



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NELINEÁRNÍ FILTRACE 3D OBRAZOVÝCH DAT
NONLINEAR FILTERING OF 3D IMAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAEL ŠKRÁŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2022

Zadání bakalářské práce



23513

Student: **Škrášek Michael**

Program: Informační technologie

Název: **Nelineární filtrace velkých 3D obrazových dat**
Non-Linear Filtering of Large 3D Images

Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy zpracování obrazu a problematiku nelineárních obrazových filtrů (medián, bilaterální filtr, non-local means filtr, apod.).
2. Seznamte se s metodami efektivní realizace těchto filtrů pro velká objemová data, která nelze celá držet v operační paměti.
3. Zvolte jeden či více filtrů a vytvořte jejich základní implementaci pro 3D obrazová data.
4. Zvolené filtry rozšiřte o podporu pro velká objemová data.
5. provedte experimenty a porovnejte dosažené. Diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát, nebo video, prezentující vaši práci, její cíle a výsledky.

Literatura:

- Buades *et al.*, "A non-local algorithm for image denoising", CVPR, 2005 (<https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.9157>).
- Banterle *et al.*, "A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain", Computer Graphics Forum, 2011 (<https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.02078.x>).

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Španěl Michal, Ing., Ph.D.**

Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.

Datum zadání: 1. listopadu 2021

Datum odevzdání: 11. května 2022

Datum schválení: 2. listopadu 2021

Abstrakt

Tato bakalářská práce se zabývá návrhem a efektivní implementací nelineárních filtrů, bilaterálního filtru a non-local means filtru, pro odstraňování šumu z 3D obrazových dat. Implementované filtry jsou optimalizovány různými technikami, jako třeba paralelním zpracováním, integračním obrázkem, vzorkováním atd. Práce obsahuje multiplatformní řešení těchto filtrů v jazyce C# a dvě demonstrační aplikace – okenní aplikaci pro Windows a multiplatformní konzolovou aplikaci – demonstrující využití knihovny. Rychlosť a výstupy implementovaných filtrů byly otestovány a porovnány s open source referenčními filtry (Itk, scikit-image), přičemž výsledkem je výrazně rychlejší filtrování představené implementace.

Abstract

This bachelor thesis deals with design and effective implementation of nonlinear filters, bilateral filter and non-local means filter, for denoising 3D image data. The implemented filters are optimized by various techniques such as parallel processing, integral image, sampling etc. This thesis project includes cross-platform solution of these filters in C# language and two demonstration applications – a GUI application for Windows and a cross-platform console application – demonstrating the use of the library. The performance and output of implemented filters were tested and compared against open source reference filters (Itk, scikit-image), resulting in significantly faster filtering of the presented implementation.

Klíčová slova

nelineární filtrace, bilaterální filtr, non-local means filtr, 3D bilaterální filtr, 3D non-local means filtr, nelineární filtrace volumetrických dat, nelineární filtrace 3D obrazových dat, rychlý bilaterální filtr, rychlý non-local means filtr

Keywords

nonlinear filters, bilateral filter, non-local menas filter, 3D bilateral filter, 3D non-local means filter, non-linear filters for volumetric data, nonlinear filters for 3D image data, fast bilateral filter, fast non-local means filter

Citace

ŠKRÁŠEK, Michael. *Nelineární filtrace 3D obrazových dat*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Španěl, Ph.D.

Nelineární filtrace 3D obrazových dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Španěla, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michael Škrášek
9. května 2022

Poděkování

Děkuji vedoucímu mé bakalářské práce, Ing. Michalu Španělovi, Ph.D., za odborné vedení, konzultace a přínosné návrhy k práci.

Obsah

1	Úvod	2
2	Nelineární filtry	3
2.1	3D obrazová data	3
2.2	Bilaterální filtr	4
2.3	Non-local means filtr	7
3	Návrh řešení nelineárních filtrů pro 3D obrazová data	11
3.1	Optimalizační techniky	11
3.2	Rychlý bilaterální filtr a rozšíření do 3D	13
3.3	Rychlý non-local means filtr a rozšíření do 3D	18
4	Implementace nelineárních filtrů pro 3D obrazová data	20
4.1	Vybrané technologie a struktura řešení	20
4.2	Implementace volumetrických dat	22
4.3	Implementace rozhraní filtru	22
4.4	Implementace 3D bilaterálního filtru	25
4.5	Implementace 3D non-local means filtru	26
5	Výsledky	28
5.1	Referenční filtry	28
5.2	Vstupní data	28
5.3	Průběh testování a testovací prostředí	29
5.4	Bilaterální filtr pro volumetrická data	30
5.5	Non-local means filtr pro volumetrická data	32
6	Závěr	35
Literatura		36
A	Obsah přiloženého paměťového média	37
B	Plakát	38

Kapitola 1

Úvod

Při pořizování 3D obrazových dat se muže v datech objevit nechtěný šum, např. ve zdravotnictví při pořizování výpočetní tomografie či magnetické resonance. Tento šum lze odstranit pomocí speciálních filtrů ... komplexnějšími z těchto filtrů, bilaterálním filtrem a non-local means filtrem, se zabývá tato práce.

Cílem této práce je seznámení se zmíněnými filtry a optimalizačními technikami pro zpracovávaní obrazu (filtraci), následně vybrané filtry efektivně implementovat pro filtraci 3D obrazových dat.

Tato práce je rozdělena do 6 kapitol, kde 1. kapitolou je tento úvod. Ve 2. kapitole je rozepsán teoretický základ nutný k návrhu a implementaci filtrů, tj. popis 3D obrazových dat v sekci 2.1 a popis bilaterálního filtru a non-local means filtru v sekcích 3.2 a 3.3. Ve 3. kapitole se nachází rozbor optimalizačních technik pro filtraci dat (sekce 3.1) a návrhy modelu bilaterálního filtru (sekce 3.2) a modelu non-local means filtru (sekce 3.3) s běžnými i vlastními optimalizacemi. Kapitola 4 se pak zabývá implementací těchto filtrů – tj. vybranými technologiemi a strukturou řešení v sekci 4.1, implementací 3D obrazových dat v sekci 4.2 a rozhraním filtrů v sekci 4.3. Kapitola je pak zakončena implementačními informacemi a zajímavostmi v sekci 4.4 pro bilaterální filtr a 4.5 pro non-local means filter. V 5. kapitole se nachází testování implementovaných filtrů v prostředí popsaném v sekci 5.3, na datech popsaných v sekci 5.2 a porovnání výsledků proti referenčním filtrům definovaných v sekci 5.1. Testování bilaterálního filtru se nachází v sekci 5.4 a testování non-local means filtru je rozepsané v sekci 5.5. Práce je zakončena závěrečnou 6. kapitolou, ve které je osobní zhodnocení výsledků a nastíněn další možný vývoj či rozšíření tohoto projektu.

Kapitola 2

Nelineární filtry

Cílem této kapitoly je seznámit čtenáře s teoretickým základem nutným k pochopení návrhu (viz kapitola 3) a implementace (viz kapitola 4) nelineárních filtrů pro 3D obrazová data, a to bilaterálního filtru a non-local means filtru. Kapitola se nejdřív v sekci 2.1 zaměří na 3D obrazová data (formát, možné využití …), která budou vstupem bilaterálního filtru a non-local means filtru. Kapitola pak pokračuje rozbořem bilaterálního filtru (sekce 2.2) a non-local means filtru (sekce 2.3) – principy, matematický popis, parametry a časová složitost.

2.1 3D obrazová data

Volumetrická či 3D obrazová data jsou data, která drží informace o konkrétním modelu ve třech rozměrech, což přináší více dostupných informací a jejich lepší interpretaci. Využití volumetrických dat je velmi rozšířeno ve zdravotnictví – magnetická resonance (MRI sken), výpočetní tomografie (CT sken) … a dalších odvětvích.

Volumetrická data se skládají z voxelů poskládaných do trojrozměrného pole, kde voxel obsahuje intenzitu daného bodu (případně RGB). Volumetrická data se obvykle pořizují po snímcích, které jsou následně rekonstruovány do 3D pole, přičemž může být rozdíl v reálných mezerách mezi snímky a mezi pixely ve snímcích, proto se k volumetrickým datům přikládá meta informace o skutečném rozložení voxelů (tzv. True Size či Spacing).

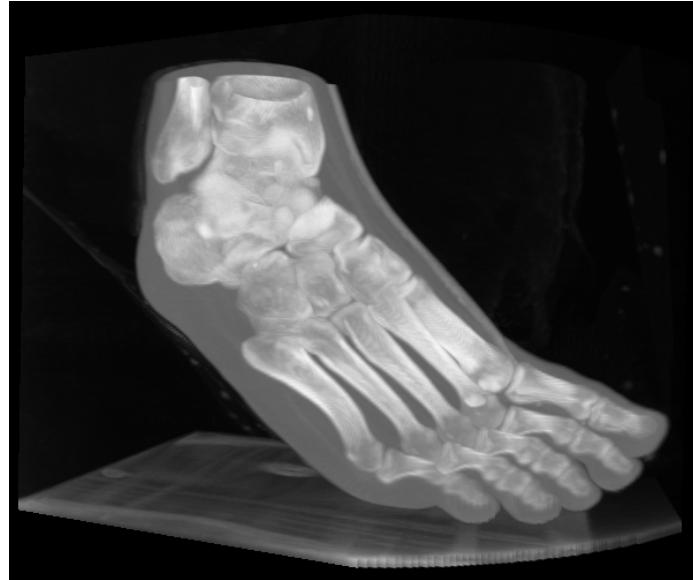
Formáty uložení volumetrických dat

Volumetrická data mohou být uložena kompaktně jako jeden soubor, nebo rozdělena po snímcích:

- RAW data je formát, který obsahuje pouze kompaktní volumetrická data bez jakýchkoliv metadat – pro další práci je nezbytné dodat další informace (rozměry dat, datový typ voxelů …).
- NRRD (Nearly Raw Raster Data) je robustní formát pro ukládání kompaktních volumetrických dat, obsahuje možnosti ukládání jako oddělená data a metadata, datový typ voxelů, komprese dat, způsob ukládání dat (endianess) … a podporu pro množství metadat (viz dokumentace¹). Umožňuje reprezentaci RAW dat dopsáním NRRD hlavičky se správnými metadaty.

¹<http://teem.sourceforge.net/nrrd/format.html>

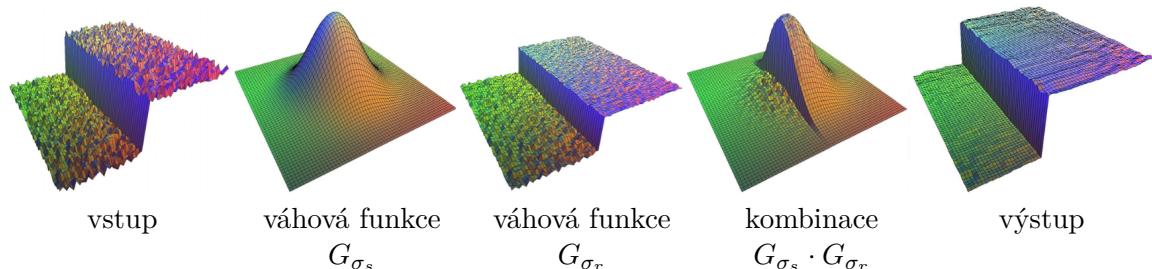
- DICOM (Digital Imaging and Communications in Medicine) je rozšířený standard pro komunikaci a správu lékařských dat, a tedy i volumetrických dat. Volumetrická data ukládá a spravuje po snímcích (viz dokumentace²).



Obrázek 2.1: Ukázka volumetrických dat (výpočetní tomografie chodidla) vykreslených pomocí raycastingu.

2.2 Bilaterální filtr

Durand a Dorsey [2] popisují bilaterální filtr jako nelineární vyhlazovací filtr zachovávající hrany, jehož výstupem je vážený průměr vstupu. Váhy jednotlivých bodů závisí na dvou Gaussových funkciích. Vstupem první Gaussovy funkce je vzdálenost bodů (klasický Gaussův filtr), díky funkčním vlastnostem Gaussovy funkce se váhy nepočítají přes celý vstup, ale jen z blízkého okolí bodu. Vstupem druhé Gaussovy funkce je rozdíl intenzit bodů, což má za důsledek snížení vah bodů s velkým rozdílem intenzit, neboli vlastnost zachování hran (viz obrázek 2.2).



Obrázek 2.2: Grafické znázornění průběhu bilaterálního filtrování, prostorové váhové funkce G_{σ_s} a rozsahové váhové funkce G_{σ_r} kolem centrálního bodu. Barvy jsou použity pouze pro demonstraci tvarů. Převzato z [2].

²<https://dicom.nema.org/medical/dicom/current/output/html/part01.html>

Bilaterální filtr je definován jako:

$$I_{out}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) G_{\sigma_r}(\|I(x_i) - I(x)\|) G_{\sigma_s}(\|x_i - x\|) \quad (2.1)$$

$$W_p = \sum_{x_i \in \Omega} G_{\sigma_r}(\|I(x_i) - I(x)\|) G_{\sigma_s}(\|x_i - x\|) \quad (2.2)$$

kde:

- I jsou vstupní data (obrazová data, volumetrická data)
- I_{out} jsou výstupní (vyfiltrovaná) data
- x jsou souřadnice právě filtrovaného bodu
- W_p je normalizační faktor
- Ω je množina bodů tvořící vyhledávací oblast (okno, krychle) se středem v bodě x , velikost oblasti závisí na parametru r , kde strana oblasti je $2r + 1$
- G_{σ_r} je rozsahová Gaussova funkce s parametrem σ_r (tzv. range kernel)
- $\|I(x_i) - I(x)\|$ je absolutní hodnota rozdílu intenzit mezi právě filtrovaným bodem x a bodem x_i , přes který je počítán vážený průměr
- G_{σ_s} je prostorová Gaussova funkce s parametrem σ_s (tzv. spatial/domain kernel)
- $\|x_i - x\|$ je vzdálenost mezi právě filtrovaným bodem x a bodem x_i , přes který je počítán vážený průměr

Parametry bilaterálního filtru

Bilaterální filtr má 2 vstupní parametry – σ_r (tzv. range sigma) a σ_s (tzv. spatial sigma), které jsou směrodatné odchylky Gaussových funkcí. Parametr σ_r řídí rozsahovou Gaussovou funkci (závažnost rozdílu intenzit bodů) a parametr σ_s řídí prostorovou Gaussovou funkci, tzn. řízení závažnosti vzdáleností bodů (viz obrázek 2.3). Parametr r (vstupní nebo vypočítaný z σ_s) udává velikost oblasti, přes kterou bilaterální filtr počítá vážený průměr. Jelikož velikost oblasti přímo ovlivňuje dobu filtrování, v praxi se proto parametr r vybírá co nejmenší z rozmezí $(0; 3\sigma_s]$. Toto rozmezí je stanovené pravidlem tří sigma [6], které říká, že 99,73 % nenulových hodnot normálního rozložení leží v rozmezí $\langle -3\sigma; 3\sigma \rangle$, a tedy vážený průměr je počítaný přes 99,73 % bodů s podstatně významnou vahou. Příliš malý parametr r negativně ovlivňuje výsledek filtru.



