VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Ruslan PACEVIČ

METHODS AND DISTRIBUTED SOFTWARE FOR VISUALIZATION OF CRACKS PROPAGATING IN DISCRETE PARTICLE SYSTEMS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (07T)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2015

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2011–2015.

Scientific Supervisor

Assoc. Prof. Dr Arnas KAČENIAUSKAS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

The Dissertation Defense Council of Scientific Field of Informatics Engineering of Vilnius Gediminas Technical University:

Chairman

Prof. Dr Habil. Romualdas BAUŠYS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

Members:

Prof. Dr Habil. Rimantas BARAUSKAS (Kaunas University of Technology, Informatics – 09P),
Dr Robertas DAMAŠEVIČIUS (Kaunas University of Technology, Informatics Engineering – 07T),
Asoc. Prof. Dr Raimundas MATULEVIČIUS (Tartu University, Informatics Engineering – 07T),
Prof. Dr Olegas VASILECAS (Vilnius Gediminas Technical University, Informatics Engineering – 07T).

The dissertation will be defended at the public meeting of the Dissertation Defense Council of Informatics Engeering in the Senate Hall of Vilnius Gediminas Technical University at **9 a. m. on 20 November 2015**.

Address: Saulėtekio al. 11, LT-10223 Vilnius, Lithuania.
Tel.: +370 5 274 4956; fax +370 5 270 0112; e-mail: doktor@vgtu.lt

A notification on the intend defending of the dissertation was send on 19 October 2015.

A copy of the doctoral dissertation is available for review at VGTU repository http://dspace.vgtu.lt and at the Library of Vilnius Gediminas Technical University (Saulėtekio al. 14, LT-10223 Vilnius, Lithuania).

VGTU leidyklos TECHNIKA 2343-M mokslo literatūros knyga

ISBN 978-609-457-855-7

ruslan.pacevic@vgtu.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Ruslan PACEVIČ

DISKREČIŲJŲ DALELIŲ SISTEMOSE SKLINDANČIŲ PLYŠIŲ VIZUALIZAVIMO METODAI IR IŠSKIRSTYTOJI PROGRAMINĖ ĮRANGA

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (07T)

VGTU
LEIDYKLA
Vilnius TECHNIKA 2015

Disertacija rengta 2011–2015 metais Vilniaus Gedimino technikos universitete.

Mokslinis vadovas

doc. dr. Arnas KAČENIAUSKAS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T).

Vilniaus Gedimino technikos universiteto Informatikos inžinerijos mokslo krypties disertacijos gynimo taryba:

Pirmininkas

prof. habil. dr. Romualdas BAUŠYS (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07T).

Nariai:

prof. habil. dr. Rimantas BARAUSKAS (Kauno technologijos universitetas, informatika – 09P),

dr. Robertas DAMAŠEVIČIUS (Kauno technologijos universitetas, informatikos inžinerija – 07T),

doc. dr. Raimundas MATULEVIČIUS (Tartu universitetas, informatikos inžinerija – 07T),

prof. dr. Olegas VASILECAS (Vilniaus Gedimino technikos universitetas, informatikos inžinerija – 07T).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties disertacijos gynimo tarybos posėdyje **2015 m. lapkričio 20 d. 9 val.** Vilniaus Gedimino technikos universiteto senato posėdžių salėje.

Adresas: Saulėtekio al. 11, LT-10223 Vilnius, Lietuva.
Tel.: (8 5) 274 4956; faksas (8 5) 270 0112; el. paštas doktor@vgtu.lt

Pranešimai apie numatomą ginti disertaciją išsiųsti 2015 m. spalio 19 d.

Disertaciją galima peržiūrėti VGTU talpykloje http://dspace.vgtu.lt ir Vilniaus Gedimino technikos universiteto bibliotekoje (Saulėtekio al. 14, LT-10223 Vilnius, Lietuva).

Abstract

Scientific visualization is becoming increasingly important in analyzing and interpreting numerical and experimental data sets. Parallel computations of discrete particle systems lead to large data sets that can be produced, stored and visualized on distributed IT infrastructures. However, this leads to very complicated environments handling complex simulation and interactive visualization on the remote heterogeneous architectures. In micro-structure of continuum, broken connections between neighbouring particles can form complex cracks of unknown geometrical shape. The complex disjoint surfaces of cracks with holes and unavailability of a suitable scalar field defining the crack surfaces limit the application of the common surface extraction methods. The main visualization task is to extract the surfaces of cracks according to the connectivity of the broken connections and the geometry of the neighbouring particles. The research aims at enhancing the visualization methods of discrete particle systems and increasing speed of distributed visualization software.

The dissertation consists of introduction, three main chapters and general conclusions. In the first Chapter, a literature review on visualization software, distributed environments, discrete element simulation of particle systems and crack visualization methods is presented. In the second Chapter, novel visualization methods were proposed for extraction of crack surfaces from monodispersed particle systems modelled by the discrete element method. The cell cut-based method, the Voronoi-based method and cell centre-based method explicitly define geometry of propagating cracks in fractured regions. The proposed visualization methods were implemented in the grid visualization e–service VizLitG and the distributed visualization software VisPartDEM. Partial data set transfer from the grid storage element was developed to reduce the data transfer and visualization time.

In the third Chapter, the results of experimental research are presented. The performance of e-service VizLitG was evaluated in a geographically distributed grid. Different types of software were employed for data transfer in order to present the quantitative comparison. The performance of the developed visualization methods was investigated. The quantitative comparison of the execution time of local Voronoi-based method and that of global Voronoi diagrams generated by Voro++ library was presented. The accuracy of the developed methods was evaluated by computing the total depth of cuts made in particles by the extracted crack surfaces. The present research confirmed that the proposed visualization methods and the developed distributed software were capable of visualizing crack propagation modelled by the discrete element method in monodispersed particulate media.

Reziumė

Mokslinis vizualizavimas tampa vis svarbesniu analizuojant sudėtingas priklausomybes skaičiavimų ir eksperimentų rezultatuose. Diskrečiųjų dalelių sistemų lygiagretūs skaičiavimai generuoja didelius duomenų kiekius, kurie saugomi ir vizualizuojami išskirstytose informacinių technologijų infrastruktūrose. Dideli, geografiškai išskirstyti duomenų kiekiai reikalauja sudėtingos programinės įrangos efektyviems skaičiavimams ir interaktyviam vizualizavimui išskirstytose heterogeninėse architektūrose. Diskrečiųjų elementų metodu modeliuojant kontinuumo mikrostruktūrą, tarp dalelių nutrūkusios jungtys gali sudaryti sudėtingos formos plyšius, kurių geometrija nėra žinoma. Sudėtinga plyšių forma su skylėmis ir apibrėžiančio skaliarinio lauko nebuvimas riboja standartinių paviršių ištraukimo metodų taikymo galimybes. Vizualizuojant plyšius, reikia sukonstruoti plyšių paviršių geometriją iš tarp dalelių nutrūkusių jungčių topologijos ir kaimyninių dalelių pozicijų. Disertacijos tikslas yra patobulinti diskrečiųjų dalelių sistemų vizualizavimo metodus bei padidinti metodų realizacijos išskirstytoje programinėje įrangoje greitaveiką.

Disertaciją sudaro įvadas, trys pagrindiniai skyriai ir bendrosios išvados. Pirmajame skyriuje apžvelgta vizualizavimo programinė įranga, išskirstytosios vizualizavimo sistemos išteklių tinklo aplinkose, dalelių sistemų modeliavimas diskrečiųjų elementų metodu ir plyšių, sklindančių dalelių sistemose, vizualizavimo metodai. Antrajame skyriuje detaliai aprašomi metodai sukurti vizualizuoti plyšius, kurie sklinda monodispersinėse dalelių sistemose, modeliuojamose diskrečiųjų elementų metodu. Pažeistuose regionuose celių kirtimo, Voronoi dekompozicijos ir celių centrų metodai geometriniais primityvais apibrėžia sklindančių plyšių geometriją. Sukurti plyšių vizualizavimo metodai įdiegti vizualizavimo e. paslaugoje VizLitG ir išskirstytojoje vizualizavimo sistemoje VisPartDEM. Dalinio duomenų rinkinio parsiuntimo iš išteklių tinklo duomenų saugyklos paslauga įdiegta VizLitG siunčiamų duomenų kiekiui sumažinti ir vizualizavimui pagreitinti.

Trečiajame skyriuje aprašomi eksperimentinių tyrimų rezultatai. E. paslaugos VizLitG greitaveika ištirta geografiškai išskirstyto išteklių tinklo atveju. Pateikti duomenų siuntimo skirtinga programine įranga iš išteklių tinklo duomenų saugyklos tyrimai. Skyriuje pateikti plyšių vizualizavimo metodų greitaveikos ir tikslumo kiekybiniai palyginimai. Voronoi dekompozicijos metodo realizacijos greitaveika palyginta su Voro++ bibliotekos greitaveika. Plyšių vizualizavimo metodų tikslumas įvertintas apskaičiavus bendrą sukonstruotų paviršių įsiskverbimo į daleles gylį. Tyrimų rezultatai parodė, kad pasiūlyti vizualizavimo metodai ir sukurta programinė įranga tinka vizualizuoti plyšiams sklindantiems monodispersinėse dalelių sistemose.

Notations

Abbreviations

API	– Application Programmin Interface;
CPU	– Central Processing Unit;
DEM	– Discrete Element Method
GB	– Gigabyte;
GHz	– Gigahertz;
GLSL	– OpenGL Shading Language;
GPU	– Graphics Processing Unit;
GridFTP	– Grid File Transfer Protocol;
GUI	– Graphical User Interface;
HDD	– Hard Disk Drive;
HDF	– Hierarchical Data Format;
HTTPS	– Hypertext Transfer Protocol Secure;
JAX-WS	– Java API for XML Web Services;
LFC	– Logical File Catalog;

MB – Megabyte;
MHz – Megahertz;
MIME – Multipurpose Internet Mail Extensions;
MPI – Message Passing Interface;
MVE – Multi-View Environment;
PC – Personal Computer;
RAM – Random-Access Memory;
SE – Storage Element;
SOAP – Simple Object Access Protocol;
SSL – Secure Sockets Layer;
vGPU – Virtual Graphics Processing Unit;
XML – Extensible Markup Language.

Contents

[1] The annexes are supplied in the enclosed compact disc

Introduction

Problem Formulation

Visualization plays an important role in the numerical analysis cycle, which consists of data preparation, computations, graphical analysis of the numerical results and further correction of design variables (Hansen *et al.* 2005). Numerical simulations of discrete particle systems (Cundall *et al.* 1979) and modern physical experiments lead to large data sets that present challenges to researchers. Scientific visualization is becoming increasingly important in analyzing and interpreting numerical and experimental data sets to make decisions in high technology design. However, large data sets and a complex visualization process present challenges to the developers of interactive visualization systems. Distributed visualization allocates different parts of the machine processing and the provided services to different computers in order to improve performance. A lot of industrial applications are solved on computer clusters and grid. With the power of the grid, scientists are able to perform simulations at previously impossible and unexplored problem scales. However, this leads to very complicated environments handling complex simulation and interactive visualization on the remote heterogeneous architectures.

Discrete particle computations are based on the positions of particles and forces acting between them (Cundall *et al.* 1979). 1D connections between

neighbouring particles are not well suited for the reliable interpolation and common visualization methods used in 3D. Micro-structure of continuum can be modelled by discrete particle system and the defects between the pairs of the neighbouring particles are identified via the broken connections (Rojek *et al.* 2011). The defects can form complex cracks of unknown geometrical shape. The main visualization task is to extract the surfaces of cracks according to the connectivity of the broken connections and the geometry of the neighbouring particles. The complex disjoint surfaces of cracks with holes and unavailability of a suitable scalar field defining the crack surfaces limit the application of the widely used surface extraction methods. Moreover, the fixed connectivity of the moving particles prevents direct application of the commonly used methods for generation of Voronoi decompositions (Aurenhammer 1991) or Delaunay triangulations (Amenta *et al.* 2001).

Relevance of the Thesis

Visualization is used for analyzing the data and presenting the results across a wide range of disciplines in science and industry (Hansen *et al.* 2005). However, large data sets and a complex visualization process require considerable development efforts and impressive computing resources. Distributed visualization systems and services for analysis of large data sets are deployed on modern IT infrastructures, such as computer clusters, grids and clouds. Efficient data transfer between remote components of distributed infrastructure is very important in order to achieve interactive visualization rates.

Fracture of materials presents challenge for visualization as well as for a wide range of other multi-disciplinary sciences. Cracking is a very common phenomenon investigated by a wide research community in different scientific areas (Gobron *et al.* 2001). Crack formation is often observed in building constructions, in ceramics (Uematsu 2014), in drying processes (Kitsunezaki 2011) and in complicated failure of powder agglomerates (Khanal *et al.* 2009). The absence of the explicitly defined crack surfaces limits a visual analysis of the computed results and complicates the understanding of the investigated fracture phenomena. Moreover, the crack surface geometry accurately defined by graphics primitives can be exported to engineering software for following analysis at the macro-level.

The Object of the Research

The object of research – visualization of discrete particle systems by using distributed software.

The Aim of the Thesis

The research aims at improving the visualization methods of discrete particle systems and increasing speed of distributed visualization software.

The Objectives of the Thesis

In order to achieve the aim, the following have to be solved:

1. To analyze technologies of distributed visualization software and visualization methods for extraction of crack surfaces from discrete particle systems.
2. To develop software implementation, which allows reducing communication between remote components of gLite/EMI grid infrastructure, and investigate its performance.
3. To develop visualization methods for extraction of crack surfaces from discrete particle systems.
4. To implement the developed methods into distributed visualization software.
5. To investigate and compare the speed of the implementation of the developed visualization methods.
6. To investigate the accuracy of the developed visualization methods.

Research Methodology

Methods of comparative and literature analysis methods were used to analyze the research object. Methods of computer graphics and computational geometry are used to develop visualization methods for crack propogating. The methods of experimental research have been used for examining performance of proposed methods and developed software.

Scientific Novelty of the Thesis

The main scientific contributions of the presented research are as follows:

1. The original implementation of partial data set transfer from gLite/EMI grid storage elements has been developed to reduce the communication between remote components of grid infrastructure in the case of interactive visualization.
2. Novel visualization methods based on Voronoi and geometric cell centre decompositions have been proposed for extraction of crack surfaces from monodispersed particle systems. The local decompositions must be

constructed according to the fixed connectivity of the moving particles, therefore, the commonly used methods for generation of Voronoi decompositions or Delaunay triangulations cannot be applied.

Practical Value of the Research Findings

The designed visualization software is employed for analysis of computational results of large discrete particle systems that can be useful for developing high technology in Lithuania. The novel Voronoi and cell centre-based methods can be used by researchers of fracture mechanics and the material sciences to speed–up the design of new structures and materials. Moreover, the commercial finite element analysis software, which is widely used by engineers, can import only continuously defined crack surfaces represented by curves or polygon meshes. The successful research into the problems of extraction and visualization of crack surfaces can help to fill the gap between the industrial requirements and the research results. The developed methods are implemented in the software, which is investigated in the project "Research and development of technologies for virtualization, visualization and security e-services" (VP1-3.1-ŠMM-08-K-01-012).

The Defended Statements

1. The developed partial data set transfer from gLite/EMI grid storage elements is able to decrease visualization time reducing the data size transferred between remote components of grid infrastructure.
2. The proposed visualization methods are capable of extracting crack surfaces and visualizing crack propagation modelled by the discrete element method in monodispersed particulate media.

Approval of the Research Findings

Research results related to the dissertation subject are published in 6 scientific publications. Three of them are published in reviewed scientific journals and are included in the Thomson Reuters Science Citation Index.

The results of the diseration were presented at five conferences. Four of them are presented at international scientific conferences:

- The Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG2015). March 24–27, 2015, Dubrovnik, Croatia.
- 7th World Congress on Particle Technology (WCPT7), 2014 May 19–22, Beijing, China.

- The Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG2013), 2013 March 25–27, Pécs, Hungary.
- 18th International Conference on Information and Software Technologies (ICIST 2012). September 13–14, 2012, Kaunas, Lithuania.
- LMA IIth conference of young scientists on "Fizinių ir technologijos mokslų tarpdalykiniai tyrimai". February 14, 2012, Vilnius, Lithuania.

Dissertation Structure

The dissertation consists of introduction, three main chapters, general conclusions, references, a list of publications by the author on the topic of the dissertation and a summary in Lithuanian. The total scope of the dissertation is 126 pages, 10 equations, 63 figures and 6 tables. For the purpose of the present dissertation, references were made to 110 source papers.

1

Overview of distributed visualization systems and visualization methods

In this Chapter a literature review on state-of-the-art visualization software, distributed environments and common visualization techniques is presented. Discrete Element Method (DEM) for crack propagation in discrete particle systems is introduced to formulate crack visualization problem and to discuss output datasets and resulting difficulties. Finally, surface extraction and crack visualization methods that can be applied in the case of DEM are reviewed and discussed.

Parts of this Charpter are published in (Kačeniauskas and Pacevič 2011), (Pacevič *et al.* 2013), (Pacevič, Kačeniauskas, *et al.* 2015), (Kačeniauskas *et al.* 2012), (Kačeniauskas *et al.* 2013), (Pacevič and Kačeniauskas 2015).

1.1. Overview of Visualization Software

Visualization is used for analyzing the data and presenting the results across a wide range of disciplines (Hansen *et al.* 2005). Computers are used to create visual images from the data, while the human mind is used to make inferences from this imagery in order to better understand the data. Some standalone visualization

systems, toolkits and packages should be overviewed in order to choose the most appropriate platform for the efficient development of distributed visualization systems. In the dawn of the visualization era, creating visualizations meant programming using a low-level graphics library. A new approach was brought forward by the Application Visualization System (AVS) (Favre *et al.* 2005). AVS (later called AVS Express) is a modular visualization environment (MVE) providing an application development environment for visualization using a visual network editor. Over 800 developed modules are available in AVS/Express visualization system. Other MVEs have significantly smaller number of modules (about 200). AVS/Express Multipipe Edition is a version of AVS/Express, which contains multi-pipe rendering extensions for use in virtual environments. It is targeted at high-level SGI systems running IRIX and clusters of Windows-based PCs.

OpenDX (Thompson *et al.* 2000) is a modular visualization system based on the dataflow model. The visualization pipeline is processed by the modules of the dataflow graph, which are arranged according to the desired output. This MVE originally began as the IBM commercial product Visualization Data Explorer, but was offered by IBM as an Open Source project in 1999. OpenDX is the best-known open-source package in the category of MVEs. High efficiency of the software is achieved by optimized cashing and hardware rendering by means of OpenGL (Shreiner *et al.* 2013). OpenDX can access modules on remote hosts by DXLink. Distributed memory architectures can be employed as well as shared memory supercomputers.

IRIS Explorer (Foulser 1995) is another MVE based on the dataflow model. IRIS Explorer uses Open Inventor (Wernecke 1994) for lower level rendering. The system provides a large selection of modules, listed in the Librarian, which the user launches into the workspace (Map Editor) and connects together with wires to form a dataflow pipeline, or map. The system can be extended by users adding their own code as modules and integrating them into IRIS Explorer using the provided API. This system provides a mechanism to allow modules within a pipeline to be run on a number of remote computing resources. IRIS Explorer transparently manages the data transfer between resources as the data passes along the pipeline, using shared memory for modules connected together on the same host and through sockets for modules connected across host boundaries.

SCIRun (Parker *et al.* 1995) is an MVE developed at the Scientific Computing and Imaging (SCI) Institute at University of Utah. It allows the user to connect a set of pre-written routines together in a workspace to form a dataflow network. These routines execute as independent threads within a single executable in contrast to other MVEs, which run modules as individual processes. SCIRun has been targeted at shared memory parallel systems. To extend beyond this limitation, more recent implementations of SCIRun have employed “bridging”.

SCIRun has been used for computational steering within the Uintah, which is designed to run on massively parallel distributed memory architectures.

Visapult (Bethel *et al.* 2000) is a visualization framework with the ability to render a huge amount of data sets (of the order of 1-5 Tb). Visapult uses parallel rendering hardware to carry out the high speed rendering processes. Using Cactus (Thomas *et al.* 2010) the data are distributed amongst many parallel nodes for volume rendering. The rendered subset of 2D image is sent to the client for local rendering.

Chromium (Humphreys *et al.* 2002) is an open source graphics library, which allows distributed network rendering for OpenGL applications. It does this by intercepting OpenGL API calls (Shreiner *et al.* 2013), and then modifying, deleting, replacing or augmenting them. Thus, for distributed rendering, the commands are split across a collection of graphics cards. A particular feature of Chromium is that it is non-invasive, because no modification or even recompiliation of the application is required.

pV3 (Haimes 1994) is a library for the real-time visualization of large-scale transient (unsteady) systems. Based on an earlier system called Visual3, it has been re-designed specifically for the multi-processing visualization of data generated in a distributed compute arena. One of its design goals is to allow the compute solver to run as independently as possible thus for example, pV3 can be configured to plug into the simulation, display the transient data, and unplug from the calculation. pV3 provides a centralized interface to a distributed computation. Computational steering is also supported by pV3.

Covise (Collaborative Visualization Environment) (Wright *et al.* 1997) falls into the modular visualization environment category. It allows a user to run modules both locally and remotely, employing a data manager process on each host to manage the flow of data beween modules. Like other distributed MVEs, such as IRIS Explorer (Foulser 1995), modules connected together on the same host communicate data through shared memory, while modules connected beween hosts pass data via the local data managers. A company named Vircinity is marketing several versions of Covise that offers collaborating visualization.

Ensight (CEI 2009) is a standalone application aimed at the visual analysis and post-processing of engineering data. An advanced version of the package (called Ensight Gold) incorporates extra features for handling large data sets including parallel processing, multi-pipe rendering for output to immersive environments, and collaboration. Ensight offers a standard set of visualization and plotting algorithms with interfaces to several engineering solvers for CFD and FEA problems but, being aimed at end-users rather than developers, is not extensible. No new algorithms can be added, although users can create their own readers to import data in custom formats.

Amira (Stalling *et al.* 2005) is an highly interactive object-oriented visualization system, based on Open Inventor (Wernecke 1994). Unlike many of the other visualization systems, it is not based on a dataflow model. Because data are passed between modules as C++ objects (as in a normal C++ program), there is no overhead for module communication. Users can extend Amira (adding new modules, data classes, editors and I/O methods) by deriving from an existing C++ class. Amira users create applications via a visual programming interface by connecting Amira modules together.

VisAD (Hibbard *et al.* 2005) is a Java component library for interactive and collaborative visualization and analysis of numerical data. It makes use of Java's RMI technology, which allows methods of remote Java objects to be invoked from other Java virtual machines, possibly on different hosts. VisAD applications can run in any of three networked modes: standalone, server or client. The construction of distributed applications in VisAD is facilitated by its event-driven design.

VTK (Kitware 2010; Schroeder *et al.* 2006) is an open source, object–oriented software system providing a toolkit for 3D computer graphics, image processing and visualization. It consists of a C++ class library, together with several interface layers including Tcl/Tk, Java and Python, which can be used to access the classes and build applications. More than 2000 separate classes, including several hundred data processing filters, are available in the toolkit. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modelling methods like implicit modelling, polygon reduction, mesh smoothing, cutting, contouring and Delaunay triangulation. VTK is based on the dataflow model supporting reference counting, which allows data to be shared instead of duplicated. Modules are connected with each other to form a visualization pipeline. VTK supports portable multithreading for shared memory implementation and portable distributed parallel processing based on MPI (Gropp *et al.* 1996). Likewise, a sort-last, parallel rendering class is provided that uses inter–processor communication to collect and then composite parallel renderings into a final image (Moreland *et al.* 2001). An open-source, turnkey application ParaView (Ayachit 2015), designed for large data visualization using distributed parallel processing, is built on the top of VTK.

1.2. Distributed Environments for Visualization

Scientific visualization is becoming increasingly important in analyzing and interpreting numerical and experimental data sets. However, large data sets and a complex visualization process require considerable development efforts and

impressive computing resources. Distributed visualization allocates different parts of the machine processing to different computers.

Grid Visualization Kernel (GVK) (Kranzlmuller *et al.* 2002) based on the Globus middleware (Foster *et al.* 2001) has been developed in EU CrossGrid project (2002). In concrete, the concept of GVK can be divided into two main parts: the interface between the simulation and the visualization, and the implementation of the visualization pipeline. The interface to GVK is established via available visualization toolkits. Well known example is the OpenDX visualization software (Thompson *et al.* 2000). The difference between the traditional approach and the GVK visualization services is that some modules of the dataflow graph are replaced by corresponding GVK modules. The advantage of this approach is that it enables the user to define arbitrary visualization pipeline configurations using the well-known dataflow approach. The pipeline can be split at any point, and the processing modules can be distributed across the grid as desired.

The portal GridSphere (Russell *et al.* 2004) providing interface for grid visualization was developed in the FP5 GridLab project (2003–2005). Data Management and Visualization Work package of the project provides the services that are needed by the grid to manage public and user-private files and creates a framework that enables applications to stream data for visualization purposes.

A lot of scientific applications of visualization (Wood *et al.* 2007) were demonstrated in projects supported by UK e-Science Core Programme. gViz (Visualization Middleware for e-Science) (Brodlie *et al.* 2003) had two main targets: first, to grid-enable two existing visualization systems, IRIS Explorer (Foulser 1995) and pV3 (Haimes 1994), so that visualization tools are available as early as possible for users of computational GRID; second, to develop some longer term thinking on distributed and collaborative visualization, where XML languages (Evjen *et al.* 2007) are used to describe visualization data, and visualization programs themselves. The modules of IRIS Explorer in a network can be distributed across a set of grid resources (Li *et al.* 2005), but controlled from the desktop. This is achieved in a secure manner using Globus middleware (Foster 2005). The COVISA (Wright *et al.* 1997) collaborative facilities become immediately available. An important application of this is in computational steering, where the simulation model runs on a remote server, but is controlled from the desktop. The grid-enabled version of pV3 is created by replacing its PVM-based communications with a web service mechanism using the gSOAP package (Robert A. Van Engelen 2002). The distributed computation provides a web service offering data for visualization and the pV3 “servers”, then connect to this as web service clients. gSOAP provides an efficient C/C++ web service implementation. The use of an XML language to describe visualization data

promises to lead to better inter-operability of visualization systems, and the more effective development of new visualization software.

The grid-Aware Portals toolkit (GAPtk) project (Nagella *et al.* 2005) aimed to provide scientists with a set of visualization utilities based on the Web and grid services (Li *et al.* 2005) model along with appropriate APIs that will enable them to exploit the power of the grid their data analysis. The layered architecture of the toolkit consists of a set of interfaces on the client side, which talks to the GAPtk server using SOAP (Englander 2002); the server communicates with the grid using Globus (Foster 2005). The toolkit was used to provide visualization capabilities within the GODIVA project, which was investigating ocean circulation and its effect on climate change.

The main aim of the project Visual beans (Cooper 2002) was to investigate the role of component technology together with advanced middleware platforms to support the construction of dynamically adaptable distributed cooperative visualization software. The project developed and extended a middleware platform called TOAST (which is CORBA-based (Brose *et al.* 2001)). A number of experimental systems were constructed as demonstrators and proof-of-concept implementations of the framework and visualization components, the latter constructed using VisAD visualization software (Hibbard *et al.* 2005).

The goal of the ICENI (Mayer *et al.* 2005) project was the provision of high-level abstractions for scientific computing, which will allow users to construct and define their own applications through a graphical composition tool. The project aimed to deliver this environment across a range of platforms and devices on the grid using a scheduling service. ICENI was being implemented in Java and JINI, but was able to interoperate with the Open Grid Services Architecture (OGSA). One of the applications of ICENI is computational steering and visualization (Stanton *et al.* 2002). The ICENI framework is used to link together a visualization client and server, and to pass data to the server from a running application. The visualization server hands the data off to a renderer (current demos are based on VTK (Kitware 2010)), which can then send the graphical output back to the visualization client. This can either be done using standard OpenGL remote rendering, or using Chromium (Humphreys *et al.* 2002). The ICENI project has also made use of the way Chromium can be configured with the ACE networking framework (Schmidt 2009) in order to send its graphics as a video stream directly to a multicast address; this effectively provides a bridge between Chromium's distributed rendering and the Access Grid.

VISPORTAL (Bethel *et al.* 2003) was built by upon the Grid Portal Development Toolkit (GPDK) (Novotny 2002). The Visualization Group of Lawrence Berkeley National Laboratory and National Energy Research Scientific Computing Center efforts explores ways to deliver advanced remote/distributed

visualization capabilities through a Grid-enabled web-portal interface. The effort focuses on latency tolerant distributed visualization algorithms and GUI designs that are more appropriate for the capabilities of web interfaces. The developed clients involve very thin GUIs such as Java applets lauched by the web-browser on the client machine that provide a front-end for massively parallel or distributed/multi-tier visualization back-ends like Visapult (Bethel *et al.* 2000) or offscreen rendering pipe access. Finally, the thick clients simply use the portal as a broker for locating remote data or services that extend the capabilities of a standalone tool like OpenDX (Thompson *et al.* 2000) or AVS Express (Favre *et al.* 2005). Eventually the VisPortal framework will migrate to a portlet OGSA model offered by GridSphere (Russell *et al.* 2004).

Cactus (Thomas *et al.* 2010) is an open source problem solving environment designed for scientists and engineers. Cactus consists of a central core component, called the flesh, and a set of modules called thorns. The thorns implement a range of computational codes, which can run on distributed computing resources while being connected to, and orchestrated by, the flesh. The flesh controls when thorns will execute and how data is routed between them. Cactus builds on the Globus Toolkit (Foster 2005) to provide secure access to remote resources, together with secure communications and job scheduling on remote resources. It also uses a number of other standard libraries and toolkits such as PETSc (Brune *et al.* 2014) for scientific computation and HDF5 (Folk *et al.* 2011) for data output. Visualization is provided via standard products such as OpenDX (Thompson *et al.* 2000), Amira (Stalling *et al.* 2005) and IRIS Explorer (Foulser 1995). These systems effectively operate as thorns connected to the Cactus system via special modules written for each system, which are able to read the data formats exported by Cactus (for example, HDF5) using the Cactus API.

RealityGrid (RealityGrid 2005) was a project, which aims to examine how scientists in the condensed matter, materials and biological sciences communities can make more effective use of the distributed computing and visualization resources provided by the grid. RealityGrid is making use of visualization as part of distributed applications in which the simulation in one place communicates with the visualization in another and a steering client in a third. The RealityGrid steering architecture is built around a library, calls to which are embedded into each of the three components (simulation, visualization and client). Because of difficulties experienced in integrating existing MVEs into larger distributed applications, RealityGrid has selected VTK (Kitware 2010; Schroeder *et al.* 2006) as a lower-level environment, along with enabling technologies such as VizServer (SGI 2009) and Chromium (Humphreys *et al.* 2002). In addition, ICENI, which use much of the same technology is being used to enable collaborative visualization and computational steering within RealityGrid.

Grid Visualization System (Naregi 2005) for distributed massive data was developed in by National Institute of Informatics and NEC in 2005**.** Generalized grid visualization services were adopted for various visualization functionalities provided by various visualization tools. Image based remote parallel visualization based on MPI was employed. FUJITSU corporation developed collaborative visualization grid environment VizGrid (Matsukura *et al.* 2004) for natural interaction between remote researchers.

The NVIDIA grid hardware serves as the foundation for their on-demand Gaming as a Service (GaaS) solution. NVIDIA GRID vGPU (Nvidia 2014) technology brings the benefit of NVIDIA hardware-accelerated graphics to virtualized solutions on cloud infrastructures (Mohammad *et al.* 2012). Amazon EC2 GPU Instance Type G2 provides graphics as a service (Yegulalp 2013) on AWS cloud platform that includes Nvidia GPUs.

1.3. Discrete Particle Systems

The discrete element method (DEM) referred to the original work of Cundall and Strack (Cundall *et al.* 1979) has been extensively used in numerical analyses of discrete particle systems from the perspective of science and engineering. The main advantage of the DEM is a possibility to model highly complex discrete particle systems using the basic data on individual particles and avoiding oversimplifying assumptions of continuum. The method allows the simulation of motion and interaction between the particles, taking into account the microscopic geometry and various constitutive models.

The DEM has been extensively applied to examine different phenomena inside the granular materials. The granular flow from hoppers and silos has a wide range of applications in industry (Zhu *et al.* 2008). The study of the bulk material pressure on the walls of a hopper is very important for hopper design (Goda *et al.* 2005). The prediction of the discharge rate is of importance for the effective operation and control of a transport system and is difficult due to inhomogeneous solid distribution, irregular velocity profile and diverse particle size (Kruggel *et al.* 2009). It is very important to understand the microscopic structure and its relations to the governing mechanisms (Parisi *et al.* 2004). DEM simulation takes into account the discrete nature of granular materials, and therefore is very effective for this purpose. The combined approach of DEM and averaging method offers a convenient way to link fundamental understanding generated from DEM-based simulations to engineering application often achieved by continuum modelling (Zhu *et al.* 2007). Over the past decade, the DEM was utilised in a variety of industrial applications (Cleary 2009; Radeke *et al.* 2010).

The main advantage of using DEM for simulation of granular systems is that, by tracking the motion of each individual particle, the detailed information about the system behaviour across a range of time- and length-scales can be obtained. However, the simulation of systems at this level of detail has the disadvantage of making DEM computationally very expensive. DEM simulations on a single workstation or ordinary PC tend to be limited to systems of several tens of thousands of particles and short time intervals. The recent simulation of large-scale systems is performed by employing the parallel computation techniques, though the number of the used particles was still much smaller than that required in industry where typically over a billion particles are dealt with. Distributed or parallel DEM computations have become an obvious option for rapidly increasing computational capability, along with recent remarkable advances in distributed software systems and computational infrastructures like computer clusters, grids and clouds.

