Virtual Reality Methods for Research in the Geosciences

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation
von

Dipl.-Inf. Rolf Westerteiger

Datum der wissenschaftlichen Aussprache: 10.07.2014

Dekan:	Prof. Dr. Klaus Schneider
Vorsitzender des Prüfungsausschusses:	Prof. Dr. Klaus Schneider
1. Berichterstatter:	Prof. Dr. Hans Hagen
2. Berichterstatter:	Prof. Dr. Bernd Hamann
3. Berichterstatter:	Prof. Dr. Christoph Garth

D (386)

Acknowledgements

I would like to use this space to thank the people who have advised and supported me both on a professional and personal level during my research.

First of all, I am grateful for the support of my advisors Andreas Gerndt, Bernd Hamann and Hans Hagen. Andreas Gerndt offered me the chance to stay at the German Aerospace Center in Braunschweig for three and a half years to conduct a majority of my research under his supervision and support. I want to thank Bernd Hamann for his continuous support, not only during my research visits to Davis but also during my time spent in Germany. And last but not least, I would like to thank Hans Hagen for allowing me to conduct my research in a productive international and interdisciplinary environment as a member of the International Research Training Group *"Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling, and Engineering"* (IRTG 1131).

I would like to thank Klaus Gwinner of the German Aerospace Center in Berlin for his collaboration and advice with respect to the geological study of Mars as well as associated technologies and data products.

Considering my work in Davis, California on the topics of fault retro-deformation as well as ocean visualization, I would like to thank Tracy Compton, Eric Cowgill, Gregory Streletz, Oliver Kreylos, Geoffrey Gebbie, Howard J. Spero and Louise Kellogg for providing geoscientific knowledge and especially Tony Bernardin for his collaboration from the computer science side.

Furthermore, I would like to thank all my friends and colleagues for their intellectual and personal support, especially Christian Wagner and Fang Chen.

Last but not least, I want to thank the members of my family for always being there for me, especially during the more stressful and difficult times.

Zusammenfassung

In der vorliegenden Arbeit untersuche ich inwieweit die Geowissenschaften durch Methoden der virtuellen Realität (VR) unterstützt werden können. Dazu habe ich immersive, kollaborative Visualisierungssyteme sowie virtuelle Werkzeuge zur Datenanalyse entwickelt. Die dabei aufgetretenen technischen Herausforderungen lege ich dar und präsentiere geeignete Lösungsansätze.

Zur Visualisierung grosser digitaler Terrain-Modelle (DTMs) in einer immersiven VR-Umgebung werden geeignete Terrain-Rendering Algorithmen benötigt. Zum einen erfordert die realistische Darstellung der Planeten-Krümmung aus grosser Höhe eine sphärische Darstellung der Oberfläche, zum anderen muss dabei stets eine interaktive Bildfrequenz garantiert werden um eine Desorientierung des Benutzers zu vermeiden. Um virtuelle Vermessungwerkzeuge in dieser Umgebung bereitstellen zu können wird zudem eine Datenstruktur benötigt, welche sich sowohl für die Visualisierung als auch zur Berechnung geometrischer Eigenschaften wie etwa Höhenprofile oder Volumina eignet. Um diesen Anforderungen gerecht zu werden, habe ich einen neuartigen Terrain-Rendering Algorithmus entwickelt, welcher auf einer gekachelten Quadtree-Unterteilung der HEALPix-Kugelparametrisierung basiert. Die Effizienz und Bildqualität des Systems untersuche ich anhand eines 500 GiB grossen Datensatzes, welcher die Mars-Oberfläche repräsentiert.

In Anbetracht der aktuellen Entwicklung preisgünstiger, flexibler Fernerkundungsplattformen wie etwa Quadcopter scheint es unausweichlich, dass diesen Systemen eine wichtige Rolle in zukünftigen Katastrophenschutz-Anwendungen zufallen wird. Die zeitnahe Bereitstellung aktueller dreidimensionaler Lageinformationen bei einem Unglücksfall und deren Präsentation in einer VR-Umgebung könnte dabei zur raschen, kollaborativen Entscheidungsfindung beitragen. Bei einer grossen Mehrheit der in der Literatur vorgestellten Terrain-Rendering Algorithm ist jedoch eine aufwendige Vorverarbeitung der Eingangsdaten notwendig, bevor diese präsentiert werden können. Um diese Problematik zu bewältigen, entwickelte ich ein verteiltes Visualisierungs-System, welches diese Daten auf Anfrage in kurzer Zeit verarbeitet. Das System besteht aus einem Rendering-Frontend, welches die dreidimensionale Darstellung

übernimmt, sowie einem Backend, bestehend aus einem kleinen Cluster, welcher für die Datenkonvertierung verantwortlich ist. Dabei kommt ein auf Grafikkarten implementierter Algorithmus zum Einsatz um Fernerkundungsdaten in das HEALPix-Koordinatensystem zu projezieren.

Bei der Untersuchung geologischer Phänomene stellt deren räumliche Vermessung einen wichtigen Arbeitsschritt dar. Um solche Messungen in einer VR-Umgebung zu ermöglichen ist es notwendig, geeignete Eingabegeräte und Interaktionsmetaphern ("virtuelle Werkzeuge") bereitzustellen. Diese Werkzeuge sollten trotz der perspektivischen Wahrnehmung eine exakte räumliche Positionierung ermöglichen sowie entsprechend der verfügbaren Daten-Auflösung akkurate Ergebnisse liefern. Die Interaktivität der Darstellung sollte dabei während des Messprozesses stets beibehalten werden. Ich habe dazu einen Satz virtueller Werkzeuge zur Messung von Höhenprofilen sowie von Volumina entwickelt, welche auf Grundlage der auch zur Visualisierung verwendeten HEALPix-Datenstruktur arbeiten. Zur Selektion von Oberflächenkoordinaten wurde eine strahlbasierte Interaktion implementiert. Der eigentliche Messvorgang ist als Hintergrundprozess ausgelegt, welcher die Datenbasis in der höchsten verfügbaren Auflösung verwendet um genaue Messergebnisse zu garantieren.

Geologische Störungen sind Brüche in der Erdkruste entlang derer ein Versatz von Gestein beobachtet wird. Die Bestimmung der Richtung und Distanz solcher Verschiebungen ist ein wichtiger Aspekt geologischer Forschung. Traditionell werden hierfür (digitalisierte) Landkarten in Draufsicht herangezogen. Diese werden, etwa mit Bildverabeitungssoftware, entlang der vermuteten Störungslinie aufgetrennt und die beiden Fragmente per Parallelverschiebung gegeninander versetzt mit dem Ziel die Kontinuität der durch die Störung versetzten Oberflächenmerkmale wiederherzustellen. Die dazu notwendige Verschiebung stellt schliesslich eine Hypothese für die Gesamtverschiebung entlang der Störung dar. In der vorliegenden Arbeit zeige ich, dass eine solche Herangehensweise durch die Einschränkung auf eine senkrechte Ansicht zu einer Akzeptanz fehlerhafter Hypothesen führen kann, da sie keine Rücksicht auf die dreidimensionale Struktur der Topographie nimmt. Zur Bewältigung dieses Problems entwickelte ich einen Deformationsalgorithmus, welcher es erlaubt innerhalb einer dreidimensionalen Terrain-Visualisierung eine Störungslinie einzuze-

ichnen und die Landschaft entlang der Linie interaktiv zu deformieren. Die dreidimensionale Visualisierung erlaubt dabei die Untersuchung einer hypothetischen Rekonstruktion aus beliebigen Blickwinkeln, wodurch die Warhscheinlichkeit einer Fehlinterpretation verringert wird. Interaktivität in Bezug auf alle Parameter wird durch eine Implementation auf der Grafikkarte gewährleistet.

Die Wissenschaft der Paläozeanographie beschäftigt sich mit der prähistorischen Entwicklung der Ozeane. Eine Schlüsselrolle fällt dabei den sogenannten Kernbohrungsexperimenten zu, welche stichprobenartig die Zusammensetzung des Meeresgrundes erfassen. Der schichtartige Aufbau der gewonnenen Kernproben korreliert mit der zeitlichen Entwicklung der Sedimentkonzentrationen des Ozeans am Ort der Entnahme. Aus diesen Daten wiederrum die zeitliche Entwicklung der Ozeanströmung zu rekonstruieren stellt jedoch ein schwieriges inverses Problem dar. Um die Entwickung numerischer Modelle der Ozeanströmung zu unterstützen habe ich ein Visualisierungssystem entwickelt, welches sowohl Primärdaten aus Kernbohrexperimenten als auch die daraus durch numerische Simulation vorhergesagten zeitabhängigen Strömungsfelder simultan darstellt. Die Erde erscheint dabei als Globus in einer VR-Umgebung, wobei die Zusammensetzung von Bohrkernen unter Verwendung einer Billboard-Technik visualisiert wird während das Strömungsfeldes durch animierte Line-Integral-Convolution (LIC) dargestellt wird. Zuätzlich können interaktiv virtuelle Partikel in das Strömungsfeld eingegeben werden um Transportwege und deren Korrelation mit den gemessenen Sedimentablagerungen zu untersuchen.

Abstract

In the presented work, I evaluate if and how Virtual Reality (VR) technologies can be used to support researchers working in the geosciences by providing immersive, collaborative visualization systems as well as virtual tools for data analysis. Technical challenges encountered in the development of theses systems are identified and solutions for these are provided.

To enable geologists to explore large digital terrain models (DTMs) in an immersive, explorative fashion within a VR environment, a suitable terrain rendering algorithm is required. For realistic perception of planetary curvature at large viewer altitudes, spherical rendering of the surface is necessary. Furthermore, rendering must sustain interactive frame rates of about 30 frames per second to avoid sensory confusion of the user. At the same time, the data structures used for visualization should also be suitable for efficiently computing spatial properties such as height profiles or volumes in order to implement virtual analysis tools. To address these requirements, I have developed a novel terrain rendering algorithm based on tiled quadtree hierarchies using the HEALPix parametrization of a sphere. For evaluation purposes, the system is applied to a 500 GiB dataset representing the surface of Mars.

Considering the current development of inexpensive remote surveillance equipment such as quadcopters, it seems inevitable that these devices will play a major role in future disaster management applications. Virtual reality installations in disaster management headquarters which provide an immersive visualization of near-live, three-dimensional situational data could then be a valuable asset for rapid, collaborative decision making. Most terrain visualization algorithms, however, require a computationally expensive pre-processing step to construct a terrain database. To address this problem, I present an on-the-fly pre-processing system for cartographic data. The system consists of a frontend for rendering and interaction as well as a distributed processing backend executing on a small cluster which produces tiled data in the format required by the frontend on demand. The backend employs a CUDA based algorithm on graphics cards to perform efficient conversion from cartographic standard projections to the HEALPix-based grid used by the frontend.

Measurement of spatial properties is an important step in quantifying geological phenomena. When performing these tasks in a VR environment, a suitable input device and abstraction for the interaction (a "virtual tool") must be provided. This tool should enable the user to precisely select the location of the measurement even under a perspective projection. Furthermore, the measurement process should be accurate to the resolution of the data available and should not have a large impact on the frame rate in order to not violate interactivity requirements. I have implemented virtual tools based on the HEALPix data structure for measurement of height profiles as well as volumes. For interaction, a ray-based picking metaphor was employed, using a virtual selection ray extending from the user's hand holding a VR interaction device. To provide maximum accuracy, the algorithms access the quad-tree terrain database at the highest available resolution level while at the same time maintaining interactivity in rendering.

Geological faults are cracks in the earth's crust along which a differential movement of rock volumes can be observed. Quantifying the direction and magnitude of such translations is an essential requirement in understanding earth's geological history. For this purpose, geologists traditionally use maps in top-down projection which are cut (e.g. using image editing software) along the suspected fault trace. The two resulting pieces of the map are then translated in parallel against each other until surface features which have been cut by the fault motion come back into alignment. The amount of translation applied is then used as a hypothesis for the magnitude of the fault action. In the scope of this work it is shown, however, that performing this study in a top-down perspective can lead to the acceptance of faulty reconstructions, since the three-dimensional structure of topography is not considered. To address this problem, I present a novel terrain deformation algorithm which allows the user to trace a fault line directly within a 3D terrain visualization system and interactively deform the terrain model while inspecting the resulting reconstruction from arbitrary perspectives. I demonstrate that the application of 3D visualization allows for a more informed interpretation of fault reconstruction hypotheses. The algorithm is implemented on graphics cards and performs real-time geometric deformation of the terrain model, guaranteeing interactivity with respect to all parameters.

Paleoceanography is the study of the prehistoric evolution of the ocean. One of the key data sources used in this research are coring experiments which provide point samples of layered sediment depositions at the ocean floor. The samples obtained in these experiments document the time-varying sediment concentrations within the ocean water at the point of measurement. The task of recovering the ocean flow patterns based on these deposition records is a challenging inverse numerical problem, however. To support domain scientists working on this problem, I have developed a VR visualization tool to aid in the verification of model parameters by providing simultaneous visualization of experimental data from coring as well as the resulting predicted flow field obtained from numerical simulation. Earth is visualized as a globe in the VR environment with coring data being presented using a billboard rendering technique while the time-variant flow field is indicated using Line-Integral-Convolution (LIC). To study individual sediment transport pathways and their correlation with the depositional record, interactive particle injection and real-time advection is supported.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work represents my research on geovisualization in virtual reality environments with the purpose of supporting the geosciences. The software solutions I have developed for this purpose include a terrain visualization system with tools for navigation and spatial analysis as well as a software package for visualizing and exploring time-variant three-dimensional prehistoric ocean flow models. My contributions as detailed in the following chapters are:

- Chapter 2 demonstrates a novel terrain rendering system based on the spherical HEALPix coordinate system, which subdivides the surface of a sphere into 12 curvilinear patches. My system performs coordinate conversion between HEALPix and euclidean coordinates in realtime during rendering on the graphics processor (GPU). Compared to other approaches to spherical terrain rendering, this solution requires a significantly lower memory footprint and a lower implementation complexity by performing most computations in a 2D domain.

- Chapter 3 is concerned with accelerating the preprocessing of terrain data for interactive visualization, which is anticipated to become a time-critical part of the visualization pipeline in future crisis management applications where live 3D data is collected by small autonomous drones to support rescue efforts. To accelerate the processing of geo-data, I have developed a distributed, GPU

based system which consists of an interactive terrain visualization on a frontend workstation as well as a number of backend processing nodes. Instead of loading terrain tiles from disk, the frontend requests the tile content from the backend which converts the source data (georeferenced raster maps) to the HEALPix coordinate system on the fly, enabling rapid 3D visualization of unprocessed maps such as terrain models or imagery.

- Chapter 4 extends the terrain rendering system described in chapter 2 by introducing virtual tools for spatial analysis. An intuitive interface is provided to measure height profiles and volumes of topographical features such as canyons, mountains or craters. I demonstrate how the HEALPix-based representation of Digital Terrain Models permits efficient computation of these geometric properties using the highest resolution data available while still maintaining interactivity in rendering which is important to maintain immersion in a virtual reality environment.

- Chapter 5 introduces a 3D system for retro-deformation of geological faults within an interactive terrain visualization. Faults are discontinuities in the planetary crust along which a differential movement of rock volumes can be observed. Geologists are interested in measuring the direction and magnitude of these translations. For this purpose, they take advantage of the fact that geographical features such as riverbeds are cut and displaced along with the crust when intersected by a fault. The traditional workflow to determine fault offsets begins with a digital top-down map of the affected region, which is subsequently cut along the (suspected) fault line using an image editor. The two resulting map fragments are then shifted with respect to each other until the offset features come back into alignment. The system I have developed allows scientists to perform this retro-deformation process interactively within a 3D terrain visualization. Using this system, I demonstrate that the established 2D approach can lead to faulty reconstructions because it does not consider the 3D implications of the deformation.

- Chapter 6 describes a system for the visualization of three-dimensional, time-dependent prehistoric ocean flow models. These flow models are obtained by

numeric simulation based on present-day data such as sediment depositions at the ocean floor. To support oceanographers in solving this challenging inverse problem and verify the simulated flow models, I have integrated several visualization techniques to correlate the primary data from drill experiments with the resulting numerical model. Using a spherical virtual globe as a context geometry, Line-Integral Convolution (LIC) and interactive particle advection are used to visualize the flow model while a billboard technique using bar charts indicates sediment concentration at documented drill sites. The visualization system is implemented in a CAVE environment in order to promote collaborative investigation of the data.

1.1 Motivation

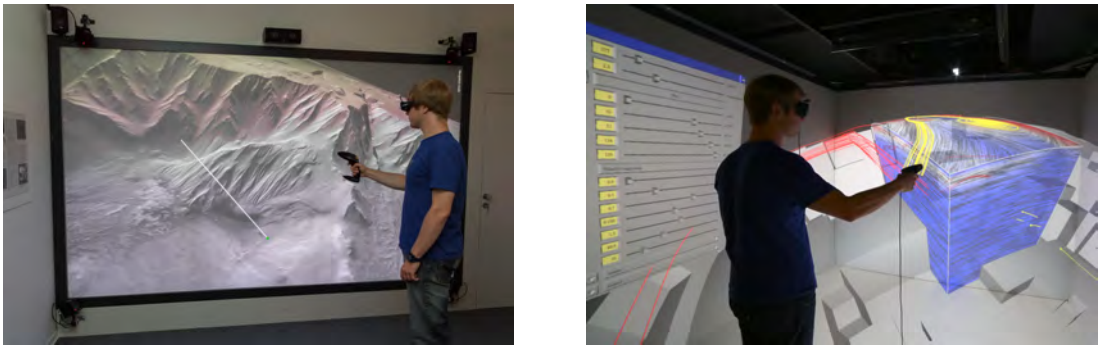
Flat maps are the medium by which geoscientific data is traditionally presented. Geographic information systems have simplified the creation and interactive interrogation of these maps but are still very much focussed on presenting data in a flat projection. This top-down perspective clearly has advantages because it can be rendered efficiently, avoids occlusion of topography (with the exception of overhanging walls, caves, etc.) and, by careful choice of a map projection, allows for the preservation of a certain subset of geometric properties.

