VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE ŠÍŘENÍ ULTRAZVUKU V LIDSKÉM TĚLE

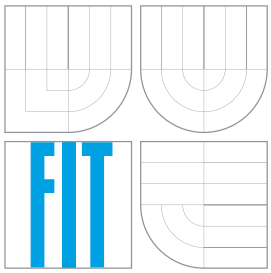
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR KLEPÁRNÍK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE ŠÍŘENÍ ULTRAZVUKU V LIDSKÉM TĚLE

VISUALISATION OF ULTRASOUND PROPAGATION IN THE HUMAN BODY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR KLEPÁRNÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2014

Abstrakt

Tato práce se věnuje 2D a 3D vizualizaci výstupů softwarového balíku k-Wave, který slouží k simulaci šíření ultrazvuku v lidském těle. Balík k-Wave běžně generuje specifická výstupní data nadměrných velikostí (v řádu desítek GB), proto je třeba řešit předzpracování těchto dat. Vizualizace těchto dat má za cíl usnadnit uživatelům zkoumání výsledků simulace. Práce se odvíjí od formátu simulačních dat - HDF5, hledá nejvhodnější způsob pro rychlé čtení těchto dat. Hlavní částí práce je návrh a popis implementace konzolové aplikace pro předzpracování rozměrných simulačních výstupů z k-Wave a aplikace s grafickým uživatelským rozhraním pro interaktivní zobrazování předzpracovaných dat. Nejpodstatnějšími funkcemi těchto aplikací je podvzorkování dat, změna formátu jejich uložení, zobrazování 2D řezů, planární a volumetrické zobrazení, animace průběhu simulace. Navržená implementace umožňuje průchod vizualizovanou doménou o velikosti jednotek GB s latencí v řádu ms - toto významně zefektivňuje práci vědcům a lékařům v oblasti HIFU.

Abstract

This work deals with the 2D and 3D visualization of simulation outputs from the k-Wave toolbox. This toolbox, designed to accurately model the propagation of ultrasound waves in the human body, usually generates immense amounts of output data (up to hundreds of GB). That is why new methods for both the visualization and the effective data representation are necessary to be developed to help users to easily understand the simulation results. This thesis elaborates on the data format, simulation outputs are stored in, with the use of the HDF5 library and looking for the best way to quickly read the simulation data. Finally, the thesis presents the design and the implementation of the console-based application for big simulation data pre-processing and the GUI-based application for interactive visualization of the pre-processed data. The most significant features of these applications are downsampling data, changing the format of storing, viewing 2D sections, planar and volumetric visualization and animation of the simulation process. The proposed implementation allows parts of the simulation domain to be visualised within tens of milliseconds even if the simulation domain comprises GBs of data - This significantly streamlines the work of scientists and clinicians in the field of HIFU.

Klíčová slova

k-Wave, Vizualizace šíření ultrazvuku, HDF5, Ultrazvuk, HIFU, Volume rendering, Vizualizace medicínských dat

Keywords

k-Wave, Visualisation of Ultrasound, HDF5, Ultrasound, HIFU, Volume rendering, Medical Data Visualization

Citace

Petr Klepárník: Vizualizace šíření ultrazvuku v lidském těle, diplomová práce, Brno, FIT VUT v Brně, 2014

Vizualizace šíření ultrazvuku v lidském těle

Prohlášení

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

.....
Petr Klepárník
26. května 2014

Poděkování

Tímto bych chtěl poděkovat vedoucímu své diplomové práce, Ing. Michalu Španělovi, Ph.D, za pomoc, připomínky, nápady a bezproblémovou komunikaci a Ing. Jiřímu Jarošovi, Ph.D, za konzultace ohledně softwaru k-Wave.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Jedním z fenoménů dnešní doby je rozvoj medicíny, lékařských postupů a hledání nových způsobů léčby, například rakoviny. Způsoby likvidace nádorů jsou různé. Poměrně novou metodou s velkým potenciálem je použití přesně nasměrovaného ultrazvuku o vysoké intenzitě (HIFU[1]), který dokáže neinvazivně ničit nádory tím, že je zahřívá uvnitř těla. Skupina počítačových výzkumníků pracuje na programových nástrojích (k-Wave), které umožňují provádět simulaci šíření ultrazvuku v lidském těle (k-Wave open source acoustics toolbox [1]). Tato simulace je prospěšná především pro následnou přesnou aplikaci ultrazvuku v reálném prostředí lidského těla.

Ze strany vývojářů projektu k-Wave vznikla potřeba rychlé a interaktivní vizualizace výstupních simulačních dat, které mohou běžně nabývat obrovských rozměrů, a to v řádu desítek GB. Aktuálně je vizualizace výstupu simulací možná pouze v programu Matlab. Vzhledem k velikosti dat je časově velmi náročná a pro běžného uživatele je tak velmi problematická. Je třeba zobrazovat data, kterými jsou například hodnoty akustického tlaku v různých časových krocích nebo rychlost kmitajících akustických částic, v kombinaci s CT[2] snímky, a to vše v trojrozměrném prostoru. Cílem mé práce je tedy navrhnout a implementovat konzolovou aplikaci pro předzpracování velkých dat (podvzorkování, změna formátu uložení) z nástroje k-Wave, která může běžet na superpočítači, a na ni navazující grafickou aplikaci pro vizualizaci předzpracovaných dat - zobrazení 2D řezů v axiální, koronální a sagitální rovině, planární zobrazení v 3D, volumetrické zobrazování, mapování hodnot na barevné spektrum, animace průběhu kroků simulace.

V této práci se nejdříve věnuji projektu k-Wave (Kapitola 2). Náplň této práce představují obrovská simulační data (Kapitola 2.2) uložená ve formátu HDF5 (Kapitola 2.1) a jejich formát uložení. Pojednává o již existujících volně dostupných nástrojích pro vizualizaci medicínských dat (Kapitola 3), o formátech, které jsou pro uchovávání dat využívány a později v návrhu o možnostech kombinace s výstupy k-Wave. V Kapitole 5 je rozepsán návrh cílových aplikací. Důležitými částmi projektu je postupné zpracování dat, uživatelské rozhraní, různé podoby zobrazení výsledků simulace (2D, 3D, Volume Rendering). Implementaci navrhovaného řešení popisuji v Kapitole 6. Experimentování, s načítáním různými způsoby uložených dat, je prezentováno v Kapitole 7. Možné budoucí pokračování projektu a návrhy na vylepšení aplikace rozebírám v závěru práce (Kapitola 8).

[1] High-Intensity Focused Ultrasound (HIFU) http://www.cancerresearchuk.org/cancer-help/about-cancer/treatment/other/high-intensity-focused-ultrasound-hifu

[2] Computed Tomography (http://www.cancer.gov/cancertopics/factsheet/detection/CT)

Kapitola 2

Softwarový balík k-Wave

k-Wave[1] je volně dostupný softwarový balík pro Matlab a C++. Je navržený pro simulaci šíření akustických vln v 1D, 2D nebo 3D fluidním médiu (vzduch, voda, měkké tkáně) v časové doméně. Využití je například ultrazvukové snímkování, plánování operačních zákroků, optimalizace ultrazvukových měničů, trénování ultrazvukových specialistů, fotoakustika a mnoho dalších. Jeho jádrem je numerický model, který přesně popisuje lineární i nelineární šíření ultrazvukové vlny v heterogenním médiu, včetně absorpce a disperze. K řešení diferenciálních rovnic používá k-prostorovou pseudo-spektrální metodu. Software je schopen běžet v distribuovaném prostředí a efektivně využívat jednotky tisíc procesorových jader. Projekt k-Wave je stále v aktivním vývoji [1, 2, 3].

Pro tuto práci jsou podstatné především výstupy nástrojů k-Wave, uložené ve formátu HDF5 a jejich následné zpracování. Možnost rychle a účelně vizualizovat výstupy simulací umožní efektivnější budoucí vývoj projektu a celkově by měla zlepšit výsledky aplikace HIFU v reálném prostředí.

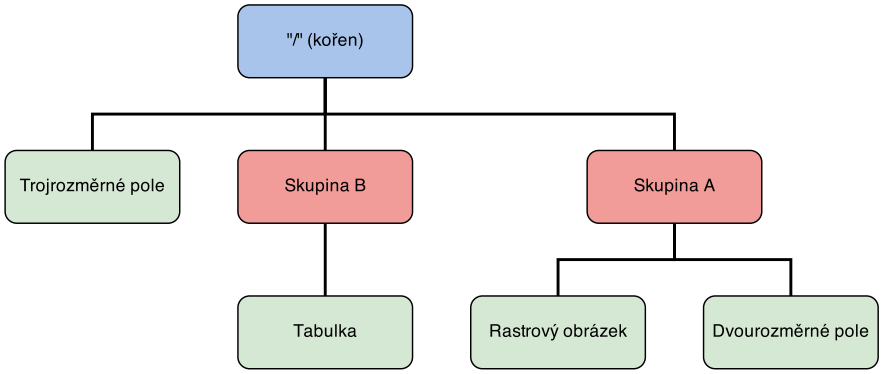
2.1 Formát HDF5

HDF5 je datový model [4], knihovna a souborový formát pro ukládání, zápis a správu objemných komplexních dat. Má již dvacetiletou historii. Knihovna podporuje libovolné datové typy, je přenositelná od běžných PC až po masivní paralelní systémy, implementována v jazycích C, C++, Fotran a Java. Pro jazyk C je k dispozici paralelní verze, která používá MPI[2] (Message Passing Interface) - zasílání zpráv mezi jednotlivými uzly počítačových clusterů. Velikost HDF5 souboru není omezena. Je to hierarchický formát, což umožňuje systematické uložení dat a struktur. Knihovna je volně dostupná a je určena pro práci s vědeckými daty. Oproti jiným knihovnám, jako jsou netCDF, PDB, TIFF, má spoustu výhod. Srovnání s konkurujícími knihovnami je dostupné ze zdroje [5]. Podle uvedených údajů vyniká například v rychlosti čtení a zápisu, ve schopnosti pracovat po síti a paralelně nebo v možnosti rozšiřování dat v jakékoliv dimenzi. HDF5 používají společnosti jako např. NASA.

Knihovna je založena na jednoduchém flexibilním datovém modelu. Primárně model obsahuje dva datové typy: datasety a skupiny. Dataset je vícerozměrné pole, jednoduchý datový typ nebo složený datový typ. Skupiny umožňují vytváření komplexních hierarchických datových závislostí podobně jako v UNIXovém souborovém systému. U datasetů a skupin

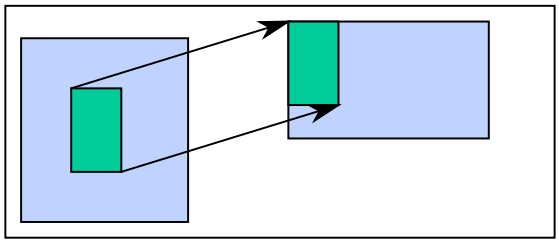
[1] http://www.k-wave.org
[2] http://www.open-mpi.org

mohou být uloženy tzv. atributy, což jsou malá metadata primitivních datových typů. Na Obrázku 2.1 je základní schéma datového modelu.

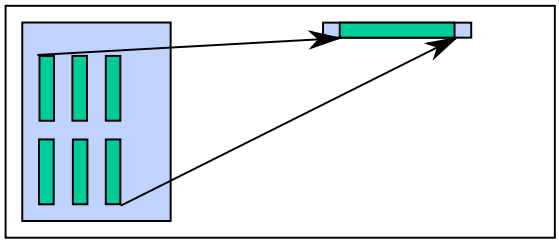


Obrázek 2.1: Znázornění datového modelu HDF5.

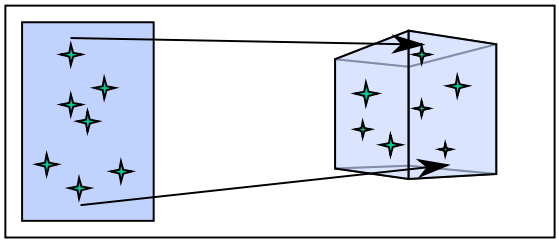
HDF5 umožňuje prostřednictvím svého API během I/O operací pracovat s tzv. hyperslabs, což jsou podmnožiny datasetů. Z trojrozměrného datasetu lze tak např. načíst pouze jeden 2D řez nebo nějaký jiný 3D útvar. Na dílčí části dat lze také aplikovat prostorové a datově typové transformace. Můžeme například vybrat sjednocení dvou množin dat nebo převádět 3D data na kontinuální jednorozměrné pole, apod. Ukázky jsou na Obrázku 2.2.



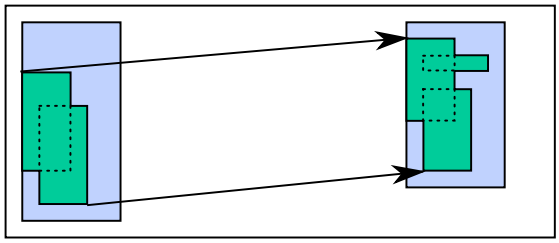
Hyperslab z 2D pole do rohu menšího 2D pole.



Série bloků z 2D pole do souvislého 1D pole.



Sekvence bodů z 2D pole do sekvence bodů v 3D poli.

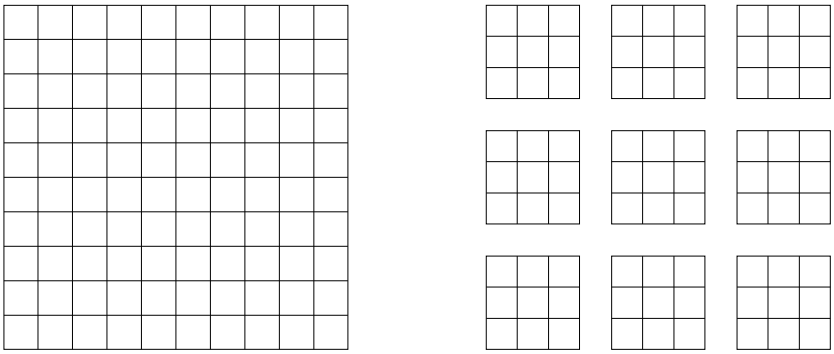


Sjednocení hyperslabů v souboru do sjednocení hyperslabů v paměti. Počet prvků musí být stejný.

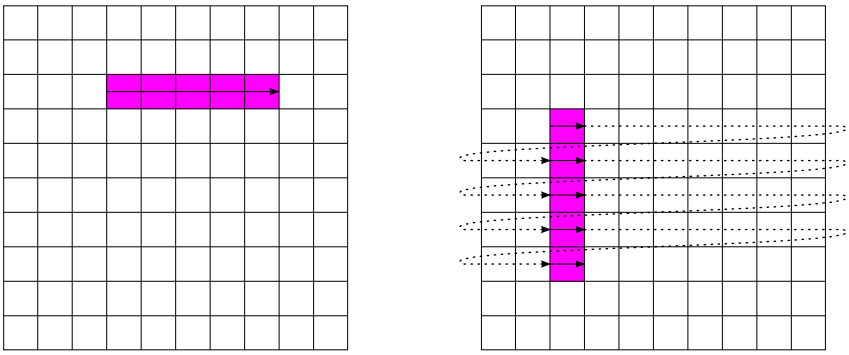
Obrázek 2.2: Příklady prostorových operací HDF5 (převzato ze zdroje [5] a upraveno).

Datasety formátu HDF5 jsou navrhovány pro maximální výkon přístupu k datům. Knihovna poskytuje způsoby specifikace toho, jak budou data uložena na disku. Datasety mohou být reprezentovány jako vícerozměrná pole (až do dimenze 32), nicméně na disku (nízkoúrovňový soubor) musí být vždy uloženy jako jednorozměrný blok dat. V HDF5 existují v základu dva způsoby tzv. rozložení dat. Prvním z nich je rozložení souvislé (contiguous layout), kde jsou data na disku uložena jako jednolitý blok. Druhým způsobem je blokové rozložení (chunked layout), které uloží data na disk do více samostatných bloků stejné velikosti. Využívání tzv. chunků zvyšuje rychlost práce s daty a také např. efektivní pa-

ralelní zpracování jednoho datasetu. Na Obrázku 2.3 je znázornění souvislého a blokového rozložení, na Obrázku 2.4 je příklad čtení souvislého datasetu a příklad čtení blokového datasetu je na Obrázku 2.5.



Obrázek 2.3: Ilustrace rozložení datasetu ve 2D. Vlevo contiguous layout, vpravo chunked layout.

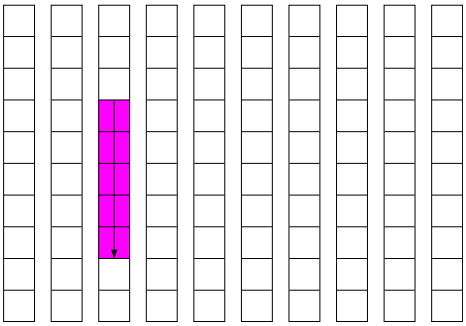


Obrázek 2.4: Příklad čtení souvisle uloženého datasetu. Vlevo část řádku, vpravo část slupce (Převzato z dokumentace HDF5 [4]).

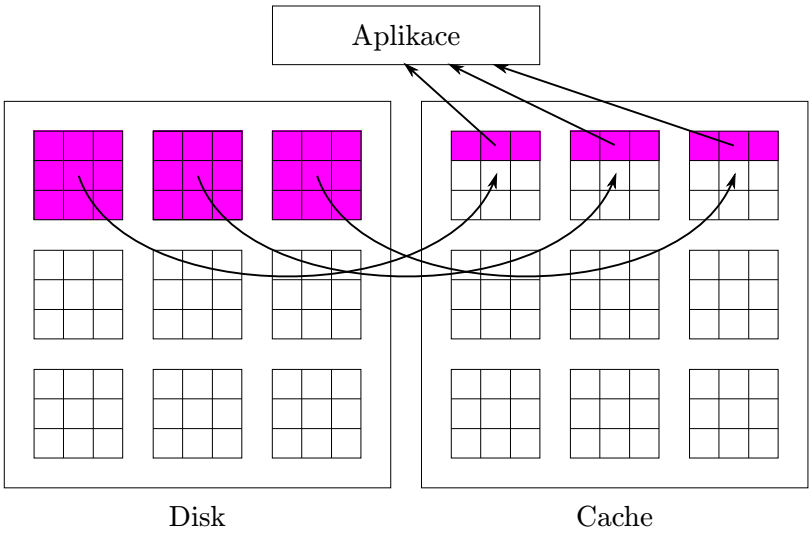
Knihovna HDF5 má implementovanou vlastní cache pro dva typy dat: meta data a čistá data (raw data). Meta data cache uchovává objekty jako hlavičku souboru, tabulky symbolů, uzly B-stromů pro skupiny a chunky, hlavičky objektů a jiné. Pro čistá data lze v případě souvislého rozložení dat nastavovat tzv. Data sieve buffer. Tento buffer je využíván pro čtení nebo zápis podmnožin datasetů. Při použití chunků se používá tzv. chunk cache. Tato dočasná paměť se snaží redukovat počet přístupů na disk tím, že si uchovává některé již zpracovávané chunky (Obrázek 2.6). Všechny parametry cache lze nastavovat při otevírání HDF5 souboru.

K dispozici je také spousta nástrojů usnadňujících práci s HDF5. Knihovna má pro různé operace rozhraní příkazové řádky (kopírování, výpis, změna rozložení,...). Sdružení HDF poskytuje aplikaci HDFView[3], která slouží k jednoduché rychlé editaci a čtení dat prostřednictvím grafického uživatelského rozhraní. Použití a integraci knihovny zjednodušují tutoriály a příklady, které jsou dostupné na webu [4].

[3] http://www.hdfgroup.org/products/java/hdf-java-html/hdfview/index.html



Obrázek 2.5: Příklad čtení blokově uloženého datasetu (převzato z dokumentace HDF5 [4]).



Obrázek 2.6: Čtení blokově uloženého datasetu při použití chunk cache [4].

2.2 Výstupy k-Wave

Výstupy simulací, se kterými budu pracovat, jsou trojrozměrná data. Jsou to buď časově proměnné série udávající akustický tlak (dataset s názvem `p`), akustickou rychlost (`u`), intenzitu (`I`) a jejich agregované hodnoty (min, max, rms), či finální hodnoty. Tato data jsou uložena buď jako jedna 3D matice (například finální rychlost nebo tlak), nebo souvisle v jednorozměrném poli, indexovány senzorovou maskou. Data tedy nemusí být pouze ve tvaru kvádru, ale mohou být libovolného tvaru (např. koule). Senzorová maska je vhodná také pro ukládání časových kroků simulace.

Velikost simulačních dat je obrovská. Pokud máme například kostku s rozlišením 256^3, která obsahuje tlak finální, maximální a tisíc 2D snímků v čase, má tento soubor asi 400 MB. V případě kostky o rozměrech 1024^3 je velikost cca 12,8 GB. Hlavní příčinou takové velikosti dat je fakt, že vypočtené hodnoty jsou datového typu `float (32-bit)`. Aby bylo reálné s takto objemnými daty pracovat (např. je ukládat a načítat po částech, paralelně), je pro jejich ukládání používána právě knihovna HDF5, popsaná v Kapitole 2.1.

Proces simulace zahrnuje alespoň dva soubory ve formátu HDF5. Jedná se o vstupní a výstupní data. Oba typy souborů mají podobnou strukturu, ale obsahují odlišná data. Ve vstupním souboru jsou především parametry simulace, ve výstupním souboru jsou hlavní vypočítané hodnoty simulace. Následně popíši důležité datasety z pohledu vizualizace.

Datasety Nx, Ny, Nz a Nt
Jak vstupní, tak výstupní soubor obsahuje informace o rozměrech simulace (Nx, Ny, Nz) a počet časových kroků (Nt). Hodnoty jsou datového typu `long (unsigned int 64-bit)`.

Dataset c0
Tento dataset je obsažen ve vstupním souboru. Jedná se o hodnoty datového typu `float`, odpovídající CT snímkům (původně ve formátu DICOM), které jsou přepočítány do rozměrů simulace.

Dataset sensor_mask_index
Jednorozměrné pole typu `long`. Obsahuje hodnoty, ze kterých se spočítají jim odpovídající pozice v prostoru, ve kterých bylo prováděno vzorkování simulace. Senzorová maska se prozatím nachází pouze ve vstupním souboru (do budoucna by měla být také v souboru výstupním), ale je důležitá pro zpracování výstupů simulace. Velikost pole odpovídá počtu všech bodů zadaných senzorovou maskou (dále také $Nsens$). Pro výpočet pozice v prostoru platí:

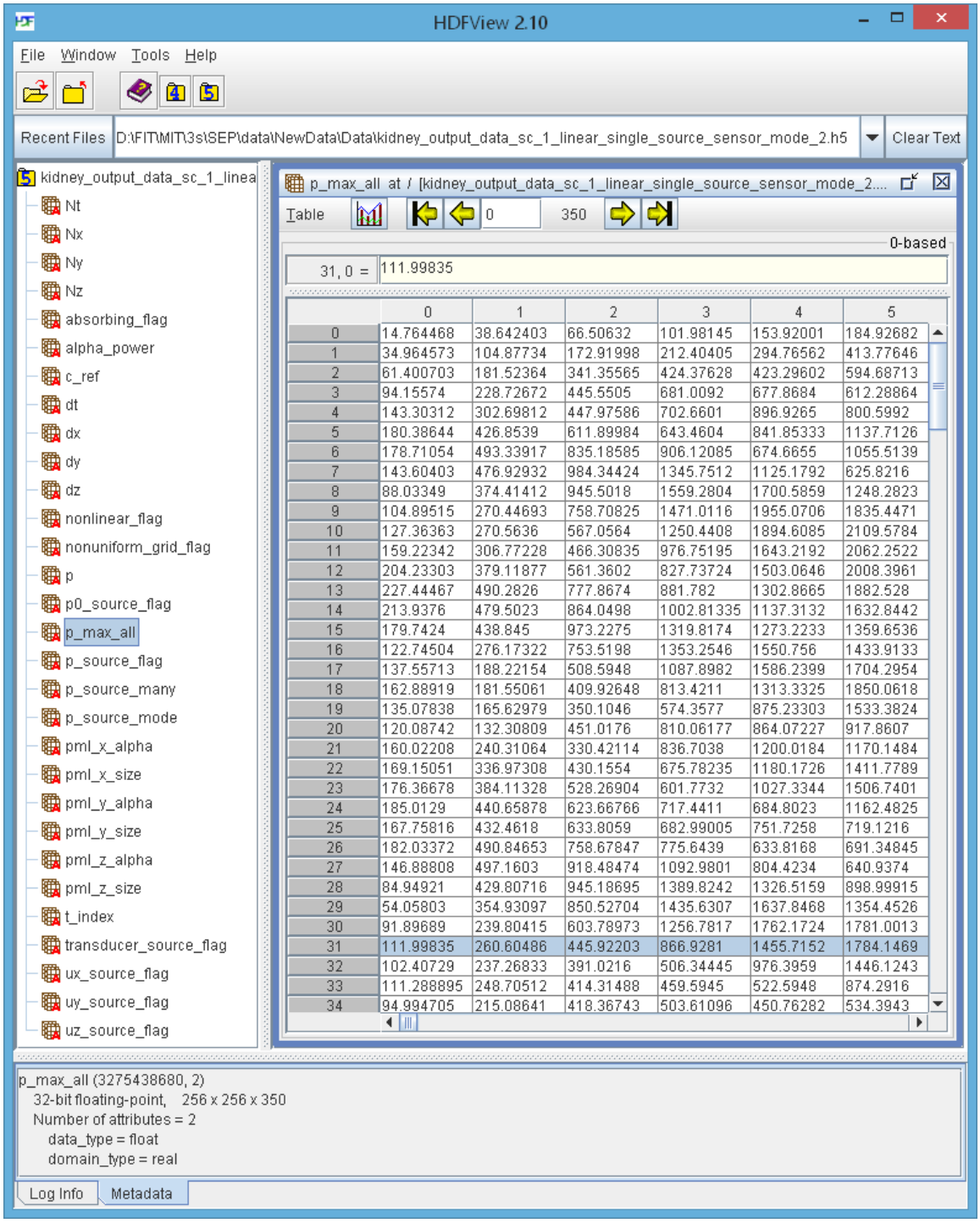
$$z = ceil(i/(Nx \times Ny)), \tag{2.1}$$
$$y = ceil((mod(i-1,(Nx \times Ny)) + 1)/Nx), \tag{2.2}$$
$$x = mod(mod(i-1,(Nx \times Ny)), Nx) + 1, \tag{2.3}$$

kde z, x, y jsou prostorové souřadnice (indexovány od 1). Nx, Ny, Nz jsou dimenze simulace, i je hodnota senzorové masky (index), která může nabývat hodnot od 1 do $(Nx \times Ny \times Nz)$, $ceil$ je funkce, která zaokrouhluje na vyšší celé číslo a $mod(x, y)$ je zbytek po celočíselném dělení x/y.