Several different particle- and lattice-based approaches (Zhu *et al.* 2008) have been developed in the frame of DEM applied to the simulation of solids and structures. Particle-based approach (Kačianauskas *et al.* 2010; Walther *et al.* 2009) is, actually, a rather straightforward extension of the original DEM. A solid is replaced with a composition of discrete particles, where the presence of the cohesive forces acting between the particles and various mechanisms of their linkage and detachment may be allowed (Markauskas *et al.* 2009). The approach, where the continuum may be represented by the material particles, interacting via the network elements, is referred to as the lattice-based model. A comprehensive review of the planar elastic lattice models for micro-mechanical applications is given by (Ostoja-Starzewski 2002). The lattice-based DEM (Kozicki *et al.* 2008; Lilliu *et al.* 2003) has been extensively applied to the simulation of heterogeneous solids to study their dynamic deformation behaviour and fracture problems. The simplest lattice – based DEM models for continuum exhibit structural analogy and equivalence to frame or truss structures that are widely modelled by the finite element method (Barauskas *et al.* 2004). However, the lack of highly accurate and reliable solutions at the industry level limits the use of DEM models in the commercial codes.

The dynamic behaviour of the non-cohesive frictional visco-elastic particle system can be simulated by the discrete element method. This system consists of the finite number of deformable spherical particles with the specified size distribution and material properties. Any particle i in the system of N particles undergoes the translational and rotational motion, involving the forces and torques originated in the process of their interaction. Although a description of the translational motion is always independent of the particle shape and is written in a linear form, the rotational motion is of the same character only for a perfectly symmetric shape, such as sphere, where the inertia tensor is defined by a single

parameter. Finally, the motion of the *i-th* contacting spherical particle in time *t* is described as follows:

$$m_i \frac{d^2 \mathbf{x}_i}{dt^2} = \mathbf{F}_i \tag{1.1}$$

$$I_i \frac{d^2 \boldsymbol{\theta}_i}{dt^2} = \boldsymbol{T}_i \tag{1.2}$$

where m_i, and I_i are the mass and the moment of inertia of the particle, respectively, while vectors $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_i$ define the position of the particle centre and the orientation of particle *i*. The vectors $\boldsymbol{F}_i$ and $\boldsymbol{T}_i$ present the sum of the contact force and the gravity force as well as the respective torques:

$$\boldsymbol{F}_i = \sum_{j=1, j\neq i}^{N} \boldsymbol{F}_{ij,cont} + m_i \boldsymbol{g} \tag{1.3}$$

$$\boldsymbol{T}_i = \sum_{j=1, j\neq i}^{N} \boldsymbol{T}_{ij} = \sum_{j=1, j\neq i}^{N} \boldsymbol{d}_{cij} \times \boldsymbol{F}_{ij,cont} \tag{1.4}$$

where *g* is the acceleration due to the gravity, d_{ij} is the vector pointing from the particle centre to the contact centre. The interparticle force vector $F_{ij,cont}$, describing the contact between the particles *i* and *j*, may be expressed in terms of normal and tangential components. The normal component, presenting a repulsion force, comprises elastic and viscous ingredients, while the tangential component reflects static or dynamic frictional behaviour. The static force describes friction prior to gross sliding and comprises elastic and viscous ingredients, while the dynamic force describes friction after gross sliding and is expressed by the Coulomb's law.

For evaluating the contact forces (1.3–1.4), all contacts between the particles and their neighbours must be detected. A cell-based method (Han *et al.* 2007) is used for contact detection. The numerical integration of the equations of motion (1.1–1.2) is performed to obtain the dynamical state of all particles at the time *t*, resulting from the action of the particle forces (1.3–1.4). The solution of these equations is obtained by using the Verlet scheme.

The materials can be modelled using the elastic perfectly brittle model of contact interaction. The lattice-based discrete element model assumes cohesive bonding between the neighbouring particles (Rojek *et al.* 2011). These lattice connections represented by springs can be broken under excessive loading, which allows us to simulate initiation and propagation of the material fracture. When

two particles are bonded, the contact force in normal direction is calculated from the linear constitutive relationships:

$$\boldsymbol{F}_{n,\,cont} = k_n u_n \tag{1.5}$$

where $F_{n,cont}$ is the normal contact force, k_n is the spring stiffness in the normal direction, u_n is the normal relative displacement. The lattice connections are broken instantaneously, when the interface strength is exceeded in the normal direction by the tensile contact force. The failure criterion can be written as:

$$\boldsymbol{F}_{n,\,cont} \leq R_n \tag{1.6}$$

where R_n is the interface strength in the normal direction. Although a compressive interaction force between the particles does not cause breakage of the connections, the material damage under macroscopic compression can be represented properly in the particle model. A compressive macroscopic load brings about tensile interactions at the microscopic level, which may lead to connection failures.

After breakage of connections, a normal contact force is calculated, using the Hertz contact model (Zhu *et al.* 2008). A frictional interaction can occur among the particles in the case of compression. The limiting friction force is evaluated, assuming the Coulomb model of friction:

$$\left|\boldsymbol{F}_{t,\,cont}\right| = \mu\left|\boldsymbol{F}_{n,\,cont}\right| \tag{1.7}$$

where μ is the Coulomb friction coefficient. The simple mathematical model (1.1–1.7) of the lattice-based DEM is provided for the sake of completeness. However, the simulation results will not be examined in detail because the presented research is concentrated on the surface extraction and the proposed visualization methods.

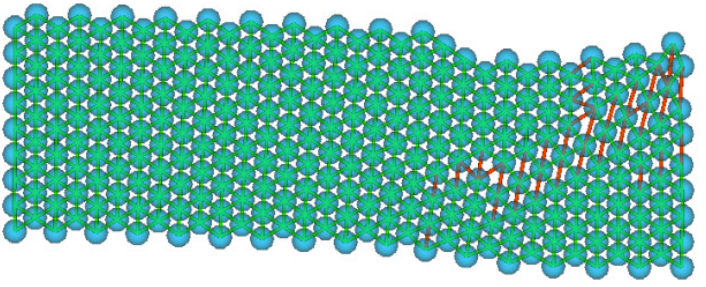


Fig. 1.1. Schematic representation of the lattice-based DEM model

DEM computations are based on the positions of particles and forces acting between them. The lattices (Figure 1.1) employed in DEM computations are assembled from the 1D springs or beams, which are not well suited for the reliable interpolation and common visualization methods used in 3D. The data sets of DEM include the positions, velocities and accelerations of particles as well as their radii and material properties. The 1D connections between the neighbouring particles represent the lattice topology. The fracture force limit and various forces are used as the attributes of the connections. The most important attribute array named the connection state indicates, when the lattice connection is broken.Based on the described approach, the defects between the pairs of the neighbouring particles are identified by the broken lattice connections (Lilliu *et al.* 2003; Vadluga *et al.* 2009).

1.4. Extraction and Visualization of Crack Surfaces

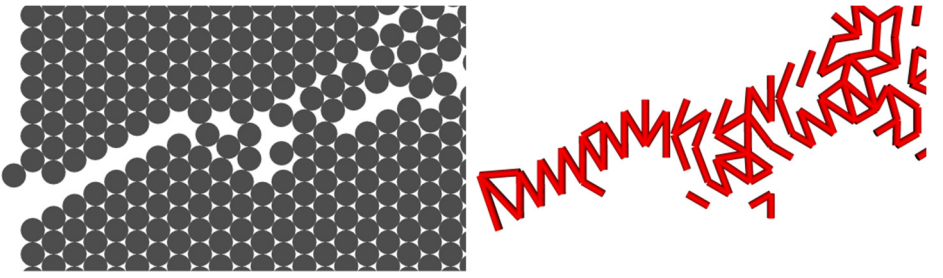
Cracking is a very common phenomenon investigated by a wide research community in different scientific areas (Gobron *et al.* 2001). Crack formation is often observed in ceramics made through powder compaction process (Uematsu 2014), in drying processes (Kitsunezaki 2011), in soil (Valette *et al.* 2006) and in complicated failure of powder agglomerates (Khanal *et al.* 2009). Geometric models propose the algorithms for obtaining crack patterns close to those produced by nature. The fracture models can be mapped onto the surface of the object, while fractures are created procedurally (Gobron *et al.* 2001). In contrast, physical approaches propose the models, which tend to accurately simulate the dynamics of natural crack patterns. Federl and Prusinkiewicz (Federl *et al.* 2004) used a physical approach to fracture modelling based on the finite element method. Gobron and Chiba (Gobron *et al.* 2001) proposed a different approach for simulating realistic propagation of various types of cracks on any triangulated surface, which was based on their 3D cellular automaton model. O'Brien *et al.* (Iben *et al.* 2009) succeeded in modelling brittle and ductile fractures by means of the finite element method. Iben and O'Brien have extended the previously presented methods in order to address the issue of quasi–static fractures (Iben *et al.* 2009). The surface is discretized by means of the finite element method and any 3D mesh can be cracked with an heuristic definition of stress fields. A dynamic model of cracks development based on a 3D discrete shrinkage volume propagation is proposed by Valette *et al.* (Valette *et al.* 2008). The DEM provided the valuable insight into the fracture phenomena at the particle level (Rojek *et al.* 2011; Vadluga *et al.* 2009). Researchers used this method to study the damage of heterogeneous solids such as concrete (Khanal *et al.* 2009) or rock (Rojek *et al.*

2011) and homogeneous materials such as ceramics (Tan *et al.* 2009) or glass (Zang *et al.* 2007).

The problem of reconstructing a surface from a set of sample points is motivated by numerous applications and, consequently, has always been a popular field of research. Most surface reconstruction methods roughly fall into two major categories: implicit surface methods (Fleishman *et al.* 2005; Kolluri 2008; Rosenthal *et al.* 2008) and Delaunay-based methods (Amenta *et al.* 2001; Dey *et al.* 2003; Labatut *et al.* 2009). A surface can be implicitly defined as a level-set of the function allowing smooth and approximating surface reconstruction. Moving least squares can handle a moderate amount of noise and be used to define the implicit functions with the signed distance to local planes as local approximants. Therefore, reconstruction guarantees are provided for sufficiently dense and uniform point clouds (Kolluri 2008). An approach that directly extracts smooth surfaces from unstructured point-based volume data without prior resampling or mesh generation is proposed by Rosenthal and Linsen (Rosenthal *et al.* 2008). Another most common approach to surface reconstruction is based on the Delaunay triangulation: the underlying idea is that when the sampling is noise free and dense enough, points close on the surface should also be close in space. Among the Delaunay-based methods, the most well-known algorithms are, preferably, the Crust (Amenta *et al.* 2001) and the Cocone (Dey *et al.* 2003) families of algorithms. Power Crust (Amenta *et al.* 2001) relies on the power diagram, the weighted Voronoi diagram of the poles. Labatut *et al.* (Labatut *et al.* 2009) formulate the reconstruction problem as an energy minimisation on the Delaunay triangulation. In the case of lattice-based DEM methods, the discussed surface extraction algorithms cannot be directly applied, because of the absence of suitable data defining the crack surface and the complex nature of crack surfaces, defined by the scattered results of numerical computations at the micro-level. The holes and disjoint pieces of crack surfaces make surface extraction extremely complicated.

Computer graphics scientists focus their attention on enhancing the realism of natural scenes, while computational researchers concentrate on building accurate numerical models. Furthermore, their principal aim is not closely related to visually attracting results. In the DEM methods, cracks and the related phenomena are often visualized in the most straightforward way. The particles coloured depending on particular scalar attributes (Bertin 2010), such as the initial high altitude or radii, are most common (Kačeniauskas, Kačianauskas, *et al.* 2011; Rojek *et al.* 2011; Zang *et al.* 2007). The propagation of large cracks is illustrated (Figure 1.2a) by using the rendered geometry of particles (Cusatis *et al.* 2006). However, this technique can be applied only to large cracks, which are of the particle size. In the case of smaller cracks, when some connections are already broken, but gaps between particles remain significantly smaller than the particle

diameter, geometry of the neighbouring particles can not properly visualize crack surface propagating in 3D. The coloured lattice connections (Karihaloo *et al.* 2003; Liu *et al.* 2007) can be treated as the main alternative to the rendered particles. The broken connections between the neighbouring particles (Figure 1.2b) indicate the fractured regions, but do not provide any valuable information about the formation of the crack surfaces.



a) b)

Fig. 1.2. Crack propagation illustrated by using: a) the rendered geometry of particles, b) the tubes on broken connections between the neighbouring particles

The main visualization task of the presented research is to construct the surfaces of cracks from the broken lattice connections and the geometry of the neighbouring particles. In 3D space propagating crack surface can be defined by 2D graphical primitives that contain more information than 1D graphical representations employed in above discussed visualization methods based on lattice connections (Figure 1.2). 2D graphical representations can be valuable until gaps between the neighbouring particles exceed the particle diameter and material starts crumbling. The complex disjoint surfaces of cracks and the unavailability of a suitable scalar field defining the crack surfaces limit the application of the widely used surface extraction methods. Thus, there is hardly any direct method of constructing the surfaces of propagating cracks from individual defects determined between the neighbouring particles and applying the available visualization methods. Moreover, the commercial finite element analysis software widely used by engineers can import only continuously defined crack surfaces represented by usual graphic primitives. The successful research into the problems of representation and visualization of crack surfaces can help to fill the gap between the industrial requirements and the research results currently obtained in the areas of fracture mechanics and material sciences.

The Voronoi diagrams can be applied to post processing of discrete particle systems as well as to lattice forming. Given a set of primitives, the Voronoi diagram partitions space into regions, where each region consists of all points that are closer to one primitive than to any other. The Voronoi diagrams have important applications in many sciences, including visualization of medical data

sets, proximity queries, spatial data manipulation, shape analysis, computer animation, robot motion planning, modelling spatial structures and processes, pattern recognition, locational optimization and selection in user interfaces (Aurenhammer 1991; Klein *et al.* 2009). Fast and resolution independent computation of the Voronoi treemaps, based on a combinatorial algorithm for the weighted Voronoi diagrams, was presented by Nocaj and Brandes (Nocaj *et al.* 2012). GPU based computation of the 3D discrete Voronoi diagrams was used for surface extraction by Rosenthal and Linsen (Rosenthal *et al.* 2009). A concept of the Centroidal Voronoi tessellation was presented in the form of graphs in (Lu *et al.* 2012). The edges of radical Voronoi diagrams were employed to construct a beam-network model for autoclaved aerated concrete (Kadashevich *et al.* 2008). Computational lattices for polydispersed particulate media are assembled according to the Voronoi diagrams by (Cusatis *et al.* 2006). However, there were no attempts to describe the contact surfaces of particles and to visualize the propagating cracks by using local space decompositions based on the Voronoi cells or the geometric cell centre. Moreover, no attempts were made to extract the crack surfaces and explicitly define them by graphics primitives in the regions of highly deformed computational lattices.

1.5. Conclusions of Chapter 1 and Formulating Task for the Dissertation

1. The most of the overviewed grid environments for visualization are based on the Globus middleware and its toolkit for service development. The important Globus functionality cannot be accessed in the gLite grid environment, therefore, most of the available visualization software cannot be applied.
2. Moreover, it is difficult to find a general purpose grid visualization e–service, which can visualize the results produced by engineering applications at interactive rates on the gLite/EMI grid infrastructures.
3. The lattices employed in DEM computations of discrete particle systems are assembled from the 1D springs or beams, which are not well suited for the reliable interpolation and common visualization techniques used in 3D.
4. The complex nature of cracks, propagating in particle media and the unavailability of a suitable scalar field defining the crack surfaces limit the application of the common surface extraction methods to visualization of cracks, defined by the scattered results of numerical computations at the micro-level. The holes and disjoint pieces of crack surfaces make surface extraction extremely complicated.

5. There is hardly any direct technique of constructing the surfaces of propagating cracks according to the connectivity of the broken lattice connections and the geometry of the neighbouring particles. Moreover, there were no attempts to visualize the propagating cracks by using local space decompositions based on the geometrical cell centres.

In order to achieve the aim, the following have to be solved:

1. To develop software implementation, which allows reducing communication between remote components of gLite/EMI grid infrastructure, and investigate its performance.
2. To develop visualization methods for extraction of crack surfaces from discrete particle systems.
3. To implement the developed methods into distributed visualization software.
4. To investigate and compare the speed of the implementation of the developed visualization methods.
5. To investigate the accuracy of the developed visualization methods.

2

Visualization Methods and Distributed Software for Discrete Particle Systems

In this Chapter novel methods are introduced for visualization of crack surfaces from monodispersed particle systems. The proposed methods for visualization of cracks were implemented in the devbeloped prototypes of grid visualization e–service VizLitG and the distributed visualization software VisPartDEM. Partial data set transfer from the grid storage element was developed to reduce the transferred data size and visualization time.

The methods presented in this Chapter are published in (Kačeniauskas and Pacevič 2011), (Pacevič *et al.* 2013), (Pacevič, Kačeniauskas, *et al.* 2015), (Kačeniauskas *et al.* 2012), (Kačeniauskas *et al.* 2013), (Pacevič and Kačeniauskas 2015).

2.1. Cell Attribute- and Cell Cut-based Methods

The cell attribute-based method and the cell cut-based method are developed for visualization of cracks propagating in monodispersed particle systems. The cell cut-based surface extraction method resulted as the extension of the cell attribute-

based method in order to define crack geometry by graphics primitives. The developed visualization methods generate global or local space decompositions from the 1D connections of the computational lattice. Generation of space decompositions from lattice connections of lower dimensionality is a time-consuming procedure. The commonly used mesh generation methods cannot be applied because of highly different input data. The vertices of the cells are determined by the particle positions. Moreover, one–dimensional lattice connections should be employed as the edges of the newly generated cells and local decompositions rather than global meshes should be used in order to save computational resources.

2.1.1. Cell Attribute-based Method

The simple cell attribute-based method is proposed for visualization of cracks in monodispersed particulate media. Figure 2.1 shows the application of the cell attribute-based method in 2D. The cell attribute-based method covers the whole computational domain by generated cells. The proposed method employs the one-dimensional lattice connections as the edges of the newly generated cells. In Figure 2.1, thin black lines represent the unbroken lattice connections, while thin red lines represent the broken connections. The method simply calculates how many 1D connections of the particular cell are broken and assigns this value to the cell attribute. In this case, spatial crack representation can be visualized as the scalar cell attribute by using colour mapping and predefined colour table. In Figure 2.1, the triangles including only one broken connection are coloured in cyan, while the cells that have two broken connections are coloured in yellow. Triangles containing three broken connections are coloured in red, while cells that have not broken connections are not displayed in Figure 2.1.

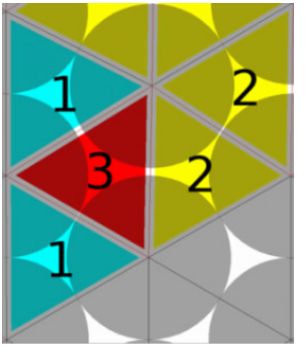


Fig. 2.1. Illustration of cell attribute-based method

The general scheme of the method is presented in Figure 2.2. Initially, the data set is read by using a nearly standard reader. The cell attribute-based method consists of two main blocks that can be implemented in software as one or several filters. The first block generates the mesh topology from 1D connections of the lattice employed in DEM computations. It is the most time consuming procedure, which is performed only once. Usually, cells are generated before visualizing the results of the first time step. Sometimes the mesh topology can be imported from the non-standard DEM computations employing the mesh for the specific computational purposes (Vadluga *et al.* 2009). The second block calculates the cell attributes from the broken lattice connections. The method simply calculates how many 1D connections of the particular cell are broken and assigns this value to the cell attribute. Finally, the calculated cell attributes are mapped to the predefined colours and rendered on screen by using the available mapper and the renderer, respectively.

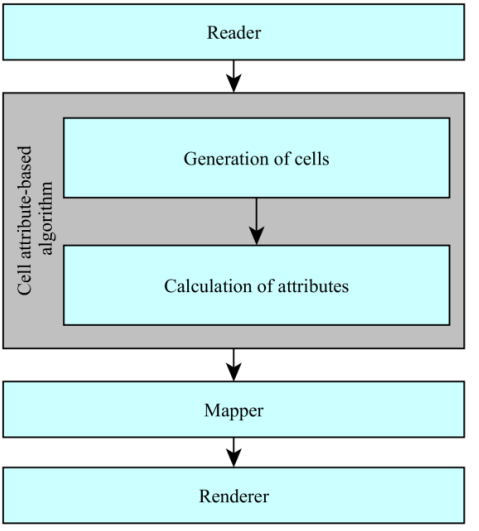


Fig. 2.2. Cell attribute-based visualization method

Figure 2.3 presents the method designed for generating the three–dimensional mesh topology. During the preparation of data structures the lists of the connected neighbours are assembled for all lattice points (particles). Then a loop, running through all points, is started. In 3D, two types of cells are considered to cover wider range of computational lattices. Thus, the generated 3D topology consists from pyramids with triangular or quadrilateral bases, which are generated in separate blocks of the method. In order to speed up computations, the point triplets and the point quadruplets are generated for identifying all possible triangular or quadrilateral bases, respectively.

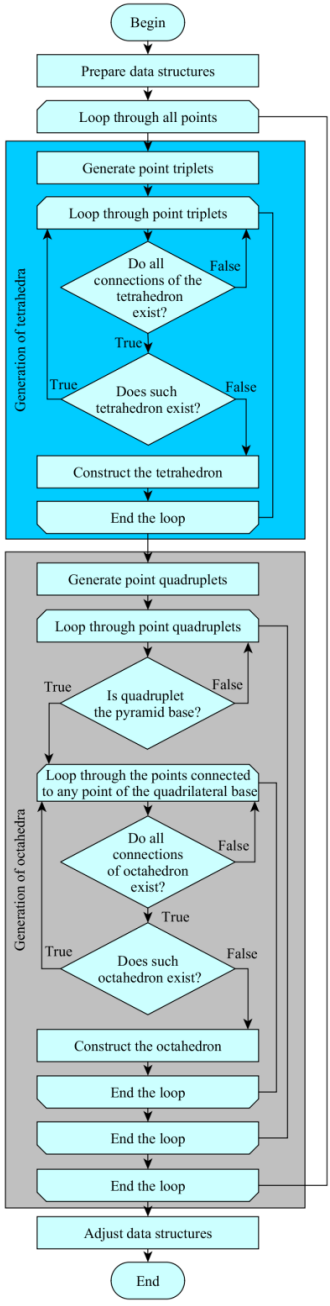


Fig. 2.3. Generating the mesh topology in 3D

The following loop runs through the generated point triplets or point quadruplets connected to the processed point. The important condition checks the existence of the connections between the point triplet forming the base of the pyramid. In the case of a pyramid with a quadrilateral base, this condition is more complex, because it is necessary to check the absence of the diagonals of the quadrilateral base. Moreover, this case is significantly more sophisticated, because resulting octahedron with eight faces can be divided to two pyramids in different ways. Thus, the additional loop, running through the points connected to any point of quadrilateral base, is performed to find the whole octahedron. The following condition checks the existence of the necessary octahedron connections. Finally, in both cases, the main block creates the new cells if such tetrahedron or octahedron has not been created yet. The octahedron is immediately divided into two pyramids with the quadrilateral base, because they can be efficiently processed by any visualization software.

Let $N= \{q_1, q_2, q_3, \ldots, q_n\}$ be a finite set of points, represented by centres of particles, in a sub-domain Ω of the space R^3. The method generates a partition of the sub-domain Ω into the non-overlapping regions Ω_i, such that:

$$\Omega = \bigcup \Omega_i \tag{2.1}$$

where each Ω_i is the 4 node tetrahedron or the pyramid defined by 5 nodes of N. Initially, the generated tetrahedra satisfy the Delaunay conditions. Four non–coplanar points q_i, q_j, q_k and q_l form a Delaunay tetrahedron D if and only if there exists a location $x \in \Omega$, which is equally close to q_i, q_j, q_k and q_l and closer to q_i, q_j, q_k, q_l than to any other $p_m \in \boldsymbol{N}$. The location x is the centre of the sphere, which passes through the points q_i, q_j, q_k, q_l and which contains no other points $p_m \in \boldsymbol{N}$. However, after some period of time the particles move, the lattice deforms and tetrahedra do not satisfy the Delaunay conditions in the highly fractured regions.

The calculation of the cell attributes from the broken lattice connections is illustrated in Figure 2.4. The input array ($S(i), i=1, \ldots, M$) indicates if the lattice connection between two neighbouring particles is broken. M is the total number of the lattice connections. The unity value of the array element means that the connection is broken. The method calculates how many 1D connections of the particular cell e are broken by using a simple formula:

$$a_e = \sum_{j=1}^{k} S(I(j)) \tag{2.2}$$

where a_e is the attribute of a cell Ω_e from formula (2.1). k is the number of cell connections. I is the array of global connection indexes of the cell e. In Figure 2.4,

the first loop runs through all cells of the newly generated mesh, while the second loop runs through all connections (edges) of the considered cell. The following block implements the trivial sum of the formula (2.2).

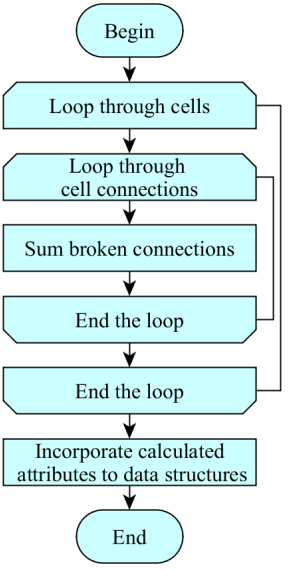


Fig. 2.4. Computation of cell attributes

The calculated value is assigned to the array of the cell attributes. Finally, the array of the calculated attributes is incorporated in data structures. At the end of the visualization pipeline (Figure 2.2) the calculated cell attributes are mapped to colours by using the predefined lookup table.

2.1.2. Cell Cut-based Method

The cell cut-based surface extraction method resulted from the development of the cell attribute-based method for crack visualization. The cell attribute-based method does not define crack geometry, therefore, its functionality was extended in the cell cut-based method. Moreover, the cell attribute-based method generates global decomposition for whole solution domain, while the cell cut-based method makes local decomposition and use the effective augmentation strategy.

Figure 2.5 illustrates the application of the cell cut-based surface extraction method in 2D. Lines between centers of particles are connections of the computational lattice. Black lines represent the unbroken lattice connections. Thin red lines represent the broken lattice connections, while red tubes visualize cracks. A simple cell cut-based method generates surfaces according to the information

about the way in which cracks cut cells of the local decomposition assembled from the lattice connections. In 2D, the method simply connects middle points of the broken connections.

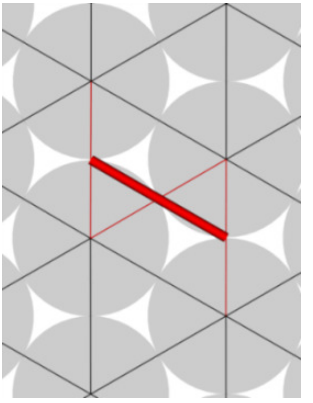


Fig. 2.5. Illustration of cell cut-based method

The simplified schema of the cell cut-based surface extraction method is presented in Figure 2.6. During the preparation of data structures, the lists of the connected neighbours were assembled for all lattice nodes. The following module generated a local space decomposition from the centres of the particles and the one-dimensional lattice connections between the particles employed in the lattice-based DEM computations.

A schema of the local decompositions method is presented in Figure 2.7. Local decompositions were generated only in the fractured regions, which were identified by marking the neighbourhood of the broken connections. In fact, these nodes are the centres of the particles. Another loop runs through the marked nodes to generate cells in the fractured regions. The developed method includes a very important condition, which checks if the current node has been already processed in the process of visualizing the preceding time steps. It saves computational resources by employing the already available cells and augmenting the local decomposition according only to the connections, which were broken during the last time step.

Several types of higher dimensionality cells might be considered to generate a suitable space decomposition based on the 1D connections of the processed lattice. Therefore, a loop running through the considered cell types should be also required. In our case, the generated 3D topology consists of the pyramids with triangular or quadrilateral bases, which are treated as different cell types and generated separately. To speed up computations, the node sets (triplets or quadruplets) were generated for identifying all possible triangular or quadrilateral

bases. In the 2D case, the triangles were considered and the pairs of nodes were generated.

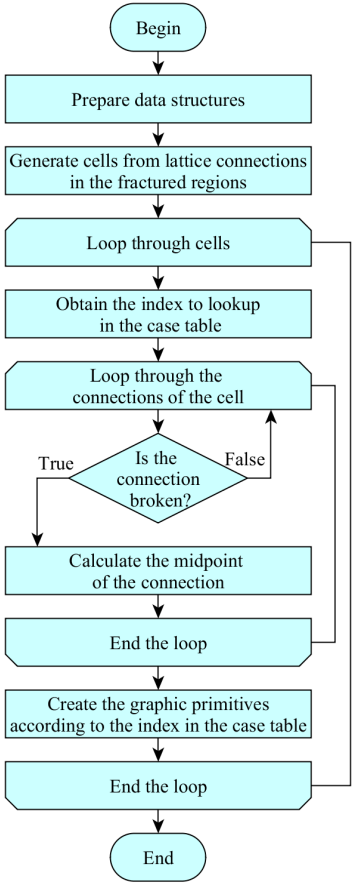


Fig. 2.6. A simplified schema of the cell cut-based method

New cells were assembled from the lattice connections in a loop, running through the generated node sets, which were connected to the node, considered in the main loop through the marked nodes of the lattice. A complex condition checks the presence of all connections in the assembled cell. The connections between the nodes of a set, forming the base of the pyramid, as well as the connections between the base of the pyramid and the considered node, were checked. In the case of a pyramid with a quadrilateral base, this condition also checks the absence of the diagonals of the quadrilateral base. Another condition checks if the current cell has not been generated yet. If all conditions are satisfied,

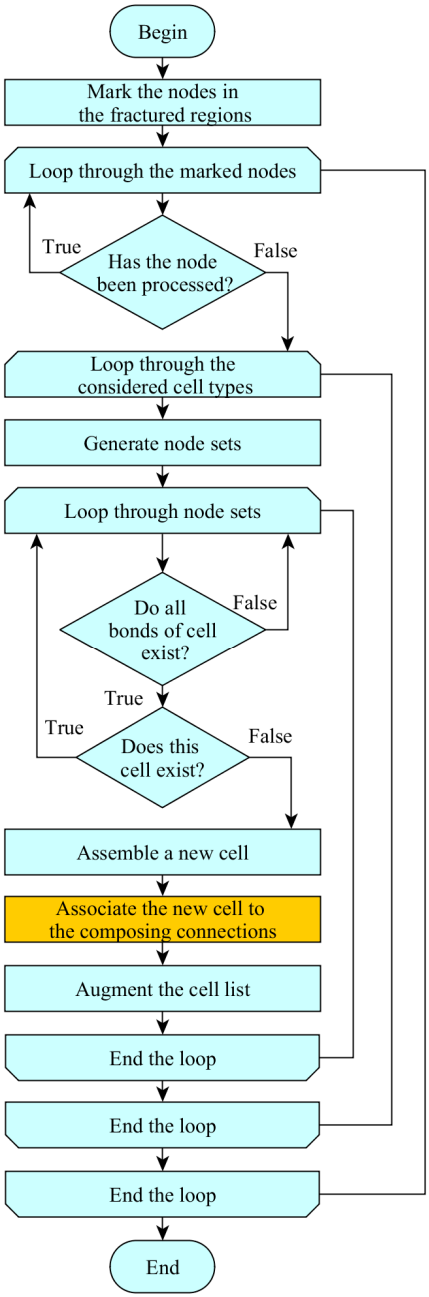


Fig. 2.7. Generating local decompositions from the lattice connections

the cell is assembled from the relevant connections. The assembled cell should be associated with the lattice connections that compose it in the case of using the geometric cell centre-based visualization method. However, the surface extraction based on the cell cut neither needs this association, nor executes. The following module fills the data structures and augments the cell list, appending a new cell to the list of the previously generated cells. It is worth noting that some cells of the list had already been generated by visualizing the preceding time steps and do not need to be generated at this stage. Finally, the loops through the generated node sets, the considered cell types and the marked nodes were completed. The resulting local decompositions cover the fractured regions and can be effectively used for extracting the crack surfaces.

The main loop (Figure 2.6) runs through all cells of the resulting decomposition. The method determined the graphics primitives required to represent the part of the crack surface passing through the cell according to the number and order of the broken connections. Initially, all possible topological states of the surface cutting the cell were explored and stored in a case table during the preparation of data structures. In the loop running through the cells, the index for looking up the relevant topological state of the processed cell in the case table was obtained. Another loop runs through all connections of the processed cell for obtaining the midpoints of the broken connections. The included condition checked if the processed connection was broken and determined if the coordinates of the midpoint had to be calculated. Finally, the graphics primitives were created according to the index of the state table. The vertices of the generated primitives were placed at the calculated midpoints of the broken connections. The main loop running through all cells of the local decomposition was completed, when the crack surfaces were described by graphics primitives in all cut cells.

2.2. Voronoi-based Method

The Voronoi-based method plots the crack on the extended contact surfaces obtained from local decompositions. During DEM computations, the defects are identified between the pairs of the neighbouring particles on the lattice connections. The broken lattice connections should be directly mapped onto the generated faces. The consistency of the generated faces of the Voronoi decomposition and the relevant lattice connections should be verified in the case of the highly deformed lattice. The standard methods for generating the global Voronoi diagrams are hardly applicable to visualization of cracks, because they use the coordinates of points, but do not take into account the lattice topology.