A major disadvantage of this approach is that even though domain scientists are trained in interpreting these representations, they are still an abstraction and do not correspond to the intuitive perception of a person standing on the ground and surveying a site.

The research question I would like to address with this work is whether interactive, three-dimensional visualizations in virtual reality environments can be beneficial in supporting the interpretation and analysis of geoscientific data.

In the following, the concepts of virtual reality, geovisualization, fault retro-deformation and flow visualization will be introduced briefly, focussing on those aspects which are relevant within the scope of this work.

1.2 Virtual reality



(a) Powerwall at German Aerospace Center (b) CAVE at UC Davis

Figure 1.1: VR systems used in this work

Virtual reality (VR) research is concerned with the development of hardware and software interfaces that provide an immersive perception of a computer-generated 3D scene, blurring the distinction between reality and artificial imagery. The degree of immersion that can be achieved with a specific VR setup depends on the display and interaction devices provided by the environment.

As far as display systems are considered, stereoscopic visualization is frequently considered a minimum requirement for VR, as exemplified by a desktop computer with a suitable monitor and shutter glasses. At the other end of the spectrum one finds the so-called CAVE (Cave Automatic Virtual Environment), which is a room where each wall is a projection surface displaying a stereoscopic image. By covering most or all of the user's visual field, a high degree of immersion can be achieved.

For my VR research concerning terrain visualization at the German Aerospace Center, a so-called stereoscopic Powerwall was used, which is a high-resolution display offering stereoscopic visualization (using shutter glasses) as well as head-tracking. Head-tracking allows the visualization system to adapt the viewing frustum of the virtual camera to the position of the observer, which produces a parallax effect when the user moves. This improves immersion by increasing 3D perception beyond what is possible with stereoscopic vision alone. The ocean flow visualization was evaluated within a 4-sided CAVE environment provided by the KeckCAVES at UC

Davis (figure 1.1). Note that due to the use of modern VR software toolkits, these applications can be readily ported between different VR hardware environments.

Considering interaction devices, at the low end of the spectrum one can find the 3D space-mouse for desktop systems. More immersive interaction is possible with handheld 6-degrees-of-freedom (6DOF) manipulators, which can be used as an extension of the user's hand to point at or grab objects in the VR environment. At the forefront of research are so-called markerless tracking systems such as the Kinect camera, which support interaction by gesture interpretation, without requiring the user to wear or carry any equipment. For the VR applications in this work, I used a so-called flystick, which is a 6DOF manipulator with a set of buttons, used here for navigation, selecting surface points for spatial analysis or injecting particles into the ocean flow field.

1.3 Geovisualization

The geosciences are concerned with understanding phenomena such as geological processes, atmospheric systems or ocean currents which pertain to Earth or other planetary bodies. Geovisualization is the application of scientific visualization and information visualization techniques to support the interpretation of the geospatial data gathered in these fields.

Historically, hand-drawn maps were the primary medium used for this purpose. Map creation has been substantially simplified through the development of remote sensing technology as well as software packages for map creation and interrogation, so-called Geographic Information Systems (GIS).

Interpretation of maps is complicated by the fact that they project a spherical, three-dimensional surface into a two-dimensional plane. Due to this projection, not all geometric relationships which exist in reality can be preserved in the resulting map. For this reason, projections are carefully chosen in order to preserve those geometric properties which are considered important for the specific application domain, such as distances, angles and areas in order to make measurements of these properties on the map consistent with reality.

1.3.1 Digital map representations

Digital maps can have different representations depending on the physical quantity being encoded, the mode of acquisition and the intended application. If a map stores elevation of the topography it is referred to as a Digital Elevation Model (DEM). The two most common representations for digital maps are triangulated irregular networks (TINs) and raster maps.

TINs are two-dimensional meshes consisting of non-overlapping triangles. The vertices store both a geographic location as well as the value of the measured quantity at that location. The triangulation then induces a (linear) interpolation of that quantity in the space between the vertices. TINs are most commonly used to represent DEMs, storing an elevation value for each vertex. In this case a three-dimensional triangle mesh of the surface can be readily obtained by displacing the 2D mesh along the third dimension according to the elevation values.

This TIN representation of DEMs arises naturally from input given as point clouds, from which a well-behaved triangle mesh can be computed using Delaunay triangulation. Point clouds, in turn, are the native representation for data gathered by remote sensing techniques such as (terrestrial) laser scanning, which exhibit varying sample density due to the effects of shadowing and perspective distortion.

An advantage of TIN-DEMs is that they can be simplified to represent areas with low curvature with a lower sample density, reducing the required storage space. A disadvantage is that 2D coordinates for each vertex as well as the topology of the triangulation have to be stored explicitly.

Besides TINs, another common representation are so called raster maps. A raster map consists of a two-dimensional array of values storing the quantity being encoded on a uniform grid. Raster maps are accompanied by a georeference, which consists of an affine transformation represented by a homogenous matrix as well as coefficients for a standard map projection, which is (in the general case) a nonlinear mapping between the 2D plane and geographic coordinates. The georeference assigns a geographic coordinate to each cell of the sample grid.

Raster maps have the advantage that only the sample values need to be stored for each grid location, as the geographic location is given implicitly by the storage index

of the sample as well as the georeference. The disadvantage is that, as opposed to TINs, sample density can not be adjusted adaptively.

Imagery is almost always represented using raster maps, for example by storing a red, green, blue tuple for each grid cell in the case of real color imagery. Raster representations are a natural choice in this case because imaging techniques such as aerial photography or satellite-mounted line-scanners in push broom configuration exhibit relatively uniform sample coverage.

Note that nowadays DEMs are frequently acquired using a process called photogrammetry, in which imagery of the same location obtained from different angles is correlated to obtain $3D$ models. The required imagery can be gathered e.g. by a satellite in a single pass over the location if multiple camera systems are mounted at different angles with respect to the ground.

1.3.2 Geographic information systems

Geographic information systems (GIS) are software packages which support the creation, visualization and analysis of geographical data which is most frequently represented using maps in top-down perspective.

A common GIS application is the creation of so-called vector maps which use 2D geometry to describe features. These can be created e.g. by direct land surveying or by by using aerial imagery as reference. Linear features such as streets or riverbeds can be described using piecewise linear curves (poly-lines), while features with two-dimensional extent can be represented using polygons which trace their boundaries.

Additionally, GIS frequently provide spatial analysis tools for measuring geometric quantities such as distance, area or height profiles along a line. The terrain visualization system developed within the scope of this work implements a subset of spatial analysis tools in order to study how these can be integrated into a VR environment (see chapter 4).

1.3.3 Terrain rendering

Three-dimensional terrain rendering is a well-established field of research. The goal is to produce a perspective rendering of the three-dimensional topography described by a DEM data set, often overlaid (textured) with associated imagery. Maintaining interactive frame rates when rendering large terrain models is one of the core challenges in this field.

The majority of modern terrain rendering algorithms use GPU acceleration to rasterize geometry, while some use CPU-based approaches such as ray-casting. The architectural limitation faced by these approaches is that data which is not located within primary memory (GPU or host RAM) can only be accessed with very large latencies. Furthermore, data sets are often orders of magnitudes larger than the capacity of the RAM, making it impossible to rasterize the entire data content at interactive frame rates.

A solution to this problem can be found when considering two properties which are given in the majority of 3D visualization pipelines:

1. **Perspective projection**: In accordance with human vision, elements of the 3D scene which are located closer to the observer position appear at a greater magnification than more distant parts of the scene.

2. **Raster output devices**: The majority of output devices such as LCD monitors and projectors use a 2D pixel raster with finite spatial resolution to display the final image. This implies that details in scene geometry smaller than a certain scale are not noticeable due to the discretization involved in rasterization.

From 2 it follows that geometric detail with a screen-space resolution higher than the pixel raster can be considered redundant as it can not be resolved by the output device. However, due to 1 the screen-space resolution of geometry depends on it's distance to the observer.

These observations motivate so-called Level-of-Detail rendering approaches, which dynamically adapt geometric detail based on observer distance. This adaptation

is driven by a metric which selects a resolution level for each (visible) part of the terrain. Conservative approximations can be used to guarantee that the screen-space geometric resolution matches or exceeds the display resolution to ensure faithful reproduction of the terrain model without incurring excessive rendering workload.

For performance reasons, terrain models are usually represented using multi-resolution data structures on secondary storage. Two classes can be distinguished: regular and adaptive representations. Regular representations are based on hierarchies of regular grids, for example given by quad-tree subdivision. Regular grids containing a fixed number of samples, as opposed to individual samples, are usually stored within each node of such a tree for performance reasons. These grids are referred to as tiles.

Regular schemes have the advantage that point coordinates do not have to be stored as they are implied by the storage order of elevation samples. Adaptive meshes, on the other hand, triangulate the surface based on an error metric, which requires a smaller amount of sample points in flat areas but necessitates storage of 3D coordinates for each point. Compression techniques are often applied to reduce this storage overhead.

Modern Level-of-Detail rendering approaches scale well enough to allow exploration of large, high-resolution terrain models all the way from global to sub-meter scale. At low observer altitudes, a planar approximation of the planetary surface is sufficient for this purpose. To obtain a realistic reproduction of topography at global scales (for high observer altitudes), however, planetary curvature has to be taken into account.

Spherical terrain rendering algorithms address this problem by rendering topography as a deformation of the surface of a sphere or ellipsoid. This requires a projection which establishes a mapping between the two-dimensional parameter space of the algorithm and points on the sphere.

Geographical coordinates (latitude, longitude) are frequently used for this purpose. Advantages are efficient conversion to and from euclidean coordinates as well as interoperability with existing systems due to the fact that the majority of map data is referenced in geographical coordinates. A major disadvantages is that a uniform sampling in the geographical coordinate space does not translate to a uniform sam-

pling of the sphere. This effect can be observed on a globe with grid lines: Lines of constant longitude converge towards the poles, which results in long, skinny grid cells. This behaviour hints at another problem: At the location of the poles ($\pm 90°$ latitude), the projection exhibits so-called singularities. At these points the longitude component of the geographical coordinate becomes undefined. Due to these issues, terrain rendering algorithms based on geographical coordinates often exhibit sampling and/or rendering artifacts in the vicinity of the poles.

It can be shown that any continuous mapping from the 2D plane to the surface of a sphere suffers from such singularities. To avoid these singularities, some approaches subdivide the surface into a set of patches. Hierarchical sampling grids are then defined for each patch by applying a subdivision scheme recursively. For example, a platonic solid can be used as the starting point and the subdivision rule refines each face of the solid by inserting it's midpoint and extruding it to the surface of the sphere. A disadvantage of defining the coordinate system through subdivision is that there is no explicit closed formula relating points in parameter space to their geographical coordinates, which complicates rendering and querying the database for measurements.

In chapter 2, I will present a novel terrain rendering algorithm based on the HEALPix [GHB+05] grid. In the HEALPix scheme, which was originally developed to represent full-sky astronomical observations, the surface of a sphere is decomposed into a set of 12 curvilinear patches. Each patch is then hierarchically subdivided using a quadtree to form a Level-of-Detail data structure suitable for interactive terrain rendering. For performance reason, tree nodes store tiles of 255×255 elevation or color samples which are uploaded to GPU memory. During rendering, a custom GPU shader applies the HEALPix projection formula to convert tile coordinates into euclidean space to obtain a spherical visualization of the topography.

1.4 Fault retro-deformation

Faults are planar fractures in the planetary crust along which a differential movement of rock volumes can be observed. Two fundamental types of faults are distinguished:

strike-slip faults with mostly horizontal movement and dip-slip faults with predominantly vertical movement. In order to help understand the historical evolution of topography, geologists are interested in quantifying the orientation and amount of translation along a fault.

Traditionally, these translations are estimated by measuring, on a map, the offsets between surface features which have been intersected and displaced by the fault. In chapter 5, I demonstrate that these reconstructions based on a top-down map perspective can lead to incorrect hypotheses and propose a retro-deformation system which augments an existing visualization package by providing tools to perform these reconstructions in a true 3D environment to avoid false interpretations. The system was applied to a section of the San Andreas Fault (a strike-slip fault) in California as well as to a Graben structure (a pair of dip-slip faults) on Mars.

1.5 Flow visualization

Flow visualization is another important field of research in scientific visualization. Understanding complex flow phenomena has become a necessity in many engineering disciplines such as aircraft and car design. Modern flow simulations are often time-variant and three-dimensional, posing additional challenges for visualizations with respect to avoiding visual clutter and enabling efficient, interactive exploration.

Within the geosciences, there are several areas of research which are concerned with flow phenomena, such as climate research, plate tectonics and oceanography. Palaeoceanography, in particular, considers the historic evolution of ocean flow patterns. Scientists perform drill experiments on the ocean floor in order to extract so-called core samples, which record the accumulated deposition of sediment at their location. These deposits are the result of sediment material settling down to the ocean flow after being transported along with the ocean currents, potentially for a long time.

In order to reconstruct the configuration of prehistoric ocean currents, scientists analyze these core samples and apply inverse numerical methods in order to obtain a time-variant flow field which is consistent with these observations. To support

the verification of these flow models against the gathered data, I have developed a virtual reality based visualization system.

The system can be used both in desktop and CAVE environments and enables exploration of time-variant 3D flow data parametrized on the globe. Established methods of flow visualization such as cut planes with animated line-integral-convolution (LIC) as well as interactive particle tracing are supported and original sediment record data can be overlaid for reference. VCR-style controls (play, pause, forward/backward) are provided to explore the data along the time axis.

1.6 Related work

The following section documents how my research is related to other scientific publications in the fields of terrain visualization, pre-processing of terrain data, spatial analysis in VR environments, fault retro-deformation and visualization of ocean flow models.

1.6.1 Terrain visualization

Interactive terrain rendering is a well-established area of research. As computing paradigms shifted over the decades, algorithms have adapted to exploit the characteristics of the hardware of the day. Early approaches such as ROAM [DWS+97], introduced by Duchaineau et al., perform triangle-level adaptation of the rendered geometry on the CPU to minimize geometric complexity before rasterization. These strategies have become prohibitively expensive on modern hardware which provides very high rasterization bandwidth using GPUs but suffers from relatively slow CPU performance and bus bandwidths.

Due to these developments, more recent solutions try to optimize batching of the geometry by treating blocks of geometry as opaque entities which are rasterized as a whole. Levenburg's CABTT algorithm [Lev02] extended classical triangle bin-tree algorithms by processing batches of geometry bounded by a triangular patch instead

of individual triangles. These patches are uploaded to GPU memory and re-used for multiple frames to reduce CPU workload and bus traffic.

The P-BDAM algorithm [CGG+03] of Cignoni et al. uses a similar type of batching but uses adaptive triangulations within each patch, which are precomputed off-line. Spherical rendering is realized by using curved patches. Distributed pre-processing is enabled by subdividing the surface of the sphere into block that can be processed independently.

The Geometry Clipmaps [LH04] scheme by Losasso and Hoppe uses a set of nested regular grids which decrease in resolution with increasing distance from the viewer. Temporal coherence in viewer movement is exploited by introducing a toroidal indexing schemes which allows to minimize the per-frame cost of updates on the GPU as new data is being loaded.

Spherical Clipmaps [CH06] extend this scheme to spherical rendering by representing the planet's hemisphere which faces the viewer using a set of circular rings. However, due to low accuracy of the $\tan^{-1}$ function on the GPU, this method can produce visible cracks in the final triangulation. Furthermore, compared to the original scheme, vertices are no longer centered on actual height field samples. Due to the additional interpolation required, the data is essentially low-pass filtered when rendering.

Crusta [BCK+11] is a terrain rendering framework which uses a 30-sided polyhedron as base geometry, whose faces are recursively subdivided using a quadtree. It is therefore conceptually similar to my approach as presented in chapter 2, however due to the purely recursive formulation of this geometric construction there is no analytical mapping between parametric and Euclidean coordinates. This prohibits performing this projection on the GPU and requires computation and storage of Euclidean vertex coordinates for individual tiles. Storing these 3-tuples for each sample at runtime produces memory overhead compared to my approach which only requires the scalar height, as the parametric coordinates are implicit and evaluated on the GPU.

Google Earth is a popular tool for exploring the surface of Earth using data streamed over the internet, which performs well even over low-bandwidth internet connections

and provides a high degree of interactivity. However, the system suffers from degenerate triangulations close to the north and south pole, leading to visible artifacts.

1.6.2 Pre-processing of raster maps

Planetary-Scale Terrain Composition [KLJ+09] by Kooima et al. is an unconventional approach for real-time composition of unregistered raster maps which defers traditional resampling to the rendering phase. A mipmap representation of the data being loaded is required, however. During rendering, sphere geometry is generated on the fly using iterative subdivision of a base icosahedron. The maximum depth of this subdivision is determined by a LOD heuristic, depending on viewer position. Finally, vertices are displaced along their normals according to height values stored in the input datasets, which are uploaded to the GPU as textures.

