ZPO technická správa
Filtrové "efekty" v obrazu

Denis Dovičic
xdovic01@stud.fit.vutbr.cz

16.4.2021

Tým: Denis Dovičic (xdovic01@stud.fit.vutbr.cz)

1 Zadanie

Navrhněte několik filtrů (cca 3), které po aplikaci na obraz vedou k vytvoření ësteticky hodnotnéhoëfektu. Může to být například obraz se zvýrazněnými hranami, "reliéf obrazu", kvantizovaný obraz apod. Výsledky zdokumentujte a prezentujte.

1.1 Upresnené zadanie

Implementácia troch filtrov extrakcia „reliéfu“ z obrazu, zvýraznenia hrán a kvantizáciu obraz. Na extrakciu „reliéfu“ využiť emboss filter a jeho varianty zo všetkých strán. Zvýraznenie hrán pomocou Sobel/Prewitt filtra aplikácia na jednotlivé farebné kanály a získanie farebne zvýraznených hrán. Kvantizácia obrazu pomocou k-means clustering.

Webkamera

Všetky filtre aplikovať na stream webkamery filtrovať obraz realtime. Pre kvantizáciu obrazu implemetácia dvoch variant, adaptívna - clustre sa updatujú pri každom snímku, statická - clustre sú spočítané len pri incializácií filtru a následne aplikované tie isté počas celého streamu.

GUI

Demonštračná aplikácia, v ktorej bude vhodne zobrazený filtrovaný stream webkamery s nastaveniami príslušných filtrov.

2 Popis dát a demonštračná aplikácia

Ako už bolo spomenuté vyššie v upresnenom zadaní, filtrované dáta budú snímky kamery. Stream bude zobrazený v demonštračnej GUI aplikácií, na ktorý bude aplikovateľný každý z požadovaných filtrov. Demonštračná aplikácia bude obsahovať aj parametrické nastavenia príslušných filtrov, na základe ktorých bude prebiehať automatická aktualizácia scény.

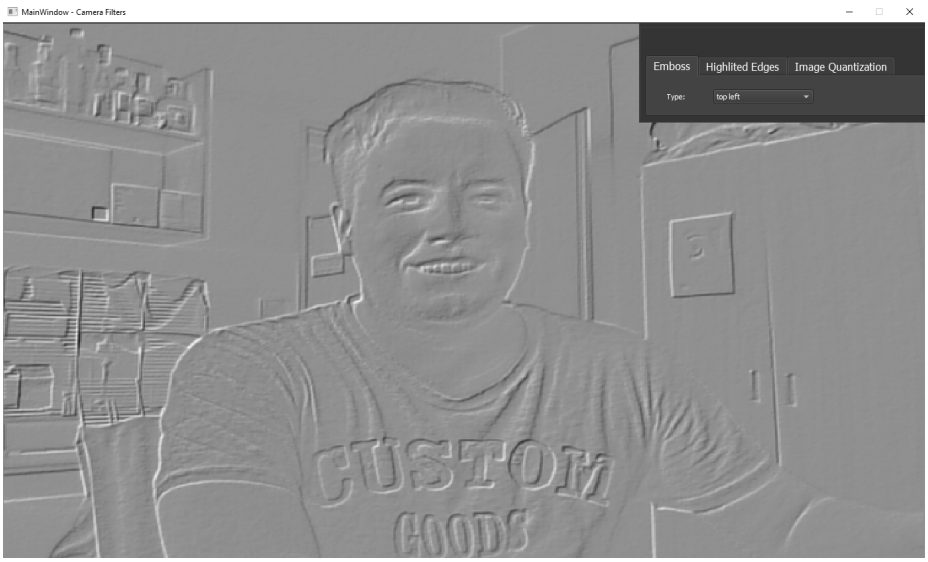
3 Filtrácie obrazu

V tejto časti technickej správy budú popísane požadované filtre z teoretického hľadiska. Najprv bude popísaný filter na extrakciu „reliéfu“ z obrazu, následne filter na zvýraznenie hrán a na koniec kvantizácia obrazu.