Figure 2.8 illustrates the application of the local Voronoi-based decomposition in 2D. Lines between centers of particles are connections of the computational lattice. Black lines represent the unbroken lattice connections, while thin red lines represent the broken lattice connections. Thick tubes represent the local Voronoi decomposition. Red tubes relevant to broken connections visualize cracks, while blue tubes show other edges of the local decomposition.

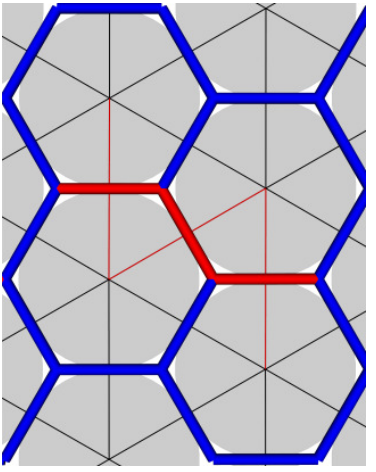


Fig. 2.8. Illustration of Voronoi-based method

A general schema of the method developed for generating the local Voronoi decompositions is presented in Figure 2.9. The method can be divided to five stages: preparation of data structures and kd-tree (K. Zhou *et al.* 2008), identification of fractured regions that are necessary to cover by local Voronoi decompositions, generation of Voronoi cells, validation of generated cells and mapping of the attributes of the lattice connections onto the relevant edges of Voronoi cells. Data structures for loops, running through the particle neighbours, which are joined to the considered particle by the lattice connections, are prepared at the beginning. The next module constructs the kd-tree (K. Zhou *et al.* 2008), which is used for visual model validation at other stages of the method. A loop, running through all lattice connections, starts the identification of fractured regions. The following condition checks if the current connection is broken. The cells of the local Voronoi decomposition are generated only around the end nodes of the broken connections, therefore, the next loop runs through two end nodes of the current connection. The next condition checks, if the current node has not been processed yet, because the Voronoi cell can be already generated around this node processing another broken connection attached to it. The following module generates the Voronoi cell around the particle centre

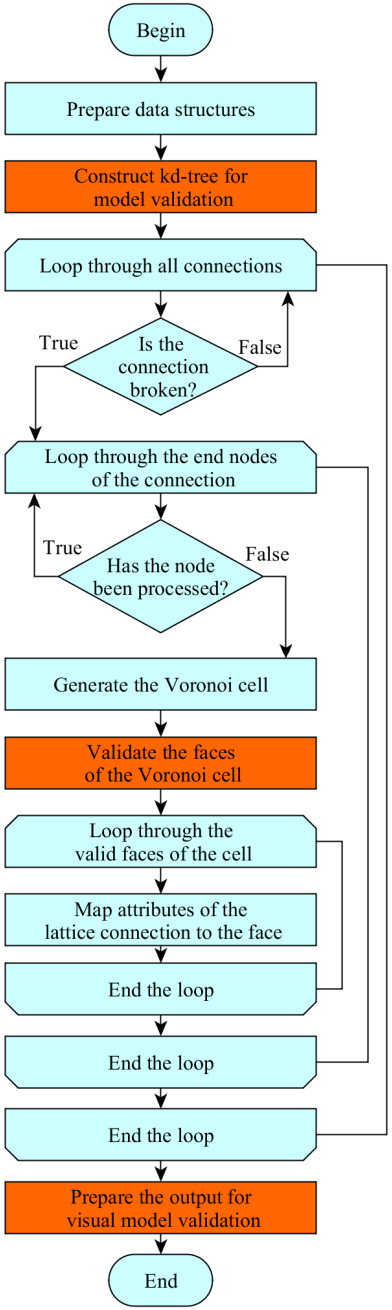


Fig. 2.9. A general schema of the local Voronoi decomposition-based method

represented by the processed node of the current connection. Each Voronoi cell is created around the node according to the available data on the node neighbours. The implemented method (Rycroft 2009) is based on the clipping of the whole domain by perpendicular planes drawn through the midpoints between the considered node and its neighbours.

All faces of the generated Voronoi cell are validated by the orange module, which will be described in the following text. The loop, running through the valid faces of the generated Voronoi cell, finishes processing of the generated contact surfaces. The following module maps the attributes of the lattice connections onto the relevant suitable faces of the generated Voronoi cell. In the 2D case, the attributes of the lattice connections are mapped onto the relevant edges of the cell. When the Voronoi cells are created around both end nodes of all broken connections, the main computations are finished. The last module prepares the output for visual model validation. Three orange modules (Figure 2.9) implement the model validation.

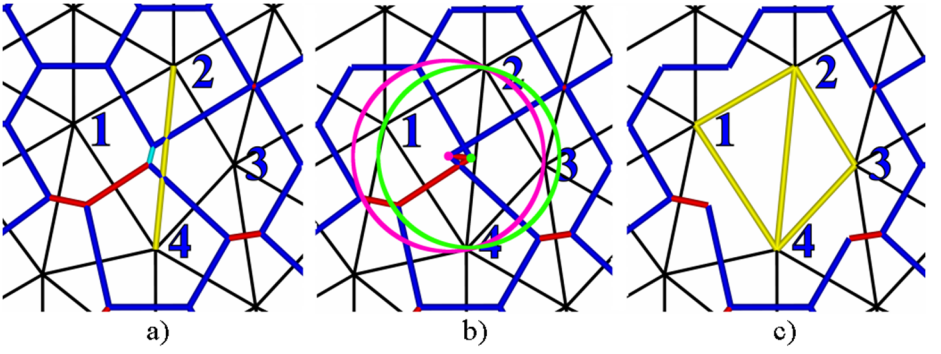


Fig. 2.10. An illustration of the inconsistencies between the lattice connections and the faces of the local Voronoi decomposition in the region of the deformed lattice: a) the global Voronoi decomposition, b) the local Voronoi decomposition, c) the visual model validation of local Voronoi decomposition

Visualization of crack surfaces is a challenging problem, while the application of the Voronoi decomposition-based methods is hardly possible in the regions of the highly deformed lattice. It is worth mentioning that the most of computational models also have limitations in such complex cases. In the lattice-based DEM computations, the lattice topology does not change in time, while the particles can significantly change their positions. The particles have more freedom to move and to deform the lattice in the regions containing a lot of broken connections. Thus, after a certain period of time, the lattice topology might become inconsistent with respect to the cells of a standard Voronoi diagram. Figure 2.10a illustrates the case, when the stationary lattice topology became inconsistent with a changing global Voronoi diagram. Blue tubes represent the

edges of the generated decompositions, while red tubes visualize cracks relevant to the broken lattice connections. The broken yellow connection of the stationary lattice joins the centres of the particles 2 and 4, while the method for generating the standard Voronoi diagram takes into account the fact that the distance between the centres of the particles 1 and 3 is shorter and does not generate the edge perpendicular to the broken connection. The available cyan edge represents the contact surface between the particles 1 and 3. The observed inconsistencies do not allow the direct mapping of the attributes of the lattice connections onto the relevant edges of Voronoi cells. Therefore, the global Voronoi diagrams cannot be applied to crack visualization in the regions of the highly deformed lattice. Figure 2.10b shows that, in such regions, the local Voronoi decomposition generated according to the stationary lattice topology contains the overlapped cells and introduces the numerical error. Figure 2.10b also includes an explanatory schema (two circles with centre points coloured in green and magenta) of visual model validation, which will be explained later. Figure 2.10c demonstrates how the visual model validation helps to indicate the regions of the highly deformed lattice by plotting marked connections as yellow tubes. The model validation also prevents the appearance of the overlapped cells.

The performed validation is based on the well-known property of the Voronoi diagrams described as follows: for each vertex q of the Voronoi diagram, there exists a unique empty sphere centred on q, which passes through at least four nodes and is the largest empty sphere centred on q (Aurenhammer 1991; Klein *et al.* 2009). Figure 2.11 presents the method for validation of each generated Voronoi cell. A loop, running through all the vertices of the considered Voronoi cell, starts the validation. The sphere is defined by the centre located in the considered vertex and the radius, which is equal to the distance from the considered vertex to the node representing the centre of the Voronoi cell. Figure 2.10b shows validation of two vertices of different Voronoi cells by using two illustrative circles. The green circle is defined by the centre located in the considered vertex (green point) and the radius equal to the distance from the vertex to the node 1 in the centre of the Voronoi cell. The magenta circle is defined by the centre located in the considered vertex (magenta point) and the radius, which is equal to the distance from the vertex to the node 3 in the centre of the Voronoi cell. The query to the initially prepared kd-tree structure provides the number of nodes belonging to the solid sphere. The considered vertex is marked as suitable for the Voronoi decomposition if none of the nodes of the lattice is located inside the solid sphere. In other cases, the processed vertex does not satisfy the required conditions of the Voronoi diagram. In Figure 2.10b, the lattice nodes 1, 2 and 4 are located on the green circle, but the node 3 violates the required condition, because it is located in the interior of the circle. The lattice node 1 is located in the interior of the magenta circle. Thus, both verified vertices are marked as

unsuitable for the Voronoi decomposition. Usually, this can be observed in the regions of the highly deformed lattice, where the lattice connection cannot be directly mapped onto the Voronoi face. The end of the loop, running through all vertices of the considered Voronoi cell, finishes the consistency check.

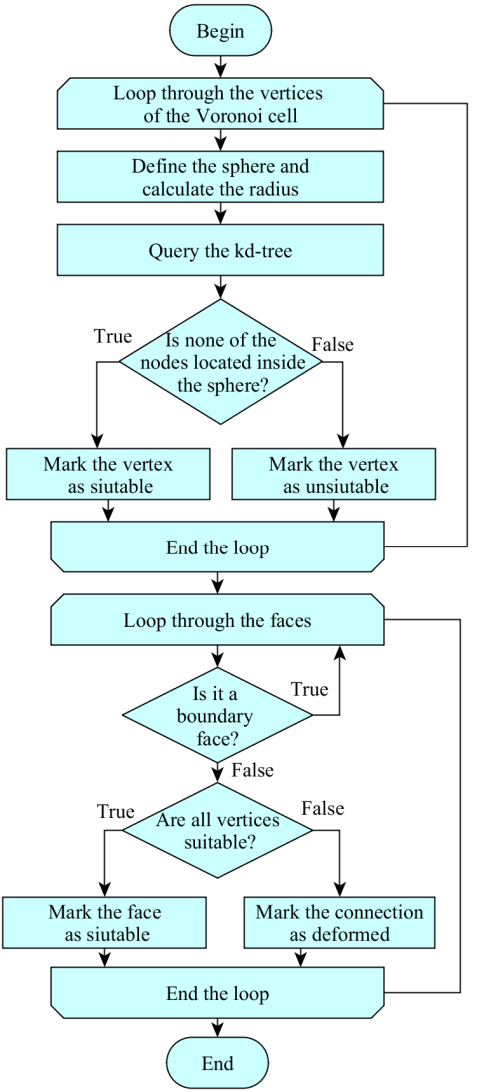


Fig. 2.11. Validation of the faces of the local Voronoi decomposition

The following loop, running through all faces of the considered Voronoi cell, is used to mark the suitable faces of the Voronoi cell and the appropriate connections of the lattice. If the considered face is a boundary face, marking is

not performed. The next condition checks all the vertices of the considered face. If all vertices have been marked as suitable, the entire face is marked as suitable. Otherwise, the connection, corresponding to the considered face, would be marked as belonging to the region of the highly deformed lattice, which could not be accurately visualized by using the Voronoi cells. The end of the loop, running through all faces of the considered Voronoi cell, finishes the marking, procedure. In Figure 2.10c, the marked connections are plotted as yellow tubes, which illustrates how the visual model validation helps to indicate the regions of the highly deformed lattice. It is worth noting that all validation modules can be removed from the pipeline to save computational time in the case of small geometric deformations in the lattice. It is worth mentioning that in the areas of the propagating cracks, the developed local Voronoi-based method generates exactly the same decomposition as the Voronoi method (Rycroft 2009) until the lattice topology is consistent with the cells of a standard Voronoi diagram.

2.3. Cell Centre-based Method

The cell centre-based method for extracting the crack surfaces was developed as an alternative to the Voronoi-based method to extend crack visualization to the regions of a highly deformed lattice.

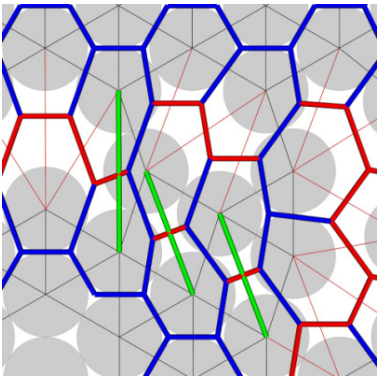


Fig. 2.12. Illustration of cell centre-based method

Figure 2.12 shows the application of cell centre-based method in the 2D region of deformed lattice. Thin red lines represent the broken lattice connections, while red tubes visualize cracks. Black lines represent the unbroken lattice connections. The initial space decomposition, which is identical to the local decomposition generated by the cell-cut method, is assembled from the lattice connections. The cell centre-based method also

generates an additional space decomposition represented by blue tubes in order to increase the accuracy of the surface extraction and to visualize the fractured regions covered by the highly deformed lattice, which is indicated by green connections in Figure 2.12.

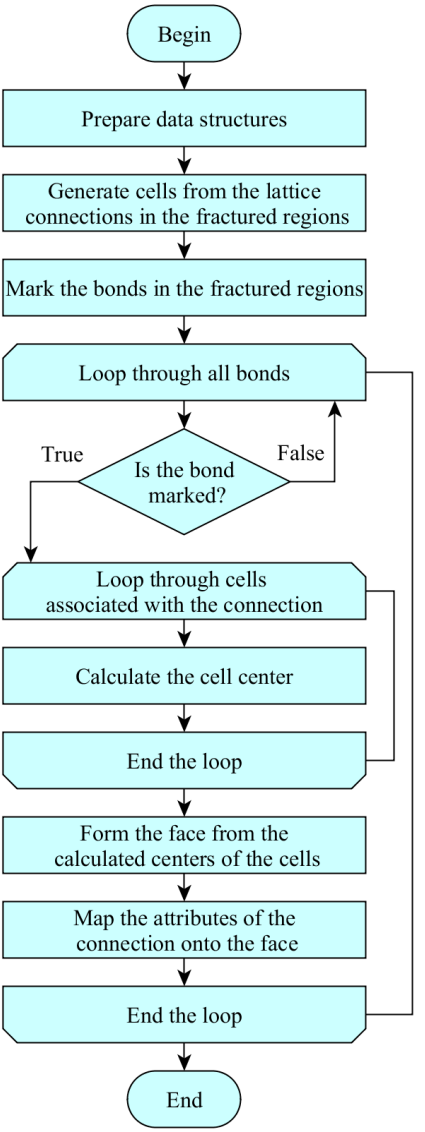


Fig. 2.13. The simplified schema of the cell centre-based method

The simplified schema of the developed method is presented in Figure 2.13. The method starts from the module, which prepares the data structures for the subsequent computations. Another module generates the initial local decomposition from the lattice connections covering the fractured regions (Figure 2.7). The next module marks the broken connections and their neighbouring connections to identify the topology of the fractured regions.

In Figure 2.13, the loop, running through all lattice connections, starts the generation of the geometry for the new local decomposition based on the geometrical cell centres. A simple condition checks if the current connection has been marked because a new decomposition is required only in the fractured regions. The second loop runs through all cells, including the processed connection. It is worth mentioning that the specific information about the cells associated with the processed connection, is necessary only for the cell centre–based method. The required list of indices of the cells is prepared in the orange module (Figure 2.7). Another module calculates the geometric centre C of the considered cell by using the formula for the centroid (Krantz *et al.* 2006) of a finite set of k vertices:

$$C = \frac{x_1 + x_2 + ... + x_k}{k} \tag{2.3}$$

where $x_1; x_2;...; x_k$, are the coordinates of vertices in R^3. The end of the loop running through all cells, including the marked connection, completes the generation of vertices for the new face crossing the processed connection. Another module forms the new face from the calculated centres of the cells associated with the processed connection. Finally, the attribute values of the marked connection were mapped onto the new face. The generated faces are represented by graphics primitives. The generated faces are not joined to the cells around the centres of the particles because this topology requires some additional storage. At the end of the pipeline, the surfaces of the propagating cracks are visualized by colouring the generated faces according to the values of the mapped attributes.

2.4. Distributed Visualization Software

The proposed methods for visualization of cracks were implemented in the grid visualization e-service VizLitG and the distributed visualization software VisPartDEM that were developed for remote visualization of discrete particle systems.

2.4.1. Grid Visualization e-service VizLitG

The client-server architecture of the visualization e-service is based on the widely recognized web standards. Java EE platform (Jendrock *et al.* 2010) provides the tools for remote client-server communication. VizLitG is implemented in GlassFish (Goncalves 2009) application server by using convenient tools such as web service designer and web service tester that allow programmers to significantly reduce the development efforts. The Message Authentication over SSL mechanism of GlassFish is employed for security purposes. The server authenticates a client of the visualization service by using a basic authentication scheme including the name/password pair of credentials. Moreover, HTTPS protocol using Secure Sockets Layer (SSL) version 2.0 is used for message integrity and confidentiality. The e-service architecture and communication schema are illustrated in Figure 2.14.

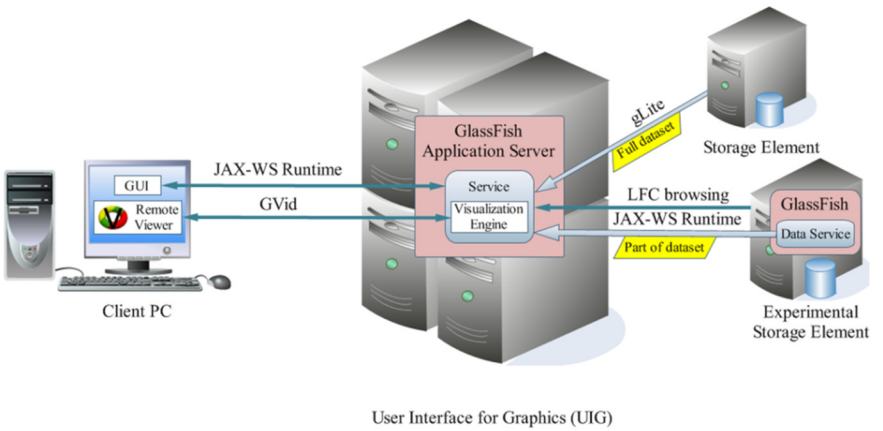


Fig. 2.14. The architecture and communication of grid visualization e-service VizLitG

A visualization pipeline is distributed between the client and the server. A visualization engine responsible for data filtering, mapping and rendering is implemented as a part of the main service, which runs on a special User Interface computer named UIG (User Interface for Graphics). Thus, a natural access to grid resources is available for the service. User authentication and the full data set transfer from SE are performed by traditional means available in gLite/EMI distribution. The client implemented as Java application handles user interaction. It consists of a GUI and a Remote Viewer. In order to simplify the installation of the client for less experienced users, the client software is downloaded by Java

Web Start (Marinilli 2001). The client does not depend on hardware and operating system, therefore, it can run on any PC.

A visualization network is interactively assembled from the provided modules by using GUI. Tabular design of GUI fields simply illustrates the dataflow. The resulting pipelines are described by XML language (Evjen *et al.* 2007) according to the developed schema. The main elements describe the selected data, the included filters, the considered mappers and the specified parameters of the renderer. Valid XML documents are automatically generated and transferred from the client to the server by JAX-WS (Kalen 2009) Runtime based on high-level SOAP protocol (Englander 2002). Data filtering, mapping and rendering are always performed on the server, therefore, a sufficiently powerful server is required in the case of a large number of simultaneously working users. However, the latency of gLite/EMI resource broker (EGEE 2009; Kačeniauskas *et al.* 2010) does not influence a visualization process.

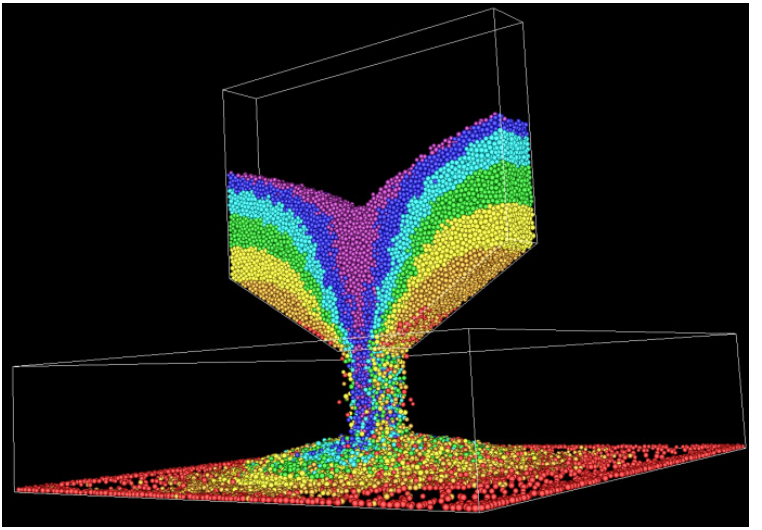


Fig. 2.15. Visualization of hopper discharge by using GLSL shaders

The visualization engine of the VizLitG is based on VTK toolkit (Schroeder *et al.* 2006). VTK objects are enwrapped by the Java programming language in order to build the service running on UIG. High flexibility of e-service is achieved retaining sufficient efficiency of the object-oriented library built by C++. GLSL shaders (Figure 2.15) supported by VTK are implemented in VizLitG to improve the performance of visualization and to exploit the increasing parallelism provided by graphics processors. Vertex and fragment shaders (Biddiscombe *et al.* 2008) are employed for fast rendering of heterogeneous particles on GPU. The developed Remote Viewer transfers the final images from visualization engine to

the client and offers high remote interactivity for VizLitG users provided by VTK widgets and GVid (Polak *et al.* 2008).

The developed GUI allows interactive data set selection. A simple tabular design of GUI is programmed by using Java Swing (Zukowski 2005). To process HDF5 (Folk *et al.* 2011) files automatically, data sets are stored in a predefined structure, allowing the software to interpret the structure and contents of a file without any outside information. HDF5 groups and data sets are automatically processed, considering the values of HDF5 attributes. Time–dependent and time-independent data are processed differently. Data sets varying in time are grouped and stored according to the time step number. GUI separates geometry and topology from the attributes such as scalars or vectors in order to emphasize their different nature.

A special data service is developed to provide users with fast access to interactively selected parts of data sets located in the experimental SE. The experimental SE has the same hardware requirements as any SE of grid infrastructure built by gLite. The standard software packages provided within glite/EMI like Disk Pool Manager (DPM) Storage Element for disk and Disk Pool Manager Storage Element for MySQL are installed on the experimental SE. The only additional software running on the experimental SE is GlassFish application server, which provides users of VizLitG with the developed Data Service (Figure 2.14) enabling a partial data transfer.

The Data Service is developed by using high-level means of GlassFish. JAX-WS Runtime transfers metadata and the selected parts of the visualized data sets between the service running on UIG and the Data Service running on the experimental SE (Figure 2.14). The MIME multipart mechanism for binary SOAP attachments (Englander 2002) was employed for sending binary data between the developed services. LFC (Logical File Catalog) system (Kaci *et al.* 2010) is employed for browsing through the content of SE and identifying the file. GUI provides interactive environment to VizLitG users and covers unnecessary details of distributed services running on a heterogeneous grid infrastructure.

Thus, remote instrumentation of the developed e-service provides users with the flexible access to the remote data files located in SE. The whole data sets located in SE can be transferred by traditional LFC (or LCG) means available in glite/EMI distribution. Alternatively, files can be transferred by using GridFTP tools available in the Java CoG Kit jGlobus module (Laszewski *et al.* 2001). Moreover, transfer of interactively selected parts of data sets located in experimental SE rather than the whole data files can save a significant amount of visualization time and overcome difficulties related with a limited network bandwidth.

2.4.2. Distributed Visualization Software VisPartDEM

The architecture of VisPartDEM (Figure 2.16) is designed for grid infrastructure build by glite/EMI middleware, later adapted for Rocks clusters and graphics workstation. Client software including GUI and Remote Viewer is downloaded by using Java Web Start technology (Marinilli 2001). VisPartDEM client implemented as Java application connects to any user interface (UI) by means of JSCH library (JCraft 2014). Traditional glite/EMI commands for user authentication and authorization, job submission and monitoring are enwrapped by Java programming language. Considered visualization pipelines, JDL files and shell scripts for running visualization engine are generated automatically in order to submit job to grid. Finally, parallel visualization engine of VisPartDEM runs on working nodes of any computing element while the compressed video stream is efficiently transferred from the zero MPI node through the network and displayed on the client by Remote Viewer.

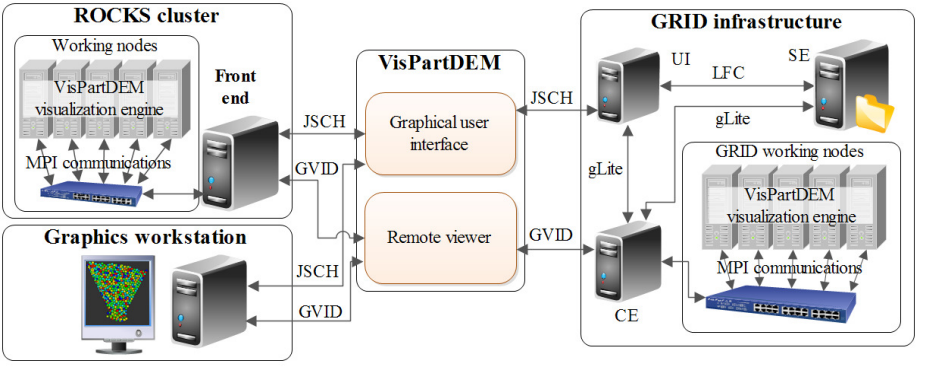


Fig. 2.16. The architecture of VisPartDEM

Distributed visualization engine of VisPartDEM is based on VTK (Schroeder *et al.* 2006). The graphics model in VTK is at a higher level of abstraction than rendering libraries like OpenGL. This means that it is much easier to create useful graphics and visualization applications. VTK applications are platform independent, which is very attractive for heterogeneous grid architectures. Data parallel model of VTK is employed for visualization of large discrete particle systems. A large data set is partitioned into many independent subsets that are processed in parallel (Figure 2.17). Copies of the same modules run on each CPU and visualize independent subsets of data. Data parallel modules are usually followed by a data parallel merge module that gathers the independently computed results and merges them into a final result on a single processor. A sort-last parallel rendering class inputs a z-buffer and image pair from each process by using MPI communication and outputs a single composite result image to MPI process zero.

GUI of VisPartDEM is designed to cover from user unnecessary details and complexities of heterogeneous grid infrastructure. The GUI allows interactive browsing of storage element (SE) content and automatic data management. In order to process HDF5 files (Folk *et al.* 2011) automatically, data sets are stored in predefined structure allowing the software to interpret the structure and contents of a file without any outside information. HDF5 groups and data sets are automatically processed considering values of HDF5 attributes. XML interface for remote data is developed to provide grid users with the interactive data set selection. The interface program reads attributes from HDF5 file and writes metadata to XML document, which also has predefined structure. Usually, large HDF5 file containing data is dislocated in remote storage element, while small XML file containing metadata on the data structure can be stored in any convenient location (client PC, UI or even SE). Finally, XML file is processed by GUI to display metadata in the corresponding fields and to provide users with the ability to select data interactively.

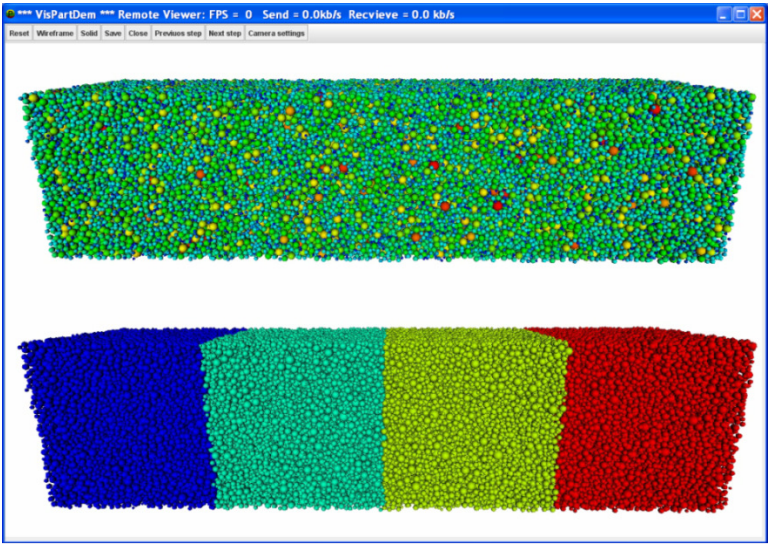


Fig. 2.17. Parallel visualization of compaction process by using 4 processes

The Remote Viewer of VisPartDEM employs GVid software (Polak *et al.* 2008) as video streaming module to provide grid users with the high interactivity level. Interactive events and video stream generated by VTK are transferred between the server and the client by using GVid. The most important VTK GLUT classes, vtkGlutOpenGLRenderWindow and vtkGlutRenderWindowInteractor, are renewed to support later VTK versions. As a result distributed visualization engine VisPartDEM runs in parallel on working nodes while the video stream is transferred through the network from 0-th MPI node and displayed on the client.

Thus, remote grid user has full interactivity provided by the Remote Viewer based on GVid software and VTK widgets.

2.5. Implementation of Crack Visualization Methods

The crack visualization methods are implemented in the distributed visualization prototypes VizLitG and VisPartDEM developed for interactive investigation of discrete particle systems. Figures 2.18, 2.19 and 2.20 show general visualization pipelines of the cell cut-based method, Voronoi-based method and cell centre–based method, respectively.

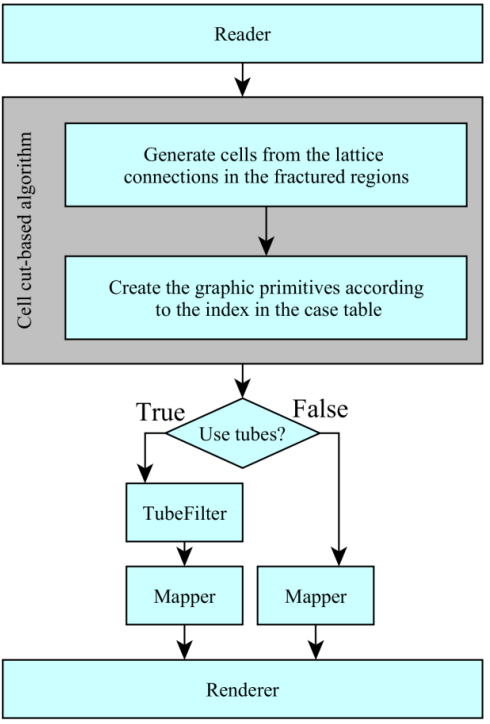


Fig. 2.18. Visualization pipeline of cell cut-based method

Initially, all visualization pipelines read data sets from HDF5 files by using the developed vtkHDF5Reader. Then, the specific modules of methods are executed. The cell cut-based method consists of two main modules (Figure 2.18). The first module generates cells of local decomposition from lattice connections in the fractured regions. It is time consuming procedure, which use the effective augmentation strategy. The second module produces graphics primitives

according to the index in the case table. In the case of Voronoi-based method (Figure 2.19), local Voronoi decompositions are generated according to the input array of attributes, referred to as the connection state, which identifies the broken lattice connections. Then attributes of the connections are directly mapped onto the relevant faces of the generated decomposition.

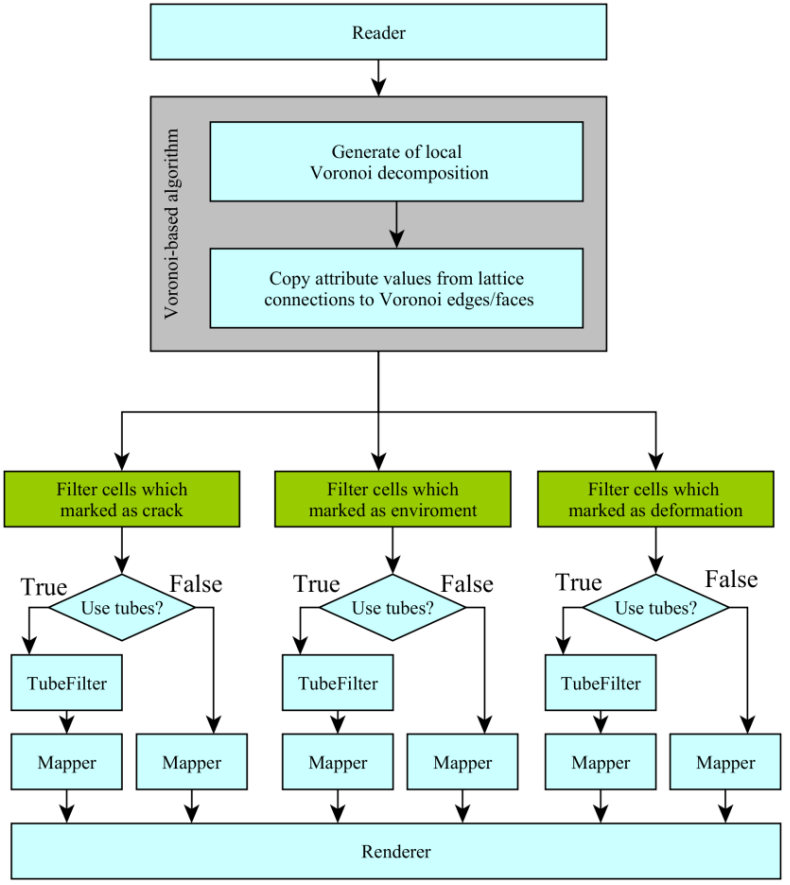


Fig. 2.19. Visualization pipeline of Voronoi-based method

The cell centre-based method consists of three main modules (Figure 2.20). The first module generates cells of the first local decomposition, which is identical to the decomposition produced by the cell cut-based method. It is worth mentioning that topology of this decomposition does not change in time, therefore, the effective augmentation strategy can be applied. The second module compute the cell centres according to formula (2.3) and forms faces of the second decomposition, which topology is similar to the Voronoi decomposition. The third

module copies attribute values from connections to corresponding faces of the cell centre-based decomposition.