Vertices of the triangle mesh being subdivided contain spherical coordinates as well as normals. Depending on whether an intersecting data set uses spherical or polar projection, either spherical coordinates or normals are used as texture coordinates to sample a mipmap representation of the original raster image. Mesh vertices must be displaced and colored accordingly. This texture look-up is efficient, as texture coordinates have already been computed at that point. The computation of texture coordinates, however, is computationally expensive. Due to numerical instability, the spherical coordinates can not be computed in a closed form but have to be maintained as the subdivision progresses. To generate spherical coordinates for newly interpolated vertices, the haversine geodesic midpoint method is used. This method is computationally expensive, especially considering that it has to be evaluated once for every vertex in every frame.

My online pre-processing system presented in chapter 3 is similar in that it resamples any given data onto a singularity-free grid. Instead of using an implicit subdivision scheme, we apply the HEALPix [GHB+05] parametrization. HEALPix has an explicit projection formula which is computationally cheap to evaluate and maps between parameter space and spherical coordinates. This allows for independent immediate projection of grid vertex positions at any refinement level, enabling highly parallel GPU based resampling. In my online pre-processing approach, this

resampling is only performed once as tiles are being generated. These tiles are then cached by the front-end visualization system.

Lambers et al. [LK10] have presented a system to generate view-dependent geometry on the fly to allow for visualization of fully dynamic datasets. The GPU-based per-frame generation of this triangulation is computationally expensive, however, and does not guarantee interactivity. Treib et al. [TRAW12] use a compressed wavelet representation for terrain data which is visualized at interactive rates using GPU-based ray-casting. The chosen representation allows for efficient real-time editing of terrain. A quad-tree topology is used for the multi-resolution data structure, however tree nodes contain only the (compressed) differences to their parent nodes as opposed to storing absolute height values. When the terrain is modified at a given resolution level, changes are propagated through the quadtree. To populate coarse resolution levels, the affected subtree is recomputed using a bottom-up construction process starting from the modified tiles. To propagate downwards in the tree, editing operations are recorded and applied on-the-fly to finer nodes as they are loaded into GPU memory. In my system, data being inserted into the tree is always represented as leaf nodes, hence only upward propagation is necessary. This is likewise achieved by updating the tree using iterative downsampling, starting from affected leaf nodes.

GPU algorithms have also been used for pre-processing remote sensing data. Thomas et al. described a system [TKR+08] for GPU-based orthorectification, which is a process to remove perspective distortions in imagery to produce a top-down view of uniform scale. In their approach, imagery obtained from an oblique perspective is projected in real-time onto an existing DEM to produce a corrected orthophoto using a technique related to shadow mapping. In the work presented here it is assumed that the input data is already available as orthophotos. In an actual crisis management scenario, one could imagine coupling both approaches to obtain a pipeline for immediate visualization of live remote sensing data.

1.6.3 Spatial analysis in VR

Publications about spatial analysis in VR include a study by Kreylos et al. [KBB+06], which examines the efficacy of performing scientific analyses within VR environ-

ments. Cartography is given as an example, and from preliminary results, the authors conclude that mapping performance in their VR environment is higher compared to performing the same task within in a desktop GIS. Navigation within the terrain data set is performed by grabbing and manipulating the terrain using a 6DOF manipulator.

In the work presented here I have used a different navigation approach by moving the camera instead. This allows for a higher degree of immersion, as it is more similar to the user actually being present at the target site.

Not all data structures used by spherical terrain rendering systems are capable of supporting efficient spatial analysis queries. For example, the Crusta system by Bernardin et al. [BCK+11] and the Planetary Scale Composition approach by Kooima et al. [KLJ+09] use implicitly defined coordinate systems. A random point query against these data structures would require testing the boundary of each traversed node within the subdivision hierarchy for intersection with the given point.

Our software extends my previous work on terrain visualization [WGH11], which used a data structure based on the HEALPix coordinate system by Górski et al. to provide spherical rendering while avoiding artifacts due to coordinate singularities. At the same time, this data structure is well suited for interactive spatial analysis, as it provides a closed projection formula relating geographic points to parametric (database) coordinates.

1.6.4 Fault retro-deformation

The method traditionally used to visualize retro-deformation is based on cutting and displacing 2D maps (orthoimages) according to measured offsets between displaced surface markers, as exemplified by Frankel et al. [FBD+07]. A disadvantage of the 2D representation is that it can lead to ambiguities in the interpretation of such reconstructions as is demonstrated in section 5.5.

Zielke and Arrowsmith [ZA12] presented the LaDiCaoz tool for semi-automatic lateral (strike-slip) fault offset computation based on digital elevation maps (DEMs). In this system the user draws a fault trace onto a hillshaded image in plan view.

The system then samples the elevation on both sides of the fault trace to produce a pair of height profiles. Keeping one side fixed, the algorithm laterally displaces the other profile to find the offset which provides the best match according to a distance metric. While this automation can reduce the time to find plausible fault offsets, the system supports only lateral offsets and provides only limited means for visual confirmation of such hypotheses. Results can only be visualized in plan view, which has the same drawbacks as the approach described previously. Furthermore, the authors report that because the algorithm requires all input data to be represented in memory, input DEMs can not be larger than about 10^8 grid points, which translates to a file size of 200 MiB for a typical 16 bit per sample representation. Larger files can be loaded but have to be downsampled in the process. In my approach, the underlying terrain renderer implements Level-of-Detail rendering and data streaming to overcome this limitation. This allows users performing a reconstruction to watch nearby marker sites for alignment at native resolution even for very large (e.g., $>$ 60 GiB) datasets.

An interesting geometric construction to describe the structural deformations due to fault movements was presented by Egan et al. [EKB+99]. However, the method has only been applied to small models and is not practical for deforming large DEM meshes interactively.

3D GeoModeller is a software package for recovering the subsurface fault configuration based on primary geological observations such as drill hole experiments, seismic and gravimetric data. Using geophysical inverse theory and a human-in-the-loop approach, the software attempts to reconstruct the most probable fault configuration consistent with the sparse observations that are available. The system focuses on recovering the present day subsurface fault geometry and does not consider the time evolution of fault slip, however. The functionality my solution provides to recover fault geometries from their surface expression could potentially be used to constrain and refine geological models constructed with this software.

Numerical simulations of faulting mechanics, which is a frictional contact problem, have been explored extensively [Bir78, XMM07]. Based on volume mesh representations of fault planes and rock layers (strata), static finite element (FE) models are able to simulate stress distribution in the Earth's crust. In recent years, these

approaches have been extended to simulate fault motion by using dynamic FE models [LL06]. However, while FE methods are useful to validate individual hypotheses about fault geometry or material composition in the subsurface, they are too computationally intensive for interactive simulation of faulting in 3D space.

Van Aalsburg et al. [VAYK+10] have demonstrated a system to interactively edit fault maps in a virtual reality (VR) environment. Their work focuses on using VR methods to improve perception of surface features and enable users to create better maps of large fault networks, which can then be used as input to FEM based solvers. While my system also provides functionality to trace fault lines directly on top of a DEM visualization, our goal is interactively simulating the retro-deformation of these faults to measure and verify offset parameters.

Mesh deformation techniques which use control points or handles to control the deformation of an existing object are well established in geometric modeling literature [SP86, Coq90, BK05]. A common goal in many of these approaches is to localize the influence of control points to provide intuitive control about the deformation. The design of my kinematic model is based on a similar notion, namely that trajectories of points close to a fault line segment should be tangential to the segment.

Von Funk et al. [vFTS06] presented an interactive mesh deformation technique that transforms the vertices of a triangle mesh by GPU-based pathline integration along a synthetic vector field that can be manipulated using different modeling metaphors. My displacement strategy can likewise be cast as vector field integration on an implicit field which is constant within the influence zones of the individual fault line segments.

Correa et al. [CS07] have demonstrated an interesting approach to rendering deformed meshes using a signed distance map representation and ray-casting. The necessary warping function for fault-retro-deformation could certainly be computed procedurally, but an implementation would essentially imply development of an efficient ray-casting based terrain render. This is complicated by the fact that the actual path of the rays is only computed on the GPU, which would require random access to the underlying level-of-detail representation.

Bruyns et al. [BSM+02] provided a survey on existing interactive mesh cutting methods and categorize them according to how the cut path is defined, whether intersected primitives are simply removed or re-tessellated and by the number of new primitives generated during re-meshing. In their categorization, my system uses a cut path defined by a template (the fault line). Intersected primitives are re-meshed and the number of newly generated primitives is minimal - new vertices are only inserted at the intersection points between the fault line and the terrain geometry. Following the survey, the authors present a VR system which uses collision detection and haptic feedback to simulate surgery using a virtual scalpel. The performance of this system is based on the assumption that cutting happens incrementally as primitives are intersected by the scalpel. Under this assumption, it would be difficult to enable interactive modification of the cut path, as is possible with my approach.

Another surgery simulation system was presented in [SSE04], in which triangle meshes that intersect with a virtual scalpel are split and re-tessellated and a triangulation of the cutting plane is added to represent the exposed tissue. Free-form deformations are then applied to the sub-meshes on either side of the incision to simulate tissue behavior. My approach transforms the terrain mesh in a similar way by cutting triangles along a fault line, adding a triangulation of the fault plane and applying a deformation to the geometry on one side of the cut.

1.6.5 Ocean flow visualization

[NMG+96] have presented a VR-integrated in-situ visualization system for ocean flow simulations. Specifically, their system generates a low-resolution 2D model of the Sea of Japan, allowing the user to change simulation parameters on a front-end VR system and obtain an updated solution on the fly from a cluster. In the application scenario presented here, however, any change of parameters would require a re-computation of the entire time-dependent 3D flow field, making interactive parameter space exploration infeasible.

LIC, introduced by [CL93], is a well-established technique for visualizing flow on surfaces. By integrating a noise texture along the flow direction, a dense texture is computed which indicates flow direction at any point of the surface. I have

implemented a GPU-based version of the algorithm, as described by [HWSE99], using 3D textures to represent the flow field. Since static LIC does not convey flow orientation, I have chosen an animated variant by convolving the grayscale values with a phase-shifted sinusoidal kernel as described in [WGP97].

[SBK07] demonstrated interactive particle tracing using GPGPU techniques. Even though this approach would certainly be applicable here, a more simple CPU-based scheme was used here as the performance proved to be sufficient.

The CAVE, introduced by [CNSD+92], is a highly immersive virtual reality environment consisting of a room whose walls serve as projection surfaces for interactive stereoscopic imagery. My system is designed to be usable both in desktop and CAVE environments.

Chapter 2

Spherical Terrain Rendering using the hierarchical HEALPix grid

2.1 Introduction

Three-dimensional visualization of terrain is a well-studied problem with a history of algorithmic approaches. However, most of these solutions assume a "flat-earth" model, where topography is mapped to a plane. This is acceptable as long as the viewer is close to the surface but breaks down at distances where planetary curvature becomes relevant. Spherical terrain rendering aims to solve this problem by representing the planetary surface as a spheroid, which allows for terrain visualization at any scale.

Naive approaches to spherical terrain rendering frequently represent the whole surface within a two-dimensional coordinate system. However, any 2D parametrization of the sphere exhibits so-called coordinate singularities, which lead to visible sampling and/or rendering artifacts. For example, in the canonical geographic coordinates (latitude / longitude), singular points appear at $\pm 90\,^{\circ}$ latitude (north and south pole).

To avoid these singularities, a 3D parametrization must be used. One class of approaches tessellates the sphere into a small set of contiguous patches and uses a 2D parametrization on each patch, effectively forming a 3D coordinate system with

one integer and two real-valued coordinates. In this chapter I propose using the HEALPix [GHB+05] indexing scheme to represent terrain data, which decomposes the sphere into 12 curvilinear quadrilateral patches of equal area. These are then uniformly subdivided as necessary to form a sampling grid for representing geological data on the sphere. A particularly attractive property of HEALPix in this context is that all grid cells on a given subdivision level represent an equal area on the sphere.

My contribution is a 3D terrain rendering system based on the HEALPix parameterization. I demonstrate that this choice of coordinate system avoids degenerate geometry produced by other solutions using geographical coordinates. Furthermore, I show that interactive frame rates can be achieved by performing the HEALPix projection in real-time on GPU during rendering. This fact allows for a reduced memory footprint as well as a much simpler implementation compared to other systems which construct and maintain triangulations on CPU, since my approach operates primarily in the 2D domain of the HEALPix parameter space.

To enable Level-of-Detail rendering a multi-resolution database structure based on a set of quadtrees is used. A subset of these trees is kept in GPU memory and updated as the viewer moves by loading data in the background (data streaming). This visible subset is triangulated on-the-fly for rendering, using a GPU implementation of the HEALPix projection formulae to produce a spherical rendering.

In the following, I will present other work related to spherical terrain rendering. This is followed by a description of the data structure used by my system as well as an efficient transformation algorithm to convert geological data sets into this storage scheme. The actual rendering algorithm for DEMs which is based on a top-down traversal of this data structure is subsequently described, including aspects of LoD selection, frustum culling, triangulation of tiles and background data streaming. This algorithm is then extended to also support the visualization of high-resolution imagery draped on top of the DEM.

The system is applied to the interactive exploration of Mars, using a hybrid of MOLA (NASA) and HRSC (German Aerospace) data sets. While HRSC is of higher resolution than MOLA, it does not yet provide full coverage of Mars (as of 2011). To obtain the benefits of each data source, I have chosen to integrate both data

sets into a single database, demonstrating that the storage scheme can efficiently capture both at their native resolution.

In Section 5.5 we present our results using these data and give performance measurements to substantiate the interactivity claim. In the final section, some areas of further research are identified, focusing on aspects of performance and image quality.

2.2 Related work

Geometry Clipmaps [LH04] is a planar terrain rendering approach using rectangular, concentric rings of geometry centered around the viewer which decrease in resolution with increasing distance. The algorithm exploits temporal coherence in viewer movement by minimizing per-frame data structure updates using toroidal addressing.

Spherical Clipmaps [CH06] extend this scheme to spherical rendering by representing the planet's hemisphere which faces the viewer using a set of circular rings. However, due to low accuracy of the $\tan^{-1}$ function on the GPU, this method can produce visible cracks in the final triangulation. Furthermore, compared to the original scheme, vertices are no longer centered on actual height field samples. Due to the additional interpolation required, the data is essentially low-pass filtered when rendering.

Crusta [BCK+11] is a terrain rendering framework which uses a 30-sided polyhedron as base geometry, whose faces are recursively subdivided using a quadtree. It is therefore conceptually similar to our approach, however due to the purely recursive formulation of this geometric construction there is no analytical mapping between parametric and Euclidean coordinates. This prohibits performing this projection on the GPU and requires computation and storage of Euclidean vertex coordinates for individual tiles. Storing these 3-tuples for each sample at runtime produces memory overhead compared to my approach which only requires the scalar height, as the parametric coordinates are implicit and evaluated on the GPU.

Google Earth is a popular tool for exploring the surface of Earth using data streamed over the internet, which performs well even over low-bandwidth internet connections

and provides a high degree of interactivity. However, the system suffers from degenerate triangulations close to the north and south pole, leading to visible artifacts. See Section 5.5 for a visual comparison with the triangulations generated by my system.

2.3 Approach

To represent locations on the sphere, the HEALPix framework uses a tessellation into 12 curvilinear quadrilateral patches with associated parametric (u, v) coordinates in $[0, 1]^2$. For sampling and data storage, the authors suggest using a hierarchy of uniform (in parametric coordinates) grids for each patch [GHB+05]. Samples are stored in quadtree order to optimize locality.

As the given application scenario requires merging multiple data sets of different resolutions, I have decided to use an explicit quadtree structure for each patch to allow for local grid refinement. For performance reasons, each node stores tiles of 255×255 samples, drastically reducing tree height and management overhead. Neighboring tiles overlap by one sample at their shared boundary, which incurs a small storage overhead but allows for C_0 continuous rendering without needing to reference neighboring data.

While the leaf nodes of the tree contain data resampled at the native input resolution, the inner nodes store subsampled representations of their children, forming a multiresolution data structure suitable for LoD-rendering (see 2.3.2).

2.3.1 Data resampling

Digital Elevation Models (DEMs) are frequently represented as georeferenced raster data. These data sets consist of a 2D matrix of height values (the raster data) as well as a so-called georeference which associates sample locations with their corresponding geographic location.

