Paralelizam i konkurentnost

Ak. god. 2020./2021.

Image processing po uzoru na MatLab i optimizacija

Autor: Robert Đaković

Nastavnik: Josip Knezović

1. Opis projektnog zadatka

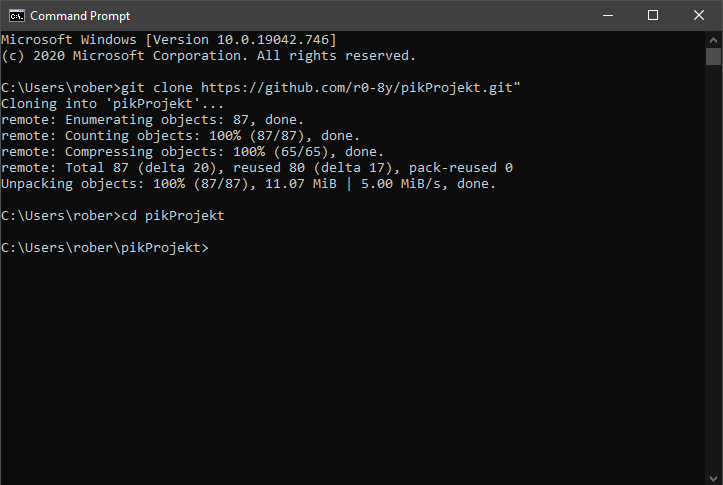
U okviru projekta za predmet Paralelizam i konkurentnost cilj mi je bio napraviti jednostavnu “aplikaciju” pomoću koje bi korisnik izabrao sliku iz priložene datoteke, koja je sastavni dio ovog projekta, te bi ju filtrirao s jednim od devet ponuđenih filtara. Naravno, zadatak mi je bio i pokušati optimizirati svoj kod uporabom OpenMP-a, te ako je moguće, filtrirati sliku koristeći grafičku karticu.

1. Upute za korištenje i testiranje
2. Sve što trebate je laptop sa c++ compilerom i pristup internetu
3. Otvorite “cmd” (pretražite “cmd” u tražilici ili stisnite “Windows key + X”, te nakon toga “Windows PowerShell”

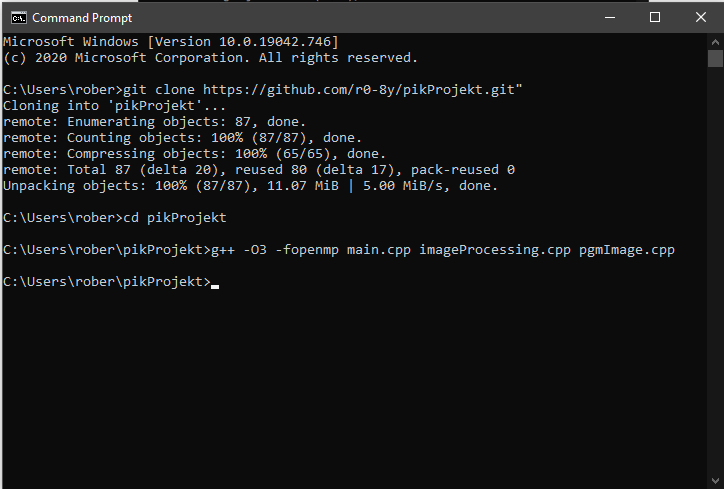
Graphical user interface, text, application, email

Description automatically generated

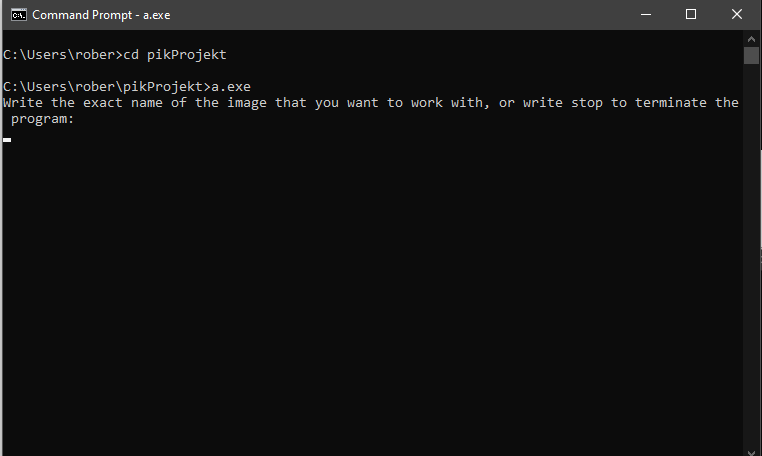
1. Pozicionirajte se u željeni direktorij te upišite “git clone <https://github.com/r0-8y/pikProjekt.git>”
2. Pozicionirajte se u upravo stvoren direktorij koji predstavlja sve datoteke potrebne za pokretanje aplikacije



