УНИВЕРЗИТЕТ У БЕОГРАДУ

ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Ђорђе М. Ђурђевић

ЕФИКАСНА ПАРАЛЕЛНА КОМПРЕСИЈА ПОЉА ВИСИНА

докторска дисертација

Београд, 2013

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRICAL ENGINEERING

Đorđe M. Đurđević

EFFICIENT PARALLEL COMPRESSION OF HEIGHT FIELDS

Doctoral Dissertation

Belgrade, 2013

Ментор:

др Игор Тартаља, доцент, Универзитет у Београду, Електротехнички факултет

Чланови комисије:

Члан 1: др Игор Тартаља, доцент, Универзитет у Београду, Електротехнички факултет

Члан 2: др Зоран Јовановић, редовни професор, Универзитет у Београду, Електротехнички факултет

Члан 3: др Душан Старчевић, редовни професор, Универзитет у Београду, Факултет организационих наука

Члан 4: др Мило Томашевић, ванредни професор, Универзитет у Београду, Електротехнички факултет

Члан 5: др Милан Прокин, редовни професор, Универзитет у Београду, Електротехнички факултет

Датум одбране: _____

Захвалница

Најпре бих желео да се захвалим својој супрузи Душици без чије безрезервне подршке, охрабривања, велике количине стрпљења и напора да на себе преузме највећи део обавеза око одгајања наше деце, овај рад вероватно никада не би био написан. Захвалност дугујем и кћерки Ангелини на весељу и радости које је унела у наше животе у току напорног периода мог ангажовања на овом раду. Нарочито сам јој захвалан што ме је (углавном) пуштала да радим и на разумевању зашто тата није могао да је носи и да се игра са њом сваког пута када је она то хтела. Сину Милану сам захвалан што није пожурио са доласком на свет, чиме би значајно допринео продужењу времена израде овог рада, који је у време његовог рођења био у завршној фази. Но, изгледа да је одлучио да своје стрпљење на други начин накнадно "наплати"...

Захвалност дугујем својим родитељима на подршци и посвећеној пажњи, и поред тога што нису били свесни да су привођење писања овог рада крају одлагали сваког пута када су ме питали када се очекује да буде готов. Захвалност дугујем и Душициним родитељима на подршци али и због тога што ме нису (никад, чини ми се) питали када ће рад бити готов.

Значајну подршку приликом осмишљавања теме и реализације рада пружио ми је и ментор, др Игор Тартаља. То је, несумљиво, захтевало велику количину труда и стрпљења, на чему сам му веома захвалан. Захвалан сам осталим члановима комисије на пажљивом читању рада и на корисним саветима и сугестијама које сам добио као повратну информацију. Такође захваљујем Ненаду Теофиловићу на пажљивом читању рада и дискусији која је потом уследила, захваљујући којој сам успео да додатно појасним оне делове текста који су били тежи за разумевање.

Водећа светска компанија за производњу графичких адаптера, NVidia, својом донацијом модерних графичких адаптера Електротехничком факултату у Београду, омогућила ми је да извршим мерења у оквиру истраживања презентованог у овом раду. Због тога сам посебно захвалан компанији NVidia, као и колеги Андрији Бошњаковићу, који је сада тамо запослен, на посредовању приликом подношења захтева за донацију.

Сарадња са компанијом HYPACK, у оквиру неколико пројеката, омогућила ми је да стекнем значајно искуство у области манипулације и приказивања поља висина. Захваљујући том искуству, у великој мери, временом су се у мојој глави родиле идеје које су довеле до овог рада. Због тога сам захвалан компанији HYPACK, а нарочито Лазару Певцу и Пату Сандерсу.

На крају, захваљујем се свима онима који су ми директно или индиректно помогли у току израде овог рада и истраживања која су му претходила, али које сам својом непажњом или заборавом пропустио да поменем.

Наслов: ЕФИКАСНА ПАРАЛЕЛНА КОМПРЕСИЈА ПОЉА ВИСИНА

Апстракт

Дисертација предлаже нову методу за брзу компресију и декомпресију правилних поља висина, са или без губитака. Правилна поља висина су најчешће усвојен начин представљања површи узоркованих на правилним растојањима дуж две осе картезијанског координатног система у хоризонталној равни. Захваљујући напретку у техници детекције на даљину (енг. *remote sensing*) у протеклој деценији, која је достигла хоризонталне и вертикалне резолуције узорковања реда метра и дециметра, респективно, потребан је значајан простор за смештање описа површи реалних терена. При наведеним резолуцијама, површ неке планете величине Земље захтева неколико десетина терабајта. Компресија поља висина је потребна не само за компактно смештање, већ и за ефикасан пренос и манипулацију тако великим количинама података. Развијена метода погодна је за SIMD паралелну имплементацију, па самим тим и за архитектуре модерних графичких процесора, који значајно надмашују централни процесор у брзини рачунања, а већ су доступни у кућним рачунарима.

Компресија са губицима постиже се апроксимацијом поља висина скупом квадратних Безјеових површи. Безјеове површи су изабране зато што су релативно једноставне за рачунање и поседују неколико пожељних особина, које су од суштинског значаја за примену предложене методе у инжењерству. Компресија без губитака постиже се додавањем два слоја резидуала на апроксимацију. Додатно, предлаже се опциони паралелни алгоритам, специјализован за компресију резидуала. Могућност додатне примене овог алгоритма дозвољава балансирање између степена компресије и брзине декомпресије методе.

Предложену методу одликују јединствене особине, које нису присутне ни у једној другој методи компресије поља висина. Метода омогућава независну декомпресију појединачних тачака, као и прогресивну декомпресију. Чак и у случају декомпресије са губицима, декомпримована површ је инхерентно без пукотина. У поређењу са савременим компетитивним методама, које се извршавају на GPU (графичком процесору), предложена метода, у комбинацији са

широко распрострањеном општенаменском методом компресије без губитака (попут DEFLATE, која се користи у популарном алату за компресију података ZIP), или комбинована са специјализованом методом за компресију резидуала без губитака, постиже поредиве степене компресије. Ефикасност методе потврђена је кроз CUDA имплементацију алгоритама компресије и декомпресије. Имплементација основне методе (без додатне компресије резидуала), по цену разумно мањег степена компресије, благо надмашује брзину савремене компетитивне методе за велика оптерећења и значајно за мала, остварујући све горе наведене јединствене карактеристике.

У циљу провере практичне употребљивости и ефикасности методе у реалној примени, алгоритам за декомпресију интегрисан је у систем за визуелизацију терена у реалном времену. Резултати показују да, захваљујући брзој декомпресији поља висина, употреба компримованих података остварује нешто краће време цртања слике у односу на употребу некомпримованих података. Међутим, чак и мало убрзање цртања слике, у комбинацији са готово потпуним растерећењем централног процесора и смањеним саобраћајем између системске меморије и меморије графичког процесора захваљујући мањој величини компримованих података, оправдава примену предложене методе у системима за визуелизацију терена у реалном времену.

Кључне речи: *поље висина, компресија са и без губитака, прогресивна декомпресија, Безјеова површ, SIMD паралелизам, алгоритам намењен GPU, CUDA програмски модел, визуелизација терена*

Научна област: Електротехника и рачунарство

Ужа научна област: Рачунарска техника и информатика

УДК број: 621.3

Title: EFFICIENT PARALLEL COMPRESSION OF HEIGHT FIELDS

Abstract

This thesis presents a novel method for fast lossy or lossless compression and decompression of regular height fields. Regular height fields are a commonly adopted solution for representing surfaces sampled at uniform rates along the two Cartesian axes in the horizontal plane. Due to the improvements in the remote sensing technology during the last decade, which has reached horizontal and vertical sampling resolutions of the order of a meter and a decimeter, respectively, significant storage space is needed for the surface of real terrains. At these resolutions, the surface of a planet of the size similar to Earth requires a few dozens of terabytes of storage space. Compression of height fields is needed not only for efficient storage, but also for efficient transfer and manipulation of such large amounts of data. The developed method is suitable for SIMD parallel implementation and thus inherently suitable for modern GPU architectures, which significantly outperform modern CPUs in computation speed, and are already present in home computers.

Lossy compression is achieved by approximating the height field with a set of quadratic Bezier surfaces. Bezier surfaces have been chosen because they are relatively simple to compute and have several desirable features, essential for the application of the method in engineering. Lossless compression is achieved by superimposing two layers of residuals over the lossy approximation. In addition, an optional parallel algorithm, specialized for compression of residuals is proposed. The possibility of additional application of this algorithm allows for a tradeoff between the compression ratio and the decompression speed of the method.

The proposed method is characterized with unique properties, which are not present in any of the height field compression methods. The method allows independent decompression of individual data points, as well as progressive decompression. Even in the case of lossy decompression, the decompressed surface is inherently seamless. In comparison with the GPU-oriented competitive state-of-the-art method, the proposed method, combined with a widely available general purpose lossless compression method (such as DEFLATE, used in the popular ZIP data compression utility), or combined with a specialized method for lossless compression of the residuals, achieves

comparable compression ratios. The method's efficiency was confirmed through a CUDA implementation of compression and decompression algorithms. The implementation of the basic method (without additional compression of residuals), at the expense of reasonably worse compression ratio, slightly outperforms the competitive state-of-the-art method for very high workloads and considerably for lower workloads, achieving all of the above mentioned unique features.

To verify the method's practical usability and efficiency in a real application, the decompression algorithm was integrated into a real-time terrain visualization system. The results show that, thanks to the fast height field decompression, using compressed data gives a slightly higher frame rate than using uncompressed data. However, even the small increase in frame rate, along with the facts that the CPU is almost completely offloaded and that lower traffic between the system and the GPU memory is achived due to smaller size of the compressed data, justifies the usage of the proposed method in real-time terrain visualization systems.

Keywords: *height field, lossy and lossless compression, progressive decompression, Bezier surface, SIMD parallelism, GPU friendly algorithm, CUDA programming model, terrain visualization*

Scientific area: Electrical and Computer Engineering

Scientific subarea: Computer and Software Engineering

UDC number: 621.3

Садржај

1. Увод

Правилно поље висина је уобичајен начин представљања једнозначно дефинисаних површи описаних тачкама узоркованим дуж две осе Декартовог координатног система (X, Y). Висине узоркованих тачака (Z) чувају се као елементи матрице. Хоризонтална (X, Y) позиција било које узорковане тачке имплицитно је одређена хоризонталном позицијом тачке која одговара елементу матрице [0,0], растојањима између еквидистантних тачака по X и Y оси и индексима (врста, колона) одговарајућег елемента матрице.

Дигитални модели терена често се представљају пољима висина. Област примене ових модела је разноврсна. Визуелизација терена брзинама погодним за интерактиван рад, планирање у грађевини или хидрологији, симулација водених токова и ерозије тла су неколико карактеристичних примера примене. Многе примене у области геоинформационих система комбинују анализу терена са визуелизацијом у реалном времену.

У току протекле деценије, првенствено захваљујући напретку у технологији детекције на даљину (енг. *remote sensing*), снимање реалних терена на хоризонталним резолуцијама реда метра и вертикалним реда дециметра постало је практично остварив. За складиштење модела реалних терена, формираних снимањем на датим резолуцијама, потребан је значајан меморијски простор. Простор за снимак једне државе величине савезне америчке државе Јута (*Utah*) износи неколико десетина гигабајта. Простор за снимање површине целе планете величине Земље износи неколико десетина терабајта. Визуелизација великих терена захтева пренос велике количине података са екстерне меморије (локални диск или удаљен мрежни диск), кроз меморију централног процесора (енг. *central processing unit*, CPU) у меморију графичког процесора (енг. *graphics processing unit*, GPU). Приликом пројектовања система за ефикасну интерактивну визуелизацију терена мора да се обрати пажња на ограничену пропусну моћ веза између ових нивоа у меморијској хијерархији.

Компресија података је опште решење овог проблема, које је усвојено у многим доступним системима. Већина примењује методе компресије података са

губицима [ASIR2005, DICK2009, GOBB2006], што их чини задовољавајућим у многим применама визуелизације терена. Међутим компресија података са губицима је недопустива у применама где је прецизност података од пресудног значаја, попут неких анализа терена. Са друге стране, методе компресије без губитака, као што су [INAN2008, KIDN1992, KIDN2003, LEWI1994, LIND2010], немају могућност прогресивне декомпресије. Ова особина је битна када се подаци преносе кроз мрежу мале пропусне моћи, а комплетни и прецизни подаци површи нису неопходни на одредишту све време. Додатно, већина објављених метода, попут [INAN2008, KIDN1992, KIDN2003, LEWI1994], који подржавају компресију без губитака, немају могућност искоришћења SIMD (једна инструкција, више података, енг. *single instruction multiple data*, према Флиновој таксономији [FLYN1972]) паралелизма, који је доступан у модерним GPU.

Истраживање представљено у овом раду мотивисано је потребом за алгоритмом компресије поља висина без губитака, прилагођеном како секвенцијалном извршењу на CPU тако и паралелном извршењу на модерним GPU јединицама, али такође погодном за примене које могу искористити могућност прогресивне декомпресије, дозвољавајући реконструкцију површи од грубих скица до потпуно прецизног описа поља висина. Тражен је приступ опште намене за компресију података поља висина, без специјализације у некој одређеној области примене. Резултујућој методи је у ранијој фази развоја [ĐURĐ2013] дат назив HFPaC, који представља скраћеницу од "Height Field Parallel Compression" (у преводу: паралелна компресија поља висина). Међутим, скраћеница се условно може читати као "half pack" на енглеском језику (односно "хаф пек" у фонетској транскрипцији на српски језик). Такав изговор назива методе садржи игру речи на енглеском језику, јер "half pack" значи "полупаковање", чиме се алудира на могућност да се подаци компримовани овом методом могу додатно компримовати још једним кораком. У каснијим фазама истраживања, основна метода је допуњена додатним (опционим) кораком за постизање још већег степена компресије.

Предложена метода декомпонује поље висина у три слоја података. Подаци из виших слојева надовезују се на податке из претходних слојева да би створили апроксимацију поља висина вернију оригиналу. Први слој састоји се од

квадратних Безјеових[1] површи [BEZI1970] и представља грубу апроксимацију поља висина. Виши слојеви садрже две врсте резидуала. Други слој садржи делове резидуала чија вредност не може да се представи задатим бројем бита. Трећи слој садржи преостале (коначне) резидуале. Ако се поље висина дели на парцеле, што је уобичајена техника за велика поља висина, описана декомпозиција примењује се на сваку парцелу, независно.

Главне особине развијене методе су:

- врло брза компресија са и без губитака, и декомпресија које могу у потпуности да се изврше на модерним GPU, захваљујући инхерентном SIMD паралелизму на нивоу тачака;
- независна декомпресија појединачних тачака поља висина, која омогућава и дохватање само једне жељене тачке из компримованог поља висина;
- прогресивна декомпресија, која дозвољава дохватање слојева компримованих података на различитим нивоима апроксимације, односно финијих детаља поља висина само онда када су неопходни, чиме смањује количину података које је потребно пренети између слојева меморијске хијерархије у јединици времена;
- реконструисана површ је инхерентно без пукотина, чак и код декомпресије са губицима.

Основна метода даје делимично компримоване податке који могу бити додатно компримовани употребом конвенционалног кодера без губитака, као што је DEFLATE [PKWA2011]. Посебно, подаци из трећег слоја могу бити додатно компримовани кодером без губитака, специјализованим за компресију просторно корелисаних података који се могу представити променљивим бројем бита. У раду је предложена паралелна имплементација једног таквог декодера заснована на модификацији RBUC алгоритма [MOFF2005]. Степен компресије, достигнут додатном компресијом, поредив је са тренутно најефиксанијом објављеном GPU методом [LIND2010], док је у односу на њу предложена метода, уз низ

[1] *Pierre Etienne Bézier*, 01.09.1910–25.11.1999, француски инжењер

специфичних особина које је чине јединственом, показала бољи потенцијал у паралелизацији за компресију и декомпресију, па стога и у брзини, на модерним GPU. Треба напоменути да је степен компресије достигнут паралелним (GPU) методама углавном мањи (2 до 4 пута) у односу на степене компресије достигнуте секвенцијалним (CPU) методама.

У наставку, текст је организован на следећи начин. У поглављу 2 дат је преглед коришћених термина, изложени су детаљи проблема који је решаван у оквиру овог рада и објашњена је мотивација за истраживање. Кратак преглед претходних радова из области компресије поља висина, као и детаљнији преглед постојеће методе која је представљала основ за развој алгоритма додатне компресије резидуала, дати су у поглављу 3. У поглављу 4 објашњена је суштина методе, а детаљи методе и процена њене временске сложености дати су у поглављу 5. У поглављу 6 дат је преглед архитектуре употребљеног GPU и приказани су најзначајнији детаљи имплементације. Експериментални резултати примене предложене методе над неколико реалних дигиталних терена, као и поређење перформанси са компетитивном методом, дати су у поглављу 7. Практична употребљивост методе потврђена је интеграцијом методе у састав система за визуелизацију терена представљених пољима висина, што је приказано у поглављу 8. На крају, у Закључку, изнет је кратак преглед најважнијих особина предложене методе и дискутовано је о правцима даљег истраживања.

2. Проблем

У овом поглављу најпре су објашњени значајни термини који се користе у остатку рада. У наставку поглавља, представљен је проблем истраживан у оквиру овог рада и објашњена је мотивација за истраживање. Истраживан је проблем паралелне компресије и декомпресије поља висина, са и без губитака, базиран на модерним графичким процесорима, са могућношћу ефикасне и независне декомпресије појединачних тачака.

2.1 Терминологија

У овом одељку дефинисани су значајни термини који се користе у остатку рада. За све термине дат је и назив на енглеском језику који је уобичајен у литератури.

- **Правилно поље висина** (енг. *regular height field*): правоугаона матрица чији елементи представљају висине (Z) репрезентативних тачака које чине модел једнозначне површи. Позиције пројекција репрезентативних тачака на хоризонталну (X, Y) раван еквидистантне су по X и Y оси, и подударне су са центрима ћелија замишљене правилне правоугаоне решетке, тако да једном елементу матрице одговара једна ћелија решетке. Хоризонталне позиције репрезентативних тачака се не памте. Памти се позиција центра ћелије решетке која одговара елементу матрице [0,0] и растојања до центара суседних ћелија дуж X и Y осе. На основу ових података рачуна се хоризонтална позиција било које репрезентативне тачке, а њена висина се добија из одговарајућег елемента матрице. У литератури се може пронаћи и скраћен назив – поље висина (енг. *height field*), а прецизирање "правилно" користи се за наглашавање да су растојања центара ћелија решетке по X и Y оси једнака. Реални терени често се представљају правилним пољем висина. Треба приметити да дефиниција не прецизира како се одређује висина репрезентативних тачака нити колике треба да буду димензије матрице. То се одређује на основу карактеристика површи која је предмет моделирања. У случају реалних терена, на пример, могуће је направити неколико мерења висина унутар сваке ћелије решетке, а затим израчунати висину репрезентативне тачке као неку средину измерених висина.

- **Парцела** (енг. *patch*): правоугаони део поља висина (субматрица). Подела поља висина на парцеле користи се код поља висина великих размера, која не могу да се ефикасно сместе у оперативну меморију рачунара. У том погледу парцеле су аналогне меморијским страницама код организације виртуелне меморије рачунара. У оквиру система за визуелизацију терена, суседне парцеле често се преклапају над једном врстом, односно колоном, која се налази на њиховој граници.
- **Плоча** (енг. *tile*): пројекција парцеле на хоризонталну раван.
- **Сегмент** (енг. *segment*): правоугаони део парцеле или правилног поља висина у целини уколико ово није парцелисано. Сегмент представља само концептуалну поделу парцеле, без физичког издвајања података. Сви сегменти једне парцеле имају међусобно једнаке димензије. Суседни сегменти преклапају се над једном врстом односно колоном ћелија.
- **Поље висина са више резолуција** (енг. *multiresolution height field*): поље висина за које постоји изграђена хијерархија детаља. У листовима хијерархије (ниво 0) налази се најдетаљније поље висина. Остали нивои хијерархије садрже подскупове скупа тачака претходног нижег нивоа (ниво l+1 добија се систематичним уклањањем једног броја тачака нивоа l) или садрже репрезенте тачака претходног нижег нивоа (добијене, на пример, усредњавањем висина).
- **Правилна мрежа троуглова** (енг. *regular triangular mesh*): мрежа правилног геометријског облика (најчешће троугао, квадрат или правоугаоник) која се састоји од идентичних фигура (најчешће правоуглих троуглова).
- **Полуправилна мрежа троуглова** (енг. *semi-regular triangular mesh*): мрежа правилног геометријског облика која се састоји од сличних троуглова.
- **4-8 хијерархија мрежа троуглова** (енг. 4-8 *triangular mesh hierarchy*): хијерархија правилних мрежа правоуглих једнакокраких троуглова. У листовима хијерархије (ниво 0) је троугао или квадрат (два троугла са

заједничком хипотенузом). Ниво l+1 добија се бисекцијом најдужих ивица (енг. *longest edge bisection*) свих троуглова нивоа l, односно додавањем једног темена на средини сваке хипотенузе на нивоу l и додавањем ивица које спајају нова темена са наспрамним теменима у правом углу. Назив ове хијерархије потиче од чињенице да нова темена имају тачно 4 суседа, а остала темена тачно 8 суседа, осим ако се не налазе на ивици мреже.

- **Пукотина** (енг. *seam*): појава која се јавља на споју суседних парцела, када се користи компресија са губицима. Код компресије са губицима не постоји гаранција да ће се тачке дуж ивица суседних парцела декомпримовати са истим резултатом. Код система за визуелизацију терена, код којих се парцеле преклапају дуж једне врсте, односно колоне, ова појава манифестује се визуелно непријатним пукотинама које се појављују у мрежи троуглова на споју парцела. Тада је потребна додатна обрада да би се пукотина затворила додатним мрежама троуглова или визуелно маскирала.
- **Компресија са губицима** (енг. *lossy compression*): врста компресије података код које није могуће реконструисати компримоване податке тако да буду идентични оригиналу. Ова врста компресије најчешће се користи код обраде видео и аудио записа и заснива се на ограниченим могућностима људских чула вида и слуха да примете разлику између оригиналних и декомпримованих података. Тако и код поља висина која описују терене или друге површи, уколико се користе само за визуелизацију, може бити примењена компресија са губицима. Код ове врсте компресије могуће је направити компромис између степена компресије и верности реконструисаних података.
- **Компресија без губитака** (енг. *lossless compression*): врста компресије код које су реконструисани подаци идентични оригиналним. Ова врста компресије генерално постиже значајно мањи степен компресије од компресије са губицима.
- **Компресија са ограниченим губицима** (енг. *near lossless compression*): врста компресије са губицима код које се гарантује максимална разлика у

вредности појединачних оригиналних и одговарајућих декомпримованих података. Ова врста компресије примерена је подацима који представљају неку физичку величину (попут висина тачака терена) и има значајну примену управо код система за визуелизацију терена.

2.2 Поставка проблема и мотивација за његово решавање

Пратећи напредак у технологији детекције на даљину (енг. *remote sensing*) у претходној деценији, ред величине тренутно заступљених хоризонталних и вертикалних резолуција скенирања у геолошким истраживањима је достигао ред величине метра и десетог дела метра, респективно [USGS2011]. Широка и јавна доступност Интернет сервиса за интерактивно разгледање 2D или 3D мапа површи Земље, као што су [GOOG2012, MICR2012], показала је да постоји велико интересовање корисника за таквом врстом информација. Континуирано унапређење услуга које нуде ови сервиси, попут проналажења одређене локације на Земљи или планирања маршруте указује на велики комерцијални потенцијал. Персонални уређаји за навигацију применом GPS технологије постали су широко распрострањени и већина модернијих мобилних телефона има интегрисану подршку за ову врсту услуга. У основи поменутих сервиса и услуга јесу дигитални терени који представљају површ Земље, или један њен део (континент, државу, област). У табели 1 дата је процена простора потребног за складиштење поља висина које представља површ планете Земље, за неколико различитих резолуција узорковања (од 5 m до 30 m), које су честе у пракси. Процена показује да је за складиштење површи планета величине Земље, чак и са релативно великим хоризонталним растојањем између два узорка, потребан простор реда величине терабајта, или већи.

Табела 1 Процена потребног меморијског простора за складиштење некомпримованих података о висини за планету Земљу. Претпоставка је да се подаци о висини памте на дужини од 16 бита. Процена је дата, уз претпотавку поделе површи на парцеле величине 1000×1000 тачака, за три различита корака узорковања, са истим кораком дуж обе осе у хоризонталној равни, за комплетну површ планете (површине 510,072,000 km^2) и за копнени део површи (површине 148,940,000 km^2).

Корак узорковања (m)	Капацитет за целу површину планете [TB]	Капацитет за површину копна [TB]
30	1.13	0.33
10	10.2	2.98
5	40.8	11.9

Некомпримовани, реални дигитални терени захтевају простор за складиштење који често превазилази десетине гигабајта [EART2011, UTAH2011]. Компресија података је неоспорно потребна за ефикасно складиштење и пренос поља висина. У пракси, у циљу ефикасног приступа подацима, велика поља висина деле се на (релативно) мале парцеле. Тиме парцеле постају јединице за обраду, пренос и складиштење. Сходно томе, оне су такође природне јединице за компресију. Треба нагласити да поља висина, у погледу поступка компресије, представљају специјалну врсту података. Ранија истраживања [KIDN2003] показала су да компресија поља висина применом метода компресије без губитака опште намене даје мањи степен компресије у односу на примену метода специјализованих за компресију поља висина без губитака, које узимају у обзир просторну корелацију која постоји између блиских (суседних) тачака. Због свега тога постоји значајан интерес за проналажење ефикасних специјализованих метода компресије поља висина без губитака, које поред високог степена компресије брзо компримују и декомпримују податке.

Методе које врше компресију података са губицима привлачне су зато што углавном постижу виши степен компресије у односу на методе без губитака. Ипак, методе компресије са губицима често немају могућност декомпресије две суседне парцеле без појаве пукотина на месту споја. Стога те методе морају да врше додатну обраду над декомпримованим подацима, да би визуелно маскирале пукотине или обезбедиле да тачке имају исте висине дуж дељене ивице у суседним парцелама. Иако су методе компресије поља висина са губицима прихватљиве за забаву и информативну визуелизацију, компресија без губитака је веома битна за значајан број инжењерских примена. Вреди набројати неколико таквих примена: рачунање путање возила (копненог, ваздушног или подводног), рачунање линије видљивости (избор локације радара или радио антене код усмерених бежичних телекомуникација, од тачке до тачке), анализа ризика од поплава, анализа ерозије, конструкција великих структура (саобраћајнице, железничке шине, мостови, бране). У идеалном случају, требало би да метода за компресију података подржи режиме компресије са и без губитака.

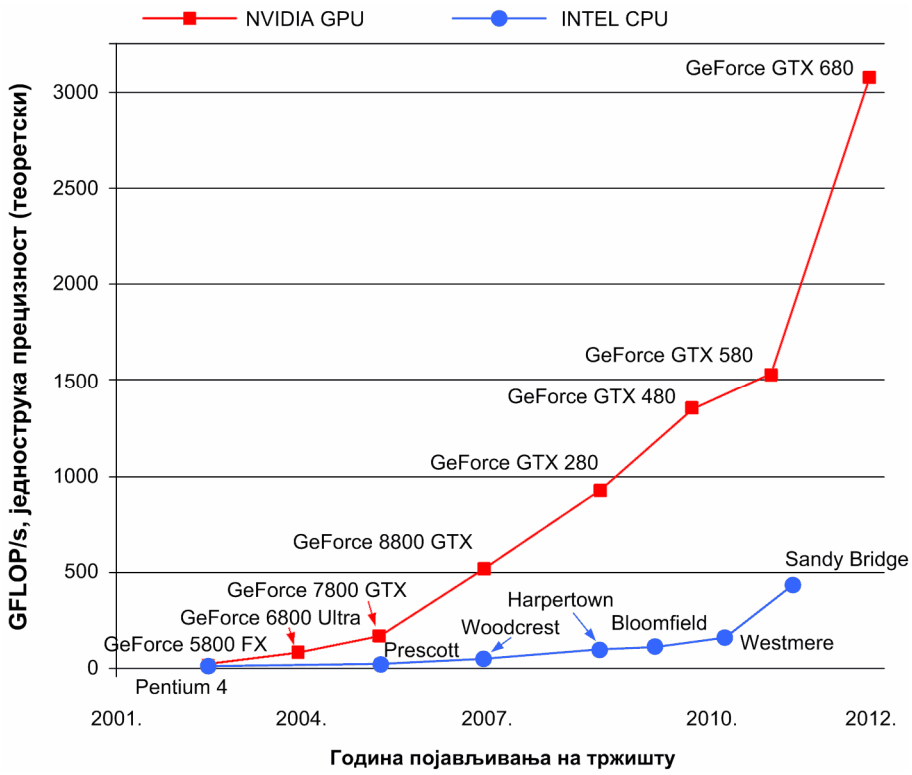
Методе компресије без губитака углавном дају предност већем степену компресије над брзином компресије и декомпресије. Брза декомпресија је

потребна у применама код којих се врши обрада огромних скупова података, или се учестано изнова приступа подацима поља висина који покривају велику област (и због тога не могу бити смештени без компресије у главну меморију рачунара) да би се над њима извршили потребни прорачуни. Уопште, свест о потреби брзе декомпресије је широко распрострањена и већи број савремених метода поставља за циљ брзу декомпресију. Ипак, код примена као што су удаљен преглед снимања површи у реалном времену или прецизно рачунање карактеристика динамички променљивог поља висина, такође у реалном времену, и брза компресија је веома битна. У оба примера, брзина компресије је од посебног значаја за ново откривене или промењене делове поља висина.

SIMD паралелизам [FLYN1972], који је доступан у модерним GPU јединицама, представља истраживачки изазов у домену веома брзих поступака компресије и декомпресије. Иако је SIMD тип паралелизма већ дуже време познат у рачунарској техници, његова примена је доживела значајну популарност и масовност тек недавно, крајем протекле деценије. Тада су се на тржишту појавили веома брзи програмабилни графички процесори који користе ту врсту паралелизма. Појаву брзих програмабилних графичких процесора пратио је и развој неопходних алата за писање програма који користе графичке процесоре за паралелне прорачуне опште намене, уместо за њихову основну намену - цртање слике. Због тога, већина метода компресије развијених у ранијем периоду не разматра паралелизацију, нарочито не специфичан SIMD паралелизам. Једна савремена компетитивна метода [LIND2010], користи SIMD паралелизам, али не паралелизам фине грануларности на нивоу тачака, већ "грубом силом" паралелно обрађује целе парцеле. На слици 1 приказана је хронологија теоретски остваривих брзина аритметичких операција над реалним бројевима у покретном зарезу на CPU компаније Intel и GPU компаније NVidia у протеклој деценији. На слици је јасно уочљива предност у брзини GPU у односу на CPU, што представља додатну мотивацију за развој паралелних GPU оријентисаних нумеричких метода.

Независна декомпресија појединачних тачака обезбеђује да додатно (режијско) време, које прати обраду малог подскупа тачака поља висина (област од интереса), буде мало. Ово је нарочито битно у применама које захтевају ефикасне прорачуне са произвољним приступом тачкама поља висина. Пример

произвољног приступа тачкама је одређивање да ли постоји оптичка видљивост између две задате тачке (линија видљивости). У тој примени, потребно је дохватити висине у свим ћелијама матрице које пресеца пројекција дужи дефинисане задатим тачкама на хоризонталну раван.



Слика 1 Поређење теоретских перформанси при извршењу аритметичких операција над реалним бројевима једноструке прецизности у покретном зарезу на GPU компаније NVIDIA, и CPU компаније INTEL. Поред сваке карактеристичне тачке дат је назив одговарајуће генерације процесора. Подаци су преузети из [NVID2012].

Када се велика количина података пољa висина преноси између два различита нивоа меморијске хијерархије, канал комуникације (мрежа или магистрала података) може брзо постати уско грло. Прогресивна декомпресија омогућава балансирање између прецизности декомпримованих података и потребне пропусне моћи комуникационог канала. Ова особина је нарочито корисна ако је декомпресија са ограниченим губицима, јер је тада гарантована максимална граница грешке апроксимације. На пример, ова особина је пожељна код визуализације терена зависне од погледа, зато што грешка у удаљеној геометрији, настала усред декомпресије, можда неће ни бити видљива. Системи за визуализацију, попут [ASIR2005, BETT2007, DICK2009, GERS2003], циљано

уносе грешку приликом састављања мреже троуглова за приказивање модела терена, редукујући број растеризованих троуглова да би смањили време потребно за цртање једне слике. Тиме се обезбеђује брз одзив који је неопходан за интерактивно управљање виртуелном камером или објектима на терену. У случају да је максимална грешка декомпримованих података мања или једнака грешци коју уноси сам систем, декомпресија са ограниченим губицима је прихватљива. Ипак, као што је претходно речено, постојеће методе компресије са губицима или са ограниченим губицима често немају могућност да изврше декомпресију две суседне парцеле без појаве пукотина, што захтева додатну обраду и има негативан утицај на ефикасност система.

Аутору није познато да је претходно објављена метода која подржава компресију са и без губитака, као и компресију са ограниченим губицима, која је погодна за ефикасну SIMD паралелизацију на нивоу тачака, која дозвољава независну декомпресију појединачних тачака, која подржава прогресивну декомпресију и која инхерентно ствара континуалну површ, без ризика од појаве пукотина.

