Spletno vizualizacijsko ogrodje z možnostjo oddaljenega sodelovanja

Primož Lavrič, Ciril Bohak, Matija Marolt

Univerza v Ljubljani,
Fakulteta za računalništvo in informatiko
primoz.lavric@gmail.com,{ciril.bohak, matija.marolt}@fri.uni-lj.si

Web based visualisation framework with remote collaboration support

In this paper we present a web based visualization framework which primarily focuses on the visualization of medical 3D mesh and volumetric data. This framework offers visualization capabilities like 3D mesh rendering, scene navigation and indirect volume rendering via volume to mesh conversion implemented with computationally efficient Marching cubes algorithm. As the volume to mesh conversion with Marching cubes presents one of the main functionalities of this framework, the performance evaluation of four different implementations (Java, C++, Javascript and ASM.js) is presented. The framework also allows users to remotely collaborate by sharing their visualisations with other users of this framework in real time.

1 Uvod

Z razvojem novih tehnologij se v vsakdanje življenje vse bolj vključujejo 3D vizualizacije podatkov s številnih področij, kot so na primer medicina, strojništvo in geodezija. V medicini dobra in pravilna 3D vizualizacija zajetih podatkov v veliko primerih močno pripomore k določanju pravilne končne diagnoze pacientov, saj omogoča natančnejši vpogled v notranjost človeškega telesa in s tem zmanjšuje potrebo po invazivnejših posegih. Volumetrični medicinski podatki so v večini primerov zajeti s pomočjo tehnik volumetričnega skeniranja s tehnikami računske tomografije (angl. computed tomography - CT) [1, 2], slikanja s pomočjo magnetne resonance (angl. magnetic resonance imaging - MRI) [3] in ultrazvoka (angl. ultrasound - US) [4]. Tako zajeti podatki so večinoma precej obsežni in posledično njihova vizualizacija predstavlja svojevrsten izziv. V večini primerov se izvaja na namenski, dovolj zmogljivi strojni in programski opremi. Takšna oprema je draga in stacionarna ter zdravnikom ne omogoča podajanja diagnoze in priprave na poseg na daljavo, kar otežuje pridobivanje drugega mnenja ali mnenja specialista na oddaljeni lokaciji.

Načine vizualizacije ločimo na posredne in neposredne. Pri posrednem upodabljanju podatke večinoma najprej pretvorimo v predstavitev s 3D mrežnimi modeli [5, 6, 7], ki jih nato izrišemo [8]. Pri neposrednem upodabljanju podatke vizualiziramo brez predhodne pretvorbe. Pri tem se uporabljajo tehnike volumetričnega upodabljanja [9, 10] in njihove sodobne izpeljanke [11].

Za namene vizualizacije volumetričnih podatkov je bilo razvitih več ogrodij in aplikacij. Exposure Renderer [12] in SimVascular[1] sta namenjeni lokalni vizualizaciji volumetričnih podatkov in podpirata napredne osvetlitvene tehnike. ParaView[2] omogoča paralelno obravnavo in vizualiziranje obsežnih podatkov z uporabo namenskih strežnikov. Predstavljena orodja ne podpirajo širokega nabora platform in so omejena na uporabo na namiznih računalnikih ali prenosnikih. ParaViewWeb [3] omogoča uporabo preko spletnega brskalnika s pomočjo oddaljenega upodabljanja, a je prav zato njena uporaba nekoliko omejena zaradi manjše odzivnosti. Predstavljeno orodje sicer deluje na veliko platformah, a enako kot prejšnja ne omogoča oddaljenega sodelovanja med uporabniki.

V članku predstavimo ogrodje Med3D, razvito z namenom vizualizacije volumetričnih podatkov neposredno v spletnem brskalniku z možnostjo oddaljenega sodelovanja med uporabniki. Ogrodje delno temelji na predhodno razvitem ogrodju Neck Veins [13]. Spletna implementacija poveča doseg uporabnikov in omogoča uporabo tudi na mobilnih napravah. Uporabniku omogoča lokalno obdelavo in vizualizacijo podatkov v okviru zmožnosti naprave, na kateri deluje, in s tem omogoča visoko stopnjo interaktivnosti. V primeru zahtevnejših vizualizacij in slabših uporabniških naprav (mobilne naprave) ogrodje omogoča izvajanje zahtevnejših izračunov na strežniku, kar lahko zajema zgolj obdelavo ali pretvorbo podatkov ali celostno oddaljeno upodabljanje. Oddaljeno sodelovanje obsega možnost deljenja pogleda z oddaljenim uporabnikom in s tem njegov vpogled v dejansko stanje pacienta.