To transform a given data set to the quadtree database format described previously, a bottom-up construction process is used. First, an optimal tree depth (resolution)

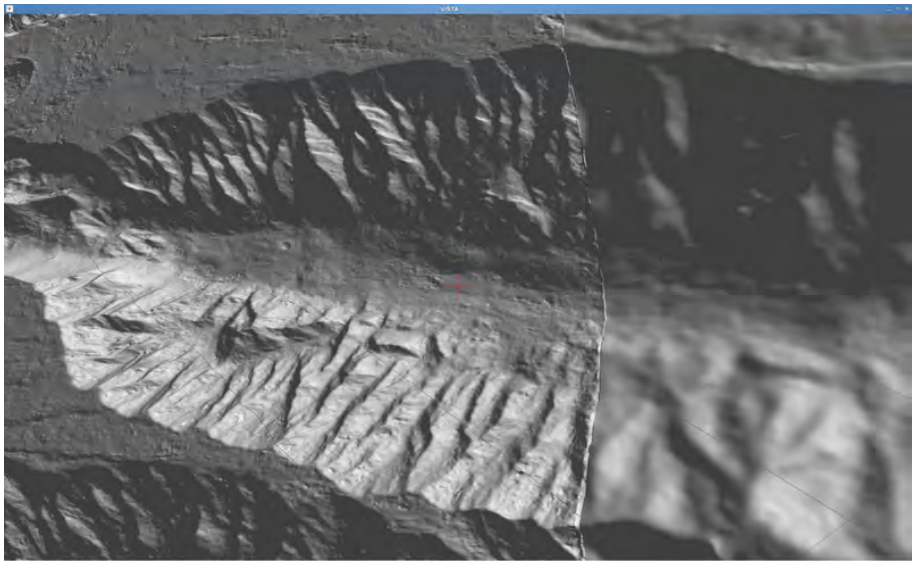


Figure 2.1: Embedding of high-res into low-res DEM (left: HRSC, right: MOLA)

is chosen to faithfully represent the input data. Then the subset of leaf nodes at this level which potentially intersect the data is identified. These nodes are then populated by resampling the raster input and finally the inner nodes are computed by downsampling. Note that the whole process is repeated for each base patch.

The choice of tree depth depends on the resolution of the input data. Due to the equal area property of HEALPix, the grid resolution for a given subdivision level is constant everywhere on the sphere. To faithfully represent the input data, the tree depth d should be chosen so that the sample density of the leaf nodes is at least as high as the native resolution of the input data.

Each of the high-resolution raster maps in the HRSC data set cover only a small fraction of the planetery surface. Therefore the set of leaf nodes to consider in resampling should be limited to a subset which is likely to intersect the data. To approximate this set, the boundary curve of each raster map is computed by projecting each boundary pixel location to geographic coordinates (using the supplied georeference) and then to HEALPix coordinates. Then, each leaf node which intersects the axis-aligned box (in HEALPix coordinates) around these points is considered in turn.

For each of the selected leaf nodes, the system iterates over all of the 255×255 sample positions, projecting each first to geographic and then to raster coordinates. Bilinear interpolation within the raster data is used to compute the resampled value. After all intersecting leaf nodes have been populated in this manner, the inner nodes of the tree are derived by iterative downsampling until the contents of the root node have been obtained.

Note that raster data sets can designate a special NODATA value, which is assigned to samples having no meaningful measurement. If any of the four input samples used in bilinear interpolation contain this value, the resulting interpolated sample is also marked as NODATA. This value is also assigned if the sampling coordinate (after projection) is not within the bounds of the raster image. If all samples of a node contain NODATA values after resampling, the node is not stored at all.

To support merging multiple data sets, it is also possible to insert data into an existing quadtree database. Already existing nodes are combined with newly generated ones by replacing their sample values. However if an incoming pixel is marked as NODATA, the previous value is kept. This treatment is required because the actual definition domain of many data sets is much smaller than their sampling support, with the difference areas being filled with NODATA samples. Using this scheme it is possible to insert sparse high-resolution data into an existing low-res DEM database as shown in Figure 3.3.

The construction process can be trivially parallelized over the 12 base patches, as the quadtrees are mutually independent. By running 12 instances of the construction tool (possibly on different machines), limiting each instance to only consider tiles within the associated base patch, a database file is generated for each. These are then merged using another tool. While this is not optimal in terms of speed up (mainly due to the required I/O for merging the databases), it is sufficient to generate the hybrid MOLA and HRSC databases in a single day.

The input data sets have a size of 2 GiB for the MOLA DEM, 24 GiB for the HRSC DEM, 54 GiB for red, green and blue channels (HRSC) and 386 GiB for the high resolution B/W nadir channel (HRSC), for a total size of 466 GiB. This data is processed to a set of five databases (DEM, R, G, B, B/W) with a total size of 1.7 TiB.

2.3.2 Rendering

To render the terrain representation, each of the 12 base-patch quadtrees is recursively traversed in top-down fashion. Recursion stops at a tree node in any of three cases:

1. The node (and all of it's children) are outside of the viewing frustum and therefore invisible. Recursion returns without rendering.

2. The node is sufficiently subdivided to meet the screen space quality requirements (see 2.3.4). The node is rendered and recursion returns.

3. The LoD heuristic decides that further refinement is necessary, however the immediate child nodes are not in memory. In this case, the background I/O thread is instructed to load the four child nodes from disk and the current node is rendered as a placeholder until that data is available.

These cases are tested for in the order given. If none of these occur, recursion continues with the four children of the node being considered. This rendering process is stateless in the sense that no information is kept about the set of nodes rendered in the previous frame, minimizing management complexity. Nodes visited during rendering are kept in memory, however. In the following, the individual components of this architecture will be described in detail.

2.3.3 Frustum culling

Frustum culling is a technique to eliminate geometry which is located outside of the field-of-view of the camera and therefore guaranteed not to be visible. For performance reasons, bounding geometries are usually employed as proxy objects for this visibility test. Specifically, we follow the classical approach of maintaining an axis-aligned bounding box for each node which is resident in memory. In computing this bounding box, minimum and maximum height values within the node as well as any user-specified height exaggeration factor have to be considered.

Given the geometry of the view frustum, which is a pyramid truncated by two parallel planes, and the extents of the node's bounding box, the separating axis theorem is used to test the two bodies for intersection. This theorem states that, given two convex shapes, an axis exists onto which their projections are separate (non-intersecting) iff they are not intersecting.

When testing two polygonal meshes A and B for intersection, the set of axes which need to be tested in this manner is small. Specifically, only the set of face normals and the cross products of all pairs of edges where one edge is taken from A and the other is taken from B need to be considered (see [GLM96]). If the projections of A and B onto any of these axes do not intersect, the original meshes do not intersect either.

The advantage of using axis-aligned bounding boxes is that the set of projection axes is constant for a given view-frustum as the set of face normals and edge directions of the bounding boxes are always the three canonical axes. Therefore, it needs to be computed only once per frame. Furthermore, the bounds of the projection of the view frustum onto this set of axes can likewise be pre-computed, accelerating the test.

2.3.4 LoD rendering

Level-of-Detail rendering takes advantage of the limited resolution of raster displays by reducing geometric complexity depending on the apparent size of the geometry on the screen, which in turn depends on viewer distance and camera perspective. In the visualization of large data sets, LoD techniques are often mandatory to bring rendering performance to interactive levels. A conservative approach using the screen-space area of bounding boxes was implemented, which is described in the following.

One of the goals of the LoD scheme used in this work is being able to guarantee maximum visual fidelity. When rendering polygon meshes, this translates into maintaining a geometric resolution of about one vertex per pixel when rasterizing. As the quadtree data-structure only allows for a selection between discrete resolution levels, the strategy is to refine nodes during rendering until at least this level of resolution is reached. However, computing the exact pixel-coverage of a node's

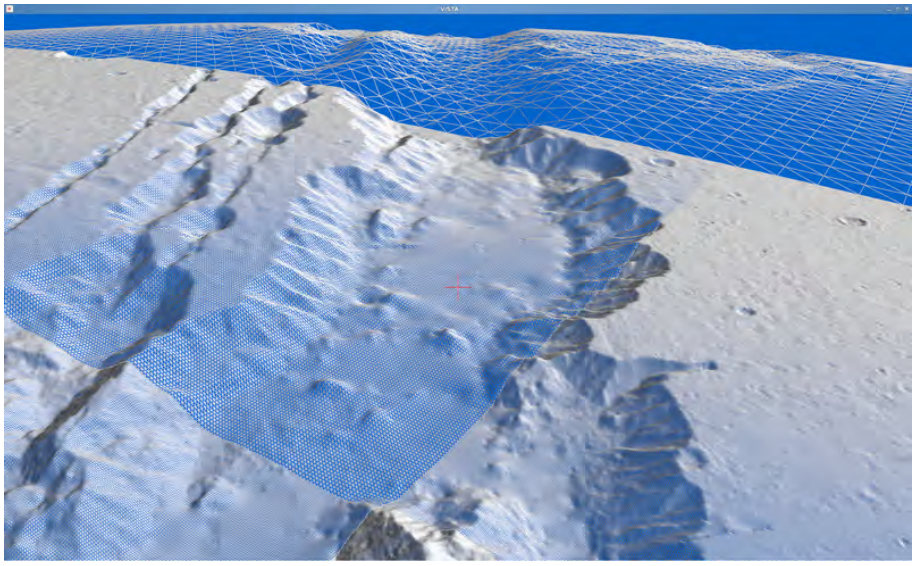


Figure 2.2: Distance-dependent LoD selection

geometry requires similar effort as rendering this geometry. Instead, the bounding boxes associated with resident nodes are used as a proxy to obtain an upper bound for the screen-space area occupied by a node, as described in the following.

In order to estimate the number of pixels occupied by a node on the screen, the node's bounding box vertices are projected into screen space using the same projection and modelview matrices as used during rendering. Then for each of the six faces of the bounding box the signed area is computed. If the area of a face is negative, it is facing away from the viewer and ignored. All the positive areas are summed up to give the total screen area of the box in pixels, which is always greater or equal to the pixel area of the actual geometry if it were rasterized.

Finally, the estimated pixel area is compared with the number of mesh vertices, which is constant and equal to the number of height samples per node (255×255). If it is smaller, the subdivision level is adequate for the current view and the node is rendered. Otherwise, the recursion continues and the heuristic is applied again to the children.

As a guaranteed resolution of one vertex per pixel is excessive in most use-cases, a user-selectable scaling factor is provided which specifies the desired average number

of vertices per pixel. Empirical studies suggest that a choice of 0.2 provides high visual quality while maintaining good interactive performance.

Figure 2.2 shows how tree nodes close to the viewer are rendered at a high resolution which decreases with distance. Note that in this example the pixel area threshold was chosen very large for illustration purposes.

2.3.5 Data streaming

In order to guarantee interactivity, it is mandatory to perform slow disk operations asynchronously. For each database, a separate I/O thread is maintained which performs these operations concurrently with the render thread. A job queue is used to store read-requests while a result queue contains the loaded data. If during rendering the LoD heuristic decides to refine a node but it's children are not yet in memory, a request is posted onto this queue to load the four child nodes. The I/O thread takes jobs out of this queue and processes them, appending the loaded data to the result queue.

At the beginning of each frame, before starting the actual rendering traversal, the render thread inspects the result queue and inserts any newly loaded nodes into the in-memory quadtree. These nodes are then available for subsequent rendering. This strategy restricts access to the quadtree to the render thread, reducing the complexity of thread synchronization, which is only required for shared access to the job and result queues.

2.3.6 Rasterization

Individual nodes are rendered using triangle meshes interpolating the height field. Each node represents a square sub-region in the parameter space of its associated HEALPix patch. The coordinates of individual mesh vertices are derived by equidistant interpolation within this region. For rendering, mesh vertices are projected to geographic coordinates (latitude / longitude) using the HEALPix formulae. These coordinates are then combined with the corresponding height samples (radii) to form polar coordinates which are eventually converted to Euclidean coordinates.

Storing a triangle mesh in Euclidean coordinates for each resident node would be expensive, however. By performing the coordinate conversion on the GPU, we compute sample locations on the fly, reducing memory usage by two-thirds.

My approach uses a single 2D proxy mesh which is a uniform tessellation of the $[0, 1]^2$ unit square as well as a per-node texture containing height samples. The vertex shader implements the following four steps to transform this proxy mesh to the final geometry:

1. Transform mesh to proper sub-region in parameter space
2. Convert parametric coordinates to geographic
3. Read height values out of texture and generate polar coordinates
4. Convert to Euclidean coordinates

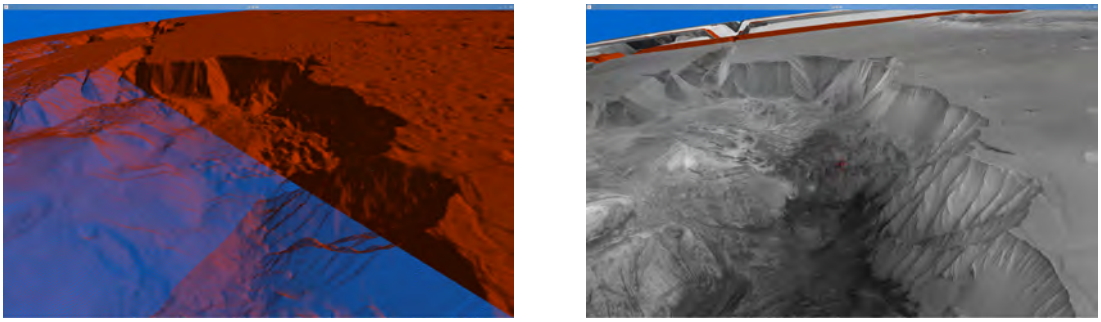
Note that texture coordinates are centered on the height samples (texels), which guarantees that the sample points are interpolated by the geometry.

To implement shading, normal vectors are estimated using central differencing of the height field. To avoid analytical computation of the u, v direction vectors necessary for the normal estimation, they are evaluated at the corners of a patch and passed to the shader, which interpolates them across the patch.

2.3.7 Imagery overlay

In the following the approach previously presented for rendering DEMs is extended to incorporate imagery. Imagery data is processed in the same way as DEMs, producing parallel quadtree databases. The advantage of not combining DEM and imagery into a single database is that both can be arbitrarily mixed and matched at runtime.

In the rendering traversal, DEM and imagery database nodes are now visited in parallel. A straightforward approach to render each pair of data is extending the vertex shader to assign vertex colors from the imagery texture. However, this is not



(a) Shaded DEM (wireframe vs. opaque) (b) DEM with high-res BW imagery

Figure 2.3: Shading vs. Texturing

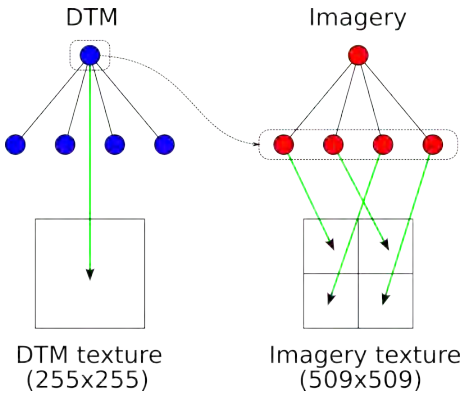


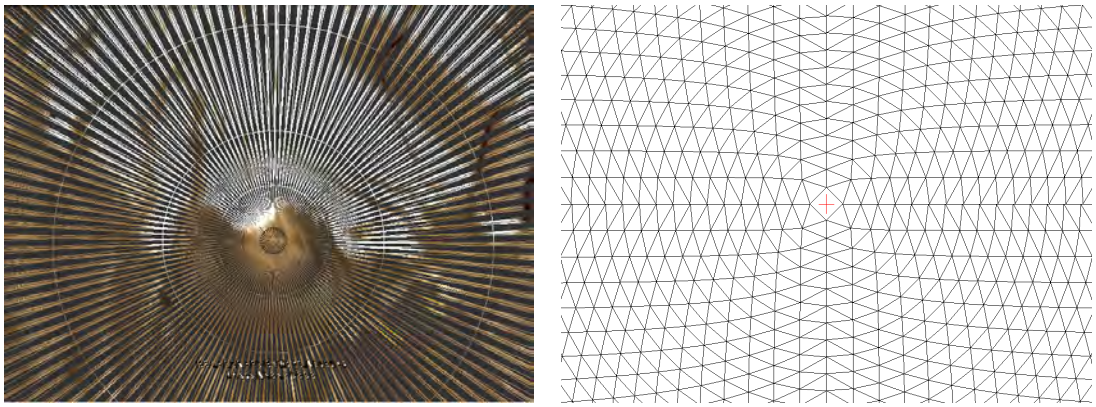
Figure 2.4: Merging multiple imagery nodes into single texture ($\Delta_h = 1$)

sufficient due to the fact that imagery data is often of higher resolution than the DEM. These additional levels of resolution are never displayed in this scheme, as the visualization is constrained by the DEM resolution.

Therefore, I have chosen a different approach as illustrated in Figure 2.4. Let Δ_h be a parameter which specifies the maximum resolution difference between a DEM node and the corresponding imagery. During rendering, the DEM and imagery tree are traversed in parallel as described above. However, instead of displaying the imagery node at the same resolution level as the DEM node, recursion into the imagery database continues to collect all child nodes of degree Δ_h. The imagery data so obtained is then merged into a single large texture. Figure 2.4 shows an example for $\Delta_h = 1$. Regular (fragment stage) texture mapping is then used to provide additional visual detail without increasing geometric primitive count.

The selection of which databases to display can be made at runtime. To simplify switching to another set of channels, the resident tiles are simply deleted from memory which results in the rendering traversal (re-)loadind any nodes required for the current view, which usually takes less than a second.

2.4 Results



(a) Google Earth (b) Our system

Figure 2.5: Comparison of triangulations at north pole

Figure 2.5 shows the triangulation quality in the vicinity of the poles, comparing Google Earth to our approach. Google Earth produces a bad triangulation consisting of long, skinny triangles due to an obvious coordinate singularity at the pole. In contrast, using a uniform sampling of the parametric HEALPix coordinates yields a well-behaved triangulation consisting of equal-area triangles with good aspect ratios.