Datasety k vizualizaci
Pro vizualizaci jsou klíčové 3 typy datasetů. Kromě datasetu `c0` jsou všechny ve výstupním souboru. Obsahují hodnoty datového typu `float 32-bit` a jsou to 3D matice. Jejich rozměry budu uvádět zápisem „(sx, sy, sz)“, kde sx, sy a sz jsou rozměry v jednotlivých dimenzích.

1. Prvním typem jsou datasety velikosti (Nx, Ny, Nz). V této formě je uložen například dataset `p_max_all`, `p_final` nebo také již zmiňovaný dataset `c0` ve vstupním souboru.
2. Dále jsou uloženy datasety o rozměrech $(Nsens, 1, 1)$, kde hodnota $Nsens$ odpovídá velikosti senzorové masky a pro indexování hodnot této dimenze do prostoru se aplikuje právě senzorová maska. Takto je uložen například dataset `p_max`.
3. Poslední podoba dat o velikostech $(Nsens, Nt, 1)$ uchovává časové kroky simulace (Nt). Jedná se tedy o Nt krát větší data než v předchozím případě. Takto je uložen například akustický tlak (dataset `p`) nebo akustická rychlost ve třech směrech.

Ostatní datasety
V souborech je velké množství dalších datasetů, které ovšem nejsou pro vizualizaci důležité. Uloženy jsou různé příznaky simulace, absorpční koeficienty, vlastnosti charakterizující prostředí simulace a jiné proměnné. Ukázka výstupního souboru je na Obrázku 2.7.



	0	1	2	3	4	5
0	14.764468	38.642403	66.50632	101.98145	153.92001	184.92682
1	34.964573	104.87734	172.91998	212.40405	294.76562	413.77646
2	61.400703	181.52364	341.35565	424.37628	423.29602	594.68713
3	94.15574	228.72672	445.5505	681.0092	677.8684	612.28864
4	143.30312	302.69812	447.97586	702.6601	896.9265	800.5992
5	180.38644	426.8539	611.89984	643.4604	841.85333	1137.7126
6	178.71054	493.33917	835.18585	906.12085	674.6655	1055.5139
7	143.60403	476.92932	984.34424	1345.7512	1125.1792	625.8216
8	88.03349	374.41412	945.5018	1559.2804	1700.5859	1248.2823
9	104.89515	270.44693	758.70825	1471.0116	1955.0706	1835.4471
10	127.36363	270.5636	567.0564	1250.4408	1894.6085	2109.5784
11	159.22342	306.77228	466.30835	976.75195	1643.2192	2062.2522
12	204.23303	379.11877	561.3602	827.73724	1503.0646	2008.3961
13	227.44467	490.2826	777.8674	881.782	1302.8665	1882.528
14	213.9376	479.5023	864.0498	1002.81335	1137.3132	1632.8442
15	179.7424	438.845	973.2275	1319.8174	1273.2233	1359.6536
16	122.74504	276.17322	753.5198	1353.2546	1550.756	1433.9133
17	137.55713	188.22154	508.5948	1087.8982	1586.2399	1704.2954
18	162.88919	181.55061	409.92648	813.4211	1313.3325	1850.0618
19	135.07838	165.62979	350.1046	574.3577	875.23303	1533.3824
20	120.08742	132.30809	451.0176	810.06177	864.07227	917.8607
21	160.02208	240.31064	330.42114	836.7038	1200.0184	1170.1484
22	169.15051	336.97308	430.1554	675.78235	1180.1726	1411.7789
23	176.36678	384.11328	528.26904	601.7732	1027.3344	1506.7401
24	185.0129	440.65878	623.66766	717.4411	684.8023	1162.4825
25	167.75816	432.4618	633.8059	682.99005	751.7258	719.1216
26	182.03372	490.84653	758.67847	775.6439	633.8168	691.34845
27	146.88808	497.1603	918.48474	1092.9801	804.4234	640.9374
28	84.94921	429.80716	945.18695	1389.8242	1326.5159	898.99915
29	54.05803	354.93097	850.52704	1435.6307	1637.8468	1354.4526
30	91.89689	239.80415	603.78973	1256.7817	1762.1724	1781.0013
31	111.99835	260.60486	445.92203	866.9281	1455.7152	1784.1469
32	102.40729	237.26833	391.0216	506.34445	976.3959	1446.1243
33	111.288895	248.70512	414.31488	459.5945	522.5948	874.2916
34	94.994705	215.08641	418.36743	503.61096	450.76282	534.3943

Obrázek 2.7: Příklad výstupního souboru otevřeného v programu HDFView.

Kapitola 3

Vizualizace medicínských dat

Současná medicína je velmi úzce spojena s počítačovým zpracováním medicínských dat, které mohou mít mnoho podob. V této kapitole se zaměřím na data obrazová, jejich zdroje, formáty a způsoby vizualizace.

3.1 Zdroje dat

Existují tzv. medicínské zobrazovací metody, jejichž výstupem jsou obrazy [6]. Díky těmto technikám vidíme skrze lidské tělo ve vysokém rozlišení a bez potřebných operačních zákroků. Mezi hlavní metody patří:

- CT: Computed Tomography, počítačová tomografie
- RTG: Rentgen, (např. digitální radiografie)
- Ultrazvuk: diagnostická sonografie
- PET: Positron Emission Tomography, pozitronová tomografie
- MRI: Magnetic Rezonance Imaging, magnetická rezonance

Každá metoda má specifické výsledky a je vhodná pro jiné typy tkání. Rentgen je například vhodný pro zobrazování tvrdých tkání (kostí). Ultrazvuk (sonografické vyšetření) je velmi citlivý na šum a hodí se pro snímání měkkých tkání. CT je vhodnější pro 3D zobrazení a snímá daleko vyšší detaily než rentgen. Při CT a RTG je pacient vystaven radiaci, u magnetické rezonance tomu tak není. MR zaznamenává působení rotujícího magnetického pole na různé tkáně, nezobrazuje kosti a je vhodnější pro měkké tkáně, které nejsou tak kontrastní v CT. Snímky získané těmito metodami se mohou zpracovávat a ukládat do různých formátů, z nichž je asi nejznámější a nejpoužívanější formát DICOM (Kapitola 3.2). Pro objemové zobrazování se data ukládají do 3D mřížek, které mohou mít různé podoby - ortogonální, polární, nestrukturované a další [7]. Uložení dat v paměti může mít hierarchickou strukturu pro úsporu paměti a zachování detailů.

3.2 DICOM

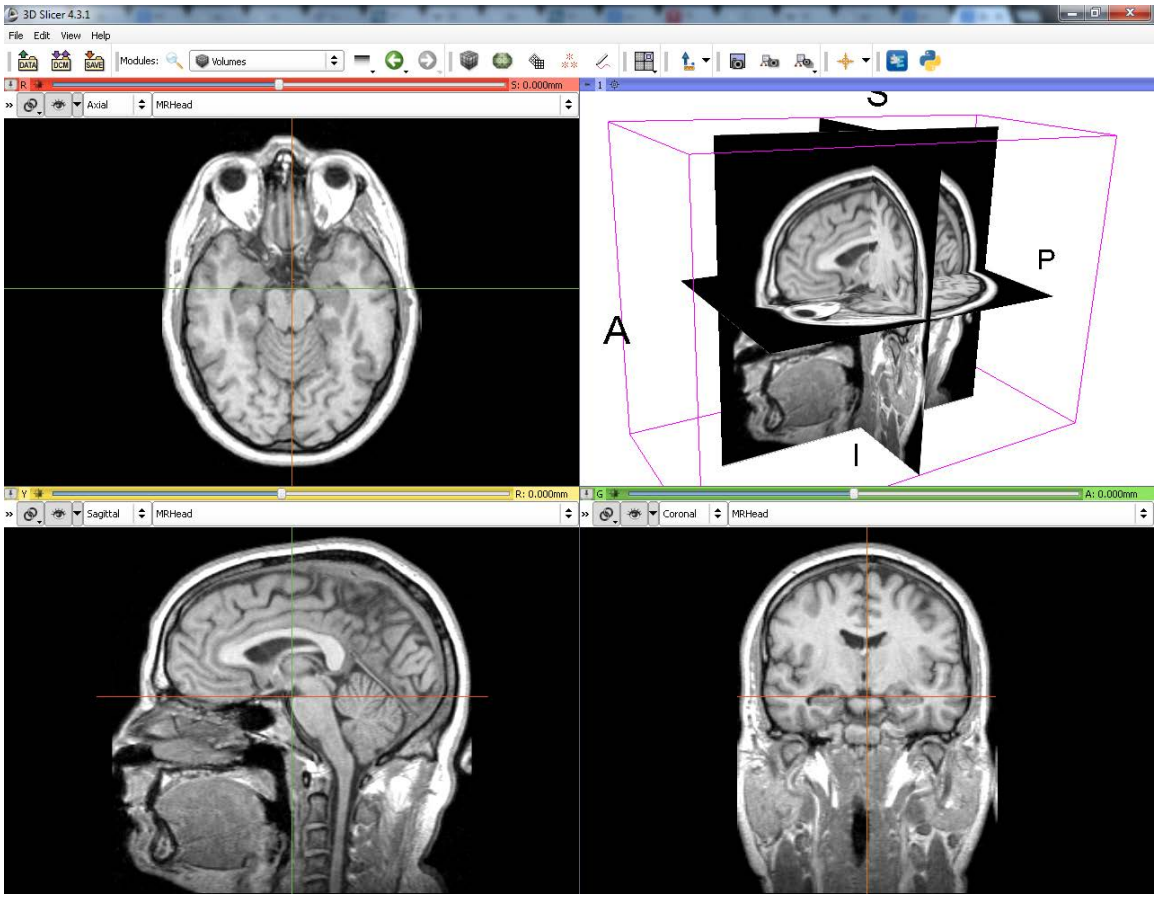
DICOM [8] je standard pro zobrazování, skladování, distribuci a tisk medicínských dat pořízených snímacími metodami. Vznikl v roce 1993. Standard je velmi rozsáhlý. Tento

formát je široce uplatňován v nemocnicích. Jeho části jsou podporovány různými medicínskými zařízeními. Umožňuje snímacím zařízením integraci do systému PACS[1]. V souboru ve formátu DICOM jsou mimo obrazových dat obsaženy také dodatečné informace, jako informace o pacientovi, typ vyšetření apod. V každém souboru je jeden datový objekt obsahující samotné pixely, odpovídající jednomu snímku nebo sérii snímků.

Kromě DICOM existují další formáty pro ukládání dat. Jedná se většinou o volumetrický formát[2], specifický pro konkrétní visualizační software. Příklady formátů jsou DF3, určený pro Povray, PVL, PVM, vol, vtk.

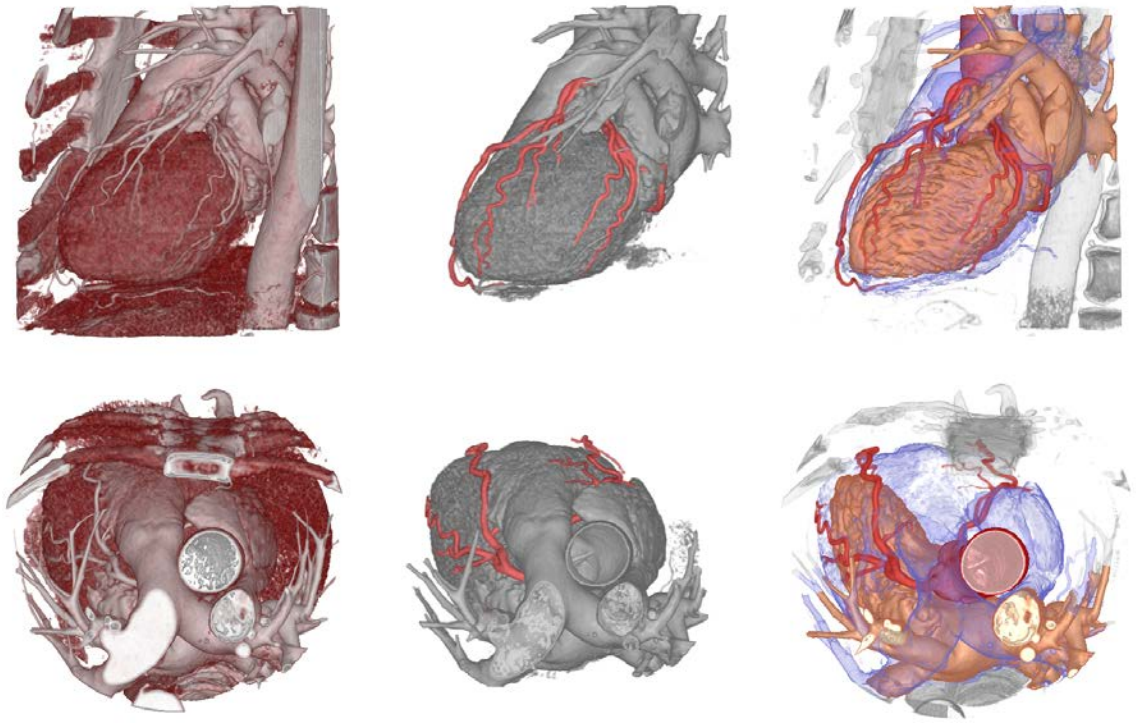
3.3 Způsoby vizualizace

Obecně existují dva způsoby vizualizace - zobrazování řezů (Obrázek 3.1) a 3D zobrazování. Řezy se mohou zobrazovat planárně - tři na sebe kolmé roviny ve 3D (axiální, sagitální a koronální) [9]. Objemová vizualizace má širší možnosti, lze ji realizovat dvěma způsoby - buď budeme zobrazovat povrch nějakého objemu nebo průhledné materiály (Obrázek 3.2).



Obrázek 3.1: Ukázka zobrazení řezů (program Slicer).

[1] Picture Archiving and Communication System
[2] http://paulbourke.net/dataformats/volumetric



Obrázek 3.2: Ukázka volumetrického zobrazení (zdroj Kitware - http://public.kitware.com/ImageVote/images/18).

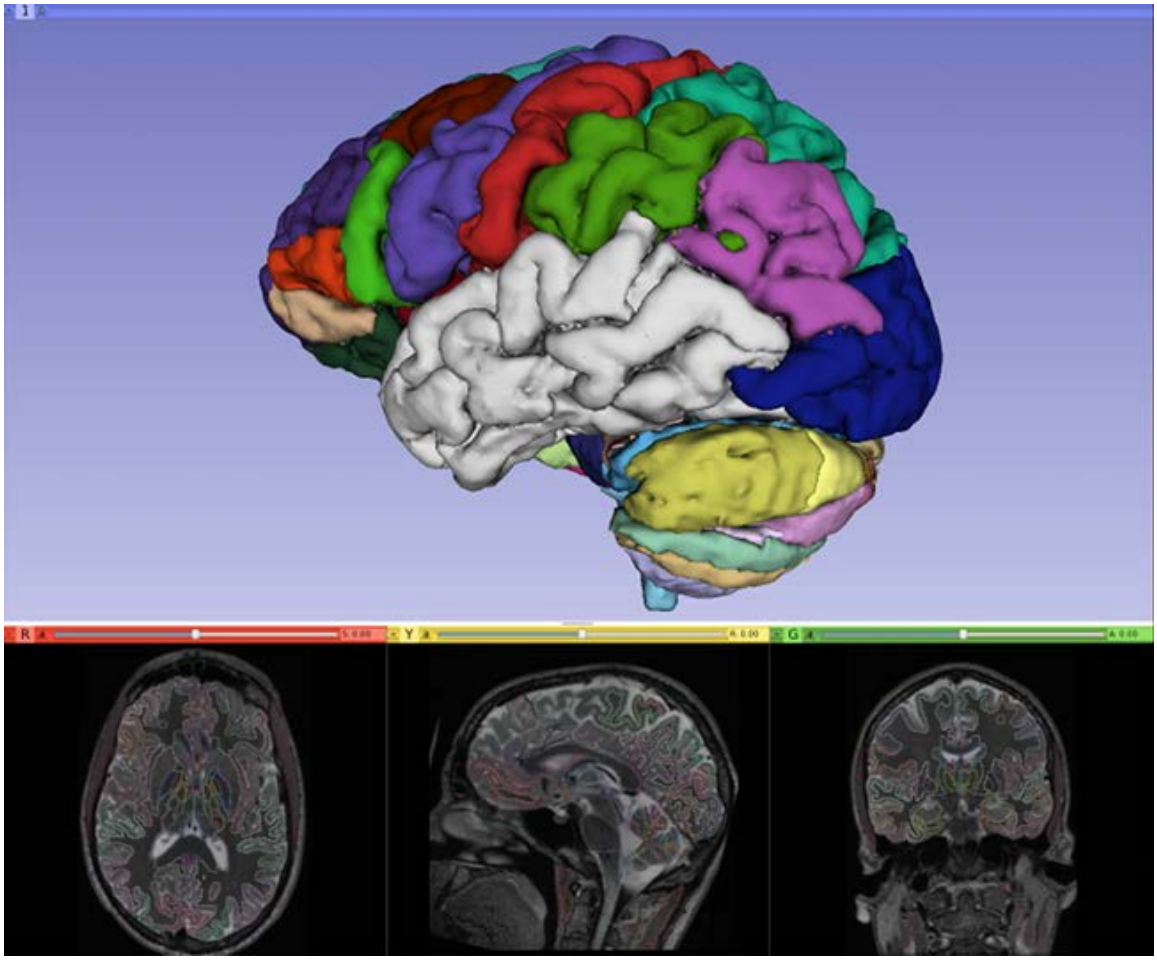
Vykreslování povrchu

Algoritmy zobrazující povrch většinou pracují tak, že se snaží z prostorových dat vytvořit 3D povrch, nejčastěji síť trojúhelníků. Provádí se segmentace dat (Obrázek 3.3). Vypočítané modely jsou potom již klasicky vykreslovány např. přes OpenGL, jako povrchový model s běžnými technikami, jako je texturování, osvětlení. Pro převod do sítě trojúhelníků se používají například algoritmy Marching Cubes nebo Marching Tetrahedra.

Volume rendering

Objemové algoritmy [10, 11] (Direct Volume Rendering) zobrazují přímo uložená data v mřížce. Zde je možnost zobrazovat poloprůhledné materiály. Oproti algoritmům pro vykreslování povrchu jsou tyto metody mnohem více náročnější na paměť. Pro „obarvování“ hodnot voxelů se používá tzv. přenosová funkce. Nejpoužívanějšími metodami pro tento způsob vykreslování jsou:

- Ray-casting - algoritmus vrhání paprsků z pohledu kamery. Výsledná barva pixelu je ovlivněna charakteristikou materiálů, které paprsek protne. Pro různé hodnoty se nastavuje útlum, průhlednost, barva. Metoda je kvalitní, ale náročná na pamět. Optimalizace spočívají v paralelizaci vrhání paprsků, přeskakování prázdných prostorů, podvzorkování, hierarchickém dělení prostoru. Pro implementaci se používají 3D textury a shadery [12, 11].
- Texturování - tato metoda spočívá v zobrazení polygonů - řezů vedených objemem, na které jsou volumetrická data texturována. Jednotlivé řezy jsou obvykle vykreslovány odzadu dopředu a jsou zprůhledněny podle různých parametrů. Průhlednost se větši-



Obrázek 3.3: Segmentace tkání (program Slicer).

nou odvíjí od konkrétních dat. Jednotlivé řezy jsou buď zarovnány kolmo ve směru nejvíce přivrácené strany, zde se používají klasické 2D textury, nebo jsou orientované do pohledu uživatele a tady je vhodné využívat 3D textury. Kvalita výsledků je dobrá při plné hardwarové akceleraci [11].

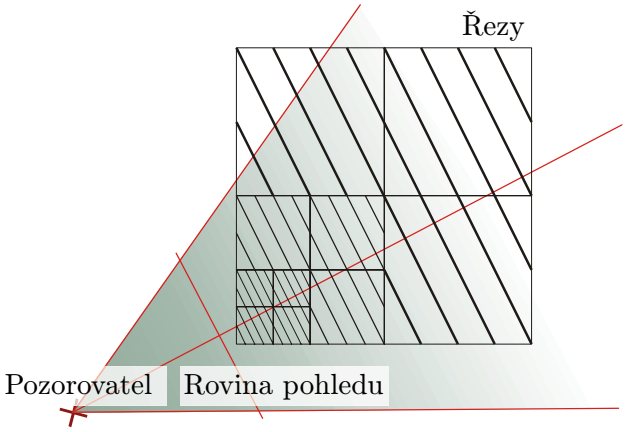
- Různé kombinace uvedených metod, optimalizace, případně další metody jako Splatting[3] nebo Shear warp[4].

3.4 Vizualizace objemných dat

Objemová vizualizace velkých dat, běžně s rozlišením větším než 512^3, je velmi problematická. Grafická karta nedokáže držet v paměti a rychle vykreslovat tak velké množství originálních dat. Tento problém počítačové grafiky se řeší různými způsoby. Následující 2 techniky jsou čerpány ze zdroje [13].

[3] http://www.cs.unc.edu/techreports/91-029.pdf
[4] http://graphics.stanford.edu/papers/shear

Multi-resolution rendering
Typ metody, která především zrychluje vykreslování. Používá se prostorová hierarchie k projekci dat na obrazovku, viz Obrázek 3.4. Pro uchovávání dat se používají struktury jako octree a podobné, jejichž uzly reprezentují data s různým rozlišením. Během vykreslování jsou vybírány data podle rozlišení displeje. Celý dataset v originálním rozlišení tedy nemusí být v paměti grafické karty, ale stále musí být v hlavní paměti.