Obrázek 2.3: Vliv parametrů na výsledek bilaterálního filtrování.

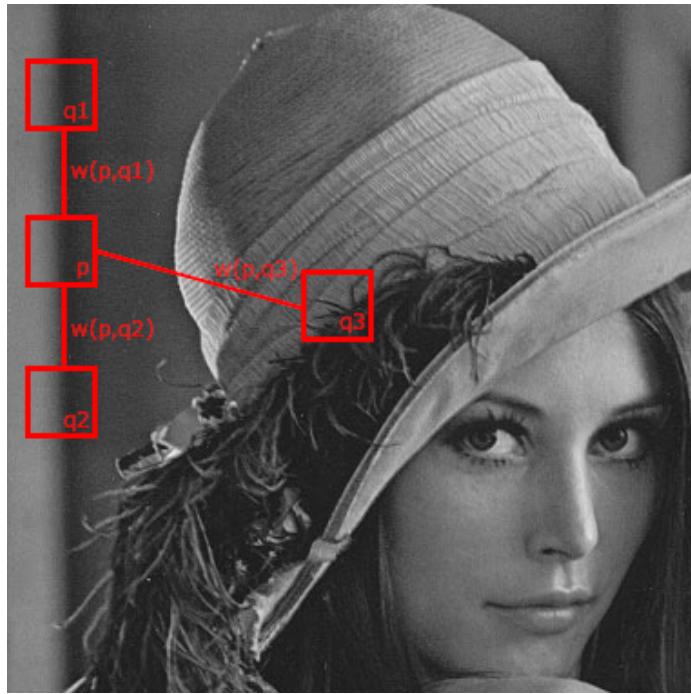
Časová složitost

Vstupní data	Rozměry	Časová složitost
signál (1D)	N	$\mathcal{O}(N \cdot (2r + 1))$
obrazová data (2D)	$W \times H$	$\mathcal{O}(W \cdot H \cdot (2r + 1)^2)$
volumetrická data (3D)	$W \times H \times D$	$\mathcal{O}(W \cdot H \cdot D \cdot (2r + 1)^3)$

Tabulka 2.1: Tabulka reprezentující časovou složitost bilaterálního filtru.

2.3 Non-local means filtr

Non-local means filtr prvně představil Buades [1] jako algoritmus, kde výstupní intenzita pixelu je počítána jako vážený průměr všech pixelů v obraze a nejen z blízkého okolí (odtud název non-local means). Váhy pixelů závisí na jejich podobnosti vůči vstupnímu pixelu, kde podobnost pixelů je souzena na základě okolí centrováných kolem porovnávaného pixelu (viz obrázek 2.4). Kvůli časové složitosti algoritmu jsou však podobné pixely vyhledávány jen v předem definovaných vyhledávacích oblastech kolem filtrovaného pixelu.



Obrázek 2.4: Strategie non-local means filtru: podobné oblasti (kolem pixelů q_1 a q_2) vůči oblasti kolem vstupního pixelu p vrátí vysoké váhy ($w(p, q_1)$, $w(p, q_2)$) a méně podobná oblast kolem pixelu q_3 vrátí nižší váhu $w(p, q_3)$. Převzato z [1].

V původní verzi algoritmu [1] Buades počítal podobnosti okolí na základě vážené euklidovské vzdálenosti intenzit jednotlivých pixelů v okolí, kde váhy odpovídají Gaussově funkci – označuje se **NLM-P α** , kde α připomíná využití vstupního parametru a jako směrodatné odchylky Gaussovy funkce. Jacques Froment v [3] zmiňuje, že se v praxi gaussovská část často neimplementuje, jelikož přínos na výsledek filtrování, za cenu větší komplexnosti a dalšího parametru, je příliš malý. Filtr bez parametru a se značí **NLM-P** a je definován jako:

$$I_{out}(i) = \frac{1}{Z(i)} \sum_{j \in \Omega} I(j) \cdot w(i, j) \quad (2.3)$$

$$Z(i) = \sum_{j \in \Omega} w(i, j) \quad (2.4)$$

$$w(i, j) = e^{-\frac{\|B(i) - B(j)\|_2^2}{h^2}} \quad (2.5)$$

kde:

- I jsou vstupní data (obrazová data, volumetrická data)
- I_{out} jsou výstupní (vyfiltrovaná) data
- i jsou souřadnice právě filtrovaného bodu
- Ω je vyhledávací oblast (okno, krychle) se středem v bodě i , tuto oblast filtr prohledává pro podobná okolí (patche), velikost vyhledávací oblasti je vstupním parametrem
- $Z(i)$ je normalizační faktor
- $w(i, j)$ je váhová funkce
- $B(p)$ je okolí kolem bodu p (tzv. patch), jedná se o vektor intenzit bodů v okolí, velikost okolí je vstupním parametrem
- $\|B(i) - B(j)\|_2^2$ je euklidovská vzdálenost intenzit bodů mezi patchi $B(i)$ a $B(j)$
- h je vstupní parametr řídící váhovou funkci

Parametry filtru non-local means

Filtr NLM-P (běžná varianta klasického NLM-Pa) má 3 vstupní parametry, parametr h řídící váhovou funkci, parametr r nastavující velikost vyhledávacího okna, kde strana oblasti je $2r + 1$, a parametr p řídící velikost patche, kde strana patche je $2p + 1$. Filtr NLM-Pa má další parametr a , který má pozitivně zlepšit vlastnosti filtru.

Časová složitost

Časová složitost klasického non-local means filtru je $\mathcal{O}(N \cdot (2r + 1)^d \cdot (2p + 1)^d)$, kde N je velikost vstupních dat, r je poloměr vyhledávací oblasti, p je poloměr patche a d je počet rozměrů vstupních dat (1D, 2D, 3D ...). Složitost takového filtru je velká, proto byly navrženy přístupy pro zbavení závislosti časové složitosti na poloměru patche p , jedním z přístupů, který je snadno rozšířitelný do 3D, je optimalizace pomocí integračního obrázku. Jacques Froment v [3] zmiňuje další 2 techniky – využití FFT či sumy invariantních řádků (sum of invariant lines).

Integrační obrázek

Integrační obrázek je datová struktura a algoritmus pro rychlé a efektivní počítání ploch obdélníků v obrazových datech [5]. Pro rychlé spočítání je potřeba vstupní data nejdřív předpočítat do integračního obrázku, který má stejné rozměry jako vstupní data. Integrační obrázek pro 2D obrazová data je definovaný jako:

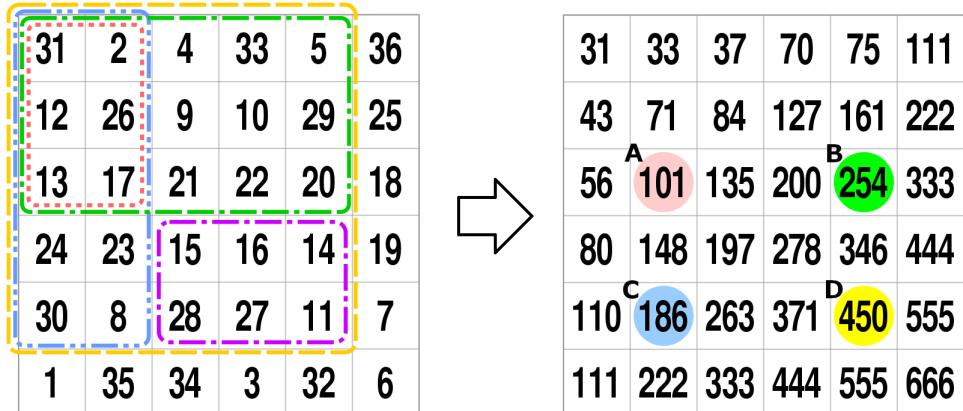
$$I(x, y) = \sum_{x' < x} \sum_{y' < y} i(x', y') \quad (2.6)$$

kde $I(x, y)$ je hodnota integračního obrázku v bodě $[x, y]$ a $i(x', y')$ je intenzita obrazových dat v bodě $[x', y']$, konkrétní plochu S ve 2D obrazových datech lze spočítat podle vzorce:

$$S = I(D) + I(A) - I(B) - I(C) \quad (2.7)$$

kde obdélník plochy S je dán počátečním bodem $[A_x + 1, A_y + 1]$ a koncovým bodem $[D_x, D_y]$ (viz obrázek 2.5).

Z N přístupů do paměti a $N - 1$ aritmetický operací (kde N je počet bodů v obdélníku) je komplexnost získání obsahu S snížena na 4 přístupy do paměti a 3 matematické operace, za cenu inicializace integračního obrázku a využité paměti.



Obrázek 2.5: Ilustrace vztahu vstupních obrazových dat (vlevo) a integračního obrázku (vpravo). Barvy ve vstupních datech znázorňují plochy odpovídající hodnotám totožných barev v integračním obrázku, fialovou plochu lze spočítat vzorcem 2.7. Převzato z [5].

Integrační obrázek pro 2D obrazová data lze spočítat jedním průchodem vstupních dat dle vzorce:

$$I(x, y) = I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) + i(x, y) \quad (2.8)$$

při průchodu daty směrem od počátečního bodu $[0, 0]$ po koncový bod $[W, H]$, nejdřív je však nutné vypočítat hodnoty integračního obrázku podél os x a y dle vzorce:

$$I(p) = I(p - 1) + i(p)$$

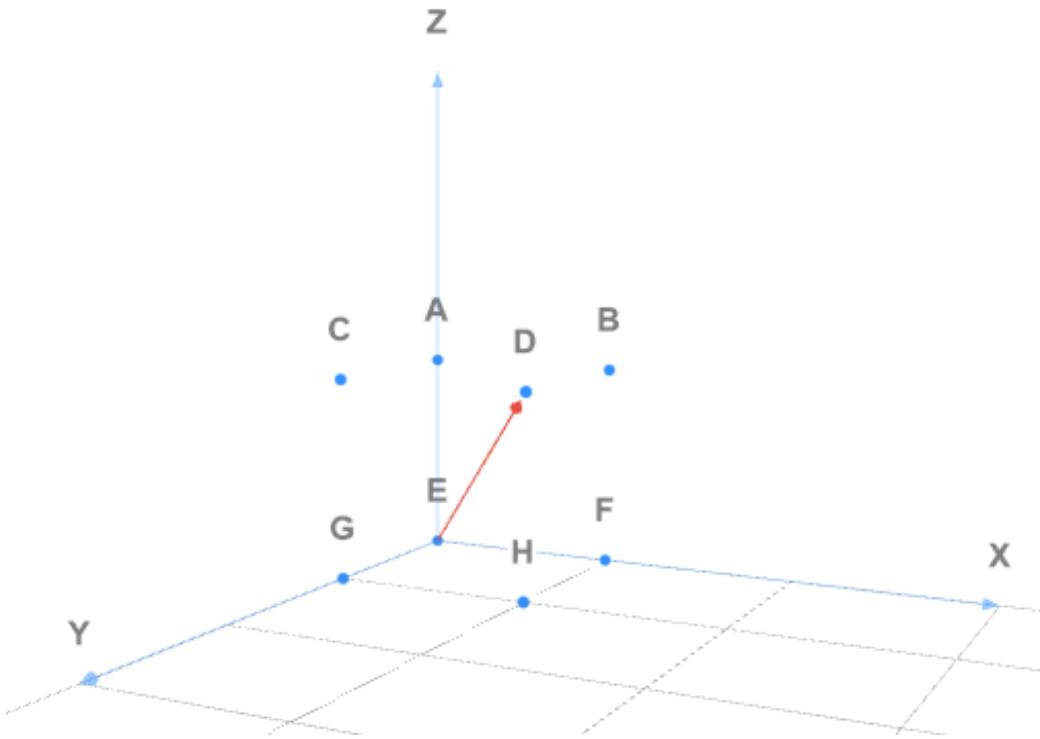
Při rozšíření integračního obrázku do 3D je kromě spočítání os x, y , a z nutné před konečným průchodem spočítat také roviny xy , xz a yz , které se počítají obdobným způsobem jako 2D integrační obrázek (viz rovnice 2.8). Konečný inicializační průchod 3D integračního obrázku je definován jako:

$$I(x, y, z) = I(H) + I(C) - I(G) + I(B) - I(F) - I(A) + I(E) + i(x, y, z) \quad (2.9)$$

kde A, B, C, E, F, G, H jsou souřadnice reprezentující body v 3D prostoru (viz obrázek 2.6) a při inicializaci jsou definovány jako:

$$\begin{aligned} A &= [x - 1, y - 1, z] \\ B &= [x, y - 1, z] \\ C &= [x - 1, y, z] \\ D &= [x, y, z] \\ E &= [x - 1, y - 1, z - 1] \\ F &= [x, y - 1, z - 1] \\ G &= [x - 1, y, z - 1] \\ H &= [x, y, z - 1] \end{aligned}$$

kde bod D označuje právě počítanou hodnotu v 3D integračním obrázku.



Obrázek 2.6: Ilustrace bodů při používání 3D integračního obrázku. Červený vektor zobrazuje směr vytváření integračního obrázku.

Objem V konkrétní oblasti ve volumetrických datech lze spočítat dle vzorce:

$$V = I(D) - I(H) + I(G) - I(C) + I(F) - I(E) + I(A) - I(B) \quad (2.10)$$

kde oblast (kvádr) definující objem V je daná počátečním bodem $[A_x + 1, A_y + 1, A_z + 1]$ a konečným bodem $[D_x, D_y, D_z]$, způsob uspořádání bodů $A - H$ v 3D prostoru je totožný způsobu v obrázku 2.6.

Složitost získání objemu oblasti V je tak snížena z N přístupů do paměti a $N - 1$ aritmetických operací (N je počet bodů v kvádru) na konstantních 8 přístupů do paměti a 7 matematických operací za cenu inicializace 3D integračního obrázku.