V drugem poglavju predstavimo razvito vizualizacijsko ogrodje. V tretjem poglavju podamo ovrednotenje ogrodja, performančno analizo in primerjavo z drugimi implementacijami. V četrtem poglavju predstavimo implementacijo oddaljenega sodelovanja med uporabniki. V zadnjem poglavju podamo zaključke in smernice za možne razširitve ogrodja.

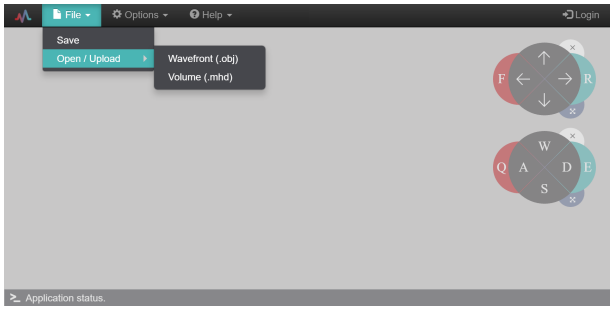
[1] http://simvascular.github.io/
[2] http://www.paraview.org/overview/
[3] http://paraviewweb.kitware.com/

2 Vizualizacijsko ogrodje

Ogrodje smo razvili kot spletno aplikacijo, povezano z zalednim delom ogrodja na strežniku. Ogrodje omogoča odpiranje in vizualizacijo podatkov v obliki poligonske mreže ter vizualizacijo volumetričnih podatkov, predstavljenih s tridimenzionalnimi skalarnimi polji, ki jih ogrodje pretvori v mrežo poligonov. Pretvorbo izvede z algoritmom Marching cubes [5], izris pa poteka strojno pospešeno z uporabo grafične knjižnice WebGL. Ogrodje omogoča navigiranje po prostoru s tipkovnico in miško ali z uporabo grafičnega vmesnika.

2.1 Vizualizacijsko ogrodje na spletu

Razvoja v obliki spletne aplikacije smo se lotili, ker je spletna platforma najbolj razširjena in dostopna tako na osebnih računalnikih kot mobilnih napravah. Vizualizacijsko ogrodje je primarno namenjeno vizualizaciji 3D medicinskih podatkov, lahko pa prikazuje tudi ostale 3D podatke tako v obliki poligonske mreže kot tudi volumetrične podatke, ki jih pretvori v poligonsko mrežo. Spletna platforma omogoča boljšo dostopnost aplikacije čim širšemu krogu uporabnikov ter poenostavi možnost uporabe oddaljenega upodabljanja in oddaljenega sodelovanja med uporabniki. Na sliki 1 je prikazan vmesnik razvitega ogrodja, oblikovan z namenom uporabe na različnih napravah z različnimi zaslonskimi ločljivostmi.



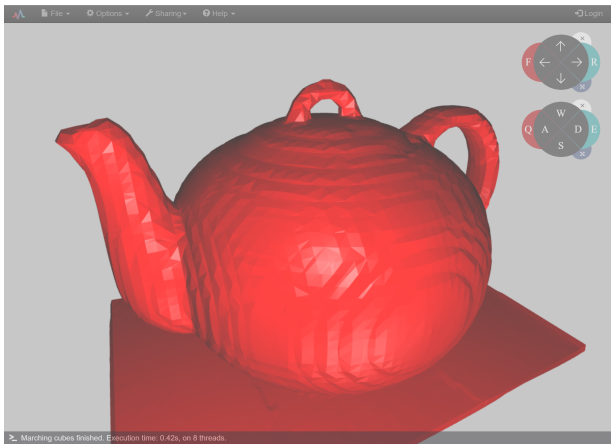
Slika 1: Slika prikazuje uporabniški vmesnik ogrodja Med3D.