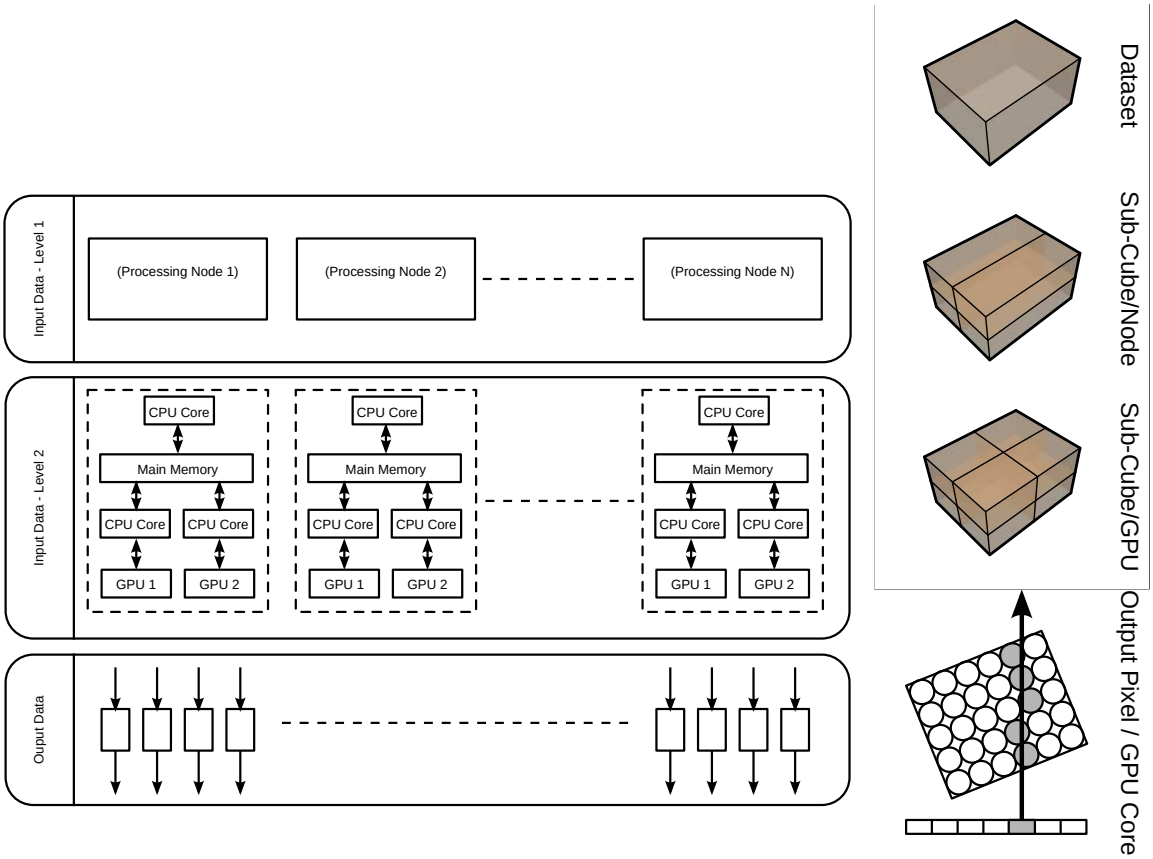


Obrázek 3.4: Multi-resolution rendering s rovinami zarovnanými do pohledu pozorovatele (převzato se zdroje [13]).

Techniky založené na vlnkové transformaci
Vlnková transformace se běžně používá pro kompresi 2D obrazů. Aplikace na objemová data spočívá v rozdělení na pravidelnou mřížku bloků, které jsou nezávisle na sobě zkomprimovány. Tato technika se dále vylepšuje hierarchií více rozlišení. Pro vykreslování se používají ray-casting techniky, které pro zobrazování využívají paralelního hardwaru, nebo se používá technika, která vykresluje „x-ray“ obrazy přímo z vlnkové reprezentace přidáním „rozplácnutých“ hodnot k odpovídajícím bázovým funkcím.

Hierarchická vizualizace velkých datasetů s využitím GPU clusterů
Systém vykreslování založený na texturování, který využívá GPU na clusterech [14]. Data jsou rozdělena na objemové části, je použita hierarchická vlnková transformace pro efektivnější velikost objemů, které mohou být drženy v paměti. Vizualizace je distribuována do prostředí clusterů. Je aplikován adaptivní vykreslovací mechanismus, který bere v potaz parametry okna, do kterého jsou data zobrazována, a vlastnosti datasetu pro přizpůsobení rozlišení, počtu vykreslených texturovaných polygonů a blendingu. Algoritmus se snaží minimalizovat čtení z framebuferu a síťovou komunikaci.

Pro GPU-cluster volume rendering [15] se používá také paralelní ray-casting, společně s rozdělením celého objemu dat na menší části (použití k-d tree). Aplikují se navíc různé optimalizace (např. pro přenos dat po síti). Takto propracované algoritmy a technologie umožňují interaktivně vizualizovat (30 snímků za sekundu) i data o velikosti 200 GB s použitím 128 jednotek GPU. Ilustrace dělení datasetu a různých úrovní zpracování dat je na Obrázku 3.5.



Obrázek 3.5: Ilustrace dělení dat s využitím k-d tree a různých úrovní zpracování [15]. První úroveň obsahuje uzly (4 uzly v příkladě). Ve druhé úrovni je pro každou dílčí část jedno GPU (celkem 8 GPU v příkladě). Na úrovni jednoho GPU jsou počítány hodnoty pixelů. Jedno vlákno GPU odpovídá jednomu pixelu výstupního snímku.

Kapitola 4

Nástroje pro vizualizaci

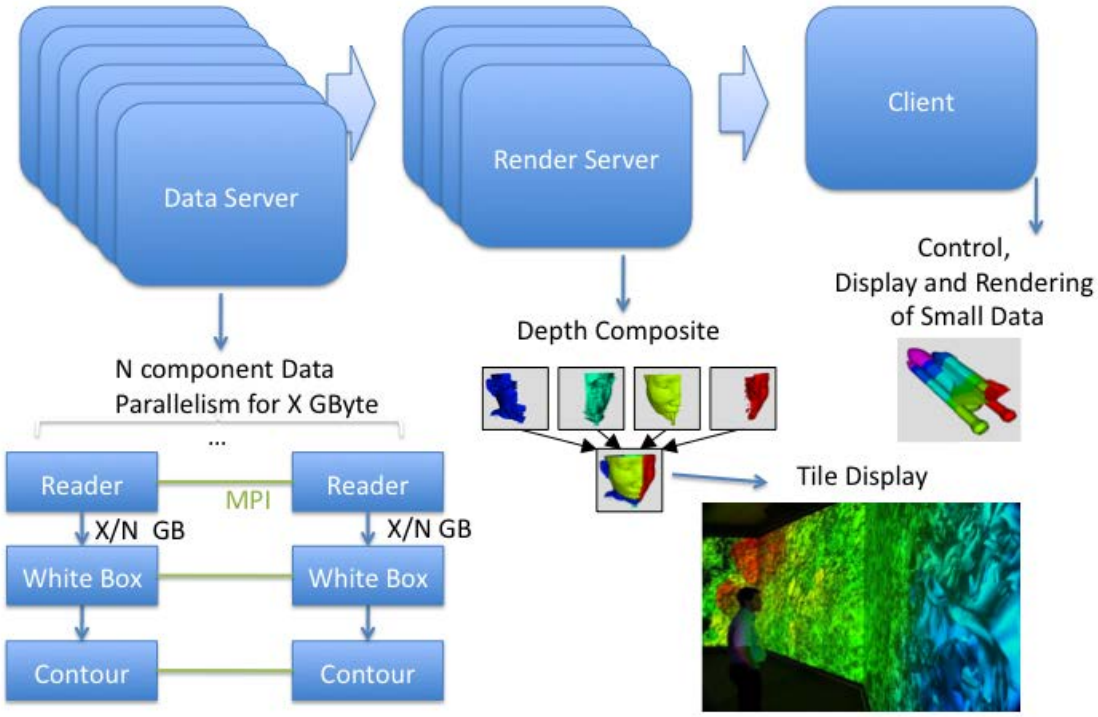
Pro práci s medicínskými daty a jejich vizualizaci existuje široká škála software. Většina nástrojů má společnou vlastnost - podporu formátu DICOM. Uvedu zde několik volně dostupných nástrojů.

VTK[1]
Visualization Toolkit - open-source, volně dostupný software pro 3D počítačovou grafiku, zpracování obrazu a vizualizaci. Obsahuje C++ knihovnu a několik vrstev rozhraní v Tcl/Tk, Javě a Pythonu. Nástroje vytváří a rozšiřuje společnost Kitware. VTK poskytuje spoustu vizualizačních algoritmů, podporuje paralelní běh algoritmů, integruje grafické uživatelské rozhraní jako Qt a Tk, je multiplatformní. Je to asi nejčastěji používaná knihovna pro objemové vykreslování a je velmi komplexní. Z hlediska tématu této práce nabízí volumetrické zobrazování, segmentaci, vykreslování řezů v 3D.

3D Slicer[2]
Program 3D Slicer používá knihovnu VTK a demonstruje její možnosti. Grafické rozhraní je velmi propracované. Hlavními schopnostmi aplikace jsou interaktivní segmentace dat, volume rendering, flexibilní rozložení zobrazení, široké možnosti nastavení mapování barev a průhlednosti na data. Aplikace je schopná komunikovat s databázemi DICOM, renderovat na GPU nebo CPU, rozšiřovat svoje možnosti pomocí pluginů. Ukázka aplikace je na Obrázku 3.1.

ParaView[3]
ParaView je open-source multiplatformní aplikace pro analýzu a vizualizaci dat. Uživatelé mohou rychle vizualizovat svoje data kvalitativními a kvantitativními technikami. Tento nástroj je vytvořen především pro analýzu extrémně velkých datasetů s využitím distribuovaných výpočetních zdrojů - může běžet na superpočítačích. ParaView používá pro zpracování dat a vykreslování knihovnu VTK, pro grafické rozhraní Qt. Zajímavou součástí nástroje je webová aplikace ParaViewWeb, která dokáže vizualizovat 3D data ve webovém prohlížeči s využitím technologií jako WebGL a WebSockets, přičemž potřebné náročné výpočty probíhají vzdáleně na clusterech. Na Obrázku 4.1 je znázorněn paralelismus a vzdálené zpracování dat nástroje ParaView.

[1] http://www.vtk.org
[2] http://www.slicer.org
[3] http://www.paraview.org/

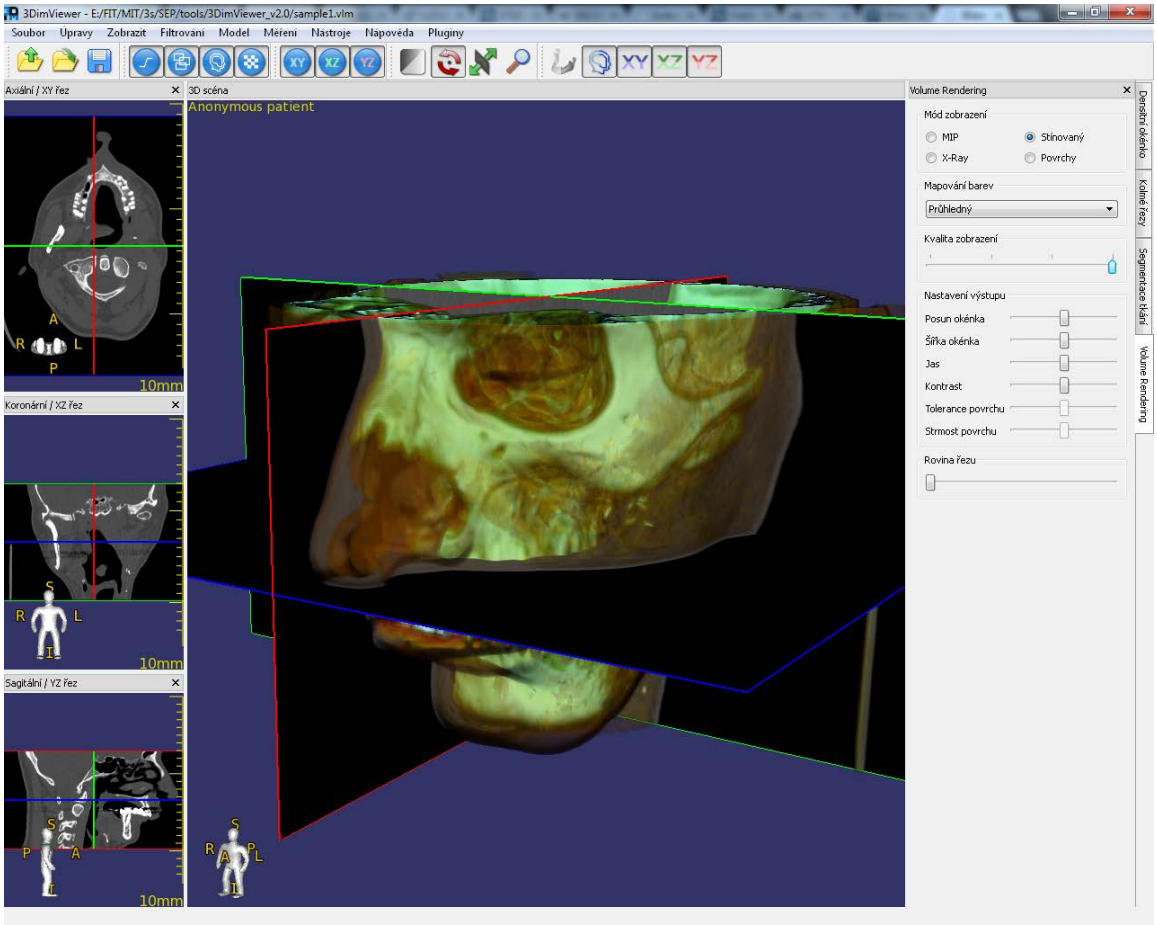


Obrázek 4.1: Paralelní síťová architektura nástroje ParaView. Obrázek pochází z dokumentace nástroje.

3DimViewer[4]

3DimViewer je jednoduchý open-source multiplatformní prohlížeč. Je naspán v C++. Podporuje formát dat DICOM, ze kterých umí vygenerovat a zobrazovat model povrchu (např. lebku). Data zobrazuje multiplanárně nebo objemově. Používá ray-casting algoritmus. V aplikaci lze nastavovat módy zobrazení, měnit parametry prahování, densitního okénka a jiné. Náhled aplikace je na Obrázku 4.2. Do této aplikace by eventuálně šlo implementovat načítání a zobrazování HDF5 3D datasetů jako rozšíření.

[4] http://sourceforge.net/projects/tridimviewer

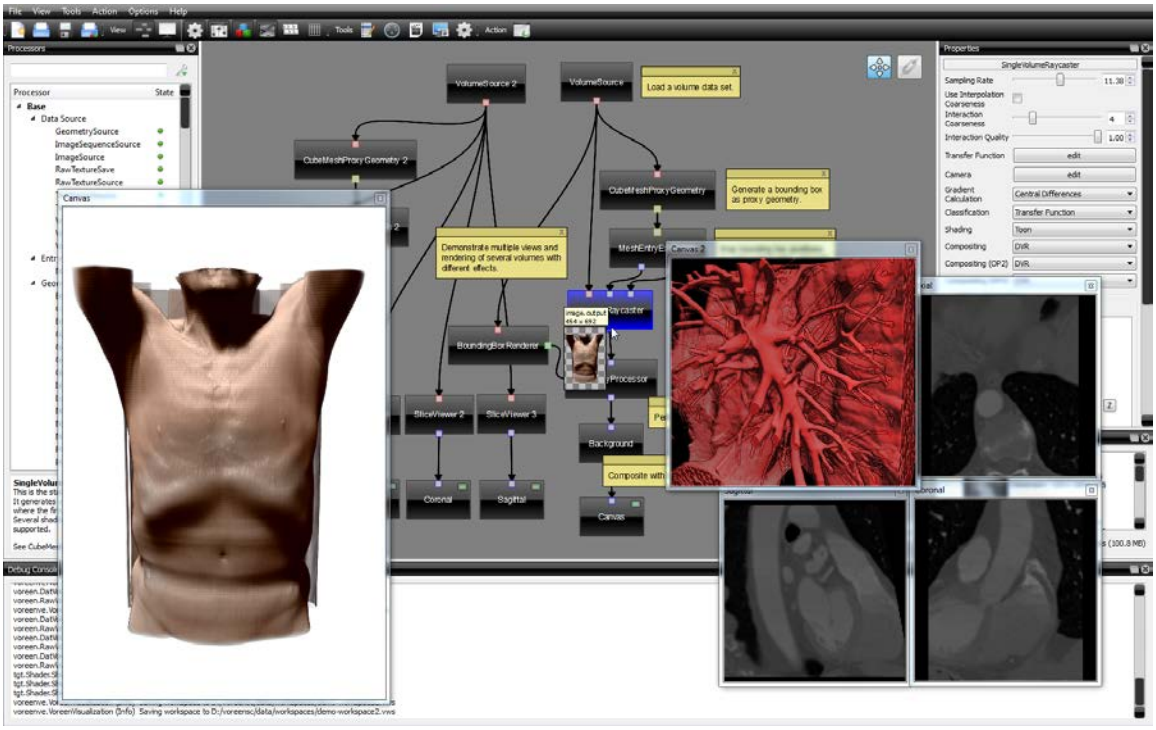


Obrázek 4.2: Aplikace 3DimViewer.

Voreen[5]

Volume rendering engine - multiplatformní open-source aplikace v C++, určená nejen k vizualizaci volumetrických dat. Renderování je postaveno na GPU. Integruje nové vizualizační techniky. Data načítá z velkého množství formátů (DICOM, TIFF stacks, RAW, interfile, Vevo, . . .). Umí generovat animace. Od předchozích aplikací se liší v lépe propracovaném uživatelském rozhraní, ale především ve visuálním programování. Uživatel v programu vytváří data-flow sítě s vysokou úrovní abstrakce, ale zároveň může programovat vlastní shadery. Data-flow koncept se skládá z tzv procesorů, které mezi sebou komunikují přes porty a jednotlivým procesorům se nastavují vlastnosti. Na Obrázku 4.3 je snímek obrazovky s aplikací.

[5] http://www.voreen.org



Obrázek 4.3: Aplikace Voreen. V hlavním okně jsou vizuálně naprogramované propojené bloky, které definují co a jak se bude vykreslovat nebo zpracovávat.

Kapitola 5

Návrh nástroje pro zpracování a vizualizaci simulačních dat

Nejdříve je nutné navrhnout konzolovou aplikaci pro postupné zpracování objemných dat do podoby, se kterou se bude dále možné efektivněji pracovat. Následně je potřebný návrh aplikace s grafickým uživatelským rozhraním, která bude simulační data načítat a interaktivně vizualizovat různými způsoby. Aplikace by měla především usnadnit zkoumání dat. Poté rozepíšu návrh základních a pokročilých vlastností, které by měly cílové aplikace splňovat. Základní vlastnosti jsou důležité a jejich implementace je prioritou této diplomové práce. Na Obrázku 5.1 je schéma činnosti navrhovaných aplikací popsaných níže.

Základní požadavky z hlediska zpracování dat

- Změna rozložení HDF5 datasetů (změna tzv. chunků)
- Podvzorkování datasetů
- Převod datasetů uložených pomocí senzorové masky
- Vyhledání maximální a minimální hodnoty v rámci datasetu nebo celé série

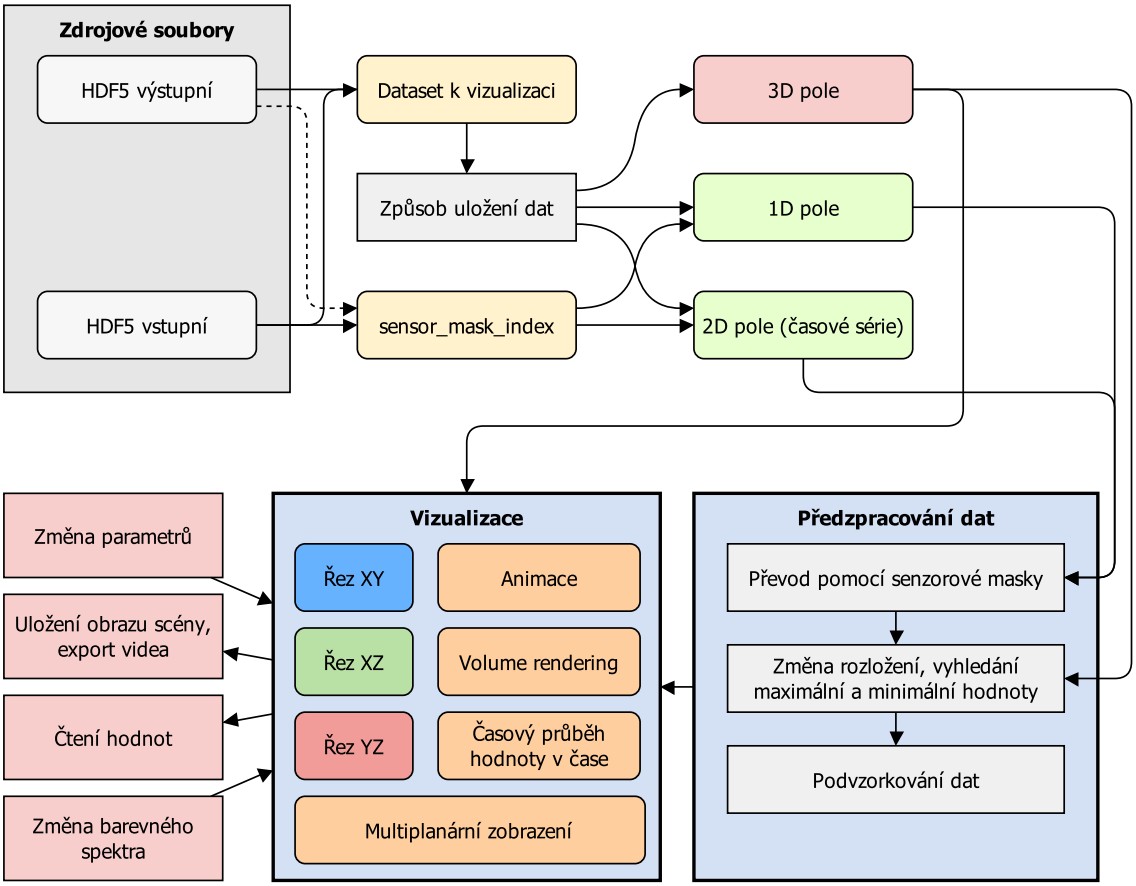
Základní požadavky z hlediska vizualizace

- Zobrazení 2D řezů (axiální, sagitální a koronální)
- Multiplanární zobrazení
- Základní volumetrické zobrazení
- Mapování hodnot na různá barevná spektra
- Změna maximálních a minimálních hodnot při vizualizaci

Pokročilé vlastnosti

- Pokročilé volumetrické zobrazení
- Zobrazení více datasetů zároveň
- Libovolné řezy objemem

- Načítání a vizualizace menších regionů v originálním rozlišení
- Zobrazení grafu časového průběhu hodnoty bodu v čase
- Paralelní zpracování dat
- Generování a export videa ve vysokém rozlišení



Obrázek 5.1: Schéma činnosti aplikace.

5.1 Rychlost načítání dat z formátu HDF5

Pro návrh aplikace je zásadní určit možnosti a omezení HDF5 formátu z pohledu rychlosti načítání dat. Při načítání datasetů prostřednictvím knihovny HDF5, uložených jako 3D matice, jsem došel k hodnotám uvedeným v Tabulce 5.1.

V případě datasetu o velikosti 1024^3 se někdy používala cache pro sousední řezy a načítání v rovině XY tak trvalo 6-8 ms a v rovině XZ asi 250 ms, v rovině YZ cache nefigurovala, pravděpodobně kvůli nedostatku paměti. Načítání v rovině XY bylo rychlejší díky souvislému uložení dat. V původních souborech jsou sice nastaveny chunky o velikosti jednoho řezu v rovině XY, ale pokud nás zajímá rychlost čtení, je to stejné, jakoby nebyly použity chunky vůbec. Pokud se načítají řezy například v rovině YZ, musí se načíst každý řez v XY, a to také odpovídá době cca 200 sekund pro rozlišení 1024^3. Testování probíhalo

Tabulka 5.1: Naměřené časy při čtení různě velkých datasetů.

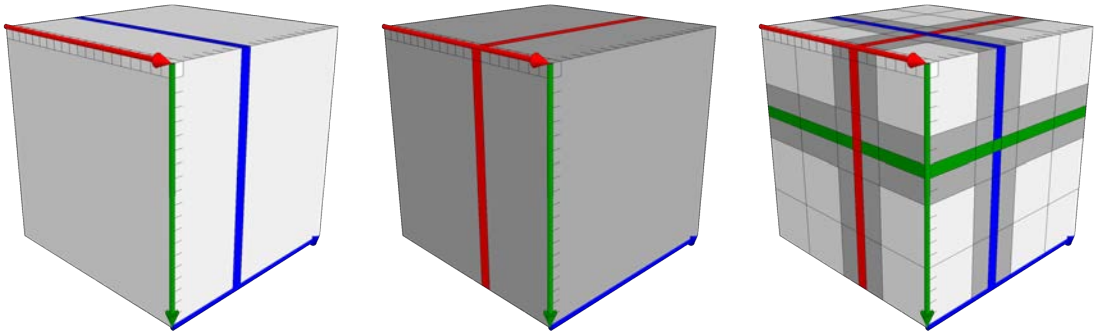
	Čas čtení desky v rovině (ms)		
Rozměry datasetu	XY	XZ	YZ
128^3	0–1	15–20	15–20
256^3	0–1	77–105	77–105
512^3	1–3	580–640	610–670
1024^3	80–300	5 000–11 000	200 000–290 000

na průměrně výkonném, cca 5 let starém notebooku (2 jádrový procesor, 3GB RAM, OS Windows 7). Pro přehlednost uvádím v Tabulce 5.2 závislost rozlišení datasetu na jeho velikosti v paměti.