1. Prevedite kod pomoću komande: g++ -O3 -fopenmp main.cpp imageProcessing.cpp pgmImage.cpp

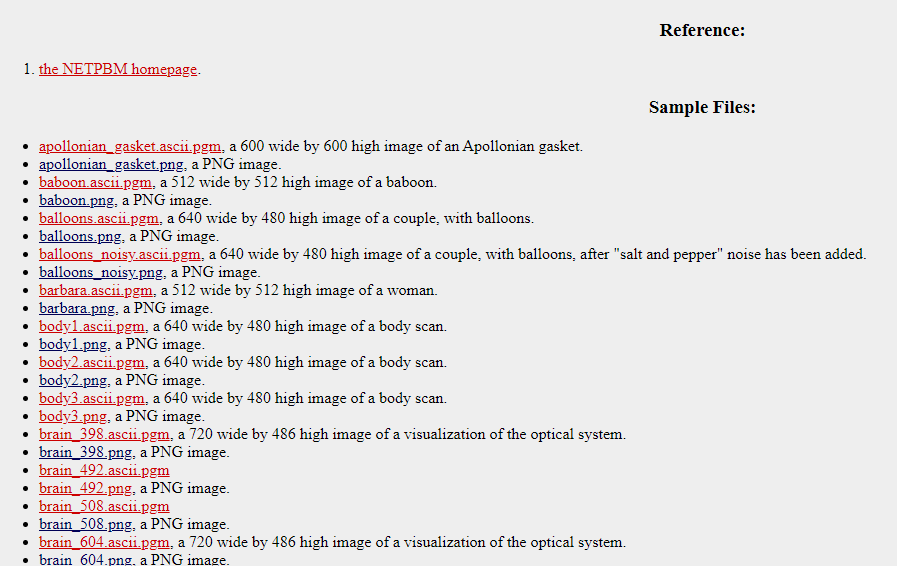


1. Pokrenite executable file sa naredbom a.exe



1. Sada će program malo pričekati, želim da otvorite tri stvari:
2. Otvorite direktorij “pikProjekt” u proizvoljnom editoru (npr. Visual Studio Code)
3. Otvorite stranicu: <https://people.sc.fsu.edu/~jburkardt/data/pgma/pgma.html>

Na ovoj stranici ako se spustite niže vidjet ćete listu “Sample Files” koja predstavlja sve slike koje sam koristio u projektu, njihov “.png” format za pregled, te “.pgm” format koji se koristio prilikom obrade (više o ovim formatima slobodno pročitajte na stranicama Wikipedije, da ne duljim ovaj dokument). Sve te slike u “.pgm” formatu nalaze se također u projektu. Izaberite jednu sliku klikanjem na instancu u “.png” formatu te joj dobro zapamtite ime!



1. Otvorite stranicu:

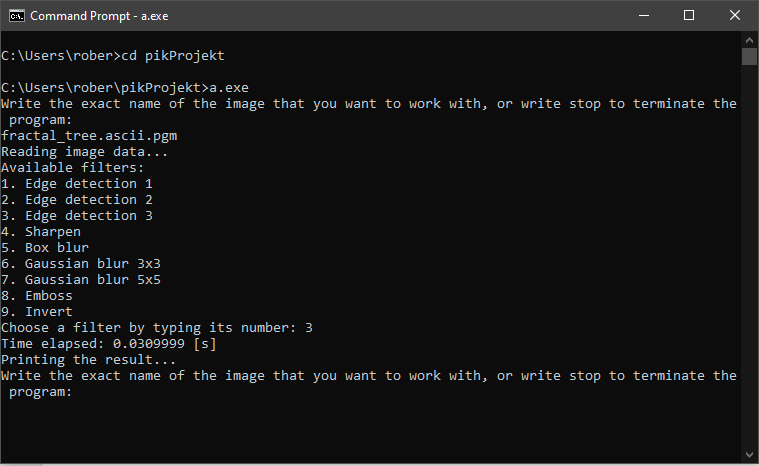
<https://www.kylepaulsen.com/stuff/NetpbmViewer/>