3. Постојећа решења

У овом поглављу дат је кратак преглед претходно објављених метода за компресију правилних поља висина. Преглед GPU оријентисаних метода дат је са више детаља, у посебном одељку, с обзиром на то да су најсроднији са предложеном методом. На крају, дат је преглед методе RBUC, која је у овом раду искоришћена као основа за додатну компресију резидуала.

3.1 Преглед CPU оријентисаних метода за компресију правилних поља висина

Методе компресије података са губицима су у великој мери прихваћене за употребу у системима за приказивање терена, зато што је главни циљ ових система ефикасна визуелизација са малим временом цртања слике, док се грешка у презентацији толерише. У [GERS2003], компресија се постиже формирањем апроксимативне мреже троуглова над редукованим скупом тачака терена. Апроксимативна висина уклоњених тачака добија се линеарном интерполацијом, односно на основу једначине равни троугла чија пројекција на хоризонталну раван обухвата (X,Y) позицију посматране тачке. У [KIM2004], аутори такође праве апроксимативну мрежу троуглова. Међутим, они апроксимативну мрежу конструишу на основу коефицијената добијених применом *wavelet* трансформације над оригиналним скупом тачака терена. Осим за потребе приказивања терена, методе компресије са губицима нашле су своју примену у прогресивном мрежном преносу [XIE2008], док је паралелна имплементација ове методе у мрежи рачунара представљена у [STOO2008]. Метода је касније унапређена за компресију 5D података у [LI2011]. У том раду, компонента методе задужена за решавање великих система линеарних једначина реализована је применом CUDA (*Compute Unified Device Architecture*) [NVID2010]. Међутим, с обзиром на то да се подаци компримују итеративним поступком који је контролисан циљаном грешком апроксимације, у питању је суштински секвенцијална метода и стога није у директној компетицији са методом предложеном у овом раду.

Метода компресије са ограниченим губицима, заснована на *wavelet update lifting* методи [SWEL1998], коришћена је у [GOBB2006], а касније је оптимизована за удаљену визуелизацију у [BETT2007]. Компресија се врши тако што се, филтрирањем поља висина, формира ново поље висина ниже резолуције, односно са мање детаља. Ново поље висина користи се за предикцију (реконструкцију) оригиналног поља висина, а за сваку тачку оригиналног поља висина памте се грешке предикције на унапред дефинисаном броју бита по тачки. У циљу постизања високог степена компресије, аутори користе мањи број бита по тачки него што је потребно за прецизну реконструкцију оригиналног поља висина. Поступак компресије се затим понавља тако што се новоформирано поље висина ниже резолуције користи као улаз за нову итерацију. На тај начин, приликом компресије терена формира се имплицитна хијерархија детаља. Хијерархија је имплицитна зато што се на сваком нивоу хијерархије памте само грешке предикције. Једино поље висина које се експлицитно памти јесте оно које се налази у корену хијерархије (најмање детаља, односно најмања резолуција). Декомпресија полази од поља висина у корену хијерархије и, понављањем поступка предикције и додавања грешака предикције, реконструише остале нивое хијерархије, са ограниченим губицима. Треба приметити да ова метода не врши компресију поља висина са више резолуција, иако се (имплицитна) хијерархија детаља природно појављује у поступку компресије.

Методе компресије без губитака, специјализоване за поља висина и посебно за дигиталне терене, по правилу користе предикцију висине дате тачке на основу познатих висина суседних тачака. Интересантно је да се слична предикција користи и за компресију дигиталних слика у формату JPEG-LS [JPEG1997]. Методе специјализоване за компресију поља висина кодирају резидуале (грешке предикције) употребом јавно доступних метода за компресију података опште намене без губитака [KIDN2003]. Као предикцију, у [KIDN1992] аутори користе троугаони предиктор у комбинацији са Хафмановим[2] кодирањем. Та метода је касније унапређена у [LEWI1994]. Унапређена метода користи оптимални

[2] David A. Huffman, 9.8.1925-7.10.1999, амерички научник и истраживач.

линеарни предиктор добијен употребом Лагранжових[3] мултипликатора. Увидом у дистрибуцију вредности резидуала и опажањем да вредности резидуала имају високу концентрацију око нуле довело је до даљег унапређења степена компресије заменом Хафмановог кодера *аритметичким* кодером [KIDN2003]. Наиме, иако се дуго веровало да Хафманов кодер производи оптималну дужину кодних речи симбола, на основу њихових вероватноћа појављивања, показало се да у општем случају није тако. Минимална дужина кодне речи симбола коју Хафманов кодер може да произведе је 1 бит. Међутим, постоје ситуације када дужина кодне речи по симболу треба да буде мања од једног бита. Ове ситуације појављују се код оних скупова података код којих постоји симбол чија је вероватноћа појављивања значајно већа од вероватноћа појављивања осталих. Пример таквог скупа података управо су резидуали, јер код њих постоји изражена асиметрија у дистрибуцији (односно вероватноћама). Због тога Хафманов кодер не може да се употреби за ефикасну компресију резидуала. *Аритметички* кодер нема овај проблем, може да представи симбол на произвољном броју бита (укључујући и реалне вредности) и због тога је његова употреба за компресију резидуала ефикаснија. Аритметички кодер користи се и у [INAN2008] у комбинацији са предикцијом на основу 8 тачака. Степени компресије који постижу ове методе по правилу превазилазе степене компресије које постижу методе за компресију дигиталних слика без губитака, попут Lossless JPEG ili JPEG-LS [INAN2008, KIDN2003]. У [FRAN1995] аутори анализирају директну примену јавно доступних метода опште намене без губитака за компресију поља висина, констатују да те методе постижу задовољавајући степен компресије и сугеришу да није потребно разматрати специјализоване методе. Ипак, у [KIDN2003] показано је да специјализоване методе постижу већи степен компресије поља висина. Све наведене методе за компресију поља висина без губитака карактерише секвенцијалност у обради података, што их чини неподесним за независну декомпресију појединачних тачака поља висина, са малим потенцијалом за SIMD паралелизацију.

[3] Joseph-Louis Lagrange, 25.1.1736-10.4.1813, француски математичар и астроном

Мали број истраживања, попут [CHEN1986, KIDN1997, KIDN1999, LEWI1994, WREN1973], било је посвећено апроксимацији поља висина употребом површи вишег реда (површ првог реда је раван). Иако апроксимација површима вишег реда природно омогућава паралелизам на нивоу тачке, ова истраживања претходила су појави програмабилних GPU и у њима није разматрана могућност SIMD паралелизације.

3.2 Преглед GPU оријентисаних метода за компресију правилних поља висина

Са појавом програмабилних GPU, развијено је неколико метода за брзу декомпресију поља висина. Ове методе првенствено су биле окренуте ка приказивању терена, па су користиле компресију са губицима или са ограниченим губицима, тако да нису у директној конкуренцији са методом која се излаже у овом раду, за коју је једна од кључних карактеристика могућност компресија без губитака. У [ASIR2005] аутори користе шему интерполативне поделе (енг. *interpolatory subdivision*) да би на GPU вршили једноставну предикцију финих детаља поља висина на основу грубих. Аутори врше компресију резидуала употребом методе за компресију слика са губицима. Декомпресија се врши на CPU, а затим се резидуали смештају у GPU меморију у виду текстуре. Аутори су предложили и алтернативан начин додавања детаља пољу висина, синтезом резидуала на GPU, употребом некорелисаног Гаусовог шума. У [DICK2009] компресија готово без губитака остварена је ограничењем прецизности висинских података на 12 бита и паковањем података потребних за генерисање апроксимативне мреже троуглова за сваку парцелу терена у бит-вектор. Декодовање и синтеза геометрије врши се на GPU, у јединици за синтезу геометрије (енг. *geometry shader unit*).

Једино у [LIND2010, ZHAN2011] аутори разматрају компресију без губитака за декомпресију која се врши на GPU. Међутим, аутори тих радова не разматрају паралелну компресију на GPU, која може бити од великог значаја у одређеним применама (видети одељак 2.2). У случају [LIND2010], поступак компресије је такав да би његова имплементација на GPU била сложена и неефикасна, нарочито ако се, као и за декомпресију, употреби графички оријентисан API. У тој методи,

аутори користе линеарну предикцију и врше компресију резидуала у бит-вектор, са променљивим бројем бита по податку. Компресија се врши применом RBUC кода [MOFF2005] (видети одељак 3.3). Метода [LIND2010] намењена је декомпресији поља висина у оквиру система за цртање терена у реалном времену. С обзиром на то да употребљен систем за цртање терена користи 4-8 хијерархију мрежа троуглова [HWA2005], поље висина подељено је на парцеле облика правоуглог једнакокраког троугла. Декомпресија сваке парцеле врши се секвенцијално. Паралелизам је остварен истовременом декомпресијом више парцела на GPU, у јединици за синтезу геометрије (енг. *geometry shader unit*). За постизање пуне ефикасности, метода мора да врши истовремену декомпресију великог броја парцела. Ипак, и поред могућности паралелизације, метода [LIND2010] у основи је секвенцијална, због чега нема могућност декомпресије појединачних тачака и не користи SIMD паралелизам на нивоу тачке. Такође, метода [LIND2010] не подржава прогресивну декомпресију, односно не може да изврши декомпресију са (ограниченим) губицима, већ искључиво декомпресију без губитака. Коначно, у контексту примене методе за цртање терена, метода [LIND2010] ограничена је на декомпресију парцела које се цртају правилном (униформном) мрежом троуглова и не може да се ефикасно употреби код полуправилних мрежа троуглова (енг. *semi-regular triangulated mesh*), која се користи у [ĐURĐ2011]. Дужина странице једне троугаоне парцеле у методи [LIND2010] је n+1 тачка (n позитиван цео број), а због употребљене 4-8 хијерархије, величина терена са више резолуција ограничена је на $(2^m n+1)^2$ тачака (m позитиван цео број, где m+1 представља број нивоа у хијерархији).

У [ZHAN2011], аутори врше декомпозицију поља висина на бит-равни (енг. *bit-planes*) и независно кодирају сваку бит-раван. Бит-раван се добија тако што се посматра само један бит у бинарној репрезентацији висина. На пример, ако се висине памте на ширини од 16 бита, поље висина је могуће декомпоновати на 16 бит-равни. Кодирање се врши применом просторне структуре података коју аутори називају BQ-стабло. У питању је модификација правилног кватернарног стабла (енг. *regular quadtree*). Декомпресија сваког чвора стабла је секвенцијална. Алгоритам постиже паралелизам истовременом декомпресијом више чворова стабла. Највеће брзине декомпресије које су аутори објавили добијене су када

сваки чвор стабла садржи 64 тачке поља висина и сваки чвор секвенцијално декодује једна нит. Метода омогућава прогресивну декомпресију, јер свака бит-раван може независно да се декомпримује. Тада би се декомпресија вршила од бит-равни које потичу од бита већег значаја ка бит-равнима које потичу од бита мањег значаја. Аутори користе CUDA за имплементацију паралелне декомпресије на GPU. Због употребљеног кватернарног стабла, парцеле су величине 2^{2n} тачака (n позитиван цео број), док је димензија целог поља висина ограничена на $2^{2(n+m)}$ тачака (m и n позитивни цели бројеви).

Однос претходних GPU оријентисаних метода према карактеристикама HFPaC методе приказан је у табели 2.

Табела 2 Преглед односа претходних GPU оријентисаних метода за компресију поља висина према карактеристикама HFPaC методе. Метода 1 је [LIND2010], а метода 2 је [ZHAN2011]. Са + означено је да метода подржава дату карактеристику, а са – да не подржава. С обзиром на то да метода 1 не подржава компресију са (ограниченим) губицима, последња врста табеле се не односи на методу 1, што је означено са X.

Особина	Метода 1	Метода 2	HFPaC
Компресија са губицима	–	+	+
Компресија без губитака	+	+	+
Брза компресија на GPU	–	–	+
Брза декомпресија на GPU	+	+	+
Независна декомпресија тачака	–	–	+
Прогресивна декомпресија	–	+	+
Површ декомпримована са губицима без пукотина	X	+	+

3.3 Метода RBUC

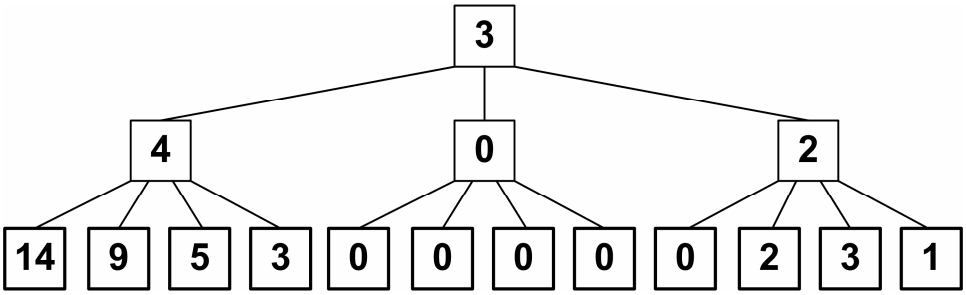
У овом одељку описана је метода RBUC (*Recursive Bottom Up, Complete*) [MOFF2005]. Иако ова метода није специјализована за компресију поља висина, овде је описана зато што је употребљена као основа за методу додатне компресије коначних резидуала у слоју 3 (видети одељак 5.6).

Метода RBUC развијена је у циљу ефикасне компресије низа података који се могу представити различитим бројем бита. У основи ове методе налази се комплетно стабло, рекурзивно формирано од дна (листова) ка врху (корену). Сви листови налазе се на истом нивоу и представљају податке које треба компримовати (сваки лист одговара једном податку). Претпоставља се да су подаци неозначени цели бројеви. Сви унутрашњи чворови стабла на истом нивоу имају исти степен гранања (исти број директних потомака), али на различитим нивоима степен гранања се може разликовати. Сваки унутрашњи чвор стабла

садржи вредност која представља минималан број бита потребан да се представи сваки његов директан потомак. Једино вредност смештену у корену стабла неопходно је памтити на унапред утврђеном броју бита. У пракси то није проблем, јер је у питању један податак, па су губици на степену компресије занемарљиви за дуже низове података, чак и када се прецени потребан број бита. Такође, с обзиром на то да број бита потребних за памћење потомака износи $\lceil \log_2(x+1) \rceil$, где је x највећа вредност потомака, за довољно дубока стабла у корену се по правилу добија вредност 2. Подразумева се да се чворови стабла густо пакују у бит-вектор.

Треба приметити једну интересантну особину компресије применом методе RBUC: уколико сви чворови-браћа (имају заједничког родитеља), на било ком нивоу стабла, садрже вредност 0, њих није потребно памтити. Наиме, њихов родитељ такође садржи вредност 0 (видети пример на слици 2, средњи чвор нивоа 1), што значи да се сваки његов потомак представља на дужини од 0 бита – односно, не заузима простор за складиштење. Ова особина је важна код примене за додатну компресију резидуала јер је могуће остварити значајну компресију у ситуацијама када већи број резидуала, који одговарају просторно блиским ћелијама поља висина (тако да одговарајући листови, односно унутрашњи чворови стабла, имају истог родитеља), имају вредност нула.

На слици 2 дат је пример једног стабла са 3 нивоа, које илуструје компресију низа од 12 кардиналних (ненегативних) 4-битних бројева применом методе RBUC. Ако би се подаци памтили на фиксном броју бита, у примеру са слике било би потребно 4 бита по податку, односно укупно 48 бита за 12 података. За памћење података применом RBUC методе, уз претпоставку да је потребно 4 бита за памћење вредности у корену, потребно је 4·(4+0+2)=24 бита за листове, 3·3=9 бита за чворове на нивоу 1 и 4 бита за корен, односно укупно 37 бита. Аутори методе RBUC не предлажу оптималан избор степена гранања за дати ниво стабла, али примећују да степен гранања треба да расте са удаљеношћу од листова и да га је за дати скуп података потребно експериментално утврдити.



Слика 2 Пример RBUC стабла са три нивоа. Подаци које треба компримовати смештени су у листовима. У унутрашњим чворовима и корену стабла памти се број бита потребан за представљање вредности смештених у потомке тих чворова.

Такође треба приметити да се компресија постиже захваљујући просторној корелацији података. Под просторном корелацијом мисли се на то да вредност суседних података, односно број бита потребан за њихово представљање, не варира значајно. За представљање 14 и 9 потребно је 4 бита, за представљање 5 потребан је један бит мање, односно 3 бита, а за представљање 3 довољна су 2 бита, па се кодирањем сва четири податка са почетка листе помоћу 4 бита губе само 3 бита од оптималног случаја. Слично томе, за представљање 3 и 2 потребно је 2 бита, а за представљање 0 и 1 довољан је 1 бит, те се за кодирање четири последња податка са 2 бита губе само 2 бита. Као што је претходно напоменуто, листове-браћу који имају вредност 0 није потребно памтити. Када би, на пример, редослед података са слике 2 био 14 0 0 0 9 0 0 3 5 2 3 1, било би потребно 57 бита за компресију RBUC методом (уз исту претпоставку о броју бита за представљање вредности у корену, као у претходном примеру), што је више него што је потребно за памћење некомпримованих података.

Декодирање RBUC стабла врши се спустом од корена ка одговарајућем листу. На сваком нивоу стабла дохвата се број бита на којем су представљени подаци смештени у потомке. Поступак се понавља док се не стигне до листова, када дохваћена вредност представља тражену вредност податка. Декодирање RBUC стабла може се извршити секвенцијално или паралелно. Редослед смештања чворова стабла зависи од начина декодирања. У наставку овог одељка објашњено је секвенцијално декодирање које су предложили аутори методе RBUC. Паралелно декодирање, које се користи у овом раду, објашњено је у одељку 5.6.

Код секвенцијалног декодирања, најефикасније је чворове сместити по префиксном (енг. *preorder*) поретку. У том случају, поступак декодирања је

једноставан и се своди на секвенцијално читање потребног броја бита из бит-вектора у којем је смештено RBUC стабло. Читањем садржаја корена, који је смештен на предефинисаном броју бита, добија се број бита w_0 на којем се смешта свако дете корена. Читањем наредних w_0 бита дохвата се садржај крајње левог детета корена, односно број бита w_1 на којем се смешта свако дете крајње левог детета корена. Поступак се понавља док се не стигне до листова, када прочитани подаци не представљају дужину у битима, већ кодиране податке. Подразумева се да су висина стабла и степен гранања на сваком нивоу стабла унапред познати, тако да декодер може једноставно да утврди када је стигао до листа и колико листова треба да прочита пре него што пређе на наредни чвор са неког од претходних нивоа стабла. Предност оваквог приступа декодирању јесте једноставност имплементације, а мана је немогућност директног приступа произвољном податку: за декодирање последњег податка у низу потребно је декодирати практично све претходне податке.

4. Суштина методе

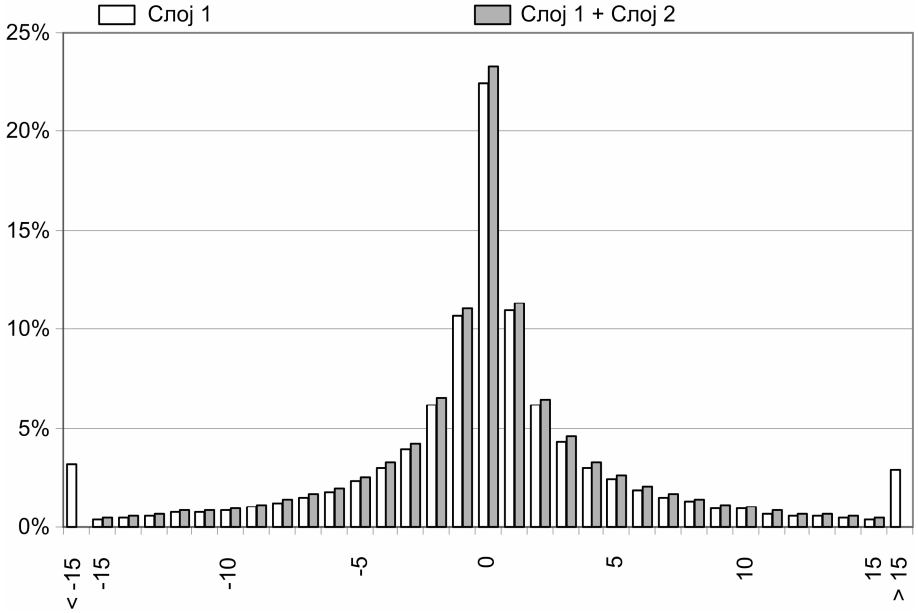
У овом поглављу биће изложена само суштина методе, док ће детаљи бити изложени у поглављу 5. Метода ће бити описана и илустрована на једноставном примеру. Биће описане и квалитативне разлике предложене методе у односу на приступе којима конкурише.

4.1 Опис методе

Приликом компресије поља висина, метода ствара три слоја података. Сваки слој, када се декомпримује и дода на податке претходног слоја, ствара апроксимацију која је вернија оригиналу. Слој 1 представља најгрубљу апроксимацију и састоји се од контролних тачака које дефинишу квадратне Безјеове површи [BEZI1970]. Разлике између поља висина и ове апроксимације могу се начелно представити на мањем броју бита него оригинални подаци. Метода омогућава да се зада циљни (релативно мали) број бита за представљање разлика (резидуала), али не захтева да се све разлике могу представити тим бројем бита, односно оставља могућност да се неке тачке памте на посебан начин. Управо слој 2 садржи разлике које се не могу представити циљним бројем бита. Према томе, овај слој представља следећи ниво апроксимације која обезбеђује гарантовану границу за максималну грешку апроксимације (обезбеђује компресију са ограниченим губицима). Слој 3 садржи преостале резидуале, који представљају најфиније детаље поља висина и који су неопходни за декомпресију без губитака, односно за реконструкцију потпуно прецизне површи поља висина. Сваки резидуал памти се на циљном броју бита.

Иницијално, поље висина се дели на скуп идентичних правоугаоних сегмената. Ова подела је само концептуална: не врши се никакво померање или екстракција података. Ако је поље висина подељено на парцеле, свака парцела посматра се као независно поље висина. Суседни сегменти преклопљени су за једну врсту, односно колону, ћелија поља висина. За сваки сегмент дефинише се једна квадратна Безјеова површ, као апроксимација одговарајућег дела поља висина. Тачке у угловима сегмента користе се као четири (од девет) контролне тачке одговарајуће Безјеове површи. Пошто суседни сегменти имају заједничке ивичне ћелије, контролне тачке на ивицама сегмената заједничке су за суседне сегменте.

Начин на који се одређују преостале контролне тачке Безјеове површи описан је у одељку 5.2. Након израчунавања контролних тачака Безијеове површи, рачунају се разлике одузимањем добијене апроксимације од оригиналног поља висина.



Слика 3 Дистрибуција резидуала за поље висина са слике 5. Приказане су дистрибуције међурезидуала, након грубе апроксимације Безјеовим површима (Слој 1) и коначних резидуала, након додавања истакнутих тачака (Слој 1 + Слој 2). Око 6% међурезидуала има вредност ван опсега [-15, 15], приближно подједнак број са обе стране опсега. Са слике се види да је дистрибуција вредности резидуала симетрична у односу на 0, што указује на то да предложена апроксимација Безјеовим површима нема тенденцију пребацивања или подбацивања вредности.

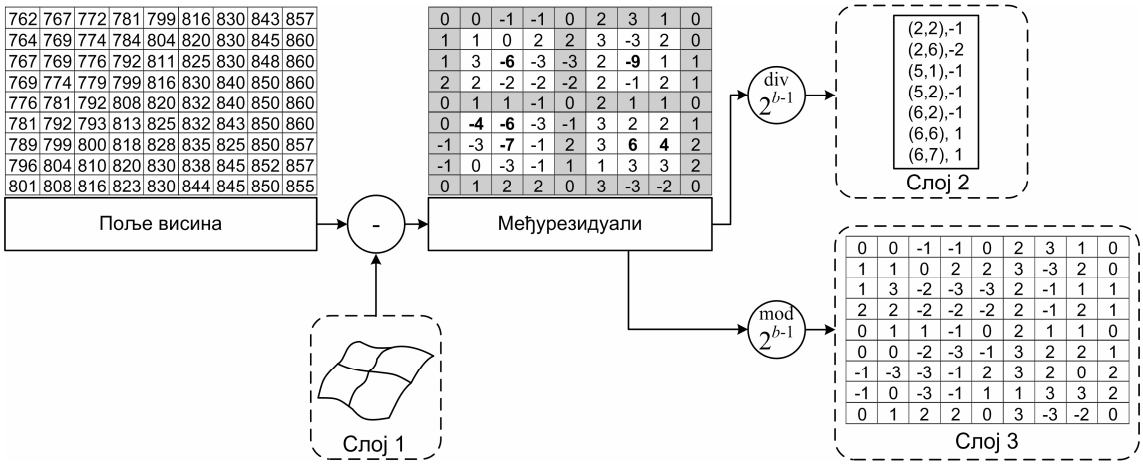
Разлике између апроксимације из слоја 1 и одговарајућих тачака поља висина чине *међурезидуале*. Ови резидуали памте се у слојевима 2 и 3. Дистрибуција међурезидуала на пољима висина типичним за географске терене показује да постоји висока концентрација вредности око нуле (видети пример на слици 3), што је заједничко за све методе компресије базиране на предикцији [KIDN2003]. Сходно томе, већина међурезидуала, који ће бити означени као *коначни резидуали*, могу бити представљени релативно малим бројем бита b. Сви међурезидуали који захтевају више од b бита биће означени као *истакнуте тачке* (ИТ). Свака истакнута тачка дели се на део вишег реда (који превазилази b бита) који се смешта у слоју 2, и део нижег реда (b бита) који се смешта у слоју 3, заједно са коначним резидуалима. У одељку 5.3 биће дискутовано о критеријуму за избор вредности b. Коначно, с обзиром на то да се у слоју 3 појављује мноштво истих вредности, слој 3 се додатно може компримовати. У овом раду, додатна компресија реализована је алгоритмом који се заснива на примени RBUC стабла

[MOFF2005] (описаног у одељку 3.3). Ова додатна компресија резидуала је опциона, тако да метода омогућава балансирање између степена компресије и брзине компресије и декомпресије.

Једноставна опциона оптимизација за компресију парцела дигиталних терена који садрже велике хоризонталне површи (попут водене површи уз обалу) би изоставила све кораке компресије ако су минимална и максимална висина исте. Ова оптимизација складишти само једну вредност висине за целу хоризонталну парцелу.

4.2 Примери

Илустрација претходног описа методе приказана је на слици 4. Слој 1 садржи четири Безјеове површи. Слој 2 садржи горње бите истакнутих тачака (ИТ) односно прекорачење локације од 3 бита, колико је предвиђено за складиштење коначних резидуала у овом примеру. Све ИТ означене су подебљаним цифрама у блоку Међурезидуали. Слој 3 садржи коначне резидуале. На пример, вредност -9 у ћелији са индексима (2,6) у блоку Међурезидуали, декомпонује се као -2 (други улаз у слоју 2) и -1 (одговарајући резидуал у слоју 3): $-2\cdot 2^{3-1}-1=-9$. Под претпоставком да се оригинални подаци памте на дужини од 16 бита, потребно је 9·9=81 16-битних речи за памћење оригиналног поља висина. Ако се примени предложена метода компресије, а подаци из сваког слоја пакују у 16-битне речи, за памћење слоја 1 потребно је 25 речи (због преклапања сегмената на заједничким ивицама, потребно је за 4 сегмента имати 25 контролних тачака које се памте на дужини од 16-бита), за памћење слоја 2 потребно је 7 речи, и за памћење слоја 3 потребно је $\lceil 9\times 9/5\rceil = 17$ речи (јер је могуће спаковати 5 резидуала представљених на дужини од 3 бита у једну 16-битну реч, уз 1 неискоришћен бит по речи). Укупно, за памћење сва три слоја, потребно је 25+7+17=49 речи, односно око 40% мање него за памћење оригиналног (некомпримованог) поља висина.



Слика 4 Илустрација разлагања поља висина у компримоване слојеве података. Слој 1 садржи четири Безјеове површи. Ивице одговарајућих сегмената означене су у блоку Међурезидуали сивом бојом. У везама између блока Међурезидуали и блокова Слој 2 и Слој 3, div и mod означавају операције целобројног дељења и дохватања остатка приликом целобројног дељења, респективно. Пример је дат за b=3.

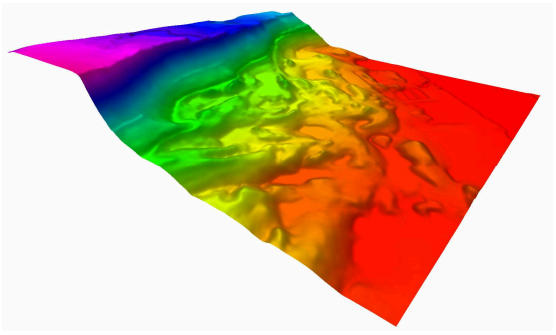
Пример компресије парцеле реалног терена која се састоји од 257×257 ћелија подељених у сегменате величине 9×9 ћелија, приказан је на слици 5. Степен компресије остварен слојем 1 је 1:15.6, слојевима 1 и 2 је 1:8, слојевима 1, 2 и 3, са смештањем резидуала на фиксном броју бита, је 1:2.2 и са слојевима 1, 2 и 3, са опционом додатном компресијом резидуала, је 1:2.6. Треба приметити да разлике између слика 5а и 5б, а нарочито између слика 5а и 5г нису једноставно уочљиве, што илуструје квалитет апроксимације Безјеовим површима.

4.3 Особине методе

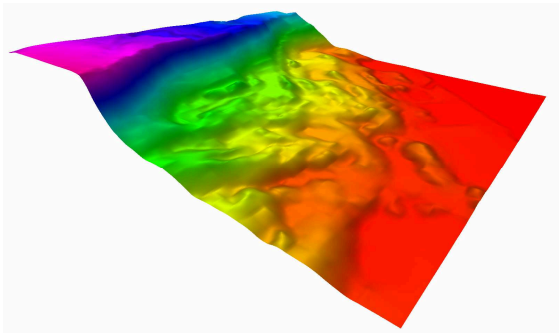
У овом одељку биће представљене основне карактеристике методе: флексибилност величине сегмента, брзина паралелне компресије и декомпресије (са или без губитака) која се постиже без већег оптерећења CPU, дохватање и декомпримовање произвољне тачке из компримованих података, прогресивна декомпресија по слојевима и непрекидност површи добијене декомпресијом, чак и код компресије за губицима.

Парцела поља висина се састоји од M×N сегмената, а сваки сегмент садржи m×n ћелија поља висина. Сходно томе, величина парцеле поља висина износи (M×(m-1)+1)×(N×(n-1)+1) ћелија. У односу на досадашње GPU базиране методе [LIND2010, ZHAN2011] (видети одељак 3.1), предложена метода нуди

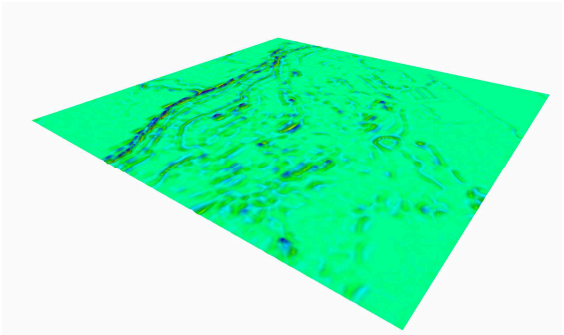
флексибилнији избор величине парцеле, јер ни сегмент нити парцела не морају имати исти број врста и колона ћелија.



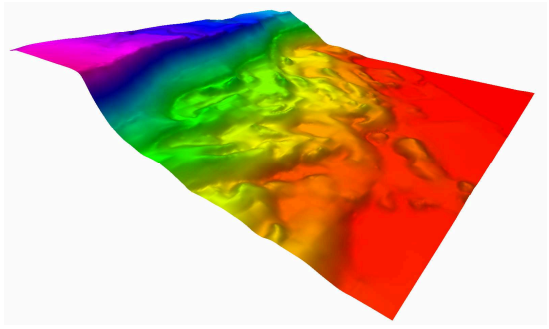
а) Оригинални подаци



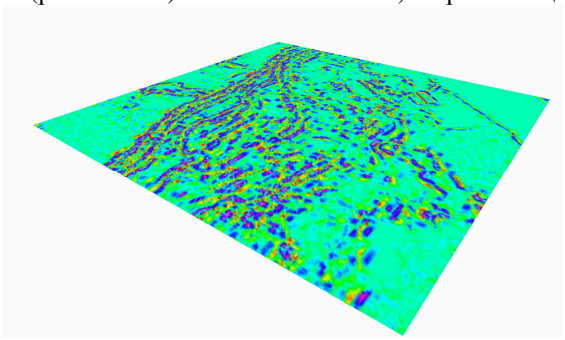
б) Апроксимација слојем 1



в) Међурезидуали (разлика а-б)



г) Апроксимација слојевима 1 и 2



д) Коначни резидуали

Слика 5 Декомпресија једне парцеле од 257×257 тачака, дигиталног терена Пацет Саунд (eng. Puget Sound), компримован употребом сегмента величине 9×9 тачака и 5 бита по коначном резидуалу. Опсег висина је од 0 до 3026 јединица на слици (а), (б) и (г), од -95 до 105 на слици (в), и од -15 до 15 на слици (д). Фактор скалирања је исти за све слике, због чега (в) и (д) изгледају готово потпуно равно. Скала боја на свакој слици прилагођена је специфичном опсегу висина за дату слику да би се међурезидуали у (в) и коначни резидуали у (д) могли уочити.

