ANALÝZA A OPTIMALIZACE MEDIÁNOVÉHO FILTRU

Molek Vojtěch
Ostravská univerzita, Přírodovědecká fakulta, Katedra informatiky a počítačů, 30. dubna 22, 701 03 Ostrava, p13097@student.osu.cz

Abstrakt

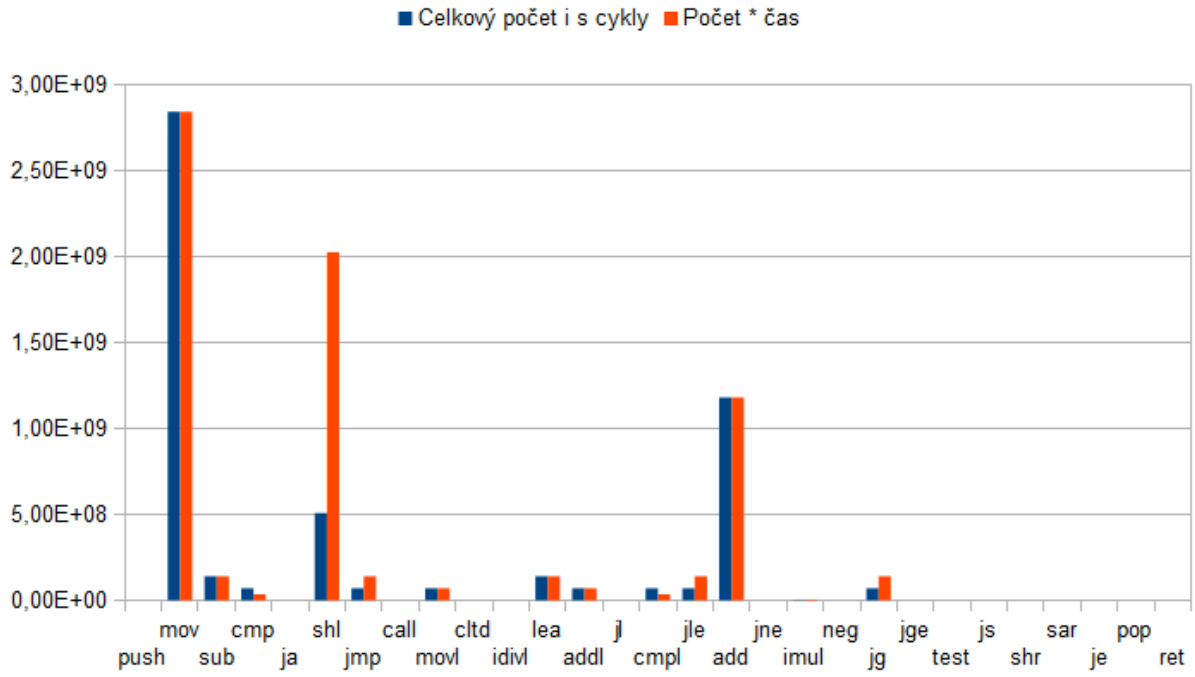
Příspěvek se týká analýzy a optimalizace mediánového filtru. Analýza je provedena na úrovni instrukcí procesoru, vylepšení logiky na základě segmentace dat při paralelismu. Dále se zaměřuje na porovnání a výběr vhodného řadícího algoritmu pro danou úlohu.

Klíčová slova: *paralelismus; mediánový filtr; optimalizace; analýza; řadící algoritmy;*

Analýza mediánového filtru

V oblasti zpracování obrazu je mediánový filtr využíván pro odstraňování šumu ze zpracovávaného snímku. Filtrace je realizována pomocí konvoluce a výběru mediánové hodnoty z konvolučního jádra. Mediánová hodnota je vybírána pro každý pixel zvlášť, jedná se tedy o výpočetně náročnou operaci. K filtraci šumu lze využít taktéž výběru průměrné hodnoty, výsledek pro tento případ bude rozmazaný obraz. Tímto nedostatkem výběr mediánu netrpí.

Algoritmus byl pomocí Qt Frameworku debuggován na úrovni instrukcí. Z tabulek [4] byly zjištěny přibližné časové náročnosti jednotlivých instrukcí.