Kapitola 3

Návrh řešení nelineárních filtrů pro 3D obrazová data

Tato kapitola se zabývá návrhem rychlých nelineárních filtrů (bilaterální filtr a non-local means filtr) pro 3D obrazová data. V kapitole jsou nejdříve rozebrány optimalizační techniky používané pro rychlé zpracování dat a filtraci (sekce 3.1). Dále je v kapitole, na základě teoretického základu z kapitoly 2, navržen model bilaterálního filtru (sekce 3.2) a model non-local means filtru s integračním obrázkem (sekce 3.3), ve kterých jsou dříve zmíněné optimalizační techniky využity.

3.1 Optimalizační techniky

Kromě zmenšení časové složitosti algoritmu lze rychlosť filtrování optimalizovat technikami, které lépe využívají potenciálu počítače, na němž filtrování probíhá.

Paralelní zpracování

Jednou z technik pro zrychlení zpracování je zpracovávat vstup paralelně na více jádrech procesoru, díky tomu je čas potřebný ke zpracování vstupu zkrácen počtem jader, v praxi je zrychlení o něco menší kvůli režii potřebné k přepínání kontextu mezi vlákny. U malých vstupních dat je možné, že režie vytváření vláken a přepínání kontextu předčí čas ušetřený paralelním zpracováním. Při zpracovávání na více vláknech je nezbytné vyhnout se co možná nejvíce konfliktům při přístupu k datům sdíleným mezi vlákny, jelikož využití zámků vede k častému přepínání kontextu, což může vést ke zpomalení zpracování vstupu.

Dalším přístupem pro paralelní zpracování dat je využití SIMD operací (Single Instruction, Multiple Data), které využívají existence velkých datových SIMD registrů (128b, 256b) v moderních počítačích. Tyto registry umožňují přístup k několika číselným proměnným zároveň (tzv. vektory), a procesor tedy může provádět základní matematické operace (sčítání, násobení, dělení ...) na vektorech, tzn. paralelně na číselných proměnných ve vektorech.

Předpočítání parametrů a funkcí

Filtraci popisují matematické rovnice, které v cyklu opakovaně počítají matematické výrazy, některé z parametrů této výrazu lze předpočítat jako koeficienty bez nutnosti je počítat při každé iteraci cyklu. Zvláště výhodné je předpočítat dělení koeficientem na násobení

inverzním koeficientem (př. $f(x) = \frac{x}{\pi} \rightarrow f(x) = \frac{1}{\pi} \cdot x$), protože dělení je mnohem nákladnější než násobení.

Pokud je znám možný rozsah vstupů funkce, je možné výsledky této funkce předpočítat do vyhledávací tabulky (tzv. lookup table), vstupy funkce se namísto výpočtu využijí pro indexaci ve vyhledávací tabulce. Pro přesné mapování vstupů na výstupy je potřeba, aby funkce měla celočíselné vstupy, pokud jsou vstupem desetinná čísla, nelze namapovat desetinný vstup na přesnou pozici a výsledek ve vyhledávací tabulce. Pro desetinné vstupy lze využít přibližné approximující řešení funkce, tj. navzorkování funkce na N vzorků, kde se desetinný vstup namapuje na nejbližší možný výstup ve vyhledávací tabulce (tzv. sampling). Počet vzorků ovlivňuje přenosnost approximujícího řešení ke skutečnému. Předpočítání funkce je výhodné pokud se musí funkce často opakovat a pokud její provedení je dostatečně komplexní (mocnina, odmocnina, goniometrické funkce ...).

Optimalizace kódu pro hardware

Načítání dat z paměti je pomalé a každý přístup do paměti, který není uložený ve vyrovnávací paměti (cache), výrazně zpomaluje program. Proto je důležité zachovávat princip lokality dat a zpracovávat data přístupem, který tento princip dodržuje (tzv. cache-friendly). Běžným příkladem je zpracovávání vícerozměrného pole, kde prvky pole jsou fyzicky uloženy vedle sebe, ale logicky jsou poskládané do matice (ve 2D) ... pro minimalizaci přístupů přímo do paměti (cache miss) je nutné správné pořadí vnořených cyklů přistupujících k prvkům ve vícerozměrném poli.

Dalším prvkem hardwaru, který může výrazně ovlivnit rychlosť, je prediktor skoků (branch predictor). Prediktor skoků má za úkol dopředu předpovědět výsledek podmínky a díky tomu optimalizovat zpracovávání instrukcí, avšak pokud jsou vstupem podmínky špatně předpověditelná data (př. zašuměná obrazová data), předpověď skoku do špatné větve programu (branch misprediction) může stát hodně procesorového času, zvláště pokud je skok opakován špatně predikován ve vnořených smyčkách zpracovávajících N -rozměrná data (obrazová data, volumetrická data ...). Omezení větvění programu může být obtížné, jednou z možností je nahrazení podmínky aritmetikou zajišťující stejný výsledek, která využívá způsobu ukládání čísel v paměti. Podmínky se objevují i ve vestavěných funkcích jako Min, Max, Abs ... tohoto faktu lze využít u předpočítávání funkcí, kde může být výhodné předpočítat funkci i pro čísla mimo definovaný obor, tzn. předpočítání i pro záporné hodnoty symetrické funkce místo využití funkce absolutní hodnoty, předpočítání funkce za krajní hodnoty (padding) místo využití funkcí Min/Max ... Příliš mnoho předpočítávaných dat může negativně ovlivnit schopnost systému výhodně ukládat data do vyrovnávací paměti.

3.2 Rychlý bilaterální filtr a rozšíření do 3D

Bilaterální filtr pro zápis do paměti využívá pouze lokální proměnné a příslušnou oblast výstupních dat, nedochází tedy ke konfliktu mezi vlákny a filtr je ideální pro paralelní zpracování. Pro filtrování barevných dat lze využít SIMD instrukce, které mohou zpracovat sdílenou část algoritmu pro RGB složky barev najednou a zmenšit tak faktor zpomalení filtru.

Protože parametr r ovlivňuje velikost průměrované oblasti, a tedy přímo ovlivňuje rychlost filtru, je výhodné zvolit co nejmenší hodnotu, která nemá negativní dopad na výsledek filtrování. Zvolením $r = 2,5\sigma_s$ lze dosáhnout slušného výsledku průměrovaného přes 98,7 % nenulových hodnot [5] a výrazného zmenšení oblasti v porovnání s největším smysluplným parametrem $r = 3\sigma$ (viz obrázek 3.1).

Váhy uvnitř cyklu jsou počítány pomocí dvou Gaussových funkcí – prostorová G_{σ_s} a rozsahová G_{σ_r} (viz rovnice 2.1), Gaussovou funkci lze definovat jako $G_\sigma(x) = \exp(c \cdot x^2)$, kde $c = -\frac{0.5}{\sigma^2}$ je předpočítatelný koeficient. Předpočítání těchto funkcí je zásadní pro rychlé filtrování, neboť násobení spojené s matematickou funkcí \exp je časové náročné na výpočet.

Předpočítání rozsahové Gaussovy funkce

Vstupem rozsahové funkce G_{σ_r} je vzdálenost intenzit neboli absolutní hodnota rozdílu intenzit mezi vstupními body v datech. Pokud předpokládáme ukládání intenzity v obrazových datech na jednom bajtu (hodnoty 0-255), vstupem funkce pak může být vzdálenost v rozsahu 0-255, vyhledávací tabulka bude mít tedy 256 hodnot. Lze využít faktu, že absolutní hodnota dělá funkci symetrickou, a předpočítat i pro záporné vstupní hodnoty, vyhledávací tabulka pro rozsahovou Gaussovou funkci pak bude mít velikost 511 hodnot (255 záporných vstupů + 255 kladných vstupů + 1 vstup s nulovým rozdílem). Index do vyhledávací tabulky se pak spočítá jako $255 + I_1 - I_2$, kde I_1 a I_2 jsou intenzity porovnávaných bodů. Způsob předpočítání rozsahové Gaussovy funkce závisí na datovém typu reprezentující intenzitu, změna dimenze filtrujících dat tedy nijak nezmění tuto optimalizaci.

Předpočítání prostorové Gaussovy funkce

Vstupem prostorové funkce G_{σ_s} je vzdálenost mezi body, protože výpočet vzdálenosti je časově náročný úkon (odmocnina, násobení) a výsledek je desetinné číslo, je výhodné Gaussovou funkci obalit novou funkcí f a předpočítat ji. Pro 2D data je funkce f definovaná jako:

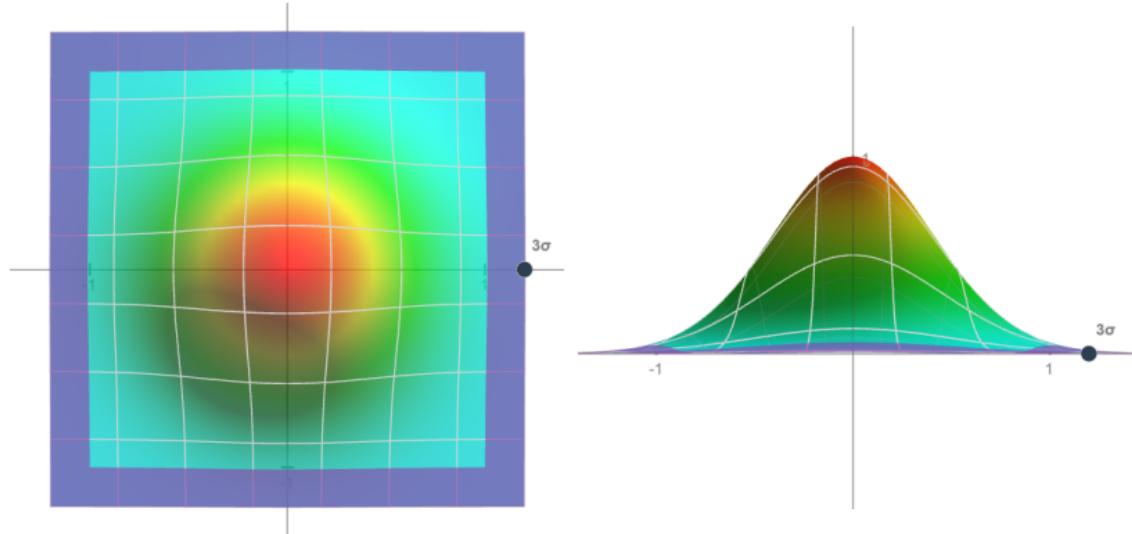
$$f(x, y) = G_{\sigma_s}(\sqrt{x^2 - y^2}) \quad (3.1)$$

$$f(x, y) = \exp(c \cdot (\sqrt{x^2 - y^2})^2) \quad (3.2)$$

$$f(x, y) = \exp\left(-\frac{0.5}{\sigma_s^2} \cdot (x^2 - y^2)\right) \quad (3.3)$$

kde parametry $x = p_x - c_x$ a $y = p_y - c_y$ jsou posuny od centrálního pixelu c k právě porovnávanému pixelu p . Obalení funkcí f umožní předpočítat hodnoty pro celočíselná vstupní data (x a y) v rozmezí $\langle -2,5\sigma_s; 2,5\sigma_s \rangle$, vyhledávací tabulka bude tedy mít formát dvourozměrného pole, kde rozměry odpovídají rozměru vyhledávací oblasti (viz obrázek 3.1). Pro 3D bilaterální filtr přibude souřadnice z a vyhledávací tabulka bude mít formát 3D pole se stejným rozměry jako vyhledávací oblast (krychle kolem centrálního bodu c). Pro 3D bilaterální filtr může určení vzdálenosti být problém kvůli odlišným vzdálenostem mezi

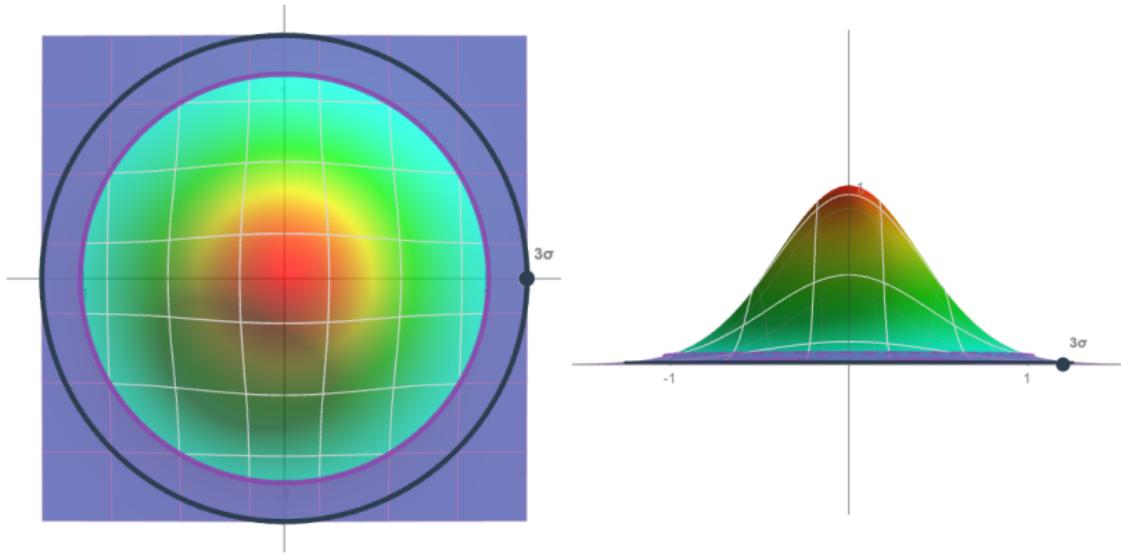
voxely způsobné druhem vytvoření volumetrických dat (viz volumetrická data 2.1), běžným řešením je rozdílné vzdálenosti nezohledňovat.



Obrázek 3.1: Ilustrace předpočítaných vah Gaussovy funkce pro 2D bilaterální filtr. Fialová oblast označuje ušetřenou oblast při použití parametru $r = 2,5\sigma_s$

Předpočítání vyhledávací oblasti

Z principu bilaterálního filtru, který předpokládá podobnost bodů na základě jejich vzdálenosti, se prostorová Gaussova funkce předpočítává jen do hranice $x^2 + y^2 = r^2$ a zbylé váhy ve vyhledávací tabulce zůstanou nulové, což zajistí, že váhy budou záviset pouze na vzdálenosti mezi porovnávanými body a ne na jejich pozici vůči sobě. Protože nulové body nemění nijak váhu, navrhoji je neprocházet vůbec a předpočítat vyhledávací oblast jako kruh (viz obrázek 3.2), to umožní snížit časovou složitost algoritmu z $\mathcal{O}(WH \cdot (2r + 1)^2)$ na $\mathcal{O}(WH \cdot \frac{\pi}{4}(2r + 1)^2)$, kde předpočítání vyhledávací oblasti znamená nastavení okrajů cyklu procházejícího vyhledávací oblast a předpočítání komplexních funkcí definující tyto okraje, tzn. $y \in \langle -r; r \rangle$ a $x \in \langle -\sqrt{r^2 - y^2}; \sqrt{r^2 - y^2} \rangle$. Komplexní funkci definující okraje souřadnice x lze předpočítat do 1D vyhledávací tabulky a zbavit se tak složitých operací násobení a odmocňování.