3.1 „Reliéf“ obrazu

K extrakcií „reliéfu“ obrázka je potrebná operácia konvolúcie a konvolučné jadro. Konvolućným jadrom je vhodné pre tento typ filtra zvoliť emboss filter. Filter dodáva efekt 3D povrchu pomocou zvíraznenia svetla a tieňa aplikovaním svojej masky na jednotlivé farebné kanály. Výsledný obrázok je nutné previesť do šedotónového formátu. V rovniciach nižšie je možné vidieť jednotlivé varianty emboss filtru aplikovateľné „z rôznych strán“ a na obrázku 1 je možné vidieť efekt spôsobený samotným filtrom. [1] [2]

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{1}$$



Obr. 1: Ukážka aplikácie emboss filtra.

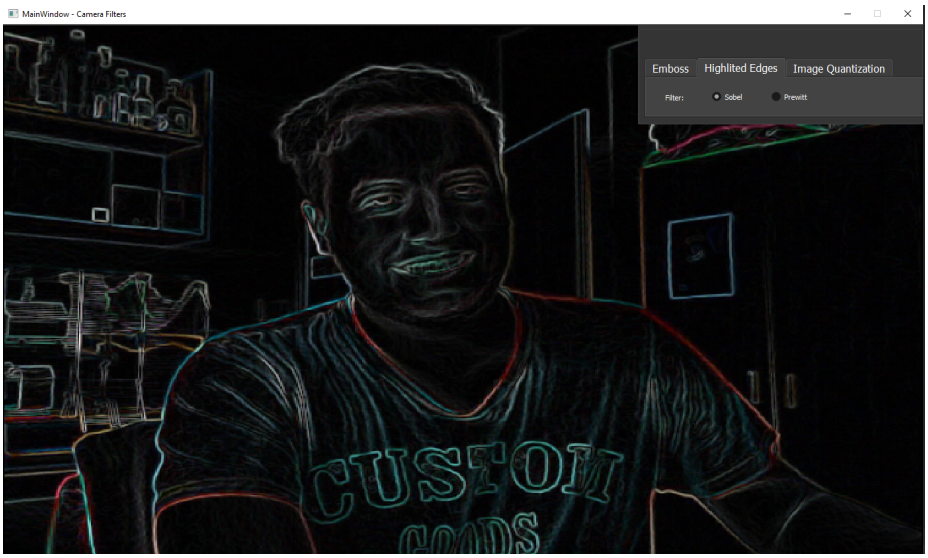
3.2 Zvýraznenie hrán

Jedným z najčastejších prístupov k zvýrazneniu hrán obrazu je využitie operácie konvolúcie a jadra s nejakými váhami. Známym jadrom pre túto filtráciu je Prewittov filter, ktorý sa skladá z dvoch masiek uvedených v rovnici 2. [2]

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad S_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

Zdvojnásobením centrálnych váh týchto masiek vznikne jeden z najznámejší filtrov pre zvýraznenie hrán, Sobelov filter. Jeho reprezentáciu masiek je možné vidieť v rovnici 3. [2]

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3}$$



Obr. 2: Ukážka aplikácie filtra na zvýraznenie hrán.

3.3 Kvantizácia obrazu

Kvantizácia je proces digitalizácie nejakého rozsahu. Kvantizácia obrazu je teda technika stratovej kompresie dosiahnutá znížením rozsahu hodnôt na niekoľko kvantových hodnôt. Aplikovať tento proces na obraz je možné napríklad znížením počtu farieb potrebných k jeho zobrazeniu. [3]

Na nižšie uvedenom obrázku 3 je možné vidieť kvantizáciu do dvoch, štyroch, šiestich a ôsmych farebných zhlukov pomocou algoritmu kmeans. Okrem toho je na obrázkoch pozorovateľný špeciálny efekt známky ako kontúrovanie. [3]

K-means zhlukovanie

Algoritmus začína náhodným zvolením centroidov, bodov reprezentujúcich stredy zhlukov (sú považované za východiskové body). Následne prebieha iteratívna aktualizácia centroidov pomocou priemernej hodnoty jednotlivých dát v aktuálnom zhluku, na základe ktorej sú všetky dáta znovu prerozdelené do svojich zhlukov ku ktorým momentálne najlepšie sedia (od ktorého centroidu majú najmenšiu vzdialenosť). Tento iteratívny postup končí v prípade, že sa už centroidy prestanú aktualizovať (našli sa najideálnejšie pozície) alebo bol dosiahnutý maximálny počet iterácií. [4]

Obr. 3: Kvantizácia obrazu na 2, 4, 6 a 8 zhlukov.