Figure 2.6 shows the surface of Mars, visualized at different LoD-thresholds. Window size is 1024×768 pixels in each case. While rendering performance is not fully interactive at a threshold of 1.0 (min. 1 triangle per pixel), visual quality at 0.2 is not noticeably worse while providing interactive frame rates. The fact that adding imagery textures does not significantly affect these results indicates that performance is limited by GPU geometry processing performance.

While image quality degradation is obvious when comparing the shaded DEMs at thresholds 0.2 and 0.01, it is hardly noticeable when the same geometries are com-

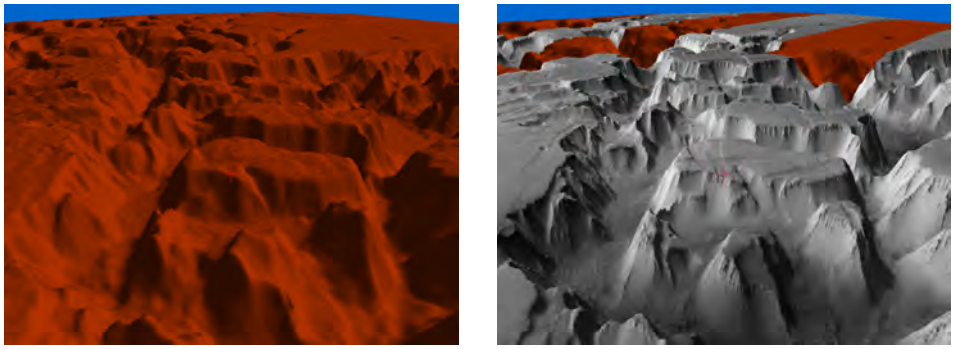
pared with imagery texturing. Therefore, very low LoD-thresholds can be used when imagery is present, resulting in highly interactive frame rates.

2.5 Conclusion and future work

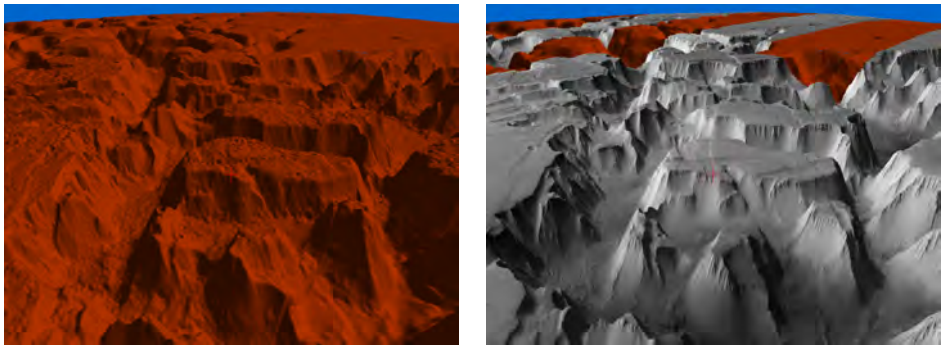
I have presented an interactive terrain rendering architecture using the HEALPix coordinate system to provide spherical rendering without singularities. By performing critical computations on the GPU, both memory consumption and management complexity are reduced compared to other schemes.

Possible future extensions including incorporation of LiDAR data by using scattered data interpolation schemes to resample data to our uniform grids, locally refining the tree depending on sample spacing.

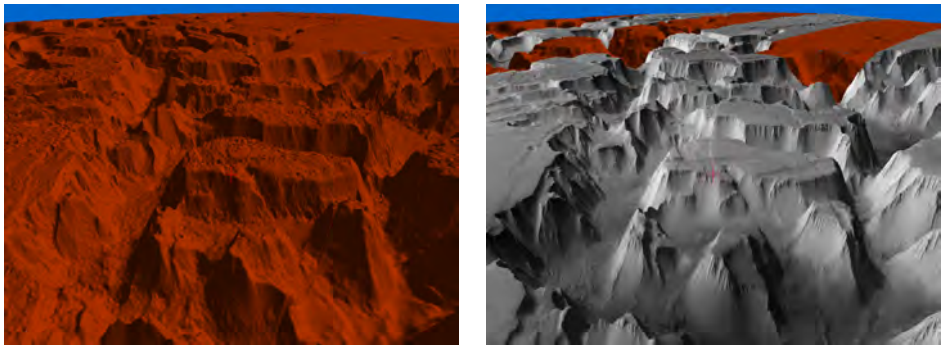
Using the same rendering scheme as for high-resolution imagery, normal mapping could be implemented to provide additional visual detail at low LoD-thresholds. These normal maps could be computed at runtime using the subtrees of the visible DEM node set.



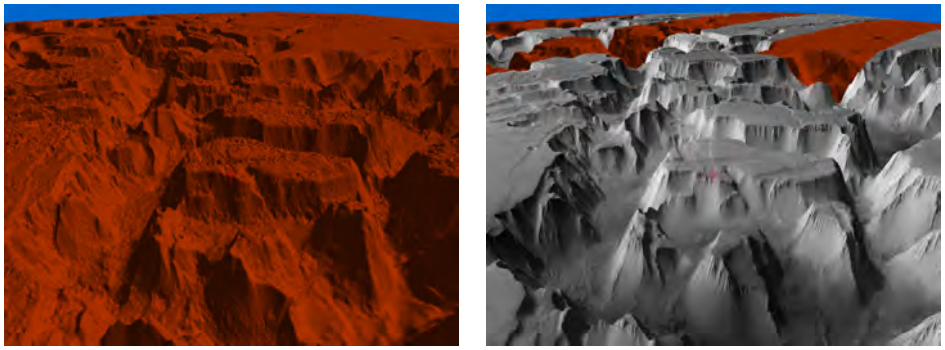
(a) LoD-threshold 0.005, 105 fps (b) 113 fps



(c) LoD-threshold 0.01, 75 fps (d) 77 fps



(e) LoD-threshold 0.2, 24 fps (f) 26 fps



(g) LoD-threshold 1.0, 9 fps (h) 10 fps

Figure 2.6: Visual quality and rendering performance at different LoD-thresholds (left: shaded DEM, right: DEM textured with high-res B/W channel ($\Delta_h = 2$))

Chapter 3

Remote GPU-Accelerated Online Pre-processing of Raster Maps for Terrain Rendering

3.1 Introduction

Interactive three-dimensional terrain visualization has become accessible to a wide audience due to the development of advanced rendering algorithms which can guarantee interactivity at very high resolutions. With the emergence of inexpensive remote sensing platforms such as quadcopters, future challenges will include the integration and visualization of changing data collected from multiple sources such as situation information necessary for disaster management or public security.

Rendering algorithms typically require data to be represented using specific multi-resolution data structures, which encode elevation models and imagery redundantly at multiple different resolution levels. The rendering algorithm typically selects a suitable level at runtime as a function of screen resolution and current view perspective. Due to the static nature of most terrain datasets, the computational effort for pre-processing remote sensing data into this format is usually not considered critical.

My contributions include a GPU algorithm to accelerate the conversion of raster maps into a multi-resolution data structure as well as a distributed, online preprocessing framework which applies this algorithm to enable rapid visualization of raster maps within a 3D terrain visualization system. Composition of multiple, potentially overlapping mixed-resolution raster maps is performed on the fly. A quad tree hierarchy on top of the HEALPix [GHB+05] sphere tessellation was used as reference data structure. The benefit of this approach is a significant reduction in turn-around time required to visualize remote sensing data stored in standard raster formats.

For benchmarking purposes, both the Mars Orbiter Laser Altimeter (MOLA) [SZF+01] and High Resolution Stereo Camera (HRSC) [GSP+10] datasets were used. The MOLA mission has provided a global Digital Elevation Model (DEM) of Mars with a resolution of $460m$ per pixel for a dataset size of 2.0 GiB. The HRSC datasets, which cover about 33% of the surface of Mars, include a high-resolution DEM at an average resolution of $90m$ per pixel (23.2 GiB) consisting of 1161 individual raster files.

Both datasets were merged into a single database using HRSC data wherever available and MOLA data to fill the gaps. This database was computed on-the-fly and visualized within an interactive spherical terrain rendering system.

3.2 Related work

Interactive terrain rendering is a well-established area of research. Terrain visualization systems are usually coupled to specific data structures. As computing paradigms shifted over the decades, algorithms have adapted to exploit the characteristics of the hardware of the day. Early approaches such as ROAM [DWS+97], introduced by Duchaineau et al., perform triangle-level adaptation of the rendered geometry on the CPU to minimize geometric complexity before rasterization. These strategies have become prohibitively expensive on modern hardware which provides very high rasterization bandwidth using GPUs but suffers from relatively slow CPU performance and bus bandwidths.

Due to these developments, more recent solutions try to optimize batching of the geometry by treating blocks of geometry as opaque entities which are rasterized as a whole. Levenburg's CABTT algorithm [Lev02] extended classical triangle bin-tree algorithms by processing batches of geometry bounded by a triangular patch instead of individual triangles. These patches are uploaded to GPU memory and re-used for multiple frames to reduce CPU workload and bus traffic.

The P-BDAM algorithm [CGG+03] of Cignoni et al. uses a similar type of batching but uses adaptive triangulations within each patch, which are precomputed off-line. Spherical rendering is realized by using curved patches. Distributed pre-processing is enabled by subdividing the surface of the sphere into block that can be processed independently.

The Geometry Clipmaps [LH04] scheme by Losasso and Hoppe uses a set of nested regular grids which decrease in resolution with increasing distance from the viewer. Temporal coherence in viewer movement is exploited by introducing a toroidal indexing schemes which allows to minimize the per-frame cost of updates on the GPU as new data is being loaded. The data structure being used to represent terrain is a pre-filtered mipmap pyramid on a regular grid.

Planetary-Scale Terrain Composition [KLJ+09] by Kooima et al. is an unconventional approach for real-time composition of unregistered raster maps which defers traditional resampling to the rendering phase. A mipmap representation of the data being loaded is required, however. During rendering, sphere geometry is generated on the fly using iterative subdivision of a base icosahedron. The maximum depth of this subdivision is determined by a LOD heuristic, depending on viewer position. Finally, vertices are displaced along their normals according to height values stored in the input datasets, which are uploaded to the GPU as textures.

Vertices of the triangle mesh being subdivided contain spherical coordinates as well as normals. Depending on whether an intersecting data set uses spherical or polar projection, either spherical coordinates or normals are used as texture coordinates to sample a mipmap representation of the original raster image. Mesh vertices must be displaced and colored accordingly. This texture look-up is efficient, as texture coordinates have already been computed at that point. The computation of texture coordinates, however, is computationally expensive. Due to numerical

instability, the spherical coordinates can not be computed in a closed form but have to be maintained as the subdivision progresses. To generate spherical coordinates for newly interpolated vertices, the haversine geodesic midpoint method is used. This method is computationally expensive, especially considering that it has to be evaluated once for every vertex in every frame.

My system is similar in that the given map data is resampled data onto a singularity-free grid. Instead of an implicit subdivision scheme, however, the HEALPix [GHB+05] sphere parametrization is used. HEALPix has an explicit projection formula which is computationally cheap to evaluate and maps between parameter space and spherical coordinates. This allows for independent evaluation of grid vertex positions at any refinement level, enabling highly parallel GPU based resampling. In the online pre-processing approach, this resampling is only performed once as tiles are being generated. These tiles are then cached by the front-end visualization system.

In the Crusta [BCK+11] terrain rendering system, a rhombic triacontahedron is used as base geometry, which is iteratively subdivided in a similar fashion as in [KLJ+09]. For each of the 30 base faces, a quadtree is used as a hierarchical multi-resolution representation. Each tree node stores a tile of 64×64 samples for efficient batching in rendering. Data is re-sampled onto the leaf node grid vertices in an off-line pre-processing step and inner nodes are computed by iterative downsampling. As in [KLJ+09], the implicit subdivision scheme makes direct sample addressing computationally expensive.

My approach is similar in that it uses a quadtree to define a hierarchy on top of a sphere tessellation. The chosen HEALPix tessellation uses 12 curvilinear patches to represent the sphere. However, as the HEALPix scheme has an explicit projection formula, coordinate transformations can be evaluated explicitly and are therefore suited for massively parallel evaluation on GPU.

Lambers et al. [LK10] have presented a system to generate view-dependent geometry on the fly to allow for visualization of fully dynamic datasets. The GPU-based per-frame generation of this triangulation is computationally expensive, however, and does not guarantee interactivity. Treib et al. [TRAW12] use a compressed wavelet representation for terrain data which is visualized at interactive rates using GPU-based ray-casting. The chosen representation allows for efficient real-time editing

of terrain. A quad-tree topology is used for the multi-resolution data structure, however tree nodes contain only the (compressed) differences to their parent nodes as opposed to storing absolute height values. When the terrain is modified at a given resolution level, changes are propagated through the quadtree. To populate coarse resolution levels, the affected subtree is recomputed using a bottom-up construction process starting from the modified tiles. To propagate downwards in the tree, editing operations are recorded and applied on-the-fly to finer nodes as they are loaded into GPU memory. In my system, data being inserted into the tree is always represented as leaf nodes, hence only upward propagation is necessary. This is likewise achieved by updating the tree using iterative downsampling, starting from affected leaf nodes.

GPU algorithms have also been used for pre-processing remote sensing data. Thomas et al. described a system [TKR+08] for GPU-based orthorectification, which is a process to remove perspective distortions in imagery to produce a top-down view of uniform scale. In their approach, imagery obtained from an oblique perspective is projected in real-time onto an existing DEM to produce a corrected orthophoto using a technique related to shadow mapping. In the work presented here it is assumed that the input data is already available as orthophotos. In an actual crisis management scenario, one could imagine coupling both approaches to obtain a pipeline for immediate visualization of live remote sensing data.

3.3 Pre-processing framework

In the following I will characterize the input data used in my approach, describe the structure of the output database and present an algorithm to construct this database. Subsequently, a GPU implementation of the same algorithm is introduced which is then applied to implement an on-line pre-processing framework. Using this framework, data tiles can be generated on-the-fly by a backend GPU cluster for immediate visualization on a frontend rendering workstation.

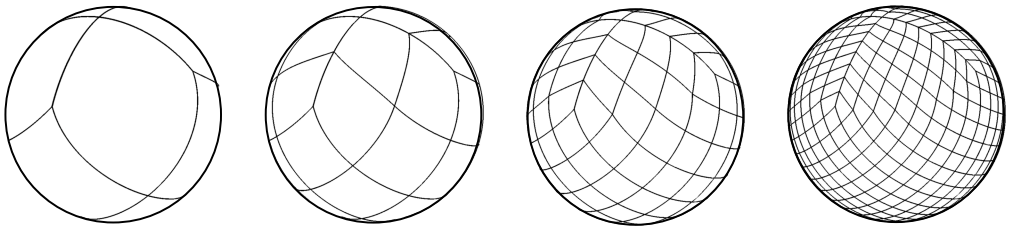


Figure 3.1: The HEALPix hierarchical sphere tessellation. All cells on a given subdivision level have equal area and their coordinates can be computed using a closed formula.

3.3.1 Data representation

The input data format expected by my framework are geo-referenced raster DEMs (Digital Elevation Maps). These raster images consist of a regular grid of height values as well as a so-called georeference which is a projection that associates sample coordinates with physical locations on the planetary surface.

For the output coordinate system I have chosen the HEALPix grid as it uniformly samples the sphere which allows for artifact-free spherical rendering without coordinate singularities. This is achieved by tessellating the sphere into a set of 12 curvilinear base patches which are uniformly subdivided to form a grid hierarchy [GHB+05].

A quad tree subdivision scheme is applied on top of this hierarchical grid to obtain a multi-resolution database suitable for Level-of-Detail terrain rendering as well as for representing sparse and mixed-resolution datasets. A forest of 12 quad trees represents the base patches and each tree node stores a tile of 255×255 samples for efficient batching in database management and rendering.

3.3.2 Off-line database construction

In the following I will describe the resampling process which converts a set of raster maps to a terrain database. This requires selecting a quad tree subdivision depth which adequately samples the input raster maps. Considering the tiling used in my

approach, the ground area of a single sample at tree depth d for a spherical planet of radius r is given by

$$A_d = \frac{4\pi r^2}{12 \cdot 254^2 \cdot 4^d},$$

which is constant everywhere due to the HEALPix equal-area property. For raster maps, the area represented by an input pixel is usually given. To preserve data resolution, a depth d has to be chosen such that the sample size is equal or small than this value.

The construction process then performs two passes over the given data. In the first pass, the boundary polygon of each raster map is projected to the HEALPix coordinate system to determine the set of intersected leaf nodes. Output file locations are assigned to nodes at this point to optimize the data layout for rendering. Note that this pass requires little I/O as only file header information is required.

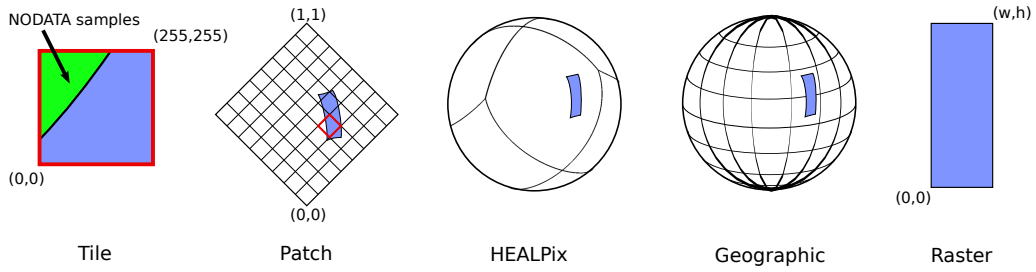


Figure 3.2: Coordinate systems involved in the resampling process

In the second pass, each raster map is loaded sequentially and the intersecting leaf nodes are populated. To compute sample values for each node, each sample coordinate pair is first converted from parametric HEALPix space to geographic coordinates. These geographic coordinates are then converted to pixel coordinates within the input raster image, according to the provided georeference (compare Figure 3.2). The image is then sampled using bilinear interpolation. Note that all coordinate system conversions are performed at double precision as single precision is not sufficient to encode distinct points on the surface at or below meter-scale [KLJ+09].

