Počítačová grafika 2D

Semestrálna práca

A logo on a black background

Description automatically generated

Meno: Lukáš Beďač

Cvičiaci: Ing. Michal Lekýr, PhD.

Štud. skupina: 5ZIS11

Školský rok: 2024/2025

# 1.Popis a analýza problému

Cieľom projektu je spracovanie YUV420 obrazu, v ktorom sa nachádza čierna čiara na bielom pozadí. Z luminančných dát je potrebné extrahovať stredovú líniu čiary, vektorizovať ju pomocou Beziérovej krivky.

Hlavnou výzvou je nájsť efektívny algoritmus, ktorý zabezpečí rýchle spracovanie obrazu, aby bol vhodný aj pre cyklické spracovanie videosignálu. Je dôležité vybrať optimálne metódy pre filtráciu obrazu, detekciu hrán a extrakciu bodov, aby bol algoritmus presný a zároveň rýchly.

# 2. Návrh samostatných častí projektu

**1. Načítanie a predspracovanie obrazu**

* Načítanie luminančných dát z YUV420 formátu
* Konverzia luminančných hodnôt do šedotieňového obrazu
* Aplikácia vysokofrekvenčného filtra (Gaussov filter)

**2. Aplikovanie detekcie hrán**

* Aplikácia Sobel Edge Detectora na obrázok po aplikovaní gaussovho filtra

**3. Aplikovanie prahovania**

* Aplikácia Otsu Threshold algorimu na obrázok po aplikovaní Sobel Edge detectora
* Prahovanie slúži na zvýraznenie hrán a lepšieho vyzobrazenia stredovej čiary

**4. Vektorizácia Beziérovou krivkou**

* Využitie obrázku po prahovacom algoritme na nájdenie vhodných bodov pre interpoláciu
* Aproximácia bodov Beziérovou krivkou pomocou MathNet.Numerics

**5. Vyhodnotenie efektivity a optimálnosti**

* Meranie času jednotlivých metód (krokov) ako aj jedného priebehu, tak isto je aplikácia prispôsobená na časové meranie cyklického spracovania obrázkov

# 3. UML štruktúra programu

A screenshot of a computer

Description automatically generated

**Hlavné triedy:**

* **Interface**
  + **IDrawable2DObject –** obsahuje metódy na vykreslenie
* **Triedy pomocné:**
  + **GaussFilter –** slúži na aplikovanie Gaussovského filtra na obrázok, tak isto obsahuje metódu na tvorbu gaussovského jadra
  + **SobelOperator –** slúži na aplikovanie Sobel Edge Detectora na obrázok upravené gaussovským filtrom, ktorý nájde v obrázku hrany stredovej čiary
  + **Otsu Threshold –** slúži na zvýraznenie hrán okolo stredovej čiary, pričom vzniká čiernobiely obrázok
  + **Bezier Curve –** slúži na nájdenie stredovej čiary, uloženia bodov a následnej interpolácie bodov pomocou MathNet.Numerics
  + **Load Image –** slúži na načítanie obrázka a jeho premenou z pola bajtov do Bitmapy
  + **Draw Image –** slúži na vykreslenie obrázka
* **Hlavná trieda**
  + **MainWindow –** slúži na aplikovanie všetkých pomocných algoritmov, tak isto na spracovanie uživateľských požiadaviek
* **Externé knižnice**
  + **MathNet.Numerics –** na interpoláciu Beziérovej krivky

# 4.Implementácia a vyhodnotenie efektivity

**Implementácia –** realizovaná vo Visual Studio 2022 s využitím .NET 8.0 a jazyka C#

**Vyhodnotenie efektivity**

* Čas potrebný na aplikovanie **Gaussovho** **filtra**: 35 – 45 ms
* Čas potrebný na aplikovanie **Sobel Edge Detectora**: 25 – 30 ms
* Čas potrebný na aplikovanie **Otsu Threshold** algoritmu: približne 5 ms
* Čas potrebný na vytvorenie **Beziérovej krivky:** 1 ms
* Čas na priebeh **jedného cyklu**: 60 – 85 ms
* Čas na priebeh **100 cyklov:** 6650 – 6800 ms
  + Bez vykreslovania obrázkov

# 5.Problémy na ktoré som narazil

Pri semestrálnej práci som narazil na veľa problémov. Pri každom postupe, ktorý sa ďalej má aplikovať som musel hľadať rôzne algoritmy na spracovanie obrázku.  
Nakoniec som využil všetky algoritmy, ktoré nám boli doporučené p. učiteľom Michalom Lekýrom v dokumente a po vyhľadaní na internete som zhodnotil že ich implementácia, rýchlosť a náročnosť sú asi najoptimálnejšie. Pri každom algoritme som bojoval z náročnými vzorcami, ktorými sa vypočítavali určité hodnoty ako napr.: pri Otsu Threshold algoritme, keď sa počítala adaptívna prahová hodnota, ktorú som nakoniec musel manuálne zmeniť z dôvodu zlého vyhodnocovania obrázkov.