In the last stage of all methods, the generated faces or marked connections are coloured and rendered. The users are able to select preferable branches of the visualization pipeline by using GUI. The output of cell cut-based method is the simplest, therefore, the pipeline contains only one branch (Figure 2.18). Various graphical representations of crack surfaces may be considered by users. The edges of resulting cracks can be thickened employing vtkTubeFilter. Finally, the graphical primitives are coloured by using the vtkPolyDataMapper and rendered by using vtkRenderer.

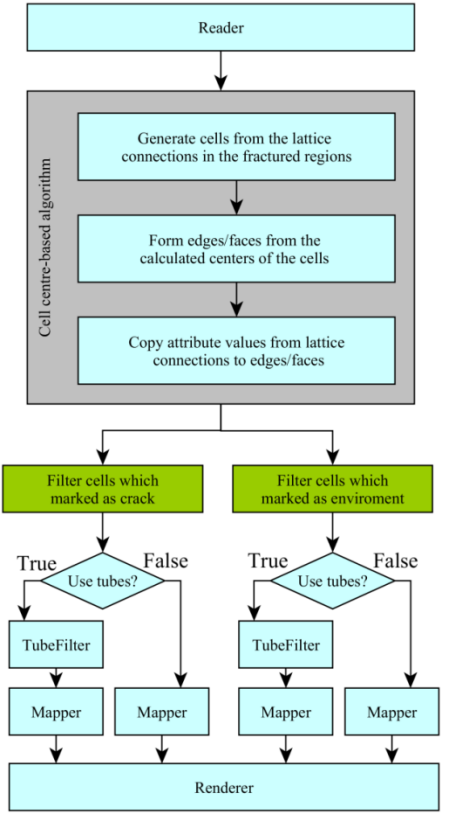


Fig. 2.20. Visualization pipeline of cell centre-based method

The output of Voronoi-based method is more complex, because it can contain the local decomposition and the results of visual model validation (Figure 2.19). The first branch of the pipeline plots crack surfaces. The second branch shows the local space decomposition of fractured regions. The last branch of the pipeline is

responsible for graphical output of visual model validation. It shows the connections that are inconsistent with the faces of Voronoi decomposition in the regions of the highly deformed lattice. All branches of the visualization pipeline can be combined and executed simultaneously. The lattice connections, the edges of the faces and 1D cracks can be represented by tubes employing vtkTubeFilter. The condition checks if tube representation is selected by users. Finally, the values of the considered attributes are mapped onto colours by the vtkPolyDataMapper, while graphical primitives are rendered by using vtkRenderer.

The output of the cell centre-based method can be visualized by two branches of pipeline (Figure 2.20). The first branch of the pipeline plots crack surfaces, while the second branch shows the local space decomposition of fractured regions. The identical branches of the pipeline are encountered in the description of the pipeline of the Voronoi-based method.

2.6. Conclusions of Chapter 2

1. The simple cell attribute-based method for visualization of cracks maps broken lattice connections to newly generated cells. The spatial crack representation is visualized as the scalar cell attribute, but exact geometry of crack surface remains undefined.
2. The functionality of the cell attribute-based method is extended in the cell cut-based method, which extracts crack surfaces as graphics primitives. The cell cut-based method generates surfaces according to the information about the way in which cracks cut cells of the space decomposition.
3. Faces of local Voronoi decompositions are used as extended contact surfaces of neighbouring particles. Attributes from lattice connections are directly mapped to relevant faces of Voronoi cells without any interpolation.
4. The Voronoi-based method cannot be applied in the highly deformed regions, because of inconsistency between the lattice connections and the faces of the Voronoi diagram. The procedure of visual model validation is developed to identify the regions of a highly deformed lattice.
5. The cell center-based method, positioning the vertices of the generated local decomposition in the geometric centers of cells, is developed to extend crack visualization to the regions of a highly deformed lattice, where the Voronoi–based method cannot be applied.
6. Interactively assembled visualization network of VizLitG is automatically described by XML language according to the developed schema. Valid XML documents are automatically generated on a client and transferred to the server by JAX-WS. The XML documents govern assembling of visualization pipelines from VTK filters in visualization engine.

7. Grid visualization e-service VizLitG provides GLSL shaders for fast rendering of discrete particle systems on GPU.
8. Grid visualization e-service VizLitG provides interactive data selection, partial data set transfer and flexible access of the remote data files located in the grid storage elements.
9. Distributed architecture of VisPartDEM is designed for interactive visualization on different infrastructures: grid, Rocks clusters and graphics workstations. VisPartDEM offers high remote interactivity for grid users provided by VTK widgets and GVid software.
10. Data parallel model for visualization of large discrete particle systems is implemented in VisPartDEM software. Parallel visualization engine runs on working nodes of any computing element while the compressed video stream is efficiently transferred from the zero MPI node through the network and

3

Experimental Research on the Proposed Visualization Methods and the Developed Software

In the third Chapter, the results of experimental research are presented. Datasets resulting from DEM simulations are described. The performance of distributed visualization software and partial data set transfer is investigated. The performance and quantitative comparrison of crack surface visualization by using the proposed methods is presented. The accuracy of the developed visualization methods was evaluated by computing the total depth of cuts made in particles by the extracted crack surfaces.

Experimental results presented this Chapter are published in (Kačeniauskas and Pacevič 2011), (Pacevič *et al.* 2013), (Pacevič, Kačeniauskas, *et al.* 2015), (Kačeniauskas *et al.* 2012), (Kačeniauskas *et al.* 2013), (Pacevič and Kačeniauskas 2015).

3.1. Description of Visualized Data Sets

In this subsection, descriptions of data sets for visualization benchmarks are provided. Polydispersed particle systems are visualized by using VizLitG and

VisPartDEM to measure performance of partial data set transfer and parallel visualization, respectively. Polydispersed particle systems are considered as a pilot application for visualization due to a large number of particles that are employed modelling actual industrial applications. The developed methods for visualization of cracks, propagating in monodispersed particle systems, are tested visualizing data sets of the uniaxial tension problem.

Visualization of polydispersed particle systems (Figure 3.1) is considered for performance analysis of VizLitG. The investigated data sets result from the solution of the tri-axial compaction problem (Kačianauskas *et al.* 2010) by the discrete element method. The compacted granular material is represented as an assembly of spherical non-cohesive visco-elastic frictional particles. The initial state of the particulate material is generated, randomly distributing particles in a three-dimensional computational domain. It imitates a representative macroscopic region element and presents a box in the form of the cube. The compaction is performed by the motion of the rigid walls of the box and is controlled in time by a constant rate. The numerical solution of tri-axial compaction helps to evaluate the unknown material properties. This is a significant problem often encountered in the area of material sciences.

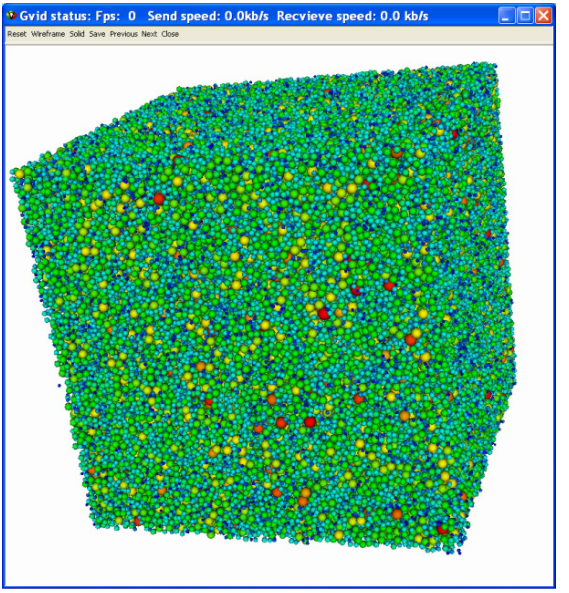


Fig. 3.1. Remote visualization of 202215 particles coloured depending on the radii

The considered benchmark is based on the glyph generation, because particles, velocities and accelerations are often represented by glyphs that can be

coloured depending on the investigated scalar values or oriented depending on the examined vector values. Alternatively, particles can be rendered by GLSL shaders to exploit the increasing parallelism of GPU.

Table 3.1. Data sets of polydispersed particle systems

Particles	50880	97036	147408	202215
Data set size, MB	1.753	3.337	5.067	6.948
HDF5 size, MB	11.663	22.227	33.756	46.300
Rendered cells	4884480	9315456	14151168	19412640
Rendered points	2544000	4851800	7370400	10110750
Glyph size, MB	232	443	673	923

Data sets of polydispersed particle systems are described in Table 3.1. The first row shows the number of particles. The second row presents the size of VTK object, which encapsulates the examined data set. Meaningful data is composed of the positions of particles and their radii. This data is interactively selected and transferred from the experimental SE employing the developed data service. The complete numerical results include a lot of values of primary and derived variables that are written in HDF5 (Folk *et al.* 2011) files. Spherical particles are represented by spherical glyphs generated by using default VTK input parameters. The last three rows show the information on the generated geometry of glyphs that can be rendered on screen.

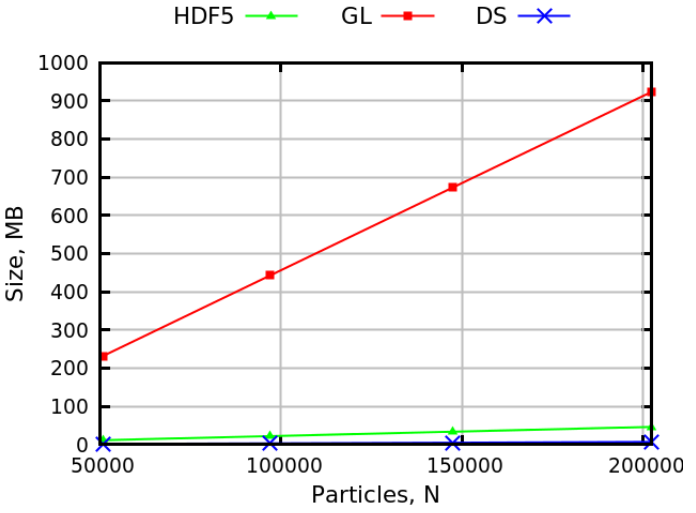


Fig. 3.2. The sizes of the considered data sets

Figure 3.2 illustrates the sizes of the considered data sets. The size of HDF5 files (the curve HDF5) is larger than the size of the investigated data sets (the curve DS), because the files include other data, which were not visualized in the performed benchmark tests. The most important fact is that the size of the rendered polygon mesh (the curve GL) is significantly larger than the size of the investigated data set (the curve DS). Usually, the second-order difference can be observed, for example, 6.948 MB and 923 MB. It makes the described benchmark very specific and inconvenient for some visualization tools. On the contrary, particle shaders can perform this benchmark very efficiently. Rather than representing particles as glyphs, they are rendered directly to viewport as spherical primitives, supplying only a position, radius and any scalar attribute mapped to colour.

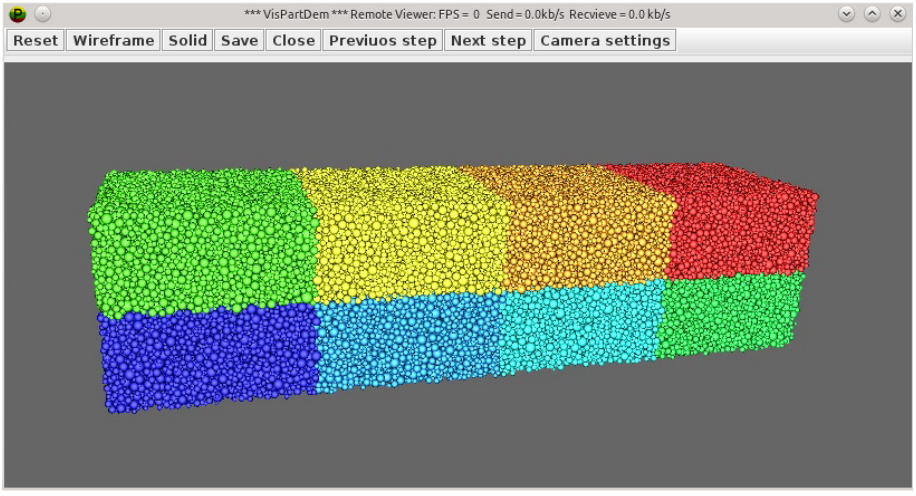


Fig. 3.3. Parallel computations of tri-axial compaction problem: particles coloured according to process ID

Parallel visualization of polydispersed particle systems (Figure 3.3) of the tri-axial compaction problem (Kačianauskas *et al.* 2010) is considered for performance analysis of VisPartDEM. The visualization benchmark is based on the glyph generation, because particles, computed velocities and obtained forces are often represented by glyphs that can be coloured by investigated scalar values or oriented by the examined vectors. The examined data sets contain 100036, 150119 and 200194 heterogeneous particles. Meaningful data are composed from the positions of particles and their radius, therefore, the real sizes of the visualized data are quite small (3.13 MB in case of 100036 particles). Numerical results include a lot of values of variables that are written in HDF5 files, therefore, the size of complete HDF5 file is equal to 21.39 MB in case of 100036 particles. The

total size of partitioned result files is up to 21.63 MB, which is close to the size of the single file. The size of the single file containing the results of 150119 polydispersed particle system is equal to 32.09 MB, while the total size of 16 partitions is equal to 32.37 MB. In case of the polydispersed particle system containing 200194 heterogeneous particles, the size of the single file is equal to 42.79 MB, while the total size of partitioned files is up to 43.07 MB. Particles are represented by generated spherical glyphs. The size of the object, which encapsulates data of generated glyphs, is equal to 326 MB in case of 100036 polydispersed particles. It makes the described benchmark very specific, because a generated geometry is larger than the initial data set.

The data sets obtained in solving the uniaxial tension problem were visualized to validate the effectiveness of the developed methods for visualization of crack propagating in monodispersed particle systems. The considered DEM model (Rojek *et al.* 2011) is able to describe the elastic solid problem exhibiting non-uniform distribution of fracture force values. To illustrate the extraction of crack surfaces fracture phenomena in the rectangular plate were considered to be a two-dimensional benchmark. Two plate boundaries were clamped by connecting them to rigid walls, while other boundaries were free. The external excitation was implemented via the motion of the clamped boundaries defined by the constant velocity (u = 0.05 m/s) in order to simulate tension in the specimen with the dimensions of 0.376×0.107 m. The simulated system consisted of 4679 particles with the radius equal to 1.58 mm. The lattice was formed of 13722 springs, which were considered to be the connections between the neighbouring particles. The initial defect, specified by 3 broken connections, is marked by using a red colour (Figure 3.4).

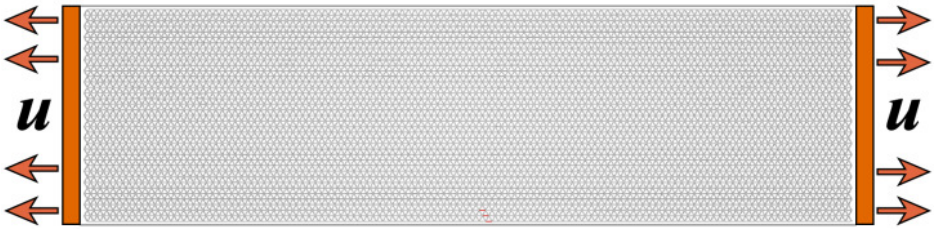


Fig. 3.4. Geometry and lattice of the 2D benchmark

A three-dimensional benchmark is based on the data sets obtained by simulating crack propagation in a rectangular cuboid. Two domain boundaries are also clamped, while their constant velocity is equal to 0.025 m/s. The dimensions of the cubical specimen are equal to 0.211×0.1×0.1 m. The simulated system consists of 46875 particles with the radius of 2 mm. The lattice is formed of 267674 springs. Three data sets, A, B and C, with the initial defects of various sizes, were investigated. The initial defects of different rectangular shapes defined

by 125, 245 and 121 broken connections are located in the middle of the specimen bottom. The geometries of the benchmark problem relevant to data set A, B, C are shown in Figures 3.5a, 3.5b and 3.5c, respectively. The considered benchmark problems are often investigated in order to understand fracture phenomena.

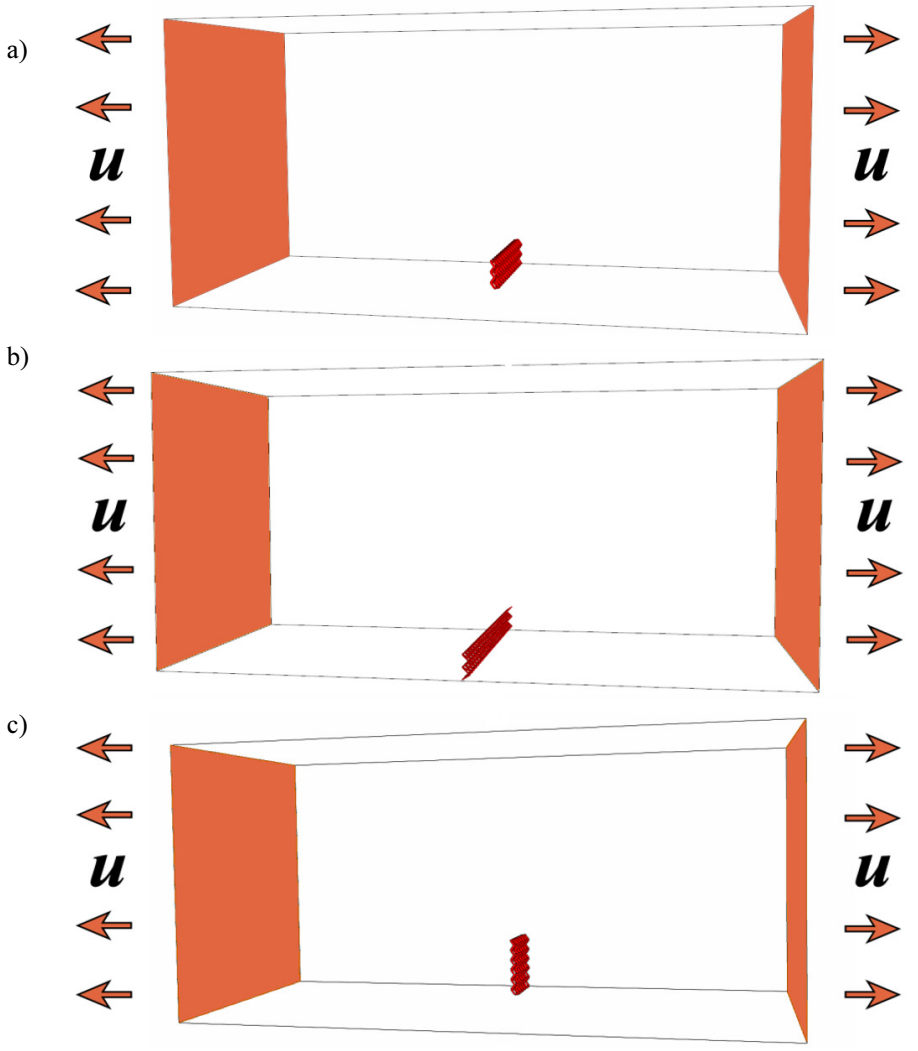
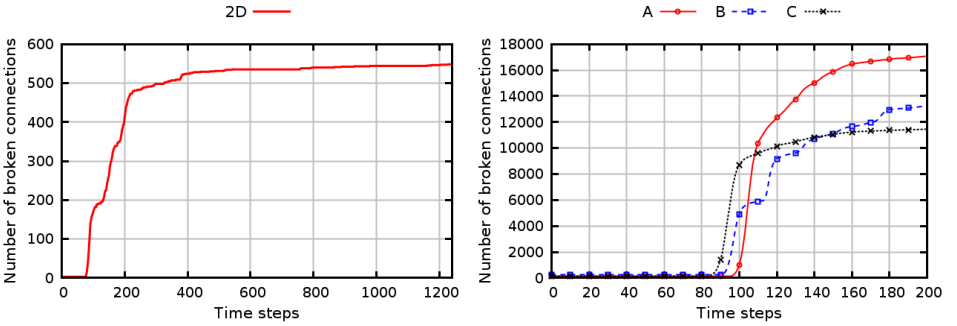


Fig. 3.5. The geometry of the 3D benchmark problem and the shape of the initial defects: a) data set A, b) data set B, c) data set C

The data set produced by DEM computations includes the positions, velocities, accelerations, radii and material properties of particles, as well as the lattice connections between the neighbouring particles and their attributes, such as the connection state, the force and the fracture force limit. The complete numerical results can include a number of values of other primary and derived variables. The size of the visualized 2D data set, storing the results of 1284 selected time steps, is equal to 1.4 Gb. The size of all three-dimensional data sets, storing the results of 200 selected time steps in HDF5 files, is about 2.5 Gb.



a) b)

Fig. 3.6. Growth of the number of broken connections: a) 2D, b) 3D

The time consumed by visualization methods depends on the number of the broken lattice connections. Figure 3.6 demonstrates the variation of the number of broken connections in time. Rapid changes in the number of broken connections can be observed between time steps 80 and 215 in the 2D case, while the same phenomenon reveals between time steps 85 and 95 in the case of all 3D data sets. In the last time step of the 3D data sets, the number of the broken connections made 6.4%, 4.9% and 4.3% of all lattice connections and 4.0% in case of the 2D data set.

3.2. Performance of Distributed Visualization Software and Partial Data Set Transfer

3.2.1. Performance of VizLitG Including Partial Data Set Transfer

A series of benchmark tests examining computational performance of the VizLitG e-service were performed on an ordinary personal computer (PC) and HP xw4600

workstation. Both computers served as UIG installed at Vilnius Gediminas Technical University (VGTU). Hardware characteristics of the PC are listed below: Intel® Core2Quad Q6600 2.40 GHz CPU (2 x 4 MB L2 cache and bus frequency equal 1067 MHz), 4 GB DDR2 RAM, 320 GB HDD (SATA II Extensions and 16 MB cache), Nvidia GeForce 9600GT (512 MB) GPU. Hardware characteristics of HP xw4600 are listed below: Intel® Core2Quad Q9450 2.66 GHz CPU (12 MB L2 cache and bus frequency equal 1333 MHz), 8 GB DDR2 RAM, 2 x 250GB HDD (SATA 3 GB/s NCQ 7200), Nvidia Quadro FX4600 (768 MB) GPU.

Hardware characteristics of the experimental SE maintained at VGTU are listed below: AMD Athlon X2 Dual Core BE-2300 1.9 GHz CPU, 2 GB DDR2 800 RAM, 3 x 500GB SATA II Extensions, Software Raid0, 1 Gbps LAN. In geographically distributed environment, the data transfer tests were performed employing SEs maintained by other LitGrid partners. Hardware characteristics of the experimental SE maintained at Kaunas Technical University are listed below: Intel®Xeon 5130 2.00 GHz CPU, 2 GB DDR2 800 RAM, 3 x 200GB SATA II Extensions, Software Raid5, 1 Gbps LAN. Hardware characteristics of the experimental SE maintained at Klaipėda University are listed below: Intel®Xeon 5110 1.6 GHz CPU, 1 GB DDR2 800 RAM, 3 x 80GB SATA II Extensions, Software Raid5, 1 Gbps LAN.

The performance analysis was concentrated on the server side of the visualization e-service, because hardware characteristics of a client PC can be very different and hardly predictable. However, three different client computers were employed to perform benchmark in geographically distributed grid environment. The PC named C-1 and connected to the network in the same building as UIG was used as a client computer to illustrate the usual conditions at a research laboratory. The hardware of C-1 was identical to hardware of the PC described above. The laptop C-2 (AMD Turion 64 X2 Mobile Technology TL-60, 2 GHz CPU, 2 GB DDR II RAM 667 MHz, ATI Mobility Radeon HD 2600 GPU with 512 MB) and other personal computer C-3 (AMD Sempron Dual Core Processor 2300, 1800 MHz CPU, 1 GB DDR II RAM 667 MHz, ATI Radeon X1200 GPU with 128 MB) with low end hardware were employed to simulate less favourable conditions like home environments. The laptop C-2 was connected to the network in other district of Vilnius, while C-3 was located in the town Alytus.

Table 3.2. Network load between user interface for graphics and storage elements.

Network load	UIG → VGTU	UIG → KTU	UIG → KU
Round-trip time, ms (Min/Average/Max)	0.07/0.08/0.09	1.00/2.06/4.39	4.15/4.65/7.88
Network bandwidth, Mbit/s	480	184	167

Performing the initial benchmark, heterogeneous particles were represented by coloured spherical glyphs. The attention was focused on the performance of the vtkGlyph3D filter and rendering of the resulting polygon meshes. Mapping was not considered because it took a very short time equal approximately to 0.0001 s. The interactive session handled by GVid software took about 1 s. The detailed investigation of interactive performance is provided at the end of this section. Parameter transferring between the client (GUI) and the server (visualization engine) was fast enough because of the small data size. Performing benchmarks, e-service received about 26.3 kB and sent about 6.9 kB. Communication lasted less than one-tenth of a second. The data set transfer between SE and visualization server performed by JAX-WS Runtime was considered. In Table 3.2 second column shows network load, round-trip time and network bandwidth messured by using Iperf (Tirumala *et al.* 2006), between UIG and experimental SE (VGTU) connected to the same switch, during the benchmark. The average system load of the SE was 10% during the benchmark. The benchmark tests were repeated up to ten times and the averaged values were examined.

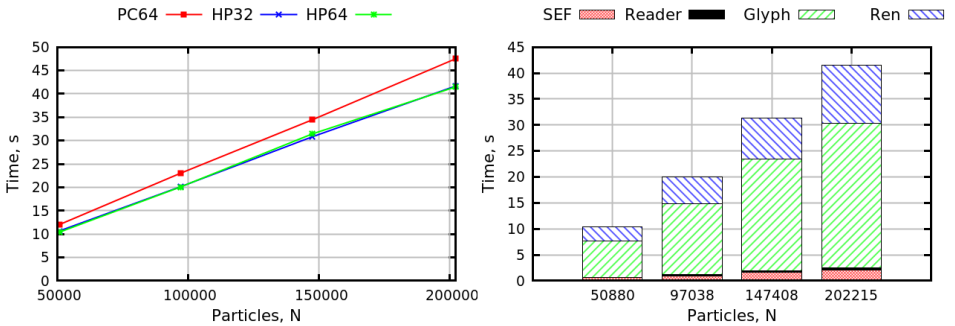


Fig. 3.7. Visualization benchmark based on glyphs: a) total visualization time, b) contribution of different visualization procedures to the total benchmark time

Figure 3.7 shows the total visualization time consumed by the VizLitG performing the benchmark based on glyphs and full data set transfer. In Figure 3.7a, the curve PC64 represents the total visualization time measured on the considered PC running Scientific Linux 5.5 64 bit operating system. The curves HP32 and HP64 show visualization time obtained on HP xw4600 running Scientific Linux 5.5 32 bit and 64 bit, respectively. Almost identical performance was observed on the HP workstation, running 32 bit and 64 bit operating systems. The measured performance difference between PC (the curve PC64) and HP (the curves HP32 and HP64) did not exceed 12.5% of the visualization time. In Figure 3.7b, the chart compares the contribution of full data transfer from storage element

(SEF), file reading from HDD (Reader), glyphs generation (Glyph) and rendering (Ren) to the total visualization time measured on HP workstation. It is obvious that glyph generation and rendering consumed the amount of time exceeding 93% of the total visualization time. The data set transfer became less important, because it consumed only about 5% of the time. It can be explained by the fact that the size of the generated glyphs was significantly larger than the size of the transferred data.

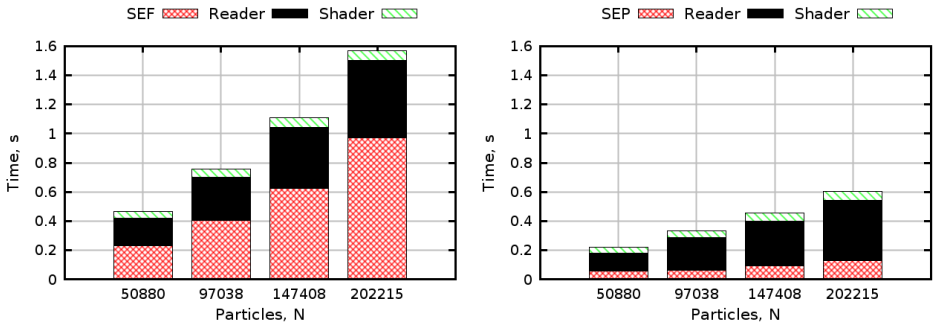


Fig. 3.8. Contribution of visualization procedures to the total visualization time of benchmark performed by using shaders: a) PC transferring the full data set, b) HP transferring part of the data set

However, visualization of spherical particles can be performed by GLSL shaders on GPU, which leads to the huge increase of performance. Figure 3.8 shows the visualization time measured performing the same benchmark by the implemented particle shader. The charts compare the contribution of full data transfer from storage element (SEF), partial data set transfer from storage element (SEP), file reading (Reader) and particle shading (Shader) to the total visualization time. Figure 3.8a shows the contribution of different visualization procedures to the total benchmark time measured on PC transferring the full data set, while Figure 3.8b shows that measured on HP transferring only the selected part of the data set. The total visualization time was reduced by an order of magnitude, because the shader performed the work made by glyph generation and rendering. The measured shading time varied from 0.04 s to 0.07 s. The full data set transfer consumed from 50% to 62% of the total visualization time in spite of the fact that time values of insecure transfer were presented in Figure 3.8a. Moreover, the contribution of data transfer to the total visualization time grew while data set size increased. It is obvious that the reduction of time consumed by data set transfer becomes crucial for interactive visualization of the considered data sets. The employed partial data set transfer reduced the transfer time up to an order of magnitude. Moreover, partial data set transfer consumed less than 21% of the total visualization time measured on HP xw4600 (Figure 3.8b).

Figure 3.9 shows the time consumed by data transfer from SE. Different transfer protocols and software was employed to present the quantitative comparison for the investigated data sets. The curve GFTP represents GridFTP from Java CoG Kit jGlobus module. The curve LFC represents the means of Logical File Catalog included in glite/EMI distribution. The curve JWS shows the time consumed by JAX-WS Runtime for transferring the complete data sets over SSL. Finally, the curve PDT means partial data set transfer from the experimental SE provided by the developed Data Service (Figure 2.14). This secure data transfer was also performed by JAX-WS Runtime. The performance of JAX-WS Runtime was the best for the considered data sets, because they were not of a large size. Employing partial data set transfer, the communication time was reduced by up to 7.6 times and became almost negligible.

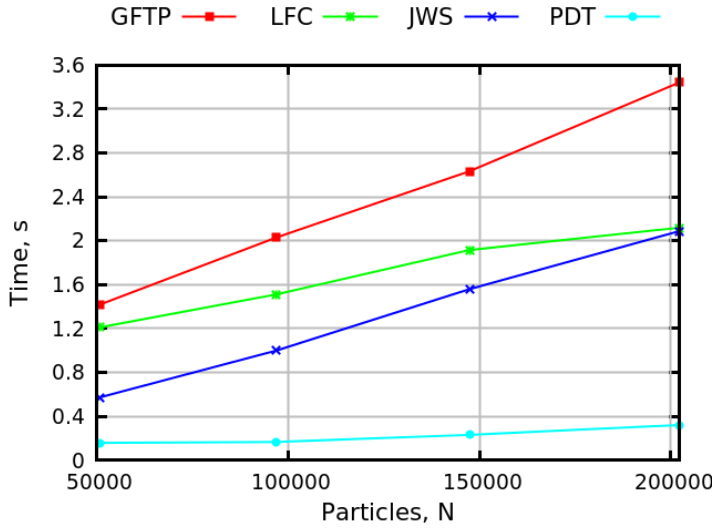


Fig. 3.9. Data set transfer from SE by using different software

Figure 3.10 illustrates how secure data transfer over SSL influenced the time consumed. The dashed curves, FnoS and FSSL, represent the time consumed by insecure full data set transfer and full data set transfer over SSL, respectively. Other curves, PnoS and PSSL, show the time consumed by insecure partial data set transfer and partial data set transfer over SSL, respectively. Insecure data transfer applied instead of the SSL mechanism reduced the time of partial data set transfer up to 40%. In the case of full data sets, the obtained percentage was even higher (about 42%), because a larger amount of data need to be encoded and transferred. However, insecure communication is not recommended for grid e-services, because the security of important data can be violated.