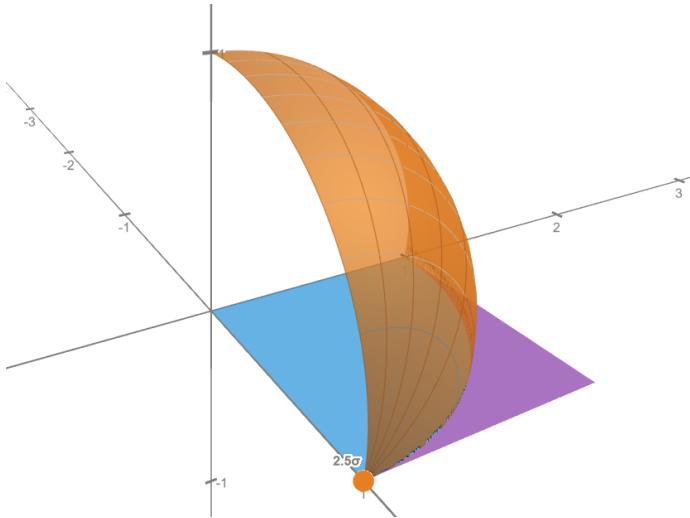


Obrázek 3.2: Ilustrace předpočítaných vah Gaussovy funkce pro 2D bilaterální filtr. Fialová oblast v rozmezí $x^2 + y^2 \geq (2, 5\sigma)^2$ označuje nízké váhy, které je možno vyřadit z váženého průměru bez významného dopadu na výsledek filtrování, ale s výrazným dopadem na zrychlení algoritmu.

Při rozšíření filtru do 3D (viz obrázek 3.3) se prostorová Gaussova funkce předpočítá pouze po hranici $x^2 + y^2 + z^2 = r^2$, a okraje cyklu budou tedy definované jako:

$$\begin{aligned}x &\in \langle -r, r \rangle \\y &\in \langle -\sqrt{r^2 - x^2}; \sqrt{r^2 - x^2} \rangle \\z &\in \langle -\sqrt{r^2 - x^2 - y^2}; \sqrt{r^2 - x^2 - y^2} \rangle\end{aligned}$$

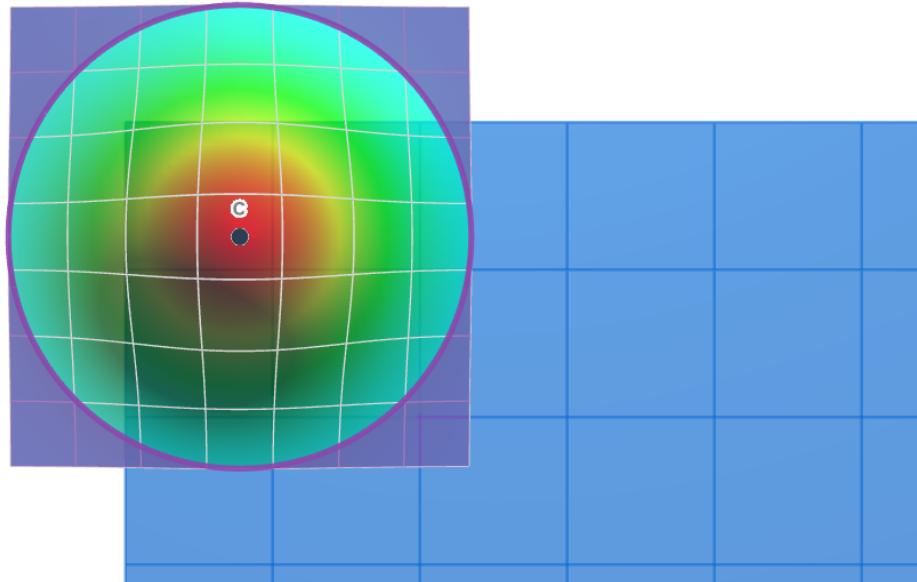
Komplexní funkci definující okraje souřadnice y lze předpočítat do 1D vyhledávací tabulky obsahující posuny hranic počítaných ze souřadnice x . Komplexní funkce definující okraje souřadnice z pak lze předpočítat do 2D vyhledávací tabulky obsahující posuny hranic počítaných ze souřadnic x a y . Časová složitost je tedy snížena z $\mathcal{O}(WHD \cdot (2r+1)^3)$ na $\mathcal{O}(WHD \cdot \frac{\pi}{6} (2r+1)^3)$ a nejsou použity složité operace násobení, odmocnina ... pro počítání okrajů vyhledávací oblasti.



Obrázek 3.3: Ilustrace hranice prohledávací oblasti Ω pro 3D bilaterální filtr. Modrá oblast označuje vysoké váhy Gaussovy funkce ve 2D (prohledávací oblast Ω), fialová oblast označuje ušetřenou část Gaussovy funkce ve 2D s nízkými (nulovými) váhami. Oranžová oblast, definovaná jako $f(x, y) = \sqrt{r^2 - x^2 - y^2}$ a $r = 2,5\sigma_s$, zobrazuje hranici mezi modrou a fialovou oblastí při rozšíření do 3D.

Překračování hranic obrazových dat

Protože při počítání váženého průměru bodů na okraji obrazových dat dochází k překračování hranic (viz obrázek 3.4), je nutné vstupní data obalit vrstvou (tzv. padding), která většinou obsahuje barvu od nejbližšího okraje. Gaussova funkce může přesáhnout okraj až o r bodů, takže data jsou vycpána r body ve všech směrech (6 směrů ve 3D).



Obrázek 3.4: Ilustrace prostorové Gaussovy funkce přesahující hranice 2D obrazových dat (modrá oblast) v okamžík filtrování pixelu v bodě c .

Alternativou je data neobalovat, ale využít funkce Min a Max pro převedení souřadnic mimo data na souřadnice nejbližšího okraje. Tyto dvě funkce lze spojit do jedné složené funkce definované jako:

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < x < N \\ N - 1 & N \leq x \end{cases}$$

kde $x \in \langle -r; N + r \rangle$ jsou souřadnice v obrazových datech a N je rozměr dat podél dané osy, díky tomu, že je znám rozsah vstupních data, lze tuto funkci předpočítat do vyhledávací tabulky a zbavit se tak funkcí Min a Max, které mohou způsobit špatné predikce skoků (branch missprediction – viz sekce 3.1). Předpočítat funkci $f(x)$ je nutné pro všechny souřadnice (x, y a z pro 3D), předpočítaná data budou mít formát 1D pole o velikosti $N + 2r$ (viz tabulka 3.1), kde N odpovídá rozměru obrazových dat podél určité osy, pro transformaci vstupních souřadnic na indexy slouží vzorec: $index = x + r$.

x	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$f(x)$	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	10	10	10	10

Tabulka 3.1: Tabulka předpočítané funkce složené z funkcí Min a Max, kde 1. řádek obsahuje vstupní souřadnice, 2. řádek obsahuje indexy, na které jsou vstupní souřadnice mapovány, a na 3. řádku jsou výstupní (předpočítané) souřadnice. Červené buňky znázorňují vstupní souřadnice mimo obrazová data.

Paměťové nároky a časová složitost 3D bilaterálního filtru

vstupní volumetrická data výstupní volumetrická data	$WHD \cdot S_{Byte}$ $WHD \cdot S_{Byte}$
předpočítaná rozsahová Gaussova funkce	$2 \cdot 255 + 1$
předpočítaná prostorová Gaussova funkce	$(2r + 1)^3 \cdot S_{Double}$
předpočítané okraje průměrované oblasti podél osy y	$(2r + 1) \cdot S_{Double}$
předpočítané okraje průměrované oblasti podél osy z	$(2r + 1)^2 \cdot S_{Double}$
předpočítané hranice volumetrických dat podél osy x	$(W + 2r) \cdot S_{Double}$
předpočítané hranice volumetrických dat podél osy y	$(H + 2r) \cdot S_{Double}$
předpočítané hranice volumetrických dat podél osy z	$(D + 2r) \cdot S_{Double}$

Tabulka 3.2: Paměťové nároky (paměť alokovaná na haldě) navrženého 3D bilaterálního filtru. První část tabulky obsahuje nezbytná data pro filtraci, 2. část tabulky obsahuje předpočítaná data. Koeficient S_{Byte} reprezentuje velikost 1 bajtu v paměti, S_{Double} reprezentuje velikost desetinného čísla v paměti (double precision).

Časová složitost navrženého filtru je $\mathcal{O}(WHD \cdot \frac{\pi}{6} (2r + 1)^3)$ což je přibližně $2 \times$ menší časová složitost ($\frac{\pi}{6} \approx 0,52$) než původních $\mathcal{O}(WHD \cdot (2r + 1)^3)$.

3.3 Rychlý non-local means filtr a rozšíření do 3D

Při zpracovávání non-local means filtrem, podobně jako u bilatelárního filtru, jsou paralelní zápisy do paměti nekonfliktní a lze tedy paralelně filtrovat bez použití zámků. Časová složitost klasického non-local means filtru závisí na velikosti patche a velikosti vyhledávací oblasti (viz non-local means filtr 2.3), závislosti na velikosti patche se dá zbavit použitím integračního obrázku. Pro rychlé filtrování je také nezbytné předpočítání váhové funkce, která se používá pro určování podobnosti patchů.

Využití integračního obrázku

Podobnost patche je počítaná na základě Euklidovské vzdálenosti, která je definovaná jako:

$$d(k, l)^2 = (k[1] - l[1])^2 + (k[2] - l[2])^2 + \dots + (k[N] - l[N])^2 \quad (3.4)$$

$$d(k, l)^2 = \sum_{i=1}^N (k[i] - l[i])^2 \quad (3.5)$$

kde $d(k, l)$ je vzdálenost mezi vektory intenzit k a l , N je velikost vektoru (počet bodů v patchi) – $(2p+1)^2$ ve 2D a $(2p+1)^3$ ve 3D. Použitím integračního obrázku lze z algoritmu odstranit tyto sumy obdélníků/kvádů (viz integrační obrázek v sekci 2.3), tzn. odstranění závislosti na velikosti patche z časové složitosti filtru. Protože se ve vzorci 3.4 kromě sčítání a odečítání také násobí, je využití integračního obrázku komplikované.

Jak zmiňuje Jacques Froment v [3], je potřeba lehce upravit algoritmus filtru a předpocítat integrační obrázek pro každý možný vektor posunu ve vyhledávacím okně, kde místo intenzity budou integrační obrázky obsahovat součty rozdílů intenzit na druhou, tzn. že vzorec inicializace 2D integračního obrázku bude pozměněn na:

$$I_n(x, y) = I_n(x-1, y) + I_n(x, y-1) - I_n(x-1, y-1) + (i(x, y) - i(x+t_x, y+t_y))^2 \quad (3.6)$$

kde I_n je integrační obrázek, $n \in \langle 0, N \rangle$ je index integračního obrázku a N je počet integračních obrázků. (t_x, t_y) je vektor posunu ve vyhledávací oblasti a $i(x, y)$ je intenzita obrazových dat v bodě $[x, y]$. Obdobně se změní inicializace 3D integračního obrázku. Protože uložení počtu integračních obrázků $N = (2r+1)^3$ ve 3D je velice paměťově náročné, lze prohodit smyčky procházení vyhledávací oblasti a procházení konkrétních bodů v datech a spočítat tak integrační obrázek pro každý vektor posunu při filtrování. Je však nutné ukládat váhy a normalizační faktory do předem definovaného pole a na konec filtru přidat konečný průchod daty, který vážené průměry normalizuje a uloží do výstupních dat.

Při paralelním zpracování je nezbytné, aby každé vlákno mělo alokovaný vlastní prostor pro integrační obrázek z důvodu konfliktů při inicializaci, protože každé vlákno přistupuje jen do konkrétní oblasti integračního obrázku, lze inicializovat jenom tuto konkrétní oblast integračního obrázku a zmenšit tak časovou složitost inicializace.

Překračování okrajů obrazových dat

Při prohledávaní prostoru pro podobné patche a následném porovnávání patchů algoritmus non-local means filtru může překročit okraje až o vzdálenost $r+p$, kde r je polovina strany vyhledávací oblasti a p je polovina strany patche. Proto je nutné vycpat obrazová data tzv. paddingem, tzn. obalit data vrstvou, která bude obsahovat data nejbližšího okraje dat. Velikost vrstvy je stejná jako maximální přesah okraje ($k = r+p$, kde k je velikost paddingu). Tyto paddingy se využijí i pro integrační obrázky a ostatní pole ukládající stav filtrování.

Vzorkování váhové funkce

Non-local means filtr obsahuje poměrně komplexní váhovou funkci (viz rovnice 2.3), kterou lze zjednodušit na $w(d^2) = \exp(-\frac{d^2}{h^2})$ kde $d^2 = \|B(i) - B(j)\|_2^2$. Za předpokladu, že intenzita voxelu je reprezentována bajtovým celým číslem, lze předpokládat rozsah vstupního parametru $d^2 \in \langle 0; (2p+1)^3 \cdot 255^2 \rangle$, kde $(2p+1)^3$ je objem patche a 255^2 je maximální vzdálenost mezi 2 intenzitami. I přes fakt, že vzdálenost je celým číslem, předpočítání všech vstupů váhové funkce by bylo paměťově náročné, proto navrhoji tuto funkci navzorkovat a počet vzorků přidat mezi vstupní parametry.

Váhová funkce bude předpočítaná do 1D vyhledávací tabulky o velikosti N , kde vstupní parametr $N \leq (2p+1)^3 \cdot 255^2$ je počet vzorků. Vzorkování začíná na 0 a krok vzorku je $\frac{(2p+1)^3 \cdot 255^2}{N}$. Mapování vstupní vzdálenosti na index probíhá dle vzorce $d^2 \cdot \frac{N}{(2p+1)^3 \cdot 255^2}$, kde $\frac{N}{(2p+1)^3 \cdot 255^2}$ je předpočítatelný parametr.