Ova stranica nam služi da pregledamo rezultat obrade.

1. Ja sam odabrao sliku “fractal\_tree”

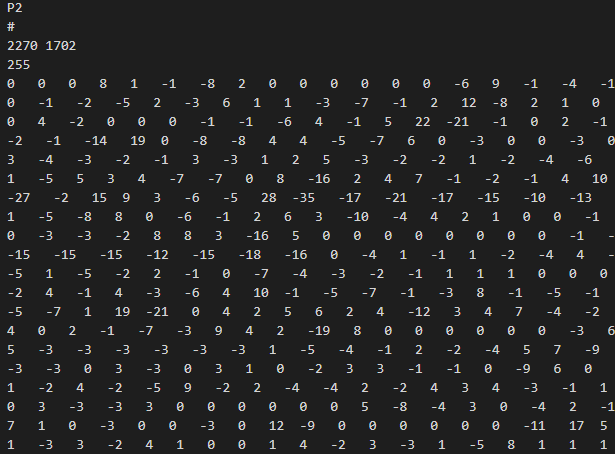


1. Upišite točno ime slike u “cmd“ gdje vam je program pokrenut



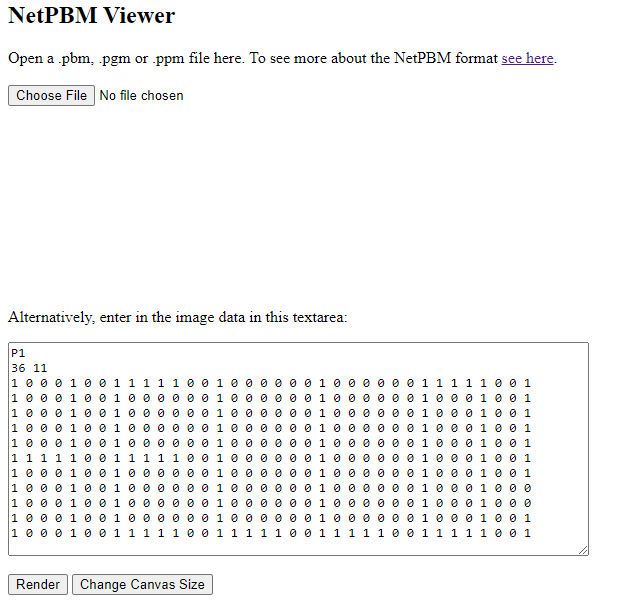
Odaberite još željeni filter, nakon čega će proces obrade početi! Rezultat obrade nalazi se u datoteci “result.pgm”.