4 Implementácia

Na implementáciu bol využitý jazyk *C++*, knižnica *Qt5* a *OpenCV* a pre preklad bol využitý *cmake*. Aplikácia sa skladá z dvoch logických častí:

- statická knižnica *ImageFilters* - obsahuje implementáciu popísaných filtrov z kapitoly 3,
- demonštračná GUI aplikácia - výzor aplikácie je možné vidieť vo všetkých obrázkoch tejto technickej správy.

4.1 ImageFilters

Statická knižnica obsahujúca implementáciu filtrov popisovaných v kapitole 3. Emboss filter zabezpečuje trieda *zpo::Emboss*, obsahuje implementáciu všetkých variant z kapitoly 3.1 a jednu kombinovanú, kedy výsledný obraz vzniká aplikáciou tretieho a štvrtého jadra z rovnice 1. Dva medzivýsledky získané z oddelenej aplikácie jadier sú spojené do finálneho obrázku maximalizáciou ich pixelov.

Zvýraznenie okrajov zapuzdruje trieda *zpo::EdgeDetection*, ktorá obsahuje implementáciu Sobelovho a Prewittovho filtra z kapitoly 3.2. Výsledný obrázok vzniká oddelenou aplikáciou Sobelových/Prewittových jadier a následným „spojením“ medzivýsledkov X a Y do jedného obrázka vzťahom:

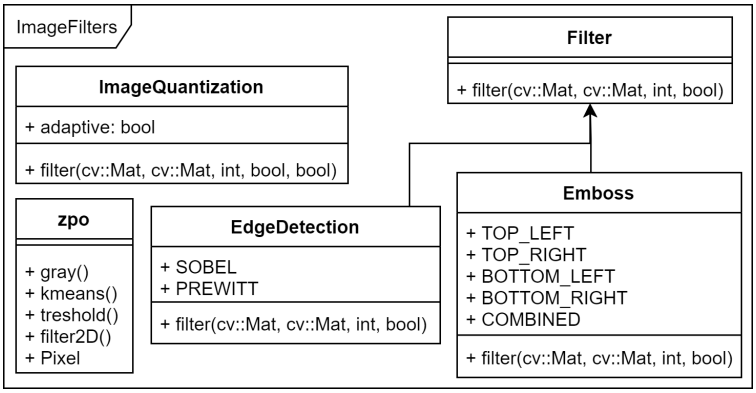
$$S_{ij} = \sqrt{Y_{ij}^2 + X_{ij}^2} \tag{4}$$

kde i a j sú indexy pixelov finálneho obrázka a medzivýsledkov. Pri aplikácií filtra oddelene na jednotlivé farebné kanály a následné spojenie vyfiltrovaných kanálov do RGB obrázku, je možné získať farebne zvýraznené hrany ako je vidieť na obrázku 2.

Kvantizáciu obrazu zabezpečuje trieda *zpo::ImageQuantization* využívajúca kmeans algoritmus k vytvoreniu farebných zhlukov (kapitola 3.3). Jednotlivé RGB zložky sú považované ako pozície v „3D“ priestore a algoritmus sa snaží nájsť ich odpovedajúce zhluky, ktorých počet je parametricky nastaviteľný. Boli implementované dva prístupy, výpočet centroidov pred zobrazovaním samotnej scény a aktualizácia centroidov pred každým snímkom. Adaptívny prístup je výpočetne náročnejší,

čo má za následok jemné „sekanie“ videa, ale v konečnom výsledku je efekt oveľa krajší. Subor *zpo.h* obsahuje implemetované pomocné metódy z *OpenCV* ako *zpo::filter2D()*, *zpo::kmeans()*, pomocnú štruktúru *zpo::Pixel* atď.