Tabulka 5.2: Srovnání velikosti dat vzhledem k rozměrům datasetu.

Rozměry datasetu	Velikost desky (B)	Velikost celého datasetu (B)
128^3	65 536	8 388 608
256^3	262 144	67 108 864
512^3	1 048 576	536 870 912
1024^3	4 194 304	4 294 967 296

Vzhledem k uvedeným naměřeným hodnotám je nutné navrhnout rychlejší způsob čtení jiných než souvisle uložených řezů (XY). Variantou řešení, kterou lze vyvážit rychlost načítání ve všech 3 směrech, je použití jiného rozložení (dále tzv. chunků). Tím, že se vytvoří menší 3D bloky, se bohužel zpomalí čtení v rovině XY, ale urychlí se načítání v ostatních směrech, jelikož knihovna HDF5 nebude nucena v případě nesouvisle uložených dat načítat celý dataset, ale pouze jednotlivé chunky, které daný řez protínají. Porovnání naměřených rychlostí při různých velikostech datasetů a chunků je zahrnuto do Kapitoly 7. Na Obrázku 5.2 je znázornění popisovaných situací.



(a) (b) (c)

Obrázek 5.2: Ilustrace principu čtení datasetu: Obrázek 5.2a a 5.2b demonstruje načtení řezů v rovině XY (modrá barva) a YZ (červená barva) bez použití chunků. Obrázek 5.2c ukazuje použití chunků (rovinu XZ značí zelená barva). Šedé zbarvení označuje bloky, které jsou načítány z disku. Šipky znázorňují směr uložení souvislých dat.

5.2 Možnost kombinace výstupů k-Wave s již existujícími vizualizačními nástroji

Pokud bych chtěl v cílové aplikaci využít již existující vizualizační nástroje, musel bych simulační data převádět například do formátu DICOM nebo do jiného formátu, který je široce podporován. Konverze souborů z HDF5 do jiného formátu by znamenal rapidní nárůst počtu souborů. Výstupy k-Wave jsou velmi specifické a mají jiný charakter než medicínské snímky. Simulační data nejsou pouze sady snímků, jak je tomu třeba u CT, ale také série časových kroků simulace uložené pomocí senzorové masky. Nabízí se možnost využít knihovnu VTK pro volumetrické zobrazování. Ostatní nástroje mají zbytečně moc možností a funkcí, které by prozatím neměly uplatnění. Pro tuto práci není ani tak důležité propracované volumetrické zobrazování nějakého dostupného nástroje, ale především zpracování obrovských dat a jejich interaktivní zobrazování.

Pokud by hlavním cílem této práce bylo využití clusterů nebo superpočítačů a paralelismu pro analýzu, zpracování a propracovanou vizualizaci dat, bylo by vhodné využít nástroje ParaView. Tento nástroj je zaměřený právě na uvedené techniky, nabízí flexibilní framework pro řešení specifických problémů. Práce by potom spočívala především v integraci zpracování specifických dat ve formátu HDF5 do frameworku ParaView a ve zprovoznění vizualizace, která je napojena na distribuované a paralelní výpočetní systémy. Výsledná aplikace by potom byla na těchto systémech závislá, ale vyřešil by se problém přesunu objemných dat na jiné zařízení a uživatel by například mohl pro zkoumání dat využívat pouze webový prohlížeč. Rozsah této práce by potom pravděpodobně přesáhl hranice diplomové práce.

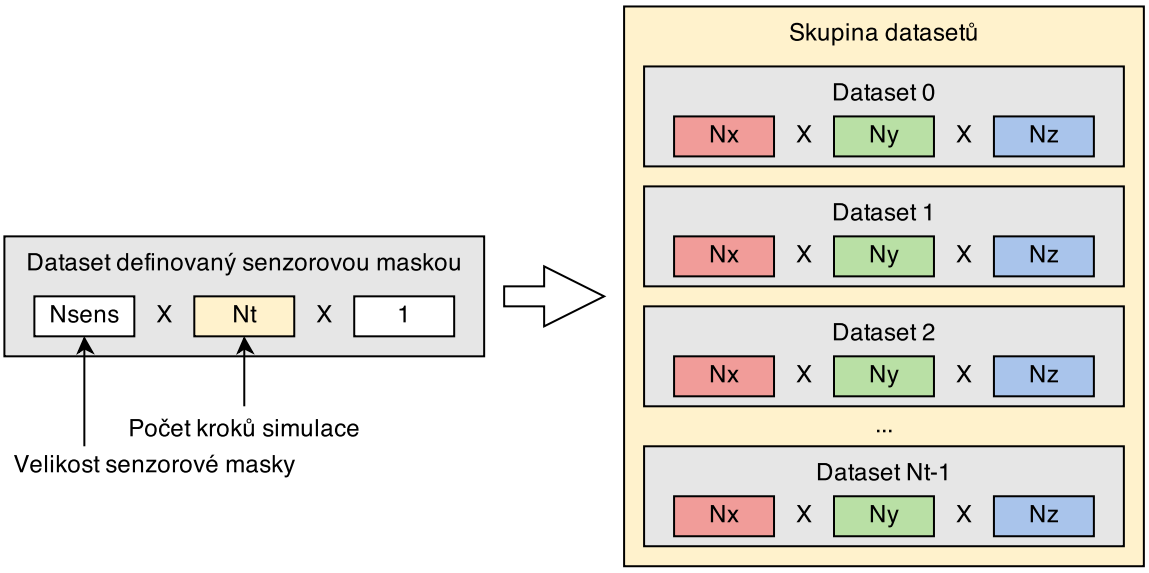
Ke knihovně HDF5 existuje sada utilit pod názvem H5utils[1]. Jednou z nich je program h5tovtk, který provádí konverzi HDF5 datasetů do VTK formátu. Tato aplikace by šla využít, ale nevýhodou je, že není přenositelná - je navržena pouze pro Linuxové systémy. Vzhledem k tomu, že projekt k-Wave je multiplatformní, měla by být přenositelná i vizualizační aplikace.

[1] http://ab-initio.mit.edu/wiki/index.php/H5utils

5.3 Předzpracování dat

Vizualizační aplikace musí být schopna číst data nejrychlejším možným způsobem, a proto je nutné nejprve provést jejich předzpracování. Pro datasety vetší než cca 256^3 bude nutná změna chunků, která umožní efektivnější čtení jednotlivých řezů. Pro rychlé náhledy datasetů a volumetrické zobrazení se budou muset objemná data podvzorkovat na menší rozlišení. Tento převod dat by měl probíhat na výkonném počítači, protože bude pravděpodobně časově velmi náročný. Konzolová aplikace vytvoří nový HDF5 soubor, do kterého nakopíruje datasety ve zvolené menší velikosti. Do nového souboru bude možné uložit dataset i ve více rozlišeních tak, aby nebyl nárůst celkové velikosti souboru extrémní.

Zpracovávat se budou všechny typy datasetů uvedené v Kapitole 2.2. V případě datasetů definovaných senzorovou maskou se tyto datasety převedou do 3D a vytvoří se skupina obsahující všechny časové kroky simulace (Obrázek 5.3). Aby nedošlo k velkému nárůstu velikosti souboru v případech, kdy senzorová maska zahrnuje jen malou část vzhledem k rozměrům simulace, využije se postupná alokace místa vyhrazeného pro dataset společně s optimální velikostí chunků.



Obrázek 5.3: Převod datasetu, definovaného senzorovou maskou s časovými kroky, na skupinu.

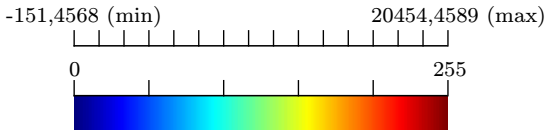
K vizualizaci budou třeba maximální a minimální hodnoty celých datasetů, ty se během zpracování vyhledají a uloží se jako atributy k příslušným datasetům. Veškerá data bude nutné načítat a ukládat po částech, aby nedošlo k nedostatku operační paměti. Vstupem navrhované konzolové aplikace bude tedy výstupní soubor simulace a vstupní soubor pro načtení senzorové masky. U aplikace bude možnost ovlivnit parametry zpracování, jako je velikost chunků, velikost podsamplování, či název výstupního souboru.

5.4 Zobrazení dat

Grafická aplikace otevře již předzpracovaný nebo ještě nezpracovaný soubor a načte z něj datasety, které lze vizualizovat, včetně nabídky jejich možných rozlišení. V případě nezpra-

covaného souboru se načtou pouze 3D datasety. Základem bude zobrazení 2D řezů, a to kolmo k jednotlivým souřadným osám. Polohy jednotlivých řezů musí být možné dynamicky měnit.

Pro zobrazování bude nutný převod z hodnot datového typu `float` na rozsah 0 až 255 a mapování na různá barevná spektra (viz Obrázek 5.4). K mapování hodnot je třeba znát minimální a maximální hodnotu buď z celého datasetu nebo pouze z aktuálního řezu. Tyto hodnoty se načtou buď z atributů daného datasetu nebo, pokud nebyl dataset přezpracován, se vyhledají. Hodnoty se budou dát měnit i ručně prostřednictvím uživatelského rozhraní a tím tak modifikovat grafický výstup.



Obrázek 5.4: Ilustrace mapování hodnot do barevného spektra.

Dále bude možné zjistit vybranou hodnotu v řezu v bodě, například najetím kurzoru myší na řez. Zobrazovat se budou jak finální data, tak časové série simulace. K dispozici bude možnost měnit aktuální krok simulace a animovat celý její průběh. Do základního konceptu zahrnuji ještě mutliplanární zobrazení v 3D a základní volumetrické zobrazování s možností ovlivňovat jeho kvalitu, průhlednost a váhu zastoupení barevných složek. Pro volumetrické vykreslování použiji texturovací metodu s polygony zarovanými směrem k pohledu kamery. Do paměti grafické karty bude nutné nahrávat celý dataset.

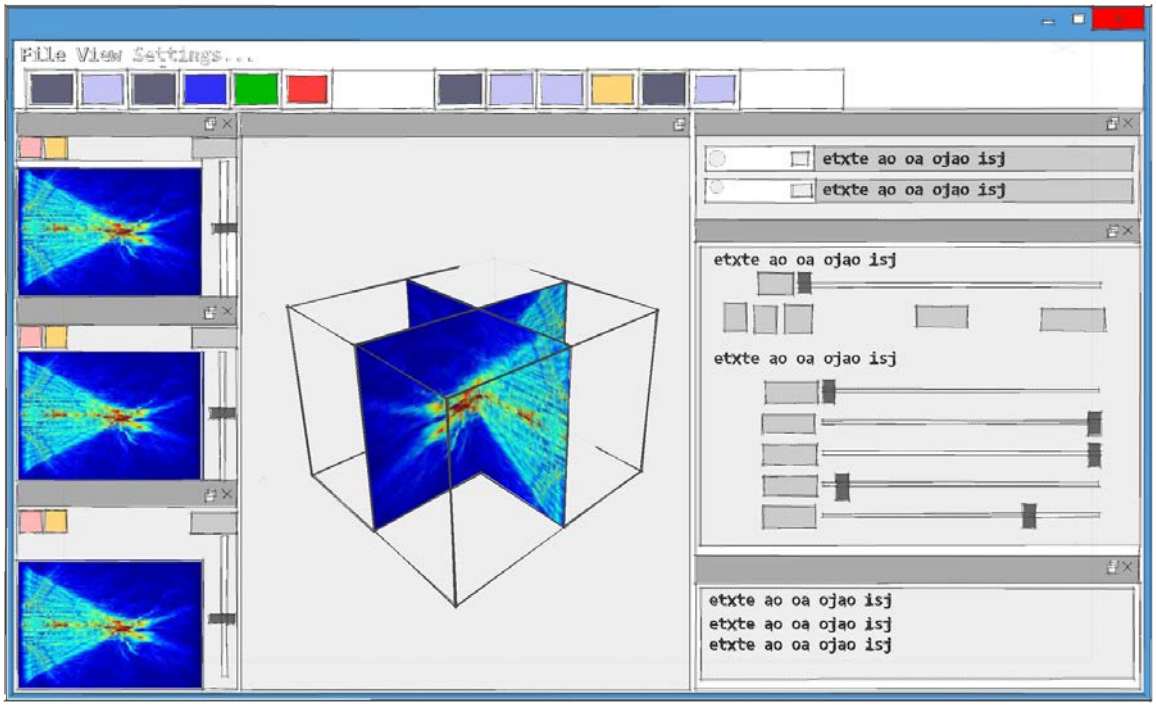
5.5 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní se bude skládat z dokovatelných panelů. Tři panely budou vyhrazeny pro jednotlivé, na sebe kolmé řezy. Další panely budou vyhrazeny pro seznam datasetů, informace a změnu hodnot vybraného datasetu a informace o aktuálním souboru. Centrální okno bude určeno pro 3D scénu. Bude k dispozici nástrojový panel s ikonami, pro rychlou změnu rozhraní. Náčrt návrhu celého rozhraní aplikace je na Obrázku 5.5.

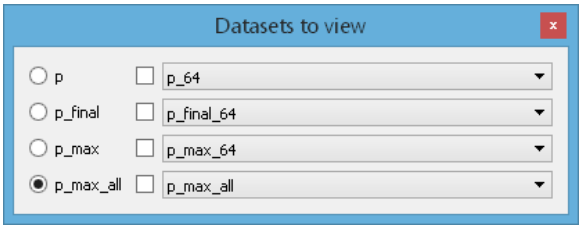
Každé okno s řezem bude obsahovat posuvník (Slider) a editovatelné vstupní pole (Spin Box) pro ruční změnu hodnoty aktuálního řezu, dále tlačítko pro možnost exportu obrázku a tlačítko pro přepínání zobrazení do skutečné či automatické velikosti. Při pohybu kurzoru myši nad vykresleným řezem se bude v oznamovacím řádku aplikace zobrazovat odpovídající hodnota datasetu v daném bodě. Při načítání řezu se bude zobrazovat animace čekání.

Do panelu pro výběr datasetu se vypíší všechny dostupné datasety z otevřeného souboru. V případě, že je k dispozici více velikostí datasetu, vytvoří se nabídka pro výběr (Combo Box) zvoleného rozlišení. Výběr datasetu bude indikovat změna přepínacího tlačítka (Radio Button). Vždy bude vybrán pro modifikaci zobrazení pouze jeden dataset. Pro kontrolu toho, které datasety jsou aktuálně zobrazeny, bude sloužit zaškrtávací políčko (Check Box) u každého datasetu. Návrh okna s výběrem datasetů je na Obrázku 5.6.

Panel pro vybraný dataset bude obsahovat informace - název datasetu, typ, velikost, počet kroků. Budou se zde nacházet především ovládací prvky pro změnu zobrazení, a to nabídka výběru barevného spektra, možnost změny maximální a minimální hodnoty pro mapování hodnot, volba oříznutí hodnot, které jsou nad či pod aktuálně zvolenou hodnotou. Dále bude možnost změny těchto hodnot pouze v rámci jednotlivých řezů. Pro ovládání série



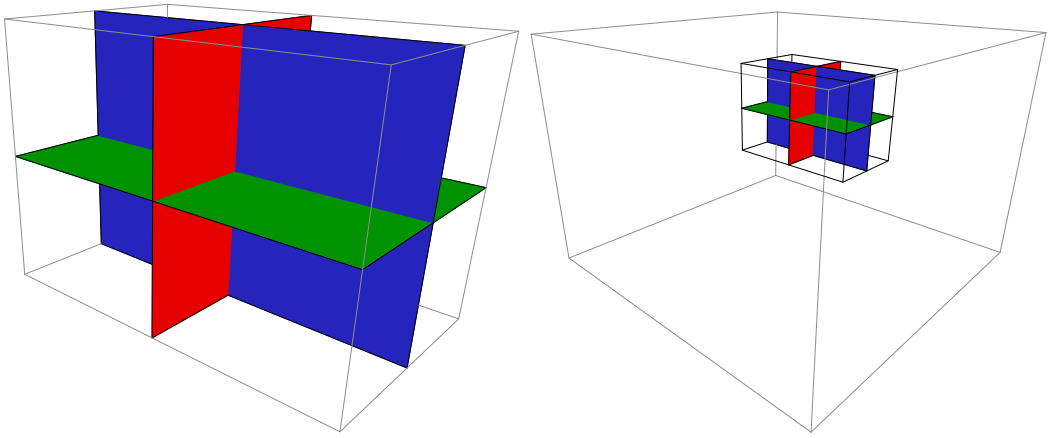
Obrázek 5.5: Náčrt grafického rozhraní aplikace.



Obrázek 5.6: Panel pro výběr datasetu.

časových kroků simulace bude sloužit posuvník pro změnu aktuálního kroku a tlačítka pro přehrávání animace. Nebude zde chybět volba změny intervalu načítání jednotlivých kroků a nastavení přírůstku. Kvalita volumetrického zobrazování půjde ovlivnit nastavením počtu částečně průhledných vykreslených řezů. Odpovídající posuvníky budou měnit průhlednost objemu a průhlednost červené, zelené a modré barvy.

Centrální panel bude obsahovat 3D scénu a v ní vykreslený vycentrovaný 3D rámeček, ohraničující rozměry datasetu. Uvnitř rámce budou 3 na sebe kolmé řezy (v rovinách XY, XZ, YZ) - multiplanární zobrazení (Obrázek 5.7) a vykreslený objem. Oba tyto módy se budou moci povolit nebo zakázat pomocí kontrolního panelu. Pokud bude dataset definován senzorovou maskou, bude se moci zobrazit buď relativně vzhledem k velikosti celého prostoru simulace, nebo se roztáhne a vyplní 3D scénu (Obrázek 5.7). Ve scéně bude možné otáčet polohou kamery kolem středu, scéna se bude moci přibližovat, oddalovat a také posouvat. To vše pomocí tlačítek a pohybu myši. Ohraničující rámeček bude možné skrýt. V kontrolním panelu budou mimo jiné také tlačítka pro základní zarovnání pohledů a export 3D scény jako png obrázku s průhledným pozadím. Při rotaci scény se automaticky degraduje kvalita objemu pro rychlejší vykreslování.



Obrázek 5.7: Vpravo relativní zobrazení, vlevo roztáhnutí a vyplnění 3D scény. Jedná se o multiplanární zobrazení.

5.6 Ostatní vlastnosti a technologie

Výsledné aplikace budou multiplatformní, kompatibilní se systémy Windows a Linux, jejich kód bude přenositelný. Vizualizační program bude vypisovat další údaje ze zpracovávaných souborů informačního charakteru. V rozhraní aplikace bude možnost exportovat aktuální 3D scénu, či jednotlivé řezy, jako obraz. Načítání dat v grafické aplikace bude probíhat v oddělených vláknech proto, aby se grafické uživatelské rozhraní nezadrhávalo. Kód bude napsaný v C++ a budou použity následující multiplatformní knihovny:

- QT 5[2]- grafické uživatelské rozhraní
- HDF5 (Kapitola 2.1) - zpracování simulačních výstupů k-Wave
- OpenCV[3] - zpracování řezů jako obrazů, mapování hodnot na barevné spektrum
- OpenGL[4] - 3D vykreslování, miminální verze 3.3
- VTK (Kapitola 4) - Případné pokročilé volumetrické zobrazování

5.7 Pokročilé vlastnosti

Zde jsou uvedeny vlastnosti, jejichž implementace by byla s největší pravděpodobností časově náročnější. Pro tuto práci nejsou klíčové, nicméně jejich návrh může být dobrou motivací na aplikaci dále pracovat a efektivně vylepšovat její možnosti.

Pokročilé metody zobrazení
V oblasti volumetrického zobrazování by se daly implementovat metody pro efektivnější práci s velkými daty, případně by se mohla využít nějaká pokročilejší dostupná knihovna. Na vykreslenou objemovou scénu by se mohly aplikovat různé grafické filtry, pro lepší rozlišování lidských orgánů, propracované přenosové funkce, případně segmentace. Dále se

[2] http://qt-project.org
[3] http://opencv.org
[4] http://www.opengl.org

nabízí možnost řezů objemem v libovolných směrech. Za velmi dobrou vlastnost bych také považoval zobrazení více datasetů v jedné 3D scéně. Uživatel by tak mohl sledovat například animaci ultrazvuku v prostředí CT datasetu (snímky lidského těla). Zajímavou schopností aplikace by byla také prostorová vizualizace vektorů rychlosti, v podobě orientovaných šipek. Pro zkoumání simulace by bylo vhodné také vykreslovat graf znázorňující průběh změny hodnoty zvoleného bodu v čase.

Zobrazení objemných dat v originálním rozlišení
Rychlé načítání celých datasetů, respektive jednotlivých řezů ve všech 3 směrech, je v případě rozlišení většího než cca 512^3 na běžném PC pomalé. Způsobem, jakým by se dala efektivně zkoumat originální objemná data, by byla možnost načítání vybraných menších regionů řezů nebo objemové scény v originálním rozlišení, zatímco by se v malém rozlišení zobrazovaly rychlé náhledy.

Paralelní zpracovávání dat na výkonném počítači
Jak jsem již zmínil v teoretické části (Kapitola 2.1), knihovna HDF5 poskytuje paralelní verzi v jazyce C s využitím MPI. Paralelní zápis by mohl urychlovat předzpracování datasetů, jako podsamplování, či změnu chunků, ale především by se dalo ze souboru číst současně z více vláken, a to by výrazně zrychlilo načítání řezů v grafické aplikaci. Paralelní čtení není bohužel ve verzi C++ možné. Dalším způsobem zrychlení vizualizace dat by teoreticky mohlo být využití výkonných počítačů (clusterů), které by měly dostatečně velkou operační paměť a rychlost čtení dat. Problém pomalého načítání dat by odpadl, ale muselo by být k dispozici dostatečně rychlé sítové připojení pro přenos grafického rozhraní aplikace po síti.

Video
Vylepšením aplikace by bylo generování videa ve vysokém rozlišení, a to především z časových sérií. Mohlo by být generováno jak z 2D zobrazení, tak z objemového zobrazení. Zobrazování a ukládání snímků videa by neprobíhalo v reálném čase, generování by trvalo delší dobu.

Jiné vlastnosti a vylepšení
V bodech uvedu další možné vylepšení aplikace, které by bylo zajímavé řešit do budoucna.

- Česká verze aplikace
- Možnost změny polohy řezů prostřednictvím 3D scény
- Převod hodnot tlaku do decibelů
- Odměřování vzdáleností v 3D scéně
- Načítání nastavení uživatelského rozhraní
- Ukládání nastavení zobrazení aktuálního datasetu do HDF5
- Zabudování předzpracování výstupních souborů do grafické aplikace
- Otevření více souborů - záložky
- Celkové vylaďování a testování pro případné zveřejnění na webu k-Wave

Kapitola 6

Implementace

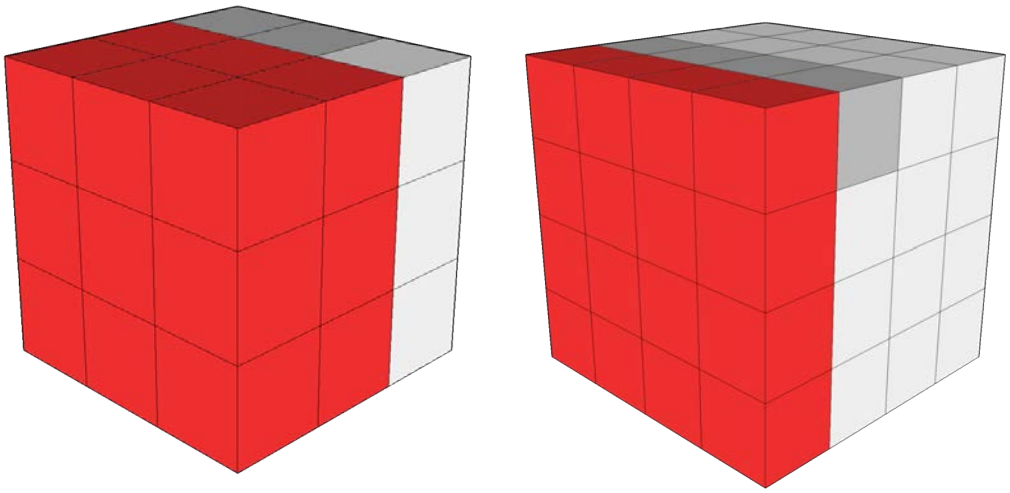
Implementaci jsem si rozdělil do několika částí. Základem je zpracování HDF5 souborů, to je využíváno jak pro konzolovou aplikaci, tak pro aplikaci s grafickým rozhraním. Oddělenou část aplikace tvoří OpenGL okno s 3D scénou. Aplikace jsou přeloženy pro Windows (64-bit), knihovny HDF5 a OpenCV jsou do ní přilinkovány staticky. Grafická aplikace linkuje knihovnu QT dynamicky. Aplikace lze přeložit i na systémech Linux. Pro překlad je využíván multiplatformní nástroj qmake[1], který na daném systému vygeneruje příslušný Makefile.