1. Otvorimo “result.pgm”

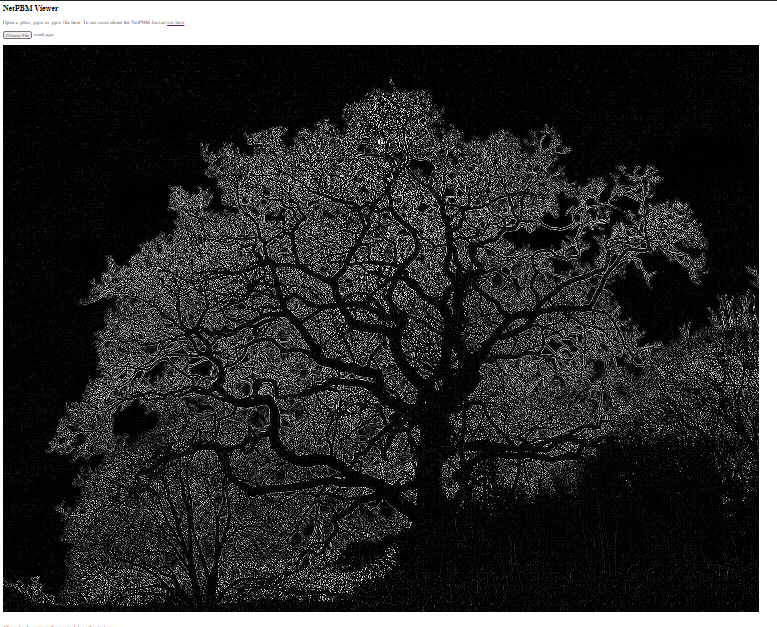


Ovo mnoštvo brojeva predstavlja svaki piksel koji se nalazi u obrađenoj slici. Možete otvoriti “fractal\_tree.ascii.pgm” u direktoriju “images” te ćete usporedbom zaključiti da vrijednosti stvarno nisu iste. Pogledajmo kako to stvarno izgleda nakon obrade!

1. Prijeđimo na stranicu “NetPBM Viewer” koju smo otvorili u 7. koraku. Stisnite gumb “Choose File”, navigirajte do našeg projekta te odaberite datoteku “result.pgm”.



1. Nakon otvaranja vidimo prekrasan rezultat. Naime “Edge detection” filter otkriva sve rubove na slici te ih naglašava.



1. Nakon ovoga možete nastaviti koristiti program upisom imena druge slike te ponavljanjem istog postupka ili napišite “stop” za prekid.

3. Opis implementacije

Implementacija je prilično jednostavna. Otvorite li kod u Visual Studio Code-u vidjet ćete da su sve metode i neke bitne linije u njima dokumentirane. Datoteke:

1. main.cpp

Datoteka koja sadrži “main” metodu s kojom se pokreće program. U main metodi napravljena je jednostavna aplikacija za unos korisnika (koji se validira) te za upravljanje toka programa. Također postoji funkcija readFile()pomoću koje se učitava slika.

1. pgmImage.cpp

Datoteka koja sadrži konstruktore te neke korisne metode koje pripadaju klasi pgmImage

1. imageProcessing.cpp

Datoteka u kojoj je ostvarena sva logika za filtere te njihove pripadajuće matrice s kojima se provodi konvolucija.

1. Ostalo

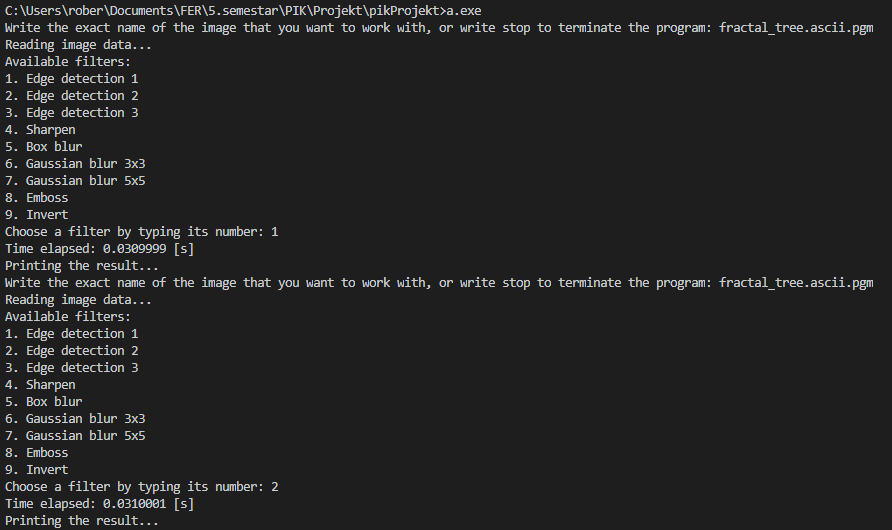
Header datoteke, executable i rezultat filtriranja.