Obrázek 1. Graf počtu jednotlivých instrukcí a jejich časové náročnosti

Z grafu je patrné, že instrukce *mov*, tedy přesun/kopírování, se vykonává nejčastěji (2846211828) a její časová složitost je jeden cyklus (počet cyklů za sekundu, je jeden z atributů procesorů, uváděný v Hz). Další instrukcí s časovou složitostí jednoho cyklu je *add* (toto je také

důvod proč je counting sort využíván v [2]), tedy sčítání. Poslední často se vyskytující instrukcí je *shl*, násobení. Časová složitost shl jsou čtyři takty.

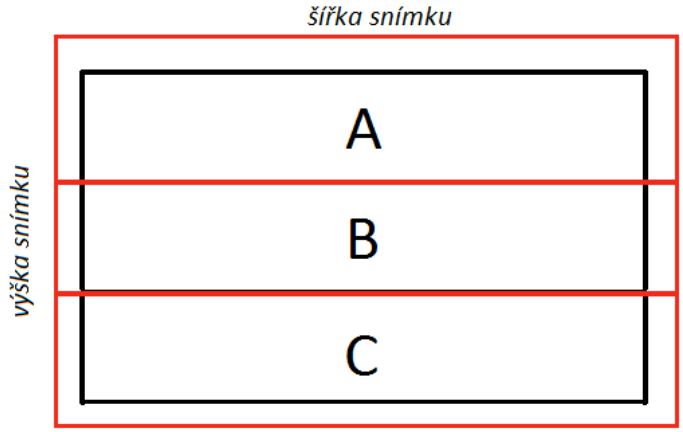
Optimalizace mediánového filtru

Původní algoritmus, který autor implementoval [1] pracuje s vlákny. Celý obraz je rozdělen do vodorovných pruhů a každý pruh je zpracován instancí metody spuštěné ve vláknu. Algoritmus však již nepočítá s tím, nad kterou částí dat je konkrétní instance spuštěna. Toto se projevilo v množství podmínek, které byly nezbytné pro jednu část dat, ale zbytečné pro část jinou. Tento nedostatek byl odstraněn rozdělením snímku do tří oblastí:

$P^A \subseteq [y^A_{min},\ y^A_{max}] \times [x^A_{min},\ x^A_{max}]$, ve které se maska mediánového filtru dostává mimo interval shora tak že: $y < y^A_{min}$.

$P^B \subseteq [y^B_{min},\ y^B_{max}] \times [x^B_{min},\ x^B_{max}]$, ve které maska mediánového filtru zůstává v intervalu tak že: $y \in P^B$.

$P^C \subseteq [y^C_{min},\ y^C_{max}] \times [x^C_{min},\ x^C_{max}]$, ve které maska mediánového filtru dostává mimo interval zdola tak že: $y > y^C_{max}$.



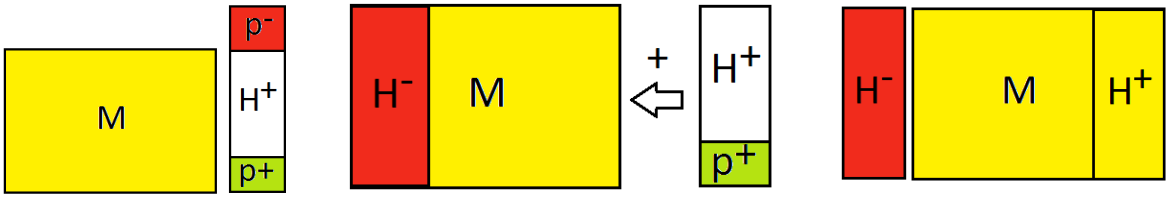
Obrázek 3. A) vrchní oblast B) prostřední pruh C) dolní oblast

Všechny pruhy musí zůstat v intervalu $x \in \left[x^{\{A,B,C\}}_{min}, x^{\{A,B,C\}}_{max}\right]$. Mějme masku mediánového filtru M, poloměr masky r, přidávaný sloupcový histogram H^+ (uchovává četnosti výskytu pixelů v sloupci), odebíraný sloupcový histogram H^-oba o stejné výšce jako maska M, odebíraný pixel p^- a přidávaný pixel p^+. Velikost masky M je $|M| = (2r+1)^2$ ve směru osy x i osy y, je tudíž čtvercová. Celý algoritmus probíhá v několika krocích:

K histogramu H^+ na pozici $H^+_x = M_x + (2r+1)$, přičti p^+ a odečti p^- tak že: $\{p^+\} \cup H^+ \setminus \{p^-\}$.