6.1 Knihovna zapouzdřující práci s HDF5

Vytvořil jsem strukturu C++ tříd, které obstarávají otevřený simulační HDF5 soubor, datasety, skupiny a atributy. Třída `HDF5File` je instancována pomocí názvu souboru. Soubor otevře a načte si hodnoty základních datasetů `Nx`, `Ny`, `Nz` a `Nt` nebo vytvoří nový. Třída obsahuje seznam datasetů a skupin (třídy `HDF5Dataset` a `HDF5Group`), umožňuje tyto objekty vytvářet, otevírat a zavírat. V této třídě jsou také metody pro přepočítávání lineární senzorové masky do 3D souřadnic a zpět. Objekty typu `HDF5Dataset` poskytují metody pro čtení 3D bloku dat odpovídajícího datasetu podle zadané pozice a velikosti, zápis definovaného bloku 3D dat a především čtení po automaticky vypočítaných blocích dat. Z knihovny HDF5 jsou využívány metody pro definici tzv. Hyperslab (podmnožina datasetu). Automatické bloky se počítají pomocí definované číselné hodnoty (`HDF5File::SIZE_OF_DATA_PART`), případně parametru funkce, značící kolik hodnot lze maximálně načíst do paměti najednou. Metoda `initBlockReading(hsize_t maxSize = HDF5File::SIZE_OF_DATA_PART` vypočte maximální možnou velikost zarovnaného 3D bloku dat s ohledem na velikost datasetu a každé zavolání funkce `readBlock` vždy načte následující blok dat, dokud není přečten celý dataset. Implementaci této funkce nebyla primitivní a považuji ji za nezbytnou. Pokud by nebylo takovéto čtení dat k dispozici, mohlo by docházet ke zbytečnému čtení dat po příliš malých blocích nebo by naopak nebylo dostatek operační paměti a docházelo by ke zpomalení v podobě swapování paměti. Tento typ čtení dat je využit především k postupnému převodu dat z lineárního způsobu uložení (senzorová maska) do 3D, pro změnu chunků a podvzorkování. Příklad výpočtu velikosti bloku: Máme datasety o velikostech 3^3 a 4^3 a maximální počet najednou načtených hodnot je 20. V případě prvního datasetu se spočte velikost bloku $2 \times 3 \times 3$ a poslední blok bude mít velikost $1 \times 3 \times 3$. U druhého datasetu se budou číst bloky o velikosti $1 \times 4 \times 4$. Znázornění je na Obrázku 6.1.

Třída `HDF5Dataset` má také metody pro vyhledávání maximální a minimální hodnoty

[1] http://qt-project.org/doc/qt-5/qmake-manual.html



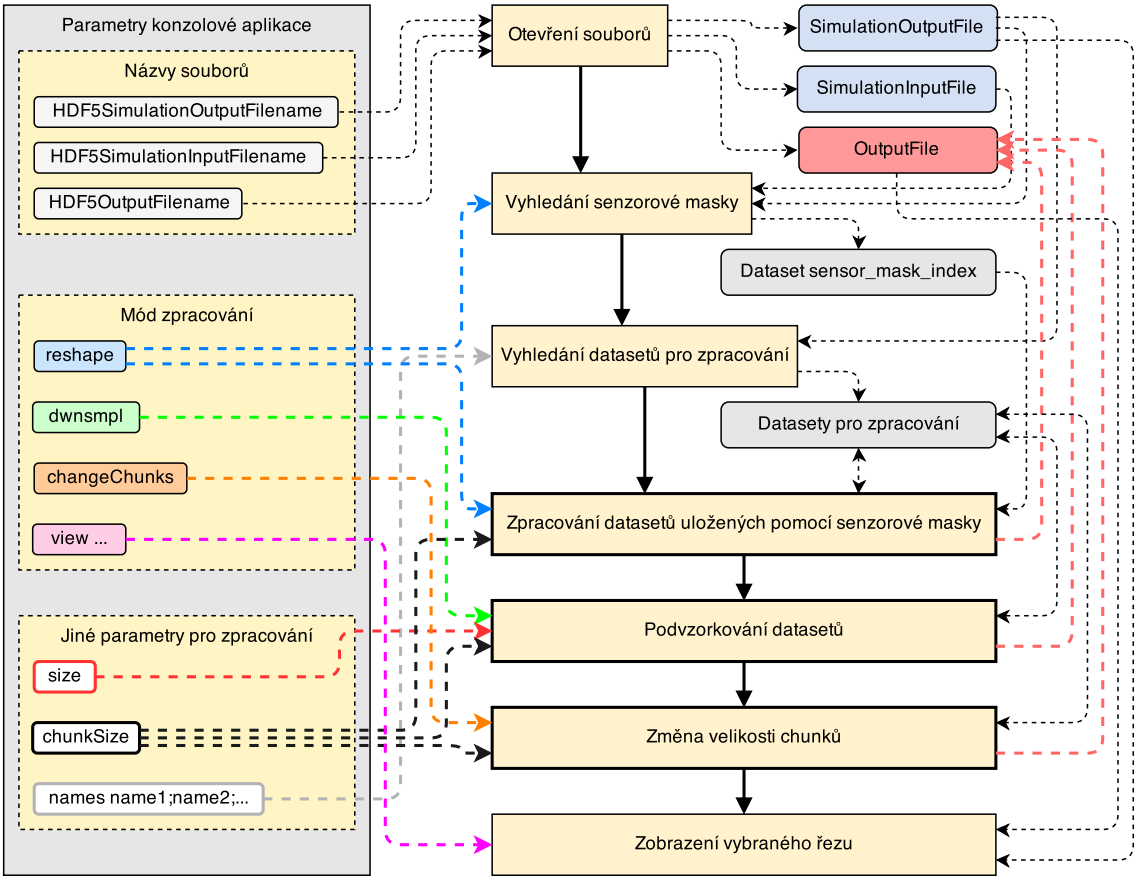
Obrázek 6.1: Příklad výpočtu velikosti bloku dat. Vlevo dataset velikosti 3^3, vpravo dataset velikosti 4^3, červeně jsou vyznačeny 3D bloky, součet červených a šedých voxelů je 20.

a získání velikosti datasetů nebo chunků. Objekty typu `HDF5Dataset` a `HDF5Group` (rodičovská třída `HDF5Object`) mohou zapisovat a číst atributy příslušné skupiny či datasetu.

Implementovaná knihovna s názvem **hdf5file**, zapouzdřující práci s HDF5, je staticky překládána a linkována (Windows) do dalších částí aplikací, využívá C++ verzi knihovny HDF5.

6.2 Konzolová aplikace

Schéma činnosti konzolové aplikace je na Obrázku 6.2. Aplikace načte parametry a otevře soubor se simulačními výstupy, ze kterého načte rozměry simulace. Pokud je zapnutý mód `reshape` vyhledá se senzorová maska, a to ve vstupním nebo výstupním simulačním souboru. Dále jsou vyhledávány datasety, které se dají zpracovávat - jsou to všechny 3D datasety, které mají velikost shodnou s rozměry simulace, datasety uložené pomocí senzorové masky a série datasetů ve skupině pro případné podsamplování. Názvy zpracovávaných datasetů mohou být zadány parametrem (`-names name1;name2;...`). Výstupní soubor, do kterého se budou zapisovat modifikované datasety a skupiny, se buď vytvoří nový nebo se otevře již existující, do kterého se budou data doplňovat nebo přepisovat. Následuje zpracování datasetů uložených pomocí senzorové masky. Podle masky jsou vypočítány velikosti chunků, tak aby bylo alokováno co nejméně místa. Je vytvořena skupina s názvem datasetu a v ní, pro každý krok simulace, dataset přepočítaný do 3D formátu, jehož název odpovídá časovému kroku. Pokud je zapnutý mód `changeChunks` jsou všechny 3D datasety (nebo datasety určené parametrem `-names`) ze vstupního souboru zkopírované do výstupního souboru s novým nastavením velikosti chunků. Mód `dwnsmpl` provádí podvzorkování originálních 3D datasetů, ale také série datasetů ve skupině. Pro tuto operaci jsou použity funkce knihovny OpenCV. Ve výstupním souboru jsou vytvořeny menší kopie datasetů, za jejich název je přidán sufix podle velikosti. Při podvzorkování se vždy najde maximální rozměr (z hodnot `Nx`, `Ny`, `Nz`), ten se zmenší na danou velikost a ostatní se změní ve stejném poměru. Během všech tří typů operací zpracování se vyhledávají maximální a minimální hodnoty v rozsahu celého datasetu, a také v rámci celé série a ukládají se jako atributy

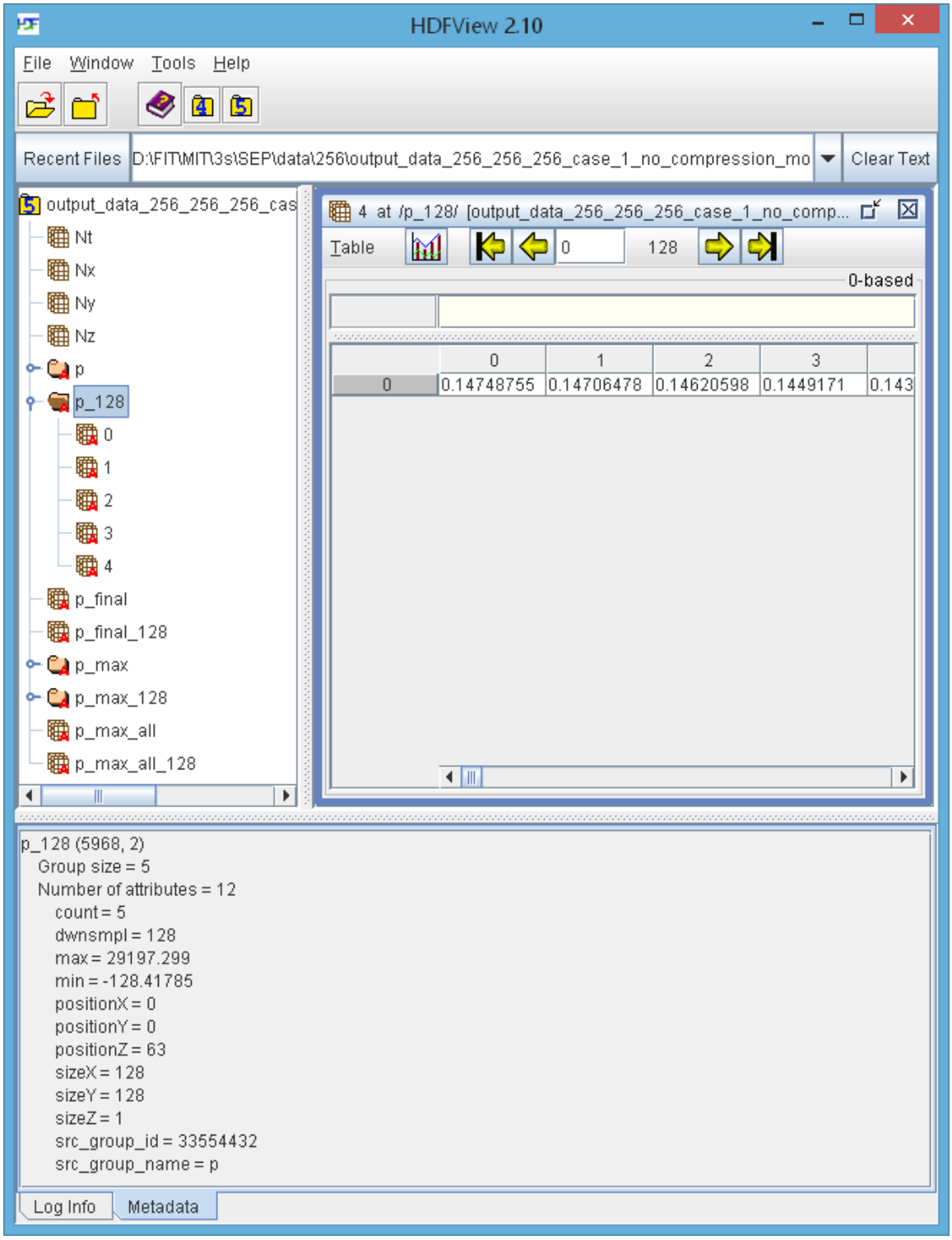


Obrázek 6.2: Diagram znázorňující činnost konzolové aplikace.

k příslušným objektům. Při zpracování maskou definovaných datasetů se do atributů skupin ukládají pomocné údaje jako je počet kroků, pozice a velikost masky v 3D prostoru. U podvzorkování se vytvoří atributy `dwnsmpl` - velikost podvzrokování, `src_group_name` - název původního objektu, `src_group_id` - id původního objektu, pro zachování konektivity s originálním objektem. Konzolová aplikace umí také zobrazit jeden vybraný řez datasetu (mód `view` s parametry). Použití konzolové aplikace `k-wave-h5-processing` je popsáno v Tabulce 6.1. Na Obrázku 6.3 je příklad výstupního souboru aplikace.

Tabulka 6.1: Použití konzolové aplikace (`k-wave-h5-processing [parametry]`).

`-f HDF5SimulationOutputFilename`	Jediný povinný parametr. HDF5 soubor se simulačními výstupy.
`-m HDF5SimulationInputFilename`	Vstupní simulační HDF5 soubor, který obsahuje senzorovou masku (`sensor_mask_index` dataset).
`-o HDF5OutputFilename`	Název výstupního HDF5 souboru, do kterého budou zapisovány modifikované datasety a skupiny. Výchozí hodnota je `HDF5SimulationOutputFilename`_modified.
`-reshape`	Aplikace provede zpracování datasetů uložených pomocí senzorové masky. Musí být k dispozici dataset `sensor_mask_index`. V novém souboru vytvoří skupinu s názvem zpracovávaného datasetu a do ní uloží převedené 3D datasety odpovídající časovým krokům simulace.
`-changeChunks`	Provede změnu chunků 3D datasetů a uloží je do výstupního souboru.
`-view datasetName cutType cutIndex`	Zobrazí řez datasetu `datasetName`. Hodnota `cutType` může být `YX`, `ZX` nebo `ZY`, `cutIndex` je hodnota indexu vybraného řezu (indexováno od 0).
`-dwnsmpl`	Provede podsamlovaní všech dostupných 3D datasetů, včetně sérií ve skupině a uloží je do výstupního souboru.
`-s size`	Volba velikosti pro podsamplování. Hodnota `size` bude odpovídat velikosti nejdelší strany podsamplovaného datasetu.
`-ch chunkSize`	Volba velikosti chunků. Velikost chunku bude `chunkSize`^{3}.
`-names name1;name2;...`	Názvy vybraných datasetů nebo skupin pro zpracování.
`-help`	Vypíše nápovědu.



Obrázek 6.3: Náhled souboru zpracovaného konzolovou aplikací. V souboru jsou ve skupinách datasety `p` a `p_max`, které byly převedeny s využitím senzorové masky (datasetu `p` odpovídají 4 časové kroky), a dále datasety `p_final` a `p_max_all`, což jsou 3D datasety. Jejich podvzorkované verze jsou pojmenovány se sufixem `_128` a zobrazeny jsou také atributy přiřazené ke skupině `p_128`.

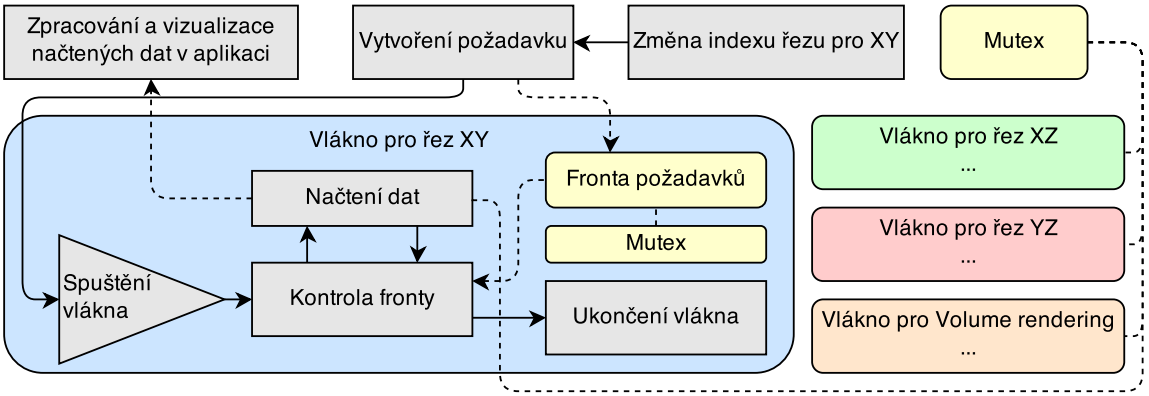
6.3 Grafická vizualizační aplikace

Grafické rozhraní vizualizační aplikace stojí na technologii QT, tedy především na systému tzv. widgetů, signálů a slotů. Signály a sloty jsou propojeny všechny potřebné komponenty GUI s jádrem aplikace, ale také jiné negrafické objekty, například vlákna. Návrh uživatelského rozhraní je popsaný ve zvláštním souboru (`mainwindow.ui`), ve formátu XML. Pro grafické prvky aplikace, jako jsou ikony, jsou využívány tzv. prostředky (Resource Collection Files (.qrc)). V těchto souborech se obvykle vytváří i definice více jazyků aplikace.

Základem aplikace je třída `MainWindow`, která zprostředkovává propojení všech částí aplikace. V této třídě probíhá inicializace všech prvků rozhraní a obsahuje sloty reagující na podněty uživatele, jako je otevření nebo zavření souboru, výběr datasetu, změna maximálních a minimálních hodnot, nastavení volumetrického zobrazení, přehrávání animace, apod.

Po otevření souboru prostřednictvím grafického dialogu se v aplikaci vytvoří struktura tříd (`OpenedH5File`, `H5ObjectToVisualize`, `H5SubobjectToVisualize`) zapouzdřující otevřený soubor a v něm objekty, které lze vizualizovat. Každý dataset nebo skupina má svou instanci třídy, která uchovává hodnoty, jako data aktuálních řezů, současné indexy řezů, maximální a minimální hodnoty a různé příznaky zobrazení. Třída `H5ObjectToVisualize` seskupuje skupiny nebo datasety stejného názvu různých rozlišení, třída `H5SubobjectToVisualize` pak obaluje konkrétní objekt. Při změně vybraného datasetu tak zůstává konfigurace zobrazení jednotlivých objektů zachována.

Reakcí na změnu hodnoty indexu aktuálního řezu je vytvoření požadavku (`Request`) na aktuální řez a spuštení vlákna (`HDF5ReadingThread`). Vlákna zajišťují asynchronní čtení dat proto, aby nezpůsobovalo zasekávání uživatelského rozhraní. Pro každý řez existuje jedno vlákno. Každé vlákno si ukládá čekající požadavek do fronty o délce 1. Vždy je tedy maximálně jeden řez načítán a maximálně jeden může čekat na vyřízení. Pokud se změní hodnota požadovaného indexu řezu a ve vlákně již čeká jiný požadavek na vyřízení, je tento odstraněn a nahrazen novým. Delší fronta by mohla být využita pro předčítání řezů okolo aktuálního řezu. Vzhledem k tomu, že nelze pomocí C++ HDF5 číst paralelně, je nutné vzájemné vyloučení operací čtení datasetu, a to je zajištěno statickým objektem typu QMutex napříč všemi vlákny. V rámci vlákna musí být také výlučný přístup ke frontě požadavků. Princip činnosti vláken je znázorněn na Obrázku 6.4. V okamžiku, kdy jsou data načtena, je odeslán signál s výsledkem. Tento signál je zachycen a dále zpracováván ve třídě `H5SubobjectToVisualize`. Z načtených dat typu `float` je vytvořen objekt knihovny OpenCV typu `cv::Mat`, na který je dále aplikován převod do barevného spektra a je odeslán do GUI (`MainWindow` a `CVImageWidget`). Řez je vykreslený buď v originálním rozlišení, nebo je převzorkován tak, aby vyplnil okno řezu.



Obrázek 6.4: Schéma načítání dat prostřednictvím vláken.

6.4 OpenGL 3D scéna

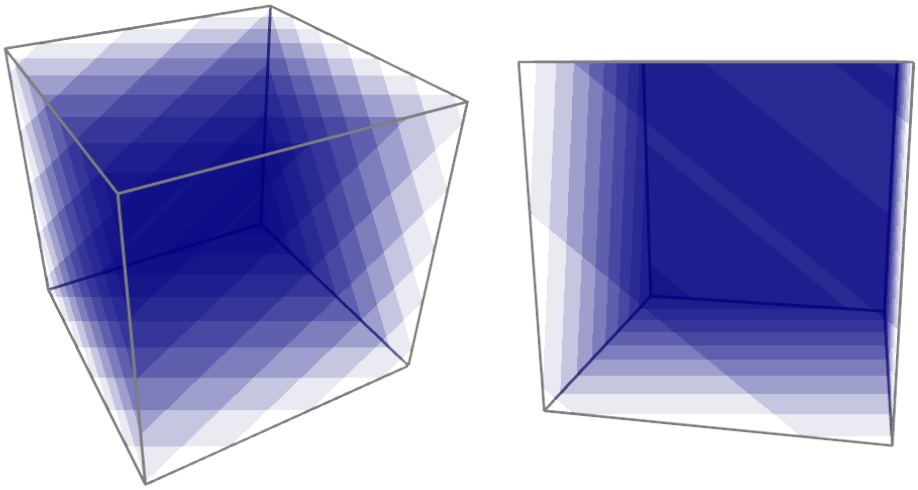
3D scéna (třída `GWindow`) je oddělena od zbytku aplikace, dědí metody a vlastnosti ze tříd `QWindow` a `QOpenGLFunctions`, obsahuje především OpenGL kontext a zachycuje události klávesnice a myši pro uživatelskou transformaci scény. Aplikace je kompatibilní s verzemi OpenGL 3.2, GLSL 1.5 a vyššími. Inicializace scény provádí načtení fragment a vertex shaderu z prostředků (resources.qrc), vytvoření textur a nastavení několika OpenGL příznaků. 3D textura je používána pro volumetrická data, 2D textury jsou pro multiplanární zobrazení a 1D textura slouží pro přenos barevného spektra do fragment shaderu (jinak také přenosová funkce).

Pro vykreslení datasetu musí scéna obdržet jeho velikost a rozměry simulace. Tyto hodnoty jsou ve scéně transformovány do rozsahu 0 až 1. Podle velikostí se vykreslí 3D rámeček ohraničující rozměry simulace nebo případně navíc menší rámeček znázorňující senzorovou masku. Vykreslení na sebe kolmých řezů se provádí na základě signálů z `MainWindow`, které doručí potřebná data a nahrají je do 2D textur.

Volumetrické vykreslování musí mít k dispozici data celého datasetu. Scéna tedy dostane signál s odkazem na objekt datasetu (`HDF5File::HDF5Dataset *`), inicializuje si 3D texturu a spustí postupné nahrávání dat do této textury. Ke čtení dat využívá, podobně jak je tomu u načítání řezů, vlákno (`HDF5ReadingThread`) s tím rozdílem, že nepřečte pouze jeden blok dat, ale způsobem postupného načítání automaticky vypočítaných bloků získá celý dataset. Objemové vykreslování je implementováno pomocí texturovaných polygonů, které jsou orientovány do pohledu kamery, viz Obrázek 6.5. Počet takto vykreslených polygonů ovlivňuje kvalitu zobrazení. Pro zachování stejné průhlednosti celého objemu při změně počtu polygonů je aplikován přepočet podle vzorce:

$$A \quad = \quad 1 - (1 - A_0)^{\frac{s_0}{s}}, \tag{6.1}$$

kde A je výsledná hodnota průhlednosti, A_0 je vstupní hodnota průhlednosti, s_0 je referenční počet polygonů a s je aktuální počet polygonů [16]. Princip vykreslování polygonů je následující: Vygeneruje se určitý počet rovin kolmých do pohledu uživatele, na tyto roviny je aplikována transformační matice posunutí v ose z a změnu měřítka podle rozměrů datasetu. Tato matice je předána přes vertex shader až do fragment shaderu a v něm se pomocí ní získají odpovídající souřadnice v 3D textuře. Ve fragment shaderu jsou roviny také ořezávány do polygonů, které nepřesahují hranice 3D rámečku datasetu.