Приликом компресије и декомпресије, метода не уноси зависност између тачака, већ се свака тачка обрађује независно од осталих. Захваљујући томе, иако компресија захтева одређене прорачуне који укључују све тачке сегмента, независна декомпресија појединачних тачака је инхерентно својство методе, што методу чини инхерентно SIMD паралелном на нивоу тачака, и омогућава врло

брзу компресију и декомпресију поља висина на модерним GPU, са или без губитака, уз занемарљиво учешће CPU, чак и за мали обим података.

Важна карактеристика методе је дохватање и декомпримовање произвољне тачке из компримованих података. Таква особина може бити кључна у интеракцији са сценом, када се жели одредити тачка поља висина на коју је указао корисник, иако је декомпримован само први слој података. С обзиром на то да се резидуали представљају на једнаком броју битова, једноставно је дохватање коначног резидуала који одговара жељеној тачки. Такође, с обзиром на то да се табела у слоју 2 може брзо претражити, провера да ли жељена тачка представља истакнуту тачку и евентуално додавање одговарајуће вредности резидуала из другог слоја на коначни резидуал је веома брзо. Друге методе [KIDN2003, LEWI1994, LIND2010] захтевају декомпресију једног броја (у најгорем случају – свих) података да би се дохватила жељена тачка из компримованих података.

Разлагање компримованих података на слојеве омогућава прогресивну декомпресију података поља висина. Предложена метода предвиђа висину свих тачака у пољу висина употребом релативно малог броја контролних тачака. Будући да су контролне тачке независне од резидуала, предикција је самодовољна и представља грубу апроксимацију поља висина (слика 5б). Насупрот томе, претходне методе које користе предикцију [KIDN2003, LEWI1994, LIND2010] нису пројектоване да направе апроксимацију и могу да декомпримују само тачне вредности. С обзиром на то да коначни резидуали (слој 3) обезбеђују само фине детаље и да они углавном заузимају највише простора за складиштење (са или без додатне компресије), они се не дохвају и декомпримују, осим ако не постоји потреба за њима. Као што је напоменуто у одељку 2.2, ово је веома битно код примена где апроксимација поља висина на основу делимично декомпримованих података из нижих слојева задовољава потребе.

Коначно, декомпримована површ је инхерентно непрекидна, чак и када је декомпресија са губицима. Ивица коју деле два сегмента користи се за конструкцију обе Безјеове површи на начин који гарантује да ће дељена ивица обе површи бити декомпримована на исти начин, као што је објашњено у одељку 5.1.

Ово је тачно и када је поље висина парцелисано и два суседна сегмента припадају суседним парцелама.

5. Детаљи методе

У овом поглављу изложени су детаљи методе. Најпре је дат кратак преглед квадратних Безјеових кривих и површи [GERA2002], са акцентом на неке особине које су од суштинске важности за предложену методу. Након тога предложен је начин израчунавања параметара за једну Безјеову површ која покрива један сегмент поља висина. Затим је описан начин на који предложена метода кодира податке у слојевима 2 и 3. Размотрене су и могућности уопштавања методе. Дата је и процена временске сложености методе. На крају, изложени су детаљи методе додатне компресије коначних резидуала из слоја 3.

5.1 Преглед квадратних Безјеових кривих и површи

Облик квадратне Безјеове криве која лежи у произвољној равни у простору (слика 6а) дефинисан је помоћу три 3D контролне тачке. У наставку ће се за контролне тачке користити ознака B_i (i=0..2). Тачке B_0 и B_2 одговарају почетној и крајњој тачки криве и биће означене као *фиксне тачке*, а тачка B_1 као *прилагођавајућа тачка*. Израз (1) дефинише произвољну тачку криве. Подразумева се да се израз (1) примењује независно на сваку просторну координату (x, y и z).

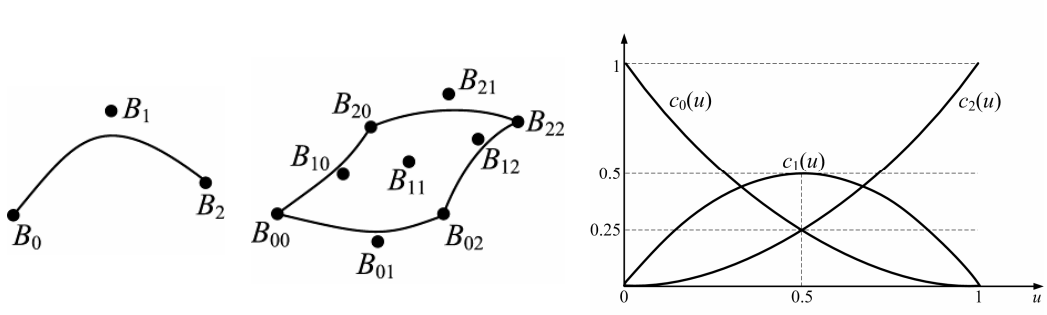
$$B(u) = c_0(u)\cdot B_0 + c_1(u)\cdot B_1 + c_2(u)\cdot B_2, \text{ где је } 0\leq u\leq 1 \quad (1)$$

$$c_i(u) = \binom{2}{i}\cdot u^i\cdot(1-u)^{2-i} \quad (2)$$

У изразу (1), коефицијенти $c_i(u)$ дефинисани су изразом (2). У суштини, коефицијенти $c_i(u)$ представљају независне доприносе контролних тачака B_i датој тачки криве. Једно интересантно својство ових коефицијената јесте да је њихов збир увек једнак 1, ако је вредност u у опсегу датом у изразу (1), што се може уочити на слици 6в. Као директна последица овог својства, за u у истом опсегу, све тачке криве налазе се унутар троугла чија темена чине контролне тачке криве. Такође, свака тачка квадратне Безјеове криве лежи у равни у којој леже контролне тачке криве. Ово својство биће од интереса у наставку. Тангенте криве у тачкама B_0 и B_2 поклапају се са дужима $\overline{B_0B_1}$ и $\overline{B_1B_2}$, респективно. Крива је глатка, непрекидна и диференцијабилна произвољан број пута у свакој тачки.

Транслација или ротација контролних тачака не мења облик криве (подразумева се да се иста трансформација примењује на све контролне тачке).

Контролна тачка B_1 не доприноси почетку и крају криве, јер је $c_1(0)=c_1(1)=0$ (слика 6в). У општем случају, крива не пролази кроз тачку B_1, већ се B_1 понаша као тачка привлачења криве. Крива пролази кроз тачку B_1 само онда када је тачка B_1 колинеарна са тачкама B_0 и B_2. Крива почиње у тачки B_0 и завршава се у тачки B_2, јер је $B(0)=B_0$ и $B(1)=B_2$, односно $c_0(0)=1$, $c_1(0)=c_2(0)=0$ и $c_2(1)=1$, $c_0(1)=c_1(1)=0$, што се такође може видети на слици 6в.



а) Квадратна Безјеова крива

б) Квадратна Безјеова површ

в) Вредност коефицијената $c_i(u)$ у зависности од u

Слика 6 Пример квадратне Безјеове криве (а) и површи (б). Зависност вредности коефицијената $c_i(u)$ од параметра u (в).

У специјалном случају, када је ${}^qB_1=({}^qB_0+{}^qB_2)/2$, где q означава било коју од просторних координата (x, y или z), израз (1) своди се на ${}^qB(u)=(1-u)\cdot{}^qB_0+u\cdot{}^qB_2$, што представља линеарну интерполацију између ${}^qB_0$ и ${}^qB_2$. Ова особина ће се у наставку користити у случају када $q\in\{x,y\}$ да би се израчунали коефицијенти c_i за било коју тачку криве или површи. Наиме, у случају који је од интереса у овом раду, а то је апроксимација скупа тачака чије су пројекције на хоризонталну раван равномерно распоређене дуж правца паралелног x (или y) оси једном Безјеовом кривом, потребно је одредити апроксимативну висину (z) неке тачке за дато x. Другим речима, од интереса је пресликавање $z=f(x)$. С обзиром на то да је Безјеова крива параметарска крива, односно да су x и z дати у функцији од u, најпре треба одредити вредност u за дато x, а затим израчунати вредност z. У општем случају, одређивање u би захтевало решавање квадратне једначине по x и доношење одлуке о избору једног од два могућа решења. Међутим, у претходно поменутом специјалном случају, пресликавање x у u је једноставно и једнозначно. Из израза $x=(1-u)\cdot{}^xB_0+u\cdot{}^xB_2$, једноставно се добија $u=(x-{}^xB_0)/({}^xB_2-{}^xB_0)$. На сличан начин може

се успоставити веза између v и y, што је потребно код квадратних Безјеових површи, које су представљене у наставку.

Квадратну Безјеову површ (слика 6б) одређује девет контролних 3D тачака које ће бити означене са B_{ij} (i=0..2, j=0..2). Израз (3) дефинише било коју тачку површи $B(u,v)$.

$$B(u,v)=\sum_{i=0}^{2}\sum_{j=0}^{2}c_i(u)\times c_j(v)\times B_{ij}, 0\le u,v\le 1 \qquad (3)$$

Из истог разлога као и у случају квадратне Безјеове криве, контролна тачка B_{11} нема утицаја на ивице површи. Ове ивице су суштински квадратне Безјеове криве: $B(u,0)$, $B(u,1)$, $B(0,v)$ и $B(1,v)$. Такође, контролне тачке B_{00}, B_{02}, B_{20} и B_{22} су четири угаона темена површи и биће означене као *фиксне тачке*. Контролне тачке B_{10}, B_{12}, B_{01} и B_{21} биће означене као *ивичне прилагођавајуће тачке*, а тачка B_{11} као *централна прилагођавајућа тачка*.

Сада ће бити разматран случај правилног поља висина. За сваки сегмент поља висина дефинише се једна Безјеова површ. Фиксне тачке одговарају тачкама у угловима датог сегмента. Пројекција централне прилагођавајуће тачке на хоризонталну раван лежи у центру пројекције сегмента, а пројекција сваке ивичне прилагођавајуће тачке лежи на средини пројекције одговарајуће ивице сегмента. Као што је претходно напоменуто, ово омогућава да се на основу хоризонталне (x, y) позиције било које тачке сегмента одреде вредности параметара u и v. Сагласно томе, апроксимативна висина тачке у врсти i и колони j сегмента рачуна се из израза (3) тако што се постави $u=i/(n-1)$ и $v=j/(m-1)$, где су n и m бројеви врста и колона сегмента, респективно. С обзиром на то да је хоризонтална позиција фиксних и прилагођавајућих тачака имплицитно одређена мрежом поља висина, у наставку рада ће се помињање позиције ових тачака заправо односити на њихове висине. Ако је број врста и колона сегмента паран, само фиксне тачке пројектоваће се тачно на центре ћелија поља висина. Међутим, ово није проблем, јер су прилагођавајуће тачке фиктивне, без директне везе са подацима поља висина.

Као што је претходно напоменуто, суседни сегменти преклапају се за једну врсту односно колону, која представља њихову заједничку ивицу. Имајући у виду да је

облик ивице дефинисан помоћу одговарајуће две фиксне тачке и једне прилагођавајуће тачке (друге контролне тачке не учествују), исте фиксне и прилагођавајуће тачке могу бити употребљене за Безјеове површи оба суседна сегмента. Стога, површ дефинисана на споју два сегмента је гарантовано непрекидна (без зазора) и мање контролних тачака је потребно да би се ускладиштио апроксимативни опис поља висина.

5.2 Одређивање контролних тачака

У овом одељку објашњава се како се рачуна (вертикална) позиција прилагођавајућих тачака за Безјеову површ која покрива један сегмент. Биће претпостављено, без умањења општости, да сегменти имају једнак број врста и колона, који ће бити означен са n. Најпре се рачуна позиција ивичних прилагођавајућих тачака. Затим се рачуна позиција централне прилагођавајуће тачке.

Посматра се једна ивица сегмента, на пример лева ивица на слици 6б. Циљ је одредити позицију ивичне прилагођавајуће тачке B_{10}, задржавајући позицију фиксних тачака B_{00} и B_{20}, тако да сума квадрата разлика између апроксимације висина ивичних тачака и висине одговарајућих тачака поља висина буде минимална. Да би се постигао овај циљ, полази се од предимензионисаног система (4). Систем је предимензионисан јер се тражи апроксимација n познатих тачака поља висина на датој ивици (P_0, ..., P_{n-1}) кривом коју дефинишу три тачке (B_{00}, B_{10}, B_{20}), а које треба одредити. Познат приступ за решавање оваквог система јесте метода најмањих квадрата [GERA2002]. Међутим, директна примена ове методе није погодна зато што она даје оптималну позицију за све контролне тачке (B_{00}, B_{10}, B_{20}), а циљ је задржати тачке B_{00} и B_{20} тачно на површи поља висина ($B_{00}=P_0$ и $B_{20}=P_{n-1}$).

$$\begin{bmatrix} c_0(u_0) & c_1(u_0) & c_2(u_0) \\ \vdots & \vdots & \vdots \\ c_0(u_{n-1}) & c_1(u_{n-1}) & c_2(u_{n-1}) \end{bmatrix} \times \begin{bmatrix} B_{00} \\ B_{10} \\ B_{20} \end{bmatrix} = \begin{bmatrix} P_0 \\ \vdots \\ P_{n-1} \end{bmatrix} \qquad (4)$$

С обзиром на то да је B_{10} једина непозната вредност у систему, независна од фиксних тачака B_{00} и B_{20}, уређивањем израза (4) добија се израз (5), над којим се

затим примени метода најмањих квадрата. Коначан израз за одређивање B_{10} је (6), где је $\boldsymbol{C}_e$ дефинисано у (7). Овај поступак примењује се на све ивичне прилагођавајуће тачке. Треба приметити да су коефицијенти c_i и вектор $\boldsymbol{C}_e$ независни од контролних тачака, па могу бити израчунати унапред за дато n. Такође треба приметити да (7), а самим тим и (6), могу бити редуковани јер важи да је $c_1(u_0)=c_1(u_{n-1})=0$, па ове компоненте $\boldsymbol{C}_e$ не доприносе позицији тачке B_{10}. Због конзистентности, ови изрази ће у наставку бити коришћени у њиховом пуном облику. Ипак, ова особина користи се у имплементацији методе (видети одељак 6.3).

$$\begin{bmatrix} c_1(u_0) \\ \vdots \\ c_1(u_{n-1}) \end{bmatrix} \times \begin{bmatrix} B_{10} \end{bmatrix} = \begin{bmatrix} P_0 \\ \vdots \\ P_{n-1} \end{bmatrix} - \begin{bmatrix} c_0(u_0) & c_2(u_0) \\ \vdots & \vdots \\ c_0(u_{n-1}) & c_2(u_{n-1}) \end{bmatrix} \times \begin{bmatrix} B_{00} \\ B_{20} \end{bmatrix} \tag{5}$$

$$B_{10} = \boldsymbol{C}_e \times \left(\begin{bmatrix} P_0 \\ \vdots \\ P_{n-1} \end{bmatrix} - \begin{bmatrix} c_0(u_0) & c_2(u_0) \\ \vdots & \vdots \\ c_0(u_{n-1}) & c_2(u_{n-1}) \end{bmatrix} \times \begin{bmatrix} B_{00} \\ B_{20} \end{bmatrix} \right) \tag{6}$$

$$\boldsymbol{C}_e = \frac{1}{\sum_{i=0}^{n-1} \left(c_1(u_i) \right)^2} \begin{bmatrix} c_1(u_0) & \dots & c_1(u_{n-1}) \end{bmatrix} \tag{7}$$

Након одређивања позиција свих ивичних прилагођавајућих тачака, потребно је израчунати позицију централне прилагођавајуће тачке B_{11}. С обзиром на то да је сада B_{11} једина непозната, рачуна се према изразу (8), који је сличан изразу (6), где је $\boldsymbol{C}_c$ дато у изразу (9). Овде се користе све тачке поља висина обухваћене датим сегментом ($P_{0,0}$, ..., $P_{n-1,n-1}$). Као и вектор $\boldsymbol{C}_e$, вектор $\boldsymbol{C}_c$ може бити израчунат унапред за дато n. Слично као и код израза (6) и (7), изрази (8) и (9) могу бити редуковани јер компоненте $\boldsymbol{C}_c$ где је $u=u_0$, $u=u_{n-1}$, $v=v_0$ или $v=v_{n-1}$ увек имају вредност 0 и не доприносе позицији B_{11}. Изрази (6) и (8) показују да су рачунања која укључују тачке поља висина (P_i и $P_{i,j}$, респективно) међусобно независна и стога погодна за паралелно израчунавање SIMD типа, какво подржавају архитектуре модерних GPU.

$$B_{11} = \boldsymbol{C}_c \times \left(\begin{bmatrix} P_{0,0} \\ P_{0,1} \\ \vdots \\ P_{n-1,n-1} \end{bmatrix} - \begin{bmatrix} c_0(u_0) \times c_0(v_0) & c_0(u_0) \times c_1(v_0) & \dots & c_2(u_0) \times c_2(v_0) \\ c_0(u_0) \times c_0(v_1) & c_0(u_0) \times c_1(v_1) & \dots & c_2(u_0) \times c_2(v_1) \\ \vdots & \vdots & \ddots & \vdots \\ c_0(u_{n-1}) \times c_0(v_{n-1}) & c_0(u_{n-1}) \times c_1(v_{n-1}) & \dots & c_2(u_{n-1}) \times c_2(v_{n-1}) \end{bmatrix} \times \begin{bmatrix} B_{00} \\ B_{01} \\ \vdots \\ B_{22} \end{bmatrix} \right) \quad (8)$$

$$\boldsymbol{C}_c = \frac{1}{\sum_{i=0}^{n-1} \left(c_1(u_i)^2 \times \sum_{j=0}^{n-1} c_1(v_j)^2 \right)} \begin{bmatrix} c_1(u_0) \times c_1(v_0) & \dots & c_1(u_{n-1}) \times c_1(v_{n-1}) \end{bmatrix} \quad (9)$$

Треба напоменути да позиције ивичних прилагођавајућих тачака, израчунате према претходно описаној методи, нису оптималне. Да би се постигла боља апроксимација сегмента, позиције ивичних прилагођавајућих тачака треба рачунати узимајући у обзир све тачке сегмента. То значи да ивичне прилагођавајуће тачке треба рачунати засебно за суседне сегменте. Међутим, у том случају, Безјеове површи које апроксимирају суседне сегменте не деле ивичну прилагођавајућу тачку на дељеној ивици. Додатно, свака дељена ивица онда мора имати два скупа резидуала. Стога, тражење боље апроксимације сегмената може редуковати степен компресије и свакако би значајно повећала комплексност имплементације без обезбеђивања непрекидности површи поља висина.

5.3 Кодирање резидуала

За сваку ћелију (тачку) поља висина рачуна се разлика (резидуал) између поља висина и његове апроксимације из слоја 1. С обзиром на то да је дистрибуција разлика најчешће таква да се вредности групишу око нуле (видети одељак 4.1), највећи број њих може бити представљен на дужини од b бита, где је b мање (често три или више пута) од броја бита потребног за представљање изворних података. Оне тачке чија разлика висина не може бити представљена на дужини од b бита називају се *истакнуте тачке*. Прецизније, свака тачка чија разлика задовољава услов $|z_1 - z_o| \geq 2^{b-1}$, где z_1 представља висину дате тачке апроксимиране слојем 1, а z_o оригиналну висину тачке, јесте једна истакнута тачка. У овом услову фигурира b-1 уместо b зато што су разлике вредности са знаком (позитивне или негативне), па је за магнитуду вредности на располагању b-1 бит.

За сваку истакнуту тачку, памти се њени индекси у матрици и вредност $(z_1\text{-}z_o)/(2^{b\text{-}1})$ у слоју 2. У том смислу, слој 2 логички представља ретко поседнуту матрицу у којој се памте виши бити резидуала истакнутих тачака. Матрица се смешта у виду вектора парова (позиција, виши бити резидуала), чиме је омогућен паралелан и независан приступ елементима матрице. Запамћена вредност (виши бити) кодира се у коду са вишком. Вредност вишка имплементационо је зависна (видети одељак 6.3). Захваљујући експлицитном памћењу позиције и независном приступу истакнутим тачкама, било која рачунска јединица може да изврши декодовање виших бита било које истакнуте тачке.

У слоју 3, заједно са осталим (коначним) резидуалима, кодира се доњих b-1 бита висине истакнутих тачака и бит знака. Сваки резидуал кодира се на дужини од b бита у коду са вишком $2^{b\text{-}1}$. У случају да се врши додатна компресија резидуала ради паралелне декомпресије на GPU, неком комплементарном методом, коначни резидуали се не пакују на претходно описан начин. У одељку 5.6 дат је пример једне методе којом се може извршити додатна компресија резидуала ради паралелне декомпресије на GPU.

Метода не предлаже како одредити вредност b за постизање највећег степена компресије. У општем случају, како се b смањује, тако се смањује величина слоја 3, али расте величина слоја 2. За неку вредност $b=b_{min}$, укупна величина слојева 2 и 3 биће минимална. Ова вредност зависи од изгледа поља висина, па стога варира од парцеле до парцеле. Најпримереniју вредност b мора одредити апликација која користи предложену методу компресије. Не захтевају све апликације највећи степен компресије. Интерактивно разгледање терена зависно од положаја посматрача може жртвовати степен компресије и поставити b за све парцеле на константну вредност која гарантује жељене границе за максималну геометријску грешку, тако да једино делови терена блиски посматрачу заправо захтевају коначне резидуале.

5.4 Разматрање генерализације методе

Методу је могуће једноставно проширити увођењем више од једне итерације апроксимација. Међурезидуали би били искоришћени као улаз за другу апроксимацију Безјеовим површима итд. Ипак, експериментима је показано да

свака нова апроксимација има негативан утицај на степен компресије, а свакако и на перформансе, јер значајно продужава поступак. Експерименти такође показују да, што се степена компресије тиче, једино има смисла користити прогресивно мање величине сегмента за сваку нову апроксимацију. Мањи степен компресије је разумљив јер међу-резидуали након прве апроксимације углавном показују мало просторне корелације. Експериментално је утврђено да је најбољи избор за више од једне апроксимације употреба две апроксимације са величином сегмента 33 у првој апроксимацији и 9 у другој. Ипак, чак и ова комбинација даје за неколико процената мањи степен компресије у односу на једну апроксимацију са величином сегмента 9.

Додатно, генерализована метода би могла да издваја истакнуте тачке након сваке итерације апроксимација, односно да се поступак итеративно понавља тек над коначним резидуалима. Ипак, експерименти показују да је степен компресије и тада значајно мањи.

5.5 Процена временске сложености методе

У овом одељку се процењује временска сложеност методе. Пажња је прво усмерена ка оцењивању сложености компресије и декомпресије једног сегмента. Проширење процене на целе парцеле је директно и једноставно и објашњено је нешто касније у тексту. Најпре ће бити процењена сложеност секвенцијалне имплементације, а затим ће бити изведена сложеност SIMD паралелне имплементације. На крају ће бити анализиран случај декомпресије целог поља висина када је број рачунских јединица ограничен.

Као што је показано у одељку 5.2, за Слој 1, векторски и матрични изрази (6) и (8) захтевају $12n$ множења и сабирања за све четири ивичне прилагођавајуће тачке и $17n^2$ множења и $9n^2$ сабирања за централну прилагођавајућу тачку (n је број врста, односно колона сегмента; без смањења општости прорачуна, овде се сматра да је сегмент "квадратни", односно да има једнак број ћелија по обе димензије). Број операција по истакнутој тачки (слој 2) и по коначном резидуалу (слој 3) је константан (видети одељак 5.3). Има n^2 резидуала и, у најгорем случају, n^2-4 истакнуте тачке. Стога, ред временске сложености секвенцијалног алгоритма за компресију је $O(n^2)$.

Секвенцијална декомпресија слоја 1, према изразу (3), захтева $18n^2$ множења и $8n^2$ сабирања. Сложеност декомпресије једне истакнуте тачке и једног резидуала је константна, па је сложеност декомпресије слојева 2 и 3 такође $O(n^2)$. Стога, ред временске сложености секвенцијалног алгоритма за декомпресију (за сва три слоја) је $O(n^2)$.

Сада се разматра SIMD паралелна компресија, уз претпоставку да једна извршна програмска нит изводи све операције које се односе на једну тачку сегмента. У идеалном случају, са константним бројем операција по тачки сегмента, као што је претходно објашњено, ова претпоставка чини да паралелна имплементација има константну сложеност. Међутим, алгоритам компресије захтева претходно поменуту векторску или матричну аритметику, као и сажимање вектора дужине n^2, за издвајање истакнутих тачака. Ред временске сложености најбољег познатог паралелног алгоритма за рачунање суме елемената вектора дужине k [HARR2007] је $O(\log k)$. Овај алгоритам је у основи паралелног алгоритма за сажимање вектора, чији је ред временске сложености такође $O(\log k)$. Због тога, ред временске сложености имплементације паралелног алгоритма за компресију једног сегмента је $O(\log n)$.

Паралелна декомпресија врши се независно за сваку тачку. Апроксимативна висина сваке тачке (слој 1) рачуна се према изразу (3) (одељак 5.1) који има константну сложеност. Дохватање виших бита истакнутих тачака (слој 2), као и дохватање коначних резидуала (слој 3), такође имају константну сложеност (одељак 5.3). Имајући у виду да паралелно сумирање или сажимање нису потребни код паралелне декомпресије, уз задржавање претходне претпоставке, закључује се да је ред временске сложености паралелне декомпресије $O(1)$.

Претходна разматрања била су ограничена на један сегмент. Када се разматра обрада парцела, треба приметити да се компресија или декомпресија сваког сегмента у саставу једне парцеле врши независно од осталих. Ако је број расположивих рачунских јединица довољан да опслужи све сегменте једне парцеле, компресија и декомпресија свих сегмената може да се врши у паралели, па претходно изведени закључци о сложености алгоритама компресије и декомпресије за сегмент важе и за парцеле.

Када је број рачунских јединица (GPU језгара) C ограничен, за поље висина које се састоји од T тачака, оптималан број парцела је $P=T/C$, јер метода дозвољава свим тачкама парцеле да буду паралелно обрађене. Треба напоменути да овако одређен оптималан број парцела имплицира да су све рачунске јединице ангажоване на обради једне парцеле, односно да се у датом тренутку обрађује само једна парцела. Међутим, метода дозвољава и другачију расподелу посла на рачунске јединице, тако да више парцела може паралелно да се обрађује. У погледу перформанси не постоји суштинска разлика, јер је свих C рачунских јединица ангажовано, односно небитно је да ли се L парцела величине D тачака обрађује у паралели, или се у датом тренутку обрађује само једна парцела величине $L \cdot D$ тачака. Ипак, имајући у виду да се парцеле преклапају над једном врстом (односно колоном) тачака, укупан број преклопљених тачака, над којима се два пута врши обрада, опада са порастом величине парцела. Из тога следи да се оптималан случај постиже са величином парцеле једнаком броју рачунских јединица C, односно да је оптималан број парцела, као што је горе наведено $P=T/C$. С обзиром на то да се парцеле секвенцијално обрађују, временска сложеност методе је $O(P)=O(T/C)$. Ако би C расло према T (а раст C се уочава из генерације у генерацију GPU, тако да ова дискусија нема само теоретски значај), P би опадало према 1 (у случају $P=1$ се цело поље висина посматра као једна велика парцела). И из тога се може закључити да је, у идеалном случају, временска сложеност декомпресије $O(1)$, као што је већ претходно и наведено. Невезано за ред сложености, треба приметити да повећање величине парцеле нема значајан утицај на степен компресије HFPaC методе (видети одељак 7.2 и слике 13, 14 и 15).

5.6 Додатна компресија резидуала методом 2DRBUC

У овом одељку разматра се примена методе RBUC [MOFF2005] за додатну компресију коначних резидуала у слоју 3. Додатна компресија је опциони корак предложене методе, који се примењује уколико је за примену важнији степен компресије од брзине компресије, односно декомпресије и уколико особина HFPaC методе да омогући директно дохватање и декомпресију поједине тачке из компримоване парцеле није кључна. Најпре ће бити објашњен појам *префиксне*

суме, која се у предложеној имплементацији користи при паралелној декомпресији RBUC стабла.

Префиксна сума (енг. *prefix sum*) је вектор **b**, дужине k, настао од вектора **a**, дужине k, применом следећа два правила: (1) b[0]=0; (2) b[i]=b[i-1]+a[i-1], $i \in [1,k-1]$. Оваква префиксна сума се прецизније зове *ексклузивна* префиксна сума, зато што не укључује последњи елемент вектора **a**. У супротном, ради се о *инклузивној* префиксној суми и тада вектор **b** има k+1 елемент. Иако је израчунавање суме инхерентно секвенцијалан поступак, могуће је ефикасно паралелизовати га [HARR2007].

За разлику од секвенцијалне декомпресије RBUC стабла, објашњене у одељку 3.3, код које су чворови смештани по префиксном редоследу, код паралелне декомпресије чворове треба сместити по нивоима. Међутим, одређивање позиције сваког чвора у бит-вектору, у којем је смештено стабло, није тривијално и захтева рачунску обраду чија сложеност је пропорционална висини стабла. За одређивање позиције неког чвора, потребно израчунати збирну величину (дужину) свих нивоа стабла који претходе нивоу у којем се налази дати чвор и померај који дати чвор има у односу на позицију почетка његовог нивоа. Паралелно одређивање збирне величине и помераја захтева израчунавање паралелне префиксне суме [HARR2007] по нивоима стабла. Ред сложености рачунања паралелне префиксне суме, у чијој основи је паралелно рачунање суме елемената вектора (видети одељак 5.5) је O(log d), где је d број чворова стабла на датом нивоу. Из претходног се види да је цена паралелне декомпресије RBUC стабла повећан ред сложености поступка декомпресије у целини, који више није O(1) (константан) већ зависи од висине RBUC стабла и степена гранања на сваком унутрашњем нивоу стабла. Променљив степен гранања, који у пракси расте са повећањем раздаљине од листова (видети одељак 3.3), значајно отежава прецизно одређивање реда сложености паралелне декомпресије RBUC стабла. Због тога ће овде бити процењена горња граница реда сложености, тако што ће се увести претпоставка да је степен гранања на сваком нивоу стабла 2. Тада је висина стабла $\log_2(n)$, где је n број листова (резидуала). На сваком нивоу стабла има $n/2^i$ чворова, где је i удаљеност од листова. Горња граница реда сложености рачунања паралелне префиксне суме на сваком нивоу стабла је онда O(log n - log 2^i) = O(log n). Из

претходног, пошто се префиксне суме по нивоима стабла израчунавају секвенцијално, множењем броја нивоа стабла са редом сложености рачунања паралелне префиксне суме, закључује се да је горња граница реда сложености паралелне декомпресије RBUC стабла $O(\log n \times \log n) = O(\log^2 n)$. То је уједно и процењена горња граница реда сложености поступка декомпресије у целини.

За директну примену RBUC методе на компресију резидуала поља висина, потребно је линеаризовати матрицу резидуала (на пример, по врстама). То би, међутим, довело до занемаривања чињенице да просторна корелација међу тачкама поља висина постоји дуж две осе у хоризонталној равни. Због тога је, за потребу провере употребљивости RBUC методе за компресију резидуала у овом раду, направљена модификација RBUC методе, која је названа 2DRBUC. Суштина модификације је у томе да се груписање листова (резидуала) врши тако да се једним родитељским чвором не обухвати низ од h^2 листова, већ субматрица од $h \times h$ листова. Исти принцип важи код груписања унутрашњих чворова стабла. Овај поступак илустрован је на слици 7, за h=2.

0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0
4	2	0	0	1	1	0	0
4	2	0	0	1	1	0	0
2	2	0	0	0	0	0	0
1	0	0	0	0	0	0	0
5	1	0	2	2	2	2	0
11	5	1	2	4	4	2	0

а)

2	0	0	0
3	0	1	0
2	0	0	0
4	2	3	2

б)

3

в)

Слика 7 Пример конструкције 2DRBUC стабла; (а) листови стабла; (б) унутрашњи чворови на нивоу 1; (в) корен. Степен гранања корена је 16, а степен гранања чворова на нивоу 1 је 4. У циљу боље прегледности, неки од унутрашњих чворова на нивоу 1 (б) и одговарајући листови (а) осенчени су сивом.

Треба приметити да овакав начин компресије резидуала захтева декодовање 2DRBUC стабла пре него што директан приступ произвољном резидуалу постане могућ. Тиме се, у одређеној мери, губи могућност независне декомпресије произвољне тачке поља висина, која представља једну од особина методе HFPaC.

Посматрањем примера на слици 7, може се приметити да је метода 2DRBUC произвела 8 чворова на нивоу 1 (слика 7б) који имају вредност 0. Основна (линеарна) метода RBUC, када се изврши линеаризација матрице по врстама, са

степеном гранања 4 (истим као и у примеру на слици 7), могла би да произведе 5 чворова који имају вредност 0 на нивоу 1. Овај пример није доказ предности методе 2DRBUC над основном методом RBUC, али указује на већи потенцијал методе 2DRBUC да постигне већи степен компресије, што је последица локалности у две димензије. Експерименти спроведени на ограниченом броју парцела показали су да метода 2DRBUC често постиже већи степен компресије од методе RBUC, а веома ретко има сличан или занемарљиво лошији степен компресије. Нису спроведени детаљни експерименти поређења ова два приступа над комплетним пољима висина зато што је метода 2DRBUC само једна могућност додатне компресије коначних резидуала, без претензије да даје најбоље резултате. Детаљи имплементације ове методе дати су у одељку 6.5.