Paměťové nároky a časová složitost 3D non-local means filtru

vstupní volumetrická data	$(W + 2k)(H + 2k)(D + 2k) \cdot S_{Byte}$
výstupní volumetrická data	$(W + 2k)(H + 2k)(D + 2k) \cdot S_{Byte}$
3D integrační obrázek	$T \cdot (W + 2k)(H + 2k)(D + 2k) \cdot S_{Long}$
vážené průměry bodů	$(W + 2k)(H + 2k)(D + 2k) \cdot S_{Double}$
normalizační faktory bodů	$(W + 2k)(H + 2k)(D + 2k) \cdot S_{Double}$
navzorkovaná váhová funkce	$N \cdot S_{Double}$

Tabulka 3.3: Paměťové nároky (paměť alokovaná na haldě) navrženého 3D non-local means filtru s integračním obrázkem. Parametr $k = r + p$ označuje padding kolem volumetrických dat, parametr N je počet vzorků a T je počet vláken použitých pro filtrování. Konstanta S_{Byte} reprezentuje velikost 8b celého čísla v paměti, S_{Double} reprezentuje velikost 64b desetinného čísla (double precision) a S_{Long} reprezentuje velikost 64b celého čísla.

Časová složitost navrženého 3D non-local means filtru s integračním obrázkem je zjednodušena z původních $\mathcal{O}(N \cdot (2r+1)^3 \cdot (2p+1)^3)$ na $\mathcal{O}((2r+1)^3 \cdot ((W+r)(H+r)(D+r) + WHD))$, kde $(W+r)(H+r)(D+r)$ představuje časovou složitost vytváření 3D integračního obrázku, časová složitost konečného průchodu daty je $\mathcal{O}(WHD)$.

Kapitola 4

Implementace nelineárních filtrů pro 3D obrazová data

Tato kapitola se zaměřuje na implementaci nelineárních filtrů dle modelů navržených v 3. kapitole. Nejprve je v sekci 4.1 vysvětlena struktura řešení a zvolené technologie, následně je v sekci 4.2 rozebrána implementace volumetrických dat. Kapitola pokračuje návrhem rozhraní filtrů pro paralelní zpracování (sekce 4.3) a nakonec jsou představeny implementace 3D bilaterálního filtru (sekce 4.4) a 3D non-local means filtru (sekce 4.5).

4.1 Vybrané technologie a struktura řešení

Pro programování jsem zvolil jazyk C# a platformu .NET (dotnet). Dotnet poskytuje multiplatformní řešení pro knihovny, konzolové aplikace ... Pro dotnet lze vyvíjet i poměrně rychlé aplikace (viz dále) a jazyk C# má kvalitní dokumentaci a velký ekosystém s vývojáři volně poskytujícími open source knihovny (tzv. NuGet balíčky). Pro vývoj lze také použít Visual Studio, které obsahuje profiler výhodný k analýze výkonu aplikace a využití prostředků počítače.

Řešení bylo vyvinuto pod .NET SDK 6.0 (software development kit) a pro spuštění je vyžadován .NET runtime (min. 6.0). Kromě grafické demo aplikace (GUI – graphical user interface) jsou všechny moduly multiplatformní včetně konzolové demo aplikace (CLI – command-line interface). Při vývoji byl využíván verzovací systém Git – veškeré zdrojové kódy včetně demo aplikací (GUI a CLI), testovacího skriptu atd. jsou dostupné online na mém osobním GitHubu¹.

Výkonné aplikace v .NET

Dotnet aplikace jsou optimalizovány ve dvou krocích:

1. při překladu zdrojového kódu do IL (Intermediate Language), který je nezávislý na platformě
2. při překladu IL na platformou specifikovanou sadu instrukcí tzv. JIT (Just-in-Time) kompilátorem

přičemž markantnější optimalizace provádí JIT.

¹<https://github.com/skrasekmichael/NonlinearFilters>

Dotnet aplikace běží s GC (garbage collector), který usnadňuje práci s pamětí a zajišťuje odstranění úniků paměti (memory leak), ale má negativní dopad na výkonnost aplikace. Díky GC je C# bezpečný pro práci s pamětí a nabízí rozhraní `Span<T>` a `Memory<T>` pro efektivní a bezpečný přístup do paměti. C# také nabízí možnost použití klíčového slova `unsafe` pro definování unsafe kontextu ve kterém lze přistupovat do paměti „nebezpečně“ pomocí odkazů (pointerů) a využívat pointerovou aritmetiku. Využitím pointerové aritmetiky lze také dosáhnout odstranění kontroly indexu pole (tzv. array bounds check) – i když JIT často dokáže odhadnout, kdy je nutné kontrolu indexu provádět, její odstranění může markantně zvýšit výkonnost.

Preferovaným způsobem paralelizmu v dotnetu je využití TBL (Task Parallel Library), jejíž koncept je založen na úlohách (tasks) a využívá rozhraní `Task`. Dokumentace [4] uvádí, že výhodou použití úloh je efektivnější a škálovatelnější využití systémových prostředků. I když dotnet nabízí přístup ke klasickému rozhraní vlákna (`Thread`), využitím vyšší abstrakce úloh přináší větší kontrolou nad asynchronním programováním a bohatší API (application programming interface) pro paralelní programování.

Dotnet umožňuje paralelní zpracování celých registrů pomocí datových typů zrychlených SIMD instrukcemi, tyto typy lze nalézt pod jemným prostorem `System.Numerics`, kde se nachází datové type pro vektory, matice, roviny ...

Struktura repositáře

- README.md – soubor se základními informacemi o projektu a výsledky měření filtrů v porovnání s referenčními filtry.
- Sources – veškeré zdrojové kódy rozděleny do modulů (v dotnetu tzv. projekty).
 - NonlinearFilters – knihovna s implementacemi 2D a 3D filtrů (bilaterální filtr a non-local means filtr) s možností paralelního zpracování.
 - NonlinearFilters.Volume – modul s rozhraním volumetrických dat a logikou pro načítání a ukládání dat v různých formátech.
 - NonlinearFilters.APP – projekt demonstrující využití knihovny s filtry v moderní grafické aplikaci (WPF framework).
 - NonlinearFilters.APP.VolumeRender – modul pro jednoduchý 3D rendering² volumetrických dat pomocí NuGet balíčku OpenTK³.
 - NonlinearFilters.CLI – projekt demonstrující využití knihovny s filtry v konzolové aplikaci s CLI pro spouštění filtrů.
- Data – obsahuje příklady zašuměných vstupních 2D obrazových a volumetrických dat.
- Images – složka s výstupními obrázky generovanými testovacím souborem, které jsou využity v README souboru.
- Python – python skripty umožňující spouštění referenčních filtrů.
- test.ps1 – testovací PowerShell skript, který testuje filtry pomocí CLI aplikace a porovnává výsledky a dobu zpracování s referenčními filtry.
- ostatní soubory a složky – Makefile, .gitignore ...

²raycasting impl. dle <https://www.willusher.io/webgl/2019/01/13/volume-rendering-with-webgl>

³OpenGL rozhraní pro C#

4.2 Implementace volumetrických dat

Volumetrická data jsou reprezentovaná třídou `VolumetricData` implementující parametry dat (rozměry, skutečná velikost ...) viz sekce 2.1) a trojrozměrné pole voxelů. Předpokládaným vstupem filtrů jsou šedotónová volumetrická data s intenzitou voxelu reprezentovanou na 8 bitech, tzn. pole voxelů je datového typu `byte[]` a index voxelu na pozici $[x, y, z]$ je vypočítán jako:

$$index = x \cdot H \cdot D + y \cdot D + z \quad (4.1)$$

kde H je výška dat a D je hloubka dat. Vzorec 4.1 (a filtry pracující s volumetrickými daty) předpokládá data uložená v paměti v pořadí x, y, z , kdy souřadnice z se mění nejrychleji a x nejpomaleji. Pokud jsou data serializovaná v pořadí z, y, x , je potřeba při čtení/zápisu dat u parametru velikosti (případně u dalších parametrů) prohodit rozměr hloubky s šírkou, o to se stará třída obsluhující soubory, kde každý formát volumetrických dat (NRRD ...) má dedikovanou třídu dědící z abstraktní třídy `BaseFileHandler`. Kromě prohození pořadí serializace můžou „FileHandlery“ formátů při čtení/zápisu data překonvertovat (př. z floatů na bajty), data komprimovat (gzip ...) a obsloužit další parametry formátů volumetrických dat (skutečná velikost). I když filtry nevyužívají parametr skutečné velikosti volumetrických dat, může tento parametr být využit při interpretaci (rendering ...).

4.3 Implementace rozhraní filtru

Filtr má 3 výpočetně náročné části:

- Inicializace filtru – předpočítá a inicializuje parametry nezávislé na parametrech filtrování, ale závislé na vstupní datech.
- Předpočítání parametrů – předpočítá parametry nezávislé na vstupních datech, ale závislé na parametrech filtrování.
- Filtrování – paralelně vyfiltruje vstupní data.

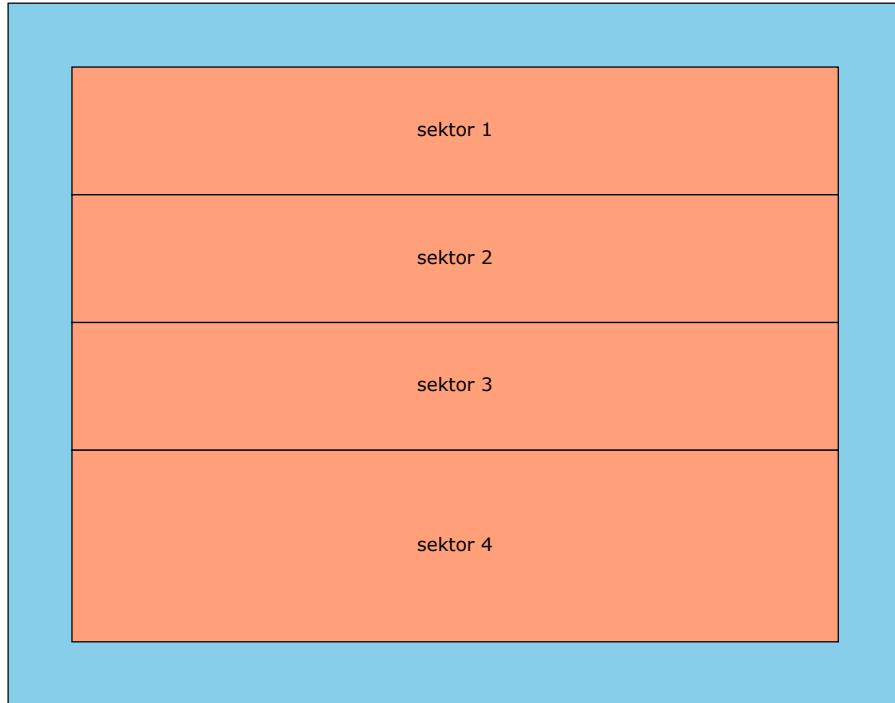
Všechny filtry dědí z generické abstraktní třídy `BaseFilter<T>`, která rozhraní těchto částí definuje, generický parametr T je datový typ parametrů filtru. Rozhraní 2D a 3D filtrů se mírně liší kvůli vstupním datům, 2D filtry dědí z `BaseFilter2<T>` a pracují s 2D obrazovými daty pomocí multiplatformního API poskytnuté NuGet balíčkem `ImageSharp`. 3D filtry dědí z `BaseFilter3<T>` a pracují s volumetrickými daty definovanými v sekci 4.2.

Kroky filtrování

1. Inicializuje se filtr, pokud dosud nebyl inicializován.
2. Obalí se vstupní data paddingem a vytvoří se nová výstupní data s paddingem.
3. Předpočítají se parametry, pokud dosud nebyly předpočítány nebo pokud se vstupní parametry filtru změnily.
4. Rozdělí se vstupní data na sektory pro paralelní zpracování.
5. Vytvoří a spustí se úlohy pro paralelní filtraci, každé úloze se přidělí sektor v datech pro filtrování. Úlohy jsou vytvořeny s možností `TaskCreationOptions.LongRunning`, která umožní lepší plánování vlákna.
6. Po dokončení všech úloh se následně odstraní padding z vyfilrovaných dat.

Rozdělení dat

Při paralelním filtrování je potřeba výstupní data rozdělit na neprotínající se sektory, aby při zápisu dat nedocházelo ke konfliktu. Kvůli zachování principu lokality (viz sekce 3.1) je výhodné data dělit podél nejpomaleji měnící se souřadnice – podél osy y pro 2D obrazová data a podél osy x pro volumetrická data. Protože rozměr dat nelze vždycky celočíselně rozdělit, poslední sektor může být větší než ostatní sektory (viz obrázek 4.1)



Obrázek 4.1: Ilustrace rozdělení 2D obrazových dat na sektory pro paralelní zpracování na 4 vláknech. Modrá barva označuje padding dat, který není potřeba filtrovat, červená barva označuje data rozdělená na sektory, kde každý sektor je přidělen úloze (vláknu).