Note that datasets can assign a special NODATA value to samples which do not have a meaningful value. This is often the case near the boundaries of a raster map when the actual support is not rectangular in shape. These NODATA values are

propagated in interpolation and for sample coordinate outside of the raster image extents, NODATA values are produced as well.

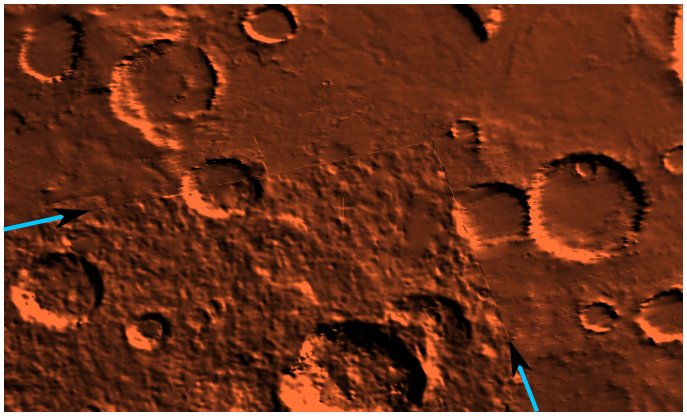


Figure 3.3: Embedding of a high-resolution DEM (HRSC) into a low-resolution DEM (MOLA). Arrows indicate dataset boundaries.

Nodes which already exist in the database are loaded and composited with newly generated data. Each existing sample is replaced by the new sample, unless that new sample was marked as NODATA. This implies that the composition result depends on the order in which input files are processed. Maps which are processed later in the sequence take precedence and replace already existing data. Therefore, care must be taken that data is specified in corresponding order from coarse to fine resolution. This could easily be automated, however, since the sample resolution of each dataset is given as header information. Figure 3.3 demonstrates the result of combining HRSC with MOLA DEMs.

After producing the finest level of the hierarchy in this manner, the inner nodes of each quad tree are generated by successive downsampling of the leaf nodes.

3.3.3 GPU-accelerated resampling

To accelerate the resampling process described above, I have developed a GPU implementation using the CUDA framework. As before, raster images are processed sequentially. Large images which do not fit into GPU memory are first subdivided into chunks which are then transformed individually.

For processing, each chunk and the associated georeference is uploaded to GPU memory and the set of intersecting tiles is produced. For performance reason, tiles are processed in batches of 64 elements. For each batch, tile coordinates are uploaded and a CUDA kernel is executed which performs projection and resampling. Kernel execution and downloading of result tiles to CPU memory are overlapped using CUDA streams, which allows finished data to be transferred as other tiles are still being processed.

As in the CPU implementation, all coordinate conversions are performed using double precision arithmetic, which was not feasible on early CUDA implementations that suffered from very low double-precision performance. On current hardware generations, however, double-precision arithmetic is efficient and achieves half of the single-precision bandwidth. After tile generation and downloading, the tree is populated by iterative downsampling as before. This is still executed on CPU, as it is a relatively inexpensive operation.

3.3.4 On-line approach

To eliminate the turn-around time for visualizing raster maps, I have developed an on-line approach in which tiles needed for visualization are generated on-the-fly by a distributed pre-processing back-end (Figure 3.4). In contrast to the previous hierarchical construction scheme, which used iterative downsampling to produce inner nodes, the online scheme uses point sampling to directly compute sample values for inner nodes.

To enable the cluster nodes to answer random queries in short time, the datasets to be visualized are loaded into host RAM when the system is started. To optimize memory utilization, input files are distributed equally (by size) across all cluster nodes. As the input files are loaded they are subdivided into chunks and tile coverage for each chunk is computed.

When all data is loaded, the cluster nodes start receiving requests from the front-end. Tile requests which are not covered by any loaded chunk generate an empty reply message. All other requests are handled using the GPU resampling algorithm. To generate a tile it is necessary that the corresponding chunk is resident in GPU

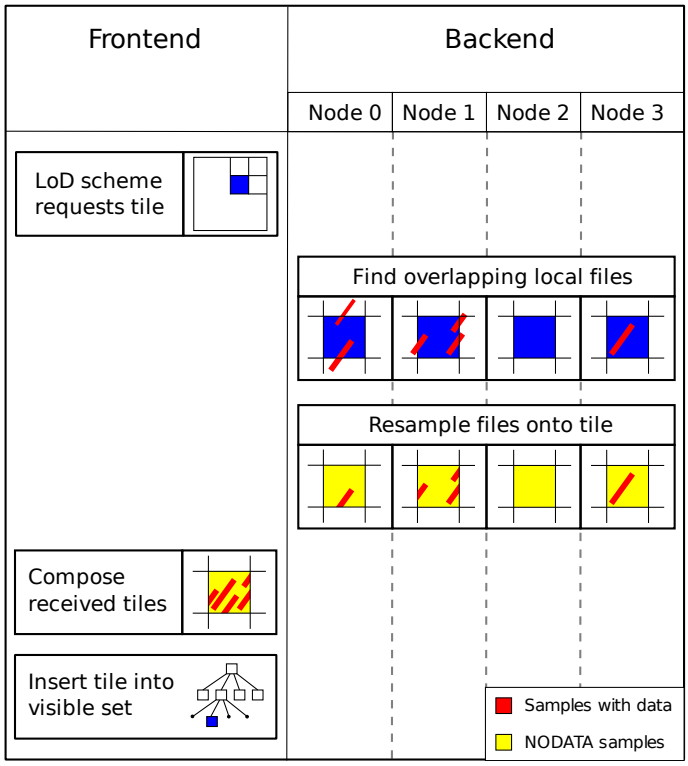


Figure 3.4: Illustration of distributed on-the-fly processing

memory. However, replacing the currently loaded chunk is expensive because it involves copying the data over a relatively slow bus.

To minimize this paging of chunks, a set of outstanding tile requests is maintained. These tile requests are grouped by the chunk they refer to and each group is processed as a batch. Tiles which are intersected by multiple chunks are included in each corresponding batch. For these tiles, the resulting data is composed as described in 3.3.2. After processing all outstanding requests, resulting tiles are sent to the rendering front-end and the process is repeated with the set of requests which have been received asynchronously in the meantime.

Once the front-end has received a reply by each cluster node for a given request, it composes the data received and incorporates it into the visible set of tree nodes.

3.4 Results

For benchmarking, I used a machine equipped with a dual Intel Xeon X5670 hexacore processor with hyper-threading, resulting in 24 "virtual" cores, as well as 48 GiB of host RAM and a NVIDIA Quadro 6000 GPU with six GiB of memory. The cluster used for on-line pre-processing consists of four machines of the same hardware configuration.

3.4.1 Raw resampling performance

Resampling performance was measured using the set of 1173 HRSC archival DEMs [GSP+10] with a total size of 23.2 GiB. Out of these, 12 files (80 MiB) had to be excluded as they use stereographic projection, which was not supported by my CUDA kernel at that time. Using the formula given in 3.3.2, I have chosen a quad tree depth of $d = 8$.

For each input file, the intersecting tile set was computed and all tiles at the finest resolution were generated, measuring total CPU and GPU implementation runtimes for tile generation only. The CPU resampling kernel was parallelized using OpenMP to make full use of all available CPU cores.

The CPU implementation required a total time of 52 minutes in tile generation while the GPU implementation completed the same workload in 2 minutes, of which 15 seconds were required for data upload. Download of results was not measured separately but is included in the given time. The GPU implementation therefore achieved a speed-up factor of 27.9 for resampling and downloading of results and 24.7 when including the time required for uploading input data.

This result is explained by the fact that the GPU architecture is well-suited for the problem since samples can be evaluated in parallel. For implicitly defined grids as in [KLJ+09] and [BCK+11] this would not be the case. Furthermore, the kernel exhibits a high arithmetic density due to the multiple coordinate system transformations involved for each sample.

3.4.2 Off-line pre-processing

To measure the impact the improvements have in an actual application, a quad tree representation of the same dataset was constructed on external storage, producing a 120 GiB database. The CPU-based construction required 101 minutes while the GPU variant required 59 minutes, a 70% performance increase. Disk I/O was identified as a bottleneck as a significant fraction of the time (about 50 minutes) was spent waiting for I/O completion.

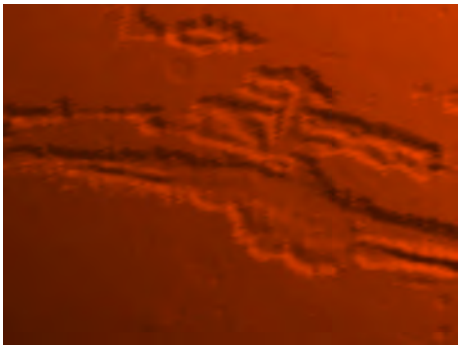
3.4.3 On-line pre-processing

To demonstrate on-line pre-processing of mixed resolution datasets, the input dataset used previously was extended by the MOLA MEG128 data, which is two GiB in size. A tree depth of five was chosen for this data.

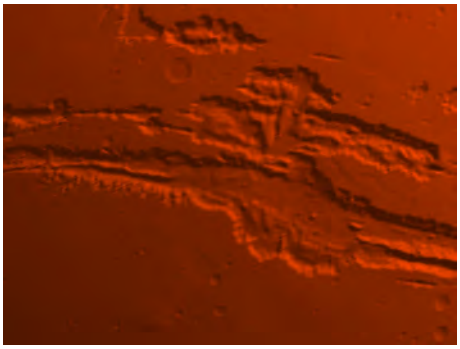
The data files (26 GiB total) were distributed about equally among the four cluster nodes. The start up time required for all server processes was 70 seconds. The additional time then required to provide a coarse initial rendering (at tree depth 0) was measured to be about ten seconds. This is explained by the fact that the root nodes of the hierarchy contain all map chunks. Therefore each chunk had to be uploaded in turn to generate the root nodes.

However, performance improved once the initial levels of the tree had been generated and cached by the front-end. See Figure 3.5 for a timeline of convergence for the first four levels in a top-down view of Valles Marineris. Figure 3.6(a) shows the average time the front-end needed to wait for a requested tile, depending on tree depth, collected during an extended browsing session.

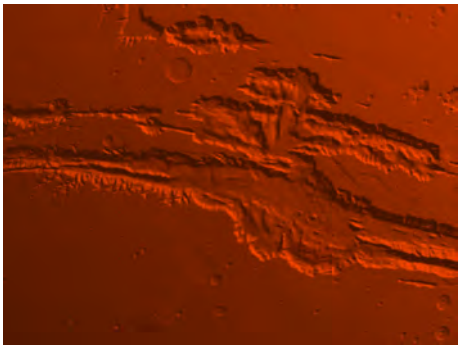
Because tree nodes contain a fixed number of height samples and the LoD scheme employed by the front-end attempts to maintain a fixed ratio between number of triangles and screen pixels, the number of visible nodes is approximately constant. As the viewer moves closer to the planetary surface, however, the number of visible chunks decreases and requests for tiles deeper in the tree are serviced faster as paging is reduced.



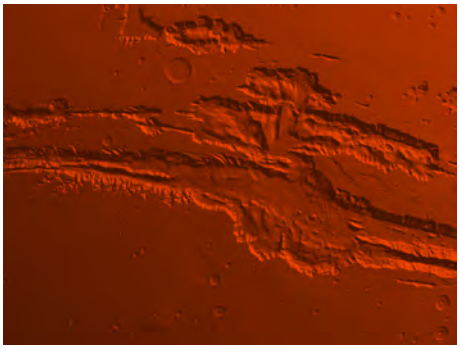
(a) Resolution level 0 (after 9.9 seconds)



(b) Resolution level 1 (8.27 seconds later)



(c) Resolution level 2 (4.72 seconds later)

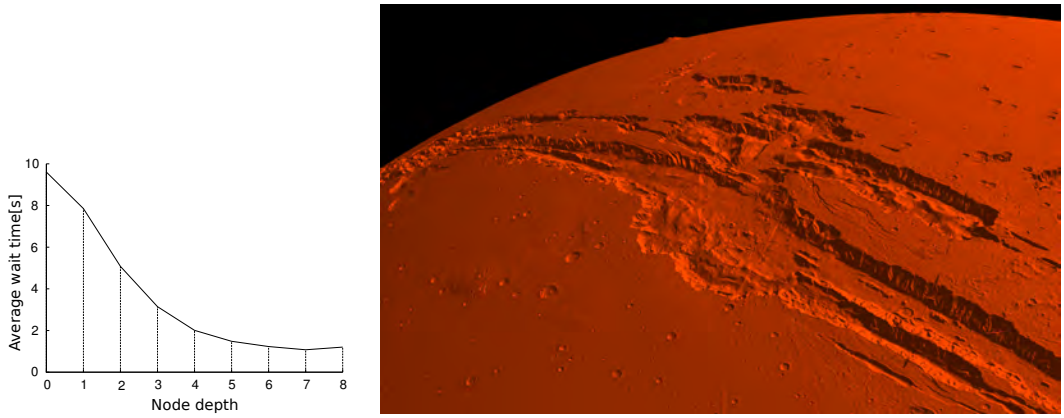


(d) Resolution level 3,(3.03 seconds later)

Figure 3.5: Top-down view of Valles Marineris showing convergence of image resolution.

Statistics revealed an average batch size of 27 tiles, meaning that for every 27 tiles produced one chunk had to be uploaded. Uploading a single chunk took 0.23 seconds on average, while resampling of all tiles in a batch required only 0.017 seconds. Bus bandwidth is therefore clearly a bottleneck in this system.

Considering the speed-up factor established in 3.4.1, the CPU implementation would be expected to require 0.47 seconds for processing a batch, which is still almost twice the time needed by the GPU solution. The break-even point of similar performance would then be expected at 13 tiles per batch.



(a) (b)

Figure 3.6: a) Average delay between the front-end issuing a tile request and receiving the data. b) Valles Marineris on Mars using both MOLA and HRSC datasets, composited on the fly. Image resolution: 1920×1200 pixels, 26.5 million triangles. Starting from an empty cache, image quality converges within 30 seconds.

3.5 Conclusion

I have presented a GPU-based approach to pre-process geo-referenced raster maps for spherical terrain rendering. Furthermore, the algorithm was extended to an online pre-processing scheme which enables interactive visualization of large datasets within minutes.

Key to my approach is the assumption that the input data is resident in host RAM. Constructing pre-filtered mipmaps of the input files on external storage as in [KLJ+09] would increase startup time of the system but allow for out-of-core visualization of larger datasets. Mipmap generation can be expected to be fast as it requires only linear I/O, as opposed to construction of a quad tree database.

Chapter 4

Spatial Analysis of Terrain in Virtual Reality

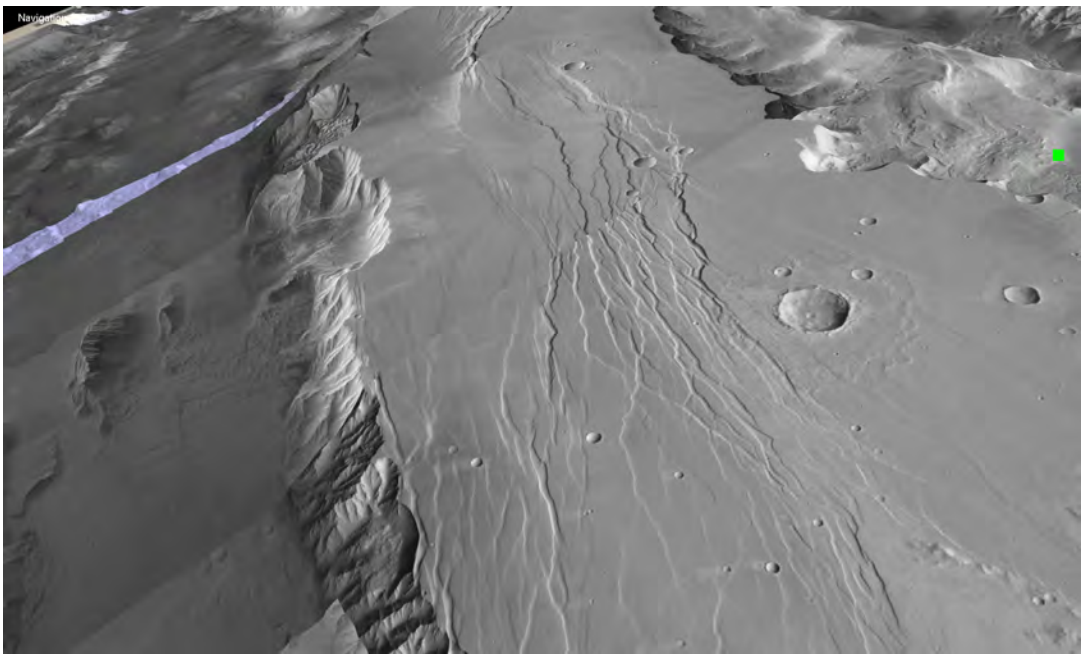


Figure 4.1: Fault network on Mars next to Valles Marineris

Measuring the geometric properties of topographic features is an important sub-task in cartography. Due to the widespread availability of remote sensing data such as Digital Terrain Models (DTMs), these measurements are nowadays almost exclu-

sively carried out in a desktop environment using Geographic Information Systems (GIS). Most GIS are limited to a top-down perspective which can lead to ambiguities in interpreting topography. Domain experts have reported, for example, that certain features like crater rims or mountain foothills covered by sediment can be difficult to locate accurately in traditional systems, but are more readily identifiable when using 3D visualization.

