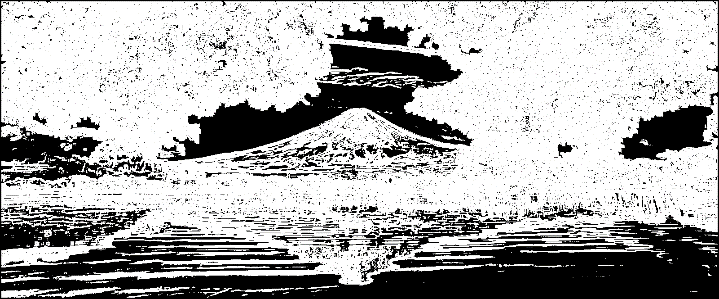
**Metod detekcije ivica nivo 0 i Prewitt 3x3 .**



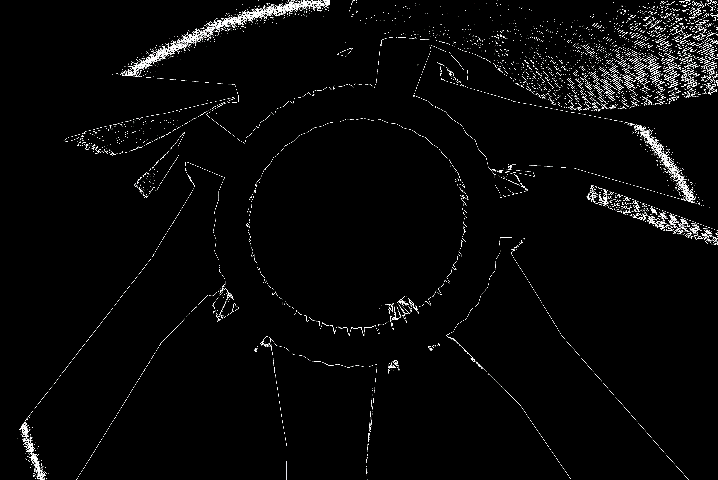
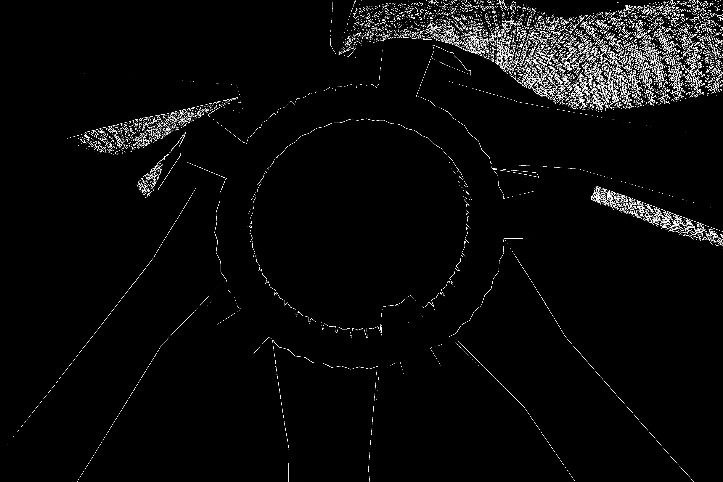
**Metod detekcije ivica nivo 1 i Prewitt 5x5.**



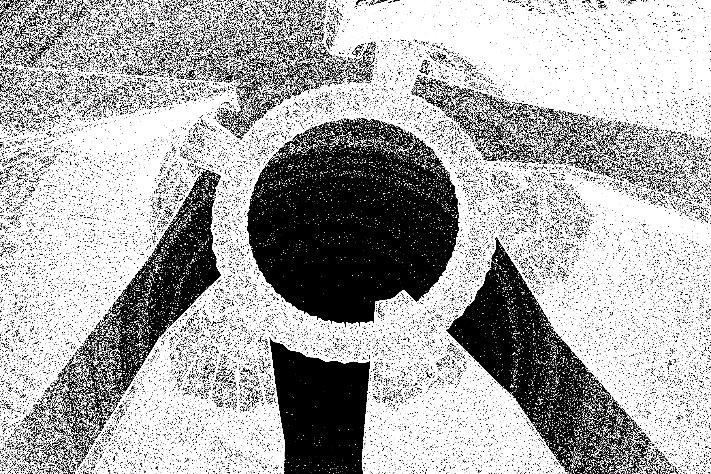
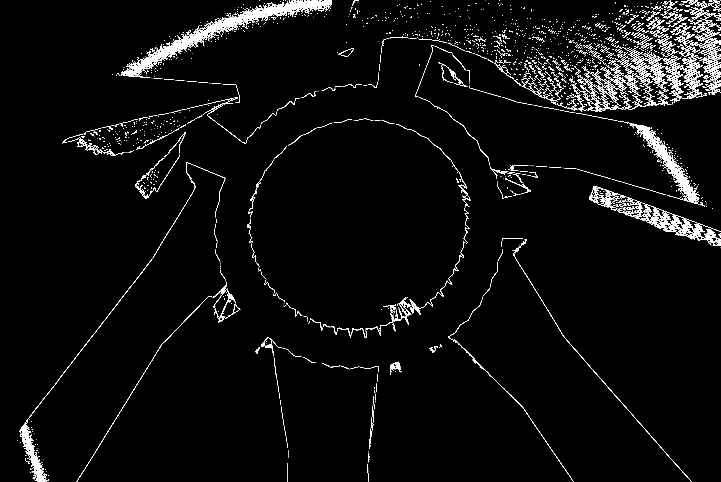
**Metod detekcije ivica nivo 2 i Prewitt 7x7.**