Obrázek 6.5: Objemové vykreslování pomocí polygonů zarovnaných do pohledu kamery (Viewport-Aligned Slices rendering).

Mimo jiné, je ve fragment shaderu přístupná také 1D textura (`uniform sampler1D uColormapSampler`), ve které je vykreslené barevné spektrum (přenosová funkce). Hodnota fragmentu datového typu `float` v řezech (2D textury) a v objemu (3D textura) je potom indexem do 1D textury, ze které je získána výsledná barva.

Implementován je také export 3D scény jako png obrázku. Cílem bylo ukládat obrázek s průhledným pozadím, který vypadá stejně jako ve scéně. Pomocí funkce `glReadPixels` jsou získány hodnoty pixelů, z nich je vytvořena OpenCV matice a ta je uložena na disk jako obrázek. Aby se uložený obraz co nejvíce podobal scéně, musí být nastaveny parametry `glBlendFuncSeparate(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA, GL_ONE, GL_ONE_MINUS_SRC_ALPHA)`. V případě samotného nastavení `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA` jsou hodnoty složky alpha příliš nízké.

Kapitola 7

Experimenty a výsledky

Hlavní otázkou, v oblasti předzpracování simulačních dat, je ideální nastavení velikosti chunků a dále parametrů knihovny HDF5, které by mohly zrychlit čtení dat. Při otevírání HDF5 souboru lze nastavit velikost tzv. Data sieve buffer a chunk cache.

Data sieve buffer má výchozí velikost 64 kB, je používán pouze pro souvisle uložené datasety a udává velikost dat, které se načítají najednou. Toto nastavení má význam pro malé datasety, ze kterých se snažíme získat různě rozmístěné malé podmnožiny dat. Účel bufferu je redukce počtu malých čtecích operací.

Jelikož se tato práce týká především obrovských dat, které jsou uloženy pomocí chunků, zajímá nás chunk cache. Tato pomocná paměť je využívána pro dočasné uchovávání již čtených nebo zapisovaných dat a tím redukuje počet přístupů na disk. Aktuální verze knihovny HDF5 zatím bohužel neumí nastavovat parametry cache automaticky a pro zlepšení výkonu aplikace musí být nastavovány manuálně. Každý dataset má svou vlastní chunk cache. Při otevření souboru lze nastavit totální velikost cache pro data v bajtech, počet prvků v této cache a tzv. Preemption policy (hodnoty od 0 do 1). Výchozí hodnoty jsou 521 prvků, 1048576 bajtů a hodnota Preemption policy je 0.75. Nevýhodou je, že nastavení cache po otevření souboru již nelze měnit a je společné pro celý soubor, tedy pro všechny datasety. Nelze tak například měnit velikost cache podle zvoleného datasetu.

V této kapitole jsou uvedeny experimenty s různým nastavením parametrů chunk cache a velikost chunků pro různě velké datasety. Cílem je najít co nejvhodnější nastavení.

7.1 Způsob testování

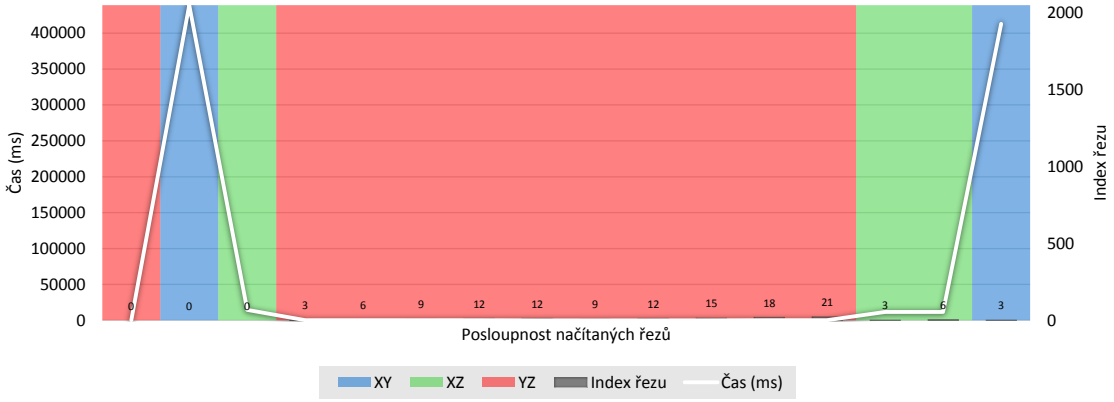
Testování rychlosti čtení dat probíhalo na datasetech o velikostech 256^3, 512^3, 1024^3 a $1536\times 1536 \times 2048$. Pro představu, velikost posledně uvedeného datasetu je přes 19 GB. Rychlosti čtení souvisle uložených datasetů jsou uvedeny na začátku návrhu konzolové aplikace (Kapitola 5, Tabulka 5.1) a z naměřených hodnot je vidět, že je souvislé rozložení pro větší datasety nepoužitelné. Každý dataset jsem tedy přeformátoval do podoby s velikostmi chunků 16^3, 32^3, 64^3 nebo 128^3 a pozoroval chování cache a rychlost čtení. Dále jsme zkoušel měnit nastavení chunk cache a sledoval změnu chování knihovny HDF5. Experimentování bylo prováděno na notebooku s konfigurací: 4 jádrový procesor AMD A10-5750M s frekvencí 2500MHz, operační paměť 8GB, 1TB 5400 RPM pevný disk, OS Windows 8.1. Tento počítač je mnohem výkonnější, než notebook, na kterém byly prováděny testy, jejichž výsledky jsou v Tabulce 5.1.

Na následně prezentovaných obrázcích jsou v grafech posloupnosti naměřených hodnot

při čtení řezů datasetu. Bílá křivka znázorňuje čas čtení řezů v milisekundách, barvy symbolizují roviny XY, XZ, YZ a šedé sloupce hodnoty indexu čteného řezu. V grafech odpovídá levá svislá osa času, pravá svislá osa indexům řezů a vodorovná osa obsahuje posloupnost načítaných řezů. Smyslem grafů je ukázat rozptyl hodnot času vzhledem k lokalitě načítaných řezů.

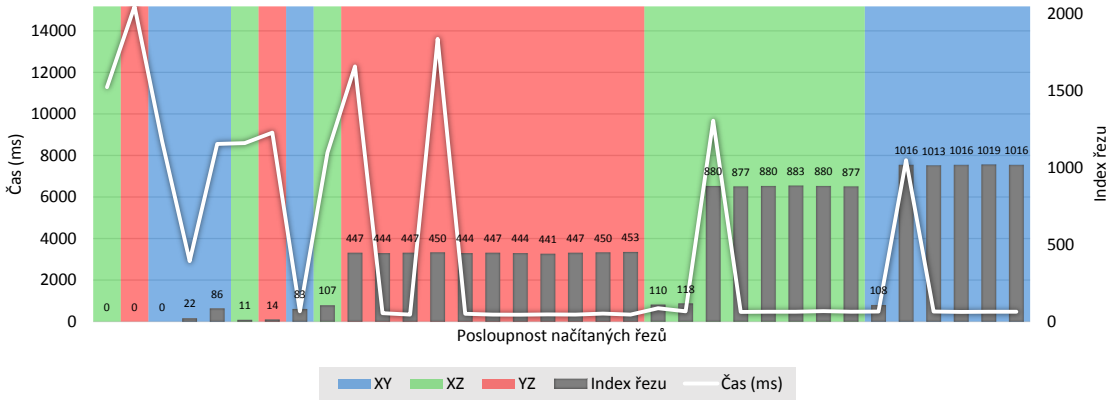
7.2 Dataset $1536 \times 1536 \times 2048$

U tohoto velkého datasetu byla aplikace chunků znatelná nejvíce. Na Obrázku 7.1 je posloupnost načtení několika řezů z původně uloženého datasetu (chunky $1 \times 1536 \times 2048$, tedy v rovině XY). Z obrázku je vidět, že čtení řezu v rovině kolmé k ose z je velmi rychlé (asi 10 až 200 ms), v rovině XZ je to kolem 15 sekund, ale v rovině YZ trvá načtení řezu přes 400 sekund, což je nepřijatelné.



Obrázek 7.1: Dataset $1536 \times 1536 \times 2048$, velikost chunku $1 \times 1536 \times 2048$.

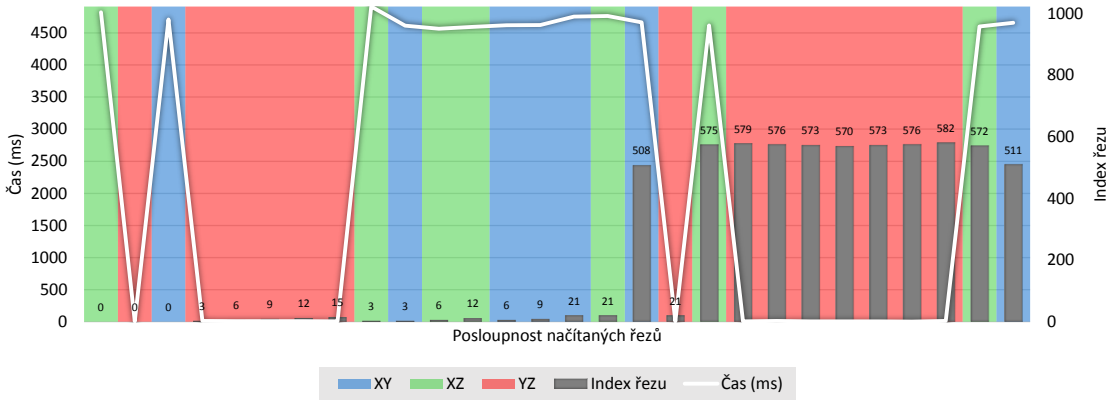
Pokud ovšem změníme rozložení datasetu, situace se podstatně zlepší. Na Obrázku 7.2 je čtení datasetu $1536 \times 1536 \times 2048$ s velikostí nastavených chunků 64^3. Z grafu můžeme vypozorovat, že v případech, kdy získáváme data z okolí aktuálně načteného řezu, trvá načtení ve všech 3 rovinách zhruba 300 až 500 ms a v extrémních případech, kdy je změna indexu řezu větší, trvá načtení od 8 do 16 sekund. Rychlosti čtení tedy nejsou ideální, ale vzhledem k původnímu rozložení dat a jejich velikosti jsou mnohokrát lepší. Z původních 400 sekund máme nyní 400 ms nebo v horším případě 16 sekund.



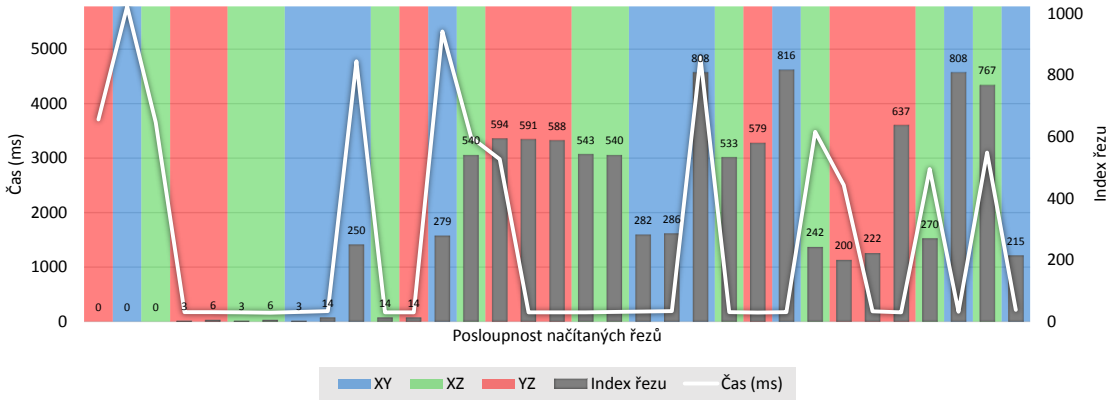
Obrázek 7.2: Dataset $1536 \times 1536 \times 2048$, velikost chunku 64^3.

7.3 Dataset 1024^3

Takto velký dataset bylo problematické číst na slabším notebooku, se kterým jsem prováděl první testy. Načtení roviny řezu YZ trvalo (podobně jako u předchozího datasetu) velmi dlouho, asi kolem 200 sekund. Na novém počítači byla situace odlišná. Na Obrázku 7.3 je graf čtení původně uloženého datasetu, ze kterého je vidět, že v rovině kolmé k ose z je čtení opět velice rychlé (7 až 10 ms), ale v dalších směrech se čas pohybuje vždy kolem 4 až 5 sekund. Rozdíl rychlostí na starém a novém PC je způsoben pravděpodobně velikostí RAM, typem operačního systému a procesorem. Disk je používán stále tentýž. Změnou chunků na velikost 64^3 se rychlosti čtení tohoto datasetu opět zlepšily. Z Obrázku 7.4 je vidět, že v blízkém okolí (podle velikosti chunku asi 64 řezů) aktuálního řezu je čtení rychlejší (asi 170 ms), než v případě souvislého uložení.



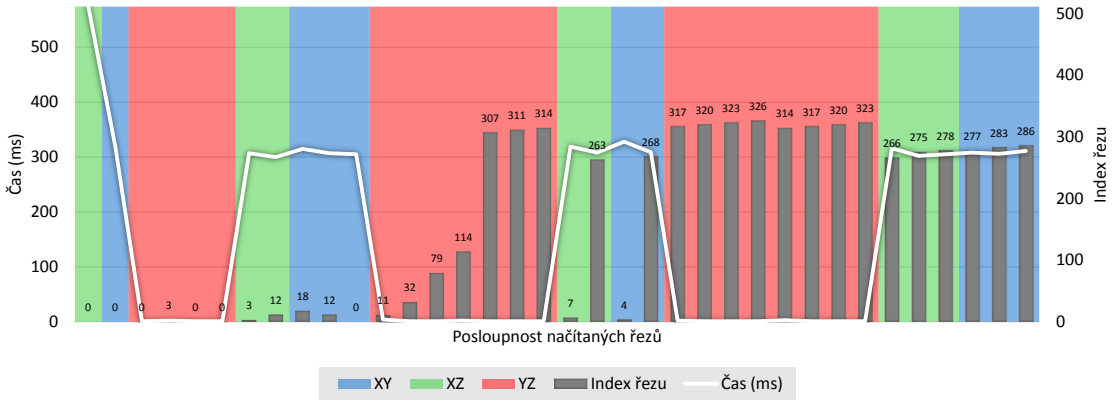
Obrázek 7.3: Dataset 1024^3, velikost chunku $1 \times 1024 \times 1024$.



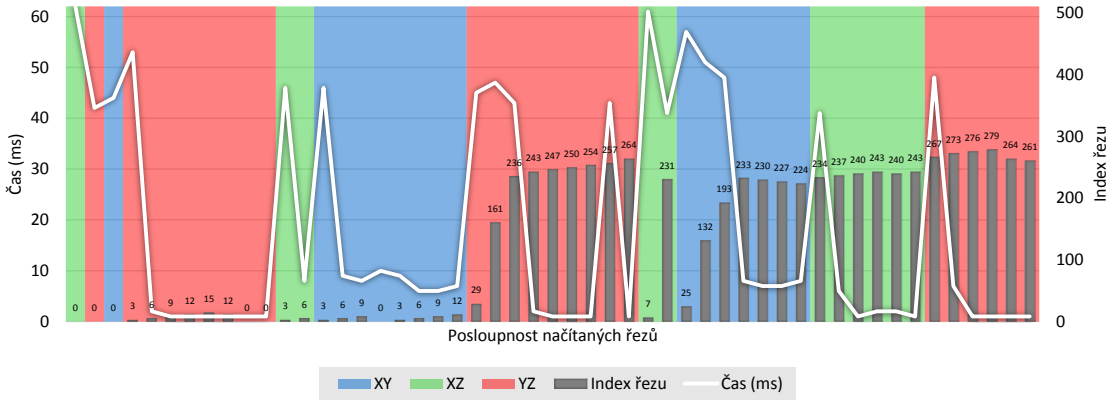
Obrázek 7.4: Dataset 1024^3, velikost chunku 64^3.

7.4 Dataset 512^3

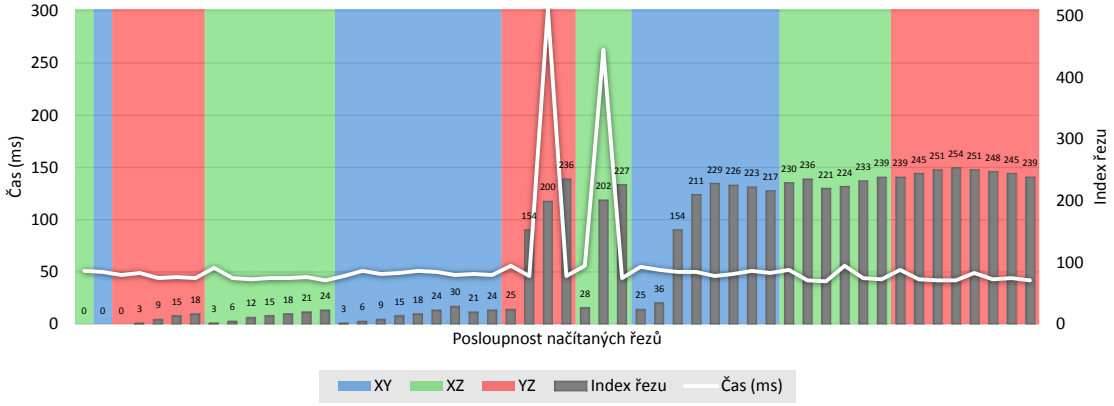
U tohoto datasetu se kromě modifikace chunků projevila také změna nastavení chunk cache. Graf původního čtení je na Obrázku 7.5. Podobně jako u předchozího datasetu je pomalejší čtení v rovinách XZ a YZ. Čas 300 ms není kritický, nicméně se mi jej podařilo optimalizovat na hodnoty 1 až 10 ms, v extrémních případech kolem 50 ms, viz Obrázek 7.6. Původní chunk cache jsem změnil z velikosti 1 MB na 130MB. Graf rychlostí čtení s původní velikostí cache je na Obrázku 7.7. S malou velikostí cache a chunky o velikosti 64^3 byla rychlost čtení kolem 50 ms, v některých případech i kolem 300 ms.



Obrázek 7.5: Dataset 512^3, velikost chunku $1 \times 512 \times 512$.



Obrázek 7.6: Dataset 512^3, velikost chunku 64^3, velikost chunk cache 130 MB.



Obrázek 7.7: Dataset 512^3, velikost chunku 64^3, původní velikost chunk cache 1 MB.

7.5 Shrnutí výsledků

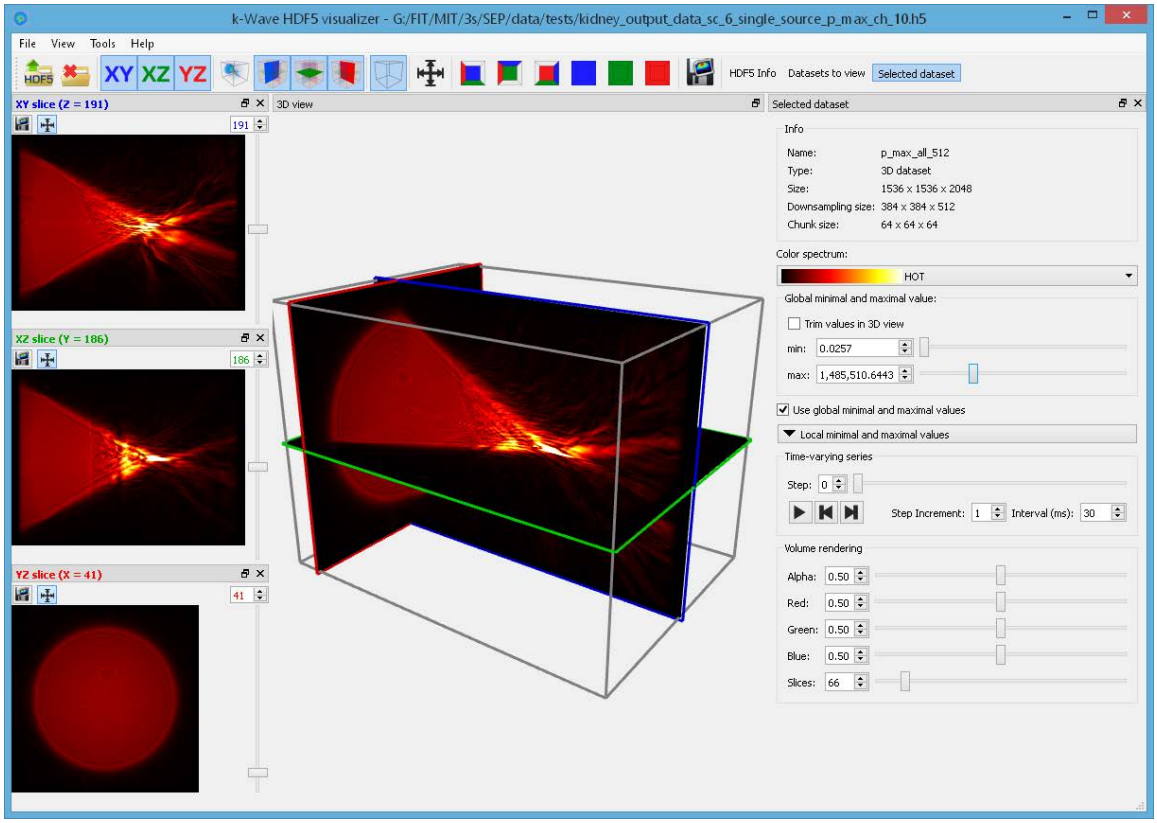
Uvedl jsem zde pouze nezajímavější výsledky experimetů. Samozřejmě jsem testoval i jiné velikosti chunků, ale jako nejlepší volba se jevila velikost 64^3. Větší velikosti chunků se projevovaly delšími extrémními časy, menší velikosti zase více zdržovaly přístupy na disk. Nastavení větší chunk cache ovlivnilo především datasety menších rozměrů, jelikož by její velikost měla být ideálně taková, aby obsahovala všechny chunky aktuálně čtených řezů. U datasetů větších než 512^3 se větší cache moc neprojevovala, ale její úplná deaktivace výsledky zhoršila. Usoudil jsem, že nechám velikost cache nastavenou na hodnotu 130MB, která není příliš velká a zároveň zrychlí čtení menších datasetů. Z odlišnosti experimentů v různých dobách běhu počítače dále usuzuji, že nějaký typ dočasného uchovávání dat kromě HDF5 provádí také operační systém, jehož chování by se dalo zkoumat podrobněji. Rychlosti čtení velmi závisí na celkové výkonnosti počítače a nejen pouze na parametrech pevného disku. Optimalizaci rychlosti čtení datasetů považuji za užitečnou, a to především proto, že s původním rozložením dat nabývaly rychlosti čtení až 100 krát větších hodnot. Naskytla se mi také možnost testovat rychlost čtení z SSD disku. Čtení velkých datasetů bylo asi 10x rychlejší než v případě klasického disku.