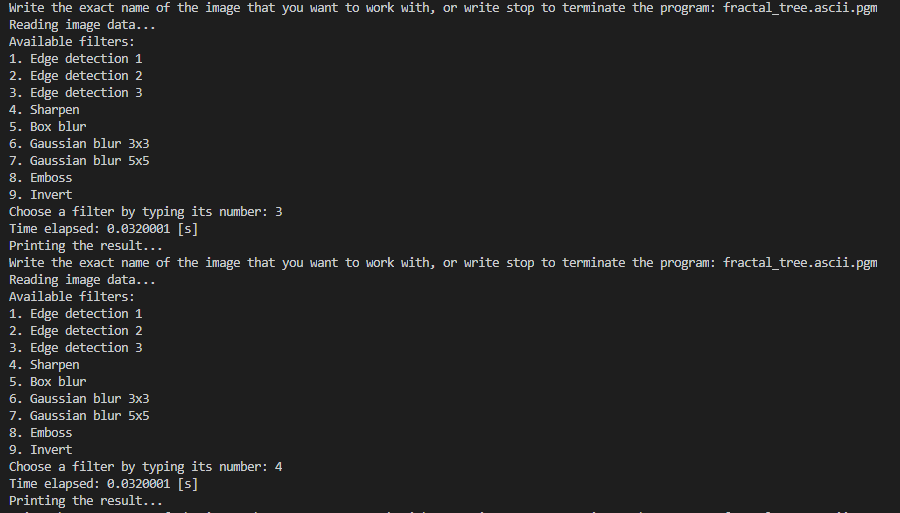
4. Optimizacije

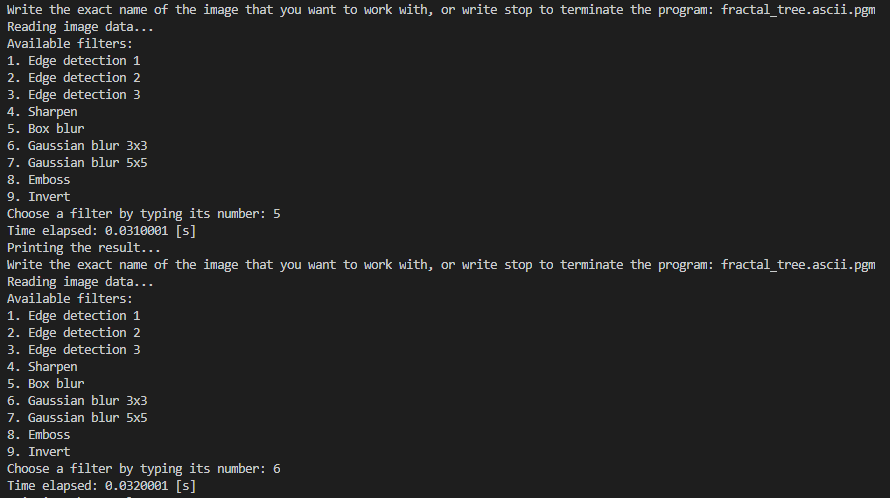
Budući da je sav kod moj, imao sam slobodu implementirati ove funkcionalnosti kako hoću. Rezultat toga je jedna jedina funkcija convolute koja provodi sve filtere. Kako je to moguće? Odlučio sam da je najspretnije imati jednu funkciju kojoj kao argumente šaljem sliku i matricu (“kernel”) kojom provodimo konvoluciju. Optizacijom te jedne funkcije automatski sam dobio bolju performansu na svim filterima. Još bih samo nadodao da filter “invert” ima posebnu implementaciju. Zašto samo on? Zato što nam za invertiranje vrijednosti piksela ne treba konvolucija nego samo svaku vrijednost oduzmemo od 255 te ju spremimo na pripadajuće mjesto. Funkcija invert također je optimizirana.