My contribution as detailed in this chapter is a virtual reality (VR) system for the spatial analysis of Digital Terrain Models (DTMs) based on an existing terrain rendering framework [WGH11]. I demonstrate that the hierarchical level-of-detail data structure used for rendering does support efficient implementation of analysis operators. As an example, I have implemented virtual tools for measuring height profiles and the volume of surface features such as mountains or craters.

Height profiles are an important geoscientific tool to help understand the morphology of predominantly linear surface features such as canyons, river beds or seismic faults (see Figure 5.6). A height profile is a cross-sectional representation of topography produced by sampling elevation along a given profile line. A common approach to analyze surface features is to place multiple profile lines at different significant points along a feature and to compare them within a single 2D plot.

The volume of surface features is an important quantity used in geoscientific reasoning, for example when trying to account for the volume of rock ejected from a crater during impact. Volume computation requires the definition of the 2D integration domain as well as a zero level surface. The signed volume (positive for mountains, negative for craters) is then computed by integrating the height difference between the elevation model and the zero level surface within the 2D footprint.

Using stereoscopic rendering and head tracking, our system improves depth perception of the measurement footprint lines with respect to the terrain, allowing for accurate placement. The system was applied to the surface of Mars using a DTM composed of Mars Orbiter Laser Altimeter (MOLA) [SZF+01] data by NASA as well as High Resolution Stereo Camera (HRSC) [GSP+10] data by German Aerospace Center (DLR).

As a reference data set I have used the high-resolution HRSC data set which provides a DTM of Mars with a resolution of about 50m per pixel. As the data capture and processing is still ongoing, only approximately 30% of the planetary surface is currently covered by HRSC. To provide a global model, the gaps in the HRSC model were filled by using the older MOLA data set by NASA, which provides a resolution of about 500m per pixel. The resulting composite multi-resolution data set is augmented by high resolution black and white imagery provided by the HRSC mission (12.5m per pixel). Note that the additional imagery is not considered by the spatial analysis operators but is used to provide visual cues for the user in identifying features.

4.1 Related work

Related literature includes the work by Kreylos et al. [KBB+06], which examines the efficacy of performing scientific analyses within VR environments. Cartography is given as an example, and from preliminary results, the authors conclude that mapping performance in their VR environment is higher compared to performing the same task within in a desktop GIS. Navigation within the terrain data set is performed by grabbing and manipulating the terrain using a 6DOF manipulator.

In the work presented here I have used a different navigation approach by moving the camera instead. This allows for a higher degree of immersion, as it is more similar to the user actually being present at the target site.

Not all data structures used by spherical terrain rendering systems are capable of supporting efficient spatial analysis queries. For example, the Crusta system by Bernardin et al. [BCK+11] and the Planetary Scale Composition approach by Kooima et al. [KLJ+09] use implicitly defined coordinate systems. A random point query against these data structures would require testing the boundary of each traversed node within the subdivision hierarchy for intersection with the given point.

My software extends my previous work on terrain visualization [WGH11], which used a data structure based on the HEALPix coordinate system by Górski et al. to provide spherical rendering while avoiding artifacts due to coordinate singularities.

At the same time, this data structure is well suited for interactive spatial analysis, as it provides a closed projection formula relating geographic points to parametric (database) coordinates.

4.2 Interaction

For interaction, a flystick is used to control a virtual pick-ray. Similar to the mapping system by Kreylos et al. [KBB+06], I have implemented separate navigation and analysis modes which the user can switch between using a button on the flystick.

For navigation, both a trackball and a free flight metaphor are provided. At planetary scales, the trackball metaphor was found to be an intuitive method for quickly navigating to sites of interest by grabbing a point on the planet and rotating it to a new position. Flying, on the other hand, is an immersive navigation metaphor useful to examine mostly linear features such as canyons or fault lines. The travel direction is determined by the orientation of the pick-ray. Additionally, a small analog joystick integrated in the flystick ("coolie hat") is used to provide pitch and yaw control.

Using the pick-ray, profile lines can be drawn directly onto the surface. Figure 4.2 shows the interactive placement of fault lines as the user manipulates the end point using the pick ray. As the profile line is swept across the topography it is updated in real time. The resulting 2D plots can be visualized within the system (Figure 4.3) or exported in standard formats for further analysis, e.g. using spreadsheet software.

To measure the volume of a surface feature, the user defines its footprint by drawing a polygon (Figure 4.5). The average surface height along the polygon defines the zero level for the subsequent integration, which executes concurrently with rendering in order to maintain interactivity.

The benefit of using Virtual Reality in this application is the accurate placement of height profiles and feature footprint polygons with respect to topography, enabled by stereoscopic rendering and head tracking.

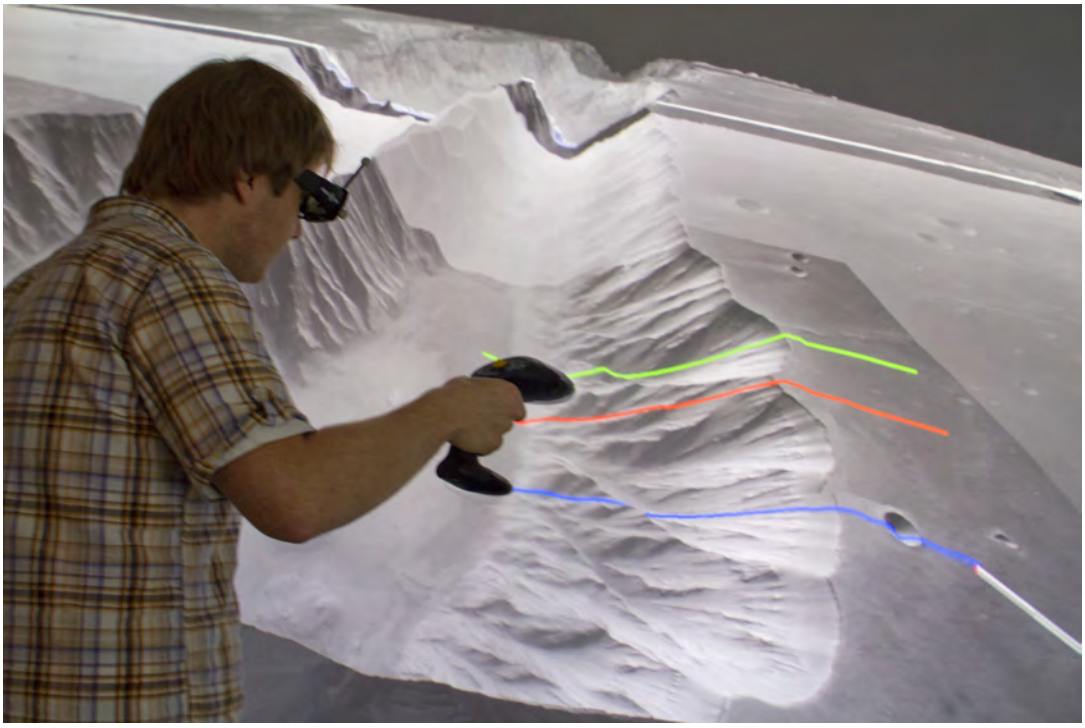


Figure 4.2: Accurate placement of profile lines in VR

4.3 Data structure

As described in [WGH11], the data structure used is based on the HEALPix [GHB+05] tessellation (Figure 4.4), which decomposes the sphere into 12 curvilinear patches. Our terrain rendering framework represents a digital terrain model using one quad tree for each of these patches. Tree nodes contain tiles of 255×255 height samples to allow for triangle batching in rendering.

The HEALPix coordinate system has two properties which make it particularly useful for spatial analysis: The equal-area property guarantees that all samples on a given resolution level (tree depth) represent the same physical area on the surface, which simplifies integration. Furthermore, it has a closed projection formula relating geographic and parametric coordinates. Compared to systems which use implicitly defined coordinates, for example by hierarchical subdivision of a platonic solid, HEALPix allows for more efficient point queries against the database.

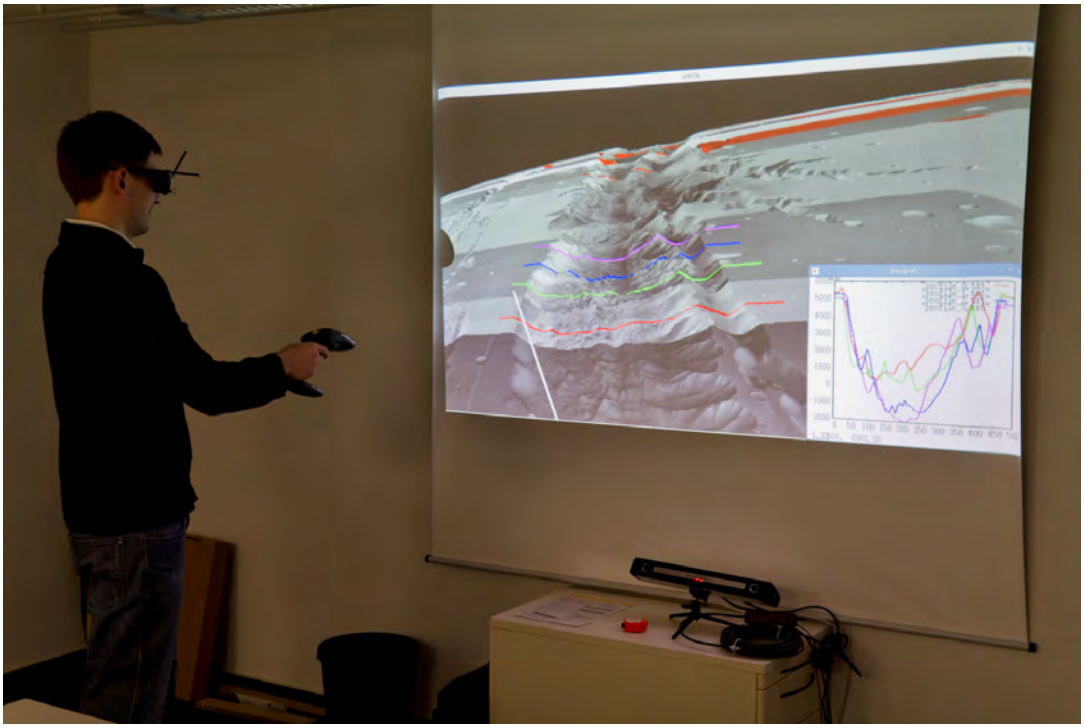


Figure 4.3: Simultaneous display of profile graph

4.4 Algorithms

The analysis algorithms which are presented in the following share the same level-of-detail database which is used by the underlying terrain renderer. Height profile and volume measurements are implemented using point and region queries against a quad tree, respectively. In both cases, selection of surface points is realized by using CPU based ray-casting to test the pick-ray against the visible subset of the hierarchical DTM.

4.4.1 Height profile

To compute a height profile, the DTM is sampled equidistantly along a line segment between two given geographic points. At planetary scales, where curvature becomes significant, the notion of a line segment as the shortest path between a pair of points has to be generalized to great circle segments, which are the shortest paths between

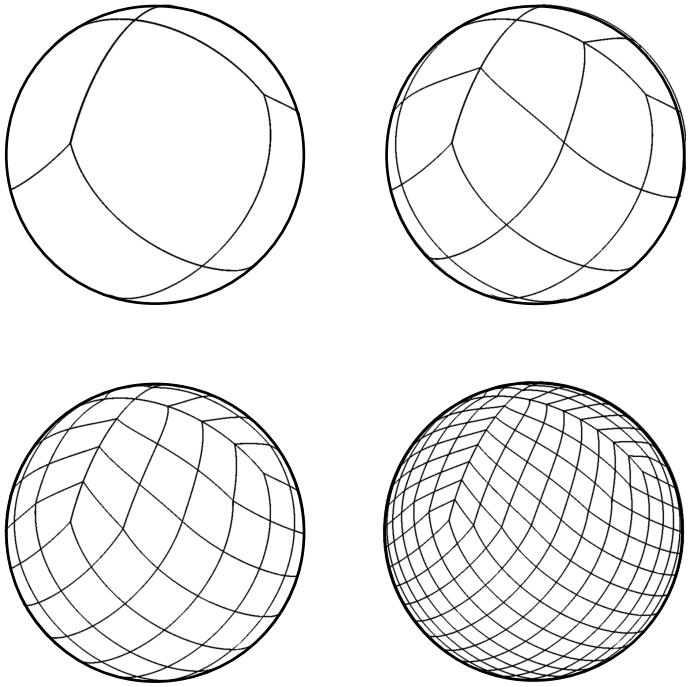


Figure 4.4: The HEALPix hierarchical sphere tessellation

two points on a spherical surface. A parametrization of the great circle segment connecting two points is given by spherical interpolation.

Note that while the end points of the profile line are specified in geographic coordinates, spherical interpolation uses three-dimensional euclidean coordinates. The intermediate sample positions are converted back to geographic coordinates.

To generate the height profile, each sample position is then converted to parametric HEALPix space. The quad tree selected by the coordinate is then traversed recursively to determine the leaf node containing the query point. Note that in the current implementation, this traversal is limited to the visible set, which is the subtree currently available in memory according to the level-of-detail metric employed by the terrain renderer.

The height value at the query point is then obtained by sampling the tile of height samples stored with the leaf node using bilinear interpolation.

As a profile line is being manipulated, the topography is sampled at 200 equidistant points along the line in each frame to provide an interactive feedback of the height

profile. Once the user accepts the position of the profile line, the system samples the topography again at a higher resolution (1000 samples) to improve the accuracy of the final result.

4.4.2 Volume measurement

For volume measurement, the given footprint polygon is first converted to parametric HEALPix space. A region query is then executed against the quad tree, recursively visiting all tree nodes which intersect the polygon.

When encountering a leaf node, the tile of 255×255 samples is tested against the polygon and those samples which intersect the polygon are integrated. This process is accelerated by using a scan line algorithm which considers each row of samples in turn. The footprint of a single row, which is a line segment in HEALPix space, is intersected with the polygon to obtain a set of intervals which are located within the polygon. The difference between each height sample and the zero level is computed and scaled by the surface area covered by the sample to obtain a volume contribution. This is simplified by the fact that due to the equal-area property of the HEALPix parametrization, the surface area represented by a given sample depends only on the depth within the tree and is independent of the geographic location.

Note that in the volume integration tool, unlike in height profile sampling, the tree traversal does not stop at the currently visible set of nodes. Instead, the on-disk database is traversed down to the finest level of resolution available to provide maximum accuracy.

To maintain interactivity, the integration was implemented to execute in parallel with the main render loop. A breadth-first traversal is used in which an active set of tree nodes to be loaded is maintained in memory. This set initially only contains the root node. Once a node has been loaded, the four child nodes are tested against the boundary polygon. Those child nodes which are found to intersect the polygon are added to the active set and scheduled to be loaded by the I/O thread. Leaf nodes which do not have any children are integrated into the total volume as described above. The volume computation is finished once the active set is empty.

To further improve interactivity, the number of nodes which are visited per frame can be limited. Empirical studies suggest that expanding 4 nodes per frame is sufficient to maintain good interactivity while still providing results within a short time frame (multiple seconds), even for large integration domains (e.g. Olympus Mons).

4.5 Results

The system was benchmarked on a workstation equipped with an Intel Xeon E5520 quad-core CPU, 24 GiB of RAM and a NVIDIA Quadro 6000 GPU. All examples were rendered at full-screen resolution of 1920×1200 pixels. In a full-screen view of Valles Marineris an idle frame rate of 68 fps was achieved for terrain rendering alone. While dragging profile lines across the canyon the system maintained interactivity at 38 fps.