2.2 Pretvorba volumetričnih podatkov v mrežni model

Večina neobdelanih volumetričnih medicinskih podatkov je podana v obliki tridimenzionalnega skalarnega polja. Kot smo omenili že v uvodu, se za upodabljanje takšnih podatkov uporabljajo neposredne in posredne tehnike upodabljanja. Zaradi želje po hitrejšem in računsko manj zahtevnem upodabljanju, kot tudi zaradi želje po hitrejšem prenosu podatkov do uporabnikov, se najpogosteje uporablja pretvorba volumetričnih podatkov v mrežni model.

Za namen pretvorbe podatkov iz volumetrične oblike v mrežni model smo v ogrodju implementirali metodo Marching cubes. Za pretvorbo obstajajo tudi sodobnejše metode [14], a smo se za uporabo Marching cubes odločili zaradi relativno nizke časovne in prostorske zahtevnosti in možnosti visoke stopnje paralelizacije [15]. To je pomembno zaradi dejstva, da se lahko algoritem poganja tudi v spletnem brskalniku na strani uporabnika, ki ponuja omejen dostop do sistemskih virov, še posebej pomnilnika. Pretvorba volumetričnih podatkov je pomembna tudi zaradi oddaljenega sodelovanja, saj je velikost poligonske mreže precej manjša od tridimenzionalnega skalarnega polja in posledično omogoča hitrejšo sinhronizacijo podatkov med uporabniki.

Pri implementaciji algoritma Marching cubes smo testirali hitrostno učinkovitost štirih implementacij. Za ogrodje sta pomembni implementaciji v programskem jeziku Javascript in njegovo podmnožico ASM.js[4]. ASM.js predstavlja nizkonivojsko podmnožico programskega jezika Javascript in dosega visoke pohitritve v primerjavi z osnovnimi implementacijami. Ti implementaciji sta pomembnejši zaradi možnosti njune uporabe tako v spletnem brskalniku, kot tudi na strežniku zalednega dela sistema. Implementacija zalednega dela sistema temelji na uporabi ogrodja Node.js[5]. Poleg omenjenih implementacij, ki temeljita na jeziku Javascipt, smo za primerjavo pri performančni analizi dodali tudi implementaciji v programskih jezikih C++ in Java. Rezultat pretvorbe volumetričnih podatkov z uporabo algoritma Marching cubes je prikazan na sliki 2.



Slika 2: Slika prikazuje rezultat Marching cubes algoritma, izvedenega nad tridimenzionalnim skalarnim poljem Utah teapot objekta.

Implementaciji algoritma Marching cubes v programskem jeziku Javascript sta realizirani z uporabo aplikacijskega vmesnika Web Workers[6], ki omogoča asinhrono in paralelno izvajanje algoritma v brskalniku in na strežniku. Aplikacijski vmesnik pri tem izkorišča paralelnost centralne procesne enote in za vsako nit sproži lastno opravilo. Komunikacija s posameznim opravilom, ki se izvaja preko vmesnika Web Workers, poteka preko standardiziranega dogodkovno vodenega sporočilnega sistema. Poleg prenašanja sporočil med kopijami objektov omogoča tudi prenos objektov s prenosom njihovega lastništva. Posledično velikih objektov (skalarno polje)

[4] http://asmjs.org/
[5] https://nodejs.org/en/
[6] https://www.w3.org/TR/workers/

ni potrebno podvajati, ampak se na posamezno opravilo vmesnika Web Workers prenese zgolj njihovo lastništvo. S tem se izognemo dodatnemu delu in povečani porabi pomnilnika, do katere pride v primeru podvajanja objektov. Pomanjkanje pomnilnika lahko zaradi omejitev brskalnikov povzroči neodzivnost in nepredvideno prekinitev izvajanja.