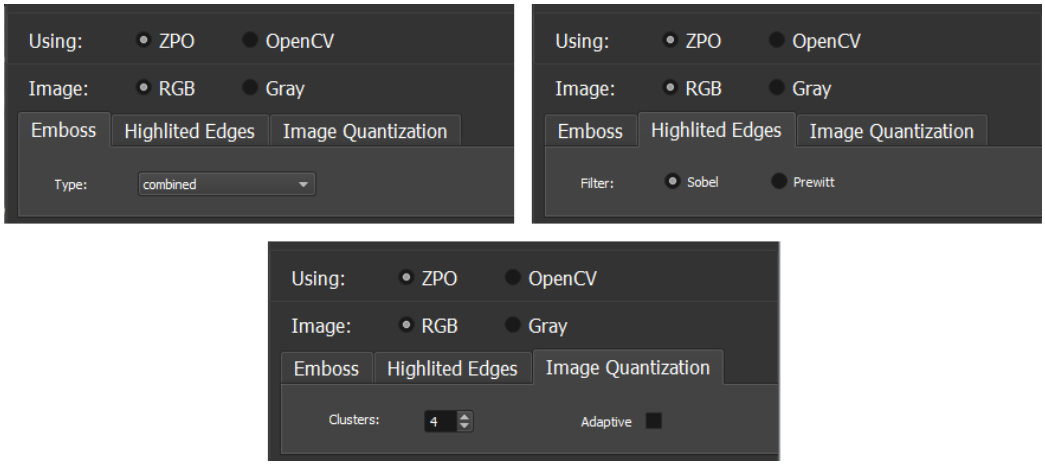
Každý z filtrov je prispôsobený na čiernobiele aj farebné obrázky a po filtrácií sú výsledky vždy normalizované do rozsahu $\langle 0, 256)$. Diagram tried popisovanej knižnice je možné vidieť na obrázku 4.



Obr. 4: Diagram tried statickej knižnice ImageFilters.

4.2 Demonštračná aplikácia

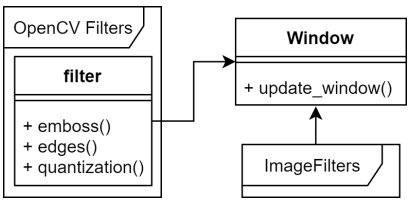
Prácu celej demonštračnej aplikácie (aktualizáciu filtrov, nastavovanie parametrov filtrov, GUI výstup...) zabezpečuje trieda *Window*. Obsahuje inštancie tried umožňujúcich realtime filtrovanie streamu kamery. Hlavné okno tvorí *QLabel* do ktorého je premietaný stream a skupina widgetov umožňujúcich nastaviť aktuálny filter a jeho parametre. Ukážku nastavení je možné vidieť na obrázku 5.



Obr. 5: Ukážka nastavení v demonštračnej aplikácií.

K lepšej demoštrácií ručne implementovaných filtrov boli doimplementované filtre prostredníctvom *OpenCV* funkcií nachádzajúcich sa v module *cvfilters* pod funkciami *cvfilters::emboss()* (kapitola 3.1), *cvfilters::edges()* (kapitola 3.2) a *cvfilters::quantization()* (kapitola 3.3). Výstupy týchto filtrov odpo-

vedajú ručne implementovaným filtrom v statickej knižnice *ImageFilters*. Diagram tried demonštračnej aplikácie je možné vidieť na obrázku 6.

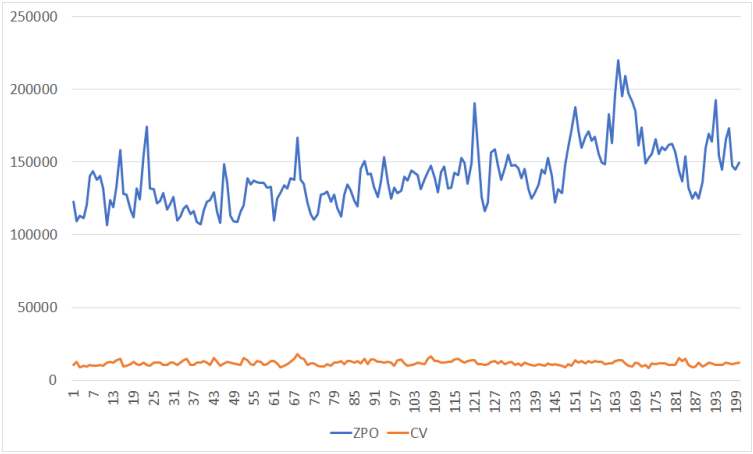


Obr. 6: Diagram tried demonštračnej aplikácie.