6. Имплементација методе

У овом поглављу, најпре је дискутован избор интерфејса за програмирање апликација (енг. *application programming interface* – API) за имплементацију предложене методе на GPU. Након тога, приказани су неки детаљи архитектуре и програмског модела изабраног API – CUDA. Затим су дати важнији елементи имплементације методе HFPaC у језику CUDA C. Коначно, дати су детаљи имплементације 2DRBUC методе за додатну компресију резидуала из слоја 3.

6.1 Приступ имплементацији

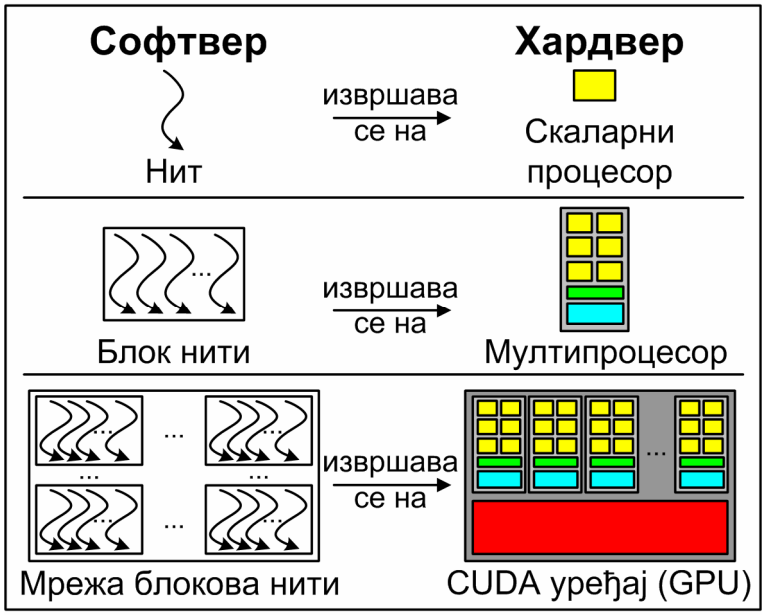
У циљу искоришћења рачунског потенцијала модерних GPU, аутор се одлучио за рачунски-оријентисан (енг. *computation-oriented*) API (CO-API). Алтернатива је графички-оријентисан (енг. *graphics-oriented*) API (GO-API), што је био уобичајен приступ у времену пре појаве CO-API. У спроведеним експериментима коришћен је језик CUDA C [NVID2010], рачунски-оријентисан језик настао из језика C, који нема директних веза са графиком. CUDA (енг. *Compute Unified Device Architecture*), производ компаније NVidia, пружа једноставан начин за сарадњу и комуникацију између програмских нити. Нити сарађују кроз *блокове нити* и комуницирају кроз брзу *дељену меморију*, смештену на самом GPU чипу. У односу на GO-API, CUDA је флексибилнија јер намеће мање ограничења и програмеру директно излаже неке елементе архитектуре. Са друге стране, да би користио GO-API за писање GPU програма, такозваних програма за сенчење (енг. *shader*), за рачунске операције опште намене, програмер мора да разуме елементе проточне обраде и основна начела на којима се заснива генерисање графике и њихова ограничења. Употребом CUDA рачунског модела, не постоји потреба да се разумеју ограничења графичке природе. На пример, било која CUDA нит може да пише податке у произвољну локацију GPU меморије. Са друге стране, локација где програм за сенчење уписује своје резултате предодређена је позицијом одговарајућег пиксела у растеру. Комуникација између нити, која је од суштинског значаја код алгоритма за компресију HFPaC, била би значајно сложенија за остваривање употребом програма за сенчење. Метода [LIND2010], који користи GO-API, наилази на ограничења у перформансама код јединице за

синтезу геометрије (енг. *geometry shader unit*), јер перформансе значајно опадају када број генерисаних темена троуглова пређе одређену границу.

Ако је визуелизација терена главни циљ, из претходне дискусије проистиче да је GO-API добар избор за имплементацију декомпресије терена, јер су декомпримовани подаци спремни за визуелизацију. У супротном, а нарочито ако је потребна нека посебна обрада терена (поред компресије и декомпресије), CO-API, попут CUDA, тренутно представља бољи избор.

6.2 CUDA архитектура и програмски модел

Најважнији детаљи CUDA архитектуре приказани су на слици 8. Рачунски потенцијал је доступан кроз неколико мултипроцесора (MP) које садржи један GPU. Сваки MP састоји се од више скаларних процесора (SP). Контролна логика за извршење програма је заједничка за све SP унутар једног MP. Зато сви SP унутар једног MP симултано извршавају исту инструкцију над (могуће) различитим подацима. Сви MP симултано извршавају исти програм, али (могуће) различиту инструкцију у истом тренутку. Нити, блокови нити и мрежа блокова нити су софтверски парњаци ове хардверске хијерархије (SP, MP, GPU, респективно).



Слика 8 Суштина CUDA архитектуре. Слика приказује пресликавање софтверских концепата (лево) на хардверске јединице (десно). Жути правоугаоници представљају скаларне процесоре, зелени правоугаоници – брзу дељену меморију, тиркизни правоугаоници – логику контроле извршења, и црвени правоугаоник – глобалну GPU меморију.

CUDA користи SIMT (једна инструкција, више нити; енг. *Single Instruction Multiple Threads*) паралелизам, који је сродан SIMD паралелизму [NVID2010]. Назив SIMT потиче из компаније NVidia, која је желела да потенцира разлику CUDA архитектуре у односу на неке конкурентске векторске SIMD архитектуре, попут SSE (*Streaming SIMD Extensions*), која је доступна у савременим Pentium централним процесорима. Кључна разлика SIMT и SSE паралелизама, са становишта програмера, лежи у начину приступа подацима и начину контроле тока програма [KIRK2010]. У SSE архитектури, подаци који се паралелно обрађују морају претходно бити смештени (спаковани по одређеним правилима) у један дељени регистар. Свака рачунска јединица чита изворне податке из унапред дефинисаног дела дељеног регистра, а затим на исти начин у дељени регистар пише резултате. Након обраде, резултати морају бити премештени из дељеног регистра у одговарајуће меморијске локације. Контрола тока програма (условна гранања) је сложена за реализацију и захтева употребу специјалних контролних инструкција. Код SIMT паралелизма, свака нит има свој скуп регистара над којима врши обраду и свој скуп регистара за чување програмског контекста. Тиме је програмер растерећен потребе за смештањем података у заједнички регистар и омогућена је једноставна контрола тока програма на нивоу појединачних нити. Укратко, за разлику од SSE архитектуре (и њој сличних), CUDA дозвољава једноставну контролу тока и манипулацију подацима на нивоу појединачних нити.

Блок нити представља колекцију нити. Све нити унутар једног блока нити извршавају се на истом MP. Један блок нити може да садржи више нити него што има расположивих SP у оквиру једног MP. Због тога, више програмских нити се конкурентно извршава на једном SP. Паралелно извршење нити реализовано је у групама од 32 нити. Ове групе ствара MP и управља њиховим извршењем, распоређујући их на доступне SP. Садржај група (расподела нити по групама) се не мења током извршења програма. Свака нит може да има независан ток инструкција, јер има сопствени бројач инструкција. Све нити у групи извршавају исту инструкцију у истом тренутку ако све прате исти програмски ток. У случају да дође до дивергенције у програмском току (на пример, због условног гранања), извршење различитих токова се серијализује. Паралелизам на нивоу нити и даље

постоји у току извршења сваког од серијализованих токова, али само за оне нити које извршавају дати ток. Остале нити чекају. Због тога, најмања ефикасност наступа онда када свака нит у групи извршава засебну грану програма (потпуна серијализација), а највећа онда када све нити у групи извршавају исту грану програма (потпун паралелизам). За сваку нит из групе, програмски контекст чува се на чипу, у оквиру MP, током целог животног века групе. Због тога, замена програмског контекста групе нити не захтева додатно време [NVID2010]. MP има могућност да одабере произвољну групу нити (која садржи нити спремне за извршавање) и покрене извршење наредне инструкције за све нити у оквиру одабране групе, приликом сваког издавања инструкције.

Све нити унутар једног блока нити могу бити синхронизоване и могу међусобно да комуницирају кроз брзу дељену меморију. Брза дељена меморија смештена је на самом чипу и понаша се као програмски контролисана брза кеш меморија за податке. Сваки блок има засебну брзу дељену меморију и њој може да се приступи једино из CUDA програма. Концептуално, блок нити је 1D, 2D или 3D вектор нити. Програмер бира димензионалност блокова сагласно са димензионалношћу проблема. Приликом позивања CUDA програмског модула, такозваног *језгра* (енг. *kernel*), програмер дефинише величину мреже и блокова. Блокови се аутоматски распоређују на доступне MP. Овакав приступ обезбеђује скалабилност без икакве модификације у програму. Уређај који има више MP покренуће више блокова нити у паралели. У оквиру језгра, позиција сваке нити унутар блока нити и позиција блока унутар мреже, доступне су кроз уграђене непроменљиве податке. GPU коришћен за спровођење експеримената у овом раду (видети одељак 7.1), NVIDIA GeForce GTX 580, има 16 MP са 32 SP по MP, и максимумом од 1024 нити по блоку нити (односно максимумом од 32 нити по SP).

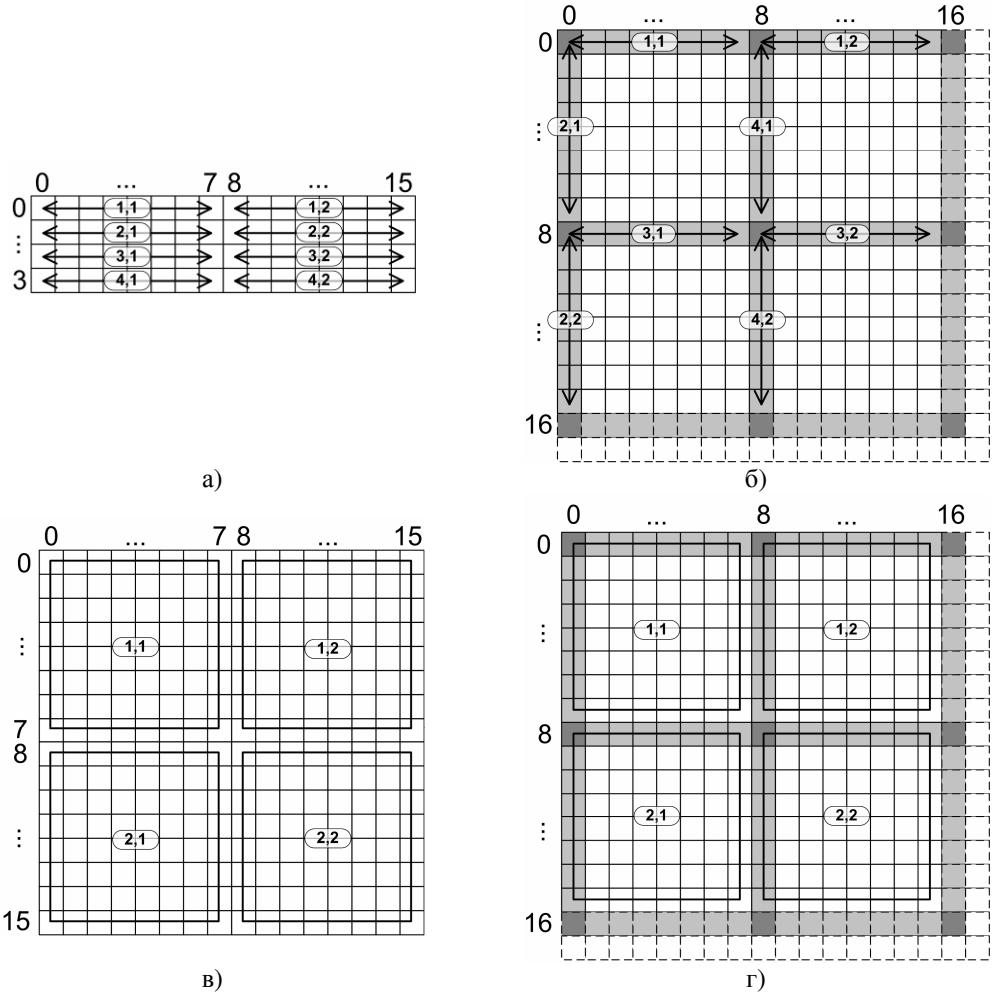
6.3 Имплементација HFPaC у CUDA C

Због ограничења од 32×32 нити по блоку (32 групе по 32 нити) које GPU, употребљен за експерименте у овом раду, има на нивоу архитектуре, највећи сегмент у имплементацији покрива 33×33 ћелије. Једна нит је везана за обраду једне ћелије, али ивичне ћелије суседних сегмената се преклапају, па је зато довољно 32×32 нити по блоку за сегмент од 33×33 ћелије (последњи ред, односно

колона, једног сегмента се обрађује блоком наредног сегмента). Последњи сегмент у реду, односно колони, састоји се од једне врсте, односно колоне, ћелија. Програмски се обезбеђује да блок нити који обрађује такав сегмент не приступа (непостојећим) ћелијама ван парцеле. Даље, овде ће бити претпостављено да се висине представљају као цели бројеви на дужини од 16 бита, што је уобичајена резолуција за податке о висини. Ипак, треба напоменути да наведена ограничења нису ограничења HFPaC методе.

Компресија се врши у неколико специјализованих програмских језгара, која се извршавају секвенцијално, једно за другим. Два језгра рачунају податке за слој 1: једно рачуна ивичне прилагођавајуће тачке, а друго рачуна централну прилагођавајућу тачку. Треће језгро рачуна податке за слојеве 2 и 3. У свим језгрима, међурезултати се памте у брзој дељеној меморији. За рачунање прилагођавајућих тачака, користи се једна нит по тачки (P_i у (6) и $P_{i,j}$ у (8)), након чега се врши паралелно сумирање [HARR2007].

Рачунање података који се смештају у слоју 1 за величину сегмента од 9×9 тачака илустровано је на слици 9. У циљу боље прегледности, на слици је претпостављено да је ограничење 16×16 уместо 32×32 нити по блоку. Слика показује да се користи један блок нити за покривање четири сегмента парцеле. Рачунање ивичних прилагођавајућих тачака у врсти или колони 16 врше други блокови. Величина блока зависи од тога да ли се рачуна ивична прилагођавајућа тачка (слике 9а и 9б) или централна прилагођавајућа тачка (слике 9в и 9г). Локација податка којег обрађује једна нит одређује се на основу локације нити унутар блока и мреже. Као илустративан пример, нити из врсте 0, означене са (1,1) и (1,2) на слици 9а, распоређене су за рачунање ивичних прилагођавајућих тачака два суседна сегмента приказана на слици 9б. Употребом оваквог пресликавања нити на сегменте постиже се боље искоришћење SP, а самим тим и већа брзина компресије, у односу на имплементацију где величина блока одговара величини сегмента. Слика 9 такође показује да се користи 8 нити за рачунање ивичних прилагођавајућих тачака на ивици сегмента на којој се налази 9 тачака. Међутим, то не представља проблем зато што је у овом случају потребно само 7 нити. Као што је објашњено у одељку 5.2, изрази (6) и (7) могу се редуковати



Слика 9 Пресликавање појединачних нити из блока нити на сегменте поља висина за рачунање ивичних прилагођавајућих тачака сегмента (а и б) и централних прилагођавајућих тачака сегмента (в и г). Свака ћелија на (а) и (в) представља једну нит. Свака ћелија на (б) и (г) представља једну тачку поља висина. Овали са истим индексима на сликама (а) и (б), односно (в) и (г) означавају пресликавање нити из блока нити на тачке у одоварајућем сегменту. Величина блока је 16×16 нити. Величина сегмента је 9×9 тачака. Фиксне тачке су обојене тамно сиво. Дељене ивице сегмената су обојене светло сиво. Други (суседни) блокови нити обрађују ћелије које су означене испрекиданом линијом на (б) и (г).

уклањањем два елемента из $\boldsymbol{C}_e$. Ипак, овде се предлаже имплементација са 8 нити због паралелног сумирања [HARR2007], јер је оно најефикасније ако се спроводи над низом чија је дужина степен 2. Поред тога, имплементација са 8 нити природно се уклапа у изабрану максималну величину блока. Тиме се избегава употреба непожељних условних гранања (видети одељак 6.2), који би били неопходни у супротном. Слично томе, за рачунање централне прилагођавајуће тачке сваког сегмента, користи се 8×8, уместо 9×9 нити (иако је заправо довољно 7×7 нити), што је приказано на сликама 9в и 9г. И поред тога што компресија слоја 1 захтева значајне операције над матрицама, попут множења матрице

вектором у изразу (6), то не представља проблем у погледу ефикасности на модерним GPU будући да су они специјализовани за интензивне и паралелне аритметичке операције. Сви подаци у слоју 1 памте се на дужини од 16 бита. Фиксне тачке памте се у изворном облику, а прилагођавајуће тачке памте се као померај у односу на фиксну тачку која се налази у горњем левом углу датог сегмента.

Приликом стварања слоја 2, поље висина се поново дели на области. Овог пута, у питању су непреклапајуће области величине 16×16 ћелија. За сваку област, применом једне нити по ћелији, врши се издвајање истакнутих тачака, према објашњењу из одељка 5.3. Виши бити истакнутих тачака смештају се у вектор 16-битних речи. За сваку истакнуту тачку, позиција у области памти се на дужини од 8 бита (по 4 бита за индексирање реда, односно колоне), а вредност, кодирана као неозначен цео број са вишком 128, на преосталих 8 бита. Након тога, примењује се паралелно сажимање [HARR2007] вектора истакнутих тачака. Након сажимања, број истакнутих тачака по области памти се у једном вектору 8-битних података.

На крају, коначни резидуали у слоју 3 пакују се у речи дужине 16 бита, са b бита по резидуалу. У циљу брзог приступа сваком резидуалу, не врши се густо паковање резидуала у бит-вектор. Због тога, b мора бити из скупа {2, 3, 4, 5, 8}. За вредности 3 и 5, један бит по 16-битној речи остаје неискоришћен. Иако је даља компресија могућа чак и са компресором опште намене без губитака (видети одељак 7.2), основна HFPaC метода не врши додатно кодирање резидуала. Имплементације које су наменски пројектоване за специфичну примену могу одлучити да даље кодирају податке у слоју 3, као и у претходним слојевима. Као пример, у одељку 6.5 дати су детаљи имплементације 2DRBUC методе, описане у одељку 5.6, која врши додатну компресију коначних резидуала уз брзу декомпресију на GPU.

Декомпресија се врши у једном програмском језгру. Свака тачка поља висина се декомпримује у засебној нити. За декомпресију слоја 1, вредности u и v (видети одељак 5.1) рачунају се на основу позиције нити у сегменту. Такође, коефицијенти $c_{0..2}(u)$ и $c_{0..2}(v)$ се рачунају, иако је могуће унапред израчунати

њихову вредност за дату величину сегмента. Таква имплементација је изабрана јер су мерења дала боље резултате у односу на имплементацију код које се унапред израчунате вредности дохватају из глобалне GPU меморије. Могуће је да би се употребом друге врсте GPU меморије, попут текстурне меморије (која поседује сопствену кеш меморију), постигла већа брзина декомпресије у односу на брзину постигнуту рачунањем вредности ових коефицијената, али та могућност није истражена у овом раду. Текстурна меморија је погоднија за дуготрајно чување унапред израчунатих коефицијената од брзе дељене меморије јер је, за разлику од брзе дељене меморије, садржај текстурне меморије могуће дефинисати пре покретања CUDA језгра. Уједно, текстурна меморија је доступна из свих блокова нити, док је брза дељена меморија индивидуална за сваки блок нити и њен садржај се губи када блок нити заврши са радом.

Током декомпресије слоја 2, нити из сваке области (величине 16×16 ћелија) посматрају се линеарно, као низ од 256 нити. Све нити чији је индекс мањи од познатог броја истакнутих тачака у датој области врше декомпресију. Остале нити остају неискоришћене. Због тога, нека од GPU језгара нису у потпуности искоришћена.

Програмско језгро врши декомпресију слој по слој. У случају да се врши декомпресија са губицима, програмско језгро завршава поступак ако подаци из наредног слоја нису доступни. Пример ситуације када подаци из наредног слоја нису доступни јесте визуелизација терена са прогресивном декомпресијом, када се велика количина података преноси са удаљеног рачунара спором везом, а потребно је да корисник што пре добије некакву, макар и непрецизну, слику терена. Тада ће се најпре пренети слој 1, затим слој 2 и на крају слој 3. Након преноса слоја 1, а пре преноса осталих слојева, корисник може добити непрецизну слику терена декомпресијом само слоја 1, који је једини у том тренутку доступан. Када остали слојеви постану доступни, подаци добијени њиховом декомпресијом додају се на претходно декомпримоване податке из слоја 1. Таква накнадна декомпресија слојева 2 и 3 (појединачно или заједно) реализовала би се у специјалним програмским језгрима намењеним декомпресији само тих слојева, што није разматрано у овом раду. Међурезултати слојевите декомпресије

смештају се у брзу дељену меморију, пре коначног писања резултата декомпресије у глобалну GPU меморију.

У имплементацији компресије и декомпресије користе се реални бројеви у покретном зарезу једноструке прецизности за рачунање Безјеових површи. Експерименти су показали да је имплементација која за рачунање Безјеових површи користи целе бројеве незнатно бржа, али значајно незграпнија. Груба процена, заснована на анализи изворног кода писаног у језику CUDA C, показује да декомпресија једне тачке која није истакнута захтева око 80 операција над реалним бројевима у покретном зарезу и око 20 операција над целим бројевима. Декомпресија истакнуте тачке захтева још око 20 додатних операција над целим бројевима. Када се пореди CUDA имплементација декомпресије са имплементацијом декомпресије у језику за сенчење, треба имати у виду да CUDA може да преклопи извршење аритметичких операција се приступом спорој глобалној GPU меморији [NVID2010]. Под условима великог оптерећења, за које је карактеристичан велики број аритметичких операција, GPU језгра су заузета све време и већи део времена извршења сакривен је неизбежним приступима меморији. Због тога, приликом поређења CUDA имплементације са имплементацијом у језику за сенчење (попут [LIND2010]), треба имати у виду да директно поређење броја операција може довести у заблуду и навести на погрешне закључке.

6.4 Псеудокод алгоритама за компресију и декомпресију

У овом одељку дат је псеудокод алгоритама за компресију и декомпресију. Сваки од приказаних листинга одговара једном програмском језгру, као што је објашњено у одељку 6.3. За означавање података који се смештају у брзу дељену меморију користи се кључна реч `shared`. За означавање места у програму на којем је потребно извршити синхронизацију нити користи се кључна реч `sync`. За означавање позиције нити унутар блока, сегмента и области (видети одељак 6.3) користе се ознаке `threadX` и `threadY` са суфиксима `B`, `S` и `A`, респективно. За означавање позиције у пољу висина коју обрађује једна нит користе се ознаке `gridX` и `gridY`. Величине блока (број врста, односно колона), сегмента и

области означене су константама BLOCK_SIZE, SEGMENT_SIZE и AREA_SIZE, респективно.

Листинг 1 приказује псеудокод алгоритма за одређивање ивичне прилагођавајуће тачке једне ивице једног сегмента. На почетку се дохватају сви подаци на посматраној ивици сегмента и смештају у низ edgeData у брзој дељеној меморији. Елементи под индексима 0 и SEGMENT_SIZE-1 овог низа су фиксне тачке. Затим се паралелно рачуна израз (6): најпре се паралелно рачунају појединачне компоненте суме и смештају у вектор sum, а затим се паралелним сабирањем компоненти овог вектора добија се тражена вредност.

Листинг 1 Псеудокод алгоритма за одређивање једне ивичне прилагођавајуће тачке. Коефицијенти C_0, и C_2 дефинисани су у изразу (2), а C_e у изразу (7). EAP означава израчунату висину ивичне прилагођавајуће тачке (*edge adjustment point*).

```
1   shared int edgeData[SEGMENT_SIZE];
2   shared float sum[SEGMENT_SIZE];
3   // data fetch
4   edgeData[threadXS] = heightField[gridY][gridX];
5   sync;
6   // compute scalar vector product
7   sum[threadXS] = (edgeData[threadXS]-
8                    edgeData[0]*C0[threadXS]-
9                    edgeData[SEGMENT_SIZE-1]*C2[threadXS]
10                  )*Ce[threadXS];
11  sync;
12  // write back
13  EAP = parallel_sum(sum);
```

Листинг 2 приказује псеудокод алгоритма за одређивање централне прилагођавајуће тачке једног сегмента, на основу израза (8). На почетку (линије 5 и 6) дохватају се све фиксне и ивичне прилагођавајуће тачке, и смештају у матрицу ctrlpts, димензија 3×3 елемента, у брзој дељеној меморији. Начин смештања контролних тачака одговара њиховом распореду у простору: фиксне тачке смештају се у ћелије (0,0), (0,2), (2,0) и (2,2), што одговара угловима матрице, а ивичне прилагођавајуће тачке између одговарајућих фиксних. Под индексом (1,1), у средини матрице, биће смештена централна прилагођавајућа тачка, означена са CAP (енг. *central adjustment point*). Након тога, аналогно поступку приказаном у листингу 1, одређује се висина CAP. Потом се рачуна апроксимација сегмента (Безјеова површ). Међурезидуали се добијају одузимањем апроксимативне висине од оригиналне. Треба приметити да се

делови апроксимације рачунају приликом одређивања висине CAP и смештају у локацију `bsurf` (линија 8). Њима је потребно додати део апроксимације који потиче од CAP да би се добила коначна апроксимација (линија 23).

Листинг 2 Псеудокод алгоритма за одређивање централне прилагођавајуће тачке. C_{ij} означава производ коефицијената c_i и c_j из израза (8). C_c означава вектор из израза (9). `CAP` означава израчунату висину централне прилагођавајуће тачке (*central adjustment point*). Матрица међурезидуала означена је са `inter_res`.

```
1  shared int ctrlpts[3][3];
2  shared float sum[SEGMENT_SIZE*SEGMENT_SIZE];
3  int threadInd = threadYS*SEGMENT_SIZE+threadXS;
4  // fetch data
5  fetch_fixed_points(ctrlpts);
6  fetch_edge_adjustment_points(ctrlpts);
7  sync;
8  float bsurf = ctrlpts[0][0] * C00[threadInd]+
9                ctrlpts[0][1] * C01[threadInd]+
10               ctrlpts[0][2] * C02[threadInd]+
11               ctrlpts[1][0] * C10[threadInd]+
12               ctrlpts[1][2] * C12[threadInd]+
13               ctrlpts[2][0] * C20[threadInd]+
14               ctrlpts[2][1] * C21[threadInd]+
15               ctrlpts[2][2] * C22[threadInd];
16
17 sum[threadInd] = (heightField[gridY][gridX]-bsurf) * Cc[threadInd];
18 sync;
19 // write back
20 CAP = ctrlpts[1][1] = parallel_sum(sum);
21 // compute Bezier surface
22 float bsurf_c = ctrlpts[1][1] * C11[threadInd];
23 inter_res[gridY][gridX] = heightField[gridY][gridX]-
24                           ROUND(bsurf+bsurf_c);
```

Листинг 3 приказује псеудокод алгоритма за издвајање истакнутих тачака и паковање коначних резидуала на фиксном броју бита у 16-битне речи. У линији 7 свака нит чита вредност одговарајућег међурезидуала (`inter_res`), који су добијени на крају листинга 2. У линији 9 издвајају се виши бити сваког резидуала. Да ли је у питању истакнута тачка, одређује се у линији 10 (вредност 1 смешта се у вектор `isProminent` ако јесте истакнута тачка, а 0 ако није). Тај вектор се касније (линија 17) користи за паралелну редукцију вектора истакнутих тачака (`prom_pts`). Добијање коначних резидуала врши се у линијама 11-13, где се од вредности међурезидуала одузима вредност истакнуте тачке (која је 0 ако није у питању истакнута тачка) и додаје вишак. У линији 15, виши бити међурезидуала и позиција у области пакују се у вектор истакнутих тачака (`prom_pts`). Након тога, у линији 17, врши се паралелна редукција низа истакнутих тачака на основу

садржаја вектора `isProminent`, када се задржавају само они елементи вектора `prom_pts` за које одговарајући елемент вектора `isProminent` има вредност 1. Након тога, врши се паковање коначних резидуала у 16-битне речи (линије 23-29), применом једне нити по речи.

Листинг 3 Псеудокод алгоритма за издвајање истакнутих тачака и паковање резидуала у 16-битне речи на фиксној дужини. Ознака за матрицу међурезидуала, која представља улаз за овај алгоритам, је `inter_res`.

```
1   shared bool isProminent[AREA_SIZE*AREA_SIZE];
2   shared unsigned short prom_pts[AREA_SIZE*AREA_SIZE];
3   shared short residuals[AREA_SIZE*AREA_SIZE];
4   int bias = (1<<(bitsForResiduals-1))-1;
5   int threadIdxA = threadYA*AREA_SIZE+threadXA;
6
7    residuals[threadIdxA] = inter_res[gridY][gridX];
8    char prom_pt =
9      (residual div (1<<(bitsForResiduals-1)))>>(bitsForResiduals-1);
10     isProminent[threadIdxA] = prom_pt != 0;
11     residuals[threadIdxA] = bias +
12                             residuals[threadIdx] –
13                             (prom_pt << (bitsForResiduals-1));
14
15     prom_pts[threadIdxA] = (prom_pt << 8)|(threadYA<<4)|(threadXA);
16     sync;
17     parallel_reduce(prom_pts, isProminent);
18     sync;
19     int residuals_per_word = 16 / bitsForResiduals;
20     int packed_words = BLOCK_SIZE*BLOCK_SIZE / residuals_per_word;
21     int threadIdxB = threadYB*BLOCK_SIZE+threadXB;
22     unsigned short packed_residuals = 0;
23     if( threadIdxB < packed_words )
24        for(int i = 0; i < residuals_per_word; i++) {
25           packed_residuals <<= bitsForResiduals;
26           packed_residuals |=
27                     residuals[threadIdxB*residuals_per_word+i];
28        }
29     final_residuals[threadIdxB] = packed_residuals;
```

Листинг 4 приказује псеудокод алгоритма за декомпресију. Први део (линије 4-19) врши декомпресију слоја 1 (конструкција Безјеове површи) над једним сегментом. Други део (линије 21-28) врши декомпресију слоја 2 (истакнуте тачке) над једном облашћу. Трећи део (линије 30-40) врши декомпресију слоја 3 (коначни резидуали). Међурезултати декомпресије сваког слоја чувају се у матрици `surface`. На крају (линија 41) резултати декомпресије уписују се у матрицу `decompressed`.

Листинг 4 Псеудокод алгоритма за декомпресију. Приказана је декомпресија Безјеове површи дефинисане над једним сегментом, декомпресија истакнутих тачака једне области и распакивање коначних резидуала над једним блоком. Резултати декомпресије сваког слоја смештају се у матрицу `surface` у брзој дељеној меморији. Коначни резултати декомпресије смештају се у матрицу `decompressed`. Вектор `prom_pts` је вектор истакнутих тачака. Вектор `residuals` је вектор спакованих резидуала.

```
1  shared int ctrlpts[3][3]
2  shared unsigned short surface[BLOCK_SIZE][BLOCK_SIZE];
3
4  int threadIdxS = threadYS*SEGMENT_SIZE+threadXS;
5  fetch_control_points(ctrlpts);
6  sync;
7  // build Bezier surface
8  float v = threadYS / SEGMENT_SIZE;
9  float u = threadXS / SEGMENT_SIZE;
10 float a = ctrlpts[0][0] * (1-u)*(1-u)*(1-v)*(1-v);
11 float b = ctrlpts[0][1] * 2*u*(1-u)*(1-v)*(1-v);
12 float c = ctrlpts[0][2] * u*u*(1-v)*(1-v);
13 float d = ctrlpts[1][0] * (1-u)*(1-u)*2*v*(1-v);
14 float e = ctrlpts[1][1] * 2*u*(1-u)*2*v*(1-v);
15 float f = ctrlpts[1][2] * u*u*2*v*(1-v);
16 float g = ctrlpts[2][0] * (1-u)*(1-u)*v*v;
17 float h = ctrlpts[2][1] * 2*u*(1-u)*v*v;
18 float i = ctrlpts[2][2] * u*u*v*v;
19 surface[threadYB][threadXB] = ROUND(a9+b9+c9+d9+e9+f9+g9+h9+i9);
20 sync;
21 // extract prominent points in area
22 int threadIdxA = threadYA*AREA_SIZE+threadXA;
23 if( threadIdxA < number_of_prom_pts ) {
24   unsigned short tmp = prom_pts[ threadIdxA ];
25   int posInArea = prom_pt & 0xFF;
26   int prom_pt = (((tmp >> 8)&0xFF)-128) << (bitsForResiduals-1);
27   surface[ (posInArea >> 4) & 0xF ][posInArea & 0xF] += prom_pt;
28 }
29 sync;
30 // extract final residuals
31 int residuals_per_word = 16 / bitsForResiduals;
32 int packed_words = BLOCK_SIZE*BLOCK_SIZE / residuals_per_word;
33 int threadIdxB = threadYB*BLOCK_SIZE+threadXB;
34 int offset = threadIdxB / residuals_per_word;
35 int residualMask = (1 << (bitsForResiduals+1))-1;
36 int bias = (1<<bitsForResiduals-1)-1;
37 int shift_bits = (residuals_per_word-1 -
38                   threadIdxB%residuals_per_word)*bitsForResiduals;
39 int res = ( (residuals[ offset ] >> shift_bits) & residualMask )
40           - bias;
41 decompressed[gridY][gridX] = surface[threadYB][threadXB]+res;
```

6.5 Детаљи имплементације 2DRBUC методе

У овом одељку представљени су детаљи имплементације 2DRBUC методе за декомпресију коначних резидуала (слој 3). У раду [ĐURĐ2013] је анализирана комбинација HFPaC методе са секвенцијалним додатним кораком компресије

(коришћен широко распрострањени програм WinZIP који користи методу DEFLATE). Посебан мотив имплементације описане у овом одељку јесте потврђивање да је могуће извршити додатну компресију резидуала уз ефикасну паралелну декомпресију на GPU. Због тога није разматрана имплементација паралелне компресије на GPU, већ је она извршена секвенцијално, на CPU. Најпре ће бити објашњена једноставна трансформација коју је потребно применити над коначним резидуалима да би 2DRBUC метода могла ефикасно да се примени. Након тога биће објашњена неоптимизована имплементација методе, која се у експериментима показала релативно спором, а затим ће бити објашњене оптимизације којима се донекле поправљају перформансе. Неоптимизована имплементација дата је као илустрација, у циљу једноставнијег разумевања оптимизоване имплементације. У мерењима, чији су резултати дати у поглављу 7, коришћена је оптимизована имплементација методе.