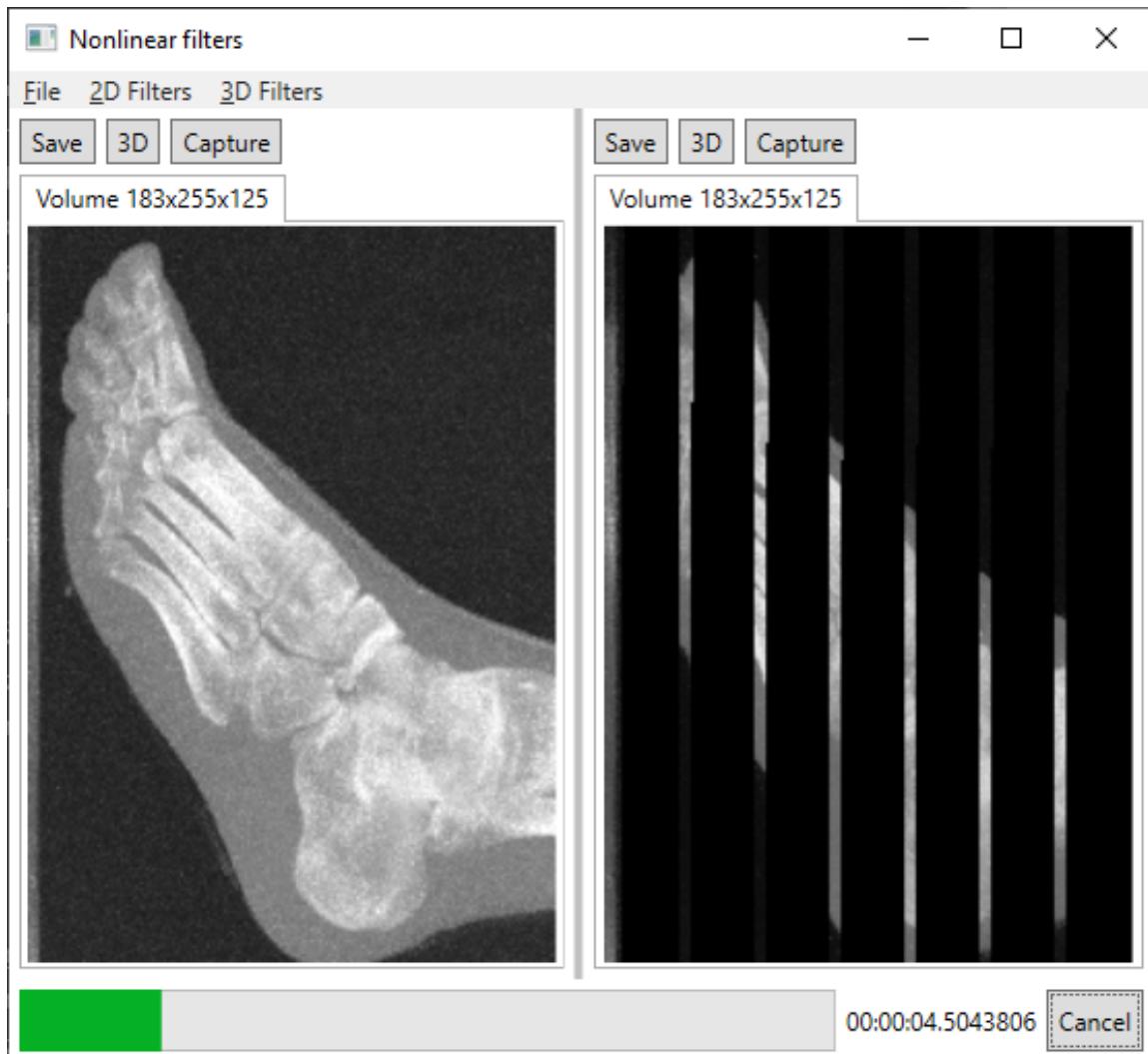
Asynchronní filtr

Při filtrování v okenních aplikacích se předpokládá nezamrznutí okna, tzn. použití asynchronního filtru, a zobrazení pokroku filtrování a případnou možnost zrušit filtrování (viz obrázek 4.2). Aby se předešlo konfliktům mezi vlákny, každé vlákno si ukládá svůj postup (počet již vyfiltrovaných bodů) do pole na pozici odpovídající indexu sektoru/vlákna, filtr poté občas vyvolá událost `OnProgressChanged` s parametrem udávajícím procento vyfiltrovaných dat. Protože malá nepřesnost v postupu je bezvýznamná, lze celkový postup vypočítat bez použití zámků jako:

$$progress = \frac{1}{N} \cdot \sum_{i=0}^T P[i]$$

kde N je počet bodů v obraze (WH pro 2D, WHD pro 3D), T je počet vláken, na kterých je filtr spuštěn a P je pole pro ukládání postupu jednotlivých vláken.

Aby šlo filtrování kdykoli zrušit, musí každý filtr při filtraci kontrolovat stav vnitřní proměnné `IsCanceled`, která udává, zda bylo filtrování zrušeno. Filtr poté ukončí činnost filtrování a provede konečné kroky pro udržení konzistence dat.



Obrázek 4.2: Ukázka z demonstrační aplikace zobrazující funkci zrušení filtrace a zobrazení postupu filtrace. Obrázek vlevo jsou vstupní zašuměná volumetrická data. Obrázek vpravo ukazuje výsledek filtrace (bilaterální filtr) zrušené po 4,5 vteřinách, z obrázku je patrné paralelní zpracování – 7 již vyfiltrovaných oblastí. Pod obrázky je `ProgressBar` zobrazující postup filtrování.

4.4 Implementace 3D bilaterálního filtru

Vstupními parametry filtrace modelem 3D bilaterálního filtru (viz sekce 3.2) jsou σ_r , σ_s řídící váhové funkce a parametr r , který udává poloměr koule reprezentující relevantní váhy prostorové Gaussovy funkce (viz obrázek 3.1). Pokud parametr r není zadán, je parametr spočítán jako $r = 2,5\sigma_s$. Navržený model 3D bilaterálního filtru nepoužívá padding dat, ale předpočítává hranice dat, tyto hranice lze jednoduše spočítat při inicializaci filtru:

```
border = new int[r + size + r];
for (int i = r; i < border.Length; i++) {
    border[i] = min(i - r, size - 1);
}
```

kde `border` je vyhledávací tabulka hraniční funkce, `r` je vstupní parametr filtru a `size` je rozměr dat ve směru právě předpočítávané osy (tzn. `width/height/depth`).

Předpočítání parametrů

Při předpočítání parametrů 3D bilaterálního filtru se předpočítávají Gaussovy váhové funkce a funkce definující okraj vyhledávací oblasti, tj. oblast definující relevantní hodnoty v 3D Gaussově funkci (neboli posun od centrálního bodu). Předpočítání je implementováno dle rovnic v sekci 3.3.

Algoritmus filtrování

Funkce obstarávající filtraci je volána pro každé vlákno a musí projít všechny voxely v datech, které jsou danému vláknu přiděleny. Aby filtrace mohla být korektně zrušena, musí filtr průběžně kontrolovat, zda se nezměnila proměnná `IsCanceled`. Pro zobrazení postupu filtrace si každé vlákno drží informaci o počtu již vyfiltrovaných bodů. Pro rychlejší zpracování je událost postupu filtru a kontrola zrušení filtrace umístěna ve 2. úrovni smyčky filtru:

```
for (int x = block.X; x < block.EndX; x++) {
    for (int y = block.Y; y < block.EndY; y++) {
        for (int z = block.Z; z < block.EndZ z++) {
            ...
            setInsensity(x, y, z, newIntensity);
            progress[threadIndex]++;
        }
        if (IsCanceled) return;
        updateProgress();
    }
}
```

kde `block` je struktura reprezentující sektor dat (kvádr) přidělený vláknu pro filtrace. Proměnná `newIntensity` představuje novou intenzitu spočítanou z podobných patchů. Proměnná `threadIndex` udává index vlákna a pole `progress` ukládá počty již vyfiltrovaných bodů jednotlivých vláken. Funkce `updateProgress` vyvolává událost změny postupu. Oblast označená třemi tečkami je vnitřní smyčka algoritmu procházející relevantní body vyhledávací oblasti, přes kterou se počítá vážený průměr a výsledná nová intenzita voxelu.

4.5 Implementace 3D non-local means filtru

Vstupními parametry filtrace modelem 3D non-local means filtru (viz sekce 3.3) jsou parametry r a p , které definují velikost vyhledávací oblasti a velikost patche. Vstupní parametr h pak řídí váhovou funkci. Čtvrtým parametrem je hodnota udávající počet vzorků, na který se má váhová funkce předpočítat, pokud je počet vzorků záporný, váhová funkce se nevzorkuje. Model non-local means filtru do dat přidává padding o velikosti $r + p$.

Předpočítání parametrů

Non-local means filtr při předpočítání parametrů vzorkuje váhovou funkci:

```
long maxSamples = patchSize * 255 * 255;
sampledWeightingFunction = new double[samples];

double val = 0;
double step = (double)maxSamples / samples;
for (int i = 0; i < samples; i++) {
    sampledWeightingFunction[i] = exp(-val / (h * h * patchSize));
    val += step;
}
```

kde `patchSize` je objem patche spočítaný jako $(p + 1)^3$, `samples` je počet vzorků a `h` je vstupní parametr řídící váhovou funkci. Proměnná `sampledWeightingFunction` je vyhledávací tabulka hodnot váhové funkce. Konkrétní hodnotu váhové funkce lze vypočítat jako `sampledWeightingFunction[distance / step]`, kde `distance` je euklidovská vzdálenost patchů na druhou (viz vzorkování váhové funkce v sekci 3.3). Filtr dále potřebuje připravit globální (stejné pro všechna vlákna) pomocné proměnné pro vážený součet intenzit a normalizační faktor (součet vah), které jsou použity pro výpočet nové intenzity ve finální smyčce.

Algoritmus filtrování

Funkce obstarávající filtraci je volána pro každé vlákno a musí projít všechny voxely v datech, které jsou danému vláknu přiděleny. Protože model 3D non-local means filtru využívá optimalizaci integračním obrázkem, jsou prohozeny smyčky procházející sektory přidělený vláknu a smyčky procházející oblast prohledávající podobné patche, proto se na konci filtrace musí spočítat výsledné intenzity voxelů z vážených součtů intenzit a normalizačních faktorů:

```
for (int x = block.X; x < block.EndX; x++) {
    for (int y = block.Y; y < block.EndY; y++) {
        for (int z = block.Z; z < block.EndZ; z++) {
            double newIntensity = weightedSum[x, y, z] / normFactor[x, y, z];
            setIntensity(x, y, z, (byte)newIntensity);
        }
    }
}
```

kde `block` je struktura reprezentující sektor dat (kvádr) přidělený vláknu pro filtraci, `weightedSum` a `normFactor` jsou globální pomocné proměnné.

Při zrušení filtrace proměnnou `IsCanceled` se v hlavní smyčce musí zavolat finalizace, aby byl ve vrácených datech zobrazen výsledek filtrování. Pro udržení rychlosti filtru je kontrola zrušení filtrace a událost postupu filtrace umístěna v 3. úrovni cyklu a ukládání informace o postupu v 5. úrovni cyklu:

```

for (int wx = -r; wx <= r; wx++) {
    for (int wy = -r; wy <= r; wy++) {
        for (int wz = -r; wz <= r; wz++) {
            calculateIntegralImage(block);

            for (int x = block.X; x < block.EndX; x++) {
                for (int y = block.Y; y < block.EndY; y++) {
                    for (int z = block.Z; z < block.EndZ; z++) {
                        ...
                    }
                    progress[threadIndex] += progressStep;
                }
            }

            if (IsCanceled) {
                finalization();
                return;
            }
            updateProgress();
        }
    }
}

finalization();

```

kde `r` je vstupní parameter filtru r , funkce `calculateIntegralImage` vypočítá integrační obrázek pro sektor přidělený vláknu a `block` je struktura reprezentující tento sektor. Proměnná `threadIndex` udává index vlákna provádějící filtrace a `progress` je pole ukládající počty již vyfiltrovaných voxelů jednotlivých vláken. Proměnná `progressStep` je krok, který udává o kolik se zvětšíl počet vyfiltrovaných bodů. Oblast označená třemi tečkami počítá váhy pomocí integračního obrázku a ukládá je do pomocných proměnných použitých při finalizaci. Funkce `finalization` provádí finalizaci, tj. spočítání výsledných intenzit voxelů (viz pseudokód výše), a funkce `updateProgress` označuje událost postupu filtrování.

Prohozením smyček vzniká problém s udržováním informace o postupu filtrace a postup nelze ukládat v celých číslech, protože ke kompletnímu vyfiltrování voxelu dochází až při finalizaci. Proto je krok postup (`progressStep`) desetinné číslo vypočítané jako:

```
progressStep = block.Depth / pow(2r + 1, 3);
```

Kapitola 5

Výsledky

Tato kapitola je zaměřená na výsledky filtrů implementovaných v kapitole 4. Nejdříve jsou v kapitole představeny referenční filtry (sekce 5.1) a vstupní volumetrická data použitá při testování (sekce 5.2), kapitola dále pokračuje průběhem testování a popisem testovacího prostředí (sekce 5.3). Kapitola je zakončena porovnáním rychlostí a výsledků 3D bilaterálního filtru (sekce 5.4) a 3D non-local means filtru (sekce 5.5) s referenčními filtry.

5.1 Referenční filtry

Při výběru referenčních filtrů pro porovnání výsledků bylo vybíráno z běžně dostupných implementací 3D bilaterálního filtru a 3D non-local means filtru.

Jako referenční 3D bilaterální filtr byla zvolena implementace ze sady nástrojů Itk¹ (The Insight Toolkit). Itk implementace 3D bilaterálního filtru plně využívá paralelního zpracování a je používaná třeba programem 3D Slicer². Pro spouštění referenčního bilaterálního filtru je použit Itk balíček pro Python (viz sekce 5.3).

Pro non-local means filtr byla vybrána implementace z knihovny scikit-image³. Scikit-image implementace 3D non-local means filtru používá algoritmus optimalizovaný integračním obrázkem (tzn. nezávislost na velikosti patche) a nevyužívá paralelního zpracování. Pro spouštění referenčního non-local means filtru je použit scikit-image balíček pro Python (viz sekce 5.3).

5.2 Vstupní data

Jako vstupní data pro testování a porovnání výsledku filtrace jsou použita volumetrická data o rozměrech $183 \times 255 \times 125$ znázorňující výpočetní tomografii chodidla (viz obrázek 5.1). Pro sledování efektu filtrace šumu je do volumetrických dat uměle nagerenerovaný bílý Gaussův šum (ADWR – additive white Gaussian noise), který bude testovanými filtry odstraňován.

¹<https://itk.org/>

²<https://www.slicer.org/>

³<https://scikit-image.org/>



původní volumetrická data (CT chodidla)

volumetrická data s přidaným šumem

Obrázek 5.1: Porovnání původních volumetrických dat (výpočetní tomografie chodidla) a volumetrická data s přidaným Gaussovým šumem.

Pro testování a porovnání rychlostí filtrů s referenčními filtry jsou připravena umělá volumetrická data s nagenerovaným bílým Gaussovým šumem (ADWR) o rozměrech:

- $50 \times 50 \times 50 = 125\,000$ voxelů
- $150 \times 150 \times 150 = 3\,375\,000$ voxelů
- $300 \times 300 \times 300 = 27\,000\,000$ voxelů

Pro srovnání: 2D obrazová data s FullHD rozlišením (1920×1080) mají 345 600 pixelů a data s 4K rozlišením (3840×2160) mají 8 294 400 pixelů.