Přidej histogram H^+ do masky M: $H^+ \cup M$ a histogram H^- na pozici $H^-_x = M_x - (2r+1)$ odeber $M \setminus H^-$.

Vyber medián $\tilde{x} = M_{2r+1}$ z masky M.



Obrázek 4. Posun histogramu o pixel dolů a přidání do jádra masky

Původní algoritmus kontroloval, pro všechna vlákna, během prvního a druhého kroku tyto podmínky:

$$p_x^+ \in \left[x_{min}^{\{A,B,C\}}, x_{max}^{\{A,B,C\}}\right] \; AND \; p_y^+ \in \left[y_{min}^{\{A,B,C\}}, y_{max}^{\{A,B,C\}}\right]$$
$$p_x^- \in \left[x_{min}^{\{A,B,C\}}, x_{max}^{\{A,B,C\}}\right] \; AND \; p_y^- \in \left[y_{min}^{\{A,B,C\}}, y_{max}^{\{A,B,C\}}\right]$$
$$H_x^+ \in \left[x_{min}^{\{A,B,C\}}, x_{max}^{\{A,B,C\}}\right]$$

V optimalizovaném algoritmu se během prvního a druhého kroku, v závislosti na na poloze, vyhodnocují následující podmínky:

P^A: $p_x^+ \leq x_{max}^A$.
P^B: $p_x^+ \leq x_{max}^B$.
P^C: $p_x^- \leq x_{max}^C$.

P^A probíhá ve dvou cyklech, první cyklus sloupcové histogramy pouze plní, protože zasahují mimo snímek. Druhý cyklus již probíhá podle obrázku 4. P^B probíhají podle obrázku 4. P^C probíhá ve dvou cyklech podobně jako P^A. První cyklus probíhá podle obrázku 4. a druhý cyklus sloupcové histogramy pouze vyprazdňuje.

Řadící algoritmy

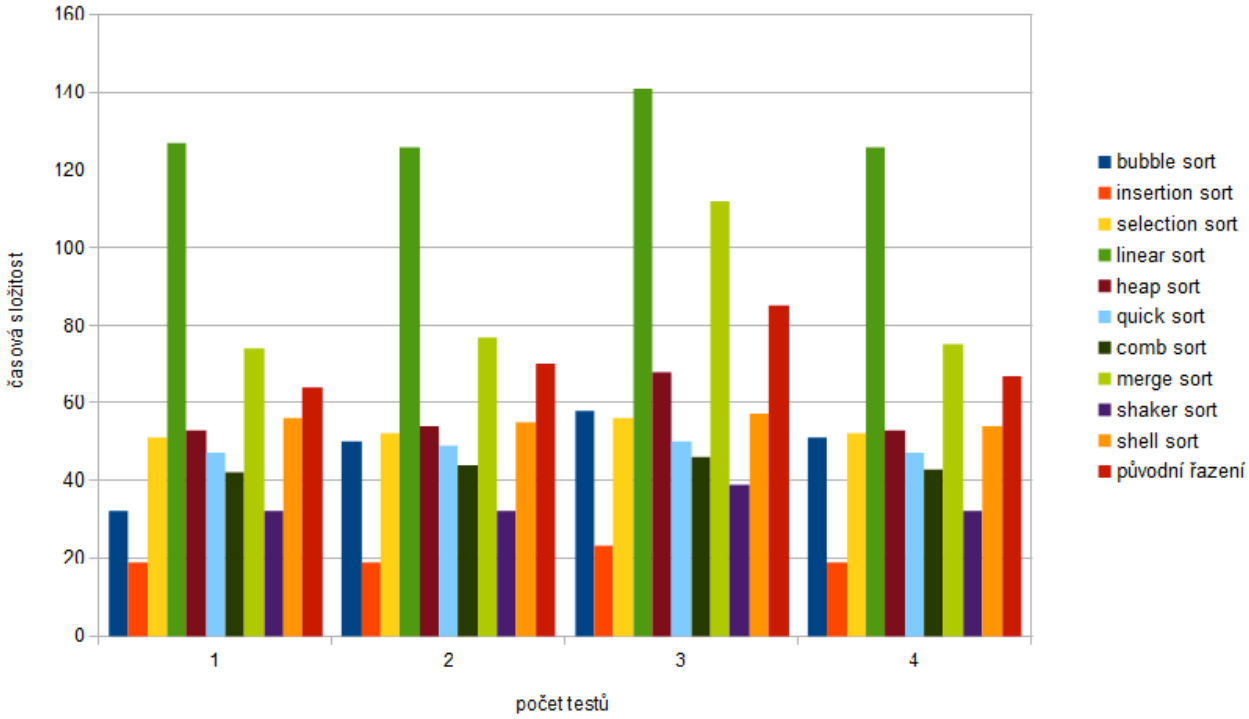
Jednou z nejdůležitějších částí algoritmu mediánového filtru, je výběr samotné mediánové hodnoty. Proto, aby jsme byli schopni tuto hodnotu vybrat, potřebujeme mít všechny hodnoty v jádru masky (místo uložení všech barevných složek, nacházejících se pod maskou v daném okamžiku). Původní algoritmus využíval řadícího algoritmu *counting sort* [3] (též *ultra sort*, nebo *math sort*.