Као што је претходно речено (видети одељак 3.3), RBUC метода оперише над целобројним подацима без знака. С обзиром на то да су резидуали по природи ствари бројеви са знаком, потребно је најпре извршити пресликавање негативних вредности у позитивне. Најједноставније би било тумачити бинарну представу резидуала као целе бројеве без знака. Међутим, с обзиром на то да су резидуали кодирани у комплементу двојке, за представљање вредности -1 би, у том случају, било потребно највише битова. Због груписања вредности резидуала око нуле (видети одељак 4.1), вредност -1 се, поред вредности 0 и 1, најчешће појављује. Неспорно, кодирање податка који се често појављује великим бројем бита негативно би утицало на степен компресије. Још неповољнија ситуација би била у случају представљања резидуала у коду са вишком 2^{w-1} (енг. *bias*), где је w број бита потребан за представљање резидуала. Тада би се три најчешће појављиване вредности међу резидуалима (0, 1 и -1) кодирале истим (0 и 1) или за 1 мањим (-1) бројем бита од броја бита потребних за представљање најдуже кодне речи. Додатно, код за вредност 0 не би био 0, већ вредност вишка, чиме би могућност RBUC методе да уопште не памти потомке унутрашњих чворова који имају вредност 0 (видети одељак 5.6) остала сасвим неискоришћена. Због тога се, у овој имплементацији, резидуали најпре трансформишу кодом који обезбеђује да се позитивне и негативне вредности представе приближно истим бројем бита

[LIND2010], и да код за вредност 0 буде 0. У табели 3 приказано је пресликавање применом овог кода, за вредности које се могу представити на дужини од 4 бита. Поступци кодирања и декодирања дати су у изразима (10) и (11), респективно. У овим изразима, v означава вредност која се кодира, а k одговарајућу кодну реч. У питању су једноставни изрази који користе искључиво целобројну аритметику, што оба поступка чини ефикасним.

$$k = \begin{cases} -2v - 1, & v < 0 \\ 2 \times v, & v \geq 0 \end{cases} \qquad (10)$$

$$v = \begin{cases} -(k \text{ div } 2 + 1), & \text{непарно } k \\ k \text{ div } 2, & \text{парно } k \end{cases} \qquad (11)$$

Табела 3 Код (k) за целе бројеве са знаком (v) који обезбеђује да број бита потребан за представљање кода расте са апсолутном вредношћу кодираног податка. Приказан је код за вредности које се могу представити на дужини од 4 бита.

Код (k)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Вредност (v)	0	-1	1	-2	2	-3	3	-4	4	-5	5	-6	6	-7	7	-8

У имплементацији која је овде описана, 2DRBUC метода примењује се независно над сваким делом поља висина који одговара величини највећег блока нити (32×32). Ово ограничење уведено је због тога што CUDA обезбеђује ефикасну сарадњу нити једино на нивоу блока нити (видети одељак 6.2), али то није концептуално ограничење методе. Код блокова који обрађују само једну врсту или колону ћелија на ивици поља висина, примењује се основна (линеарна, 1D) RBUC метода. С обзиром на величину блока, постоји ограничен број могућности за избор степена гранања различитих нивоа стабла. Експерименталним путем утврђено је да се највећи степен компресије добија када се узме да је степен гранања корена 8^2, и степен гранања првог нивоа стабла 4^2. Висина резултујућег стабла је 2. Треба приметити да висина 2DRBUC стабла не зависи од величине поља висина (или парцеле), зато што се по једно стабло формира за сваки део поља висина обухваћен једним блоком нити.

Имајући у виду да се блокови декодирају независно, позиција почетка података сваког блока (односно одговарајућег стабла) у бит-вектору памти се експлицитно. Када би цело поље висина било обухваћено једним 2DRBUC стаблом, ове позиције би било могуће израчунати (уместо експлицитно памтити), чиме би се

постигао релативно мали добитак у погледу степена компресије. За узврат, на архитектури GPU употребљеног за експерименте у овом раду, било би неопходно покретање посебног програмског језгра за рачунање позиција блокова у бит-вектору, чиме би се изгубило на перформансама.

Након декомпресије прва два слоја података, неоптимизована имплементација 2DRBUC методе најпре прочита вредност у корену стабла, која је запамћена на фиксној дужини од 4 бита. Према дискусији из одељка 5.6, ова дужина је прецењена, али су губици практично занемарљиви, тако да није разматрано одређивање оптималног броја бита за памћење вредности у корену. Вредност у корену стабла истовремено чита 64 нити (пошто је степен гранања корена 64), да би се избегла експлицитна синхронизација између тих нити. Након тога, свака од поменутих 64 нити чита одговарајућу вредност на нивоу 1 стабла и смешта је на одговарајућу позицију у вектор предвиђен за рачунање префиксне суме. Потом, паралелно се рачуна префиксна сума. Након тога, доступна је комплетна информација о позицији сваког листа у бит-вектору и свака нит у блоку може да приступи читању одговарајућег резидуала из глобалне GPU меморије. У овој имплементацији, усвојено је да се на једном нивоу стабла чворови-браћа (чворови који имају заједничког родитеља) смештају по врстама. Као пример, на слици 10 приказана је префиксна сума добијена применом описаног поступка на ниво 1 стабла са слике 7б. Треба имати у виду да је у примеру илустрованом на слици 7 степен гранања корена 16.

2	0	0	0	3	0	1	0	2	0	0	0	4	2	3	2

а)

0	2	2	2	2	5	5	6	6	8	8	8	8	12	14	17

б)

Слика 10 Вектор дужина и одговарајућа префиксна сума; (а) матрица са слике 7б линеаризована по врстама; (б) префиксна сума добијена из вектора (а).

Мерења брзине декомпресије показала су да је описана неоптимизована имплементација, код које се комплетан поступак декомпресије резидуала врши након завршене декомпресије прва два слоја, преко два пута спорија у односу на основну имплементацију методе HFPaC (без додатне компресије резидуала). Мерења су такође показала да читање података различите дужине из бит-вектора у којем је смештено 2DRBUC стабло има значајан удео у додатном времену

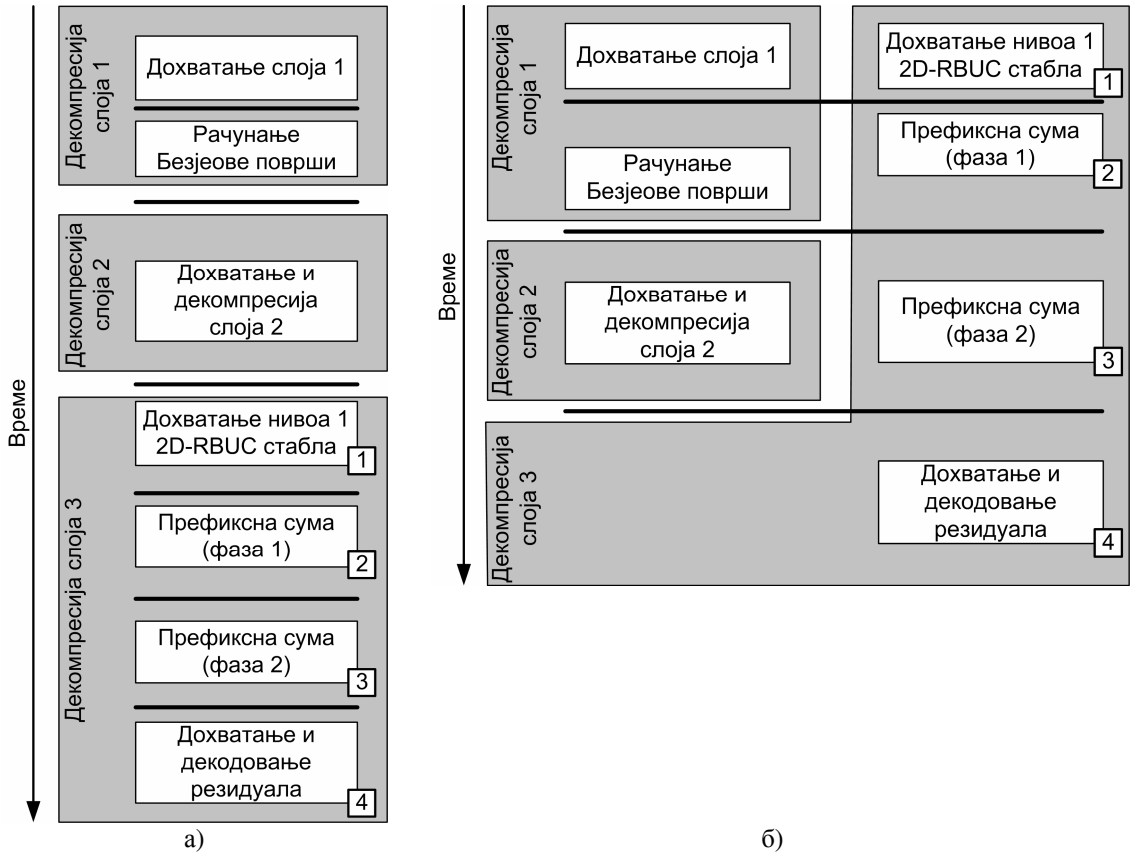
декомпресије. У наставку ће бити описане измене претходног поступка декомпресије, које су реализоване у овој имплементацији, у циљу повећања брзине декомпресије.

Повећање брзине декомпресије остварено је делимичном паралелизацијом декодовања унутрашњих чворова 2DRBUC стабла и декомпресије слојева 1 и 2. Циљ ове оптимизације је да позиције и дужине свих листова 2DRBUC стабла у бит-вектору у којем је смештено стабло буду познате у тренутку када се заврши декомпресија прва два слоја података, односно када на ред дође декомпресија резидуала. Оптимизација је остварена премештањем одговарајућих делова програма на места где је уочен простор за паралелизацију.

Простор за паралелизацију постоји у фази алгоритма у којој се у брзу дељену меморију довлаче подаци из слоја 1, јер тих података има релативно мало, па је мали број нити ангажован на њиховом довлачењу. Неактивне нити се тада ангажују за довлачење комплетног 2DRBUC стабла у брзу дељену меморију. Простор за паралелизацију такође постоји код декомпресије слоја 2, јер истакнутих тачака углавном има релативно мало, па често постоји незанемарљив број нити које тада чекају. Треба напоменути да, у оба уочена простора за паралелизацију, потпуна паралелизација није увек могућа. Наиме, број нити потребан за паралелно довлачење свих података 2DRBUC стабла зависи од оствареног степена компресије. Ако је степен компресије мали, потребан број нити може превазићи број расположивих (неактивних) нити. Зато је довлачење података 2DRBUC стабла реализовано у циклусу, и неке групе нити податке довлаче у две или више итерација. Овим се делимично продужава време потребно за декомпресију слоја 1, али се скраћује укупно време декомпресије. Слично, ако приликом декомпресије истакнутих тачака број неактивних нити није довољан, део декодовања 2DRBUC стабла серијализује се са декомпресијом једног броја истакнутих тачака.

Поред паралелизације, било је неопходно извршити измену у структури 2DRBUC стабла, да би се, што је могуће више, смањио број читања података променљиве дужине. Усвојено је да корен увек садржи вредност 4 (па није потребно памтити га), односно усвојено је да се вредности свих чворова на нивоу 1 стабла

представљају на дужини од 4 бита. Ова дужина је довољна, јер је 8 највећи број бита на којем се представљају коначни резидуали. Негативна последица ове измене структуре стабла јесте смањење степена компресије око 10%, али се тиме убрзава целокупни поступак декомпресије око 60% за највеће разматране парцеле од 1025×1025 тачака (видети одељак 7.1). За исту величину парцела, тиме је успорење у односу на основну методу HFPaC (без додатне компресије резидуала) сведено на око 30%.



Слика 11 Илустрација измена у имплементацији алгоритма за декомпресију уз декомпресију 2DRBUC стабла. Активност која се обавља у датом тренутку означена је у правоугаоницима који представљају одговарајуће блокове инструкција; а) основна имплементација алгоритма; б) оптимизована имплементација алгоритма; блокови инструкција у саставу декомпресије слоја 3 су на исти начин нумерисани на (а) и (б) у циљу једноставнијег уочавања измена; хоризонталне линије између блокова инструкција означавају тачке у којима се врши синхронизација нити. Део декомпресије слоја 3 (блокови инструкција 1 и 3) одвија се физички паралелно са декомпресијом слојева 1 и 2 само ако у том тренутку има довољно слободних нити. Блок инструкција 2 никад се не одвија у потпуности физички паралелно са декомпресијом слоја 1. Нити које извршавају блок инструкција 2 након тога ангажоване су на рачунању Безјеових површи, док остале нити рачунају само Безјеове површи. Због тога су ова два блока инструкција делимично преклопљена на слици.

Описане измене у структури програма илустроване су на слици 11. Слика показује да се премештени блокови инструкција добро уклапају у постојеће тачке синхронизације нити, што има значајан утицај на повећање брзине декомпресије.

Наиме, паралелно рачунање префиксне суме одвија се у две фазе између којих је неопходно извршити синхронизацију нити. С обзиром на то да први ниво стабла има 64 чвора, потребне су 32 нити (односно једна група нити) за рачунање префиксне суме. За то време, све остале нити у блоку нити чекају. Премештањем одговарајућих блокова инструкција обезбеђује се упослење (једног дела) осталих нити. Ово је могуће извести због специфичности CUDA архитектуре, у погледу организације нити у групе нити. Једна група нити врши прву фазу рачунања префиксне суме (од дна ка врху) пре него што рачуна Безјеове површи. Остале групе нити одмах рачунају Безјеову површ. Тиме је укупно време рачунања Безјеове површи продужено занемарљиво мало, јер рачунање траје значајно дуже од прве фазе рачунања префиксне суме. Са друге стране, добит је у томе што су све нити упослене. Слично се постиже премештањем друге фазе рачунања префиксне суме (од врха ка дну), јер је тада одређен број нити, које би иначе чекале, упослен при декомпресији слоја 2. Као што је претходно напоменуто, ако је број истакнутих тачака довољно мали, овде може доћи и до потпуне паралелизације.

7. Резултати и дискусија

У овом одељку најпре ће бити изложени циљ и услови под којима су спроведена мерења перформанси предложене методе. Затим ће бити приказани и анализирани добијени резултати у вези са степеном компресије, грешком апроксимације компресије са губицима и перформансама методе. Биће дискутовани узроци и последице добијених резултата. На крају, биће извршено поређење предложене HFPaC методе са савременим компетитивним приступом [LIND2010].

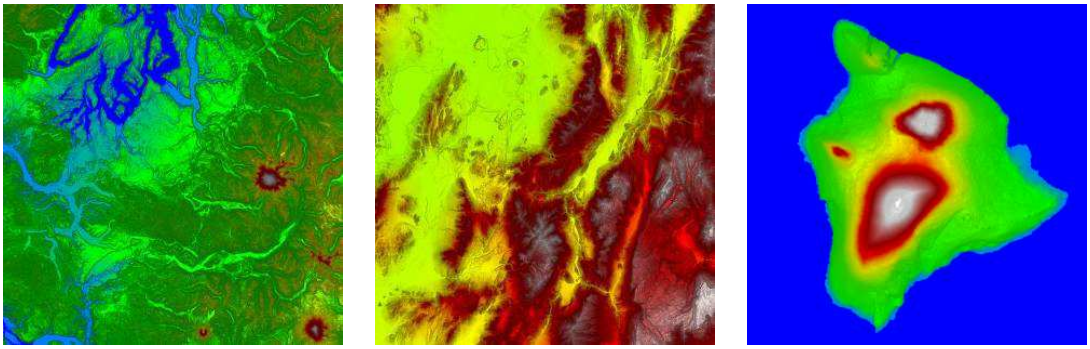
7.1 Циљеви и услови мерења

Први циљ мерења спроведених у оквиру овог рада био је одређивање зависности степена компресије од величине парцеле и сегмента. Други циљ био је процењивање прецизности компресије са губицима. Трећи циљ био је оцењивање брзине предложене методе.

Да би се постигао први циљ, величина парцела и сегмената варирана је на следећи начин. Број врста и колона парцела ограничен је на 2^r+1 тачака, где је r цео број већи од 0. Тај број врста и колона одабран је због тога што се на природан начин уклапа у циљану архитектуру модерних GPU. У спроведеним експериментима, r је варирано у опсегу од 7 до 10, што одговара величинама парцела од 129×129 до 1025×1025 тачака, респективно. Слично томе, коришћени су сегменти величине 2^s+1 тачака, где је s цео број за који важи $s \leq r$. У спроведеним експериментима, s је варирано у опсегу од 2 до 5, што одговара величинама сегмента од 5×5 до 33×33 тачке. Мерење степена компресије је извршено и за случај када се коначни резидуали (слој 3) додатно компримују методом 2DRBUC, као и за случај када се на резултате компресије методом HFPaC примени ZIP компресија. Да би се остварио други циљ, мерен је квалитет апроксимације за компресију са губицима (слој 1) и за компресију са ограниченим губицима (слој 1 + слој 2). Квалитет апроксимације процењен је на основу три индикатора грешке апроксимације: максималне апсолутне грешке, просечне апсолутне грешке и просечне средње квадратне грешке. За остварење трећег циља, спроведена су следећа мерења за све парцеле које чине поље висина, секвенцијално (парцела по парцела), варирајући величину парцела и сегмената. Најпре, мерена су времена компресије и

декомпресије у циљу добијања одговарајућих брзина. Додатно, да би се оценила брзина декомпресије података компримованих са губицима, мерено је време декомпресије слоја 1 и време декомпресије слојева 1 и 2 заједно. Такође, мерено је време декомпресије без губитака када су коначни резидуали из слоја 3 компримовани методом 2DRBUC. Коначно, мерено је време компресије и декомпресије секвенцијалне имплементације која је извршена на CPU, као и укупно време секвенцијалног преноса свих парцела (парцела по парцела), из и у GPU меморију, у циљу процене и илустрације постигнутог убрзања паралелне GPU имплементације наспрам секвенцијалне CPU имплементације. Мерења степена компресије и брзине декомпресије када се користи метода 2DRBUC извршена су у циљу процене расположивог простора за постизање већег степена компресије и његове цене у виду губитка на перформансама.

Код анализе степена компресије и брзина компресије и декомпресије, спроведене су две врсте мерења. У првој врсти мерења, које је означено као *променљив* BPR, дозвољено је да *b*, број бита по резидуалу (енг. *bits per residual*), варира од парцеле до парцеле. За сваку парцелу, изабрана је вредност *b* која даје највећи степен компресије. У другој врсти мерења, које је означено као *фиксан* BPR, иста вредност *b* употребљена је за све парцеле. Ова вредност израчуната је као најмањи цео број из скупа {2, 3, 4, 5, 8} који је већи или једнак просечном *b*, узимајући у обзир вредност *b* за сваку парцелу у првој врсти мерења (као што је објашњено у одељку 6.3). Изузетак је случај где је средња вредност била незнатно већа од вредности из овог скупа: просечна вредност била је 3.07, па је усвојено 3 уместо 4. У мерењима са променљивим BPR, време итеративног рачунања *b* (да би се одредило *b* оптимално за парцелу) укључено је у измерено време компресије. У мерењима са фиксним BPR, вредност *b* израчуната је унапред, а време рачунања није укључено у измерено време компресије. Различити услови мерења брзине компресије за променљив и фиксан BPR коришћени су да би се установила цена рачунања најбоље вредности *b*.



а) Пацет Саунд (16K×16K) б) Јута (32K×32K) в) Хавајци (16K×16K)

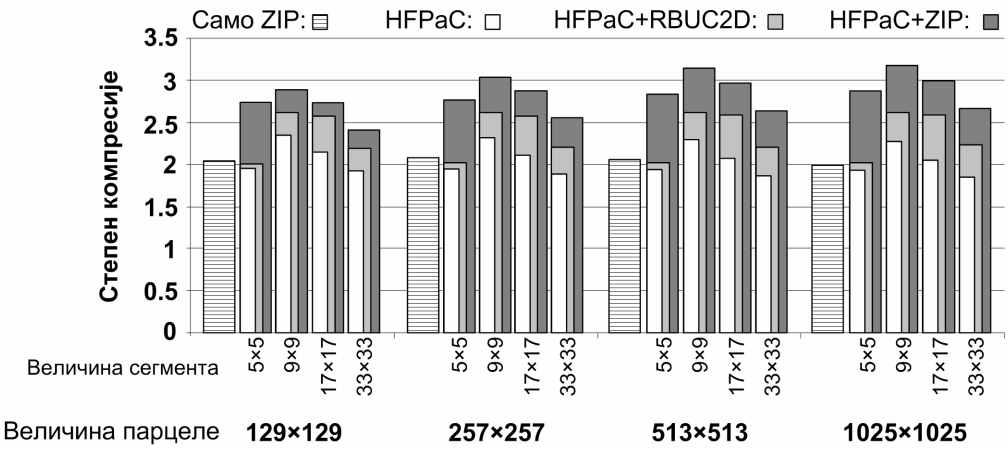
Слика 12 Сенчене рељефне мапе квадратних терена коришћених за одређивање перформанси. Висина је кодирана бојом. У заградама су наведене димензије терена изражене у тачкама. Дужина ивице сваког терена је 163,84 km. Хоризонтална резолуција терена на сликама (а) и (в) је 10 метара по тачки. Резолуција терена на слици (б) је 5 метара по тачки.

У спроведеним експериментима употребљена су три репрезентативна поља висина која потичу од реалних квадратних терена (слика 12). Сва три терена су делови територије САД. Дужина и ширина свих терена је 163,84 km. Терен Пацет Саунд (енг. *Puget Sound*) [PUGE2012], приказан на слици 12а, често се користи у литератури из области компресије терена или визуелизације терена у реалном времену као пример терена сложеног и захтевног за обраду, јер је претежно храпав и садржи мало благих стрмина или равних делова. У питању је део територије савезне државе Вашингтон, јужно од града Сијетл. Терен Јута (енг. *Utah*) [UTAH2011], приказан на слици 12б, део је територије савезне државе Јута, јужно од града Солт Лејк Сити. Обухвата планинске ланце и висоравни. Терен Хавајци (енг. *Hawaii*) [HAWA2012], приказан на слици 12в, представља главно Хавајско острво. Поред копна, у знатној мери обухвата и приобалне водене површи које су потпуно равне (висина 0). На овом терену коришћена је оптимизација за поља висина која садрже равне делове (видети одељак 4.1). Вертикална резолуција за сва поља висина била је 0.125 m. Сваки терен подељен је на парцеле квадратног облика.

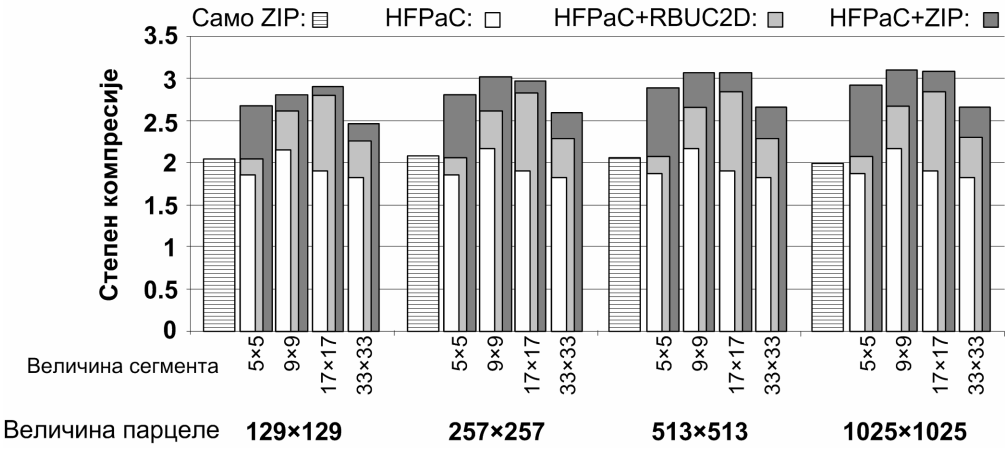
Мерења, чији резултати су приказани у наставку, спроведена су на следећој конфигурацији: Intel i5 2400, 4 GB RAM, NVIDIA GTX 580 са 1.5 GB RAM, Windows XP SP 3. За имплементацију методе коришћена је CUDA 3.2. Такође, спроведен је подскуп експеримената на NVIDIA GTX 280 графичкој картици у циљу квантитативног поређења са компетитивном методом [LIND2010].

7.2 Анализа степена компресије

На сликама 13, 14 и 15 приказани су степени компресије, остварени применом методе HFPaC на терене Пацет Саунд, Јута и Хавај, респективно. Мерење је извршено за различите величине парцела (од 129×129 до 1025×1025) и сегмената (од 5×5 до 33×33). Упоређен је степен компресије остварен искључивом применом ZIP компресора са степеном компресије који остварује предложена метода (HFPaC), предложена метода комбинована са 2DRBUC методом (HFPaC+2DRBUC) и предложена метода комбинована са ZIP компресијом (HFPaC+ZIP). Експерименти показују да величина сегмента 9×9 тачака даје најбоље резултате у свим случајевима. За ову величину сегмента, метода остварује најбољи однос величина простора за смештање слојева 1, 2 и 3.

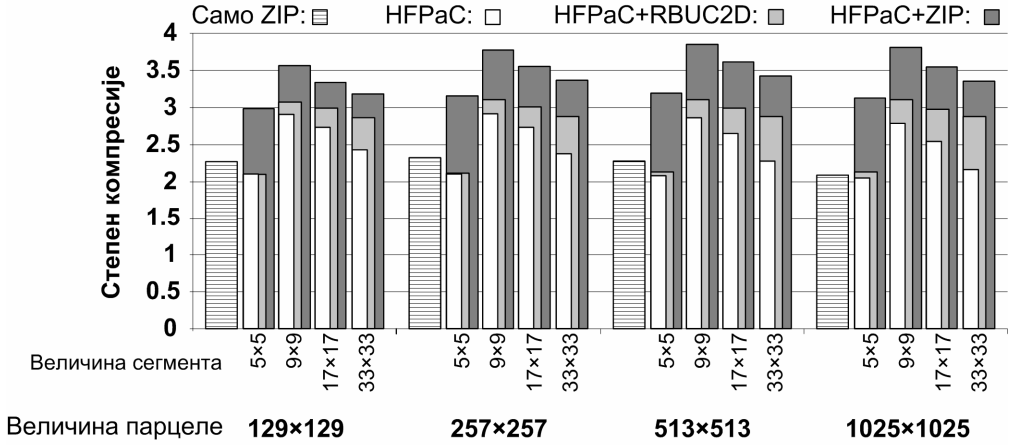


а) Променљив BPR

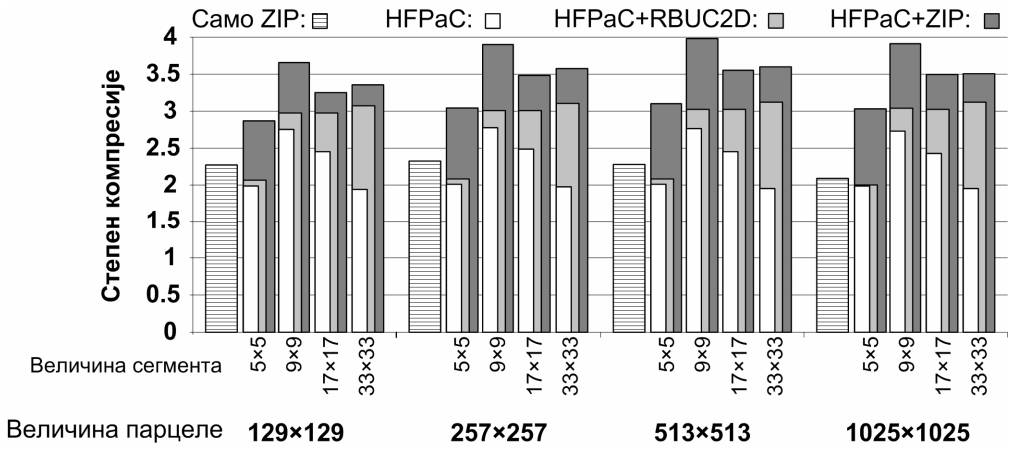


б) Фиксан BPR

Слика 13 Поређење остварених степена компресије за терен Пацет Саунд.



а) Променљив BPR

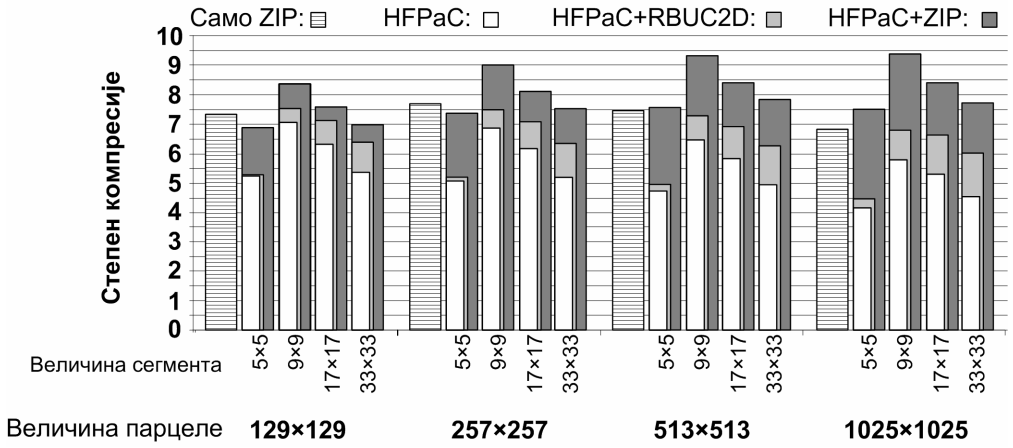


б) Фиксан BPR

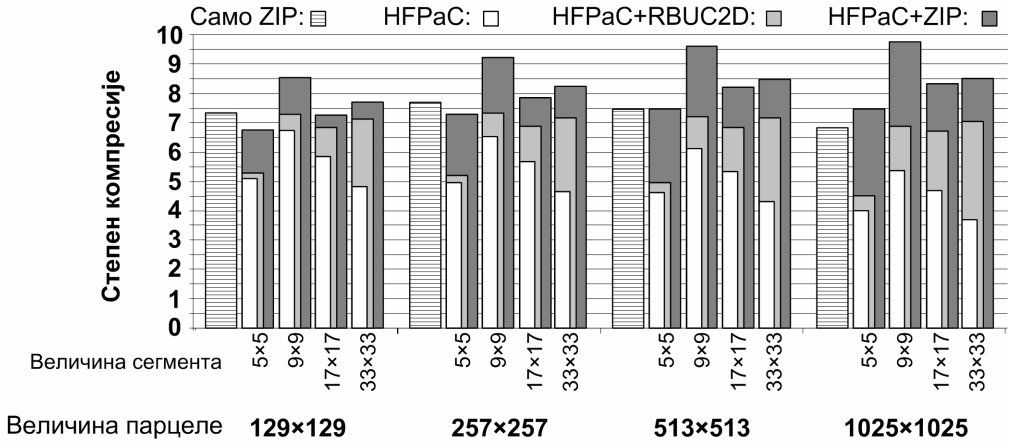
Слика 14 Поређење остварених степена компресије за терен Јута.

У случају терена Пацет Саунд и Јута, метода HFPaC постиже нешто већи степен компресије него директна примена општенаменске методе за компресију без губитака (у овом случају ZIP) на оригинално поље висина. То није случај са тереном Хавајн, који садржи велику равну површ. Међутим, подаци компримовани методом HFPaC могу додатно бити компримовани применом ZIP. На терену Јута (слика 14), за парцеле величине 1025×1025, после примене ZIP компресора на резултате методе HFPaC, добијен је 80% већи степен компресије у односу на компресију када се користи само ZIP. Ово је посебно корисно за архивирање или пренос кроз мрежу мале пропусне моћи. Чак и у случају терена Хавајн (слика 15), где употреба само ZIP компресора даје боље резултате него само HFPaC, комбинација HFPaC+ZIP даје боље резултате од самосталне ZIP компресије за оптималну величину сегмента (9×9), независно од величине парцеле. Такође, у случају већих парцела (513×513 тачака и 1025×1025 тачака),

HFPaC+ZIP увек даје боље резултате од самосталне ZIP компресије, независно од величине сегмента. Међутим, када се додатна метода, попут ZIP, примени на податке компримоване методом HFPaC, декомпресија захтева додатан корак пре могућности декомпресије појединачних тачака.