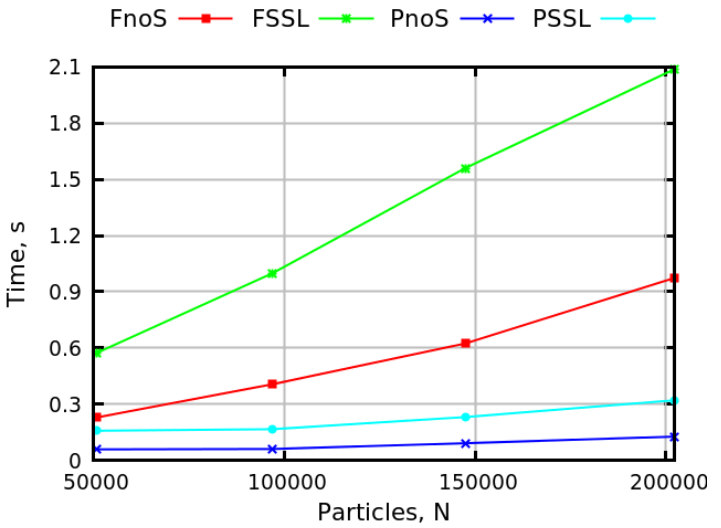


Fig. 3.10. Time consumed by secure and insecure data set transfer

Figure 3.11 illustrates the performance of the developed Data Service in the geographically distributed grid environment. SE components located in the towns Kaunas and Klaipėda were employed for measuring the time consumed by the data transfer performed by JAX-WS Runtime. Table 3.2 shows network load, between UIG at VGTU and SE at KTU in Kaunas, and SE at KU in Klaipėda, during the benchmark. The average system load of the investigated SE at KTU and KU was 30% and 35%, respectively. The curves, PD_V, PD_Ka and PD_Kl, represent the time consumed by the partial data set transfer from the SE located in Vilnius, SE located in Kaunas and SE located in Klaipėda, respectively. The dashed curves, FD_V, FD_Ka and FD_Kl represent the time consumed by the full data set transfer from the SE located in Vilnius, SE located in Kaunas and SE located in Klaipėda, respectively. As expected, data transfer from the SE located in Kaunas was faster than that from the SE in Klaipėda because of lower latency and higher network bandwidth. In the case of full data set transfer, the significant time increase was observed when the data was transferred between the distant locations. On the contrary, in transferring partial data sets, the time difference varied about 0.1 s, which was insignificant.

The performance of interactivity (Figure 3.12) was also investigated in the case of the geographically distributed grid. The Remote Viewer is based on the GVid software, which transports the efficiently compressed standard video stream to the remote output device and handles interactive events. Video stream is encoded by using XviD codec (Xvid 2009). The transfer of several encoded frames of different size (179.7 kB and 308.6 kB) was investigated. These typical frames result from interactive processing of the image shown in Figure 3.1**,** which

consists of 900 x 900 pixels. The image encoding (Encode), frame sending (Send) and receiving (Receive) as well as decoding (Decode) and displaying (Display) were considered in the performed benchmark. The image encoding and frame sending time is measured on the server UIG (HP xw4600), while receiving, decoding and displaying time was measured on three different clients. Interactive events were captured and transferred very quickly, therefore, their time consumption was not included.

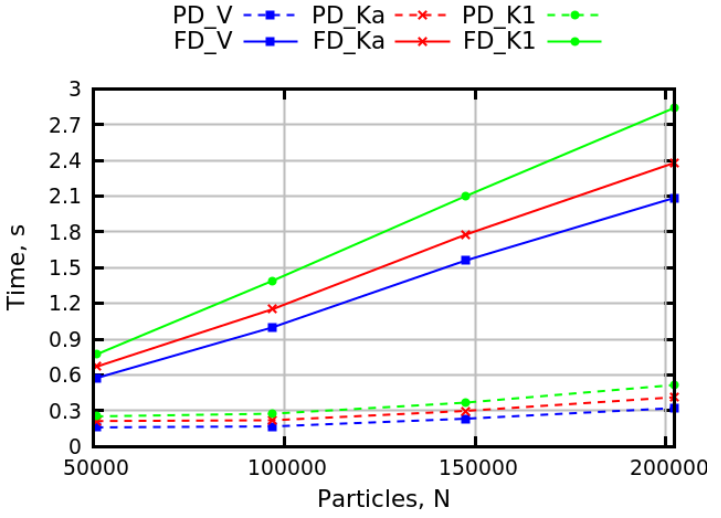


Fig. 3.11. Data set transfer from geographically dist ributed SE components to UIG

Table 3.3 shows network load, round-trip time and network bandwidth messured by using Iperf (Tirumala *et al.* 2006), between UIG and three different clients, located in the VGTU building (C1), located in another district of Vilnius (C2) and located in Alytus (C3). A low quality network was tested to simulate extreme cases representing a bottleneck for interactive visualization. The benchmark tests were repeated up to one hundred times and the averaged values were presented.

Table 3.3. Network load between user interface for graphics and clients

Network load	UIG → C1	UIG → C2	UIG → C3
Round-trip time, ms (Min/Average/Max)	0.17/0.19/0.21	1.89/2.82/44.77	3.12/18.93/1417.73
Network bandwidth, Mbit/s	933	177	2.1

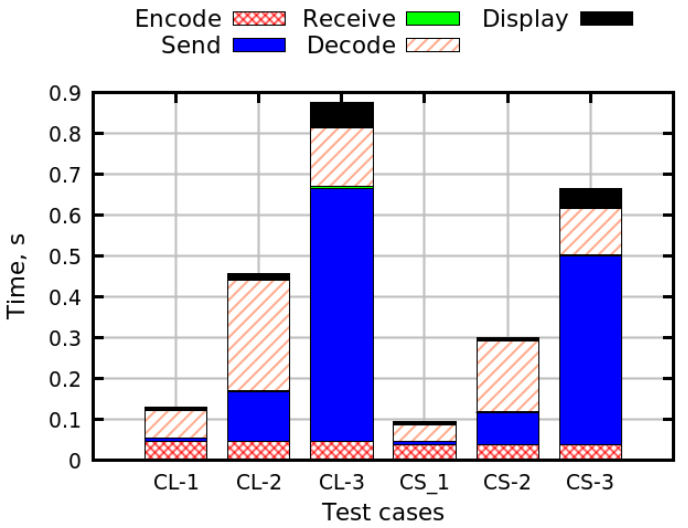


Fig. 3.12. Image coding and frame transfer for remote interactivity

Figure 3.12 shows the time consumed by GVid on three clients and the server UIG. The stacked columns, CL-1, CL-2 and CL-3, represent the time consumed by processing larger encoded frame (308.6 kB), which was transferred to the clients C-1, C-2 and C-3, respectively. The stacked columns, CS-1, CS-2 and CS-3, represent the time consumed by processing smaller encoded frame (179.7 kB), which was transferred to the clients C-1, C-2 and C-3, respectively. The encoding was performed on the server, therefore, it consumed almost the same amount of time for the defined frame. The time of encoding of different frames was slightly different. Different time values were measured transferring the frames to different clients, while the receiving time was negligibly small in all cases. Frame decoding strongly depended on the client hardware. The C-1 equipped by the Intel® Core2Quad Q6600 2.40 GHz CPU was significantly faster than other clients. Displaying of the decoded frame lasted about 0.01 s. However, in the case of the low end graphics cards like ATI Radeon X1200 installed on the client C-3, a longer time was observed. It is evident that frame sending strongly depends on the network connection and the frame size. It is not suitable for interactive purposes in the case of the low bandwidth and high latency networks like the connection between the UIG and C-3 located in Alytus. However, the GVid is well designed for a variable or low bandwidth, because of the efficient compression codec XviD and rate adaptation to the current network bandwidth. Thus, the frame rate was automatically adapted to a low bandwidth, and the transferred data was reduced.

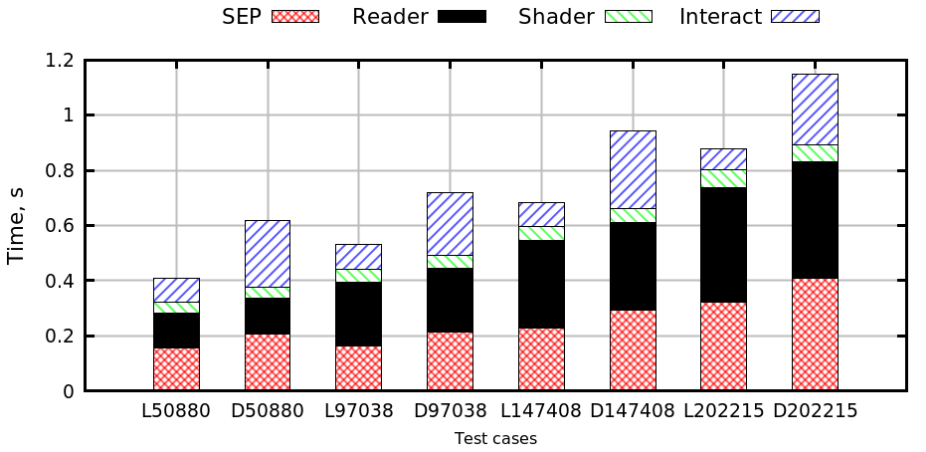


Fig. 3.13. Contribution of visualization procedures to the total visualization time

Figure 3.13 presents a comparison of the total visualization time consumed by the locally distributed e-service with that measured in geographically distributed grid environment. The stacked columns compare the contribution of secure partial data set transfer from the storage element (SEP), file reading (Reader), particle shading (Shader) and minimal interactive session including two calls of vtkRenderWindow (Interact) to the total visualization time. Test cases L50880, L97038, L147408 and L202215 represent visualization benchmark of different polydispersed particle systems performed on the locally distributed components of e-service. Thus, UIG, SE and the client C-1 were located in the VGTU building. Moreover, UIG and SE were connected to the same switch. Test cases D50880, D97038, D147408 and D202215 represent the same benchmark performed on geographically distributed grid components. The UIG (HP xw4600) was located at VGTU in Vilnius, while the SE was located in Kaunas. Moreover, the client C-2 was connected to the network in another district of Vilnius to simulate visualization in home environment. The employed hardware and network characteristics measured during the benchmark were provided above.

The geographically distributed components of the infrastructure increased the total visualization time from 24% to 34% (Figure 3.13). As expected, time values of particle shading were very close in all cases because shading was performed by the same GPU. The differences in time consumed by data reading were insignificant. The most significant time increase was observed during the interactive session, because of the low performance of the laptop hardware employed for frame decoding (Figure 3.12). The increase in time consumed by the partial data set transfer from the distant SE was not significant. In the case of the largest system of particles, the measured difference was equal to 0.091 s. It can be concluded that the developed Data Service, providing a secure partial data

set transfer from the SE, considerably reduced the size of the transferred data and demonstrated good performance in the geographically distributed grid environment.

3.2.2. Performance of VisPartDEM Including Parallel Visualization

The desktop-delivered visualization and grid computing might become the solutions providing sufficient performance by visualizing a relatively large data set with the help of relatively cheap hardware. A benchmark was run to illustrate the performance of the developed software for visualization of discrete particle systems.

VisPartDEM benchmark was performed on BalticGrid-II site ce2.grid.vgtu.lt collected from ordinary PCs equipped by GPUs. This glite/EMI CE maintained by VGTU was considered for benchmark, because it supported direct GPU rendering and it was based on the multi-core architecture. The CE consisted of 14 HP Compaq dc7900 personal computers (nodes) including Pentium(R) Dual-Core CPU E5300 2.60 GHz, 4 GB DDR2 RAM 800 MHz, 500 GB HDD. Each node is equipped by GPU (Nvidia GeForce 9600GT 512 MB 256 bit). Nodes are connected to 1 Gbps Ethernet LAN by 3Com Baseline Switch 2928-SFP Plus (24 auto sensing 10/100/1000Mbps Base–TX ports). Hardware characteristics of the storage element se.grid.vgtu.lt (SE-1) maintained by VGTU are listed below: AMD Athlon X2 Dual Core BE-2300 1.9 GHz CPU, 2 GB DDR2 800 RAM, 3 x 500 GB SATA II Extensions, Software Raid0 and 1 Gbps LAN. The SE-1 was connected to the same switch as the ce2.grid.vgtu.lt. In geographically distributed environment, the data transfer tests were performed employing the storage element se.bg.ktu.lt (SE-2) located at Kaunas Technical University (KTU). Hardware characteristics of the SE-2 are listed below: Intel®Xeon 5130 2.00 GHz CPU, 2 GB DDR2 800 RAM, 3 x 200 GB SATA II Extensions, Software Raid5, 1 Gbps LAN.

The tests were repeated up to ten times and the averaged values were investigated. The main attention was focused on the performance of the data transfer from SE, speed-up of visualization procedures and the interactive performance. Mapping was not considered, because it took a very short time equal approximately to 0.0001 s. In the performed benchmark, MPI communication for message passing between working nodes is employed for composition of the final image. Detailed investigation of MPI communication is not presented, because it lasts negligible time (less than 0.002 s). HDF5 files were transferred from the SE to WNs by using LFC means. Each process of VisPartDEM transferred its data

file independently, in asynchronous fashion. In case of the considered data sets asynchronous data transfer can be up to 11 times faster than the synchronous one.

Table 3.4. Network load between compute element to storage

Network load	CE → SE-1	CE → SE-2
Round-trip time, ms (Min/Average/Max)	0.161/0.229/2.218	1.00/2.06/4.39
Network bandwidth, Mbit/s	583	184

The asynchronous data transfer is investigated in geographically distributed grid. Each process independently transfers its data file from the SE to its WN. The longest transfer time is picked up from times consumed by all parallel processes. The network load, between CE to SE-1 at VGTU in Vilnius and CE to SE-2 at KTU in Kaunas, is described in Table 3.4. The average system load, during the geographically distributed benchmark, of the SE-1 and SE-2 was equal to 10% and 30%, respectively.

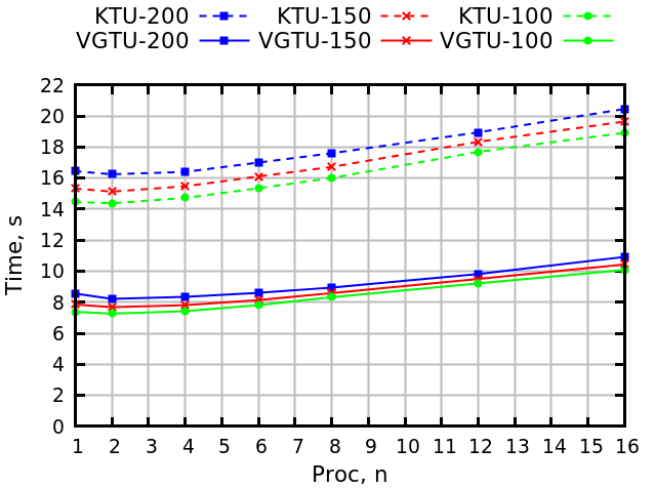


Fig. 3.14. Time consumed by the data transfer from SEs to WNs

Figure 3.14 shows the time consumed by the asynchronous data transfer. The curves VGTU-100, VGTU-150 and VGTU-200 represent time consumed by transferring data sets of 100036, 150119 and 200194 particles, respectively, from SE-1 to WNs of ce2.grid.vgtu.lt. The dotted curves KTU-100, KTU-150 and KTU-200 represent transferring data sets of 100036, 150119 and 200194 particles, respectively, from SE-2 located at KTU to CE located at VGTU. The asynchronous data transfer helps to reduce transferring time in case of very small number of processes. All parallel processes use the same network equipment and

the same hardware of the employed SE. Encountered bottlenecks do not allow attaining parallel speed-up in data transfer. The distant data transfer strongly depends on the network load, therefore, the curves representing data sets of different size are not so gradually distributed. As expected, data transfer from the distant SE-2 was slower than that from SE-1 located in the same building as CE.

Table 3.5. Network load between compute element and client

Network load	CE → C-1	CE → C-2
Round-trip time, ms (Min/Average/Max)	0.17/0.19/0.21	1.75/9.25/71.64
Network bandwidth, Mbit/s	933	1.95

The performance of interactivity was also investigated in the case of the geographically distributed grid. The Remote Viewer is based on the GVid software, which transports the efficiently compressed standard video stream to the remote output device and handles interactive events. Video stream is encoded by using XviD codec (Xvid 2009). The transfer of the encoded frame of 302.6 kB size was investigated, which consists of 1100 x 600 pixels. The image encoding (Encode), frame sending (Send) and receiving (Receive) as well as decoding (Decode) and displaying (Display) were considered in the performed benchmark. The image encoding and frame sending time is measured on CE in VGTU, while receiving, decoding and displaying time was measured on two different clients. Interactive events were captured and transferred very quickly, therefore, their time consumption was not included. Table 3.5 shows network load between CE to the client C-1 located in VGTU building and C-2 located in Alytus. A low quality network was tested to simulate extreme cases representing a bottleneck for interactive visualization. The benchmark tests were repeated up to one hundred times and the averaged values were presented.

Figure 3.15 shows the time consumed by GVid on two clients (the curves C-1 and C-2) and the CE. The encoding was performed on the server, therefore, it consumes almost the same amount of time for the defined frame. Different time values were measured transferring the frames to different clients, while the receiving time was negligibly small in all cases. Frame decoding strongly depended on the client hardware. The C-1 equipped by the Intel® Core2Quad Q6600 2.40 GHz CPU was significantly faster than the other client. Displaying of the decoded frame lasted about 0.01 s. However, in the case of the low end graphics cards like ATi Radeon X1200 installed on the client C-2, a longer time was observed. It is evident that frame sending strongly depends on the network connection and the frame size. It is not suitable for interactive purposes in the case of the low bandwidth and high latency networks like the connection between the CE and C-2 located in Alytus.

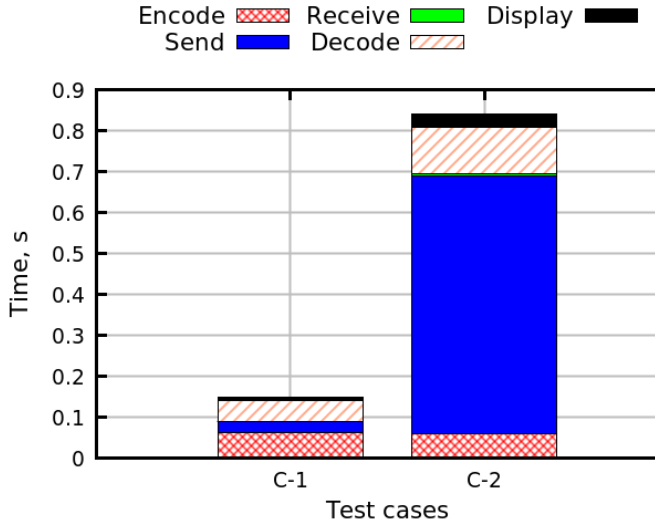


Fig. 3.15. Image coding and frame transfer for remote interactivity

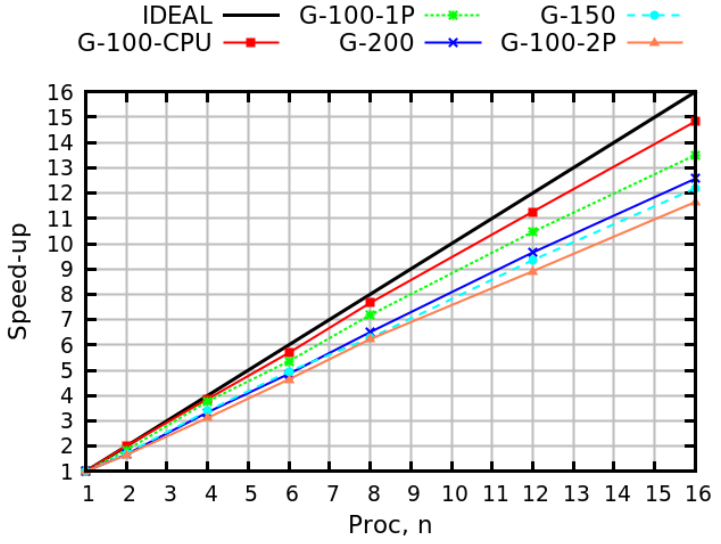


Fig. 3.16. Parallel speed-up attained

Figure 3.16 illustrates parallel speed-up of VisPartDEM. The curves G-200, G-150 and G-100-2P represent visualization of 200194, 150119 and 100036 particles employing GPU rendering performed by two processes per multi-core node, respectively. The curve G-100-1P represents visualization of 100036 particles employing GPU rendering performed by one processes per node while

the curve G-100-CPU represents visualization of 100036 particles employing CPU rendering. The special curve Ideal shows the ideal speed-up.

Parallel speed-up of visualization employing GPU rendering is lower than that of visualization based on CPU rendering. However, execution time of visualization employing GPU is significantly shorter than that of using CPU rendering. Higher speed-up was measured visualizing larger discrete particle systems. It becomes obvious that in usual grid conditions, when two processes use one GPU on multi-core architecture, parallel speed-up achieved by GPU rendering is moderate. It can be concluded that Figure 3.16 proves sufficient speed-up of visualization performed on grid testbed based on multi-core architecture.

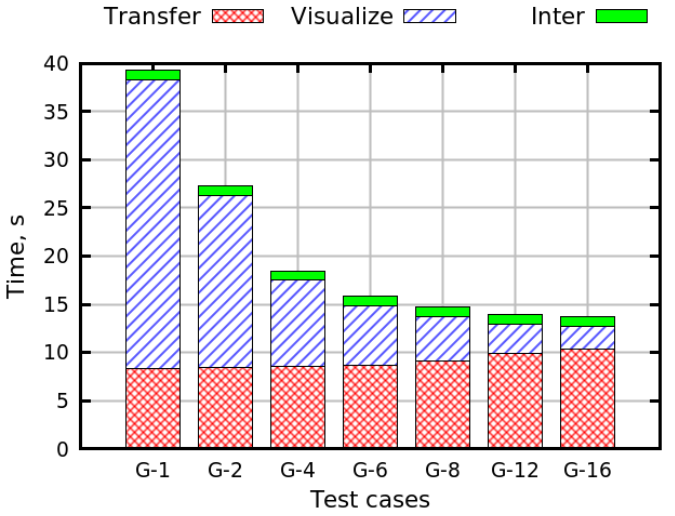


Fig. 3.17. Contribution of visualization procedures to the total benchmark time

In Figure 3.17, the chart compares the contribution of data transfer (Transfer), visualization (Visualize) and interactive session (Inter) to the total visualization time of 200194 particles measured on the CE. Visualization includes data reading, glyphs and parallel GPU rendering, while interactive session consists of image encoding, frame sending, receiving, decoding and displaying as well as processing interactive events. The stacked columns G-1, G-2, G-4, G-8, G-12 and G-16 represent visualization time on grid site by using 1, 2, 4, 8, 12 and 16 processes, respectively. Figure 3.17 shows that visualization time was significantly reduced employing parallel processing. Moreover, performing glyphs-based benchmark the time consumed by interactive session is negligible. However, the overall problem is not scalable, because of the data transfer from SE, which is insignificantly growing.

3.3. Performance of Crack Surface Generation and Visualization

3.3.1. Visual Results of the Proposed Methods

The developed visualization methods were applied to visualize geometry of propagating cracks.

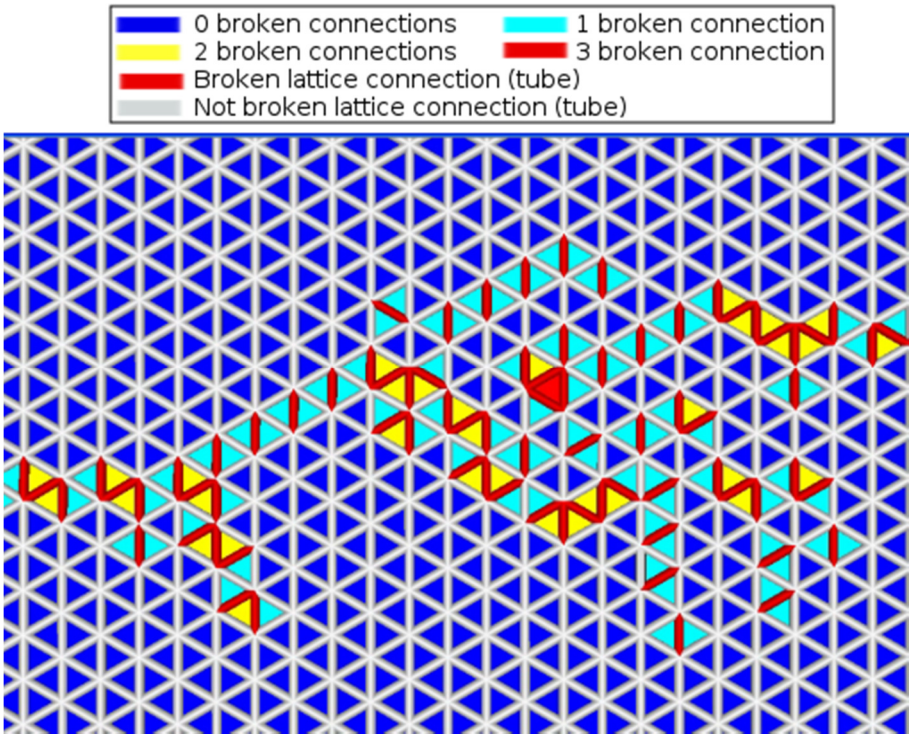


Fig. 3.18. Visualization of the lattice and cracks in 2D

Figure 3.18 presents visualization of cracks performed by the cell attribute-based method in 2D. The lattice connections represented by coloured tubes are plotted for illustrative purposes. A red tube indicates that the corresponding lattice connection is already broken and the force coupling neighbouring particles is equal to zero. Cracks are visualized by colour mapping of the calculated cell attributes on the generated triangles (2.1). The predefined colour table is employed for relevant visualization of the investigated phenomenon. The cells coloured in blue do not contain cracks, which indicates zero values of the cell attribute calculated by the formula (2.2). The triangles including only one broken

connection are coloured in cyan. The cells that have two broken connections are coloured in yellow. It means that crack cuts the cell. The triangles coloured in red contains the branching crack, because all edges of the cell are broken and the investigated region is highly fractured.

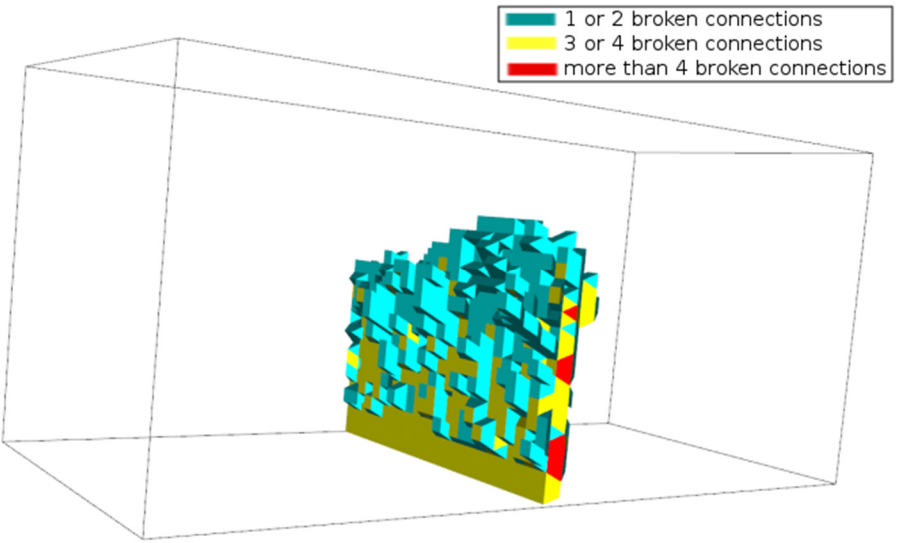


Fig. 3.19. Visualization of cracks in 3D

Figure 3.19 shows crack propagation in 3D. In this case, the colour mapping is more sophisticated, because the generated topology contains two types of cells, i.e. pyramids with triangular or quadrilateral bases. The 3D cells that do not contain the broken connections are extracted by the filter vtkExtractCells. Pyramids containing small defects, i.e. one or two broken connections, are coloured in cyan. The yellow colour represents the cells containing 3 or 4 broken connections, which illustrates the cells cut by the crack surface. The red pyramids, containing more than 4 broken connections, indicate the highly fractured regions of material. The crack propagating upwards from the specimen bottom, damaged by the initial defect, can be clearly observed in Figure 3.19.

Figure 3.20 presents visualization of cracks propagating in the rectangular plate. The local Voronoi decomposition is generated in the vicinity of cracks to define the extended contact surfaces of the particles coloured in blue. The cracks are represented by red tubes that are plotted on the contact surfaces. Model validation is illustrated by rendering the coloured lattice connections. The elongated connections are represented by green tubes while the high deformation areas are shown by yellow tubes. In the regions containing a lot of broken lattice connections the particles have more freedom to move. Consequently, high

deformations can prevent generating the Voronoi decompositions suitable for visualization purposes.

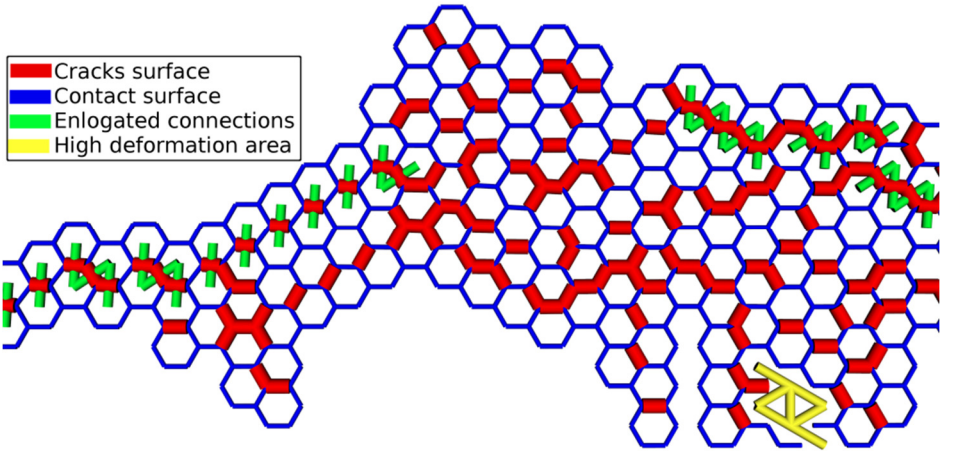


Fig. 3.20. Visualization of cracks, extended contact surfaces and model validation

Figure 3.21 shows visualization of the crack surfaces propagating in the three-dimensional domain. The cracks are plotted on the extended contact surfaces represented by the faces of the local Voronoi decomposition. The crack surface is coloured in red, while the edges of the relevant Voronoi faces are represented by tubes. It is worth noting that the numerical results of the lattice–based DEM at the micro-level are scattered, because of the randomly distributed material properties of the individual particles. It is well-known that fractures can be of stochastic nature and have scattered defects at the micro-level (F. Zhou *et al.* 2005). The visualization confirms that the crack surface is continuous only at the location of the initial defect. The holes and disjoint pieces of the surfaces appearing due to the scattered nature of fractures can be observed in the upper part of the cracked region. The visualization of cracks on the extended contact surfaces facilitates the analysis of the structure and topological connectivity of crack surfaces as well as the identification of holes.

