ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

Semestrálna práca

Počítačová grafika – 2024

Bc. David Kučera
Akademický rok 2024/25, zimný semester

Obsah

Zadanie a cieľ práce

Načítajte obrázok zo súboru, formát obrázka je YUV420. V prvej časti súboru sa nachádzajú dáta Luminancie. Veľkosť obrázka je 512 x 512 px, luminancia je zakódovaná ako hodnota 0..255 svietivosti pre 1 pixel. Farebné dáta v obrázku, ktoré nasledujú môžete ignorovať. V obrázku sa nachádza čierna čiara na bielom pozadí. Vašou úlohou je získať stred čiary a vektorizovať ho pomocou Beziérovej krivky.

Cieľom tejto práce je nájsť taký postup pre spracovanie obrázkov, aby bol z výpočtového hľadiska čo najefektívnejší a dokázal spracovávať dáta z kamery v čo najkratšom čase.

Taktiež je nutné riešenie implementovať v prostredí Microsoft Visual Studio 2022 na platforme .NET 8.0 v programovacom jazyku C#. Nie je dovolené používať OpenCV.

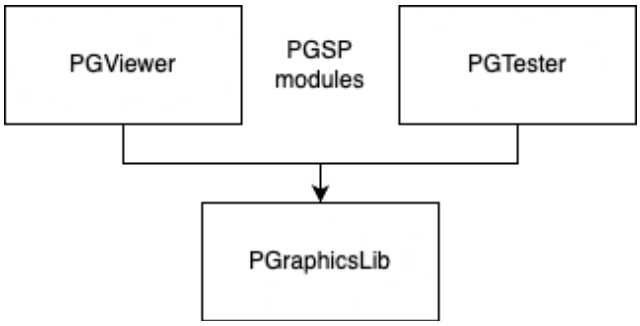
Popis a analýza problému

Počas riešenia úlohy budú potrebné rôzne metódy spracovania obrazu, napríklad vysokofrekvenčný Gaussov filter, alebo iný vhodný filter, histogram, algoritmus pre adaptívne prahovanie, napr. Otsu treshold, hľadanie hrán, napr. Sobel edge detector, práca s maticami a vektormi, knižnica na prekladanie krivky bodmi, napr. MathNet.Numerics.

Na otestovanie funkčnosti navrhnutého riešenia boli poskytnuté nasledovné 3 dátové súbory: *NewImage.txt*, *NewImage2.txt*, *NewImage3.txt*. Každý súbor obsahuje obrázok veľkosti 512 x 512 – na ňom sa nachádza čierna čiara na bielom pozadí. Taktiež bolo poskytnutých niekoľko podobných obrázkov, avšak veľkosti 640 x 480. Na týchto obrázkoch bolo testované cyklické spracovanie.

Návrh samostatných častí programu

Práca bola implementovaná v programovacom jazyku **C#** na platforme **.NET 8.0** a bol vyvíjaný vo vývojom prostredí Visual Studio 2022, podľa požiadaviek zadania. Obsahuje program s grafickým rozhraním pre načítanie a zobrazenie obrázkov typu YUV420, poskytuje aj rôzne možnosti ich úpravy pomocou implementovaných operácií. Ďalej obsahuje aj jednoduché konzolové rozhranie pre testovanie rýchlostí jednotlivých algoritmov, či už jednotlivo, alebo aj cyklicky. Tieto dva programy využívajú spoločnú knižnicu, v ktorej sú algoritmy implementované. Práca teda obsahuje dokopy 3 moduly – ***PGViewer*** (GUI), ***PGraphicsLib*** (DLL), ***PGTester*** (CLI). Prepojenie týchto modulov je možné vidieť v nasledovnom diagrame.



Obrázok 1 Diagram modulov

Riešenie problému hľadania beziérovej krivky

V tejto práci je cieľom zo vstupného súboru dostať ako výstup beziérovu krivku, ktorá bude čo najlepšie opisovať čiernu čiaru obsiahnutú v dátových súboroch.

Algoritmus je nasledovný:

1. **Načítanie** dátového súboru formátu YUV420. Tento obrázok obsahuje čiernu čiaru na pozadí, ktoré je odlišné od čiernej – biele, odtiene šedej.
   Je možné načítať len niektoré časti obrázka, pre zrýchlenie trvania algoritmu, napr. stačí prečítať každý 10. riadok.
2. Vykonanie **filtrácie** pomocou Gaussového algoritmu. Tento krok sa nemusí vykonať, avšak je dobré ho zahrnúť, pretože obrázky môžu obsahovať istý šum, ktorý vďaka tomuto kroku odfiltrujeme. Taktiež vďaka tomu zanedbáme nepotrebné detaily.
3. Binarizácia obrázka na čiernu a bielu farbu pomocou **prahovania**. Pre výpočet prahu je v práci implementovaný aj Otsu prah, ale v mojom testovaní nebol tak dobrý, ako jednoduché **priemerné** prahovanie.
4. Hľadanie **stredu** čiary. Tento stred čiary sa po predošlých operáciách hľadá pomerne ľahko, nakoľko v ideálnom prípade máme biele pozadie a čiernu čiaru – tak stačí len nájsť súradnice čiernych pixelov a vziať ich stred. Takto

to spravíme pre každý (alebo každý n-tý) riadok. Niektoré z týchto bodov nám budú popisovať výslednú beziérovu krivku.