5.3 Průběh testování a testovací prostředí

Pro každý filtr (bilaterální filtr a non-local means filtr) jsou provedeny 3 testy:

1. Test odfiltrování šumu a porovnání výsledku s referenčním filtrem.
2. Otestování časové závislosti filtru na počtu použitých vláken pro filtraci.
3. Porovnání rychlosti filtru s referenčním filtrem na vícero velikostech volumetrických dat.

kde testování probíhá na prostředí s parametry:

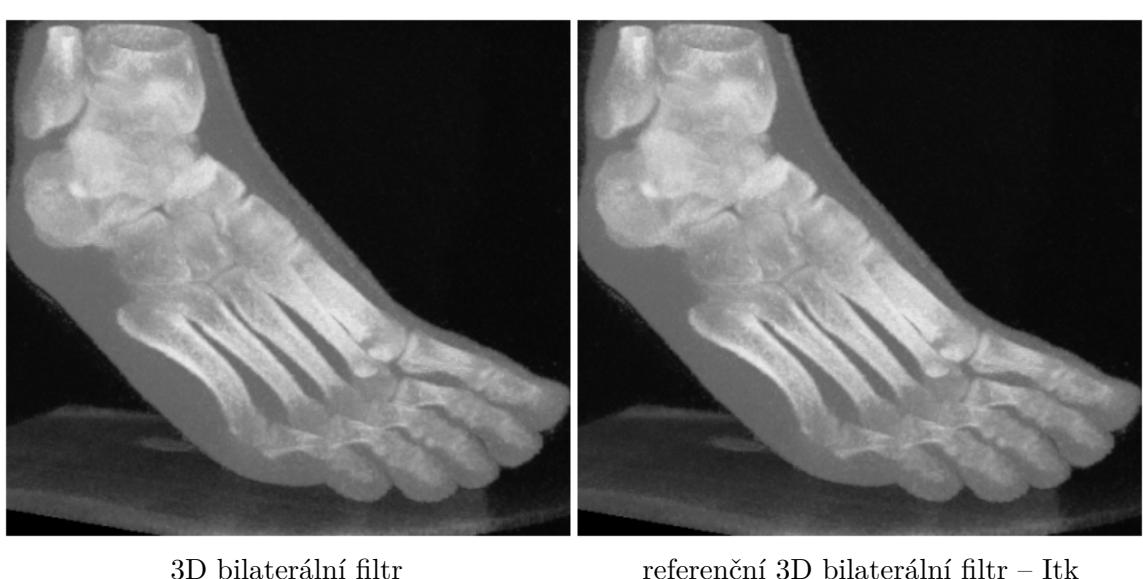
- operační systém: 64bitový Windows 10 (build 19044)
- procesor: Intel(R) Core(TM) i7-7700HQ 2,80GHz (jádra 4/8)
- .NET runtime: 6.0.3
- python interpret: 3.10.2 [MSC v.1929 64 bit (AMD64)]
- verze balíčku Itk: 5.3.0
- verze balíčku scikit-image: 0.19.2

5.4 Bilaterální filtr pro volumetrická data

Filtrační schopnosti implementovaného 3D bilaterálního filtru lze vidět na obrázku 5.2, kde výsledná data po aplikaci filtru mají zřetelně méně šumu, kvůli strategii bilaterálního filtru je kolem hran vidět menší neodstraněný šum. Na obrázku 5.3 je porovnání výstupů bilaterálního filtru a referenčního filtru (Itk). Mezi výslednými daty je mírný rozdíl, který není postřehnutelný okem, tyto rozdíly jsou pravděpodobně způsobeny zmenšením vyhledávací oblasti (viz předpočítání vyhledávací oblasti v sekci 3.2).

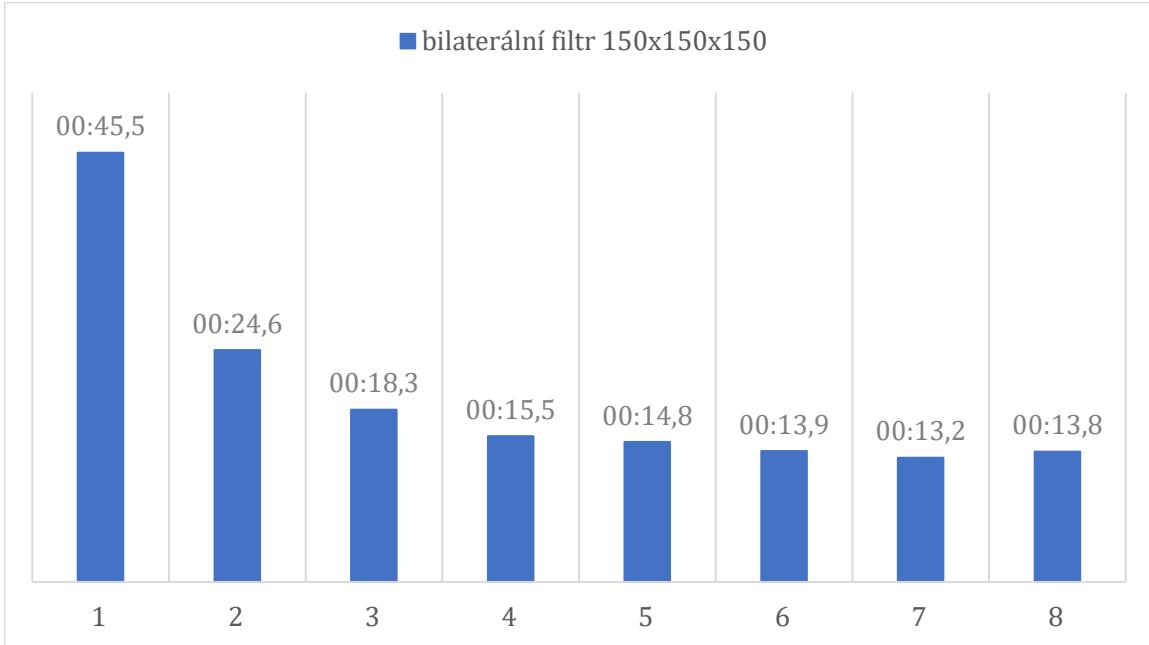


Obrázek 5.2: Porovnání vstupních zašuměných dat a výsledku po aplikaci 3D bilaterálního filtru s parametry $\sigma_s = 5$ a $\sigma_r = 20$.



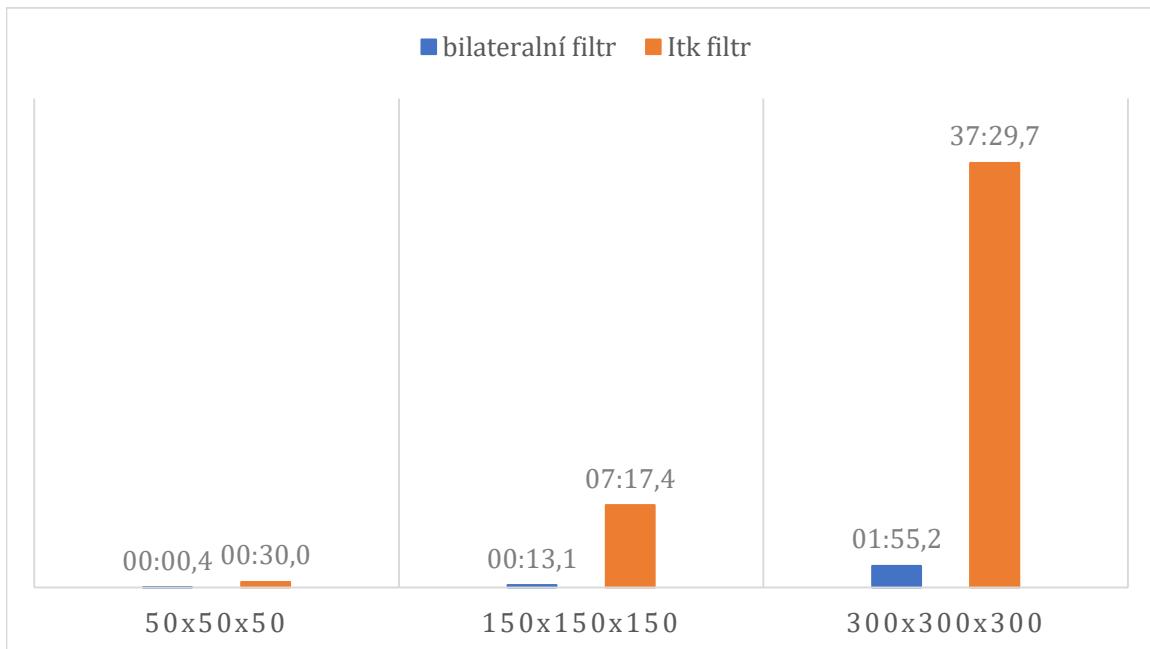
Obrázek 5.3: Grafické porovnání výstupů bilaterálního filtru a referenčního filtru (Itk) po provedení filtrace se stejnými vstupními parametry: $\sigma_s = 5$ a $\sigma_r = 20$.

Na obrázku 5.4 je graf zobrazující časovou závislost implementovaného bilaterálního filtru na počtu použitých vláken při paralelném filtrování. Z grafu je patrný exponenciální pokles efektu počtu vláken na rychlosť filtrování a po 4 vláknech má přidání dalších vláken pro filtraci minimální účinek. Nejrychlejšího zpracování dosáhl filtr při filtrování na 7 vláken a při přidání dalšího vlákna (filtrace na 8 vláknech) dosáhl mírně pomalejší výsledku, proto je při porovnání rychlostí bilaterálního filtru s referenčním filtrem použita varianta s paralelním zpracováním na 7 vláknech.



Obrázek 5.4: Graf zobrazující časovou závislost 3D bilaterálního filtru na počtu vláken při filtrování volumetrických dat o velikosti $150 \times 150 \times 150$ a vstupním parametrem $r = 12$.

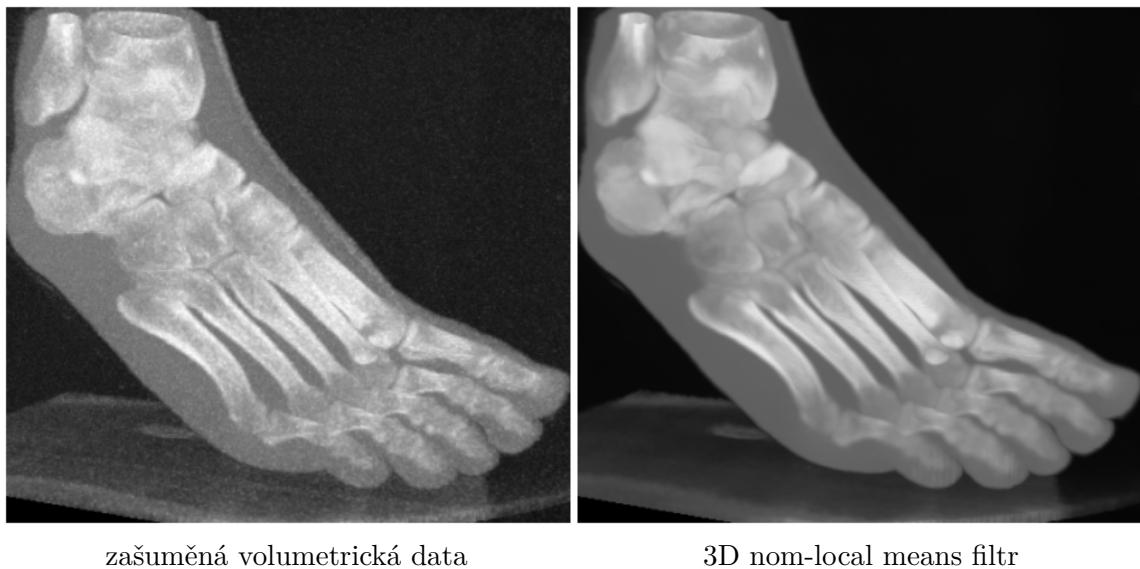
Graf na obrázku 5.5 zobrazuje porovnání rychlostí implementovaného 3D bilaterálního filtru a referenčního bilaterálního filtru (Itk). V grafu jsou vidět velké rozdíly v rychlosti filtrů, již při nejmenší velikosti vstupních dat ($50 \times 50 \times 50$) trvala filtrace referenčnímu filtru 30 vteřin, což je $75 \times$ více než testovanému bilaterálnímu filtru, který stejná data vyfiltroval přibližně za 400 ms. Při filtraci objemnějších dat ($300 \times 300 \times 300$) se implementovaný filtr drží v řádech jednotek minut, zatímco filtrace referenčního filtru trvala 37 minut.



Obrázek 5.5: Graf porovnávající rychlosť 3D bilaterálneho filtro (7 vláken) a referenčného filtro (Itk) na rôznych velikostech volumetrických dat se vstupným parametrom $r = 12$.

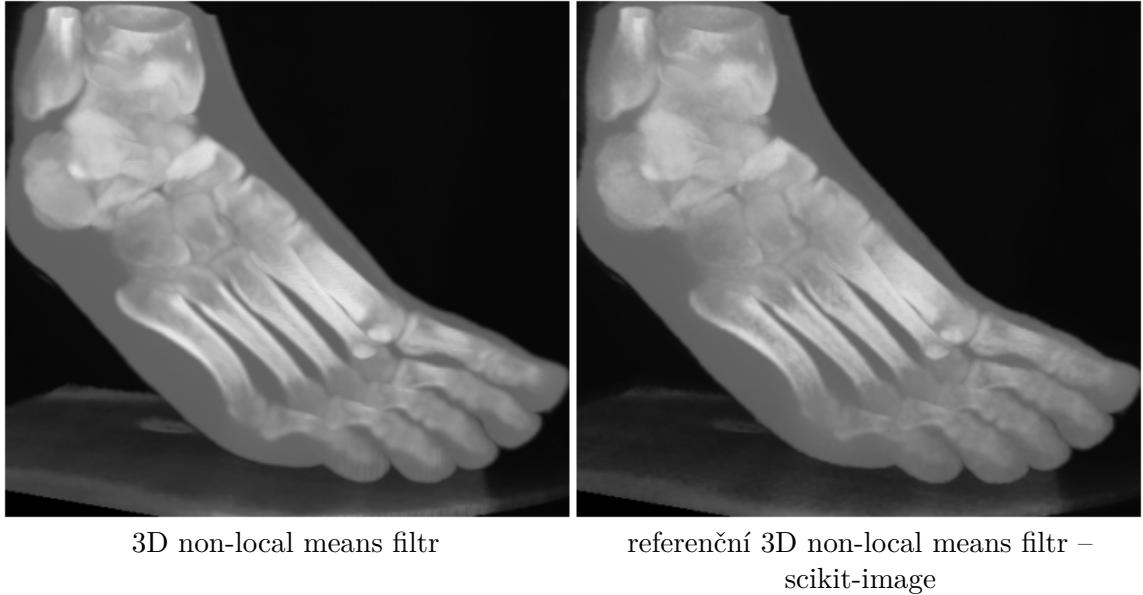
5.5 Non-local means filtr pro volumetrická data

Filtračné schopnosti implementovaného 3D non-local means filtro lze vidieť na obrázku 5.6, kde výsledná volumetrická data po aplikaci filtro nemají téměř žádný šum. Oproti výsledku bilaterálneho filtro (obrázek 5.2) non-local means filtr odstranil šum i kolem hran.