Figure 3.22 presents visualization of the crack curves extracted by applying two different methods in a case of the two-dimensional benchmark. The cracks are represented by red tubes, while local decompositions are shown by blue tubes. Figure 3.22a and Figure 3.22c show the cracks extracted by using the cell cut-based method, while Figure 3.22b and Figure 3.22d illustrate an application of the cell centre-based method. It is worth noting that the methods generate local decompositions of different topologies. The cracks obtained by using the cell cut-based method seem to be smaller and contain separate vertices, representing

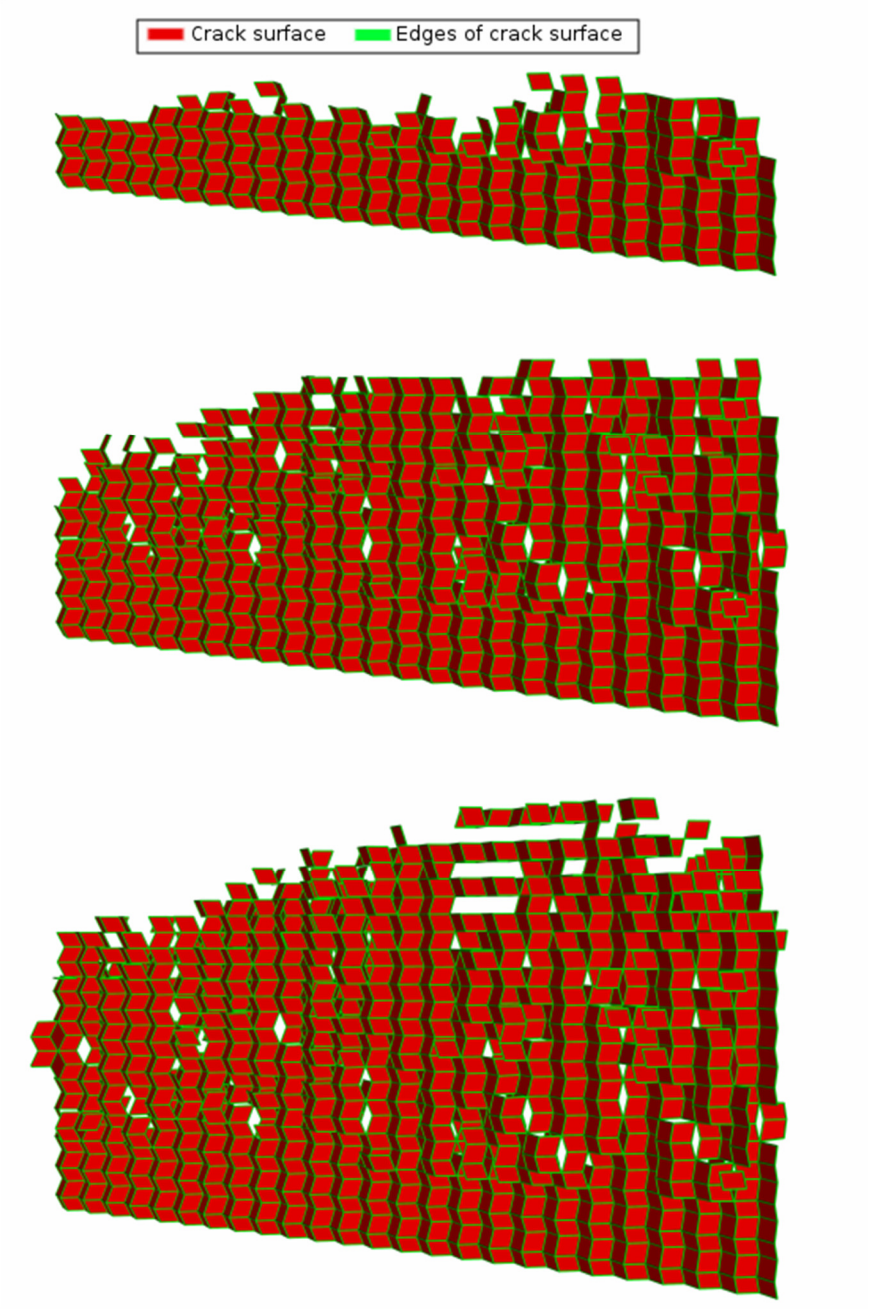


Fig. 3.21. Visualization of cracks surfaces using local Voronoi-based method

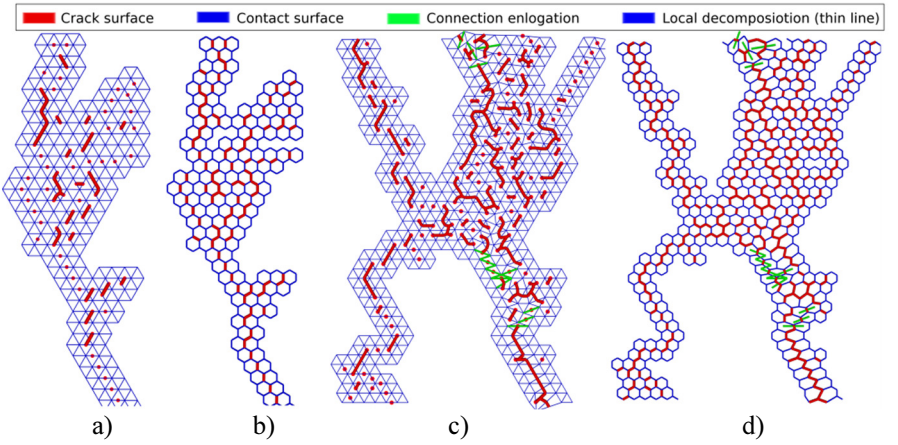


Fig. 3.22. Local decompositions and cracks extracted by the investigated methods in 2D: a) the cell cut-based method, b) the cell centre-based, c) application of the cell cut-based technique to highly fractured and deformed regions, d) application of the cell centre-based technique to highly fractured and deformed regions

spatially disconnected broken connections. The cell centre-based method represents such connections as disconnected lines (tubes) plotted in the contact region between the neighbouring particles. Figure 3.22c and Figure 3.22d demonstrate the application of the developed extraction methods to the fractured regions of a highly deformed lattice marked by green connections. In the regions containing a lot of broken connections of the lattice, the particles have more freedom to move. In the fractured regions, the initial topological connectivity of the nearest contacting particles fixed by the connections of the computational lattice does not hold because of the intensely changing positions of the particles. Consequently, high deformations of the computational lattice can prevent generation of accurate space decompositions suitable for visualization purposes. In spite of these difficulties both developed methods were able to extract cracks in the last time step of computations containing 548 broken connections.

Figure 3.23 shows visualization of cracks propagating in a three–dimensional domain. Figure 3.23a presents 103-rd time step of the data set A visualized by using the cell cut-based method, while Figure 3.23b shows visualization of the same time step obtained by using the cell centre-based method. The crack surfaces are coloured in red, while the edges of the faces and lines of the cracks are represented by green tubes. Red points show the disjoint vertices of the cracks resulting from the application of the cell cut-based method to separate broken connections, having no direct topological connectivity with the fractured regions. The transparent faces of the cracks help to explore the overlapping surfaces. In the presented figure, the extracted surfaces are not smoothed to visualize the

simulation results accurately and to provide researchers with the possibility visually to check the accuracy of the employed numerical method. The curved surfaces extracted by the cell centre-based method are more continuous. Moreover, they consist of faces and do not have any primitives of lower dimensionality, such as lines and vertices. However, the surfaces extracted by the cell cut-based method perfectly illustrate the straightforward propagation of the main crack.

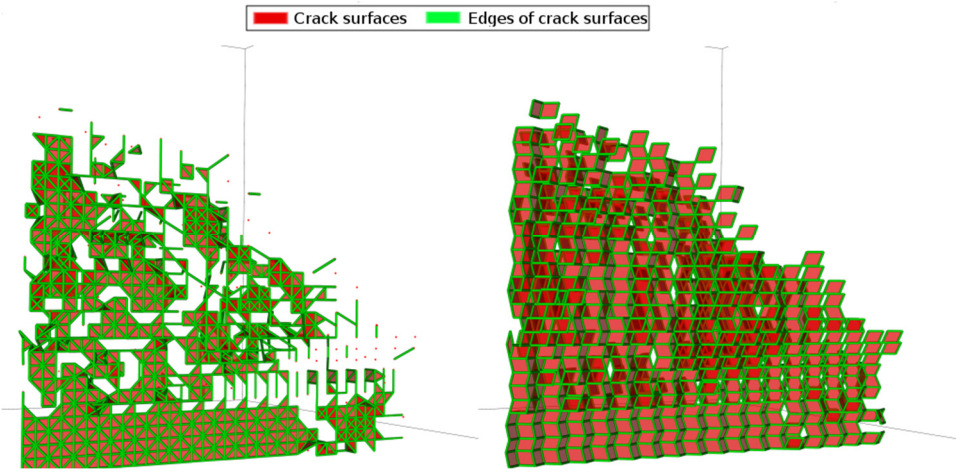


Fig. 3.23. Visualization of crack surfaces extracted by using different methods: a) cell cut-based method, b) cell centre-based method

Visualization confirms that the crack surface is only continuous at the location of the initial defect. The holes and disjoint pieces of the surfaces formed due to the scattered nature of the fracture phenomena can be observed in the upper part of the cracked region. The extraction of the crack surfaces facilitates the analysis of the structure and topological connectivity of cracks as well as the identification of holes.

3.3.2. Performance Analysis

The benchmark tests were carried out on two personal computers to validate the computational performance of the visualization methods. Hardware characteristics of the personal (C1) are listed below: Intel® Core i7-3770 3.40 GHz CPU, 2x1 TB HDD, 16 GB DDR3 1600 MHz RAM and Nvidia GeForce GTX 660 Ti GPU (1344 CUDA cores, 2 GB GDDR5, 144.2 GB/sec memory bandwidth). Hardware characteristics of the personal (C2) are listed below: Intel®

Core i7-4790 3.60 GHz CPU, 2x1 TB HDD, 32 GB DDR3 1600 MHz RAM and NVIDIA Quadro K5000 GPU (1536 CUDA cores, 4 GB GDDR5, 173 GB/sec memory bandwidth). Performing the benchmark, the attention was focused on the performance of the crack extractions methods, reader and renderer of the resulting polygon meshes. Mapping was not considered, because it took a very short time equal approximately to 0.0001 s. The benchmark tests were repeated up to ten times and the averaged values were examined.

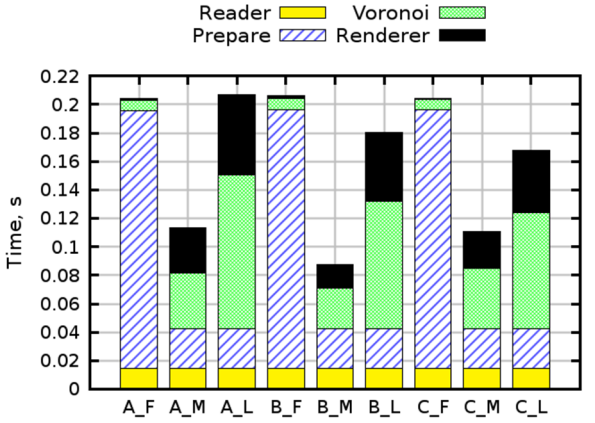


Fig. 3.24. Contribution of visualization procedures of the local Voronoi-based method to the total benchmark time

Figure 3.24 shows the contribution of the execution time of the various visualization procedures used in performing the 3D benchmark of local Voronoi–based method on personal computer C1. The columns A_F, B_F and C_F represent visualization of the first time step of the simulations of the data sets A, B and C, respectively. The columns A_M, B_M and C_M represent visualization of the middle time step of the simulations of the data sets A, B and C, respectively. The columns A_L, B_L and C_L represent visualization of the last time step, containing a large number of broken connections of the data sets A, B and C, respectively. The chart compares the contribution of the data reader (Reader), preparation of data structures (Prepare), generation of Voronoi decomposition (Voronoi) and rendering (Render) to the total visualization time.

In the visualization of the first time step, the reader took 7.1% of the total benchmark time. In the case of other time steps, the time percentage for data reading did not exceed 8.6% of the total time. The preparation of data structures also took a considerable time in the case of the first time step, but later it did not exceed 16.5%. The time consumed by rendering, made 27.4%, 26.9% and 26.1% of the execution time in visualizing the last time step on the data sets A, B and C,

respectively. In the case of the last time step, the method took the largest amount of time, because it generated the largest local decomposition. The generation of the local Voronoi decompositions took 52.2%, 49.7% and 48.7% of the execution time of the data sets A, B and C, respectively.

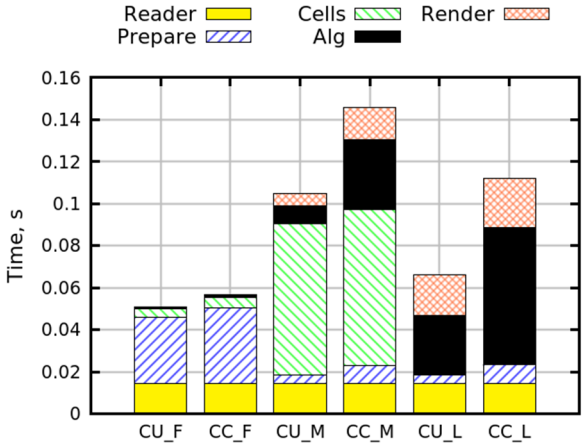


Fig. 3.25. Contribution of different visualization procedures of the cell cut-based method and the cell centre-based method to the total execution time

Figure 3.25 shows the contribution of the execution time of various procedures used in visualizing the data set A of a 3D benchmark on personal computer C1. The columns CU_F and CC_F present the visualization of the first time step of the simulations by using the cell cut-based method and the cell centre-based method, respectively, while the columns CU_L and CC_L present the visualization of the last time step. The columns CU_M and CC_M present the visualization of the time step requiring the longest execution time. Usually, it happens during the time step, when quickly propagating cracks occupy the largest new area and local decomposition needs the largest augmentation. The chart compares the contribution of the data reader (Reader), preparation of data structures (Prepare), generation of cells from the lattice connections (Cells), generation of crack surfaces (Alg) and the rendering time (Render) with respect to the total visualization time.

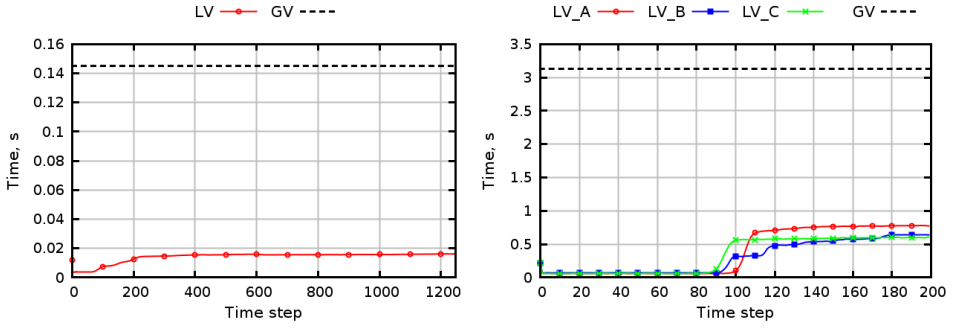
Data reading took nearly equal time intervals, which did not exceed 0.015 s. In the case of the time step requiring the maximum amount of computations, the reader took up to 13.6% of the total benchmark time. The preparation of data structures took more than 60% of the execution time, visualizing the first time step, but in other presented cases, it did not exceed 8.0% of the total benchmark time. On the contrary, the longest rendering time was measured by visualizing the last time step of the computations, because of the largest number of graphical primitives employed to represent the developed cracks. In general, the cell cut–

based method produced a large number of disjoint vertices or lines, which were rendered more quickly than the faces generated by the cell centre-based extraction method. In the case of the data set A, the measured difference was equal to 3.6% of the total benchmark time. In the case of other data sets, the obtained difference did not exceed 7.6% of the total visualization time. In the case of the maximum visualization load, the generation of cells from the lattice connections took approximately 0.074 s, which makes 68.7% and 51.0% of the total visualization time measured by the cell cut-based method and the cell centre-based method, respectively. It can be easily observed that the generation of the local cell-based decomposition consumed the largest amount of time, when it was necessary to cover the significant percentage of space by cells. It is obvious that the cell cut-based method generated the crack surfaces faster than the cell centre-based method, because it produced additional local decomposition based on the geometrical cell centre. The observed difference made 33.1% of the total execution time in the case of the last time step, when the geometric cell centre-based decomposition was the largest. It is worth noting that the geometry of this decomposition depends on the positions of particles, which change in time. Therefore, it is hardly possible to use the effective augmentation strategy in this case.

Figure 3.26 presents the quantitative comparison of the execution time of local Voronoi-based method including visual validation and global Voronoi diagrams on personal computer C1. Figure 3.26a shows the dependency of the execution time on the visualized time step of the 2D benchmark, Figure 3.26b presents this dependency for the 3D benchmark. The curves LV_A, LV_B and LV_C represent the execution time of the local Voronoi-based method of data sets A, B and C, while the curve GV represent the generation of the global Voronoi diagrams. Voro++ library (Rycroft 2009) was employed to generate the global Voronoi diagrams for the sake of quantitative comparison. The obtained results show that the execution time of the local Voronoi-based method was performed much faster than the construction of the global Voronoi diagrams. In spite of long execution time required for model validation, the local Voronoi–based method took 25.0%, 20.4% and 19.1% of the time required for generating the global Voronoi diagram. In the case of 2D benchmark, containing, the local Voronoi-based method took only 7.8% of time consumed by the global Voronoi diagrams.

The time consumed for generating the local decompositions depends on the number of the broken lattice connections and the area of the fractured region. Figure 3.27 demonstrates the time variation of the number of broken connections and the number of faces of the generated local decompositions. The curves Broken_A, Broken_B and Broken_C represent the number of broken connections in the case of data sets A, B and C, respectively. The curves Faces_A, Faces_B

and Faces_C represent the number of generated faces in the case of data sets A, B and C, respectively. Rapid changes in the number of broken connections can be observed between the time steps 80 and 110 in the case of all data sets. A considerable increase in the generated faces of the Voronoi cells was also observed in this interval. In 2D case, the faces of the local Voronoi decomposition had 16.1% of the faces of the global Voronoi diagram covering the whole solution domain. The generated faces of the local decompositions had 20.2%, 17.4% and 15.5% of the faces of the global Voronoi diagram in the case of the 3D data sets A, B and C, respectively.



a) b)

Fig. 3.26. Time consumed by local Voronoi-based method and the global Voronoi diagrams: a) in 2D, b) in 3D

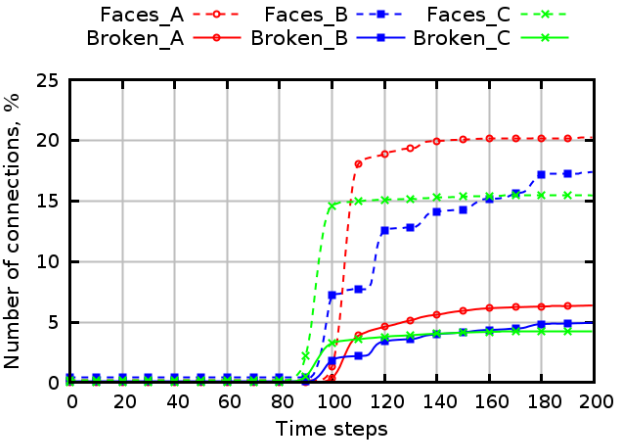


Fig. 3.27. The number of broken connections and faces of the generated local decompositions

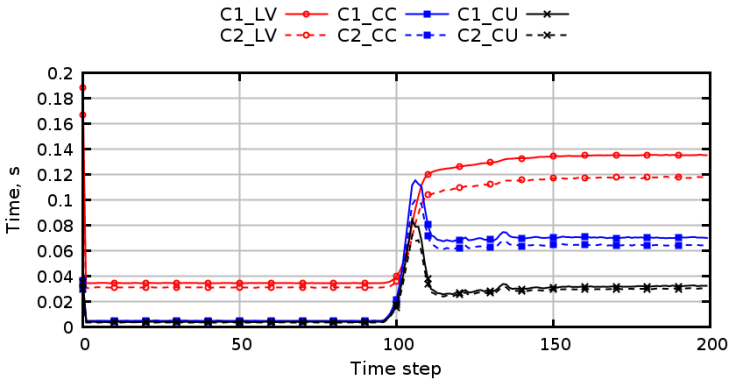
Figure 3.28 shows the quantitative comparison of the time consumed by three different visualization methods on two personal computers named C1 and C2. Figures 3.28a, 3.28b and 3.28c presents the visualization time of data sets A, B and C, respectively. The curves including acronyms LV, CC and CU represent the extraction of crack surfaces performed by the local Voronoi-based method, the cell centre-based method and the cell cut-based method, respectively. The curves including acronyms C1 and C2 represents the execution time measured on computers C1 and C2, respectively.

In the first time step, high values were obtained, because of the time consuming preparation of data structures (Figures 3.24 and 3.25). The main increase in the consumed time could be observed when approaching the 100-th time step, because of the fastest increase in the number of the broken connections and the relevant fractured regions, which had to be covered by local decompositions. In the case of the cell centre-based method and the cell cut–based method, the effective augmentation strategy caused the following decrease in the growth ratio leading to the reduction of the consumed time, because the largest fractured regions had been already covered by the generated decompositions. In the case of the local Voronoi-based method, the execution time remained at the same level, when fractured regions stopped to grow. Thus, the Voronoi based-method required the longest execution time.

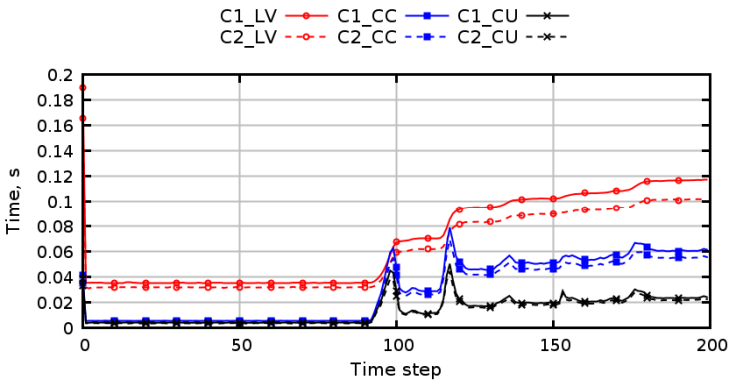
It can be easily observed that the cell cut-based method helps to extract the crack surfaces much faster than the cell centre-based method. At the end of the visualized time interval, the surface extraction performed by the cell cut-based method took 46.2%, 39.5% and 41.8% of the time consumed by the cell centre–based extraction of cracks in the case of data sets A, B and C, respectively. It is worth noting that the visualization workload strongly depends on the size and distribution of the fractured regions as well as on the percentage of the broken connections in the fractured regions. In the opposite, the employed different computes did not have the large influence to visualization time. For example, at the end of computational interval of the data set A the difference equal to 12.80%, 8.24% and 6.07% of the total execution time was observed on different hardware in the case of the local Voronoi-based method, the cell centre-based method and the cell cut-based method, respectively.

Figure 3.29 presents the obtained dependence of execution time of visualization on the number of broken connections. Figures 3.29a, 3.29b and 3.29c presents the visualization time of data sets A, B and C, respectively. The curves LV, CC and CU represent the extraction of crack surfaces performed by the local Voronoi-based method, the cell centre-based method and the cell cut–based method, respectively. In the case of the local Voronoi-based method, almost linear growth rates can be observed until the number of broken connections reached some threshold value, which was different for the investigated data sets. This

a)



b)



c)

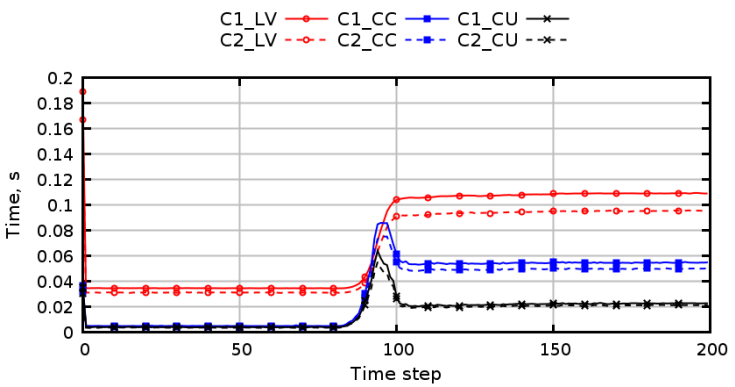


Fig. 3.28. Time consumed by using different visualization methods:
a) data set A, b) data set B, c) data set C

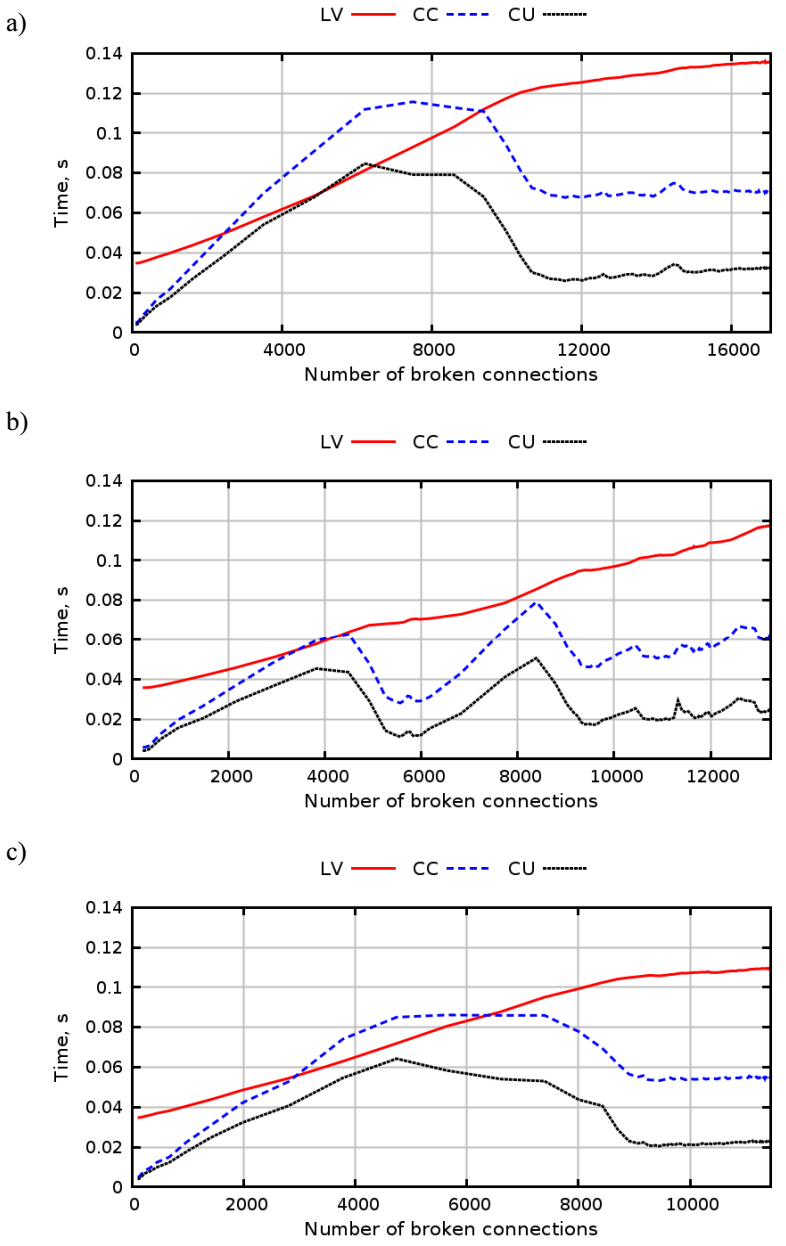


Fig. 3.29. The dependence of the execution time on the number of broken connections: a) data set A, b) data set B, c) data set C

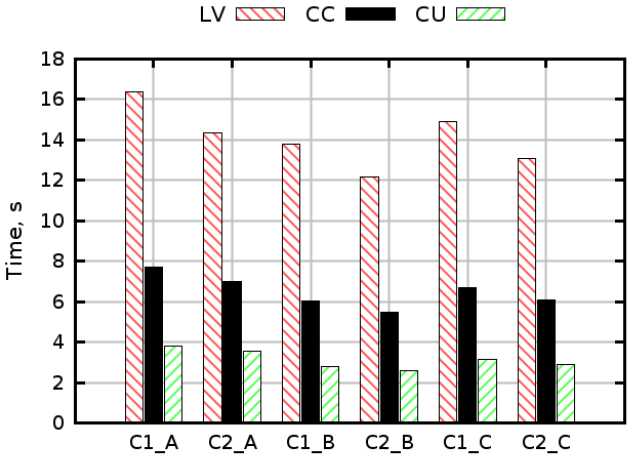


Fig. 3.30. Accumulated execution time consumed by the different visualization methods

value was related to the development of the fractured region covered by the local decomposition. The effective augmentation strategy caused the reduction of visualization time despite the increasing number of broken connections and the growing fractured region in the case of the cell centre-based method and the cell cut-based method.

Figure 3.30 shows the execution time consumed by the different surface extraction methods for visualization of all time steps of data sets A, B and C on different computers. The columns LV, CC and CU represent visualization of cracks performed by the local Voronoi-based method, the cell centre-based method and the cell cut-based method, respectively. The columns including acronyms C1 and C2 represent the execution time measured on computers C1 and C2, respectively. The columns including acronyms A, B and C present the visualization time of data sets A, B and C, respectively. It can be observed that the cell cut-based method extracted the crack surfaces much faster than the other methods. The local Voronoi-based method needed the longest execution time, because it was hardly possible to implement the effective augmentation strategy into its structure. The surface extraction performed by the cell cut-based method took 23.4%, 20.2% and 22.2% of the time consumed by the local Voronoi-based extraction of cracks in the case of data sets A, B and C, respectively. The visualization performed by the cell centre-based method took 47.2%, 43.8% and 46.4% of the time consumed by the local Voronoi-based visualization in the case of data sets A, B and C, respectively. The largest difference in hardware performance equal to 12.35% was observed in the case of the local Voronoi–based method, because the longest execution time. The performance difference

measured on different computers was equal to 9.19% and 7.55% in the case of the cell centre- and cell cut-based methods.

3.3.3. Accuracy

Accurate visualization of crack surfaces is still a challenging problem, particularly, in the regions of the highly deformed lattice. Particles have more freedom to move and to deform the lattice in the regions containing large numbers of the broken connections. It is worth mentioning that the most of computational models also have limitations in these complex cases. The surface extraction methods based on the Voronoi diagrams cannot be applied in such regions, because of increasing inconsistency between the lattice connections and the faces of the generated Voronoi decompositions.

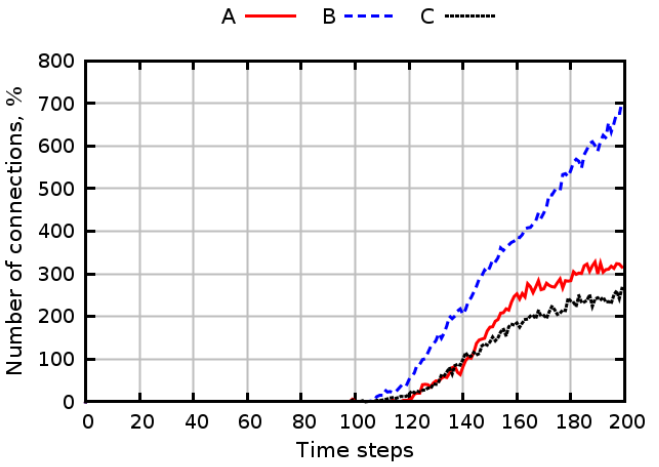


Fig. 3.31. Time variation of the number of inconsistencies between the lattice and cells of Voronoi decomposition

In general, the crack surfaces can cut particles, reducing the accuracy of the applied surface extraction methods. Faces of local Voronoi decompositions are used as extended contact surfaces of neighbouring particles. Thus, the Voronoi–based method accurately defines crack surfaces, because faces are located between neighbouring particles (Figure 2.8). The cell centre-based method for extracting the crack surfaces was developed as an alternative to the cell cut–based method to obtain the crack surfaces more accurately by taking into account the spherical shape of the particles. This method can be applied in the regions of the highly deformed lattice, where the Voronoi-based method cannot be applied.

However, the faces of cell centre-based decomposition cut particles in the regions of deformed lattice (Figure 2.12).

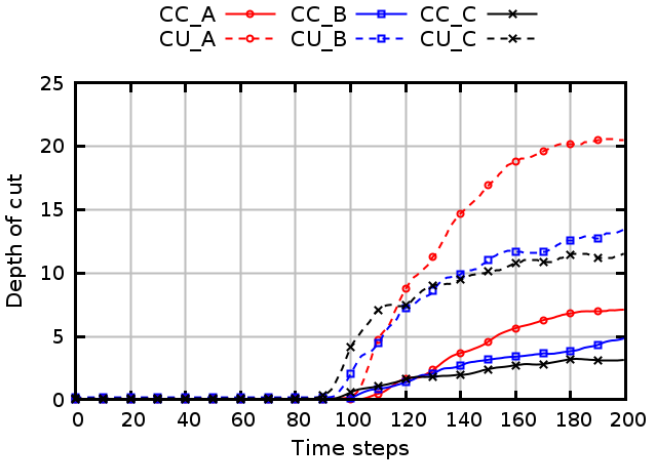


Fig. 3.32. Time variation of the total depth of the cuts made by the extracted crack surfaces in particles

The penetration depth of the particular cut can serve as a good accuracy measure. Figure 3.32 shows time variation of the total depth of the cuts produced by the extracted crack surfaces. The curves CU_A, CU_B and CU_C represent the total depth resulting from the application of the cell cut-based method to the data sets A, B, and C, respectively. The curves CC_A, CC_B and CC_C represent the cell centre-based method applied to the data sets A, B, and C, respectively. The total depth of the cuts resulting from the application of the cell centre-based method made 12.6%, 14.1% and 8.9% of the depth produced by the cell cut-based method in the case of the data sets A, B and C, respectively. The largest difference as well as the largest total depth value could be observed in the case of the data set A, which had the largest number of the broken lattice connections. The extracted crack surfaces most deeply penetrated the particles in the highly fractured regions covered by the highly deformed lattices.

3.4. Conclusions of Chapter 3

1. The grid visualization e-service VizLitG, employing GLSL shaders and partial data set transfer from SE, was able to efficiently perform remote visualization of the considered discrete particle systems.

2. The applied GLSL particle shaders reduced the visualization time by an order of magnitude, while the full data set transfer from SE consumed more than 50% of the total benchmark time.
3. The performance of different transfer protocols and software was investigated in order to present the quantitative comparison of time consumed by the data transfer. The performance of JAX-WS Runtime was the best for the considered data sets.
4. Employing partial data set transfer, the communication time of JAX-WS Runtime was reduced by up to 7.6 times and became almost negligible.
5. High parallel speed-up of visualization equal to 14.8 is achieved on 16 working nodes of grid testbed by using CPU rendering. Lower speed-up equal to 13.5 is measured performing visualization based on GPU rendering on 16 working nodes. However, execution time of visualization employing GPU is significantly shorter than that of using CPU rendering.
6. In the 3D case, the number of the broken connections made 6.4%, 4.9% and 4.3% of all lattice connections, while the local Voronoi decompositions had 20.2%, 17.4% and 15.5% of the faces of the global Voronoi diagram in the case of the data sets A, B and C, respectively.
7. The quantitative comparison showed that the generation of the local Voronoi decompositions including model validation took 25.0%, 20.4% and 19.1% of the time consumed by the global Voronoi filter in the case of the data sets A, B and C, respectively.
8. The surface extraction performed by the cell cut-based method took 23.4%, 20.2% and 22.2% of the time consumed by the local Voronoi-based extraction of cracks in the case of data sets A, B and C, respectively.
9. The surface extraction performed by the cell centre-based method took 47.3%, 43.8% and 46.4% of the time consumed by the local Voronoi-based extraction in the case of data sets A, B and C, respectively.
10. The number of inconsistent connections makes 0.12%, 0.26% and 0.10% of the total amount of connections in the case of the data sets A, B and C, respectively.
11. The total depth of the cuts resulting from the application of the cell centre–based method made 12.6%, 14.1% and 8.9% of the depth produced by the cell cut-based method in the case of the data sets A, B and C, respectively.