5. Nakoniec sa z bodov vyberie začiatok a koniec – tie nám budú pevne definovať začiatok a koniec beziérovej krivky. K nim sa vyberú ešte dva kontrolné body, ktoré budú určovať tvar **beziérovej krivky**.

Štruktúra programu

Odovzdaný repozitár obsahuje nasledovné priečinky:

- data – obsahuje vstupné dáta (taktiež aj v prípade, že chceme nové dáta pridať, je nutné ich vkladať sem, inak sa v PGViewer nezobrazia),
- doc – obsahuje dokumentáciu, UML diagramy,
- modules – obsahuje zdrojové kódy projektov (.csproj, .cs, ...),
- solution – obsahuje solution (.sln) pre otvorenie projektov v prostredí VS2022.

Po prípadnom builde vo VS2022 pribudnú nasledovné priečinky:

- run/run64/run_D/run64_D – obsahuje .exe binárky pre spustenie programu na základe konfigurácie (viď nižšie),
- tmp – dočasné súbory (priečinok obj),
- output – obsahuje výstup z testovacej aplikácie, ak sa zvolí prepínač.

Programy je možné zbuildovať a následne spustiť v nasledovných konfiguráciách:

- Debug
  - x64 (priečinok run64_D)
  - x86 (priečinok run_D)
- Release – na tejto sa robili testy v poslednej časti práce
  - x64 (priečinok run64)
  - x86 (priečinok run)

Modul PGViewer

Modul obsahuje grafické rozhranie implementované pomocou **WinForms**, nakoľko sa s ním pracovalo na cvičeniach tohto predmetu.

OBRÁZOK ako vyzerá program

Okno obsahuje v ľavej časti niekoľko možností:

- voľba dátového súboru,
- nastavenie šírky a dĺžky obrázka pre prípad iných rozmerov,
- aplikovanie Gaussovho šumu s rôznym parametrom sigma (udáva mieru zašumenia),
- aplikovanie tresholdingu,

- aplikovanie Sobel Edge Detection,
- zobrazenie Beziérovej krivky,
- zobrazenie histogramov obrázkov,
- zobrazenie stredovej čiary čiernej čiary.

V pravej časti sa nachádza po načítaní originálny obrázok spolu s manipulovaným obrázkom, aby bolo možné ihneď porovnať rozdiely. Taktiež pod obrázkami je možné zobrazovať ich histogramy.

Modul PGraphicsLib

Táto knižnica obsahuje všetky implementované operácie a algoritmy pre manipuláciu s obrázkami.

Diagram tried modulu PGraphicsLib

UML TU

Bezier.cs

dsdsds

Gauss.cs

dsds

GrayscaleImage.cs

ňdsds

MiddlelineExtractor.cs

dsd

Sobel.cs

dsds

Thresold.cs

dsds

Modul PGTester

Tento modul obsahuje len jeden súbor *Program.cs*, ktorý obsahuje jedinú triedu Main. V nej sú všetky potrebné náležitosti pre vykonanie testovania vybraných algoritmov tejto práce. Je možné si výsledky testov ukladať do externého csv súboru, napríklad pre ďalšiu analýzu, prípadne pre tvorbu prehľadných grafov.

Možnosti testov

Test je možné spustiť príkazom **run** a s prepínačmi:

- -n, kde n predstavuje počet opakovaní, napr. **run -10**,
- -c, kde tento prepínač spustí cyklické spracovanie všetkých poskytnutých obrázkov rozmeru 640 x 480, napr. **run -c**,
- -s, kde tento prepínač zabezpečí to, že sa výsledky uložia do externého csv súboru do priečinku output, napr. **run -10 -s**.

Externé knižnice

V práci sa nepoužívajú žiadne externé knižnice.

Vyhodnotenie rýchlostí jednotlivých častí a celkového trvania algoritmu pre spracovanie 1 obrázka a tiež cyklické spracovanie

SEM GRAFY

Porovnanie rýchlostí jednotlivých obrázkov

NewImage vs NewImage2 vs NewImage3

Porovnanie rýchlostí cyklického spracovania

Image01 – všetky 640 x 480 obrázky

Porovnanie rýchlostí pri Release a Debug konfigurácií

Porovnanie x64 a x86 Debug

Porovnanie x64 a x86 Release

Porovnanie trvania Sobel Edge Detection

aj nejaky popis preco sa nepouziva v praci , ale je to implementovane aj sa hlada stred medzi dvoma bielymi ciarami ... preco je to nevyhodne atd.

Záver