Zaradi počasnejšega delovanja algoritma Marching cubes, implementiranega v programskem jeziku Javascript, kjer lahko pretvorba velikih skalarnih polj v mrežni model traja tudi več kot minuto, smo se odločili za uporabo programske knjižnice ASM.js. Programska knjižnica ASM.js dosega do desetkratno pohitritev izvajanja. Takšne pohitritve dosega zaradi uporabe prevajalnika AOT (angl. ahead-of-time), ki prevede in optimizira programsko kodo ob zagonu spletne aplikacije. Posledično se je potrebno odpovedati dinamičnosti programskega jezika Javascript in uporabiti zapis v obliki SSA (angl. static single assignment form). Zaradi nepraktičnosti takšnega zapisa se za generiranje programske kode ASM.js uporablja namenske prevajalnike. Le-ti ustrezno programsko kodo generirajo iz drugega nižjenivojskega programskega jezika npr. C, C++. V našem primeru smo za osnovo uporabili programsko kodo zapisano v programskem jeziku C++ in jo prevedli v obliko, primerno za uporabo s programsko knjižnico ASM.js, z uporabo prevajalnika Cheerp[7].

Glavni povod za izbiro prevajalnika Cheerp je pomnilniška organizacija, uporabljena v prevedeni programski kodi. Prevedena koda uporablja osnovno Javascript pomnilniško organizacijo, zato podatkov pred obdelavo ni potrebno dodatno pretvarjati oz. prenašati. To je bistvena prednost pred uporabo nekaterih drugih prevajalnikov, npr. prevajalnika Emscripten[8], ki uporablja lastno pomnilniško organizacijo, emulirano na pomnilniški organizaciji Javascripta.

3 Performančno ovrednotenje

Del našega dela predstavlja tudi performančna analiza pretvorbe volumetričnih podatkov v mrežne modele z algoritmom Marching cubes. V preformančno analizo smo vključili štiri zgoraj opisane implementacije: Java, C++, Javascript in ASM.js. Testirali smo jih na sistemu z Intel i7 6700K (4.2 GHZ) štirijedernim procesorem, 16 GB pomnilnika in operacijskim sistemom Windows 10. Spletni implementaciji Javascript in ASM.js smo testirali v 64-bitnemu brskalniku Chrome[9] (različica 51). Kot vhodne podatke smo uporabili enakostranične volumne v treh razsežnostih (128^3, 256^3 in 512^3), generirane z enačbo enodelnega hiperboloida $vol_{xyz} = x^2 + y^2 - z^2 - 25$, in realni primer vratnih žil, velikosti $512 \times 512 \times 390$. Realni primer smo testirali s tremi vrednostmi praga: nizko, srednjo in visoko, pri katerih dobimo mrežne modele s približno 20, 5 in 0,5 milijona trikotnikov. Vse omenjene implementacije omogočajo tudi izkoriščanje vzporednega izvajanja, zato smo poleg zaporednega testa v eni niti hitrost izmerili tudi pri uporabi dveh in štirih niti. Vsaka konfiguracaija je bila ovrednotena s povprečnim rezultatom desetih iteracij. Dobljen povprečni rezultat pa smo ustrezno zaokrožili glede na standardni odklon.

Implementacije, ki omogočajo prevajalniško optimizacijo (C++, ASM.js), so bile za testiranje prevedene z optimizacijo za čim hitrejše izvajanje. Poleg C++ in ASM.js pa tudi Java omogoča optimizacije, vendar šele med izvajanjem. Te realizira s pomočjo sprotnega prevajalnika, ki pogosto izvedeno vmesno kodo prevede v domorodno obliko. Zaradi tega smo pri testiranju implementacije v Javi to najprej "ogreli", tako da smo pred merjenjem časa izvedli dve iteraciji algoritma, ki nista bili upoštevani v rezultatih.

Rezultati ovrednotenja so prikazani v tabeli 1. Kot bi pričakovali, je iz rezultatov razvidno, da je v vseh primerih najhitrejša implementacija v programskem jeziku C++. Pri umetno generiranih podatkih se ji pri večjih velikostih povsem približa implementacija v programskem jeziku Java. Kljub temu, da se omenjenima implementacijama pričakovano ne približa nobena implementacija v programskem jeziku Javascript, je iz rezultatov razvidno, da je uporaba programske knjižnice ASM.js smiselna, saj doseže na umetno generiranih podatkih tudi več kot osemkratno pohitritev, na izbranem primeru realnih podatkov pa do 1,5-kratno pohitritev izvajanja.