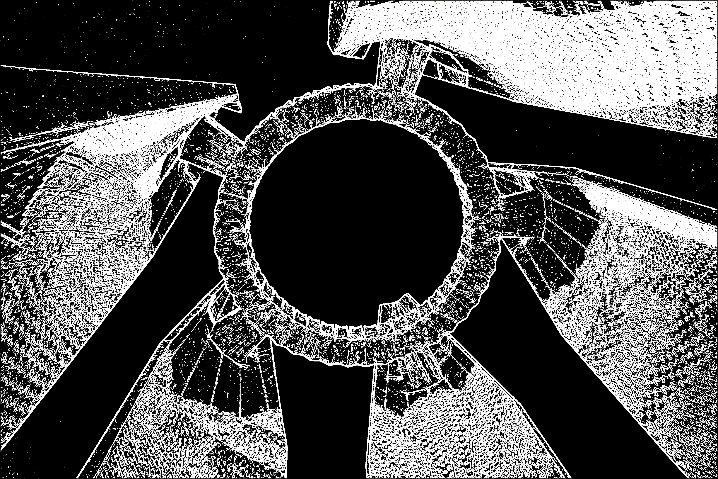
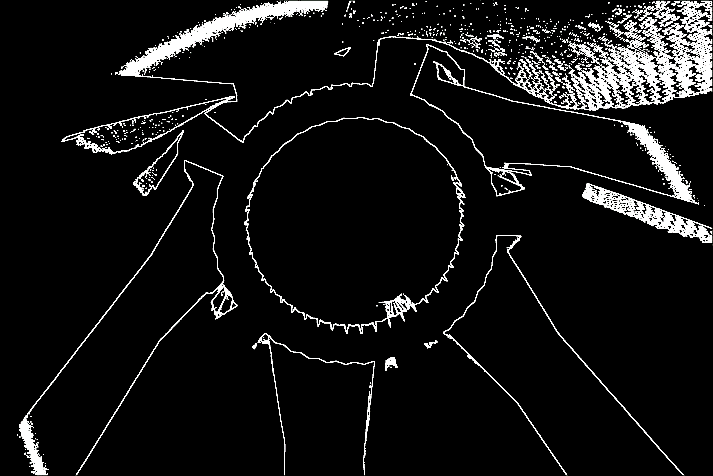
**Metod detekcije ivica nivo 0 i Prewitt 3x3 .**

**Metod detekcije ivica nivo 1 i Prewitt 5x5.**



**Metod detekcije ivica nivo 2 i Prewitt 7x7.**



Kao što se može videti iz datih slika, algoritam koji koristi Prewitt operator se pokazao značajno boljim od osnovnog algoritma. Najznačajnija stvar kod Prewitt operatora je sam operator tj. vrednosti koje on sadrži. Rezultat koji se dobija sa Prewitt operatorom veličine 5x5 gde su i brojevi unutar njega dosta različitij je dosta lošiji u odnosu na verziju 3x3 ili verziju 7x7.

Moj zaključak bi bio da se koristi što manji Prewitt operator, jer je sama obrada brža od verzije sa većim operatorom, a iz prikazanih slika se može videti da su rezultati bolji.

Što se tiče ubrzanja, ja smatram da su rezultati dobri. Na nekoj boljoj konfiguaciji bi verovatno bili bolji. Ono o čemu se još treba voditi računa je i veličina slike zbog ranije pomenutih razloga.