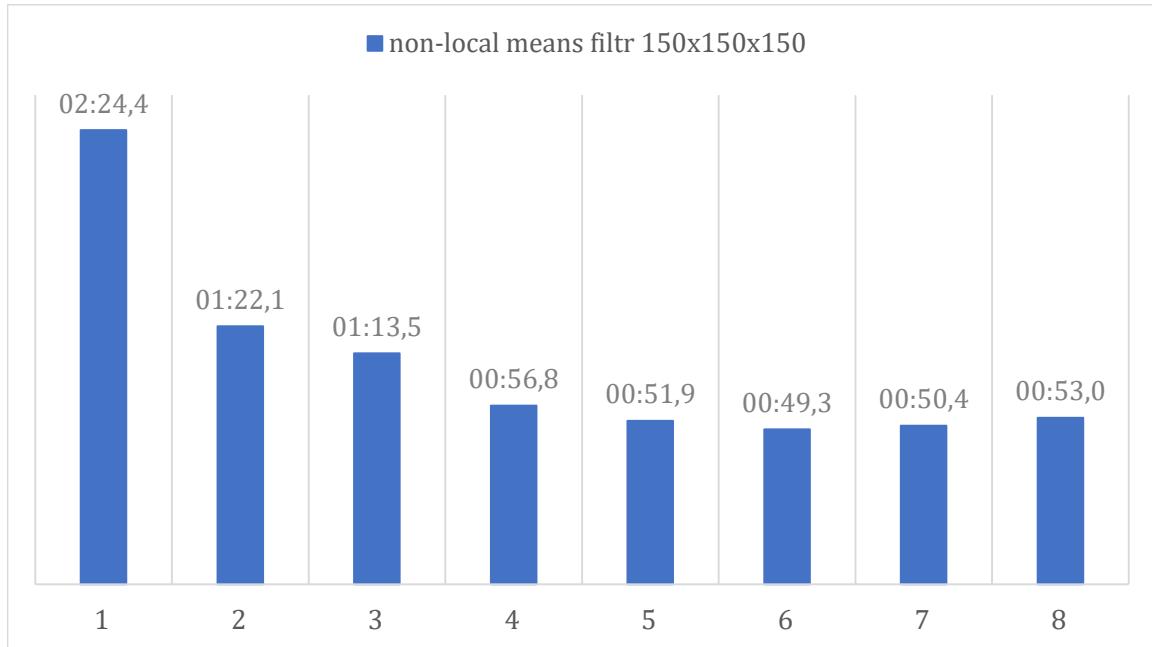


Obrázek 5.6: Porovnání vstupních zašuměných dat a výsledku po aplikaci 3D non-local means filtro s navzorkovanou váhovou funkcí (500 vzorků) a vstupními parametry $p = 1$, $r = 7$ a $h = 20$.

Na obrázku 5.7 je vidět porovnání výstupů implementovaného 3D non-local means filtru a referenčního filtru (scikit-image). Odlišnosti ve výstupech jsou postřehnutelné okem. Protože implementace referenčního filtru také využívá optimalizaci integračním obrázkem, jsou rozdíly pravděpodobně způsobeny jinými optimalizacemi či chybami v implementaci.



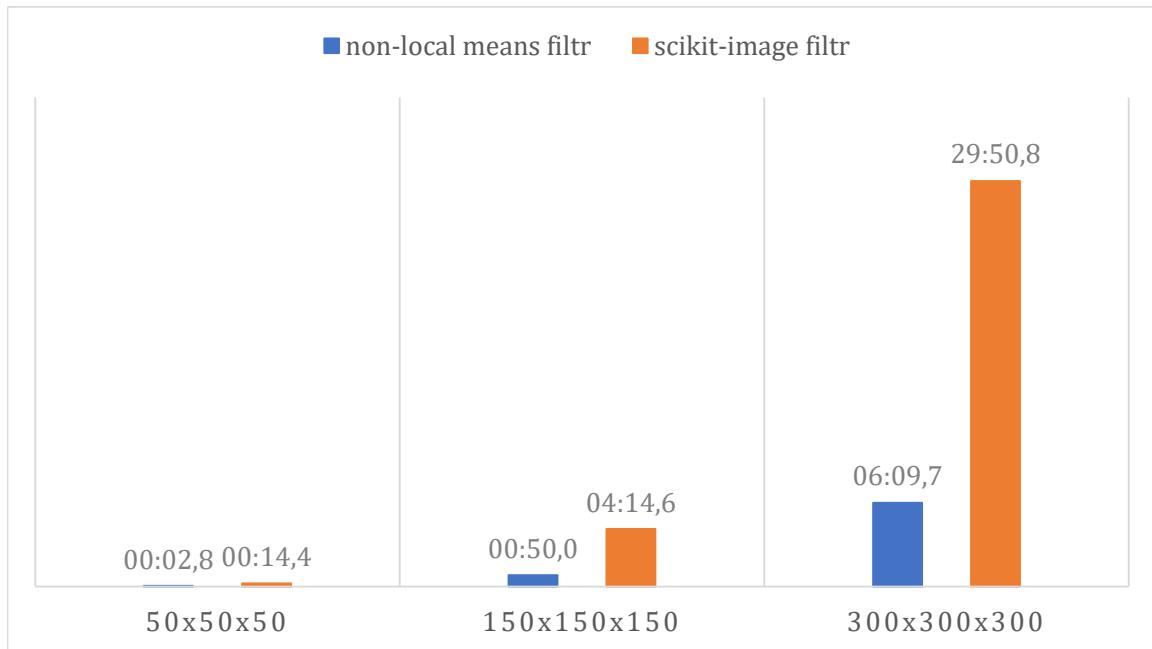
Obrázek 5.7: Grafické porovnání výstupů non-local means filtru (500 vzorků) a referenčního filtru (scikit-image) po filtrace se stejnými vstupními parametry: $p = 1$, $r = 7$ a $h = 20$.



Obrázek 5.8: Graf zobrazující časovou závislost 3D nom-local means filtru na počtu vláken při filtrování volumetrických dat o velikosti $150 \times 150 \times 150$ a vstupních parametrech $r = 7$ a $p = 1$.

Graf z obrázku 5.8 zobrazuje časovou závislost implementovaného non-local means filtru na počtu použitých vláken při paralelném filtrování. Z grafu je patrný exponenciální pokles efektu počtu vláken na rychlosť filtrovania. Problém horší paralelní zpracovateľnosťi non-local means filtru s integračným obrázkem se podepsal v grafu, když počet použitých vláken začal zhoršovať rychlosť už od 7 vláken (tj. o 1 vlákno dôvod než bilaterálny filtr). Pro udrženie konzistentnosti s testy bilaterálneho filtra je testovanie rychlosťi non-local means filtru provádzané na 7 vláknoch.

Na obrázku 5.9 je graf, ktorý porovnáva rychlosť filtrace implementovaného 3D non-local means filtra a referenčného non-local means filtra z knihovny scikit-image. Z grafu je patrné, že implementovaný non-local means filtr je markantne rychlejší než referenčný filtr. Protože implementácia referenčného filtra nepodporuje paralelnú filtrace, porovnanie s paralelne pracujúcim filtrom není adekvátné, ale i pri porovnaní neparalelného zpracovania volumetrických dat o veľkosti $150 \times 150 \times 150$ je implementovaný filtr (02:24,4) 1,7× rychlejší než referenčný filtr (04:14,6).



Obrázek 5.9: Graf porovnávající rychlosť 3D non-local means filtra (7 vláken, 500 vzorkov) a referenčného non-local means filtra (scikit-image) na rôznych veľkostach volumetrických dat se vstupnými parametrami $r = 7$ a $p = 1$.

Kapitola 6

Závěr

Cílem této práce bylo načerpat znalosti o zpracovávání obrazu, nelineárních filtroch a optimalizačích používaných při filtrování. Následně implementovat zvolené nelineární filtry – bilaterální filtr a non-local means filtr – pro 3D obrazová data.

Rychlosť a kvalita výsledků implementovaných filtrov ukazují na poměrně kvalitní implementaci filtrov. Implementované filtry jsou několikanásobně rychlejší než referenční filtry s malou nebo téměř žádnou odchylkou výsledků. Využití jazyka C# pravděpodobně nezhoršilo rychlosť filtrov, ale naopak napomohlo vytvoření multiplatformní knihovny implementující tyto filtry. Součástí řešení je také ukázková okenní aplikace a konzolová (CLI) aplikace, které demonstrují využití této knihovny.

S vyšší časovou dotací či lepším rozložením práce by šlo filtry dále optimalizovat použitím SIMD operací či zpracováváním na grafické kartě. Nadstavbou projektu pro další vývoj by mohla být úprava filtrov pro filtraci velkých volumetrických dat, což jsou data s velkými rozměry, případně data, která se celá nevejdou do operační paměti. Nepřímým rozšířením projektu by také mohla být reimplementace filtrov do známých open source knihoven či aplikací pracujících s volumetrickými daty, případně vývoj aplikace, která tyto filtry sama využije.

Literatura

- [1] BUADES, A., COLL, B. a MOREL, J.-M. A non-local algorithm for image denoising. In:. Červenec 2005, sv. 2, s. 60– 65 vol. 2 [cit. 12.4. 2022]. DOI: 10.1109/CVPR.2005.38. ISBN 0-7695-2372-2. Dostupné z: <https://www.researchgate.net/publication/4156453>.
- [2] DURAND, F. a DORSEY, J. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. In: *Proc. 29th Annu. Conf. Comput. Graph. Interact. Techn. SIGGRAPH*. Srpen 2002, s. 257–266 [cit. 5.4. 2022]. Dostupné z: <http://people.csail.mit.edu/fredo/PUBLI/Siggraph2002/DurandBilateral.pdf>.
- [3] FROMENT, J. Parameter-Free Fast Pixelwise Non-Local Means Denoising. *Image Processing On Line*. 2014, sv. 4, s. 300–326, [cit. 13.4. 2022]. Dostupné z: <https://doi.org/10.5201/itol.2014.120>.
- [4] MICROSOFT. *Task-based asynchronous programming* [online]. 2022 [cit. 2.5. 2022]. Dostupné z: <https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming>.
- [5] WIKIPEDIA CONTRIBUTORS. *Summed-area table* [online]. Prosinec 2021 [cit. 8.4. 2022]. Dostupné z: https://en.wikipedia.org/wiki/Summed-area_table.
- [6] WIKIPEDIA CONTRIBUTORS. *68–95–99.7 rule* [online]. Duben 2022 [cit. 6.4. 2022]. Dostupné z: https://en.wikipedia.org/wiki/68%25E2%2580%259395%25E2%2580%259399.7_rule.

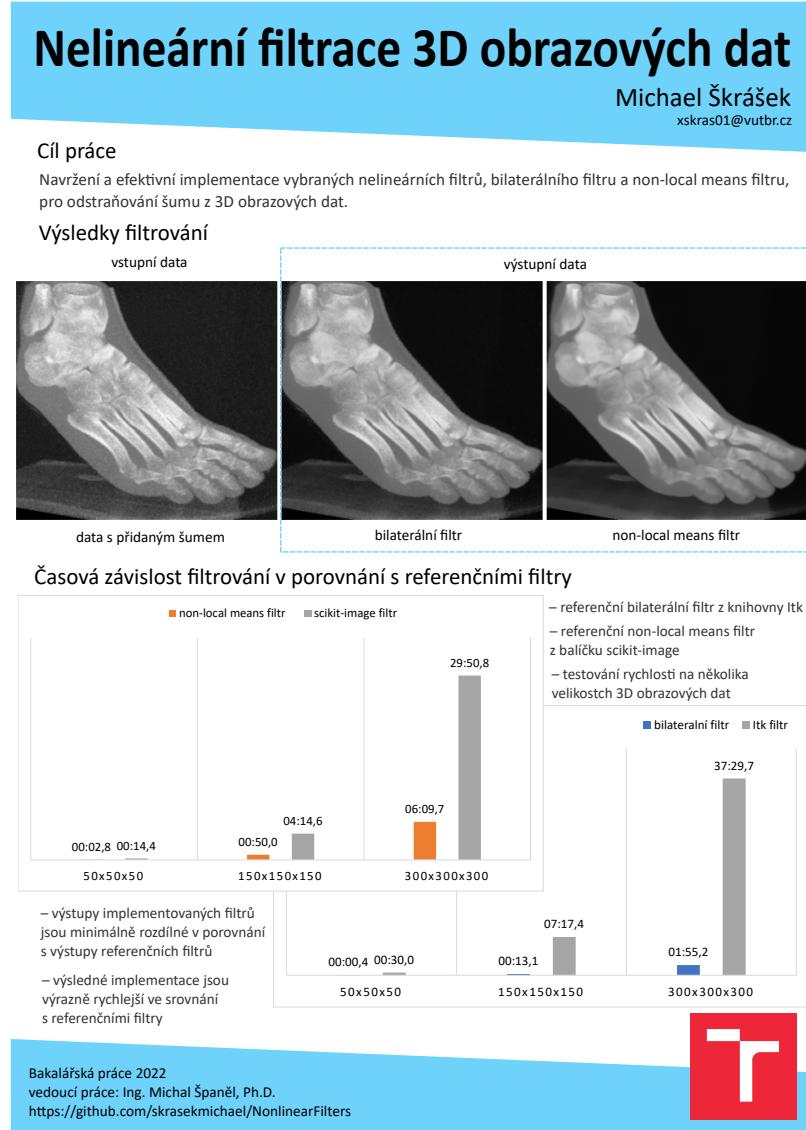
Příloha A

Obsah přiloženého paměťového média

```
/  
├── .git  
├── .gitignore  
├── Makefile  
├── README.md ..... informace o projektu  
├── README.txt ..... návod na spuštění demo aplikací  
├── test.ps1 ..... testovací PowerShell skript  
├── Data ..... složka se vzorovými daty  
├── Demo  
│   ├── CLI ..... konzolová demo aplikace (CLI)  
│   └── GUI ..... grafická demo aplikace (Windows)  
├── Images ..... složka s obrázky využitými v README.md  
├── Python ..... složka s python skripty referenčních filtrů  
├── Sources ..... složka se zdrojovými kódami  
│   ├── NonlinearFilters.sln  
│   ├── NonlinearFilters  
│   ├── NonlinearFilters.APP  
│   ├── NonlinearFilters.APP.VolumeRenderer  
│   ├── NonlinearFilters.CLI  
│   └── NonlinearFilters.Volume  
└── Thesis ..... složka s textem práce napsaném v LATEXu  
    ├── project.pdf ..... PDF s textem bakalářské práce  
    └── poster.pdf ..... plakát ve formátu A3  
    ...
```

Příloha B

Plakát



Obrázek B.1: Ilustrace plakátu.