а) Променљив BPR



б) Фиксан BPR

Слика 15 Поређење остварених степена компресије за терен Хаваји.

Степени компресије остварени применом методе 2DRBUC за додатну компресију коначних резидуала такође су приказани на сликама 13, 14 и 15. Са једне стране, резултати показују да, када се примени ова додатна метода, оптимална величина сегмента није увек 9×9. Приближно исти степени компресије добијају се и са сегментима величине 17×17 тачака, с тим што постоје ситуације када једна од наведених величина сегмената има благу предност. Такође, на теренима Јута и Хавајн, са фиксним BPR, сегменти величине 33×33 тачке показују значајно повећање степена компресије и блиски су степенима компресије постигнутим сегментима величине 9×9 тачака. Са друге стране, за величину сегмента 5×5

повећање степена компресије остварено применом методе 2DRBUC је значајно мање (готово занемарљиво) у односу на друге величине сегмента. Разлог је тај што се са величином сегмента 5×5 постиже најбоља апроксимација у слоју 1, па је за памћење резидуала без додатне компресије потребан мањи број бита по резидуалу у односу на остале величине сегмента (видети табелу 4). Тај број бита је довољно мали да метода 2DRBUC нема простора за оствари значајну компресију. Сагласно томе, добит од примене методе 2DRBUC, код реалних терена, може се очекивати тек за веће вредности броја бита са представљање резидуала *b*. Због тога, оптимална вредност *b*, која је изабрана за мерења степена компресије методе HFPaC (видети одељак 7.1), може бити неоптимална када се за додатну компресију користи метода 2DRBUC. Утврђивање оптималне вредности *b* у том случају није разматрано у овом раду. Даље, резултати показују да комбинација HFPaC+2DRBUC увек даје лошије резултате од комбинације HFPaC+ZIP. У најнеповољнијем случају по комбинацију HFPaC+2DRBUC на терену Хаваји (променљив BPR, парцела 1025×1025 тачака, сегмент 5×5 тачака), степен компресије је око 40% мањи у односу на HFPaC+ZIP. У најповољнијем случају на терену Паџет Саунд (фиксан BPR, парцела 129×129 тачака, сегмент 17×17 тачака), степен компресије је само око 5% мањи. Треба напоменути да ово не имплицира да ZIP врши бољу компресију резидуала од 2DRBUC. Наиме, као што је раније напоменуто (поглавље 4), ZIP врши додатну компресију сва три слоја компримована методом HFPaC, док 2DRBUC врши додатну компресију само слоја 3 (коначни резидуали). Поред тога, за разлику од компресије података применом методе ZIP, компресија резидуала методом 2DRBUC омогућава ефикасан приступ појединачним тачкама, односно омогућава паралелну декомпресију на нивоу тачака.

Може се уочити и да је степен компресије остварен HFPaC методом за променљив BPR је већи него степен компресије за фиксан BPR, углавном до 10%. И ово је очекивано, јер се код промењивог BPR ради са прилагођеним бројем бита за представљање резидуала на свакој плочи индивидуално, док се код фиксног BPR ради са заокруженом просечном вредношћу потребног броја бита по плочама. Међутим, када се примени ZIP, ситуација је обрнута у неким случајевима, па фиксни BPR често има благу предност (5-10%). Слично запажање односи се и на

примену методе 2DRBUC, осим за терен Јута, где променљив BPR углавном има благу предност. На овом терену, једино за сегмент 33×33 фиксан BPR увек даје већи степен компресије од променљивог BPR када се примени метода 2DRBUC. Имајући све ово у виду, може се закључити да је примена фиксног BPR, који би се унапред одредио за неки терен, савим оправдана.

Мерен је допринос сваког слоја података укупној величини компримованих података, усредњен над свим парцелама, за дату величину парцеле. Резултати су приказани у табели 4. Као што се очекивало, резултати показују да са порастом величине сегмента допринос слоја 1 опада, јер се смањује број сегмената па тиме и контролних тачака које се памте некомпримоване, а допринос слоја 3 расте, јер се квалитет апроксимације слојем 1 погоршава. Изузетак (објашњен ниже) уочљив је у случају терена Паџет Саунд, за величину сегмента 33×33 и фиксан BPR. Са друге стране, величину слоја 2 није могуће једноставно предвидети јер зависи од броја истакнутих тачака, који опет зависи од топологије терена и апроксимације слојем 1. За фиксан BPR, величина слоја 2 такође зависи од прецењености *b*, што додатно утиче на величину слоја 3. Значајна прецењеност доводи до малог слоја 2 и великог слоја 3. Ово је нарочито изражено за терен Паџет Саунд, за фиксан BPR и величину сегмента 17×17. Тада, у поређењу са осталим величинама сегмената, усвојена вредност *b* значајно премашује вредност *b* измерену у случају променљивог BPR (8 у односу на 6.7), што резултује значајно мањим слојем 2 и већим слојем 3 код фиксног BPR у односу на случај са променљивим BPR. То објашњава претходно поменут изузетак, који се манифестује на следећој величини сегмента 33×33, где је прецењеност *b* мања (8 у односу на 7.7), па самим тим је и удео слоја 2 већи, а слоја 3 мањи, у односу на величину сегмента 17×17. Због тога, код терена Паџет Саунд, за фиксан BPR, удео слоја 3 није монотоно растући са порастом величине сегмента. Код осталих терена, значајна прецењеност *b* постоји само за највећу величину сегмента употребљену у мерењима, због чега у добијеним резултатима није могуће проверити постојање поменутог изузетка на тим теренима. Величина слоја 1 је доминантна (око 50%) код свих терена само за величину сегмента 5×5, што је очекивано, јер та величина сегмента пружа најбољу апроксимацију Безјеовим површима (што смањује величину слојева 2 и 3) и користи највише контролних

тачака. За величину сегмента 9×9, која је дала најбољи степен компресије код основне HFPaC методе (слике 13, 14 и 15), допринос слоја 3 је око 70% у свим мереним ситуацијама, а допринос слоја 1 је мањи од 20%. Очигледно, даље повећање степена компресије значајно зависи од успешне компресије слоја 3. У табели 4 такође је приказан допринос сваког слоја података када се изврши додатна компресија података из слоја 3 методом 2DRBUC. Уочава се да је глобални однос доприноса слојева непромењен: за величину сегмента 5×5 доминантан је слој 1, а за остале величине сегмента доминантан је слој 3. Ипак, уочава се да је допринос слоја 3 смањен на рачун повећања доприноса слојева 1 и 2, што је и очекивано, јер је додатна компресија извршена једино над слојем 3.

Табела 4 Допринос сваког слоја података (C1-C3) укупној величини након компресије без (основна HFPaC метода) и са (HFPaC+2DRBUC) додатном компресијом коначних резидуала, за променљив BPR и фиксан BPR. Резултати, изражени у процентима, дати су за сваки терен (T), за величину парцеле (P) која даје највећи степен компресије код основне HFPaC методе (слике 13, 14 и 15) и за све величине сегмената (S) које су коришћене у мерењима. Дати број бита по резидуалу (b) усредњен је над свим парцелама за променљив BPR, изузев за терен Хаваји, где су равне парцеле изузете из усредњавања.

T	P	S	променљив BPR							фиксан BPR						
				HFPaC			HFPaC+2DRBUC				HFPaC			HFPaC+2DRBUC		
			b	C1	C2	C3	C1	C2	C3	b	C1	C2	C3	C1	C2	C3
Пацет Саунд	1025×1025	5×5	3.4	48.5	9.6	41.9	50.8	10.1	39.1	4	47.1	5.9	47	52.1	6.5	41.3
		9×9	4.6	14.3	16.1	69.6	16.5	18.5	65	5	13.7	13.5	72.8	16.8	16.5	66.7
		17×17	6.7	3.3	9.7	87	4.2	12.2	83.6	8	3.1	0.8	96.1	4.6	1.2	94.2
		33×33	7.7	0.8	10	89.2	1	12.1	87	8	0.7	7.3	92	0.9	9.1	90
Јута	513×513	5×5	3.1	52.3	7.4	40.3	53.6	7.6	38.8	3	50.2	9.8	40	52.6	10.3	37.2
		9×9	4	18.1	10.5	71.4	19.7	11.5	68.8	4	17.5	13.4	69.1	19.2	14.7	66
		17×17	4.7	4.3	16.4	79.3	4.9	18.5	76.6	5	3.9	14.3	81.7	4.8	17.7	77.5
		33×33	6.5	0.9	9.9	89.2	1.1	12.5	86.4	8	0.8	1.9	97.2	1.3	3.1	95.7
Хавајн	1025×1025	5×5	2.7	52.7	6	41.3	56.9	6.5	36.6	3	50.5	4.7	44.7	57.3	5.3	37.4
		9×9	3.4	18.4	13.1	68.5	21.6	15.4	63	4	17	11.2	71.8	21.9	14.4	63.7
		17×17	4.3	4.3	16.1	79.6	5.4	20.1	74.5	5	3.7	13.5	82.8	5.3	19.4	75.3
		33×33	5.4	0.9	15.9	83.1	1.2	20.9	77.9	8	0.8	1.4	97.8	1.5	2.7	95.8

7.3 Анализа грешке апроксимације

Мерена су три индикатора грешке апроксимације код декомпресије са губицима методе HFPaC: максимална апсолутна грешка (MAX), просечна апсолутна грешка (AVG), и просечна средњеквадратна грешка (RMS). AVG и RMS су усредњени над свим парцелама (за дату величину сегмента). Резултати, нормализовани према опсегу висина за дато поље висина, приказани су у табели 5. Најпре, резултати

Табела 5 Грешке апроксимације слоја 1 (C1) и слојева 1 и 2 заједно (C1+C2), за променљив BPR и фиксан BPR, нормализоване према опсегу висина датог поља висина. Опсег висина сваког поља висина дат је у метрима, поред назива поља висина. Индикатори грешке апроксимације су: максимална апсолутна грешка (MAX), просечна апсолутна грешка (AVG) и просечна средњеквадратна грешка (RMS). Резултати су приказани за сваки терен (T), за сваку парцелу (P) која даје највећи степен компресије (слике 13, 14 и 15), и за све величине сегмената (S) коришћене у претходним мерењима. Вредности су усредњене над свим парцелама, осим за терен Хаваји, где су равне парцеле искључене из усредњавања.

T	P	S	C1			C1+C2, променљив BPR			C1+C2, фиксан BPR		
			MAX [$\times 10^{-2}$]	AVG [$\times 10^{-5}$]	RMS [$\times 10^{-5}$]	MAX [$\times 10^{-2}$]	AVG [$\times 10^{-5}$]	RMS [$\times 10^{-5}$]	MAX [$\times 10^{-2}$]	AVG [$\times 10^{-5}$]	RMS [$\times 10^{-5}$]
Пацет Саунд (4393 m)	1025×1025	5×5	1.88	3.87	7.91	0.04	2.87	4.50	0.02	2.99	4.87
		9×9	2.72	11.89	21.20	0.36	8.54	12.49	0.04	8.37	12.41
		17×17	3.44	33.66	55.06	0.36	30.13	45.36	0.36	32.86	52.19
		33×33	4.02	86.62	131.72	0.36	72.07	100.39	0.36	71.88	101.41
Јута (2334 m)	513×513	5×5	2.30	4.71	8.73	0.08	3.86	6.11	0.02	3.43	5.30
		9×9	3.22	10.66	16.87	0.68	9.10	12.64	0.04	8.25	11.41
		17×17	2.96	22.44	33.95	0.68	17.67	23.78	0.08	17.08	23.14
		33×33	3.31	52.43	77.07	0.68	48.90	68.55	0.68	52.06	75.78
Хаваји (4200 m)	1025×1025	5×5	1.10	1.99	4.02	0.04	1.61	2.47	0.01	1.58	2.44
		9×9	1.64	4.85	9.79	0.04	3.54	5.18	0.02	3.33	5.03
		17×17	2.01	12.02	22.71	0.38	8.99	13.24	0.04	7.80	11.52
		33×33	3.76	28.33	48.01	0.38	23.21	33.51	0.38	23.51	39.17

показују да сви индикатори грешке апроксимације слоја 1 очекивано расту са порастом величине сегмента, са изузетком једног случаја када се MAX грешка смањи на терену Јута, за величину сегмента 17×17. Уочен изузетак не може се другачије објаснити него као случајност. Тенденција раста грешке такође је присутна када се слој 2 укључи у декомпресију, али највећа апсолутна грешка је тада ограничена по природи ствари, јер метода то обезбеђује (видети одељак 4.1). Затим, декомпресија слоја 2 редукује средњу апсолутну грешку између 10% и 30%. Фиксан BPR углавном даје нешто бољу апроксимацију. Коначно, декомпресија са ограниченим губицима C1+C2 са фиксним BPR и величином сегмента 9×9, која постиже најбољи степен компресије код основне HFPaC методе, даје максималну апсолутну грешку мању од 2 m, средњу апсолутну грешку мању од 0.4 m, и просечну средњеквадратну грешку мању од 0.6 m на

терену Пацет Саунд. На остала два терена, ове грешке су мање од 1 m, 0.2 m и 0.3 m, респективно.

Треба приметити да је HFPaC у основи осмишљен као метода компресије без губитака. Ипак, њени деривати обезбеђују компресију са губицима или са ограниченим губицима, са максималном грешком ограниченом на 2^{b-1}, где је b број бита за представљање резидуала из слоја 3. Ова особина не чини HFPaC компетитивном са другим методама компресије са ограниченим губицима, јер степен компресије, за упоредиву RMS грешку, није већи од степена компресије других метода. За разлику од неких метода компресије са ограниченим губицима, HFPaC не може да изврши компресију података са циљаном RMS грешком, али HFPaC може да циља максималну грешку. Грубо поредећи, уз напомену да грешка апроксимације зависи од терена и да услови мерења нису исти за методу HFPaC и методу [BETT2007], метода [BETT2007] постиже степен компресије 1:70 (RMS грешка 1 m и непозната MAX грешка) на терену Сардинија. HFPaC са слојевима 1 и 2 постиже степен компресије 1:27 (RMS грешка 0.47, MAX грешка 1.875 m и величина сегмента 17×17), и степен компресије 1:169 (RMS грешка 1.65 m, MAX грешка 15.875 m, и величина сегмента 33×33) на терену Хавајн, када се равне парцеле искључе из усредњавања. Дати степени компресије за HFPaC добијени су без употребе додатне методе компресије без губитака (ZIP).

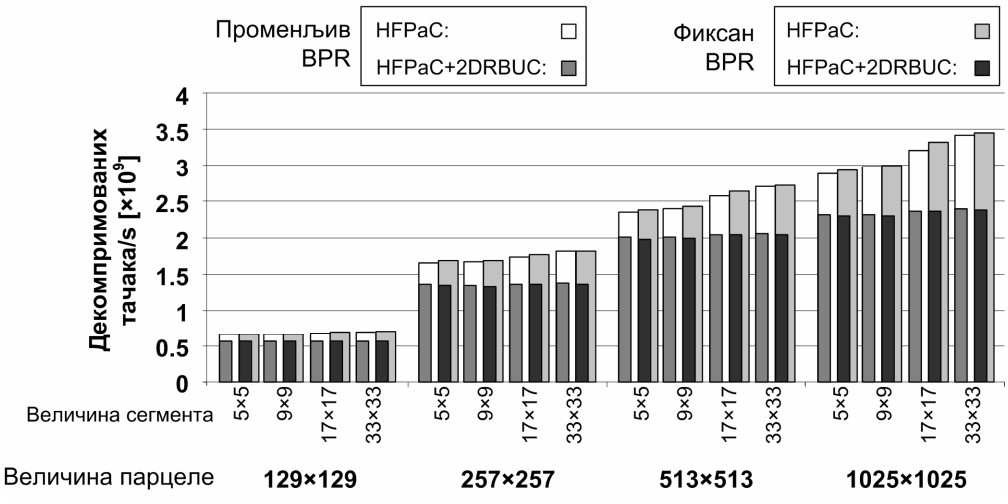
7.4 Анализа перформанси

Брзине компресије и декомпресије које остварује метода HFPaC приказане су на сликама 16, 17 и 18, за терене Пацет Саунд, Јута и Хавајн, респективно. Мерење перформанси је извршено за исте величине парцела (од 129×129 до 1025×1025) и сегмената (од 5×5 до 33×33) као и мерење степена компресије. Анализиране су брзине компресије и декомпресије предложене методе (HFPaC) за променљив и фиксан број бита за представљање резидуала. Такође, анализирана је и брзина декомпресије уз примену методе 2DRBUC за додатну компресију резидуала.

На сликама се недвосмислено уочава да је метода асиметрична, јер је декомпресија значајно бржа од компресије. Ова особина нема негативну конотацију, јер је потреба за брзом декомпресијом начелно много већа од потребе за брзом компресијом. Брзине компресије и декомпресије повећавају се са

повећањем величине парцеле. Ова појава је очекивана, јер више паралелизма може бити примењено на већим парцелама. Са друге стране, режијско време позивања CUDA програмских језгара постаје мање значајно. Слика 19 приказује однос режијског времена позивања CUDA језгара и времена извршења појединачних фаза методе. На најмањим парцелама (129×129), занемарљив је утицај величине сегмента на брзину компресије и декомпресије, независно од променљивости BPR, што се може објаснити слабо искористивим паралелизмом.

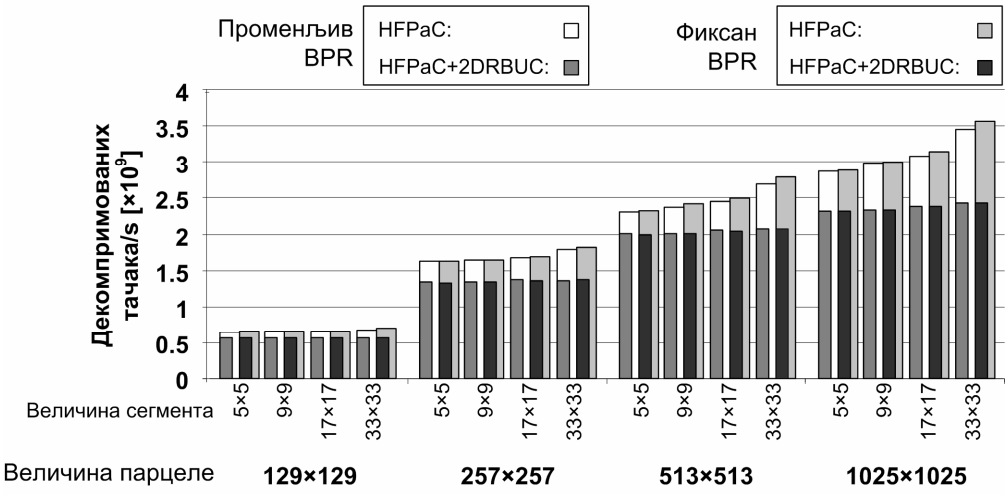
а) Брзина компресије



б) Брзина декомпресије

Слика 16 Поређење брзина компресије и декомпресије терена Пацет Саунд.

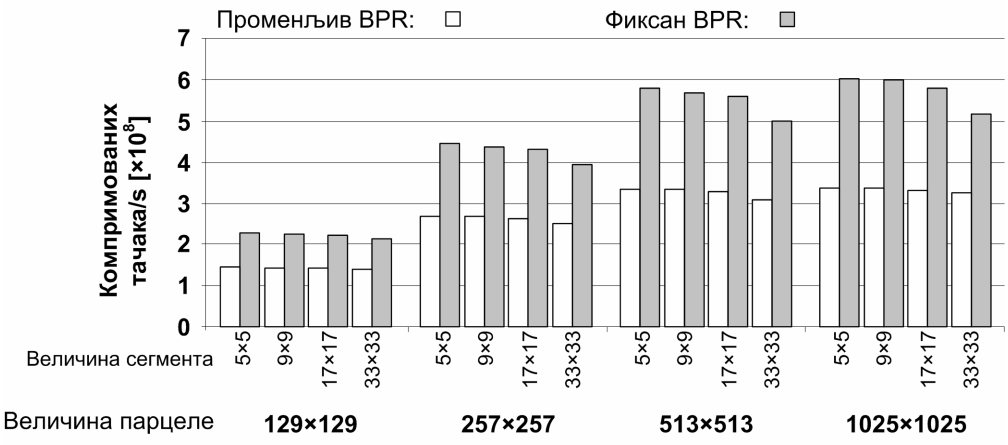
а) Брзина компресије



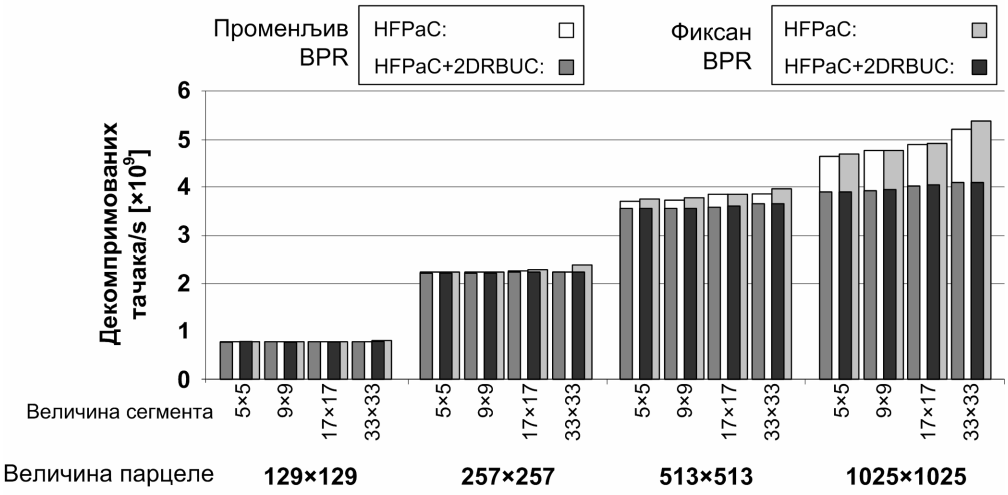
б) Брзина декомпресије

Слика 17 Поређење брзина компресије и декомпресије терена Јута.

Због времена потребног за рачунање оптималне вредности b, метода постиже значајно веће брзине компресије за фиксан BPR. Брзина компресије са променљивим BPR показује мали степен зависности од величине сегмента, иако се може уочити тенденција благог смањења брзине са порастом величине сегмента. Не запажа се да је овакво понашање узроковано карактеристикама поља висина (односно терена). Са друге стране, за фиксан BPR, ова тенденција је изражена, нарочито за веће парцеле. Објашњење за овакво понашање лежи у чињеници да су коефицијенти c_i, вектор $\boldsymbol{C}_e$ и нарочито вектор $\boldsymbol{C}_c$, чије дужине расту са порастом величине сегмента (видети одељак 5.2), смештени у глобалну (спору) GPU меморију. За мање величине сегмената, више нити у једном блоку нити истовремено дохватају и користе исте коефицијенте и елементе вектора, па су стога ефикаснији у дохватању дељених података. Овај ефекат није толико



а) Брзина компресије



б) Брзина декомпресије

Слика 18 Поређење брзина компресије и декомпресије терена Хаваји.

истакнут за променљив BPR, зато што је, у великој мери, прикривен одређивањем оптималне вредности *b*.

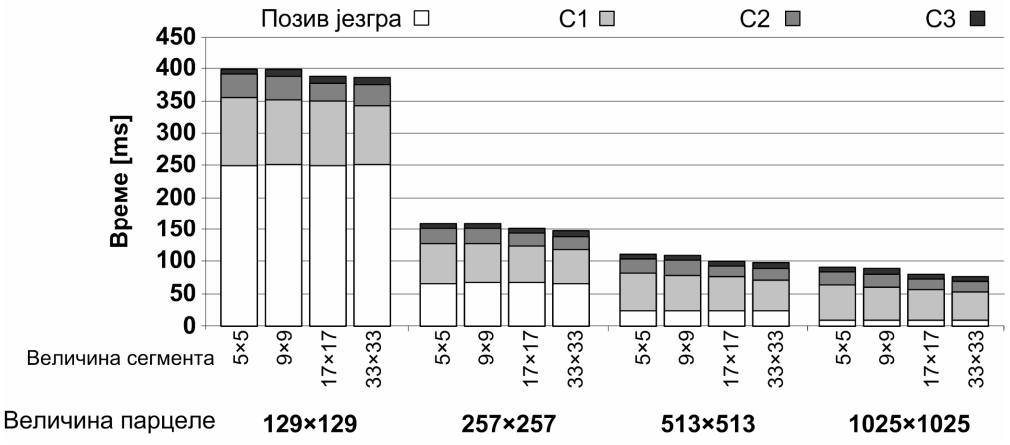
Брзина декомпресије показује малу зависност од тога да ли је BPR променљив или фиксан, а фиксан BPR има благу предност. Насупрот компресији, брзина декомпресије повећава се са порастом величине сегмента. Додатна мерења, приказана на слици 19, указују на то да се добитак на брзини декомпресије за веће сегменте претежно појављује код декомпресије Безјеових површи (слој 1). Ово је очекивано, зато што сегменти величине 5×5 тачака захтевају око 32 пута више контролних тачака за покривање истог дела поља висина које покрива један сегмент величине 33×33 (289 тачака насупрот 9 тачака). Довлачење већег броја података одједном производи већи број конфликтних приступа меморијским

банкама и узрокује успорење декомпресије, јер у предложеној имплементацији методе HFPaC није оптимизован редослед приступа меморији.

Брзине компресије и декомпресије терена Пацет Саунд и Јута показују велику сличност. Међутим, одговарајуће брзине компресије и декомпресије терена Хавај показују нека одступања. Најпре, разлика између брзине компресије за фиксан BPR и променљив BPR је мања него код друга два терена. Такође, укупна брзина компресије је око 10% мања него код друга два терена. Са друге стране, брзина декомпресије је најмање 10% већа. Узрок овој појави је оптимизација за равне парцеле (видети одељак 4.1), која се користи само на терену Хавај. Током компресије, сваку парцелу најпре треба испитати да ли је равна, што неизбежно доводи до смањења брзине компресије у случају терена Хавај. Али, када се врши декомпресија, равне парцеле не захтевају довлачење података нити рачунање, што доводи до значајног повећања у брзини декомпресије. Такође, у случају терена Хавај, повећање брзине декомпресије са порастом величине сегмента није занемарљиво само за највеће парцеле (1025×1025), што није случај са теренима Пацет Саунд и Јута, где је повећање брзине декомпресије са порастом величине сегмента уочљиво и за мање парцеле (осим за најмање, 129×129). Узрок ове појаве је повећање доступног паралелизма са порастом величине плоче. На терену Хавај, ова појава је прикривена великим бројем малих равних парцела. Појава се прогресивно појављује како расте величина парцеле, јер опада број потпуно равних парцела.

Када се за додатну компресију коначних резидуала користи 2DRBUC метода, брзина декомпресије је, очекивано, мања. Смањење брзине је врло мало (занемарљиво у случају терена Хавај) за парцеле величине 129×129 тачака, и прогресивно расте до око 30% за парцеле величине 1025×1025 тачака и величину сегмента 33×33. Може се приметити да је утицај променљивости BPR на брзину декомпресије још мање изражен него код основне методе, а променљив BPR има незнатну предност. Такође се може приметити да брзина декомпресије углавном зависи од величине парцеле, док је утицај величине сегмента практично занемарљив. Благи пораст брзине декомпресије са порастом величине сегмента уочљив је једино на терену Хавај, за парцеле величине 1025×1025 тачака. Тада, због великог броја равних приобалних површина, које није потребно памтити у

2DRBUC стаблу (видети одељак 5.6), метода 2DRBUC брже врши декомпресију стабла него у осталим случајевима, па утицај брзине декомпресије осталих слојева (а која зависи од величине сегмента) постаје уочљивији.



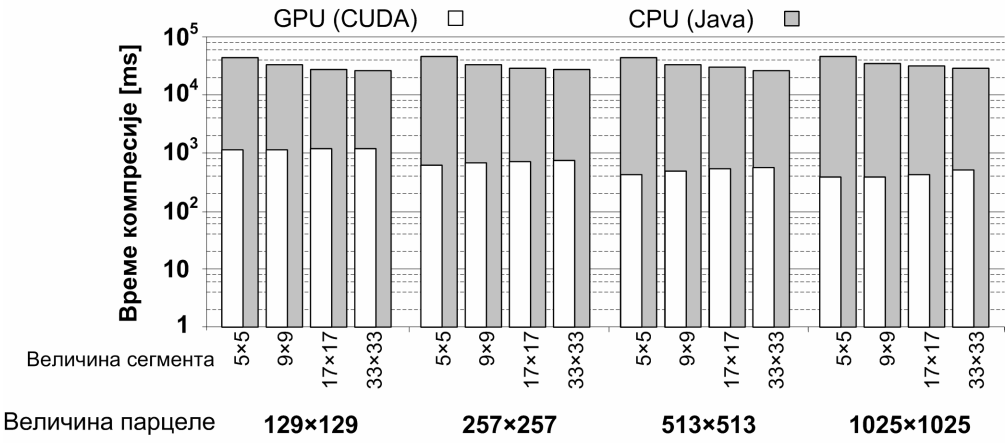
Слика 19 Детаљна времена декомпресије по слојевима C1, C2 и C3, укључујући и време позива CUDA програмског језгра, за терен Пацет Саунд, фиксно BPR.

Укупно време декомпресије терена Пацет Саунд, као и удео времена позива CUDA језгра и времена декомпресије слојева у укупном времену, за фиксан BPR, приказан је на слици 19. Да би се измерило време позива CUDA језгра, уметнута је наредба `return` као прва наредба језгра. Посебно је мерено време декомпресије за слој 1 (C1), слојеве 1 и 2 (C1+C2), и сва три слоја. Резултати показују да укупно време позива CUDA језгара, апсолутно и релативно у односу на укупно време, опада са порастом величине парцеле. Око 60% времена декомпресије троши се на позив CUDA језгра за парцеле величине 129×129 тачака, и само око 10% за парцеле величине 1025×1025 тачака. Понашање је очекивано, с обзиром на то да је код већих парцела искоришћење паралелизма веће, те се мањи број пута покрећу језгра.

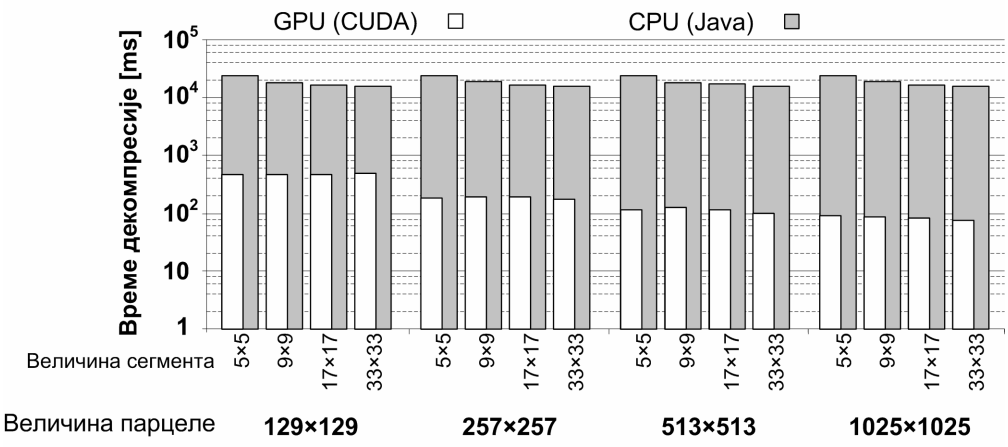
Резултати показују да се највећи део корисног времена декомпресије (када се не узме у обзир време позива језгра) троши на декомпресију слоја 1, што не представља изненађење с обзиром на број рачунских операција потребних за декомпресију слоја 1. Допринос слоја 3 овом времену је увек најмањи. Време декомпресије слоја 1 опада са порастом величине парцеле, јер је за веће парцеле доступно више паралелизма и већа је искоришћеност рачунских јединица (GPU језгара). Наиме, у предложеној имплементацији (видети одељак 6.3) користе се блокови од 32×32 нити, па је се декомпресија парцела величине 129×129 тачака

врши употребом 25 блокова. Од тога, само 16 блокова (64%) у потпуности ангажује свих 1024 нити. Од преосталих 9 блокова, 8 врши декомпресију само једне врсте (односно колоне) парцеле, чиме је ангажовано само 32 од могућих 1024 нити по блоку, а један блок врши декомпресију само једне (фиксне) тачке, за шта је ангажована само једна нит. За веће парцеле, удео блокова који потпуно ангажују све нити је већи, и износи 79%, 89% и 94%, за парцеле величине 257×257, 513×513 и 1025×1025 тачака, респективно. Може се уочити и блага тенденција опадања времена декомпресије слоја 1 са порастом величине сегмента, за дату величину парцеле, што је последица опадања броја тачака које описују Безјеове површи, а које треба дохватити из глобалне GPU меморије приликом декомпресије. Време декомпресије слојева 2 и 3 не показује правилност промене са порастом величине парцеле или сегмента, зато што та времена зависе од више фактора, као што су број истакнутих тачака и број бита за представљање коначних резидуала, што је објашњено у одељку 7.2.