General Conclusions

1. Literature review and initial analysis show that data transfer between remote parts of distributed visualization systems and infrastructures consumes significant part of visualization time, which is very difficult to reduce. Moreover, visualization of crack surfaces, propagating in discrete particle systems, presents great challengers to researchers because of disjoint pieces of surfaces and unavailability of a suitable scalar field defining the geometry of cracks.
2. The quantitative comparison of the performance of the developed service with other data transferring software available on grid revealed that the performance of JAX-WS Runtime was the best for the considered data sets. Moreover, the developed partial data set transfer reduced the transfering data size, therefore, the communication time was diminished up to 7.6 times in comparison with full dataset transfer.
3. The quantitative comparison showed that the developed software, based on the local decompositions, significantly outperformed the code based on the global Voronoi diagrams. The generation of the local Voronoi decomposition took from 19.1% to 25.0% of the time consumed by the global Voronoi filter, respectively.
4. The quantitative comparison of visualization methods revealed that the cell cut-based method was capable of extracting the crack surfaces much faster

than the other methods. The surface extraction performed by the cell cut–based method took up to 23.4% of the time consumed by the method based on the geometrical cell centre. The observed difference in time consumed by the local Voronoi decomposition and the decomposition based on the cell centre was up to 47.3% of the total benchmark time.

5. The local Voronoi decompositions do not cut particles revealing the highest accuracy. The accuracy of the cell centre-based surface extraction is significantly higher than that of the cell cut-based method. The total depth of the cuts resulting from the application of the cell centre-based surface extraction did not exceed 14.1% of the depth obtained, when the cell cut–based method was used.

References

Amenta, N., Choi, S., Kolluri, R. K. 2001. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications – SMA '01* (pp. 249–266). New York, USA: ACM Press.

Aurenhammer, F. 1991. Voronoi diagrams---a survey of a fundamental geometric data structure. *ACM Computing Surveys*, *23*(3):345–405.

Ayachit, U. 2015. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Incorporated.

Barauskas, R., Kačianauskas, R., Belevičius, R. 2004. *Baigtinių elementų metodo pagrindai*. Vilnius: Technika.

Bertin, J. 2010. *Semiology of Graphics: Diagrams* (1 edition.). Esri Press.

Bethel, W., Siegerist, C., Shalf, J., Shetty, P. 2003. VisPortal: Deploying grid-enabled visualization tools through a web-portal interface. In *Proceedings of WACE 2003*.

Bethel, W., Tierney, B., Lee, J., Gunter, D., Lau, S. 2000. Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization. *ACM/IEEE SC 2000 Conference (SC'00)*.

Biddiscombe, J., Graham, D., Pierre, M. 2008. Visualization and analysis of SPH data. *ERCOFTAC Bulletin*, *76*:9–12.

Brodlie, K., Walton, J., Wood, J. 2003. GViz - Visualization Middleware for e-Science. In *Proceeding VIS "03 Proceedings of the 14th IEEE Visualization 2003 (VIS"03)* (p. 82). IEEE Computer Society.

Brose, G., Vogel, A., Duddy, K. 2001. *Java programming with Cobra: advanced techniques for building distributed applications*. John Wiley & Sons Inc.

Brune, S. B. and S. A. and M. A. and J. B. and P., Gropp, and K. B. and V. E. and W., Knepley, and D. K. and M., Smith, and L. C. M. and K. R. and B., Zhang, and H. 2014. *PETSc Users Manual*.

CEI. 2009. EnSight. https://www.ceisoftware.com/. Accessed 9 February 2015

Cleary, P. W. 2009. Industrial particle flow modelling using discrete element method. *Engineering Computations*, *26*(6):698–743.

Cooper, C. 2002. Visual Beans project.

Cundall, P. A., Strack, O. D. L. 1979. A discrete numerical model for granular assemblies. *Géotechnique*, *29*(1):47–65.

Cusatis, G., Bažant, Z. P., Cedolin, L. 2006. Confinement-shear lattice CSL model for fracture propagation in concrete. *Computer Methods in Applied Mechanics and Engineering*, *195*(52):7154–7171.

Dey, T. K., Goswami, S. 2003. Tight Cocone: A Water-tight Surface Reconstructor. *Journal of Computing and Information Science in Engineering*, *3*(4):302.

EGEE. 2009. GLite. http://glite.web.cern.ch/glite/. Accessed 9 February 2015

Englander, R. 2002. *SOAP*. O'Reilly Media.

Evjen, B., Sharkey, K., Thangarathinam, T., Kay, M., Vernet, A., Ferguson, S. 2007. *Professional XML*. Wrox.

Favre, J. M., Valle, M. 2005. AVS and AVS/Express. In *Visualization Handbook* (pp. 655–672).

Federl, P., Prusinkiewicz, P. 2004. Finite Element Model of Fracture Formation on Growing Surfaces. In M. Bubak, G. van Albada, P. Sloot, & J. Dongarra (Eds.), *Computational Science - ICCS 2004* (Vol. 3037, pp. 138–145). Springer Berlin / Heidelberg.

Fleishman, S., Cohen-Or, D., Silva, C. T. 2005. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, *24*(3):544.

Folk, M., Heber, G., Koziol, Q., Pourmal, E., Robinson, D. 2011. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases - AD '11* (pp. 36–47). New York, USA: ACM Press.

Foster, I. 2005. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *Network and Parallel Computing* (pp. 2–13). Springer Berlin Heidelberg.

Foster, I., Kesselman, C., Tuecke, S. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Journal International Journal of High Performance Computing Applications*, *15*(3):200 – 222.

Foulser, D. 1995. IRIS Explorer. *ACM SIGGRAPH Computer Graphics*, *29*(2):13–16.

Gobron, S., Chiba, N. 2001. Crack pattern simulation based on 3D surface cellular automata. *The Visual Computer*, *17*(5):287–309.

Goda, T. J., Ebert, F. 2005. Three-dimensional discrete element simulations in hoppers and silos. *Powder Technology*, *158*(1-3):58–68.

Goncalves, A. 2009. *Beginning Java™ EE 6 Platform with GlassFish™ 3: From Novice to Professional*. Apress.

Gropp, W., Lusk, E., Doss, N., Skjellum, A. 1996. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, *22*(6):789–828.

Haimes, R. 1994. PV3 - A distributed system for large-scale unsteady CFD visualization. In *32nd Aerospace Sciences Meeting and Exhibit* (pp. 94–103). Reston, Virigina.

Han, K., Feng, Y. T., Owen, D. R. J. 2007. Performance comparisons of tree-based and cell-based contact detection algorithms. *Engineering Computations*, *24*(2):165–181.

Hansen, C. D., Johnson, C. R. 2005. *The Visualization Handbook*. Academic Press.

Hibbard, W., Rueden, C., Emmerson, S., Rink, T., Glowacki, D., Whittaker, T., Murray, D., Fulker, D., Anderson, J. 2005. Java distributed components for numerical visualization in VisAD. *Communications of the ACM*, *48*(3):98–104.

Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P. D., Klosowski, J. T. 2002. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Transactions on Graphics*, *21*(3):1–10.

Iben, H. N., O'Brien, J. F. 2009. Generating surface crack patterns. *Graphical Models*, *71*(6):198–208.

JCraft. 2014. JSch - Java Secure Channel. http://www.jcraft.com/jsch/

Jendrock, E., Evans, I., Gollapudi, D., Haase, K., Srivathsa, C. 2010. *The Java EE 6 Tutorial: Basic Concepts*. Addison-Wesley Professional.

Kačeniauskas, A., Kačianauskas, R., Maknickas, A., Markauskas, D. 2011. Computation and visualization of discrete particle systems on gLite-based grid. *Advances in Engineering Software*, *42*(5):237–246.

Kačeniauskas, A., Pacevič, R., Bugajev, A., Katkevičius, T. 2010. Efficient visualization by using ParaView software on BalticGrid. *Information Technology and Control*, *39*(2):108–115.

Kaci, M., Méndez Muñoz, V., Amorós Vicente, G. 2010. A Decentralized Deployment Strategy and Performance Evaluation of LCG File Catalog Service. *Journal of Grid Computing*, *9*(3):345–354.

Kačianauskas, R., Maknickas, A., Kačeniauskas, A., Markauskas, D., Balevičius, R. 2010. Parallel discrete element simulation of poly-dispersed granular material. *Advances in Engineering Software*, *41*(1):52–63.

Kadashevich, I., Stoyan, D. 2008. A beam-network model for autoclaved aerated concrete and its use for the investigation of relationships between Young's modulus and microstructure. *Computational Materials Science*, *43*(2):293–300.

Kalen, M. 2009. *Java Web Services: Up and Running*. O'Reilly Media.

Karihaloo, B. L., Shao, P. F., Xiao, Q. Z. 2003. Lattice modelling of the failure of particle composites. *Engineering Fracture Mechanics*, *70*(17):2385–2406.

Khanal, M., Tomas, J. 2009. Oblique impact simulations of high strength agglomerates. *Advanced Powder Technology*, *20*(2):150–157.

Kitsunezaki, S. 2011. Crack growth in drying paste. *Advanced Powder Technology*, *22*(3):311–318.

Kitware. 2010. *VTK User's Guide*. Kitware Inc.

Klein, R., Langetepe, E., Nilforoushan, Z. 2009. Abstract Voronoi diagrams revisited. *Computational Geometry*, *42*(9):885–902.

Kolluri, R. 2008. Provably good moving least squares. *ACM Transactions on Algorithms*, *4*(2):1–25.

Kozicki, J., Tejchman, J. 2008. Modelling of fracture process in concrete using a novel lattice model. *Granular Matter*, *10*(5):377–388.

Krantz, S. G., McCarthy, J. E., Parks, H. R. 2006. Geometric characterizations of centroids of simplices. *Journal of Mathematical Analysis and Applications*, *316*(1):87–109.

Kranzlmuller, D., Kurka, G., Heinzlreiter, P., Volkert, J. 2002. Optimizations in the grid visualization kernel. In *Proceedings 16th International Parallel and Distributed Processing Symposium* (p. 237). Washington, DC, USA: IEEE Computing society.

Kruggel, H., Rickelt, S., Wirtz, Sw., Scherer, V. 2009. A Numerical Study on the Sensitivity of the Discrete Element Method for Hopper Discharge. *Journal of Pressure Vessel Technology*, *131*(3):111–121.

Labatut, P., Pons, J.-P., Keriven, R. 2009. Robust and Efficient Surface Reconstruction From Range Data. *Computer Graphics Forum*, *28*(8):2275–2290.

Laszewski, G., Foster, I., Gawor, J., Lane, P. 2001. A Java commodity grid kit. *Concurrency and Computation: Practice and Experience*, *13*(8–9):645–662.

Li, M., Baker, M. 2005. *The Grid: Core Technologies*. Wiley.

Lilliu, G., van Mier, J. G. . 2003. 3D lattice type fracture model for concrete. *Engineering Fracture Mechanics*, *70*(7-8):927–941.

Liu, J. X., Deng, S. C., Liang, N. G. 2007. Comparison of the quasi-static method and the dynamic method for simulating fracture processes in concrete. *Computational Mechanics*, *41*(5):647–660.

Lu, L., Lévy, B., Wang, W. 2012. Centroidal Voronoi Tessellation of Line Segments and Graphs. *Computer Graphics Forum*, *31*(2pt4):775–784.

Marinilli, M. 2001. *Java Deployment with JNLP and WebStart*. Sams Publishing.

Markauskas, D., Kačianauskas, R., Džiugys, A., Navakas, R. 2009. Investigation of adequacy of multi-sphere approximation of elliptical particles for DEM simulations. *Granular Matter*, *12*(1):107–123.

Matsukura, R., Koyamada, K., Tan, Y., Karube, Y., Moriya, M. 2004. VizGrid: Collaborative visualization grid environment for natural interaction between remote researchers. *FUJITSU Scientific and technical journal*, *40*(2):205–216.

Mayer, A., McGough, S., Furmento, N. 2005. ICENI: an integrated Grid middleware to support e-Science. *Component Models and Systems for Grid Applications*, 109–124.

Mohammad, H., Ladan, T. 2012. Cloud Computing Uncovered: A Research Landscape. *Advances in Computers*, *86*:41–85.

Moreland, K., Wylie, B., Pavlakos, C. 2001. Sort-last parallel rendering for viewing extremely large data sets on tile displays. In *PVG '01 Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics* (pp. 85–92). IEEE.

Nagella, S., Sastry, L. 2005. Visualization on the UK National Grid Service using GAPtk, a generic toolkit. In *Proceedings of the UK e-Science All Hands Conference 2005* (pp. 1–4). Engineering and Physical Sciences Research Council.

Naregi. 2005. Naregi Project. http://www.naregi.org/. Accessed 10 February 2015

Nocaj, A., Brandes, U. 2012. Computing Voronoi Treemaps: Faster, Simpler, and Resolution-independent. *Computer Graphics Forum*, *31*(3pt1):855–864.

Novotny, J. 2002. The Grid Portal Development Kit. *Concurrency and Computation: Practice and Experience*, *14*(13-15):1129–1144.

Nvidia. 2014. NVIDIA GRID vGPU. http://www.nvidia.com/object/virtual-gpus.html. Accessed 9 March 2015

Ostoja-Starzewski, M. 2002. Lattice models in micromechanics. *Applied Mechanics Reviews*, *55*(1):35.

Parisi, D. R., Masson, S., Martinez, J. 2004. Partitioned Distinct Element Method Simulation of Granular Flow within Industrial Silos. *Journal of Engineering Mechanics*, *130*(7):771–779.

Parker, S., Johnson, C. 1995. SCIRun: A Scientific Programming Environment for Computational Steering, 52.

Polak, M., Kranzlmüller, D. 2008. Interactive videostreaming visualization on grids. *Future Generation Computer Systems*, *24*(1):39–45.

Radeke, C. A., Glasser, B. J., Khinast, J. G. 2010. Large-scale powder mixer simulations using massively parallel GPU architectures. *Chemical Engineering Science*, *65*(24):6435–6442.

RealityGrid. 2005. RealityGrid Project. http://www.realitygrid.org/. Accessed 10 February 2015

Robert A. Van Engelen, K. A. G. 2002. The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks. In *CCGRID '02 Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid* (p. 128). IEEE Computer Society.

Rojek, J., Oñate, E., Labra, C., Kargl, H. 2011. Discrete element simulation of rock cutting. *International Journal of Rock Mechanics and Mining Sciences*, *48*(6):996–1010.

Rosenthal, P., Linsen, L. 2008. Smooth surface extraction from unstructured point-based volume data using PDEs. *IEEE transactions on visualization and computer graphics*, *14*(6):1531–1546.

Rosenthal, P., Linsen, L. 2009. Enclosing Surfaces for Point Clusters Using 3D Discrete Voronoi Diagrams. *Computer Graphics Forum*, *28*(3):999–1006.

Russell, M., Wehrens, O., Novotny, J. 2004. GridSphere: an advanced portal framework. In *Proceedings. 30th Euromicro Conference, 2004.* (pp. 412–419). IEEE.

Rycroft, C. H. 2009. Voro ++ : a three-dimensional Voronoi cell library in C ++. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *19*(4):1–16.

Schmidt, D. C. 2009. The Adaptive Communication Environment (ACE). http://www.cs.wustl.edu/~schmidt/ACE.html. Accessed 15 April 2015

Schroeder, W., Martin, K., Lorensen, B. 2006. *The visualization toolkit : an object-oriented approach to 3D graphics*. Kitware Inc.

SGI. 2009. Remote visualization software VizServer. https://www.sgi.com/productssoftware/vizserver/. Accessed 10 February 2015

Shreiner, D., Sellers, G., Kessenich, J., Bill, L.-K. 2013. *OpenGL Programming Guide*. Addison-Wesley Professional.

Stalling, D., Westerhoff, M., Hege, H.-C. 2005. Amira: a highly interactive system for visual data analysis. In *Visualization Handbook* (pp. 749–767).

Stanton, J., Newhouse, S., Darlington, J. 2002. Implementing a Scientific Visualisation Capability Within a Grid Enabled Component Framework. In *Proceedings of 8th International Euro-Par Conference, Lecture Notes in Computer Science* (Vol. 2400, pp. 885–888). Paderborn, Germany: Springer Berlin Heidelberg.

Tan, Y., Yang, D., Sheng, Y. 2009. Discrete element method (DEM) modeling of fracture and damage in the machining process of polycrystalline SiC. *Journal of the European Ceramic Society*, *29*(6):1029–1037.

Thomas, M. W., Schnetter, E. 2010. Simulation Factory: Taming Application Configuration and Workflow on High-End Resources. *CoRR*, *1*:10.

Thompson, D., Braun, J., Ford, R. 2000. OpenDX: Paths to Visualization. VIS. *Inc., Missoula, MT*.

Tirumala, A., Qin, F., Dugan, J., Ferguson, J., Gibbs, K. 2006. Iperf. http://iperf.fr/. Accessed 9 January 2015

Uematsu, K. 2014. Processing defects in ceramic powders and powder compacts. *Advanced Powder Technology*, *25*(1):154–162.

Vadluga, V., Kačianauskas, R. 2009. Lattice-based six-spring discrete element model for discretisation problems of 2D isotropic and anisotropic solids. *Mechanika*, *76*(2):11–19.

Valette, G., Prévost, S., Lucas, L., Léonard, J. 2006. SoDA project: A simulation of soil surface degradation by rainfall. *Computers and Graphics*, *30*(4):494–506.

Valette, G., Prévost, S., Lucas, L., Léonard, J. 2008. A Dynamic Model of Cracks Development Based on a 3D Discrete Shrinkage Volume Propagation. *Computer Graphics Forum*, *27*(1):47–62.

Walther, J. H., Sbalzarini, I. F. 2009. Large-scale parallel discrete element simulations of granular flow. *Engineering Computations*, *26*(6):688–697.

Wernecke, J. 1994. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*. Addison-Wesley Professional.

Wood, J., Brodlie, K. 2007. Computational steering in visualization dataflow environments. In *MODSIM 2007: International Congress on Modelling and Simulation* (pp. 3077 – 3083). Modelling & Simulation Soc Australia & New Zealand Inc.

Wright, H., Brodie, K., Wood, J. 1997. Collaborative visualization. In *Visualization '97., Proceedings* (pp. 253–259). IEEE Computer Society.

Xvid. 2009. XVid codec 1.1.3. https://www.xvid.com/. Accessed 10 March 2015

Yegulalp, S. 2013. Amazon ushers in graphics as a service. *InfoWorld*. http://www.infoworld.com/article/2612788/amazon-web-services/amazon-ushers-in-graphics-as-a-service.html. Accessed 9 March 2015

Zang, M. Y., Lei, Z., Wang, S. F. 2007. Investigation of impact fracture behavior of automobile laminated glass by 3D discrete element method. *Computational Mechanics*, *41*(1):73–83.

Zhou, F., Molinar, J.-F., Shioya, T. 2005. A rate-dependent cohesive model for simulating dynamic crack propagation in brittle materials. *Engineering Fracture Mechanics*, *72*(9):1383–1410.

Zhou, K., Hou, Q., Wang, R., Guo, B. 2008. Real-time KD-tree construction on graphics hardware. *ACM Transactions on Graphics*, *27*(5):1.

Zhu, H. P., Zhou, Z. Y., Yang, R. Y., Yu, A. B. 2007. Discrete particle simulation of particulate systems: Theoretical developments. *Chemical Engineering Science*, *62*(13):3378–3396.

Zhu, H. P., Zhou, Z. Y., Yang, R. Y., Yu, A. B. 2008. Discrete particle simulation of particulate systems: A review of major applications and findings. *Chemical Engineering Science*, *63*(23):5728–5770.

Zukowski, J. 2005. *The Definitive Guide to Java Swing*. Apress.

A List of Scientific Publications by the Author on the Topic of the Dissertation

Papers in the Reviewed Scientific Journals

Pacevič, R.; Kačeniauskas, A.; Markauskas, D. 2015. Visualization of cracks by using the local Voronoi decompositions and distributed software. *Advances in engineering software* 84:85–94, ISSN 0965-9978. (THOMSON JCR 2013: 1.422), doi:10.1016/j.advengsoft.2015.02.004.

Pacevič, R.; Kačeniauskas, A.; Markauskas, D.; Radvilavičius, L.; Kutas, R. 2013. Cell attribute-based algorithm for crack visualization. *Information technology and control* 42(3):253–259. ISSN 1392–124X. (THOMSON JCR 2013: 0.813), doi:10.5755/j01.itc.42.3.2575.

Pacevič, R.; Kačeniauskas, A. 2011. VizLitG. Grid visualization e-service enabling partial data set transfer from storage elements of gLite-based grid infrastructure. *Journal of Grid Computing* 9(4):573–589, ISSN 1570–7873. (THOMSON JCR 2011: 1.310), doi: 10.1007/s10723-011-9193-0.

Other Papers

Pacevič, R., Kačeniauskas, A. 2015. Deployment of visualization software and GPU rendering on an OpenStack Cloud Infrastructure. *Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*. Civil-Comp Press, Vol. 107, 1–11, doi:10.4203/ccp.107.19.

Pacevič, R., Kačeniauskas, A., Markauskas, D. 2013. Analysis of crack geometry using distributed visualization software. *Proceedings of the Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering,* Civil-Comp Press, Vol. 101, 1–13, doi:10.4203/ccp.101.51.

Pacevič, R.; Kačeniauskas, A. 2012. Influence of network communications to the final performance of grid visualization software. *Communications in Computer and Information Science: Information and Software Technologies,* Springer-Verlag Berlin Heidelberg, vol. 319, 312–323, doi: 10.1007/978-3-642-33308-8_26.

Summary in Lithuanian

Įvadas

Problemos formulavimas

Vizualizavimo priemonės vaidina svarbų vaidmenį mokslinės analizės bei matematinio modeliavimo cikle, kurį sudaro duomenų paruošimas, skaitinis uždavinio sprendimas, grafinė rezultatų analizė ir tolesnis valdančiųjų parametrų koregavimas (Hansen et al. 2005). Tiek diskrečiųjų dalelių sistemų (Cundall et al. 1979) moksliniai skaičiavimai, tiek ir modernūs fiziniai eksperimentai generuoja didelius duomenų kiekius, kuriuos analizuoti bei suvokti tampa vis sunkiau. Gautų rezultatų vizualizavimas tampa vis svarbesniu, siekiant išanalizuoti sudėtingas priklausomybes, greitai suvokti rezultatus ir priimti teisingus sprendimus aukštųjų technologijų kūrimo procese. Dideli nagrinėjamų duomenų kiekiai ir nutolusi jų dislokacija kelia vis naujus iššūkius vizualizavimo sistemų kūrėjams. Išskirstytosios vizualizavimo sistemos atlieka intensyvius skaičiavimus ir teikiamas paslaugas skirtinguose kompiuteriuose, kad pagerintų efektyvumą. Daugelis aktualių uždavinių sprendžiami „Rocks“ kompiuterių klasteriuose ir „glite/EMI“ išteklių tinklo infrastruktūroje. Išskirstytieji išteklių tinklo (GRID) resursai suteikia naudotojams didžiules galimybes, bet kartu iškelia ir sudėtingus uždavinius. Analizuojant rezultatus reikia išskirstytosios vizualizavimo programinės įrangos, kuri greitai atlieka vizualizavimo užduotis ir efektyviai tvarko duomenis moderniose informacinių technologijų (IT) infrastruktūrose.

Diskrečiųjų dalelių sistemų (Cundall et al. 1979) modeliavimas yra pagrįstas dalelių pozicijomis ir tarp jų veikiančiomis jėgomis. 1D jungtys tarp dalelių netinka pilnavertei

interpoliacijai ir standartinėms vizualizavimo technikoms 3D erdvėje. Diskrečiosiomis dalelėmis modeliuojant kontinuumo mikrostruktūrą, tarp dalelių nutrūkusios jungtys identifikuoja mikropažeidimą (Rojek et al. 2011). Jungdamiesi tarpusavyje pažeidimai gali sudaryti sudėtingos formos plyšius, kurių geometrija nėra žinoma. Vizualizuojant plyšius, reikia sukonstruoti plyšių paviršių geometriją iš tarp dalelių nutrūkusių jungčių topologijos ir kaimyninių dalelių pozicijų. Sudėtinga plyšių forma su skylėmis ir plyšį apibrėžiančio skaliarinio lauko nebuvimas riboja standartinių paviršių ištraukimo metodų taikymo galimybes. Jungčių tarp judančių dalelių topologija yra fiksuota, todėl negalima tiesiogiai taikyti standartinių Voronojaus dekompozicijų (Aurenhammer 1991) ir Delauney tinklų (Amenta et al. 2001) generavimo metodų.

Darbo aktualumas

Vizualizavimas tampa galingu įrankiu duomenims analizuoti ir rezultatams pateikti įvairiose mokslo ir pramonės srityse (Hansen et al. 2005). Dideli duomenų kiekiai ir sudėtingas vizualizavimo procesas reikalauja daug programinės įrangos kūrėjų pastangų ir kompiuterinių resursų. Dideliems informacijos srautams analizuoti taikomos vizualizavimo sistemos ir e. paslaugos, veikia moderniose IT infrastruktūrose: kompiuterių klasteriuose, išteklių tinkluose (GRID) ir „debesyse“. Siekiant interaktyvių vizualizavimo greičių, efektyvus duomenų perdavimas tarp išskirstytosios infrastruktūros komponentų tampa ypatingai aktualiu.

Medžiagos pažeidimų evoliucijos analizė yra didelis iššūkis daugeliui tarpdalykinių mokslų, tame tarpe ir vizualizavimui (Gobron et al. 2001). Plyšių susidarymas aktualus statybinėse konstrukcijose, keramikoje (Uematsu 2014), džiovinimo procesuose (Kitsunezaki 2011) ir miltelių aglomeratuose (Khanal et al. 2009). Tiksliai apibrėžta plyšio paviršiaus geometrija palengvina vizualinę skaičiavimo rezultatų analizę ir irimo procesų suvokimą. Plyšio geometriją sukonstravus iš grafinių primityvų, gautą paviršių galima eksportuoti į inžinerinių uždavinių sprendimo programinę įrangą tolimesnei makrostruktūrų analizei.

Tyrimų objektas

Darbo tyrimų objektas – diskrečiųjų dalelių sistemų, modeliuojamų diskrečiųjų elementų metodu, vizualizavimas.

Darbo tikslas

Disertacijos tikslas – patobulinti diskrečiųjų dalelių sistemų, modeliuojamų diskrečiųjų elementų metodu, vizualizavimo metodus bei padidinti metodų realizacijos išskirstytojoje programinėje įrangoje greitaveiką.

Darbo uždaviniai

Darbo tikslui pasiekti ir mokslinei problemai spręsti darbe buvo iškelti šie uždaviniai:

1. Išanalizuoti diskrečiųjų dalelių sistemų išskirstytosios programinės įrangos vizualizavimo technologijas bei plyšių, sklindančių dalelių sistemose kontinuumui modeliuoti, geometrijos konstravimo ir vizualizavimo metodus.

2. Sukurti programinį modulį, leidžiantį sumažinti duomenų kiekį tarp išteklių tinklo infrastruktūros komponentų ir ištirti jo realizacijos greitaveiką.
3. Sukurti naujus plyšių, sklindančių diskrečiųjų dalelių sistemose, paviršių geometrijos konstravimo ir vizualizavimo metodus bei realizuoti pasiūlytus metodus išskirstytojoje vizualizavimo programinėje įrangoje.
4. Ištirti ir palyginti sukurtų diskrečiųjų dalelių sistemų vizualizavimo metodų realizacijų išskirstytosiose vizualizavimo sistemose greitaveiką.
5. Ištirti sukurtų diskrečiųjų dalelių sistemų vizualizavimo metodų tikslumą.

Tyrimų metodika

Darbe taikomi lyginamosios analizės ir literatūros analizės metodai, naudoti siekiant išanalizuoti tyrimo objektą ir atlikti literatūros analizę. Kompiuterinės grafikos ir skaičiuojamosios geometrijos žinios buvo taikomos plyšių vizualizavimo metodams kurti. Eksperimentinio tyrimo metodai buvo taikomi vykdant sukurtų vizualizavimo metodų ir prototipų efektyvumo bandymus.

Darbo mokslinis naujumas

Darbo mokslinis naujumas pagrįstas šiais rezultatais:

1. Sukurta originali dalinio duomenų rinkinio siuntimo iš „gLite/EMI“ išteklių tinklo duomenų saugyklos realizacija, kuri interaktyvaus vizualizavimo metu sumažina siunčiamų duomenų kiekį tarp resursų tinklo komponentų.
2. Sukurti nauji plyšių, sklindančių diskrečiųjų dalelių sistemose kontinuumui modeliuoti, geometrijos vizualizavimo metodai, pagrįsti Voronojaus ir geometrinių celių centrų dekompozicijomis. Lokalios dekompozicijos sudarytos fiksuotos jungčių tarp judančių dalelių topologijos pagrindu, todėl standartiniai Voronojaus dekompozicijų ir Delauney tinklų generavimo metodai negali būti pritaikyti.

Darbo rezultatų praktinė reikšmė

Darbe sukurta vizualizavimo programinė įranga leis analizuoti didelių diskrečiųjų dalelių sistemų modeliavimo rezultatus, reikalingus kuriant aukštąsias technologijas Lietuvoje. Nauji kintančių paviršių geometrijos vizualizavimo metodai, pagrįsti Voronojaus ir geometrinių celių centrų dekompozicijomis, reikalingi irimo mechanikos ir medžiagotyros mokslams bei naujų konstrukcijų ir medžiagų kūrimo procesui pagreitinti. Tikslus kintančio plyšio geometrijos nustatymas leis daug toliau ir giliau nagrinėti medžiagos irimo procesą bei mikro- ar net nanolygiuose gautus rezultatus tiesiogiai perkelti į taikomąją programinę įrangą, kuri inžinerinius uždavinius sprendžia makrolygyje. Sukurti metodai įdiegti projekte „Virtualizavimo, vizualizavimo ir saugos e. paslaugų technologijų kūrimas ir tyrimai“ (VP1-3.1-ŠMM-08-K) tiriamoje programinėje įrangoje.

Ginamieji teiginiai

1. Dalinio duomenų rinkinio siuntimo iš gLite/EMI duomenų saugyklos realizacija sutrumpina vizualizavimo laiką, mažindama tarp grid infrastruktūros komponentų siunčiamų duomenų kiekį.

2. Sukurti originalūs metodai sukonstruoja plyšio geometriją ir leidžia vizualizuoti plyšio sklidimą monodispersinėse dalelių sistemose, modeliuojamuose diskrečiųjų elementų metodu.

Darbo rezultatų aprobavimas

Disertacijos tema paskelbti 6 moksliniai straipsniai. Trys iš jų yra publikuoti recenzuojamuose mokslo žurnaluose, kurie įtraukti į Thomson Reuters ISI Web of Science duomenų bazę ir turi citavimo indeksą.

Disertacijos rezultatai buvo aprobuoti 5 konferencijose, keturios iš jų yra tarptautinės mokslinės konferencijos:

- The Fourth International Conference on Parallel, Distributed, GRID and Cloud Computing for Engineering (PARENG2015). 2015 m. kovo 24–27, Dubrovnikas, Kroatija.
- 7th World Congress on Particle Technology (WCPT7), 2014 m. gegužės 19–22, Beijing, Kinija.
- The Third International Conference on Parallel, Distributed, GRID and Cloud Computing for Engineering (PARENG2013), 2013 m. kovo 25–27, Pécs, Vengrija.
- 18th International Conference on Information and Software Technologies (ICIST 2012). 2012 m. rugsėjo 13–14, Kaunas, Lietuva.
- LMA IIth jaunųjų mokslininkų konferencija „Fizinių ir technologijos mokslų tarpdalykiniai tyrimai“. 2012 m. vasario 14, Vilnius, Lietuva.

Disertacijos struktūra

Disertaciją sudaro įvadas, trys pagrindiniai skyriai, bendrosios išvados, literatūros šaltinių sąrašas, autoriaus publikacijų disertacijos tema sąrašas, santrauka lietuvių kalba. Darbo apimtis – 126 puslapiai neskaitant priedų, tekste yra 10 formulės, 63 paveikslai ir 6 lentelės. Rašant disertaciją buvo panaudota 110 literatūros šaltinių.

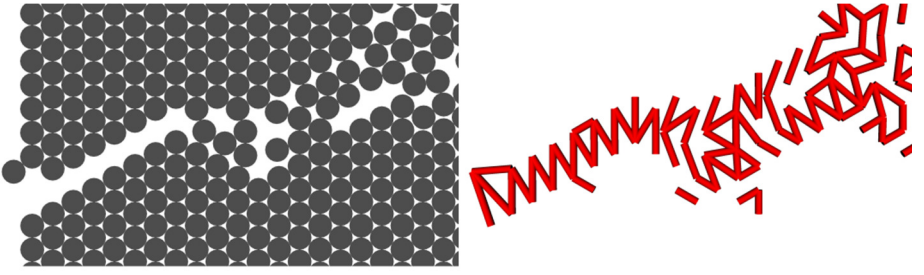
1. Išskirstytųjų vizualizavimo sistemų ir plyšių vizualizavimo metodų apžvalga

Skyriuje apžvelgta vizualizavimo programinė įranga, išskirstytosios vizualizavimo sistemos išteklių tinklo (GRID) aplinkose, dalelių sistemų modeliavimas diskrečiųjų elementų metodu ir plyšių, sklindančių diskrečiųjų dalelių sistemose, vizualizavimo metodai.