4 Oddaljeno sodelovanje

Pri vizualizaciji podatkov se velikokrat pojavi potreba po delitvi in skupni interpretaciji vizualizacij z oddaljenimi osebami. Na primeru medicine to pogosto pomeni pridobivanje mnenja oddaljenih specialistov. Delitev podatkov, lokalno vizualiziranje ter interpretacija so pogosto precej neučinkoviti tako z vidika časovne potratnosti kot tudi z vidika prenosa informacij med uporabniki. Ta problem smo v predstavljanem vizualizacijskem ogrodju rešili z implementacijo oddaljenega sodelovanja.

Implementacija oddaljenega sodelovanja omogoča delitev vizualiziranih podatkov in realnočasovno sinhroniziranje prikaza podatkov (scene) in pogleda kamere med uporabniki. Ti uporabniki lahko z ustreznimi dovoljenji tudi prevzamejo nadzor nad kamero in podatki. Tako implementirano sodelovanje omogoča bistveno hitrejšo interpretacijo vizualizacije podatkov in prenos informacij ter znanja med uporabniki.

4.1 Sinhronizacija scene med uporabniki

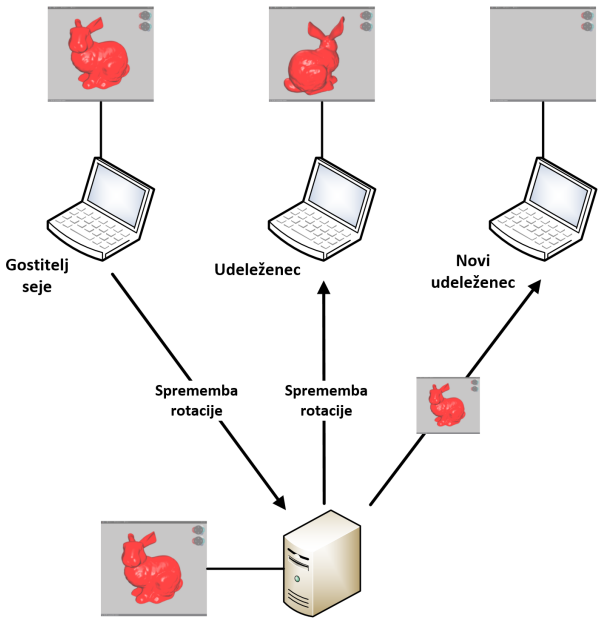
Ko uporabnik vzpostavi želeno sceno, lahko ta prikaz prične deliti z ostalimi uporabniki vizualizacijskega ogrodja. Ob inicializaciji se scena in njeno trenutno stanje zapiše v obliki formata JSON in se preko protokola WebSocket prenese na strežnik. Preko strežnika poteka tudi vsa nadaljnja komunikacija. Strežnik ob prejemu scene ustvari novo sejo, v katero se lahko preko protokola WebSocket včlanijo uporabniki vizualizacijskega ogrodja in tako pridobijo sceno ter začnejo prejemati njene nadaljnje posodobitve. Ko gostitelj seje svojo sceno sinhronizira s strežnikom, lahko začne pošiljati posodobitve v obliki sprememb deljenih parametrov objektov v sceni.

[7] http://leaningtech.com/cheerp/
[8] https://github.com/kripken/emscripten
[9] https://www.google.com/chrome/

Tabela 1: Rezultati performančne analize izvajanja algoritma Marching cubes na različnih podatkih pri različnih stopnjah vzporednega računanja.

Generirani podatki					Volumen vratnih žil				
Vol. Size	**C++**	**Java**	**Javascript**	**ASM.js**	**Prag**	**C++**	**Java**	**Javascript**	**ASM.js**
1 nit					**1 nit**				
128^3	0,02 s	0,03 s	0,66 s	0,16 s	Nizek	0,80 s	1,50 s	22,20 s	15,30 s
256^3	0,11 s	0,21 s	4,85 s	0,98 s	Srednji	0,57 s	0,91 s	20,70 s	15,20 s
512^3	0,79 s	1,32 s	38,00 s	4,63 s	Visok	0,52 s	0,85 s	20,20 s	15,10 s
2 niti					**2 niti**				
128^3	0,01 s	0,03 s	0,38 s	0,11s	Nizek	0,47 s	0,70 s	12,30 s	8,90 s
256^3	0,06 s	0,09 s	2,70 s	0,61s	Srednji	0,30 s	0,46 s	11,50 s	8,70 s
512^3	0,41 s	0,76 s	21,80 s	3,95s	Visok	0,28 s	0,44 s	11,20 s	8,70 s
4 niti					**4 niti**				
128^3	0,01 s	0,01 s	0,26 s	0,08 s	Nizek	0,34 s	0,42 s	8,70 s	5,80 s
256^3	0,04 s	0,05 s	1,60 s	0,42 s	Srednji	0,19 s	0,29 s	7,50 s	5,80 s
512^3	0,27 s	0,47 s	12,50 s	2,42 s	Visok	0,17 s	0,26 s	7,40 s	5,50 s