У циљу стицања грубе слике добити у перформансама GPU имплементације у односу на CPU имплементацију, поређене су брзине секвенцијалне имплементације HFPaC на CPU, са паралелном GPU имплементацијом. За секвенцијалну CPU имплементацију, коришћен је програмски језик Јава, верзија 1.6.0 (update 24). Резултати за терен Паџет Саунд приказани су на слици 20. Паралелна CUDA компресија је бржа између 20 и 120 пута за величине парцела 129×129 тачака и 1025×1025 тачака, респективно. За исте величине парцела, декомпресија је бржа између 30 и 260 пута. Ово недвосмислено показује предност SIMD паралелне имплементације методе HFPaC над секвенцијалном CPU имплементацијом.



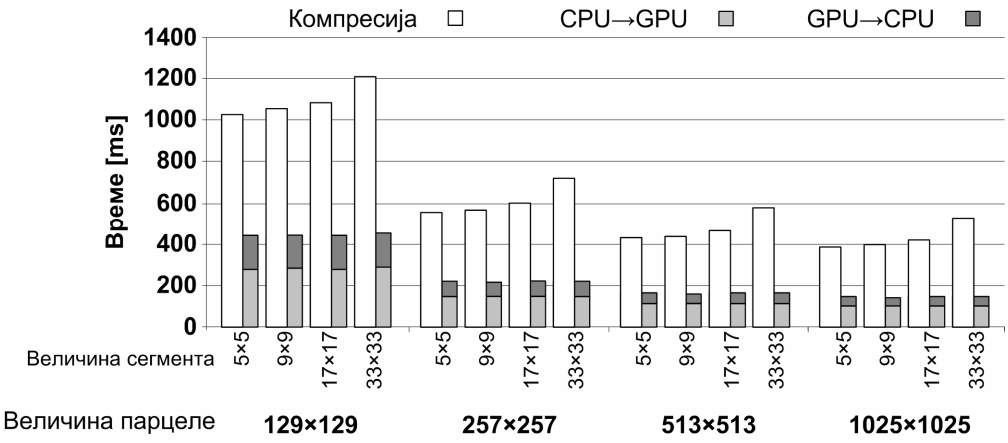
а) Време компресије



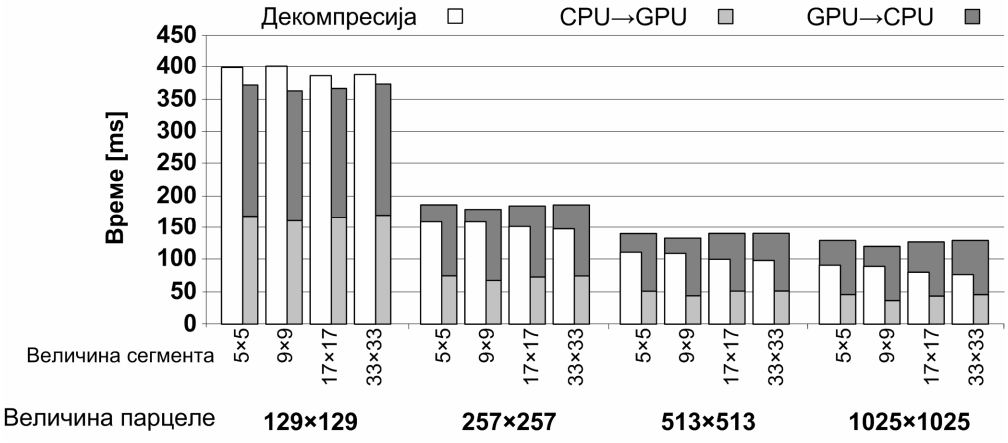
б) Време декомпресије

Слика 20 Поређење времена секвенцијалне (CPU) и паралелне (GPU) имплементације за терен Пацет Саунд; временска оса је логаритамска.

У циљу стицања потпуног увида у цену употребе GPU уместо CPU за компресију и декомпресију, такође је мерено време потребно за пренос података, некомпримованих и компримованих, из оперативне (CPU) меморије у GPU меморију, и обратно. Добијени резултати за терен Пацет Саунд приказани су на слици 21. Резултати показују да укупно време потребно за секвенцијални пренос парцела (парцела по парцела) није занемарљиво у поређењу са временом потребним за секвенцијалну компресију терена, такође парцела по парцела. Ситуација је још неповољнија код декомпресије, зато што је, за све величине парцела веће од 129×129, време потребно за пренос података веће од времена потребног за декомпресију. Међутим, с обзиром на то да CUDA подржава асинхроне преносе података, који могу бити обављени паралено са извршењем програмског језгра [NVID2010], велики део преноса може бити преклопљен у



а) Пренос некомпримованих података, компресија и враћање компримованих података



б) Пренос компримованих података, декомпресија и пренос декомпримованих података

Слика 21 Поређење укупног времена компресије (а) и декомпресије (б) са временом преноса одговарајућих података (из CPU у GPU меморију и из GPU у CPU меморију) за терен Пацет Саунд.

времену (симултан) са декомпресијом, чиме би се ефективно сакрио одговарајући део времена преноса. Напредна имплементација методе може извршити пренос не једног, већ већег броја парцела, у једном великом пакету. Таква имплементација би ефикасније искористила пропусну моћ магистрале података и преклопила пренос са декомпресијом у што је могуће већој мери. На пример, резултати на слици 21б указују на то да треба очекивати да време потребно за пренос четири парцеле величине 257×257 тачака у виду једне парцеле од 513×513 тачака буде мање од времена потребног за секвенцијалну декомпресију четири парцеле величине 257×257 тачака. Закључак се доноси на основу чињенице да исти терен има 4 пута више парцела величине 257×257 тачака него парцела величине 513×513 тачака, те су посматрани резултати са слике 21б за парцеле различитих

величина директно поредиви. Додатно, на напредним графичким адаптерима, CUDA такође подржава симултане бидирекционе преносе података. Употребом ове технике, пренос података би могао да буде у потпуности прикривен извршењем програмског језгра. Такође, треба приметити да, у хипотетичкој геоинформатичкој апликацији која би користила предложену методу, анализа терена на GPU може да се врши непосредно након декомпресије. Таква апликација уопште не би враћала у CPU меморију декомпримовану парцелу, већ само резултате анализе терена. У том случају, сви преноси података могли би бити сакривени извршењем језгара (декомпресије и анализе) чак и без претходно поменуте напредне имплементације или двосмерних преноса.

7.5 Поређење са компетитивном методом

У овом одељку биће извршено поређење методе HFPaC са методама у којима се разматра компресија без губитака за декомпресију која се врши на GPU (видети одељак 3.2). И поред тога што су измерене перформансе приступа [ZHAN2011] поредиве са [LIND2010] или HFPaC, нису сасвим компетитивне. У [ZHAN2011] аутори користе NVidia Tesla C2050 GPU, са 448 CUDA скаларних процесора, док аутори [LIND2010] користе NVidia GTX 280 GPU, са 240 CUDA скаларних процесора. Према томе, на GPU са приближно двоструко већом процесорском снагом, декомпресија у [ZHAN2011] је око 10% спорија у односу на [LIND2010]. Осим тога, у [ZHAN2011] нису објављени постигнути степени компресије. Због тога, поређење ефикасности приступа [ZHAN2011], са једне стране, и приступа [LIND2010] и HFPaC, са друге, било би вршено под непознатим условима. У складу са тим, овде ће се једино приступ [LIND2010] посматрати као компетитиван приступ методи HFPaC.

Квантитативно поређење савременог компетитивноног приступа [LIND2010] са методом HFPaC показује да метода HFPaC постиже мањи степен компресије од [LIND2010], али има потенцијал да постигне поредив степен компресије у комбинацији са методом која је намењена додатној компресији података. У комбинацији са ZIP, HFPaC остварује сличне степене компресије као и [LIND2010] за терен Пацет Саунд (око 1:3.2), али мањи (иако поредив) степен компресије за терен Хаваји (око 1:9.8 насупрот 1:10.7). У комбинацији са методом

2DRBUC за компресију коначних резидуала које произведе метода HFPaC, постиже се мањи степен компресије (око 1:2.8 за терен Пацет Саунд, односно 1:7 за терен Хаваји). Мањи степен компресије је очекиван због тога што се, за разлику од ZIP компресије која се примењује на све податке, 2DRBUC компресија примењује само на коначне резидуале. Приликом поређења степена компресије које остварују методе HFPaC и [LIND2010], треба имати у виду и то да метода [LIND2010] памти само резидуале, док резидуали представљају само један од укупно три слоја података које памти метода HFPaC, што је цена концептуалних предности које метода HFPaC има над методом [LIND2010].

Треба напоменути да директно поређење степена компресије може довести у заблуду. Наиме, степени компресије објављени у [LIND2010] дати су за комплетну хијерархију терена са више резолуција, док су мерења за методу HFPaC спроведена једино за најфинији ниво детаља. Примена методе HFPaC на остале нивое детаља показује да укупан степен компресије опада. Анализа степена компресије по нивоима показала је да се компресија остварује на првих неколико (2 до 3) нивоа хијерархије, док на осталим нивоима долази до експанзије, односно резултати методе HFPaC су већи од улазних података. Разлог томе јесте мала просторна корелација података на вишим нивоима. Када је просторна корелација између тачака терена мала, Безјеове површи не могу квалитетно да апроксимирају терен. Лоша апроксимација терена за собом повлачи велики број истакнутих тачака и велику вредност b, што имплицира велике слојеве 2 и 3. Коначно, треба имати у виду да је терен у [LIND2010] представљен као 4-8 хијерархија мрежа троуглова, док је за HFPaC погоднија репрезентација кватернарним стаблом, која је значајно мања у величини од 4-8 хијерархије. Због тога, када је терен са више резолуција у питању, бољи степен компресије који остварује метода [LIND2010] не имплицира и мању величину компримованих података у поређењу са величином компримованих података коју производи метода HFPaC примењена на поље висина са више резолуција.

Теоретски посматрано, ефикасност декомпресије методе HFPaC поредива је са ефикасношћу методе [LIND2010], у оптималном случају за сваку од метода. Ред временске сложености обе методе, у одговарајућем оптималном случају за сваку од метода, је O(1). Код обе методе оптималан случај настаје када је број

рачунских јединица (GPU језгара) C једнак броју тачака поља висина T (T=C), јер се тада постиже идеалан паралелизам. У општем случају, тачке поља висина су организоване у P парцела, тако да свака парцела садржи T/P тачака. У методи [LIND2010], да би се постигао идеалан паралелизам, свака јединица треба да декомпримује по једну парцелу (P=C), што значи да једна јединица секвенцијално декомпримује T/C тачака. Због тога, у наведеном оптималном случају (T=C), сложеност [LIND2010] је O(T/C)=O(T/T)=O(1). Међутим, треба запазити да у том случају свака парцела садржи само по једну тачку поља висина, што је неприхватљиво ограничење у пракси. Као што је објашњено у одељку 5.5, код методе HFPaC сложеност у оптималном случају је такође O(1). При томе није од посебног значаја да ли су све тачке поља висина распоређене у више или само једну парцелу, јер метода предвиђа и паралелну обраду више парцела.

Кључна разлика између ова два приступа је у томе што HFPaC користи паралелизам на нивоу тачака, док [LIND2010] користи паралелизам на нивоу парцела. У питању је концептуална, а не имплементациона предност методе HFPaC. Ова предност може постати и практична (под условом да је C<<T, што је тачно за савремене GPU и величине поља висина) ако C расте, а то је очекивана тенденција у будућности. Како C расте, у [LIND2010] P мора да расте да би остварио оптималну ефикасност, а у методи HFPaC P може да опада, јер број тачака по парцели може да расте заједно са C, али не мора јер се и више парцела може обрадити паралелно. Пораст P у [LIND2010] доводи до смањења величине парцела, што води ка смањењу степена компресије. Са друге стране, смањење P код методе HFPaC доводи до повећања величине парцела, што нема битан утицај на степен компресије, као што је показано на сликама 13, 14 и 15 (у неким ситуацијама, степен компресије HFPaC+ZIP је благо већи за веће величине парцела). Суштински, у [LIND2010] C је ограничен на вредност значајно мању од T, због најмање разумне величине парцеле, док HFPaC нема такво ограничење.

Иако декомпресија у методи [LIND2010] захтева мање рачунских операција по тачки у односу на HFPaC, измерена ефикасност декомпресије значајно зависи од оптерећења, односно од броја парцела које треба декомпримовати. Конкретно, метода [LIND2010] мора да декомпримује велики број парцела, које садрже више од 10 милиона тачака, да би се приближила брзини коју HFPaC постиже са само 1

милионом тачака на теренима Пацет Саунд и Јута. На савременим GPU, са ограниченим бројем GPU језгара, ова предност методе HFPaC губи на значају са повећањем оптерећења и, за одређен број парцела (тачака), метода [LIND2010] се изједначава по перформансама са методом HFPaC. Додатно, треба напоменути да већи број рачунских операција по тачки не смањује значајно перформансе, јер су модерни GPU оптимизовани за интензивна рачунања са реалним бројевима у покретном зарезу. На GPU који је коришћен у [LIND2010], NVidia GTX 280, HFPaC достиже брзине декомпресије од око 2.14 милијарди тачака у секунди за терене Пацет Саунд и Јута, са парцелама величине 1025×1025 и сегментима величине 17×17 тачака, и око 2.92 милијарде тачака по секунди за терен Хаваји, са истим величинама парцела и сегмената. Насупрот томе, најбољи резултати у [LIND2010] у брзини декомпресије су нешто мањи од 2 милијарде тачака у секунди. Са друге стране, када се примени 2DRBUC метода за компресију коначних резидуала, највећа процењена брзина декомпресије на парцелама 1025×1025 на NVidia GTX 280, добијена скалирањем на основу измерених перформанси NVidia GTX 280 и NVidia GTX 580, је око 1.5 милијарди тачака, што је приближно и брзина коју постиже метода [LIND2010] за исти број тачака. Коначно, није могуће поредити брзину компресије методе HFPaC са методом [LIND2010], зато што у [LIND2010] није разматрана паралелна компресија поља висина.

7.6 Резиме анализе резултата

У закључку овог поглавља треба приметити да HFPaC показује обећавајуће брзине компресије и декомпресије, као и добар потенцијал за постизање степена компресије, који су поредиви са савременом компетитивном GPU методом. Инхерентан SIMD паралелизам омогућава методи HFPaC да надмаши компетитивну методу у брзини компресије и декомпресије без губитака, нарочито за мања поља висина. Поља висина компримована методом HFPaC могу успешно бити додатно компримована употребом познатих компресора опште намене без губитака, као и употребом метода специјализованих за компресију резидуала, које омогућавају њихову брзу паралелну декомпресију на GPU. У раду је развијена

једна таква метода (2DRBUC) којом су демонстриране могућности додатне компресије, али је отворен и нови простор за даља истраживања.

8. Примена предложене методе

У овом поглављу разматра се практична применљивост методе HFPaC као дела система за визуелизацију поља висина. Овај систем развијен је у оквиру ширег истраживања које неће бити детаљно описано, већ само његови најзначајнији делови. Најпре ће бити изложена суштина методе *Domino Tiling* којом овај систем формира непрекидну мрежу троуглова за цртање поља висина. Детаљнији опис ове методе дат је у раду [ĐURĐ2011]. Затим ће бити наведени најважнији детаљи имплементације система, при чему ће употреби HFPaC декомпресора у систему бити посвећена посебна пажња. На крају биће приказани резултати мерења перформанси које остварује систем након повезивања са HFPaC декомпресором, а опционо и са 2DRBUC декомпресором резидуала. Перформансе ће бити упоређене са одговарајућим перформансама које систем остварује без примене компресије поља висина.

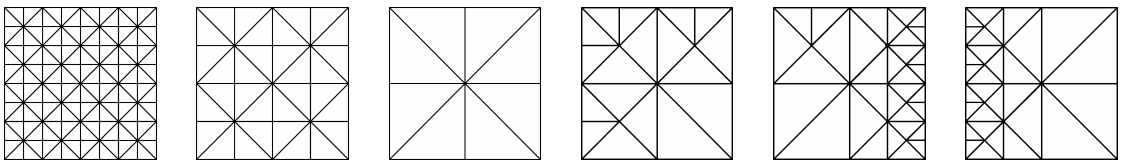
8.1 Суштина методе *Domino Tiling*

У овом одељку биће укратко описана суштина методе *Domino Tiling* [ĐURĐ2011]. Намена методе је брзо састављање непрекидне полуправилне 2D мреже троуглова на GPU, у циљу визуелизације поља висина. Њене основне карактеристике, које је разликују од других приступа су: брзо паралелно састављање непрекидне мреже троуглова спајањем полуправилних узорака мрежа (узорци су променљиве сложености, независни од података конкретног поља висина), погодност за динамичке измене поља висина и неуниформна процена грешке апроксимације мреже троуглова у простору екрана.

Алгоритам врши конструкцију 2D мреже за цртање видљивих делова поља висина састављањем мозаика унапред дефинисаних узорака мрежа, које су претходно смештене у GPU меморију. За сваки узорак, могуће је наћи неколико *сагласних* (енг. *conforming*) узорака у скупу узорака, са којима дати узорак формира непрекидну мрежу, без пукотина. Поступак има сличности са игром домина, одакле потиче назив методе: у игри домина, само плочице које се слажу по броју тачака могу бити постављене уз неку плочицу са датим бројем тачака. Свака парцела црта се употребом једног од узорака мрежа. Избор узорака мрежа врши

се у време цртања, сагласно дозвољеној екранској грешци за текућу позицију камере, уз задовољење услова непрекидности мреже на споју са суседним мрежама. С обзиром на то да метода конструише 2D мрежу, она оперише над *плочама* у равни уместо над парцелама. Метода *Domino Tiling* ограничена је на квадратне парцеле. Свака парцела састоји се од $p \times p$ тачака, где је $p=2^n+1$, а n позитиван цео број.

Скуп узорака мрежа независан је од поља висина. Узорци се формирају само једном, у току претпроцесирања, независно од поља висина која ће бити цртана коришћењем узорака. Свака мрежа састоји се од правоуглих једнакокраких троуглова у хоризонталној равни. Скуп узорака мрежа садржи *основне* и *прелазне* узорке. Основни узорци имају правилну структуру. Прелазни узорци имају полуправилну структуру и служе као прелаз са једног нивоа детаља на други. Пример основних и прелазних узорака приказан је на слици 22. У припремној фази алгоритма, сви узорци смештају се у GPU меморију, док се у CPU меморију смешта структура података (табела пресликавања) потребна за приступ узорцима за време цртања.



Слика 22 Примери основних и прелазних узорака мрежа; (а)-(в) основни узорци мрежа; (г)-(ђ) прелазни узорци мрежа; узорци мрежа (а) и (д) могу се сместити без пукотина лево, а (в) десно од (ђ); такође (б) може бити смештен лево или изнад (г), и изнад (д).

У фази избора узорака мрежа, на основу геометријских каратеристика плоче и растојања посматрача од плоче, метода *Domino Tiling* рачуна нумеричку вредност за сваку видљиву плочу. Ова вредност, означена са *ldr* (енг. *level of details reduction*), додељује се центру плоче и представља степен редукције детаља у односу на узорак мреже са најфинијим нивоом детаља. Сложеност редуковане мреже довољна је за цртање одговарајуће парцеле највећом потребном прецизношћу за дату резолуцију приказивача. С обзиром на то да се *ldr* центра плоче рачуна независно за сваку плочу, паралелизација поступка једноставно се постиже ангажовањем једне рачунске јединице по плочи. Након рачунања *ldr* центра плоче, сваком темену сваке видљиве плоче додељује се минимална

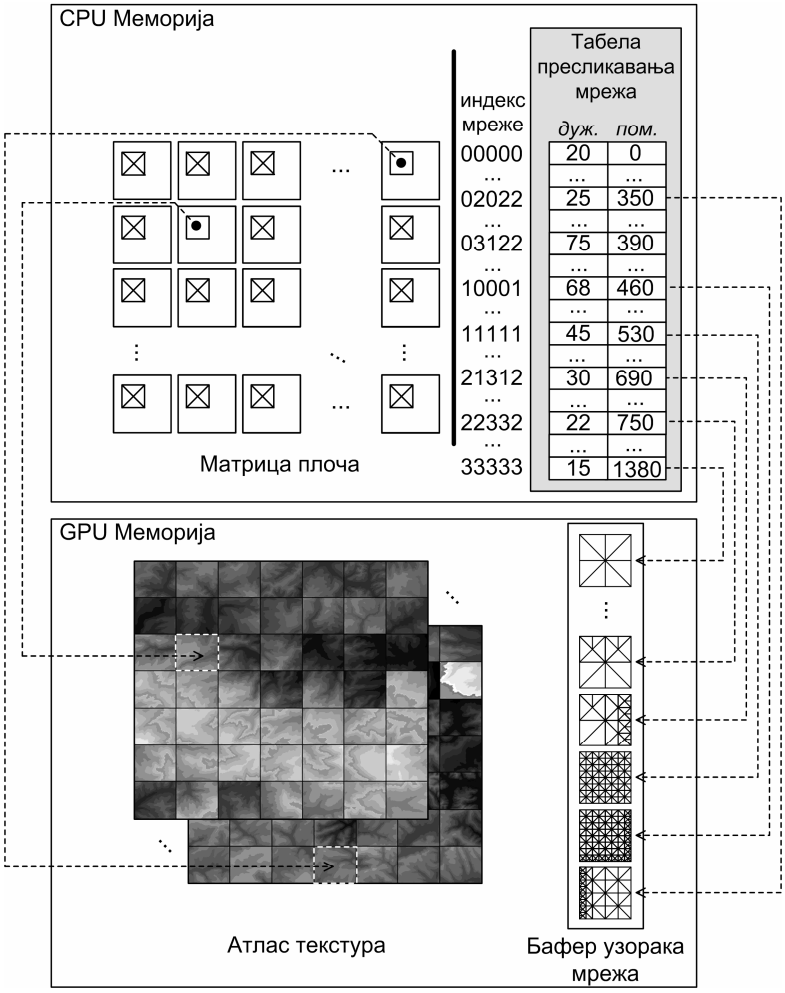
вредност *ldr* (која одговара најфинијем нивоу детаља) оних плоча које деле то теме. И овде се паралелизација једноставно остварује, ангажовањем једне рачунске јединице по дељеном темену. На основу ових 5 израчунатих вредности (једна у центру и четири у теменима плоче) формира се индекс узорка мреже, који једнозначно одређује узорак мреже из скупа узорака смештених у GPU меморију. Тиме је завршен избор узорка мреже који ће бити употребљен за цртање дате плоче у фази цртања. Овај узорак одговара израчунатом нивоу детаља за приказивање дате плоче и уклапа се без зазора са узорцима мрежа изабраним за приказивање суседних плоча.

У фази цртања поља висина, за сваку видљиву плочу издаје се наредба цртања изабраног узорка мреже. Јединствени индекс узорка мреже користи се за приступ табели пресликавања, из које се дохватају адреса узорка мреже (у GPU меморији) и његова дужина (број тачака у његовом саставу). Добијене адреса и дужина користе се као параметри позива графичкој библиотеци којим се врши цртање узорка мреже. Приликом цртања, висина сваког темена произвољног узорка мреже рачуна се на GPU, у програму за обраду темена (енг. *vertex shader*), на основу одговарајуће парцеле смештене у GPU меморију у виду монохроматске текстуре.

Две суседне парцеле преклапају се над једном врстом односно колоном тачака. Скалабилност методе постиже се груписањем парцела по принципу (комплетног) кватернарног стабла. Због тога, димензије поља висина ограничене су на $2^k \times 2^k$ парцела, где је k ненегативан цео број.

8.2 Детаљи имплементације система за визуелизацију

У овом одељку биће објашњени најбитнији детаљи имплементације система за визуелизацију поља висина, у оквиру којег је вршена практична провера применљивости методе HFPaC. Са нешто више детаља биће објашњени поступак цртања и примене методе HFPaC за декомпресију парцела. Остали детаљи имплементације дати су у [ĐURĐ2011]. За имплементацију овог система, употребљена је графичка библиотека OpenGL.



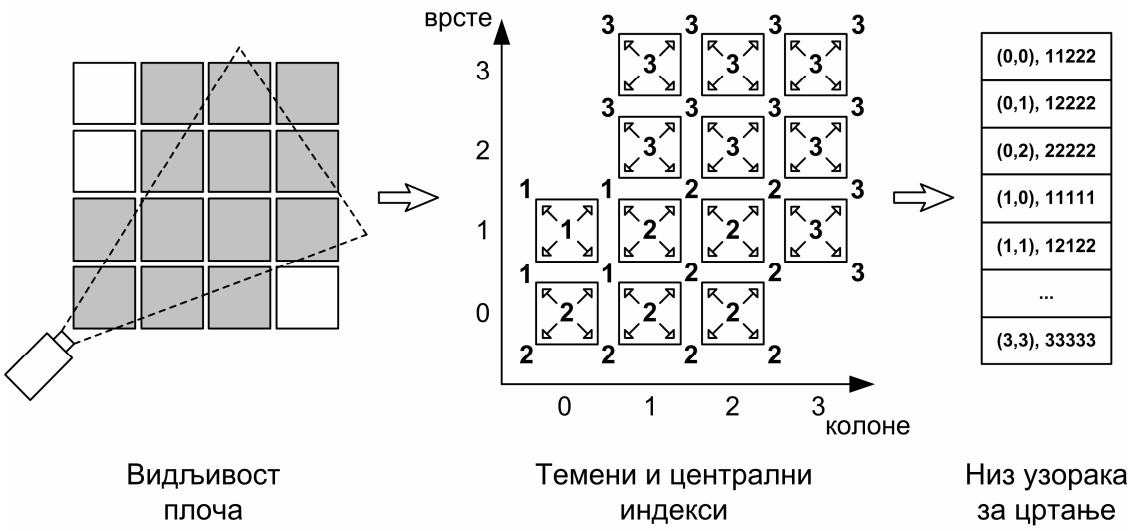
Слика 23 Структуре података које се формирају у припремној фази. Приказан је део садржаја ових структура и назначено је у коју меморију се смештају. У Табели пресликавања мрежа, *дуж.* означава дужину, а *пом.* означава померај узорка у односу на почетак бафера узорака мрежа.

У припремној фази, која се извршава једном, пре осталих фаза, формирају се четири структуре података: *матрица плоча*, *табела пресликавања мрежа*, *бафер узорака мрежа* и *атлас текстура*. Матрица плоча и табела пресликавања мрежа смештају се у CPU меморију, а бафер узорака мрежа и атлас текстура у GPU меморију. На слици 23 приказана је матрица плоча, део табеле пресликавања мрежа, одговарајући узорци у баферу узорака мрежа и два слоја (први и последњи) атласа текстура. Матрица плоча формира се на основу свих парцела поља висина, а једној плочи одговара једна парцела. У оквиру сваке плоче памти се да ли је одговарајућа парцела присутна у атласу текстура и, ако јесте, која је њена позиција у атласу (слој, врста, колона). Ови подаци постављају се и мењају динамички у време цртања, што ће бити објашњено у наставку. Поред наведених података, свака плоча садржи и податке неопходне за одређивање видљивости

дате плоче и нивоа детаља потребног за цртање одговарајуће парцеле за дату позицију камере и дату максималну грешку у екранским координатама. Ови подаци нису приказани на слици 23, а више детаља о њима дато је у [ĐURĐ2011]. Табела пресликавања мрежа служи за дохватање целобројних вредности *дужина* и *померај* на основу задатог јединственог индекса узорка мрежа. Дужина одговара броју темена који чине траку троуглова (енг. *triangle strip*) којом се црта дати узорак мреже, а померај одговара позицији датог узорка мреже у баферу узорака мрежа. Иако би начелно било могуће израчунати дужину као разлику помераја два суседна узорка мрежа, дужина се експлицитно памти због специфичног редоследа смештања узорака мрежа у бафер, као последице чињенице да се неки од узорака могу добити ротацијом или огледањем других. За смештање бафера узорака мрежа у GPU меморију користи се VBO (енг. *vertex buffer object*). VBO је OpenGL објекат специјализован за складиштење темена (геометријских података). Узорцима се приступа на основу јединственог индекса мреже и садржаја Табеле пресликавања мрежа. За атлас текстура се, у овој фази, само алоцира меморија, без дефинисања њеног садржаја. Атлас текстура представљен је низом 2D текстура (*2D texture array*, у терминологији библиотеке OpenGL). Сваки слој у атласу одговара једном елементу низа и представља једну 2D текстуру, независну од осталих. Сви слојеви (текстуре) имају исте димензије које се задају приликом алокације меморије. Због имплементационо ограничене дужине низа текстура (односно броја слојева), која износи 2048 на графичком адаптеру употребљеном у експериментима, уместо смештања једне парцеле по слоју, сваки слој у атласу има логичку структуру 2D матрице парцела, што је приказано на слици 23. На слици је такође приказано пресликавање из две плоче, садржане у матрици плоча, у одговарајуће парцеле смештене у првом слоју атласа. Низ 2D текстура употребљен је уместо независних текстура у циљу постизања бољих перформанси. Наиме, време потребно за промену текстуре при цртању није занемарљиво, а у случају независних текстура промена се врши при цртању сваке парцеле, што има значајан негативан утицај на брзину цртања. Са друге стране, време промене елемента низа текстура је занемарљиво.

У фази избора узорака мрежа за конструкцију непрекидне мреже у равни, најпре се одреде видљиве плоче (које одговарају видљивим парцелама) у зависности од

текуће позиције камере. Затим се за све видљиве плоче одреде јединствени индекси узорака мрежа. Ови индекси користе се за приступ узорцима који ће бити употребљени за цртање одговарајућих парцела. Јединствени индекс мреже за једну видљиву плочу састоји се од централног индекса, чија вредност је *ldr* дате плоче, и четири темена индекса (видети одељак 8.1). На крају, као резултат фазе избора узорака мрежа, формира се низ узорака за цртање, који за сваку видљиву плочу садржи један запис. Сваки запис састоји се од позиције (координата) плоче у матрици плоча и израчунатог јединственог индекса узорка мреже. Фаза избора узорака мрежа приказана је на слици 24, а фаза цртања узорака на слици 25.

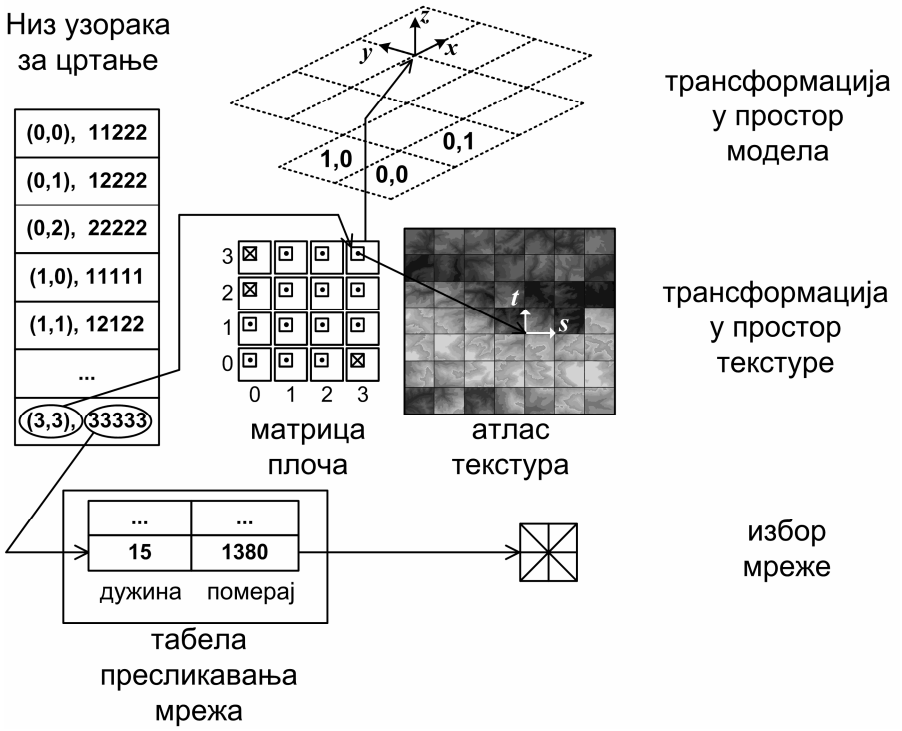


Слика 24 Фаза избора узорака мреже. У овој имплементацији, јединствени индекс добија се навођењем најпре вредности темених индекса (редом горњи леви и десни, затим доњи леви и десни), а затим централног индекса.

У фази цртања, за сваки елемент низа узорака за цртање, најпре се из матрице плоча дохвата одговарајућа плоча и провера да ли у атласу текстура постоји одговарајућа парцела поља висина. У случају да не постоји, дохвата се и смешта у атлас текстура. Након дохватања нових парцела и смештања у атлас, у новом пролазу кроз низ узорака за цртање, непосредно пре цртања сваке плоче реферисане у низу, задају се трансформација у простор реалног света и трансформација у простор текстура. Трансформацијом у простор реалног света координатни почетак смешта се на позицију дате плоче и врши се скалирање према њеним димензијама. Трансформацијом у простор текстура координатни почетак смешта се на позицију одговарајуће парцеле у атласу текстура. Након тога, црта се изабрани узорак мреже позивом OpenGL наредбе `glDrawArrays` са

параметрима *дужина* и *померај*. Ови параметри дохватају се из табеле пресликавања мрежа на основу јединственог индекса узорка мреже (слика 23). Фаза цртања приказана је на слици 25.