V aplikaci lze tedy pracovat s řezy datasetů v originálním rozlišení s tím, že vždy nebude změna řezu tolik efektivní. Nicméně v případě zkoumání okolí řezu je rychlost čtení

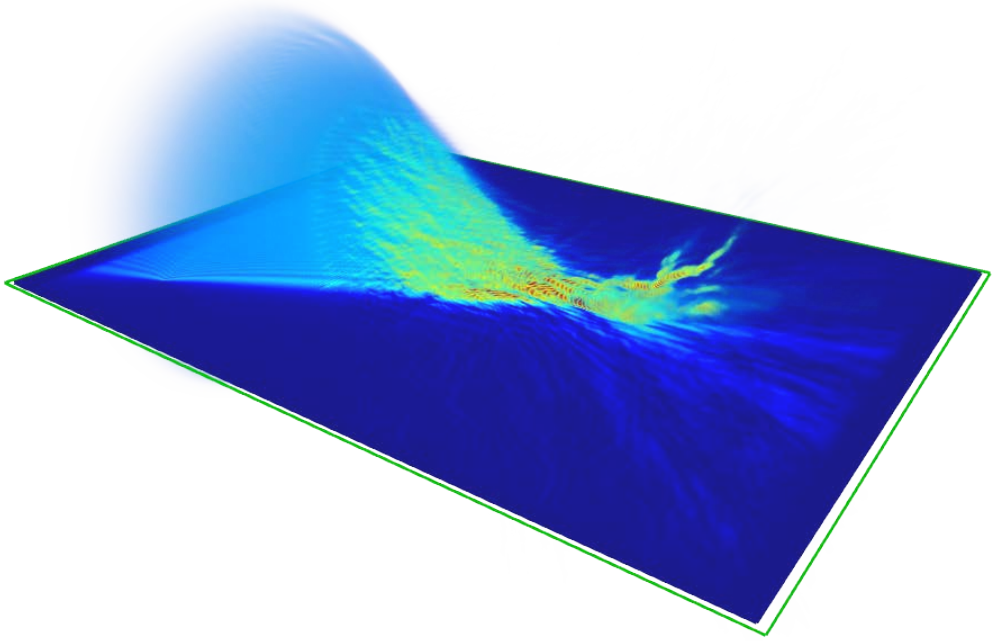
dostatečná. Pro volumetrické vykreslování je vhodné používat datasety do velikosti cca 512^3. Větší velikosti jsou pro běžnou grafickou kartu problematické, protože je nutné načíst do její paměti celý dataset (např. dataset 1024^3 zabírá přes 4 GB). Velké datasety je tedy vhodné pro objemové vykreslování nejdříve podvzorkovat pomocí konzolové aplikace.

7.6 Ukázky výsledné aplikace

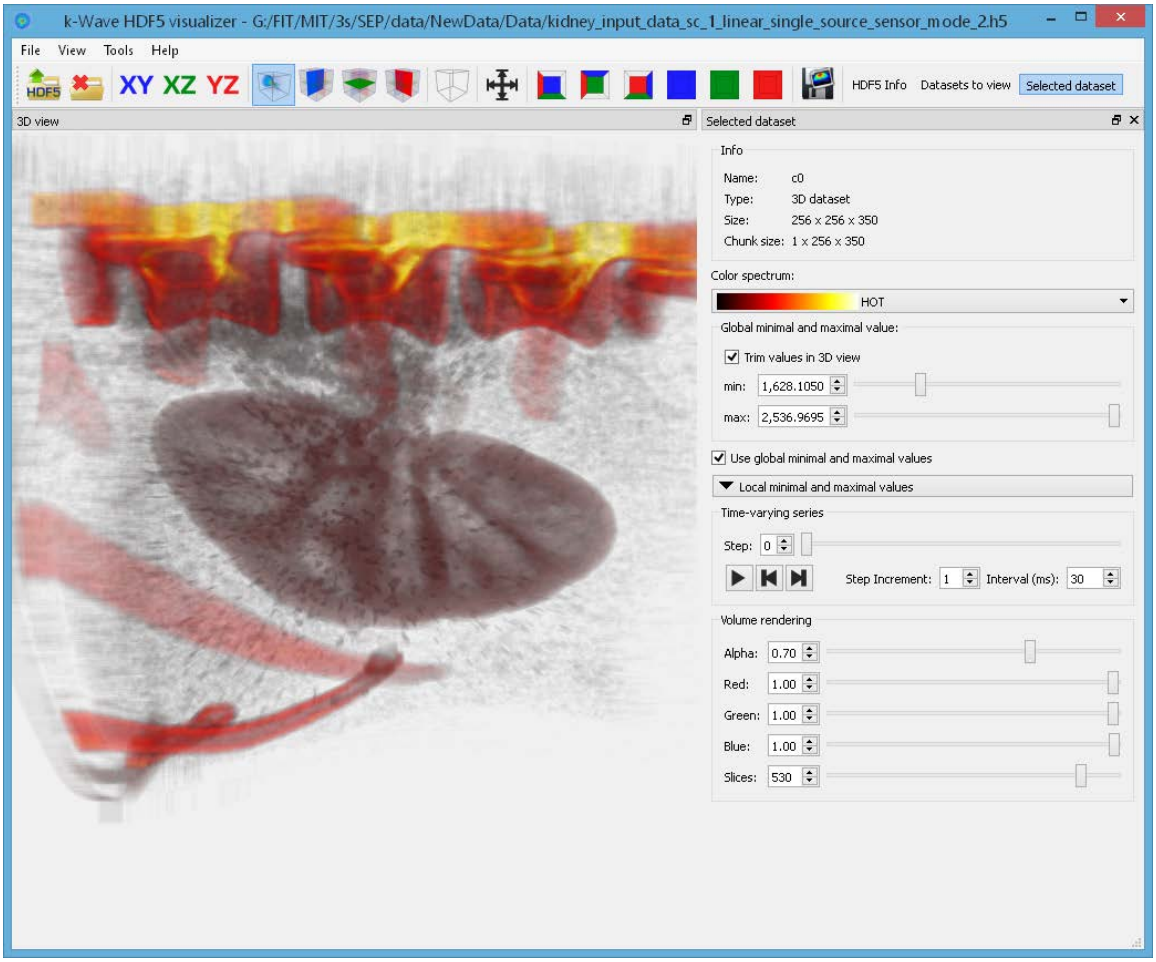
Na následujících obrázcích jsou klíčové snímky výsledné aplikace. Jedná se především o kompletní náhled aplikace (Obrázek 7.8), kombinace multiplanárního a volumetrického zobrazení (Obrázek 7.9) a zobrazení datasetů zadaných seznzorovou maskou (Obrázek 7.11).



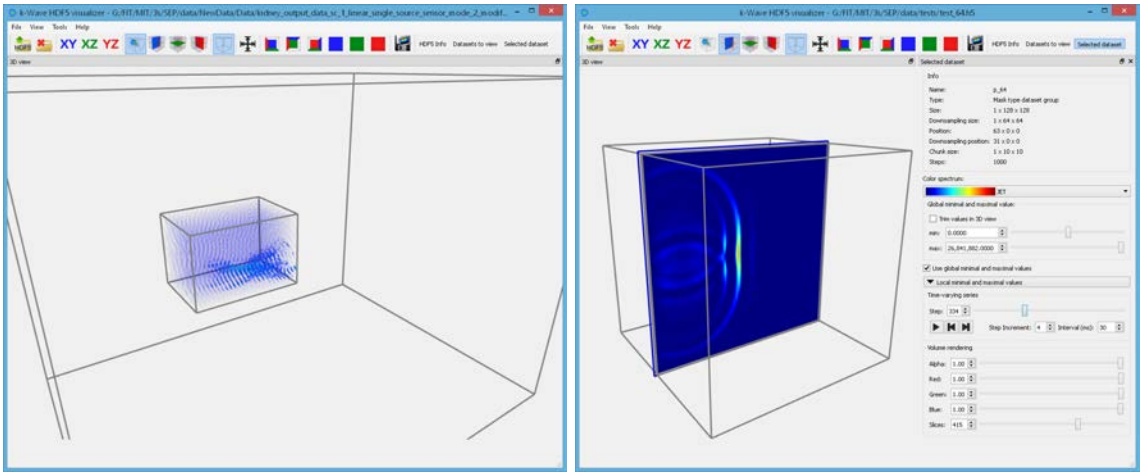
Obrázek 7.8: Náhled kompletní grafické aplikace. Multiplanární zobrazení řezů. Zobrazený dataset obsahuje hodnoty maximálního tlaku získané během ultrazvukové simulace. Rozlišení datasetu: $384 \times 384 \times 512$.



Obrázek 7.9: Ukázka kombinace volumetrického a planárního zobrazení. Snímek je exportován z aplikace.



Obrázek 7.10: Volumetrické zobrazení. Jedná se o dataset obsahující hodnoty CT - část lidského těla s ledvinou. Kvalita - 530 vykreslených texturovaných polygonů.



Obrázek 7.11: Ukázky zobrazení datasetů definovaných senzorovou maskou. Jedná se o časové kroky simulace.

Kapitola 8

Závěr

Cílem této práce bylo zpracování a vizualizace specifických simulačních dat. Tato data jsou specifická právě proto, že nabývají obrovských rozměrů (desítky GB) a jsou uložena pomocí knihovny HDF5 různými způsoby - 3D datasety, senzorová maska, časové série simulace. Data obsahují například simulaci šíření utrazvuku v lidském těle. Cílem těchto simulací je co nejpřesněji napodobit chování ultrazvuku v reálném prostředí, konkrétněji třeba chování tzv. fokusovaného ultrazvuku o vysoké intenzitě, který se používá k likvidaci rakovinotvorných nádorů. Aby bylo možné výsledky simulace efektivně zpracovávat a dále zlepšovat je nutná interaktivní vizualizace těchto dat.

V rámci této práce jsem nastudoval formát výstupních simulačních dat, analyzoval datasety formátu HDF5, které jsou vhodné pro vizualizaci. Podrobněji jsem studoval knihovnu HDF5 a její možnosti. Dále jsem se zaměřil na existující nástroje k vizualizaci medicínských dat, které jsem vyzkoušel, a také na metody, které jsou pro vizualizaci používány. Vytvořil jsem návrh konzolové aplikace pro předzpracování dat ve formátu HDF5. Hlavním smyslem této aplikace je postupné načítání dat, jejich podvzorkování, změna blokového rozložení dat a především převedení dat uložených pomocí senzorové masky do použitelné podoby. Změna rozložení (chunked layout) je způsob, kterým se výrazně zrychlilo čtení axiálních, koronálních a sagitálních řezů rozměrných datasetů, které bylo v případě souvislého rozložení dat nepřijatelné. Poté jsem navrhl propracovanou aplikaci s grafickým uživatelských rozhraním pro vizualizaci HDF5 souborů především s možnostmi zobrazování řezů datasety, multiplanárního 3D zobrazení, volumetrického zobrazování a animace průběhu simulace. Navrhl jsem základní vlastnosti aplikací, které jsem také implementoval (QT, OpenGL, OpenCV), ale také vlastnosti pokročilé, které by bylo dobré do budoucna řešit. Ke konci práce jsem provedl experimentování v rámci zpracování datasetů a uvedl zajímavé výsledky.

Naměřené výsledky mě potěšily především proto, že jsem změnou blokového rozložení datasetů (velikost chunků) dokázal urychlit načtení některých dat i 100×. Implementovaná konzolová aplikace je podle mého názoru dobře použitelná především díky postupnému zpracování dat, oddělitelnosti od vizualizační aplikace a přenositelnosti. Grafická aplikace je implementována do podoby, kdy se dá celkem dobře používat ke zkoumání simulačních dat a myslím si, že by mohla být pro výzkumníky k-Wave velkým pomocníkem. 3D zobrazení dává uživatelům nový pohled na věc. Možností pokračování na tomto projektu je spousta, některé jsou uvedeny v návrhu aplikací. Myslím, že by bylo dobré řešit například paralelní předzpracování dat, generování videa ve vysokém rozlišení nebo třeba pokročilejší metody objemového zobrazování. Práce na projektu mě bavila a nebránil bych se budoucí spolupráci, protože má podle mého názoru k-Wave velký potenciál.

Literatura

[1] Treeby, B., Cox, B. a Jaros, J. *K-Wave: A MATLAB toolbox for the time domain simulation of acoustic wave fields: User Manual* [online]. [cit. 2014-05-19]. Dostupné na: <http://www.k-wave.org/manual/k-wave_user_manual_1.0.1.pdf>.

[2] Treeby, B. E. a Cox, B. T. K-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *Journal of biomedical optics*. 2010, roč. 15, č. 2.

[3] Treeby, B. E., Jaros, J., Rendell, A. P. et al. Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method. *The Journal of the Acoustical Society of America*. 2012, roč. 131, č. 6. S. 4324–4336.

[4] The HDF Group. *Hierarchical data format version 5*. 2000-2014. Dostupné na: <http://www.hdfgroup.org/HDF5>.

[5] The HDF Group. *HDF5 Wins 2002 R&D 100 Award* [online]. [cit. 2013-12-29]. Dostupné na: <http://www.hdfgroup.org/HDF5/RD100-2002/All_About_HDF5.pdf>.

[6] Soukup, T. *TECHNIKA — Počítače v medicíně ukládají, zobrazují, analyzují a léčí* [online]. [cit. 2013-01-09]. Dostupné na: <http://vtm.e15.cz/pocitace-v-medicine-ukladaji-zobrazuji-analyzuji-a-leci>.

[7] Pelikán, J. *Visualizace objemových dat* [online]. [cit. 2013-12-31]. Dostupné na: <http://cgg.mff.cuni.cz/~pepca/lectures/pdf/volume.pdf>.

[8] *The latest DICOM specification* [online]. [cit. 2013-01-09]. Dostupné na: <ftp://medical.nema.org/medical/dicom/2011/>.

[9] Tišnovský, P. *Zobrazení objemových dat v POV-Rayi - root.cz* [online]. [cit. 2013-12-31]. Dostupné na: <http://www.root.cz/clanky/zobrazeni-objemovych-dat-v-pov-rayi>.

[10] Drebin, R. A., Carpenter, L. a Hanrahan, P. Volume Rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. 22. vyd. New York, NY, USA: ACM, 1988. S. 65–74. SIGGRAPH '88. Dostupné na: <http://doi.acm.org/10.1145/54852.378484>. ISBN 0-89791-275-6.

[11] Engel, K., Hadwiger, M., Kniss, J. M. et al. Real-time Volume Graphics. In *ACM SIGGRAPH 2004 Course Notes*. New York, NY, USA: ACM, 2004. SIGGRAPH '04. Dostupné na: <http://doi.acm.org/10.1145/1103900.1103929>.

[12] Pawasauskas, J. *Volume Visualization With Ray Casting* [online]. únor 1997 [cit. 2013-01-09]. Dostupné na: <http://web.cs.wpi.edu/~matt/courses/cs563/talks/powwie/p1/ray-cast.htm>.

[13] Guthe, S., Wand, M., Gonser, J. et al. Interactive Rendering of Large Volume Data Sets. In *IEEE Visualization*. 2002. S. 53–60. Dostupné na: <http://dblp.uni-trier.de/db/conf/visualization/visualization2002.html#GutheWGS02>.

[14] Strengert, M., Magallón, M., Weiskopf, D. et al. Hierarchical Visualization and Compression of Large Volume Datasets Using GPU Clusters. In *Proceedings of the 5th Eurographics Conference on Parallel Graphics and Visualization*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004. S. 41–48. EG PGV'04. Dostupné na: <http://dx.doi.org/10.2312/EGPGV/EGPGV04/041-048>. ISBN 3-905673-11-8.

[15] Hassan, A. H., Fluke, C. J. a Barnes, D. G. A Distributed GPU-based Framework for real-time 3D Volume Rendering of Large Astronomical Data Cubes. *CoRR*. 2012, abs/1205.0282.

[16] Ikits, M., Kniss, J., Lefohn, A. et al. *Chapter 39, Volume Rendering Techniques*. [b.m.]: Addison Wesley, 2004. S. 667–692.

Příloha A

Obsah CD

Na přiloženém disku se nachází následující soubory a adresáře:

- `/bin/` - Složka obsahuje spustitelné aplikace `k-wave-h5-processing.exe` (konzolová aplikace) a `k-wave-visualizer.exe` (grafická aplikace) a potřebné dynamické knihovny (QT, Microsoft Visual C++ Runtime Libraries x64). Aplikace jsou přeloženy na systému Windows 8 64-bit, spustitelnost a funkčnost byla ověřena i na systému Windows 7.
- `/images/` - Tato složka obsahuje obrázky vyprodukované aplikací a screenshoty aplikace.
- `/src/gui/gui/*` - Zdrojové kódy grafické aplikace. Pro překlad je využívána také knihovna hdf5file, jejíž zdrojové kódy se nachází ve složce `/src/k-wave-h5-processing/k-wave-h5-processing/hdf5file/`. Nachází se zde projektový soubor `gui.pro` s definicemi cest k OpenCV a hdf5file.
- `/src/k-wave-h5-processing/k-wave-h5-processing/*` - V tomto adresáři je projektový soubor pro konzolovou aplikaci `k-wave-h5-processing.pro`.
- `/src/k-wave-h5-processing/k-wave-h5-processing/hdf5file/` - Zdrojové kódy knihovny hdf5file. Soubor `hdf5file.pri` obsahuje definici cest k HDF5 C++ knihovně.
- `/src/k-wave-h5-processing/k-wave-h5-processing/main/` - Zdrojový kód pro konzolovou aplikaci a projektový soubor `main.pro`, který obsahuje definici cest ke knihovnám OpenCV a hdf5file.
- `/tex/*` - Zdrojové soubory technické zprávy diplomové práce.
- `/poster.pdf` - Plakát prezentující diplomovou práci.
- `/projekt.pdf` - Tato technická zpráva v pdf.

Příloha B

Manuál

B.1 Přeložení zdrojových kódů

Pro překlad je využíván multiplatformní nástroj `qmake`[1]. Tento nástroj automaticky generuje soubory Makefile pro platformu, na které je prováděn překlad. Pro vývoj aplikací je vhodné používat Qt Creator, který automaticky vygeneruje pomocí qmake Makefile a následně aplikaci přeloží. Při použití příkazové řádky např. na systému linux stačí provést v umístění projektového souboru příkaz `qmake` a poté `make`.

k-wave-h5-processing
Projektový soubor konzolové aplikace (`k-wave-h5-processing.pro`) se nachází ve složce `/src/k-wave-h5-processing/k-wave-h5-processing/`. Pro překlad konzolové aplikace je nutné nastavit správné cesty (`INCLUDEPATH a LIBS`) ke knihovně HDF5 v souboru `/src/k-wave-h5-processing/hdf5file/hdf5file.pri` a ke knihovně OpenCV v souboru `/src/k-wave-h5-processing/main/main.pro`.

k-wave-visualizer
Projektový soubor grafické aplikace je umístěn v adresáři `/src/gui/gui/`. V tomto souboru (`gui.pro`) je nutné nastavit správné cesty ke knihovně OpenCV a dále ke statické knihovně hdf5file, která je vytvořena při překladu konzolové aplikace. Při zachování adresářové struktury by měly být cesty k hdf5file nastaveny správně. Pro překlad grafické aplikace je vyžadována verze Qt 5.1 a vyšší.

B.2 Použití aplikací

Použití konzolové aplikace je popsáno v nápovědě při spuštění. Aplikace `k-wave-h5-processing.exe` je pro systém Windows přeložena s potřebnými knihovnami staticky, nepotřebuje tedy žádné pomocné soubory `*.dll`.

Po spuštění grafické aplikace se zobrazí hlavní okno a pro výpis pomocných informací se na pozadí spustí konzole. Dokovatelné panely lze vyjmout z hlavního okna a měnit jejich velikost. **Nástrojový panel** obsahuje zleva tlačítka pro:

- otevření HDF5 souboru,
- zavření souboru,

[1] http://qt-project.org/doc/qt-5/qmake-manual.html

- zobrazení/skrytí panelů pro řezy,
- zapnutí/vypnutí volume renderingu,
- zobrazení/skrytí řezů v 3D zobrazení,
- zobrazení/skrytí 3D rámečku,
- roztáhnutí datasetu definovaného senzorovou maskou,
- 6 tlačítek pro zarovnání 3D scény do hlavních pohledů,
- tlačítko pro export obrázku z 3D scény
- a tlačítka pro zobrazení/skrytí panelů pro: seznam datasetů, informace o HDF5 souboru a nastavení zobrazení datasetu.

Každý panel pro **2D zobrazení** řezu obsahuje:

- slider pro změnu polohy řezu,
- tlačítka pro přepínaní mezi originální a přizpůsobenou velikostí,
- a tlačítko pro export obrázku.

V **nastavení zobrazení** vybraného datasetu lze:

- měnit spektrum pro mapování barev,
- upravovat maximální či minimální hodnoty, podle kterých je mapování prováděno,
- zapnout nebo vypnout zobrazení hodnot přesahujících zadaný rozsah
- a pro jednotlivé řezy lze povolit lokální změny minimálních a maximálních hodnot.

Panel dále obsahuje **ovládání animace** pro časově proměnné série datasetů, kde lze:

- spustit nebo zastavit animaci,
- přeskočit na začátek či konec série,
- ručně měnit aktuální krok
- a měnit časový interval a inkrement pro změnu kroků při animaci.

Volume rendering lze ovlivnit změnou:

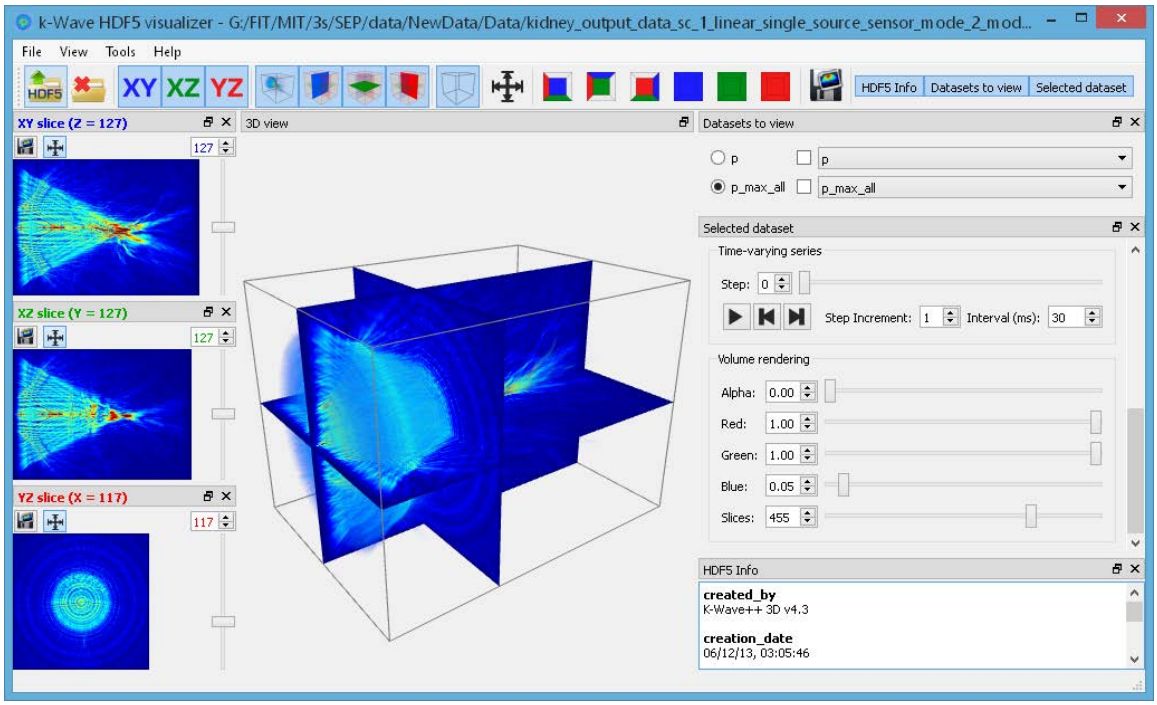
- průhlednosti (Alpha) zarovnaných řezů,
- zastoupení jednotlivých barevných složek (Red, Green a Blue)
- a počtu vykreslených řezů (Slices) - kvalita volumetrického zobrazení.