Tako lahko uporabnik prenese spremembe transformacij objektov, spremembe geometrije in druge podatke. Minimalni čas med posodobitvami se nastavi dinamično in ga je mogoče prilagajati glede na kvaliteto povezave med gostiteljem in strežnikom. Prejeto sceno strežnik lokalno shrani in jo posodablja s posodobitvami gostitelja seje. S tem gostitelja seje razbremenimo razpošiljanja celotne scene novim udeležencem, saj lahko novi udeleženec pridobi trenutno sceno in njeno zadnje stanje neposredno iz strežnika.



Slika 3: Slika prikazuje shemo komunikacije pri oddaljenem sodelovanju. Levo zgoraj je gostitelj seje, ki deli sceno z ostalimi uporabniki spletne aplikacije. Gostitelj je na shemi že sinhroniziral sceno s strežnikom (spodaj desno) in mu posreduje posodobitev deljene scene, katero strežnik razpošlje vsem naročnikom in posodobi svojo lokalno kopijo scene. Zgoraj na sredini je prikazan udeleženec seje, ki je že prenesel sceno iz strežnika in sedaj prejema posodobitve gostitelja. Desno zgoraj pa je nov udeleženec seje, ki iz strežnika prenaša zadnjo verzijo scene.

Shema komunikacije pri oddaljenem sodelovanju je prikazana na sliki 3, kjer je prikazano, kako poteka sinhronizacija scene med dvema povezanima uporabnikoma na primeru spremembe orientacije kamere in potek priklopa novega uporabnika, ki prejme celoten opis scene.

Takšna implementacija oddaljenega sodelovanja omogoča zelo odzivno interakcijo z objekti v sceni in ponuja enak vpogled v podatke več uporabnikom ogrodja sočasno.

5 Zaključek in nadaljnje delo

V tem delu smo predstavili spletno vizualizacijsko ogrodje, razvito z namenom širše dostopnosti vizualizacije 3D medicinskih in volumetričnih podatkov ter z možnostjo oddaljenega sodelovanja med uporabniki. V delu smo predstavili tudi performančno analizo algoritma Marching cubes, ki ga uporabljamo pri pretvorbi volumetričnih podatkov. V prihodnosti nameravamo v ogrodje dodati naprednejše algoritme za pretvorbo v mrežni model in vgraditi tako podporo za neposredno upodabljanje volumetričnih podatkov kot tudi oddaljeno upodabljanje na strani strežnika. Kot je v delu omenjeno, želimo z ogrodjem podpreti tudi performančno manj zmogljive naprave in omogočiti računsko zelo kompleksne vizualizacije s pomočjo oddaljenega upodabljanja.

Literatura

[1] C. R. Crawford and K. F. King, "Computed tomography scanning with simultaneous patient translation," *Medical Physics*, no. 17, pp. 967 – 982, 1990.

[2] W. A. Kalender, W. Seissler, E. Klotz, and P. Vock, "Spiral volumetric CT with single-breath-hold technique, continuous transport and continuous scanner rotation," *Radiology*, no. 176, pp. 181 – 183, 1990.

[3] P. A. Rinck, *Magnetic Resonance in Medicine. The Basic Textbook of the European Magnetic Resonance Forum. 9th edition*. TRTF, 2016, vol. 9.1, e-Version.

[4] D. Krakow, J. Williams, M. Poehl, D. L. Rimoin, and L. D. Platt, "Use of three-dimensional ultrasound imaging in the diagnosis of prenatal-onset skeletal dysplasias," *Ultrasound in Obstetrics and Gynecology*,