Пре првог цртања поља висина, атлас текстура је празан (недефинисан садржај). Нови садржај динамички се додаје у атлас, пре сваког цртања, довлачењем видљивих парцела које треба цртати, а које се у том тренутку не налазе у атласу. Нова парцела смешта се на слободну позицију у атласу текстура, и веза ка тој позицији успоставља се у одговарајућем елементу матрице плоча. За потребе тестирања у оквиру овог рада, имплементирана је једноставна FIFO (енг. *first in first out*) метода замене садржаја атласа текстура, када више нема слободног места за нове парцеле у атласу. На слици 23 приказан је пример где је веза између плоча и одговарајућих парцела успостављена само за две плоче. За остале плоче на слици 23, одговарајуће парцеле не налазе се у атласу. Довлачење парцела реализовано је применом пројектног узорка Стратегија (енг. *strategy*). Као конкретне стратегије, реализоване су варијанте довлачења некомпримованих парцела и парцела компримованих методом HFPaC (са и без додатне компресије резидуала).

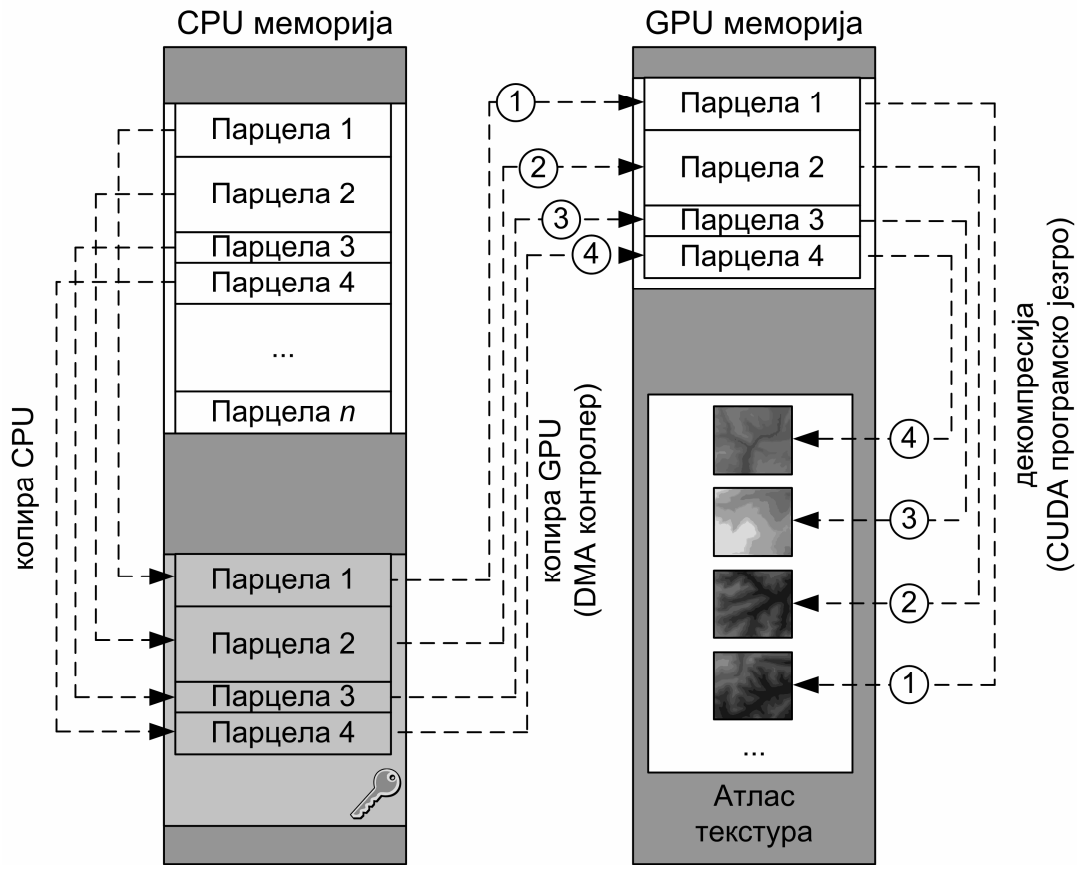


Слика 25 Фаза цртања мрежа. Приказан је поступак цртања мреже за плочу на позицији (3,3) у матрици плоча. Употребљен је низ узорака за цртање са слике 24.

У наставку одељка биће објашњен начин употребе HFPaC декомпресора у описаном систему за визуелизацију поља висина. Када се користи метода HFPaC, довлачење нове парцеле и њено смештање у атлас текстура захтева да се парцела у компримованом облику смести у GPU меморију, а затим изврши њено декомпримовање у други део GPU меморије, који одговара позицији дате парцеле у атласу текстура. Због тога, када се врши декомпресија већег броја парцела, раније поменута могућност симултаног преноса података и извршења CUDA програмског језгра (видети одељак 7.4) може бити употребљена за ефикасну имплементацију декомпресора. Међутим, CUDA пружа могућност за постизање још веће ефикасности кроз механизам *токова* (енг. *streams*). Све операције над подацима (асинхроно копирање меморије или покретање програмског језгра) издате једном току обављају се према редоследу издавања. Активности у различитим токовима обављају се конкурентно, независно једне од других. Истовременом употребом неколико токова, тако да сваки ток добије задатак да најпре копира једно компримовано поље висина у GPU меморију, а затим да изврши његову декомпресију, може се побољшати искоришћење графичког процесора. Побољшано искоришћење је од највећег значаја за мале парцеле (129×129 или 257×257 тачака) за које је у одељку одељку 7.4 показано да дају најмању брзину декомпресије због најмање доступног паралелизма.

Да би компримовани подаци могли да се пренесу из CPU у GPU меморију симултано са извршењем програмског језгра, неопходно је да они буду смештени у меморијски простор који се не може привремено изместити у екстерну меморију механизмом виртуелне меморије [NVID2010]. У наставку, такав меморијски простор биће означен као *закључан* (енг. *page locked memory* или *pinned memory*). Пренос компримованих података из закључане меморије у GPU меморију врши GPU, прецизније DMA контролер у саставу графичког адаптера. У овој имплементацији, CPU врши копирање потребних компримованих парцела у закључану меморију из неког другог дела CPU меморије (где су, на пример, претходно дохваћени из екстерне меморије на основу предикције кретања камере). Није разматрано довлачење компримованих парцела из других нивоа меморијске хијерархије. Тиме је претпостављено да комплетно поље висина може да се смести у CPU меморију, али не и у GPU меморију. Након копирања једне

парцеле у закључану меморију, CPU издаје команду копирања те парцеле у GPU меморију, а затим покреће CUDA програмско језгро за њену декомпресију. Након тога, CPU може да почне са копирањем наредне парцеле у закључану меморију, што се врши симултано са претходно иницираном обрадом на GPU. Копирање и покретање језгра врше се у истом току, чиме је обезбеђено да се копирање заврши пре него што програмско језгро почне са извршењем. Токови се користе по кружном принципу: након последњег тока у употреби, поново се користи ток 1. Обрада парцела на CPU је секвенцијална, али би се могла и паралелизовати (са степеном паралелизма који допушта CPU), што није разматрано у овом раду. На слици 26 приказана је употреба четири тока за декомпресију парцела. Декомпресија се врши уписом података директно у одговарајући слој текстуре која представља атлас текстуре, употребом CUDA *surface* објекта [NVID2010].



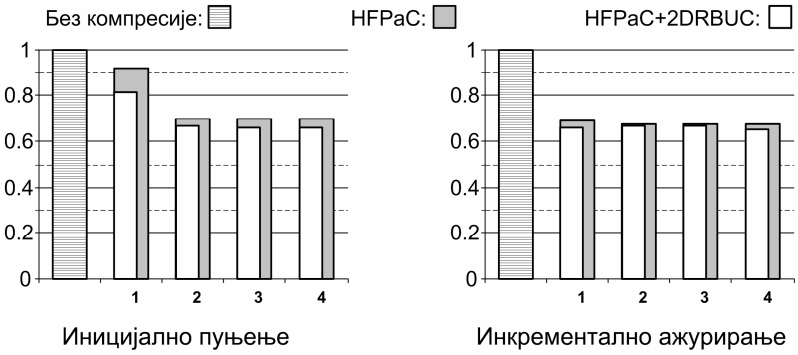
Слика 26 Декомпресија парцела компримованих HFPaC методом уз примену четири CUDA тока. Токови одговарају парцелама за које је закључено да се цртају у фази избора узорака мрежа. Закључан меморијски простор у CPU меморији обележен је симболом кључа. Приказан је само део атласа текстура који представља четири декомпримоване текстуре. Висина правоугаоника који представају парцеле означава величину компримоване парцеле.

8.3 Резултати

У овом одељку биће најпре изложени циљ и услови под којима су спроведена мерења перформанси система за визуелизацију поља висина у реалном времену, када се користи HFPaC декомпресор, односно комбинација HFPaC декомпресора са 2DRBUC декомпресором, за ажурирање садржаја атласа текстура. Затим ће бити приказани и анализирани добијени резултати у вези са постигнутом брзином декомпресије и бројем исцртаних слика у секунди (енг. *frame rate*). Поређење ће бити извршено са брзинама које се постижу у раду са некомпримованим подацима.

Циљ спроведених мерења јесте утврђивање утицаја примене методе HFPaC за декомпресију парцела на перформансе система, односно на број исцртаних слика у секунди. Извршене су три врсте мерења: ажурирање атласа без компресије парцела, са компресијом парцела применом основне методе HFPaC и са компресијом парцела применом методе HFPaC и додатном компресијом резидуала методом 2DRBUC. Резултати добијени мерењем у случају када парцеле нису компримоване употребљени су као референтни. Мерено је време формирања низа узорака за цртање, време ажурирања садржаја атласа текстура и време цртања поља висина. Као укупно време цртања једне слике узет је збир ова три времена. Мерена су наведена три времена у циљу одређивања њиховог појединачног удела у укупном времену цртања слике. Коришћена је CUDA имплементација алгоритма за формирање низа за претрагу узорака [ĐURĐ2011]. Сва мерења обављена су на истој конфигурацији као у одељку 7.1. Коришћен је терен Паџет Саунд (видети одељак 7.1, слика 12a). Виртуелна камера кретала се дуж дијагонале терена (од доњег левог до горњег десног угла на слици 12a), на висини 1434 m (што одговара једној трећини максималне висине терена) и брзини од 8000 m/s (28800 km/h). Велика брзина кретања камере узета је да би се обезбедило да подсистем за ажурирање атласа текстура у првом делу трајекторије камере буде оптерећен значајним бројем парцела при сваком цртању, у циљу јаснијег истицања евентуалних разлика у брзини употребљених приступа за ажурирање атласа. Величина атласа текстура одређена је унапред на начин да може да смести приближно 55% од укупног броја парцела терена са више резолуција (којих има 21845). Мерења су вршена само за величину парцела

129×129, за коју су на терену Пацет Саунд постигнуте најбоље перформансе [ĐURĐ2011]. У мерењима није разматрана декомпресија са губицима.



Слика 27 Релативна времена додавања нових парцела у атлас текстура, нормализована према времену додавања некомпримованих парцела (без компресије). Иницијално пуњење односи се на прво цртање поља висина, када се додаје највише парцела, јер је атлас текстура празан. Инкрементално ажурирање односи се на сва наредна додавања парцела у атлас, а приказане вредности представљају просек за сва додавања. Времена додавања нових компримованих парцела мерена су за различит број употребљених CUDA токова (1-4). Мерења су извршена за примењену методу HFPaC као и за комбиновану методу HFPaC са методом за додатну компресију резидуала 2DRBUC.

Резултати мерења времена потребног за додавање нових парцела у атлас текстура (пуњење атласа) приказани су на слици 27. Посебно је приказано време иницијалног пуњења (леви дијаграм на слици 27), зато што представља специјалан случај: тада се у атлас додаје највише парцела, јер је атлас иницијално празан. Анализа времена иницијалног пуњења је значајна зато што до додавања великог броја парцела у општем случају може доћи при свакој наглој промени позиције или оријентације камере, током интерактивног разгледања поља висина. На слици 27 приказано је и средње време пуњења (десни дијаграм), где је из поступка усредњавања изостављено време иницијалног пуњења. Резултати показују да се у оба случаја (иницијално пуњење и инкрементално ажурирање), применом методе компресије предложене у овом раду, остварује незанемарљив добитак у брзини пуњења (краће време) у односу на референтно мерење (где се не користи компресија). Такође, у оба случаја, комбинација метода HFPaC и 2DRBUC остварује нешто краће време у односу на основну методу HFPaC. Може се приметити да употреба више CUDA токова показује значајно боље резултате у односу на један ток само при иницијалном пуњењу атласа текстура. При том, мерења у којима се користе 3 или 4 тока показују занемарљиву разлику у односу на коришћење 2 тока. У светлу овог опажања, може се довести у питање оправданост употребе више од једног CUDA тока. У реалним апликацијама, код

којих су нагле промене позиције или оријентације камере очекиване и честе, примена два CUDA тока била би свакако оправдана, јер би пружила бржи одзив код сваке такве промене.

Табела 6 Расподела времена извршења појединих делова фазе цртања поља висина код првог цртања, када је атлас текстура празан (Иницијално пуњење), и код наредних цртања, када се садржај атласа инкрементално ажурира (Инкрементално ажурирање). Резултати су дати за референтно мерење (без компресије парцела).

		Време [ms]	%
Иницијално пуњење	**Формирање низа узорака мрежа за цртање**	0.400	0.15
	Иницијално пуњење атласа текстура	260.663	96.46
	Цртање	9.172	3.39
Инкрементално ажурирање	**Формирање низа узорака мрежа за цртање**	0.327	6.28
	Ажурирање атласа текстура	0.221	4.24
	Цртање	4.653	89.46

У табели 6 приказана је расподела трајања појединих делова фазе цртања за референтно мерење. Дати су резултати за цртање прве слике, када је атлас текстура празан и врши се иницијално пуњење, и усредњени резултати за цртање осталих слика, када се садржај атласа инкрементално ажурира. При том, у процесу усредњавања није укључено цртање прве слике. Са једне стране, резултати показују на иницијално пуњење атласа текстура одлази преко 96% укупног времена цртања прве слике. Стога, свако скраћивање времена пуњења позитивно би утицало на побољшање укупних перформанси. Са друге стране, резултати показују да је средње време ажурирања садржаја атласа текстура значајно мање од средњег времена цртања осталих слика. У укупном времену фазе цртања, удео ажурирања атласа текстура мањи је од 5%. Због тога, брже ажурирање атласа текстура не може значајно да повећа укупну брзину цртања, односно број нацртаних слика у секунди. Ипак, ово не значи да компресија поља висина у системима за њихову визуелизацију није од интереса. Напротив, компресија генерално омогућава бржи пренос података између различитих нивоа меморијске хијерархије, а то је од нарочитог интереса за поља висина која се не могу у потпуности сместити у GPU или CPU меморију, већ се морају динамички

довлачити из екстерне меморије. Међутим, као што је раније речено, у овом раду није разматрано довлачење података из екстерне меморије, што би свакако требало да буде један од праваца даљег развоја овог система за визуелизацију поља висина.

8.4 Резиме

У закључку овог поглавља треба приметити да је применом предложене методе за компресију поља висина у систем за визуелизацију поља висина остварено између 30% и 35% краће време пуњења атласа текстура парцелама из CPU меморије, у односу на време пуњења без компресије. Значај овог скраћења времена пуњења умањује чињеница да је, у просеку, удео времена пуњења у укупном времену цртања слике релативно мали (око 5%). Ипак, главна корист декомпресије парцела на GPU је у томе што се подаци преносе кроз целу меморијску хијерархију у компримованом облику, а CPU се незнатно ангажује приликом декомпресије. У суштини, за практичну употребу HFPaC декомпресора у оквиру система за визуелизацију поља висина, било је битно утврдити да време пуњења атласа текстура компримованим парцелама није веће у односу на време пуњења некомпримованим парцелама, што су мерења и показала.

9. Закључак

У раду је представљена нова метода за брзу компресију и декомпресију правилних поља висина која ефикасно користи паралелизам високог нивоа доступан у модерним графичким процесорима. Метода је јединствена према скупу карактеристика које је одликују. Она дозвољава компресију са или без губитака, прогресивну декомпресију и независну декомпресију појединачних тачака поља висина, а инхерентно производи површ без пукотина, чак и у случају декомпресије са губицима. Прогресивна декомпресија је могућа захваљујући разлагању компримованих података у слојеве. Већ први слој компримованих података, заснован на апроксимацији квадратним Безјеовим површима, даје висок степен компресије уз одличан квалитет декомпримованог поља висина. Висок ниво паралелизма и добро искоришћење графичког процесора постиже се независним израчунавањем појединих резидуала. Предложена је и опциона метода за паралелну декомпресију додатно компримованих резидуала, а применљивост резултата проверена у систему за визуелизацију терена.

Ефикасност методе измерена је на три репрезентативна и релативно велика поља висина. Имплементација методе у CUDA C показала је да поља висина од 16K×16K тачака могу бити компримована и декомпримована за мање од секунде на модерним графичким процесорима, у случају да су сви подаци присутни у локалној меморији графичког процесора. Постигнута брзина декомпресије поредива је (мало је и већа) са брзином декомпресије претходно објављене најбоље методе прилагођене графичком процесору. Међутим, предложена метода постиже велику брзину декомпресије и под значајно мањим оптерећењем (количином података које треба декомпримовати), што поред наведених јединствених карактеристика представља посебну предност у односу на претходну методу, захваљујући финијој грануларности паралелизма на нивоу обраде појединих тачака, а не целих парцела. Такође, за разлику од претходне методе, развијен је и ефикасан паралелан алгоритам за компресију. То указује на потенцијал методе да буде примењена и на динамички променљива поља висина.

Основна метода (HFPaC), на пољима висина на којима је испитивана, постиже степен компресије без губитака у опсегу од 1:1.8 до 1:7.1, што је нешто лошије од

савремене компетитивне методе. Међутим, утврђено је да основна метода може добро да се комбинује са додатним кораком компресије података када се могу постићи степени компресије поредиви са савременом компетитивном методом и показано је да постоји отворен простор за даље истраживање у домену паралелне компресије резидуала. Применом компресора опште намене без губитака (ZIP), који се извршава на централном процесору, на компримоване податке које производи основна метода, остварује се укупан степен компресије у опсегу од 1:2.4 до 1:9.8, што је поредиво са степеном компресије оствареним савременом компетитивном методом. Недостатак је секвенцијална природа додатне компресије, те се даље истраживало у правцу алгоритама за додатну компресију који се могу успешно паралелизовати. Степен компресије остварен применом развијене паралелне варијанте алгоритма 2DRBUC за декомпресију коначних резидуала (који имају највећи удео у компримованим подацима основне методе) је у опсегу од 1:2.1 до 1:7.5, што је боље у односу на основну методу, али нешто лошије у односу на први приступ додатној компресији. Највеће измерено смањење брзине декомпресије применом 2DRBUC методе је око 30% у односу на основни метод, чиме је брзина декомпресије и даље поредива са брзином коју постиже компетитивна метода. Ипак, будући да примена додатног корака на графичком процесору незанемарљиво умањује брзину декомпресије, одлука о његовој примени препушта се апликацији приликом балансирања између степена компресије и брзине декомпресије.

Проверена је практична применљивост предложене методе у оквиру система за интерактивну визуелизацију поља висина, када се у меморију графичког адаптера не може сместити комплетно поље висина. Тада је, са становишта уштеде, како простора у оперативној меморији, тако и времена преноса из оперативне меморије у меморију графичког адаптера, од интереса да се у оперативној меморији чува поље висина у компримованом облику. Резултати недвосмислено показују да предложена метода декомпресије, која се извршава тек на графичком процесору, не успорава цртање поља висина, чак у просеку незнатно доприноси повећању укупне брзине цртања у односу на случај када се некомпримоване парцеле преносе из оперативне меморије у меморију графичког процесора. Убрзање цртања је само споредан ефекат – суштина добити применом методе је чињеница

да се подаци, како на удаљеном рачунару, затим на диску, а коначно и у оперативној меморији рачунара домаћина могу чувати у компримованом облику, уз уштеде у меморијском простору и времену преноса преко мреже, односно између нивоа меморијске хијерархије, а све то без битног ангажовања централног процесора за декомпресију, која се готово у целини обавља на графичком процесору.

Предложена метода отвара неколико могућих праваца даљег истраживања. У овом раду анализирана је употреба квадратних Безјеових површи за грубу апроксимацију поља висина. Од интереса би било да се проучи ефекат који би имала употреба Безјеових површи вишег реда. Са једне стране, биће потребно памтити више контролних тачака за дефинисање Безјеове површи над једним сегментом, чиме се повећава величина првог слоја, за исту величину сегмента коришћену са квадратним Безјеовим површима. Са друге стране, може се очекивати да површи вишег реда произведу квалитетнију апроксимацију, чиме би се смањиле величине другог и трећег слоја, што би могло да компензује повећање величине првог слоја и тиме повећа степен компресије. Такође треба имати у виду да број рачунских операција потребних за одређивање тачака на Безјеовој површи расте са порастом реда површи, што би могло да негативно утиче на брзину компресије и декомпресије.

Други правац истраживања могао би да буде усмерен ка ефикаснијој паралелној компресији резидуала на графичком процесору, уз задржавање могућности појединачне декомпресије сваке тачке. Примена 2DRBUC методе за компресију резидуала показала је да је могуће реализовати ефикасну декомпресију на графичком процесору, али поступак ипак није идеално паралелизован и доводи до незанемарљивог пада перформанси. Циљ би био да се постигне висок степен компресије, уз финију грануларност и већи степен паралелизма. Такође, имајући у виду компресију динамички променљивих поља висина, од интереса би био и развој паралелног 2DRBUC алгоритма за компресију резидуала.

Променљива величина сегмената, у саставу једне парцеле, која се прилагођава локалним карактеристикама поља висина, могла би да помогне код повећања степена компресије. Мања величина сегмента би се користила у зонама поља

висина где су промене висине скоковите и нагле. Већа величина сегмента би се користила у зонама поља висина где су промене благе и прогресивне. Обезбеђивање непрекидности површи код декомпресије са губицима је један од проблема који би истраживање у овом правцу морало да реши.

Комбиновање HFPaC методе за компресију најфинијих детаља са неком другом методом за компресију грубљих детаља поља висина са више резолуција би могло такође представљати један од праваца даљег истраживања.

Коначно, интересантно би било размотрити могућност примене предложене методе за компресију дигиталних слика. У суштини, свака дигитална слика може се посматрати као више поље висина, где сваки канал (у RGB, HSV, YUV или неком другом простору боја) представља једно поље. Због своје погодности за паралелизацију (уз извршавање на савременим графичким процесорима, који се данас налазе већ у кућним рачунарима) и могућности независне декомпресије појединачних тачака, предложена метода би могла да се користи и за компресију текстура. Хардверска подршка методи би омогућила да се компримоване текстуре чувају у локалној меморији графичког процесора и користе без претходног декомпримовања. То би представљало значајан квалитативни напредак у односу на тренутну хардверску подршку компресији текстура, која нуди једино компресију са губицима.

У резимеу, може се рећи да је развијена нова метода са ефикасним алгоритмима за компресију и декомпресију поља висина, коју одликује скуп јединствених карактеристика и која је поредива по степену компресије и перформансама са тренутно најбољом познатом методом. Проверена је практична применљивост методе и отворено више праваца за нова истраживања.

10. Литература

[ASIR2005] Asirvatham A, Hoppe H. (2005) Terrain rendering using GPU-based geometry clipmaps. In: Pharr M (ed) GPU Gems 2, Addison Wesley, pp 27-45

[BETT2007] Bettio F, Gobbetti E, Marton F, Pintore G (2007) High-quality networked terrain rendering from compressed bitstreams. Proc. of the 12-th int. conf. on 3D web technology, Perugia, Italy, April 15-18, pp 37-44

[BEZI1970] Bézier P. (1970) Emploi des Machines à commande numérique. Masson et Cie, Paris.

[CHEN1986] Chen ZT, Tobler WR (1986) Quadtree representations of digital terrain. Proc. Auto-Carto London, London, England, Sept 14-19, pp 475-484

[DICK2009] Dick C, Schneider J, Westermann R (2009) Efficient geometry compression for GPU-based decoding in realtime terrain rendering. Comput. Graph. Forum 28(1):67-83

[ĐURĐ2011] Đurđević Đ, Tartalja I (2011) Domino tiling: a new method of real-time conforming mesh construction for rendering changeable height fields, J. comput. sci. technol. 26(6):971-987

[ĐURĐ2013] Đurđević Đ, Tartalja I (2013) HFPaC: GPU Friendly Height Field Parallel Compression, Geoinformatica 17(1): 207-233, doi: 10.1007/s10707-012-0171-x

[EART2011] Earth explorer: http://earthexplorer.usgs.gov/ (accessed: 12 sept 2012)

[FRAN1995] Franklin WR (1995) Compressing elevation data. SSD '95 Proc. of the 4th Int. Symp. on Advances in Spatial Databases, Portland, Maine, USA, Aug 6-9, pp 385-404

[FLYN1972] Flynn M (1972) Some computer organizations and their effectiveness. IEEE Trans. on Comput. C-21(9):948-960

[GERA2002] Gerald F (2002) Curves and surfaces for CAGD: a practical guide. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, 5th ed

[GERS2003] Gerstner T (2003) Multiresolution visualization and compression of global topographic data. Geoinformatica 7(1):7-32

[GOBB2006] Gobbetti E, Marton F, Cignoni P, Benedetto MD, Ganovelli F (2006) C-BDAM - compressed batched dynamic adaptive meshes for terrain rendering. Comput. Graph. Forum 25(3):333-342

[GOOG2012] http://maps.google.com (accessed: 17 Aug 2012)

[HARR2007] Harris M, Sengupta S, Owens JD (2007) Parallel prefix sum (scan) with CUDA. In: Nguyen H (ed) GPU Gems 3, Addison Wesley, pp 851-876
[HAWA2012] http://rocky.ess.washington.edu/data/raster/tenmeter/hawaii/index.html (accessed: 23 Aug 2012)
[HWA2005] Hwa LM, Duchaineau MA, Joy KI (2005) "Real-time optimal adaptation for planetary geometry and texture: 4-8 tile hierarchies. IEEE Trans. on Vis. and Comput. Graphics, 11(4):355-368
[INAN2008] Inanc M. (2008) Compressing terrain elevation datasets. PhD dissertation, Rensselaer Polytechnic Institute Troy, NY, USA
[JPEG1997] JPEG Committee, FCD-14495, Lossless and near-lossless coding of continuous tone still images (JPEG-LS), http://www.jpeg.org/public/fcd14495p.pdf, (accessed 22 Aug 2012).
[KIDN1992] Kidner DB, Smith DH (1992) Compression of digital elevation models by Huffman coding. Comput. & Geosci. 18(8):1013-1034
[KIDN1997] Kidner DB, Smith DH (1997) Storage-efficient techniques for representing digital terrain models. Innovations in GIS 4, Taylor & Francis, pp 25-41
[KIDN1999] Kidner DB, Dorey M, Smith DH (1999) What's the point? Interpolation and extrapolation with a regular grid DEM. Proc. of GeoComputation'99, Virginia, USA, July 25-28, http://www.geovista.psu.edu/sites/geocomp99/Gc99/082/gc_082.htm (accessed 08 Nov 2011)
[KIDN2003] Kidner DB, Smith DH (2003) Advances in the data compression of digital elevation models, Comput. & Geosci. 29(8):985-1002
[KIM2004] Kim JK, Ra JB (2004) A real-time terrain visualization algorithm using wavelet-based compression. Vis. Comput. 20(2):67-85
[KIRK2010] Kirk DB, Hwu W (2010) Programming Massively Parallel Processors: a Hands-on Approach. Morgan Kaufmann Publishers.
[LEWI1994] Lewis M, Smith DH (1994) Optimal predictors for compression of digital elevation models. Comput. & Geosci. 20(7-8):1137-1141
[LI2011] Li Y (2011) CUDA-accelerated HD-ODETLAP: a high dimensional geospatial data compression framework. PhD dissertation, Rensselaer Polytechnic Institute Troy, NY, USA

[LIND2010] Lindstrom P, Cohen JD (2010) On-the-fly decompression and rendering of multiresolution terrain. Proc. of the 2010 ACM SIGGRAPH symp. on Interactive 3D Graphics and Games, Washington, D.C., USA, February 19-21, pp 65-73
[MICR2012] http://maps.microsoft.com (accessed: 17 Aug 2012)
[MOFF2005] Moffat A, Ahn VN (2005) Binary codes for non-uniform sources. Data Compression Conf., Snowbird, Utah, USA, March 29-31, pp 133-142
[NVID2010] NVIDIA Corp (2010) NVIDIA CUDA C Programming Guide v3.2. https://developer.nvidia.com/cuda-toolkit-archive (accessed 02 Nov 2012)
[NVID2012] NVIDIA Corp (2012) NVIDIA CUDA C Programming Guide v4.2. https://developer.nvidia.com/cuda-toolkit-archive (accessed 02 Nov 2012)
[PKWA2011] http://www.pkware.com (accessed: 27 Sept 2011)
[PUGE2012] http://www.cc.gatech.edu/projects/large_models/ps.html (accessed: 23 Aug 2012)
[STOO2008] Stookey J, Xie Z, Cutler B, Franklin WR, Tracy DM, Andrade MA (2008) Parallel ODETLAP for terrain compression and reconstruction. Proc. of the 16th ACM SIGSPATIAL int. conf. on Advances in geographic information systems, Irvine, California, USA, November 5-7, doi:10.1145/1463434.1463456
[SWEL1998] Sweldens W (1998) The lifting scheme: A construction of second generation wavelets. SIAM J. Math. Anal., 29(2), 511–546, doi: 10.1137/S0036141095289051
[UTAH2011] http://gis.utah.gov/ (accessed: 27 Sept 2011)
[USGS2011] U.S. Geological Survey National Geospatial Program Lidar Guidelines and Base Specification, http://lidar.cr.usgs.gov/USGS-NGP Lidar Guidelines and Base Specification v13(ILMF).pdf (accessed: 12 Oct 2011)
[WREN1973] Wren EA (1973) Trend surface analysis-a review. Can. J. of Explor. Geophys. 9(1):39-44
[XIE2008] Xie Z, Andrade MA, Franklin WR, Cutler B, Inanc M, Muckell J, Tracy DM (2008) Progressive transmission of lossily compressed terrain. Conf. Latinoamericana de Informática (CLEI 2008), Santa Fe, Argentina, September 8-12, http://www.ecse.rpi.edu/Homepages/wrf/p/127-andrade-prog-trans-2008.pdf (accessed: 12 Oct 2011)

[ZHAN2011] Zhang J, You S, Gruenwald L (2011) Parallel Quadtree Coding of Large-Scale Raster Geospatial Data on GPGPUs. Proc. of the 19th ACM SIGSPATIAL int. conf. on Advances in geographic information systems, Chicago, Illinois, USA, November 1-4, doi: 10.1145/2093973.2094047

Биографија аутора

Ђорђе Ђурђевић рођен је 21.6.1976. године у Београду. Дипломирао је 2001. године на Електротехничком факултету Универзитета у Београду, са највећом просечном оценом у школској 2000./01. години на Одсеку за рачунарску технику и информатику. Након завршених магистарских студија на Електротехничком факултету Универзитета у Београду, на Одсеку за рачунарску технику и информатику, смер Архитектура и оргранизација рачунара и рачунарских мрежа, магистрирао је 26.6.2006. године на Електротехничком факултету Универзитета у Београду из области Рачунарска графика, на тему "Динамичка реконфигурација модела терена са неправилним ивицама", ментор др Игор Тартаља.

Од октобра 2001. године до јануара 2005. године ради на Електротехничком факултету Универзитета у Београду при Катедри за општу електротехнику, а од јануара 2005. ради при Катедри за рачунарску технику и информатику. Држао је аудиторне вежбе на 7 различитих предмета.

Учествовао је на пројектима Министарства науке Републике Србије ТР13001, ТР32039 и ТР32047. Учествовао је у ТЕМПУС пројекту JEP-40091-2005.

Учествовао је на неколико софтверских пројеката, чији се резултати успешно примењују у пракси:

- 3DShape Designer – CAD алат за пројектовање 3D објеката и њихову анимацију;
- 3D Terrain Viewer/Dredge Viewer – апликација за приказивање дигиталних мапа терена;
- Elvis (Editing LIDAR Visual Interactive System) – апликација за филтрирање и едитовање тачака терена прикупљених снимањем из ваздуха помоћу LIDAR система;
- Пакет апликација за планирање, развој и предикцију који користи Агенција за контролу летења Републике Србије

Коаутор је око 20 стручних и научних радова, од којих су 3 објављена у часописима међународног значаја са импакт фактором. Добитник награде

друштва ЕТРАН за најбољи рад младог истраживача 2002. године. Главни стручни и истраживачки интереси су у областима 2D и 3D рачунарске графике, објектно-оријентисане анлизе, пројектовања, програмских језика и програмирања, дигиталне обраде сигнала и слика.

Прилог 1.

Изјава о ауторству

Потписани Ђорђе Ђурђевић ,

број индекса _____

Изјављујем

да је докторска дисертација под насловом

Ефикасна паралелна компресија поља висина

_____

_____

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, 14. III 2013.

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора _Ђорђе Ђурђевић_____

Број индекса _____

Студијски програм _____

Наслов рада ___Ефикасна паралелна компресија поља висина_____

Ментор _____др Игор Тартаља, доцент_____

Потписани _Ђорђе Ђурђевић_____

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предаоа за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 14. III 2013.

_____

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Ефикасна паралелна компресија поља висина

_____

_____

која је моје ауторско дело.

Дисертацију са свим прилозима предао сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 14. III 2013.

_____