Ovládání 3D scény

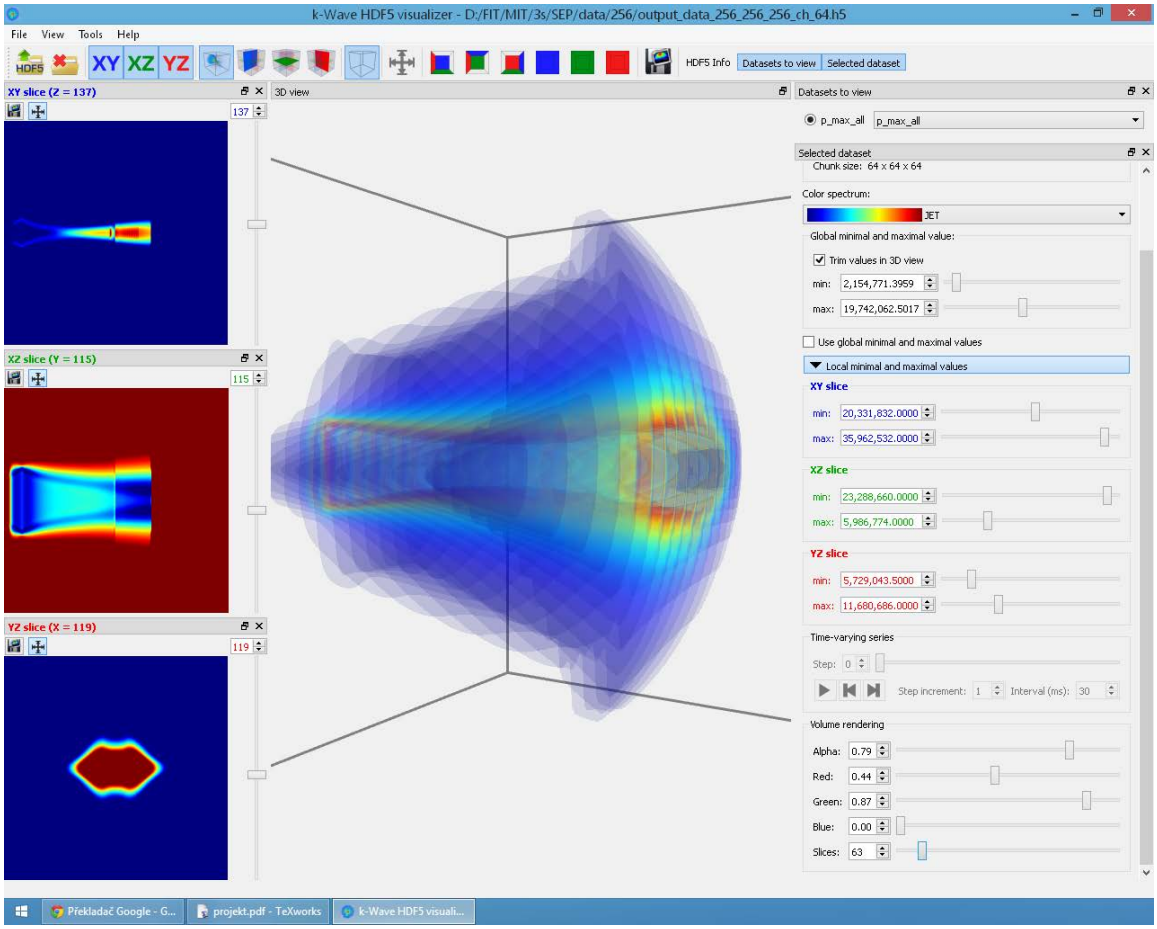
- Rotaxe scény - Držením levého tlačítka myši a pohybem myší.
- Přiblížení/oddálení scény - Rotací kolečka myši.
- Posun scény - Držením pravého tlačítka myši a pohybem myší.
- Reset pozice scény - Kliknutí prostředního tlačítka (kolečka) myši.

Příloha C

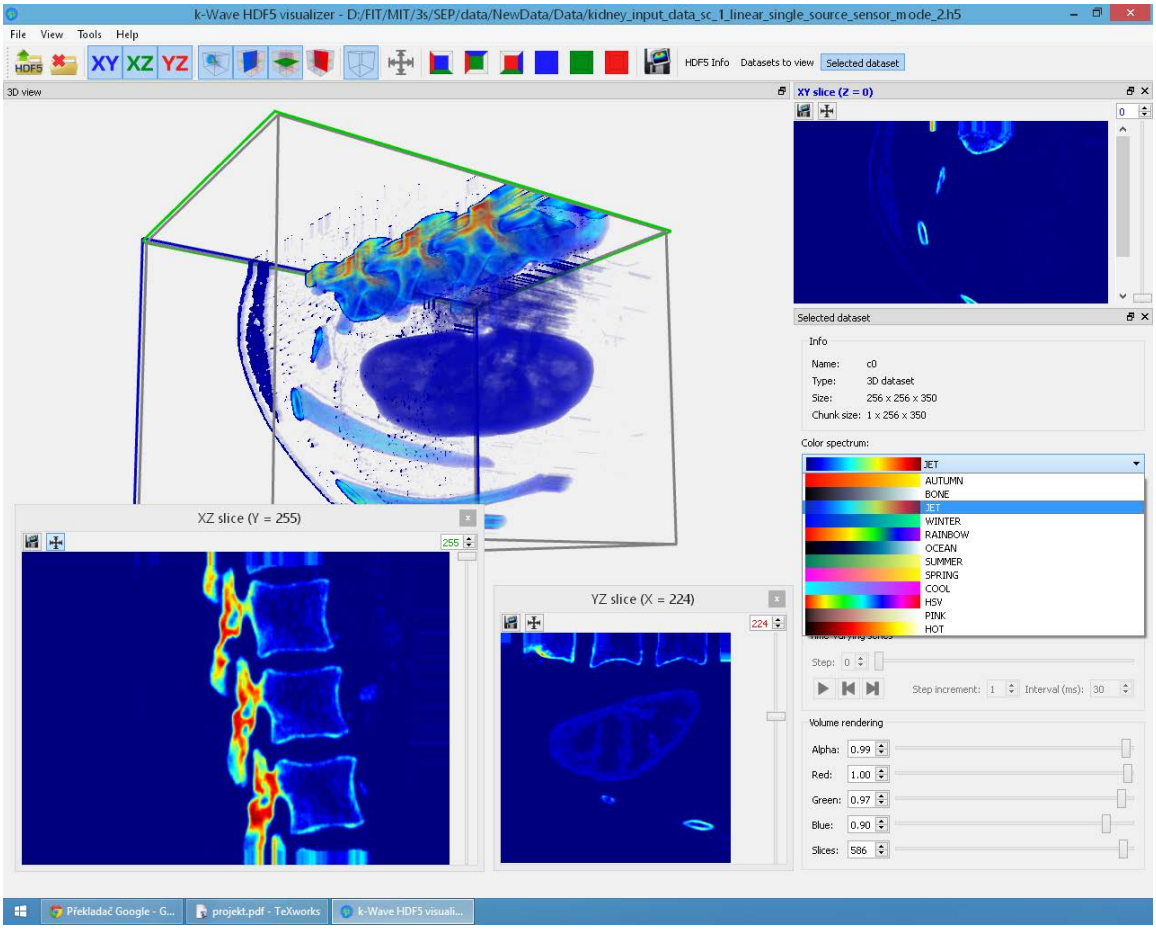
Obrázky



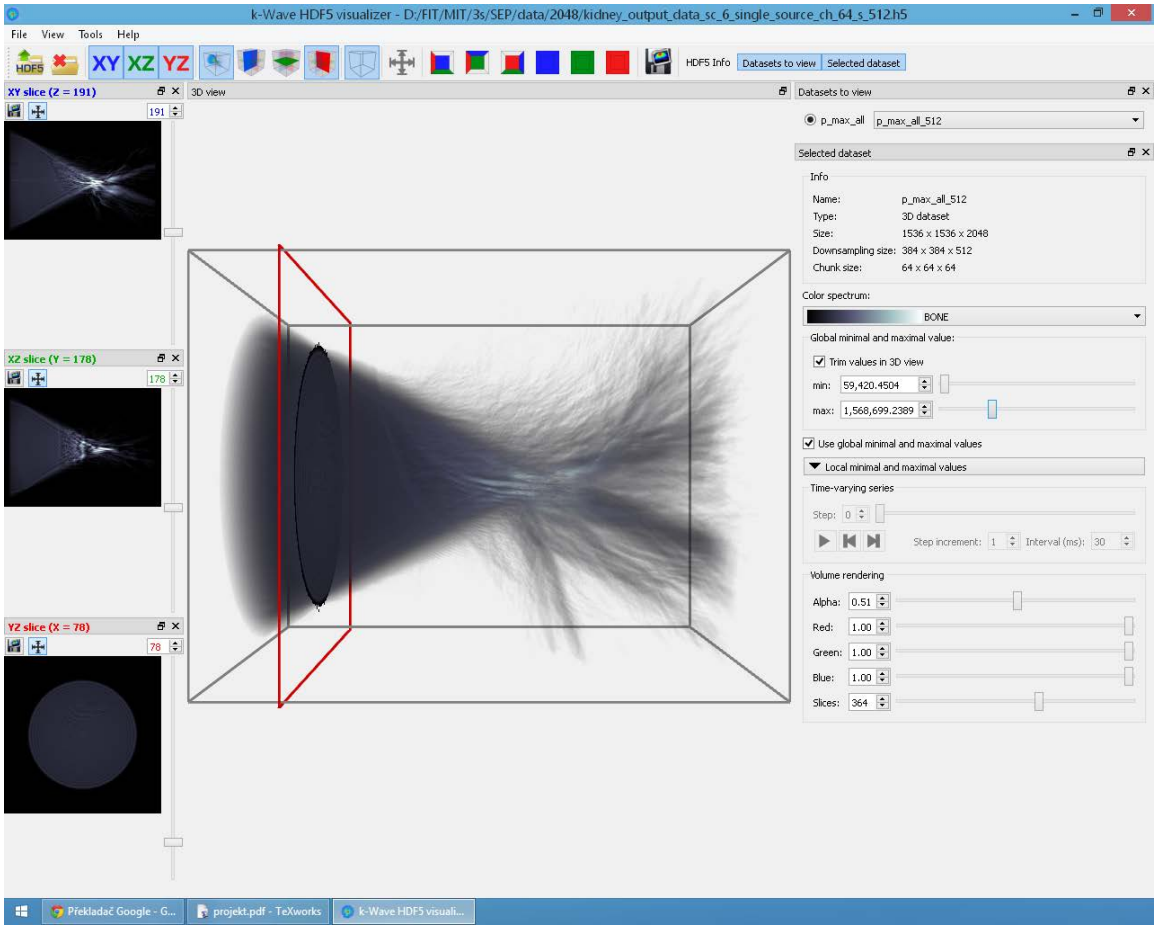
Obrázek C.1: Dataset $256 \times 256 \times 350$, maximální tlak - simulace HIFU.



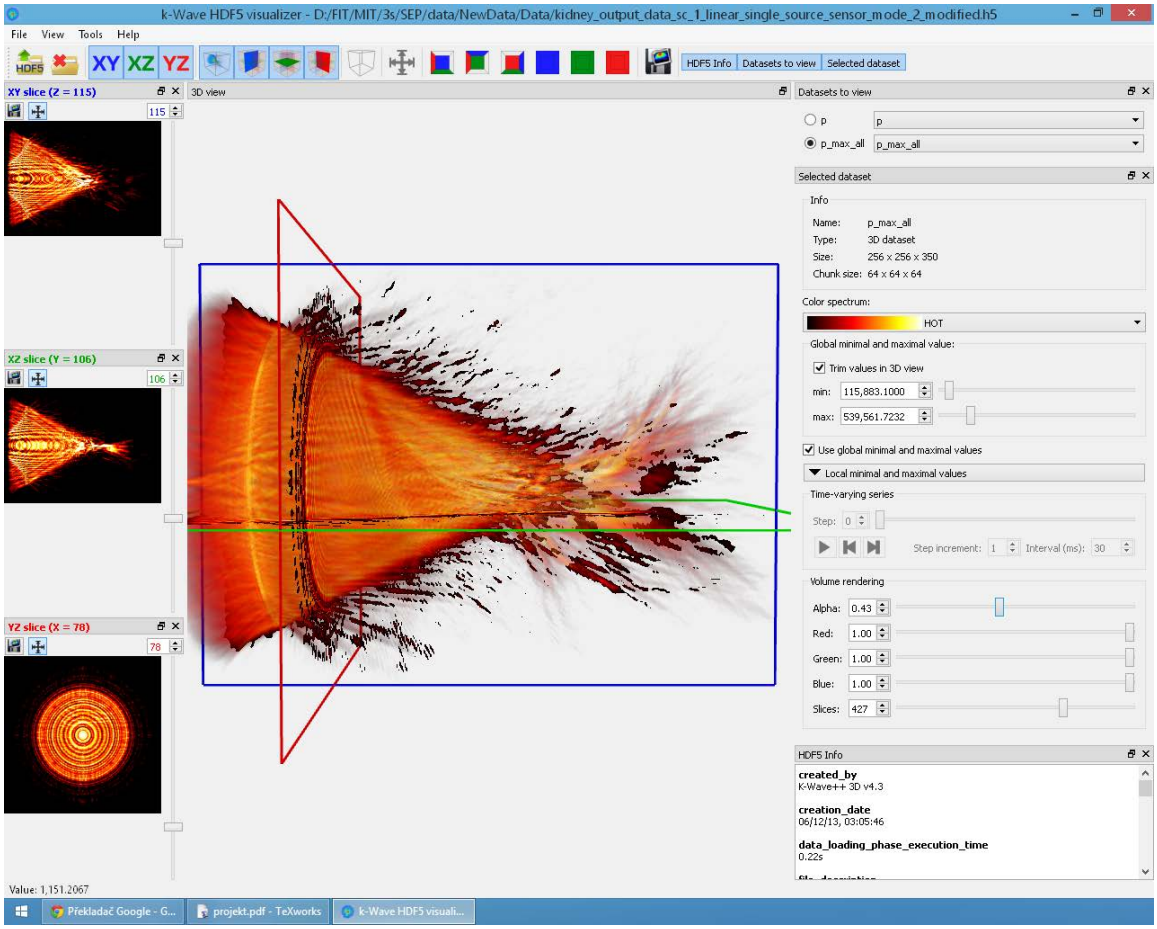
Obrázek C.2: Demonstrace nižší kvality volumetrického zobrazení a lokální změny min/max hodnot.



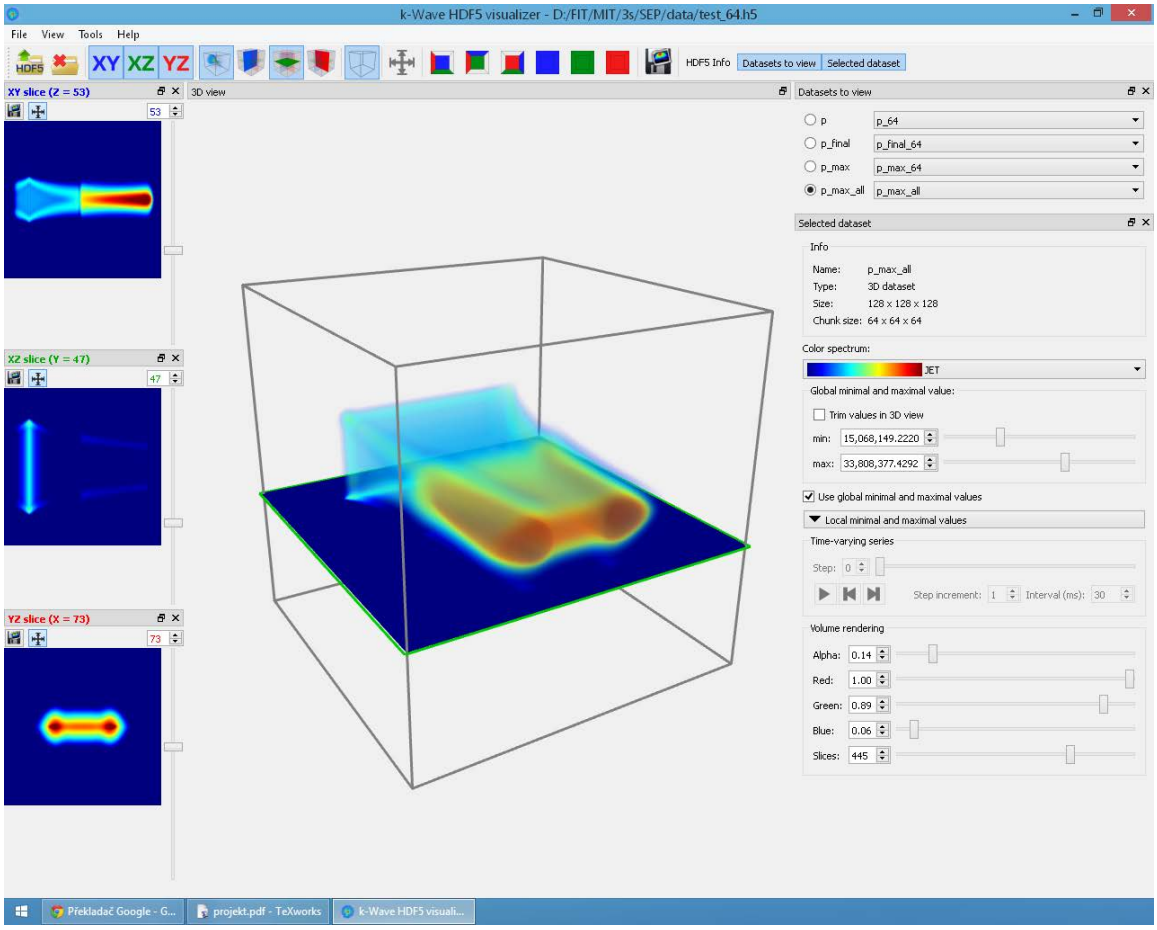
Obrázek C.3: Různé možnosti GUI, barevná spektra.



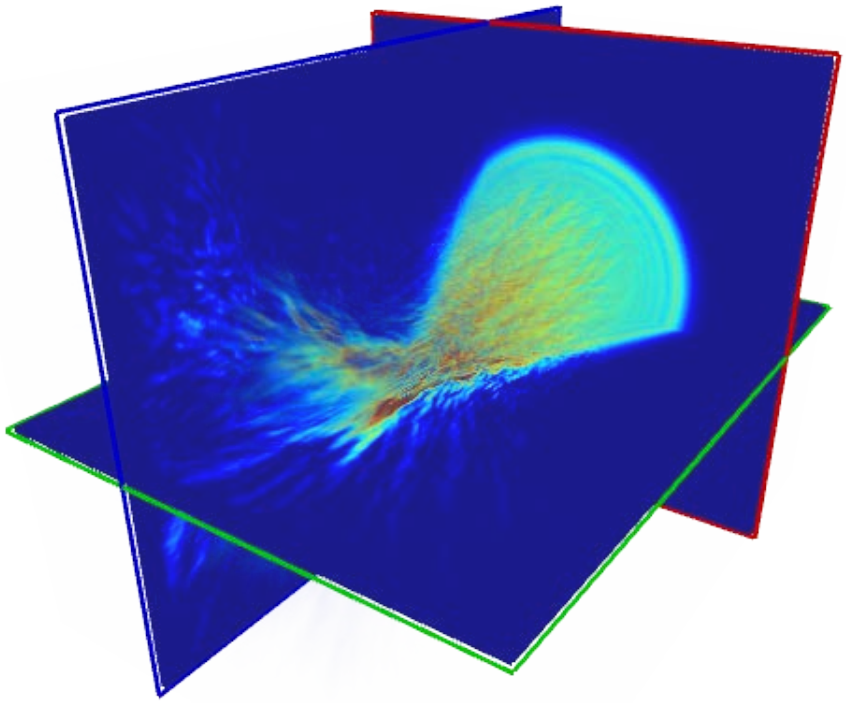
Obrázek C.4: Dataset $384 \times 384 \times 512$, simulace HIFU.



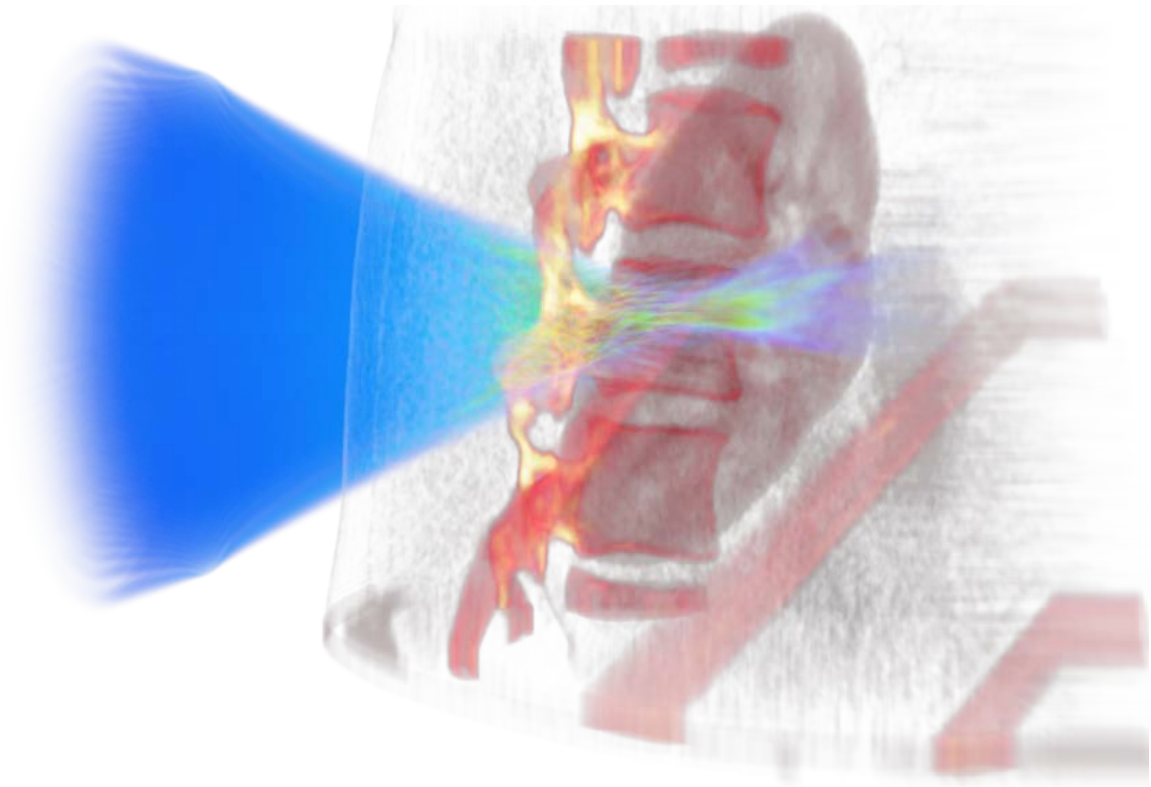
Obrázek C.5: Demonstrace ořezávání hodnot.



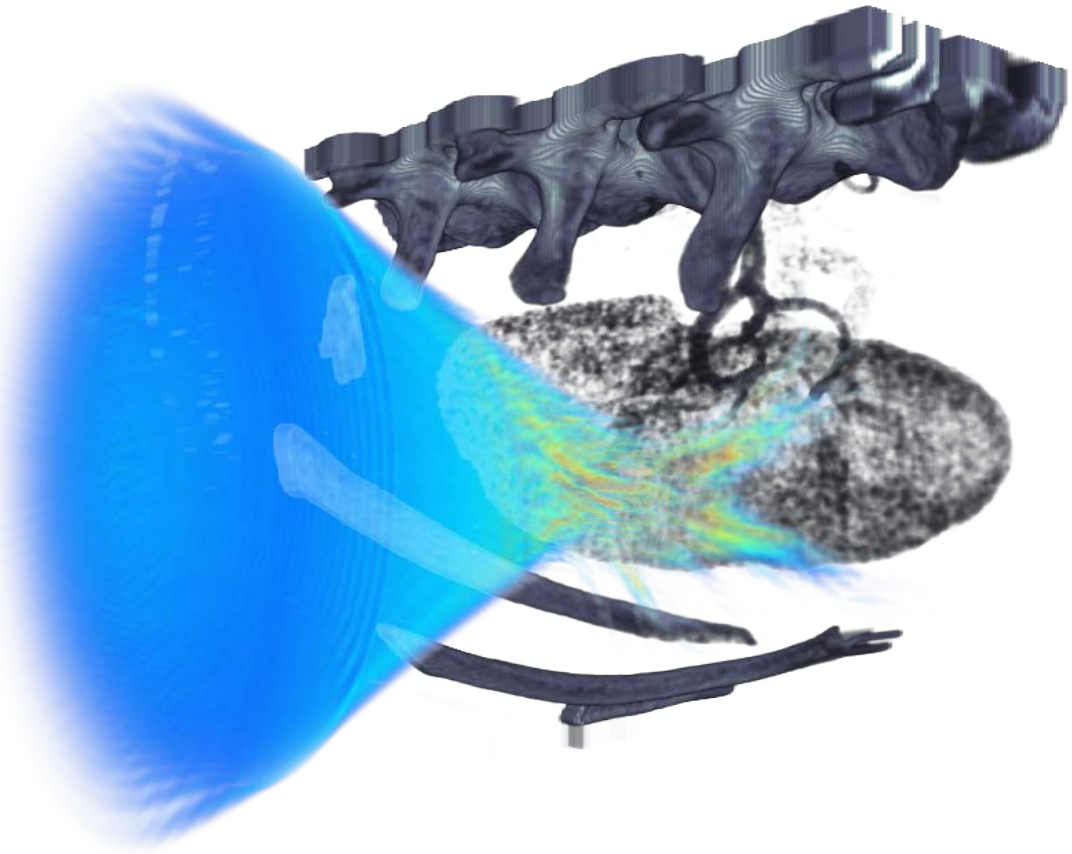
Obrázek C.6: Testovací data.



Obrázek C.7: Obrázek exportovaný z aplikace.



Obrázek C.8: Obrázky exportované z aplikace a zkombinované v externím programu.



Obrázek C.9: Obrázky exportované z aplikace a zkombinované v externím programu.

Příloha D

Plakát