Counting sort řadí pomocí vytváření pole četností výskytu (každý index pole reprezentuje řazené číslo a hodnota pole na tomto indexu reprezentuje četnost výskytu). Poté stačí pouze vypsat jednotlivé indexy tolikrát, kolik je hodnota v poli na daném indexu. Pole je tak přirozeně seřazeno, neboť $i < i + 1$. Tato struktura je totožná s histogramem.

Při posunu, tzn přidání H^+ a odebrání H^- dochází celkem k 3×256 operací sčítání a 3×256 operací odečítání v rámci histogramů. Proto byla provedena analýza řadících algoritmů, které by nepoužívaly pole o velikosti 256, ale pouze $2r + 1$. Důsledkem je odstranění operací sčítání a odčítání histogramů, ale pouhé prohazování prvků mezi jádrem masky a sloupcovými histogramy. Hodnocené kritérium byla časová složitost při opětovném přidávání a odebírání $\sqrt{n}$ prvků, kde n je počet prvků v jádře masky. Testovaná data byla náhodně generována.

Závěr

Byla provedena analýza na úrovni instrukcí, ze které vyplynulo, že algoritmus nejvíce využívá nejméně složitou operaci sčítání. Došlo k zrychlení algoritmu (medián měření původního algoritmu $\tilde{x} = 81\ ms$ a optimalizovaného $\tilde{x} = 76\ ms$), v důsledku redukce podmínek a segmentace. Zrychlení bylo pouze v řádech jednotek milisekund. Testy ukázaly, že řadící algoritmus *insertion sort* (stabilní řadící algoritmus s časovou složitostí $[O(n),\ O(n^2)]$) je pro danou úlohu nejrychlejší, přibližně tři-krát, viz Obrázek 5. Po jeho implementaci se očekává zrychlení v řádech desítek milisekund.



Obrázek 5. Časová složitost řadících algoritmů při parametrech: $n = 16$, $\sqrt{n} = 4$ a počet opakování $= 160000$

Poděkování

Práce byla podporována projektem SGS18/PrF/2014.

Literatura

[1] MOLEK, Vojtěch. Vylepšení kvality videa pořízeného web kamerou [online]. 2013 [cit. 2014-03-25]. Bakalářská práce. OSTRAVSKÁ UNIVERZITA V OSTRAVĚ, Přírodovědecká fakulta. Vedoucí práce Petr Hurtík. Dostupné z: http://theses.cz/id/isnrix/.

[2] Perreault, S.; Hebert, P., Median Filtering in Constant Time Image Processing, IEEE Transactions , vol.16, no.9, pp.2389,2394, Sept. 2007. Dostupné z: http://nomis80.org/ctmf.pdf.

[3] Donald E. Knuth. 1998. The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[4] FOG, Agner. TECHNICAL UNIVERSITY OF DENMARK. Instruction tables Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs. 2013. Dostupné z:http://www.agner.org/optimize/instruction_tables.pdf.

Abstract

This contribution is related to analysis and optimization of median filter. It's dealing with analysis on instructions level, improving of logic based on data segmentation and parallelism. Further it focus on comparing and choosing suitable sorting algorithm for particular problem.