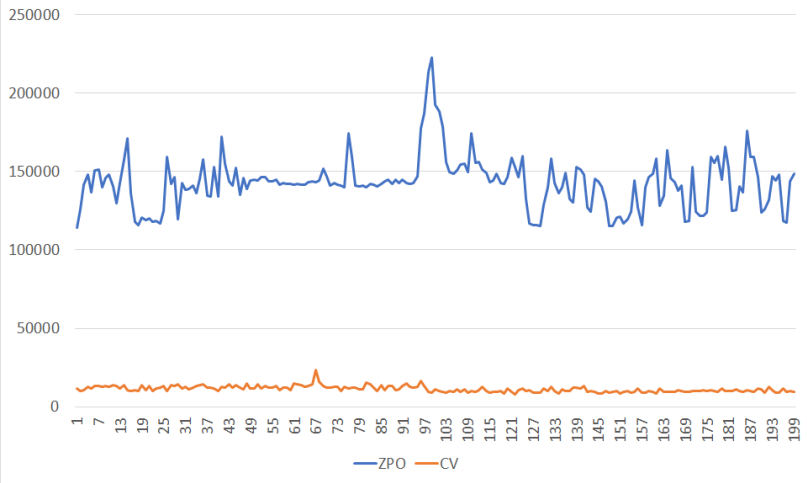
5 Meranie

Meranie výkonnosti statickej knižnice *ImageFilters* oproti filtrom implementovaných pomocou funkcií *OpenCV*, prebehlo prostredníctvom *C++* knižnice *std::chrono*. Pre každý z filtrov bolo získaných 2000 vzoriek, ktoré boli spriemerované po 10 hodnotách a z nich boli zostavené nasledujúce grafy. Nakoniec sú algoritmy medzi sebou vyhodnotené prostredníctvom priemerného času potrebného na filtráciu jedného snímku.

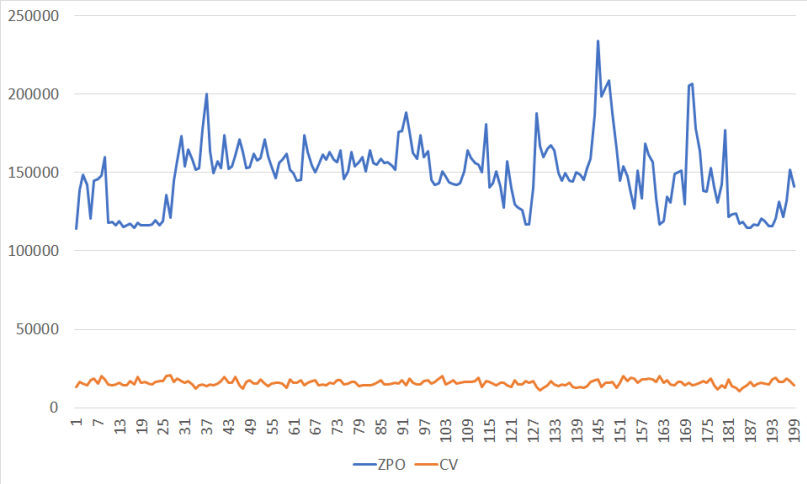
Pre každý graf platí, že vertikálna os predstavuje čas potrebný na filtráciou snímku v μs a horizontálna os predstavuje len poradové hodnoty. Z grafov 7, 8 a 9 je zrejmé, že implementácia *OpenCV* funkciami je oveľa stabilnejšia a rýchlejšia. Pri kvantizácií obrazu v grafe 10 je zasa manuálna implementácia rýchlejšia a stabilnejšia oproti *cv::kmeans()*. Dôvodom je, že manuálna implementácia je priamo prispôsobená na kvantizáciu obrazu zatiaľ čo funkcia *cv::kmeans()* vyžaduje predspracovanie a následnú aplikáciu zhlukov na obraz. Pri adaptívnej kvantizácií, kedy sú pri každom snímku dotrénované lepšie centroidy zhlukov sa už časové nároky na filtráciu snímku oboch prístupov začínajú vyrovnávať, no manúalna implementácia je oproti *OpenCV* stále o niečo rýchlejšia. Porovnanie priemerných časov potrebných na vykreslenie jedného snímku je možné vidieť v tabuľke 1.



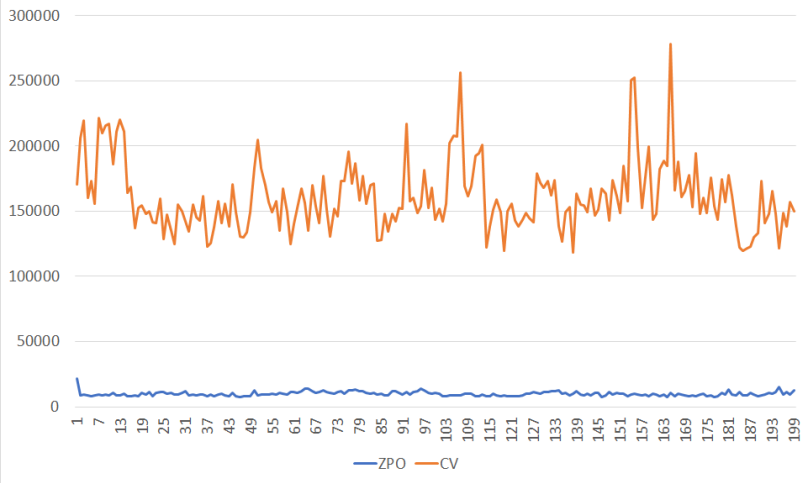
Obr. 7: Emboss filter. Horizontálna os predstavuje čas potrebný na vykreslenie snímku v μs a vertikálna poradové hodnoty.