Vizualizavimo e. paslaugos sparčiai populiarėja išteklių tinkle ir kitose išskirstytosiose informacinių technologijų infrastruktūrose, nors jų valdymas ir duomenų perdavimas heterogeninėse sistemose yra labai sudėtingi. Atlikta programinės įrangos apžvalga parodė, kad daugelis vizualizavimo sistemų nėra tiesiogiai pritaikytos išteklių tinklui, todėl reikalauja didelių programuotojų pastangų siekiant jas naudoti pasirinktoje išteklių tinklo aplinkoje. Dauguma apžvelgtų nutolusių vizualizavimo naudotojo sąsajų išteklių tinklo aplinkose labai priklauso nuo išteklių tinklo programinės įrangos sistemos arba

„midleware“, komunikacijos sąsajų, taikomųjų uždavinių tipo bei bazinės vizualizavimo sistemos. Literatūros analizė atskleidė, kad pakankamai daug universalių vizualizavimo sistemų integruota į išteklių tinklus, pagrįstuose Globus sistemine įranga. Didelė dalis Europos išteklių tinklo infrastruktūros, sukurtos gLite/EMI sisteminės įrangos pagrindu. Didelė dalis vizualizavimo technologinių sprendimų negali būti tiesiogiai perkelta iš Globus į gLite/EMI aplinką. Dėl šios priežasties sudėtinga rasti universalias nutolusio vizualizavimo sistemas, veikiančias gLite/EMI aplinkoje ir gebančias atsiųsti tik dalį vizualizuojamo duomenų rinkinio iš duomenų saugyklų. Egzistuoja modernūs vizualizavimo įrankiai, pritaikyti tik siųsti pilnus duomenų rinkinius ir vizualizuoti atskirus uždavinius išteklių tinklo aplinkoje.

Medžiagoje atsirandančių pažeidimų evoliucijos analizė yra didelis iššūkis daugeliui tarpdalykinių mokslų, tame tarpe ir vizualizavimui. Esant didelėms apkrovoms medžiagoje atsiranda mikropažeidimai. Jungdamiesi tarpusavyje pažeidimai sudaro sudėtingos geometrinės formos plyšius. Egzistuojantys metodai nustato atsirandančius pažeidimus pradinėje stadijoje, bet negali tiksliai apibrėžti plyšio geometrijos. Plyšio atsiradimas 1D jungtyse nustatomas remiantis jėgos tarp dviejų dalelių dydžiu, o plyšys sklinda tarp daugelio dalelių 3D erdvėje. Diskrečiųjų elementų metodu modeliuojamuose dalelių sistemose plyšiai dažniausiai vizualizuojami technikomis, kurios nekonstruoja plyšio geometrijos. Dalelės spalvinamos pagal įvairių atributų reikšmes. Plyšiai, kurių plotis artimas ar didesnis už dalelių skersmenį, gali būti vaizduojami nuspalvintomis dalelėmis (S1a pav.). Tais atvejais, kai dalis jungčių jau nutrūko, bet tarpai tarp dalelių išlieka mažesni nei dalelių skersmuo, dalelių geometrija negali aiškiai pavaizduoti susidariusio plyšio paviršiaus 3D erdvėje. Pagrindine alternatyva, kuri plačiai paplitusi ne tik diskretinių elementų metodo (DEM), bet ir baigtinių elementų metodo programiniuose paketuose, laikomos nuspalvintos nutrūkusios jungtys (S1b pav.), vaizduojamos cilindrais ar atkarpomis. Nutrūkusios jungtys parodo vietą kurioje pažeistas medžiaga, bet nesuteikia jokios informacijos apie susidariusio plyšio geometriją. 3D erdvėje sklindančių plyšių paviršiai, apbrėžti 2D grafiniais primityvais, pateikia daugiau informacijos nei 1D nutrūkusių jungčių grafinės reprezentacijos, kuriomis pagrįsti paplitę vizualizavimo metodai.



a) b)

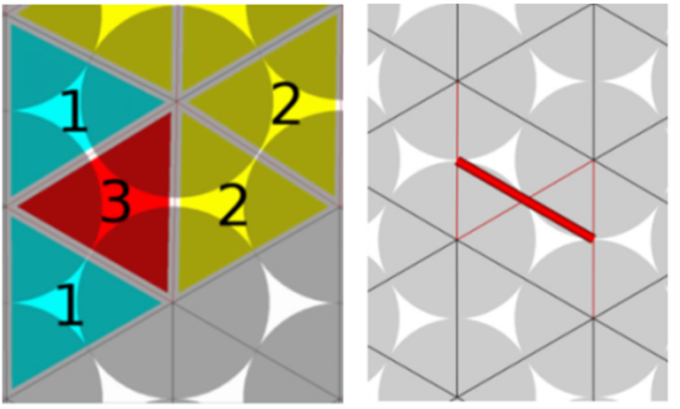
S1 pav. Plyšio sklidimas vizualizuotas 2D erdvėje: a) dalelių geometrija, b) nutrūkusios jungtys tarp dalelių

Atlikta literatūros analizė atskleidė, kad standartiniai paviršiaus rekonstrukcijos metodai netinka plyšiams vizualizuoti, nes nėra tinkamo skaliarinio lauko ar patogios erdvės diskretizacijos. Paviršiaus konstravimas yra labai komplikuotas, nes sudėtinga plyšių geometrija yra sudaryta iš atskirų dalių su skylėmis. Literatūros analizė parodė, kad standartiniai Voronojaus diagramų generavimo metodai taip pat netinka diskretiniu elementų metodu (DEM) modeliuojamoms dalelių sistemoms vizualizuoti. Skaičiavimams naudojama 1D jungčių topologija nekinta laiko atžvilgiu, tačiau to negalima pasakyti apie dalelių pozicijas, apibrėžiančias dalelių sistemos geometriją. Po tam tikro laiko tarpo dalelių sistemos geometrija ženkliai pasikeičia, o jos pradinio 1D jungčių tinklelio topologija tampa nesuderinama su standartine Voronojaus diagrama. Standartiniai metodai generuoja Voronojaus diagramos pagal dalelių pozicijas, o defektų atsiradimas nustatomas remiantis pradinėmis 1D jungtimis tarp dalelių.

2. Vizualizavimo metodai ir jų realizacijos išskirstytojoje programinėje įrangoje diskrečiųjų dalelių sistemoms vizualizuoti

Skyriuje detaliai analizuojami sukurti metodai, kurie naudojami plyšio paviršiams vizualizuoti.

Celių atributų ir celių kirtimo metodai. Celių atributų metodas skirtas plyšiams vizualizuoti monodispersinėje dalelių sistemoje. Siūlomas metodas generuoja srities dekompoziciją, naujas celes kurdamas iš 1D tinklelio jungčių. Taikant metodą suskaičiuojamos kiekvienos celės nutrūkusias briaunas, o rezultatą išsaugo skaliariniame celės atribute. Celės spalvinamos pagal atributo reikšmes, remiantis pasirinkta spalvų paieškos lentele (S2a pav.). Taip parodomos celės, kuriose yra nutrūkusių jungčių, o jų spalva pateikia kiekybinę informaciją apie nutrūkusių jungčių skaičių.



a) b)

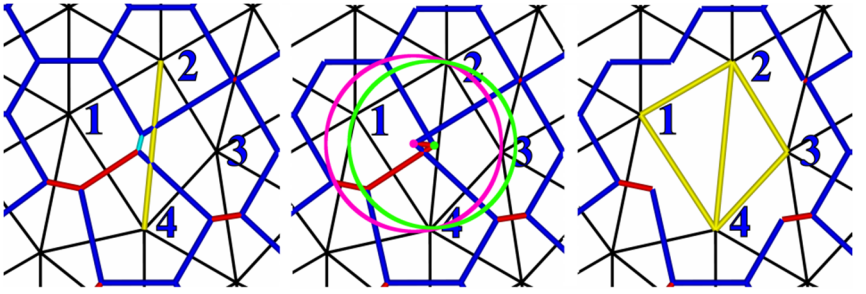
S2 pav. Metodų iliustracijos: a) celių atributų metodas; b) celių kirtimo metodas

Deja, celių atributo metodas neapibrėžia plyšio geometrijos, todėl jo funkcionalumas buvo praplėstas celių kirtimo metode. Sugeneravęs lokalią srities dekompoziciją ir

kiekvienoje celėje suskaičiavęs nutrūkusias jungtis, celių kirtimo metodas generuoja plyšio paviršių. Naudodamas informaciją apie nutrūkusias celės jungtis, metodas apytiksliai nustato plyšio paviršiaus geometriją nagrinėjamoje celėje, panašiai kaip ir žygiuojančių kubų metodas. Paviršiaus geometrija nusakoma grafiniais primityvais. Siekiant padidinti metodo greitaveiką, celių kirtimo metodas generuoja lokalią srities dekompoziciją ir taiko veiksmingą auginimo strategiją.

S2 paveikslai iliustruoja metodų taikymą 2D erdvėje. Plonos linijos vaizduoja tinklelio jungtis, kur raudona plona linija žymi nutrūkusią jungtį. S2a paveikslas vaizduoja celių atributų metodo schemą. Metodo sukurtos celės spalvojamos pagal apskaičiuoto skaliarinio atributo reikšmę. Žydrai spalvojamos celės, kurių skaliarinio atributo reikšmė lygi vienam, t. y. celė turi vieną nutrūkusią jungtį. Geltonai spalvojamos celės turinčios dvi nutrūkusias jungtis. Raudonai spalvojamos celės turinčios tris nutrūkusias jungtis. S2b paveikslas vaizduoja celių kirtimo metodo schemą. Metodas sujungia nutrūkusių jungčių vidurinius taškus, o gautą liniją vizualizuojame raudonu vamzdeliu. Celių kirtimo metodas nėra labai tikslus, nes net reguliaraus dalelių išsidėstymo atveju sugeneruotas paviršius kerta daleles.

Voronojaus dekompozicijos metodas. Sukurtas metodas yra skirtas lokaliai Voronojaus dekompozicijai generuoti, panaudojant dalelių pozicijas ir nekintančias 1D jungtis tarp dalelių. Lokalios dekompozicijos generavimo principas pagrįstas plokštumų, statmenų jungtims tarp dalelių, susikirtimais.

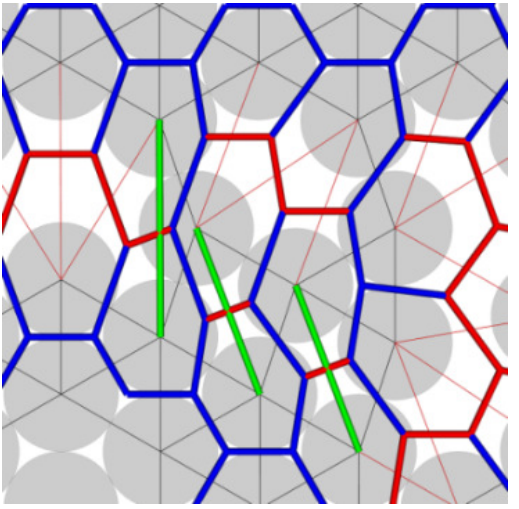


S3 pav. Voronoi dekompozicijos metodo iliustracijos

Pagrindinis ciklas vykdomas per daleles, jungiamas nutrūkusių jungčių, apie kiekvieną dalelę kuriant Voronojaus celes. Po to vykdomas ciklas per nagrinėjamos dalelės kaimynus – sukuriama kaimynines daleles skirianti plokštuma ir ieškoma sankirtų su turimomis plokštumomis. Daugiakampių ir tinklelių jungčių atitikimas turi būti patikrintas didelės deformacijos regionuose, kai standartinė Voronojaus diagrama neatitinka stacionarios 1D jungčių topologijos (S3a pav.). Sugeneruota celė bus Voronojaus celė, jeigu visi pradiniai mazgai atitiks tuščios sferos sąlygą (S3b pav.). Nustačius didelių deformacijų regioną, persidengiančios lokalios dekompozicijos celės nepiešiamos, o regionas pažymimas nuspalvintomis jungtimis (S3c pav.). Nutrūkusios jungtys, daleles veikiančios jėgos ir kiti atributai iš jungčių tiesiogiai perkeliami į atitinkamas Voronoi celių briaunas tolimesniam vizualizavimui. Šis metodas generuoja Voronoi dekompoziciją tik plyšio aplinkoje, siekiant taupyti kompiuterio išteklius.

S3 paveiksle vaizduojamas plyšys vizualizuotas lokalia 2D erdvės Voronoi dekompozicija. Plyšio geometrija vizualizuota raudonais cilindrais, o mėlyni cilindrai vaizduoja lokalią Voronoi dekompoziciją. Geltonais cilindrais pažymėti didelių deformacijų regionai. Žalias ir rausvas apskritimai iliustruoja sferos sąlygos tikrinimą deformuoto jungčių tinklelio regione.

Celių centrų metodas. Celių centrų metodas skirtas išplėsti plyšių vizualizavimo sritį, nes, skirtingai nei Voronoi dekompozicijos metodą, jį galima taikyti stipriai deformuoto jungčių tinklelio regionuose (S4 pav.). Celių centrų metodas tiksliau apibrėžia plyšio paviršiaus geometriją nei celių kirtimo metodas, nes atsižvelgia į dalelių sferinę formą.



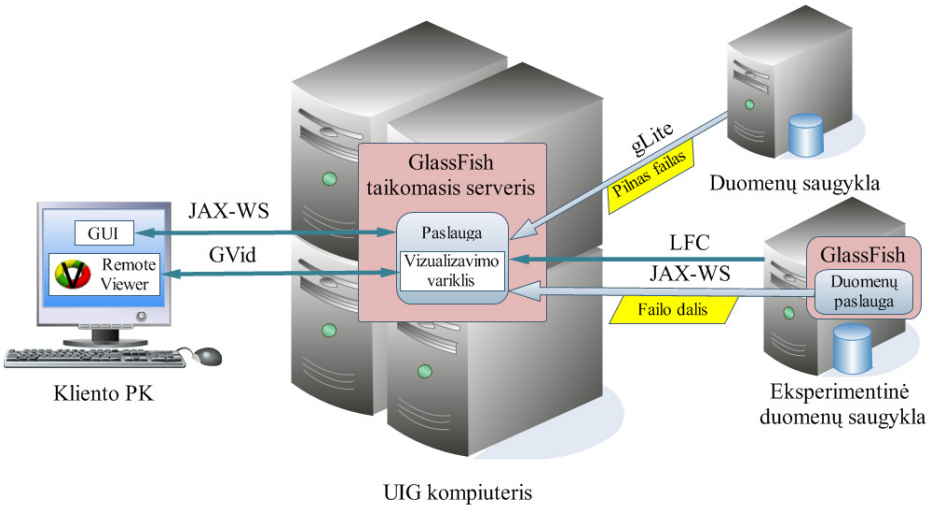
S4 pav. Plyšio vizualizavimas celių centrų metodu deformuotame 1D jungčių tinklelio regione

Pirmiausia, celių centrų metodas pažeistuose medžiagos regionuose iš 1D jungčių tarp dalelių generuoja srities dekompoziciją, analogišką celių kirtimo metodo dekompozicijai. Po to apskaičiuoja dekompozicijos celių geometrinius centrus. Finale iš pirmosios dekompozicijos celių geometrinių centrų sudaro antrąją erdvės dekompoziciją. Sugeneruota dekompozicija apibrėžia plyšio geometriją ne taip tiksliai, kaip lokali Voronoi dekompozicija, bet gali būti taikoma didesnių jungčių tinklelio deformacijų regionuose. Siekiant padidinti metodo greitaveiką, celių centrų metode taikoma veiksmingą dekompozicijų auginimo strategiją.

S4 paveiksle iliustruoja celių centrų metodo taikymą deformuotame 1D jungčių tinklelio regione. Plonos raudonos linijos nurodo nutrūkusias 1D jungtis, o juodos linijos vaizduoja nenutrūkusias jungtis. Plyšio paviršiai pavaizduoti raudonais cilindrais, o mėlyni cilindrai vaizduoja lokalią dekompoziciją. Žali cilindrai vaizduoja didelių deformacijų regionus.

Sukurti plyšių vizualizavimo metodai įdiegti vizualizavimo e. paslaugoje VizLitG ir išskirstytojoje vizualizavimo sistemoje VisPartDEM. Sukurtas vizualizavimo e. paslaugos

VizLitG (S5 pav.) prototipas, skirtas skaičiavimų rezultatams, dislokuotiems nutolusiose išteklių tinklo duomenų saugyklose (SE), interaktyviai nagrinėti, patogiai atsiųsti ir efektyviai vizualizuoti. VizLitG paremta kliento-serverio architektūra. E. paslaugos serveris realizuotas GlassFish taikomųjų programų serveryje, o kliento grafinė aplinka realizuota Java Swing biblioteka.



S5 pav. Vizualizavimo e. paslaugos VizLitG architektūra

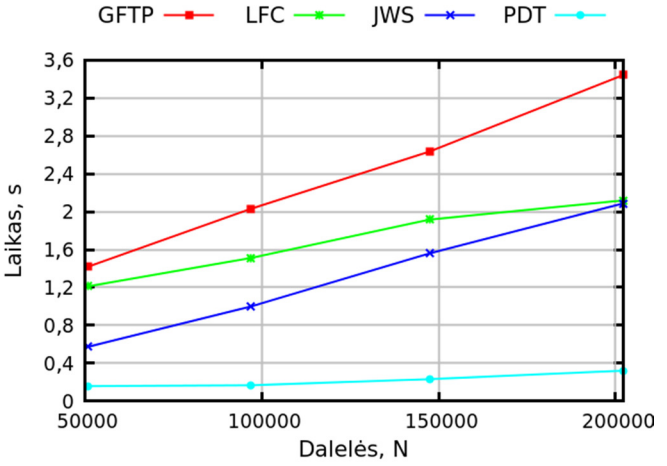
Įdiegta lanksti skaičiavimų rezultatų, saugomų HDF5 formatu, interaktyvaus nuskaitymo programinė įranga, nepriklauso nuo programavimo kalbos ir operacinės sistemos, todėl puikiai tinka heterogeninėms išteklių tinklo sistemoms. VizLitG leidžia ne tik atsiųsti visą rezultatų failą LFC/LCG ar GridFTP įrankiais. Duomenų saugykloje įdiegta paslauga „Data Service“ vartotojui suteikia galimybę parsisiųsti į VizLitG severį tik pasirinktas duomenų rinkinio dalis.

3. Pasiūlytų vizualizavimo metodų ir sukurtos programinės įrangos eksperimentiniai tyrimai

Skyriuje aprašomi atlikti dalinio duomenų persiuntimo iš duomenų saugyklos tyrimai. Diskrečios dalelių sistemos vizualizuojamos GLSL šešėliuokliais, kurių pagalba galima pasiekti interaktyvius vizualizavimo greičius. Didelių duomenų rinkinių siuntimas iš duomenų saugyklų sudaro didžiąją laiko dalį vizualizavimo procese, todėl pasirinktos dalinio duomenų siuntimo technologija leidžia sumažinti siunčiamų duomenų kiekį ir padidinti vizualizavimo greitį.

S6 paveiksle vaizduojamas duomenų siuntimo iš saugyklos SE laikas. Siekiant pateikti kiekybinį duomenų siuntimo rezultatų palyginimą, buvo ištestuoti skirtingi protokolai ir įvairi programinė įranga. Kreivė GFTP iliustruoja rezultatus, gautus taikant

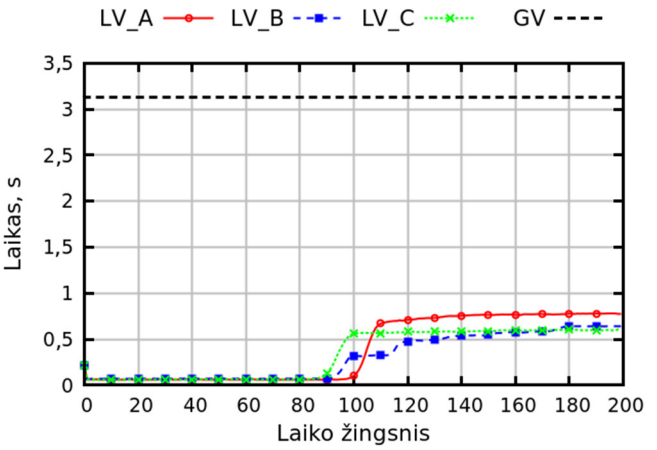
GridFTP protokolą, įdiegtą Java CoG Kit jGlobus modulyje. Kreivė LFC skirta loginio failų katalogo (Logical File Catalog) priemonių, įtrauktų į gLite distribuciją, rezultatams vaizduoti. Kreivė JWS iliustruoja laiką, kurį sunaudojo JAX-WS visam duomenų rinkiniui siųsti. Kreivė PDT iliustruoja laiką, kurį JAX-WS sugaišo siunčiant tik duomenų rinkinio dalį, būtiną vizualizavimui. Atlikti testai parodė, kad siunčiant nagrinėjamo dydžio duomenis, JAX-WS dirba efektyviausiai. Dar daugiau, siunčiant tik būtiną duomenų rinkinio dalį, siuntimo laiką pavyko sumažinti apytiksliai 7,6 karto, todėl jis tapo beveik nereikšmingu, lyginant su visu vizualizavimo procesu.



S6 pav. Duomenų siuntimas iš SE

Darbe nagrinėti plyšių vizualizavimo metodų greitaveikos ir tikslumo kiekybiniai palyginimai. Atliktas kiekybinis lokalios Voronoi dekompozicijos metodo realizacijos greitaveikos palyginimas su kitų autorių sukurtos Voro++ bibliotekos (Rycroft 2009) greitaveika. S7 paveiksle pateikti programų vykdymo laikai nagrinėjamais laiko žingsniais. Kreivės LV_A, LV_B ir LV_C rodo lokalios Voronoi dekompozicijos metodo vykdymo laiką vizualizuojant duomenų rinkinius A, B ir C. Kreivė GV iliustruoja globalios Voronoi diagramos generavimo, naudojant Voro++ biblioteką, laiką. Visų duomenų rinkinių atveju generuojama labai panaši globali Voronoi diagrama, todėl sugaištamas vienodas laikas ir pateikta tik viena kreivė. Atlikti testai parodė, kad lokali Voronoi dekompozicija generuojama daug greičiau negu globali Voronoi diagrama. Nepaisant ilgo laiko, reikalingo patikrinti tinklelio jungčių atitikimus, lokalios Voronoi dekompozicijos metodo vykdymo laikas sudarė 25,0 %, 20,4 % ir 19,1 % globalios Voronoi diagramos generavimo laiko duomenų rinkiniams A, B ir C.

S8 paveiksle pateiktas trijų vizualizavimo metodų kiekybinis palyginimas naudojant du kompiuterius, pavadintus C1 ir C2. S8a, S8b ir S8c paveiveikslai iliustruoja vykdymo laiką naudojant duomenų rinkinius A, B ir C. Kreivės LV, CC ir CU vaizduoja lokalios Voronoi dekompozicijos, celių centrų ir celių kirtimo metodų greitaveiką. Staigų vykdymo laiko augimą ties 100 laiko žingsniu, lėmė greitas nutrūkusių jungčių skaičiaus padidėjimas tuo laiko momentu. Celių kirtimo ir celių centrų metodai naudoja veiksmingą dekompozicijos auginimo laike strategiją. Praėjusiame laiko žingsnyje sugeneruota



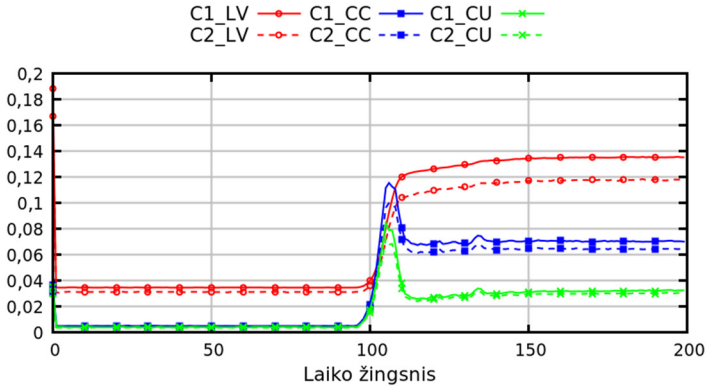
S7 pav. Vykdymo laikas naudojant globalų Voronoi metodą ir lokalų Voronoi dekompozicijos metodą

dekompozicija išsaugoma sekančio laiko žingsnio rezultatams vizualizuoti, todėl vykdymo laikas naudojamas tik dekompozicijos auginimui. Dėl šios priežasties po staigaus šuolio vykdymo laikas sumažėja, nes didžiausi pažeistų regionų plotai buvo padengti lokalia dekompozicija praėjusiuose laiko žingsniuose, o nagrinėjamame laiko žingsnyje nutrūkusių jungčių skaičius ir naujai pažeistų regionų plotas yra pakankamai nedideli. Lokalaus Voronoi dekompozicijos metodo vykdymo laikas išlieka toks pats, net kai plyšys nedidėja, nes Voronoi dekompozicijos celių viršūnių koordinatės kinta kiekviename žingsnyje, o metodas negali išnaudoti praėjusio laiko žingsnio topologijos. Dėl šios priežasties lokalios Voronoi dekompozicijos metodo vykdymo laikas buvo ilgiausiais visų nagrinėjamų duomenų rinkinių atveju.

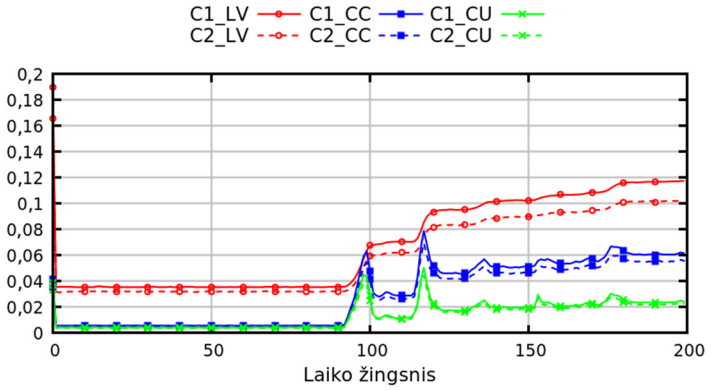
Rezultatai parodė kad celių kirtimo metodas vizualizuoja plyšio paviršių greičiau negu celių centrų metodas. Celių kirtimo metodo vykdymo laikas sudarė 46,2 %, 39,5 % ir 41,8 % celių centrų metodo vykdymo laiko, duomenų rinkiniams A, B ir C. Verta pastebėti, kad vykdymo laikas stipriai priklauso nuo plyšio dydžio ir nutrūkusių jungčių pasiskirstymo. Vykdymo laiko skirtumas naudojant du skirtingus kompiuterius sudarė 12,80 %, 8,24 % ir 6,07 % vykdant, lokalios Voronoi dekompozicijos, celių centrų ir celių kirtimų metodus, atitinkamai.

Ištraukti plyšių paviršiai gali kirsti daleles, taip sumažindami metodų tikslumą. Tik Voronoi dekompozicijos metodu ištrauktas paviršius nekerta dalelių, bet jo negalima taikyti stipriai deformuotose jungčių tinklelio regionuose. Celių centrų metodo sukurtas paviršius kerta daleles tik stipriai deformuoto jungčių tinklelio regionuose. Celių kirtimo metodas nėra tikslus, nes net reguliaraus dalelių išsidėstymo atveju sugeneruotas plyšio paviršius kerta daleles. Paviršių įsiskverbimo į daleles gylis pasirinktas sukurtų metodų tikslumui matuoti.

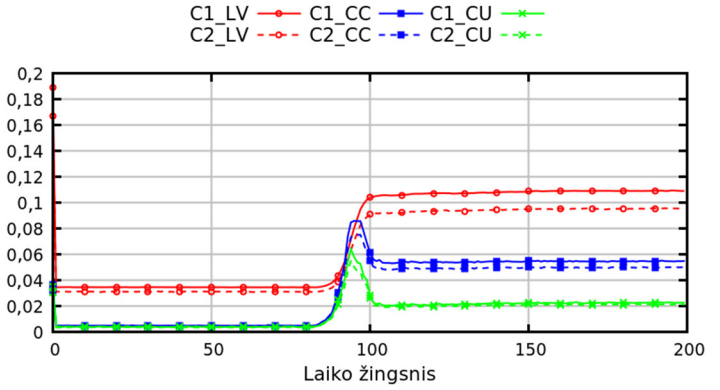
a)



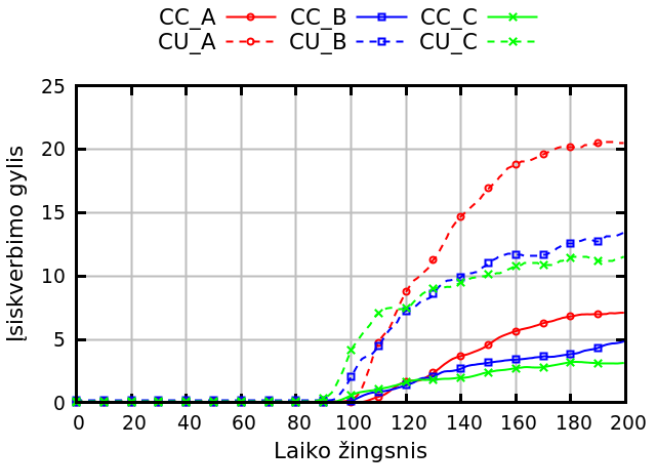
b)



c)



S8 pav. Vizualizavimo metodų vykdymo laikas:
a) duomenų rinkinys A, b) duomenų rinkinys B, c) duomenų rinkinys C



S9 pav. Bendras kirtimų gylis kertant daleles ištrauktu plyšio paviršiumi

S9 paveikslas iliustruoja susumuotą, sugeneruotų paviršių, įsiskverbimo į visas daleles gylį. Kreivės CU_A, CU_B ir CU_C vaizduoja suminį celių kirtimo metodo įsiskverbimo gylį, o kreivės CC_A, CC_B ir CC_C – suminį celių centrų metodo įsiskverbimo gylį. Celių centrų metodo įsiskverbimo gylis sudarė tik 12,6 %, 14,1 % ir 8,9 % celių kirtimo metodo įsiskverbimo gylio duomenų rinkiniams A, B ir C. Didžiausias skirtumas pastebėtas duomenų rinkinyje A, kadangi jame yra didžiausias nutrūkusių jungčių kiekis. Ištraukti plyšio paviršiai giliausiai kerta daleles stipriai deformuoto tinklelio regionuose.

Bendrosios išvados

1. Literatūros apžvalga atskleidė, kad duomenų siuntimas tarp išskirstytųjų vizualizavimo programinės įrangos komponentų sunaudoja didelę vizualizavimo laiko dalį. Plyšių, sklindančių diskrečiųjų dalelių sistemose, vizualizavimas kelia daug iššūkių, nes plyšiai sudaryti iš atskirų dalių su skylėmis.
2. Dalinio duomenų siuntimo realizacijos greitaveikos kiekybinis palyginimas su alternatyvios išteklių tinklo duomenų siuntimo programinės įrangos greitaveika parodė, kad JAX-WS Runtime nagrinėjamus duomenų rinkinius siuntė greičiausiai. Dalinis duomenų rinkinio siuntimas sumažino perduodamų duomenų kiekį, todėl siuntimo laikas sumažėjo iki 7,6 karto, lyginant su pilnų duomenų rinkinio siuntimu.
3. Kiekybinis greitaveikos palyginimas atskleidė, kad lokalios dekompozicijos generuojamos žymiai greičiau negu globalios Voronojaus diagramos. Lokali Voronojaus dekompozicija sunaudojo nuo 19,1 % iki 25,0 % globalios Voronojaus diagramos generavimo kitų autorių sukurta programine įranga laiko.

4. Plyšių vizualizavimo metodų realizacijų greitaveikos palyginimas atskleidė, kad panaudojus celių kirtimo metodą, galima sukonstruoti plyšio paviršių greičiau negu kiti metodai. Celių kirtimo metodo vykdymo laikas sudaro iki 23,4 % celių centrų metodo vykdymo laiko. Celių centrų ir lokalios Voronojaus dekompozicijos metodų vykdymo laiko skirtumas sudarė iki 47,3 % vizualizavimo testo vykdymo laiko.
5. Lokali Voronojaus dekompozicija nekerta dalelių, todėl užtikrinamas didžiausias vizualizavimo tikslumas. Celių centrų metodu sukonstruotų plyšių paviršių tikslumas yra ženkliai didesnis negu celių kirtimo metodu sukonstruotų paviršių. Suminis celių centrų metodo įsiskverbimo gylis sudaro iki 14,1 % celių kirtimo metodo įsiskverbimo gylio.

Annexes[1]

Annex A. Declaration by the author of the thesis
Annex B. The coauthor's agreements to present publications for the dissertation defence
Annex C. Copies of scientific publications by the autor on the topic of the dissertation

[1]The annexes are supplied in the enclosed compact disc

Ruslan PACEVIČ

METHODS AND DISTRIBUTED SOFTWARE
FOR VISUALIZATION OF CRACKS PROPAGATING
IN DISCRETE PARTICLE SYSTEMS

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (07T)

Ruslan PACEVIČ

DISKREČIŲJŲ DALELIŲ SISTEMOSE SKLINDANČIŲ
PLYŠIŲ VIZUALIZAVIMO METODAI IR IŠSKIRSTYTOJI
PROGRAMINĖ ĮRANGA

Daktaro disertacija

Technologijos mokslai,
informatikos inžinerija (07T)

2015 10 19. 10,5 sp. l. Tiražas 20 egz.
Vilniaus Gedimino technikos universiteto
leidykla „Technika“,
Saulėtekio al. 11, 10223 Vilnius,
http://leidykla.vgtu.lt
Spausdino UAB „Biznio mašinų kompanija“,
J. Jasinskio g. 16, 01112 Vilnius