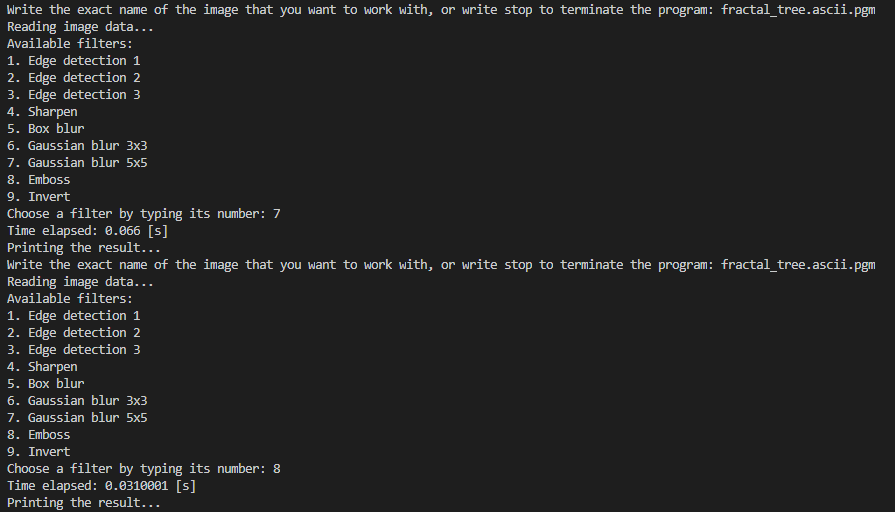
5. Rezultati

Rezultati mjerenja (samo dio sa konvolucijom) prije optimizacije:

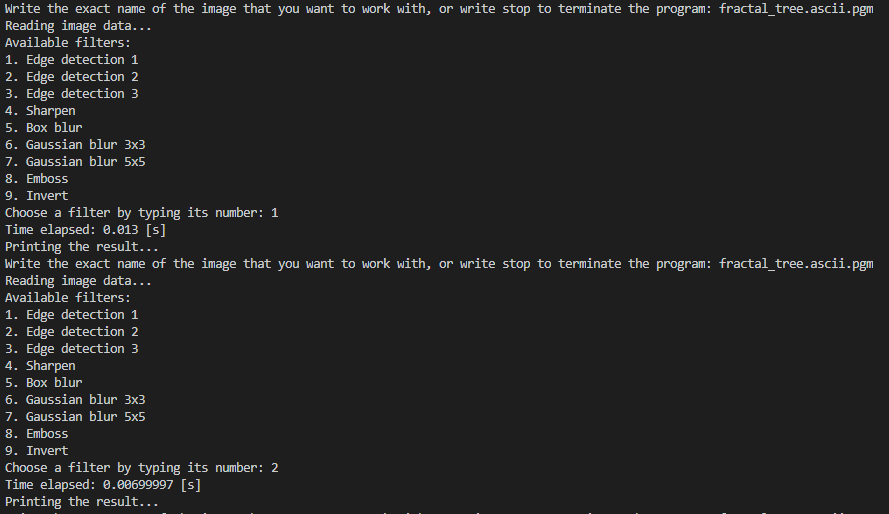


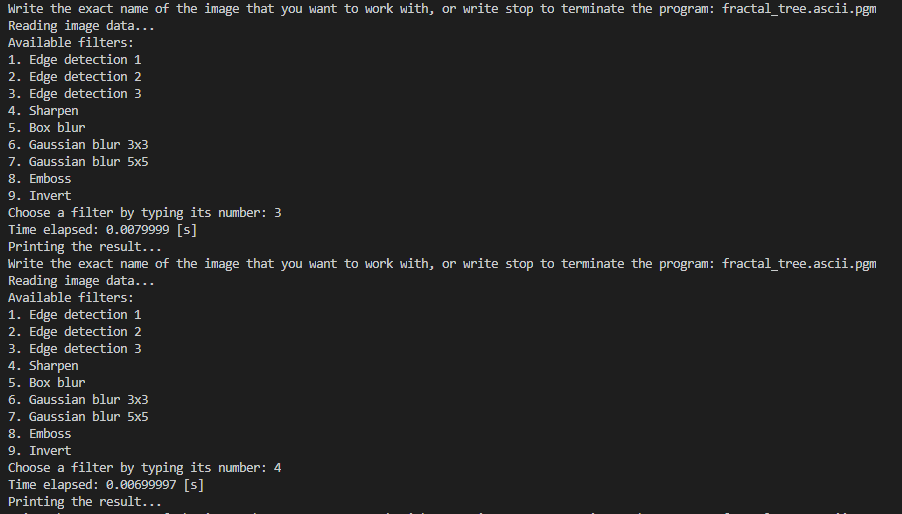


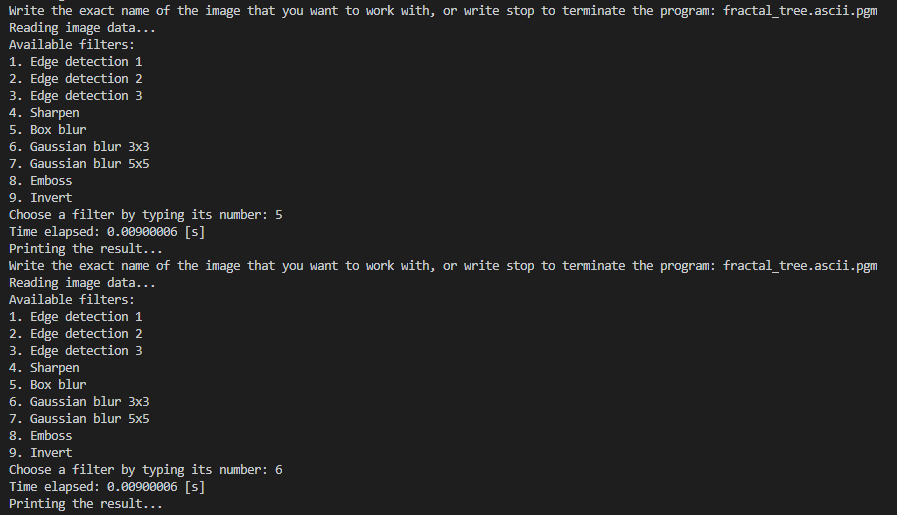


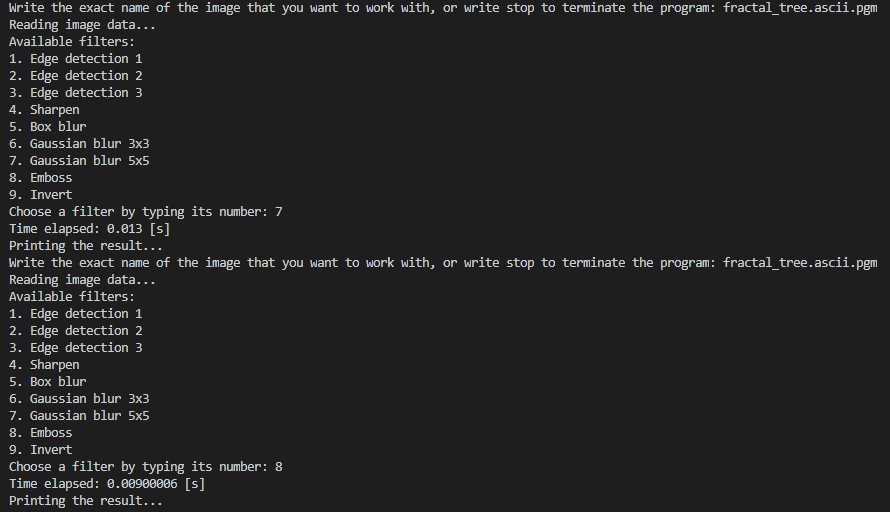


Rezultati mjerenja poslije optimizacije:







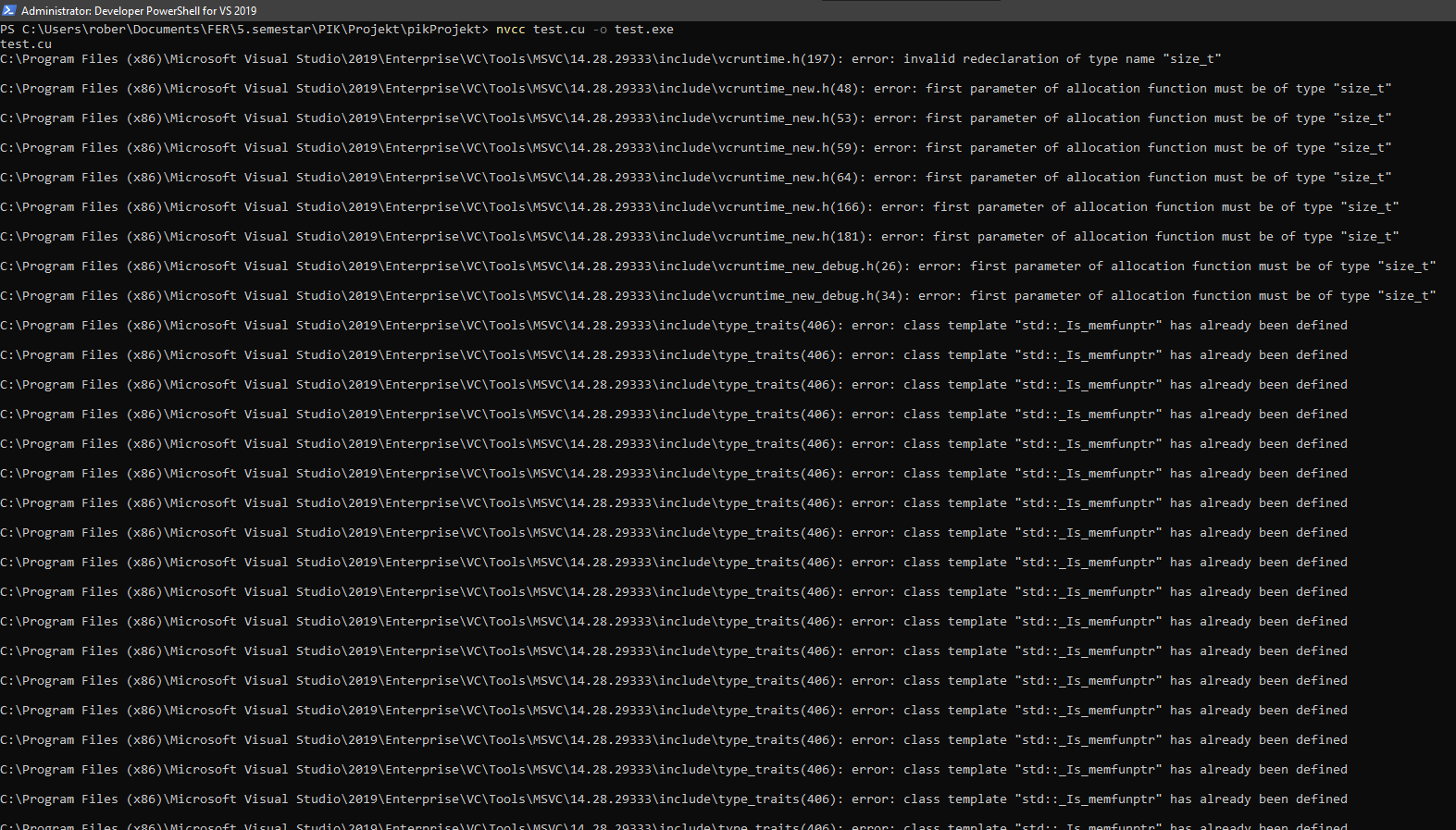


Grafički prikaz vremena potrebnog za izvođenje optimiziranog i neoptimiziranog programa:

Uočimo kako je vrijeme izvođenja za filter “Gaussian blur 5x5” neobično duže od ostalih filtera. Razlog tome je što je za navedeni filter matrica veličine 5x5, a ne 3x3.

6. Zaključak i ideje za daljnja poboljšanja

Ovaj projekt mi je bio jako zabavan i koristan, htio sam sve napraviti sam, kakvo god da je ispalo na kraju, moje je. Napravio sam osvnovne zahtjeve jer sam za vrijeme izrade ovog projekta bio u ogromnoj stisci s vremenom. Isti je razlog i zašto sam morao odustati od prenošenja ovog koda na GPU. Pokušao sam instalirati potrebnu podršku (Visual Studio, nVidia Cuda), ali nešto je jednostavno pošlo po zlu.



Probao sam eksplicitno navesti da se prevodi 64-bitni program, ali ni to nije uspjelo.

Evo iskreno se nadam da će ovo biti dostatno i da se vidi uložen trud. Bude li bilo potrebno slobodno me pitajte sve vezano za implementaciju.