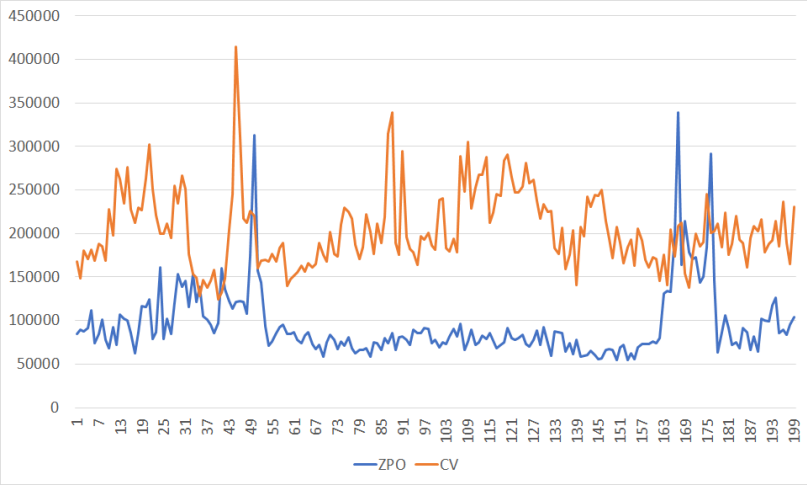
Obr. 8: Sobelov filter. Horizontálna os predstavuje čas potrebný na vykreslenie snímku v μs a vertikálna poradové hodnoty.



Obr. 9: Prewittov filter. Horizontálna os predstavuje čas potrebný na vykreslenie snímku v μs a vertikálna poradové hodnoty.



Obr. 10: Kvantizácia obrazu. Horizontálna os predstavuje čas potrebný na vykreslenie snímku v μs a vertikálna poradové hodnoty.



Obr. 11: Adaptívna kvantizácia obrazu. Horizontálna os predstavuje čas potrebný na vykreslenie snímku v μs a vertikálna poradové hodnoty.

	ZPO	OpenCV
Emboss filter	140247.31	11798.37
Sobelov filter	142719.54	11236.5
Prewittov filter	148821.49	15787
Kvantizácia obrazu	9878.78	160958.35
Adaptívna kvantizácia	94767.56	204447.38

Tabuľka 1: Tabuľka primerných časov potrebných na filtráciu jedného snímku v μs. Priemer je spočítaný z 2000 vzoriek pre oba prístupy (manuálna implementácia - ZPO, a funkcie OpenCV - OpenCV).

6 Záver

Bolo implementovaných niekoľko rôznych filtrov určených k filtrácií obrazu, emboss filter, sobelov filter, prewittov filter a kvantizácia obrazu s využitím kmeans algoritmu. Tieto filtre boli následne demonštrované v grafickej demonštračnej aplikácií a porovnané s filtrami implementovanými funkciami OpenCV. Pri Kvantizácií obrazu pomocou kmeans algoritmu sa podarilo dosiahnuť rýchlejšiu odpoveď filtra než použitím OpenCV funkcie *cv::kmeans()*.

Literatúra

[1] Lode. Lode's computer graphics tutorial. https://lodev.org/cgtutor/filtering.html. [Online; navštívené 21.3.2021].

[2] ScienceDirect. Sobel edge detection. https://www.sciencedirect.com/topics/engineering/sobel-edge-detection. [Online; navštívené 21.3.2021].

[3] TutorialsPoint. Concept of quantization. https://www.tutorialspoint.com/dip/concept_of_quantization.htm. [Online; navštívené 21.3.2021].

[4] Michael J. Garbade. Understanding k-means clustering in machine learning. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1. [Online; navštívené 21.3.2021].