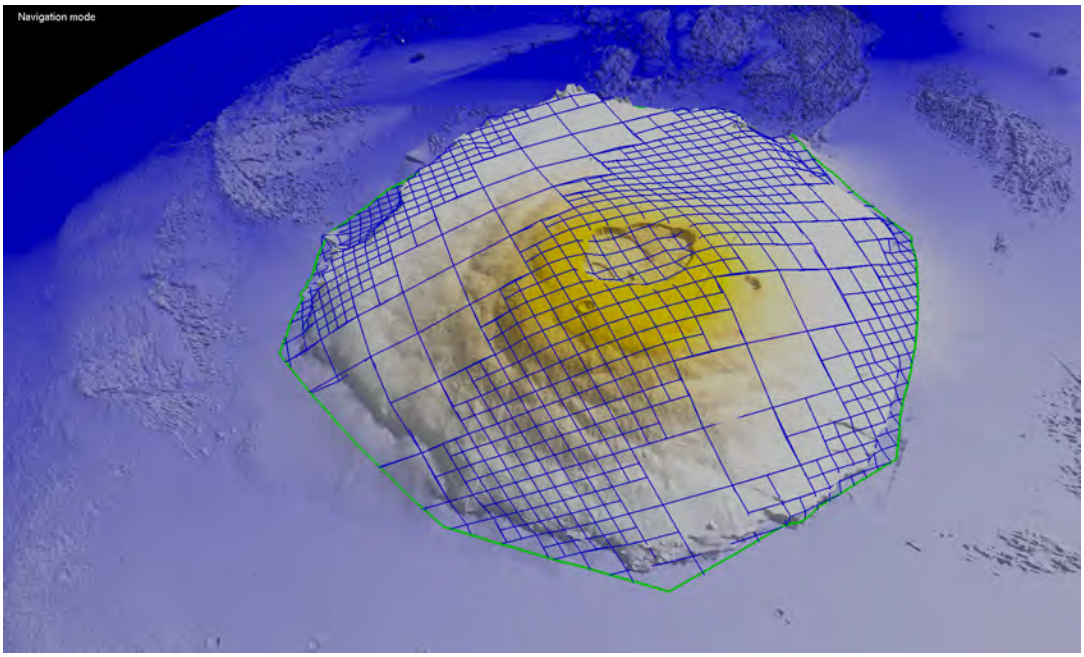


Figure 4.5: Measuring the volume of Olympus Mons

Figure 4.5 shows a view of Olympus Mons which was rendered at 69 fps. During volume computation, which took 10 seconds, performance dropped to still interactive levels at 40 fps. For illustration purposes, the visited leaf nodes of the terrain

database are outlined in blue. The difference in sizes between these nodes is explained by the fact that Olympus Mons is represented by mixed resolution data (MOLA and HRSC). A total of 1106 leaf nodes were integrated which required loading 274 MiB of data. A volume of 2.67×10^6 km^3 was measured which is in close agreement with literature [WGJS81].

4.6 Conclusion and future work

I have described a system for spatial analysis of digital terrain models in virtual reality environments. Using a large digital terrain model of Mars, I have demonstrated that existing level-of-detail data structures which are designed for terrain visualization can be used to sample height profiles interactively and perform volume measurements of very large surface features in a short amount of time while maintaining interactive navigation. The benefit of using virtual reality technology for these tasks was the superior perception of the location of measurements with respect to the topography being examined.

In future work, I would like to improve immersion by simulating a walk on the surface of Mars and providing more intuitive tools to enable virtual land surveying. One challenge that was identified during this study is that even high-resolution HRSC imagery does not sufficiently resolve features on the scale of a human, which implies that the ground below and in the close vicinity of the user would appear very blurry due to excessive magnification of the available imagery. This situation could be improved by introducing artificial detail, for example using normal maps based on procedurally generated noise. Care must be taken in this approach, however, to avoid distorting the actual data content.

Furthermore, I would like to examine whether haptic feedback can improve perception of topography. A desktop-based Phantom Omni device, for example, could be used for probing small to medium sized linear features such as the fault networks shown in Figure 5.6. Using the device for point selection in spatial analysis could be beneficial, as the perception of height and slope discontinuities might improve identification of the natural boundary of a surface feature such as the rim of a crater.

From a technical point of view it would be straightforward to provide contact information for a single point using a point query against the database. Haptic feedback, however, requires a much higher update rate (e.g. 5kHz) than interactive rendering which might pose additional challenges.

Chapter 5

Interactive Retro-Deformation of Terrain for Reconstructing 3D Fault Displacements

The present day topography of terrestrial planets is the accumulated result of a history of geological transformations. These processes leave observable marks on the landscape that are expressions of their interaction with the planetary surface. Geologists are interested in isolating and quantifying individual processes as well as determining the chronological sequence of their influence to help reconstruct the time line of geological evolution. Of particular interest to this goal are faults, which

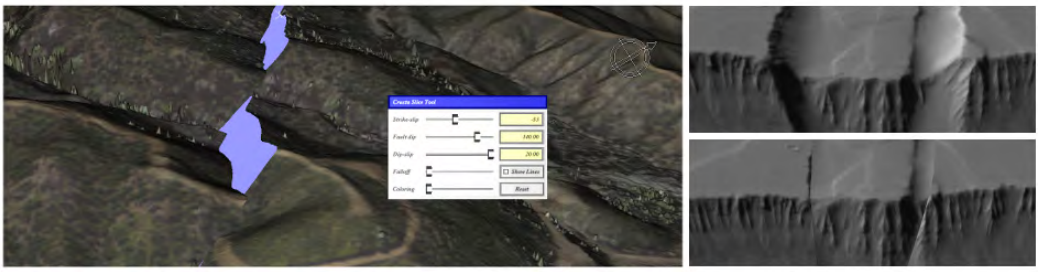


Figure 5.1: Left: Real-time deformation of a terrain model as fault parameters are changed. Right: Application of the system to reconstruct a graben structure on Mars (top: present-day state, bottom: reconstruction hypothesis).

are fracture surfaces within the planetary crust. Where these fractures intersect the surface, they form characteristic fault lines.

Individual earthquakes produce differential motion within the crust which can displace landforms a few centimeters to several meters along a fault line. Because these offsets are often well-expressed in topography, they serve as valuable anchor-points for unraveling the complexity of multiple interacting processes. To illustrate this phenomenon, consider a fault cutting across a river bed which is actively being eroded by a stream. As the two volumes of rock on opposing sides of the fault experience differential motion, the evolution of the river bed can be modified, halted, a new bed can be created or a combination of these effects can occur. Thus, distinct landforms are produced at different times, each recording a different amount of offset from which the time-evolution of the fault can be inferred.

Estimation of fault parameters is typically performed using a topographic map by identifying pairs of corresponding landforms which have been cut and displaced by the fault. The intersection points of these segments with the fault surface are called piercing points. The displacement accumulated along the fault since the feature has been cut can be quantified by the vector between these piercing points. A common approach to visualizing the pre-faulting state of the topography is to apply image editing software to raster maps. The map image is cut along the fault line and the two blocks are displaced according to the measured offset. This two-dimensional (2D) approach is limited, however, because it cannot properly convey the three-dimensional component of the displacement behavior.

My contribution is an interactive software system for modeling fault displacements within a 3D terrain visualization system where 3D navigation can properly reveal the spatial relation between fault geometries and the affected landscape. The terrain geometry is sliced along fault planes which approximate the fracture interface. The two resulting terrain blocks can then be interactively displaced relative to each other, which allows for a human-in-the-loop exploration of fault parameters by giving instant visual feedback on their effect on the topography. The demonstrated system enables geologists to:

1. explore the continuous evolution of a given retro-deformation rather than only its initial and final expression
2. visualize the 3D implications of the retro-deformation from any perspective to identify topographic variations not seen in a 2D plan-view
3. interactively perform three-dimensional oblique slip reconstructions by allowing arbitrary relative motion on the fault plane

The fault simulation framework presented in this work was implemented within two different terrain rendering systems: The Crusta Virtual Globe [BCK+11], which was used for the San Andreas study, as well as in another spherical terrain rendering system [WGH11] which was used for the Mars reconstructions. As all geometric operations are performed in shader programs which execute late in the rendering pipeline, porting the approach to both systems was straightforward.

Section 5.1 provides an overview of current methodology for exploring terrain retro-deformation. Section 5.2 describes our kinematic model for a single fault segment and Section 5.3 the resulting real-time visualization algorithm. In Section 5.4 we extend both the model and the visualization to incorporate a piece-wise linear segmented approximation to simulate curved fault geometries. We demonstrate the efficacy of our approach in Section 5.5 by applying it to parts of the San Andreas fault in northern California as well as to a graben structure in the Noctis Labyrinthus region on Mars.

5.1 Related work

The method traditionally used to visualize retro-deformation is based on cutting and displacing 2D maps (orthoimages) according to measured offsets between displaced surface markers, as exemplified by Frankel et al. [FBD+07]. A disadvantage of the 2D representation is that it can lead to ambiguities in the interpretation of such reconstructions as is demonstrated in Section 5.5.

Zielke and Arrowsmith [ZA12] presented the LaDiCaoz tool for semi-automatic lateral (strike-slip) fault offset computation based on digital elevation maps (DEMs).

In this system the user draws a fault trace onto a hillshaded image in plan view. The system then samples the elevation on both sides of the fault trace to produce a pair of height profiles. Keeping one side fixed, the algorithm laterally displaces the other profile to find the offset which provides the best match according to a distance metric. While this automation can reduce the time to find plausible fault offsets, the system supports only lateral offsets and provides only limited means for visual confirmation of such hypotheses. Results can only be visualized in plan view, which has the same drawbacks as the approach described previously. Furthermore, the authors report that because the algorithm requires all input data to be represented in memory, input DEMs can not be larger than about 10^8 grid points, which translates to a file size of 200 MiB for a typical 16 bit per sample representation. Larger files can be loaded but have to be downsampled in the process. In our approach, the underlying terrain renderer applies Level-of-Detail rendering and data streaming to overcome this limitation. This allows users performing a reconstruction to watch nearby marker sites for alignment at native resolution even for very large (e.g., $>$ 60 GiB) datasets.

An interesting geometric construction to describe the structural deformations due to fault movements was presented by Egan et al. [EKB+99]. However, the method has only been applied to small models and is not practical for deforming large DEM meshes interactively.

3D GeoModeller is a software package for recovering the subsurface fault configuration based on primary geological observations such as drill hole experiments, seismic and gravimetric data. Using geophysical inverse theory and a human-in-the-loop approach, the software attempts to reconstruct the most probable fault configuration consistent with the sparse observations that are available. The system focuses on recovering the present day subsurface fault geometry and does not consider the time evolution of fault slip, however. The functionality which my system provides to recover fault geometries from their surface expression could potentially be used to constrain and refine geological models constructed with this software.

Numerical simulations of faulting mechanics, which is a frictional contact problem, have been explored extensively [Bir78, XMM07]. Based on volume mesh representations of fault planes and rock layers (strata), static finite element (FE) models

are able to simulate stress distribution in the Earth's crust. In recent years, these approaches have been extended to simulate fault motion by using dynamic FE models [LL06]. However, while FE methods are useful to validate individual hypotheses about fault geometry or material composition in the subsurface, they are still too computationally intensive for interactive simulation of faulting in 3D space.

Van Aalsburg et al. [VAYK+10] have demonstrated a system to interactively edit fault maps in a virtual reality (VR) environment. Their work focuses on using VR methods to improve perception of surface features and enable users to create better maps of large fault networks, which can then be used as input to FEM based solvers. While my system also provides functionality to trace fault lines directly on top of a DEM visualization, our goal is interactively simulating the retro-deformation of these faults to measure and verify offset parameters.

Mesh deformation techniques which use control points or handles to control the deformation of an existing object are well established in geometric modeling literature [SP86, Coq90, BK05]. A common goal in many of these approaches is to localize the influence of control points to provide intuitive control about the deformation. The design of my kinematic model is based on a similar notion, namely that trajectories of points close to a fault line segment should be tangential to the segment.

Von Funk et al. [vFTS06] presented an interactive mesh deformation technique that transforms the vertices of a triangle mesh by GPU-based pathline integration along a synthetic vector field which can be manipulated using different modeling metaphors. My displacement strategy can likewise be cast as vector field integration on an implicit field which is constant within the influence zones of the individual fault line segments.

Correa et al. [CS07] have demonstrated an interesting approach to rendering deformed meshes using a signed distance map representation and ray-casting. The necessary warping function for fault-retro-deformation could certainly be computed procedurally, but an implementation would essentially imply development of an efficient ray-casting based terrain render. This is complicated by the fact that the actual path of the rays is only computed on the GPU, which would require random access to the underlying level-of-detail representation.

Bruyns et al. [BSM+02] provided a survey on existing interactive mesh cutting methods and categorize them according to how the cut path is defined, whether intersected primitives are simply removed or re-tessellated and by the number of new primitives generated during re-meshing. In their categorization, my system uses a cut path defined by a template (the fault line). Intersected primitives are re-meshed and the number of newly generated primitives is minimal - new vertices are only inserted at the intersection points between the fault line and the terrain geometry. Following the survey, the authors present a VR system which uses collision detection and haptic feedback to simulate surgery using a virtual scalpel. The performance of this system is based on the assumption that cutting happens incrementally as primitives are intersected by the scalpel. Under this assumption, it would be difficult to enable interactive modification of the cut path, as is possible with my approach.

Another surgery simulation system was presented in [SSE04], in which triangle meshes that intersect with a virtual scalpel are split and re-tessellated and a triangulation of the cutting plane is added to represent the exposed tissue. Free-form deformations are then applied to the sub-meshes on either side of the incision to simulate tissue behavior. My approach transforms the terrain mesh in a similar way by cutting triangles along a fault line, adding a triangulation of the fault plane and applying a deformation to the geometry on one side of the cut.

5.2 Fault Model

As illustrated in Figure 5.2, the standard model for faults [TM92] approximates the geometry of the associated rock fracture with a single plane. This partitions the crust into two rigid blocks for which the relative motion is restrained to a linear displacement parallel to this fault plane. The corresponding displacement vector is called slip.

To match geoscientific terminology, this slip vector is decomposed into so-called strike and dip components. The strike is the projection of the slip onto the locally horizontal plane and relates to the azimuthal component of the displacement whereas

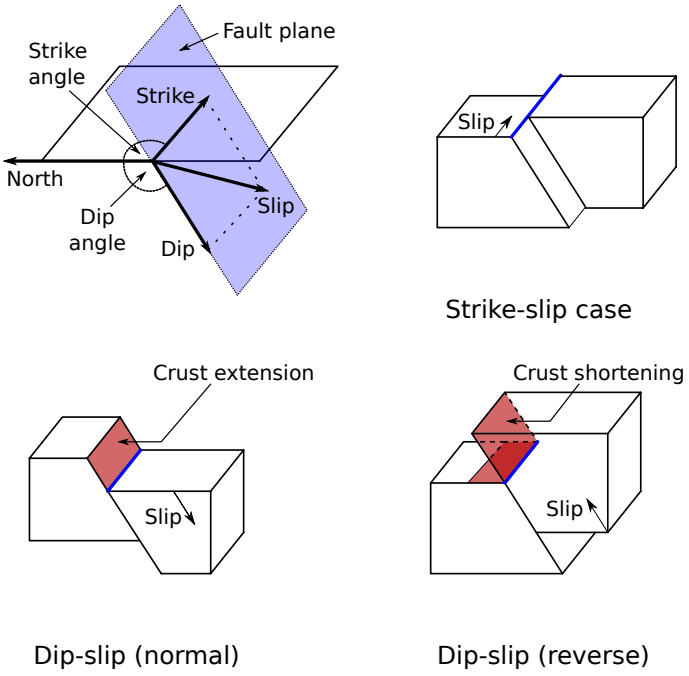


Figure 5.2: Fault model and kinematics of typical displacements.

the dip is the projection of the slip onto the vertical plane and relates to the slope component.

Faults are typically classified based on their strike and dip such that faults with predominantly horizontal slip are categorized as strike-slip, while vertically dominant cases are referred to as dip-slip. The latter are further subdivided into normal and reverse faults. A normal fault is characterized by the hanging wall moving downward relative to the footwall, exposing rocks which were previously located underneath the surface. Normal faults can also occur as parallel pairs, in which case the enclosed block of land which is downthrown is referred to as a graben. The opposite case to normal faults are so-called reverse faults. These are characterized by the hanging wall moving upward and are accompanied by a shortening of the crust.

In my system, fault geometry is defined by tracing fault lines directly on the visualized topography. While specifying the dip angle using a slider, the resulting fault planes are visualized to support the user in matching the slope of a fault scarp.

5.3 Visualizing planar fault displacement

In the following we shall consider the case of a planar fault with a single fault plane, which is the approximation generally used in practice. In the kinematic model I have chosen, the fault plane bisects the terrain into a fixed and a moving fault block. The moving block experiences a constant linear translation which is the superposition of displacements in the fault strike- and dip-directions. The strike direction is given by the orientation of the fault line while the dip direction corresponds to motion along the slope of the fault plane, perpendicular to both strike direction and plane normal. The amount of displacement in each direction is adjusted using two sliders, which allows the user to explore the continuum of possible offsets while observing the correspondences between landforms on opposing sides of the fault plane. Note that within this model, canceling out the effects of fault motion is equivalent to simulating the forward progression of a fault with reversed displacement directions.

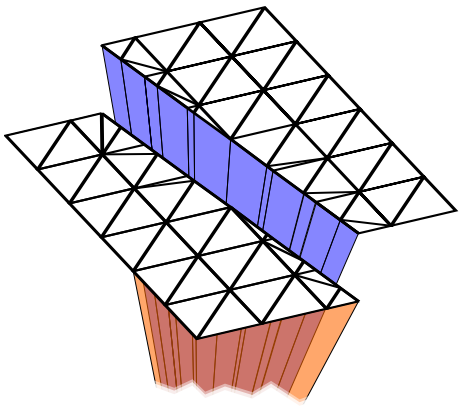


Figure 5.3: Surface mesh is cut and re-tessellated along the fault, triangles representing the fault plane are inserted (blue and orange) and the moving block (right) is displaced.

To apply this model to a triangle mesh and maintain the solid appearance of the individual fault blocks, it is necessary to bisect the mesh along the fault plane and close the resulting holes in both sub-meshes (Figure 5.3). Bisecting the mesh requires re-meshing of all triangles which are intersected by the fault plane, producing three sub-triangles each. In order to close the resulting holes at the interface of the two sub-meshes, a tessellation of the new mesh boundary is generated, which is a subset

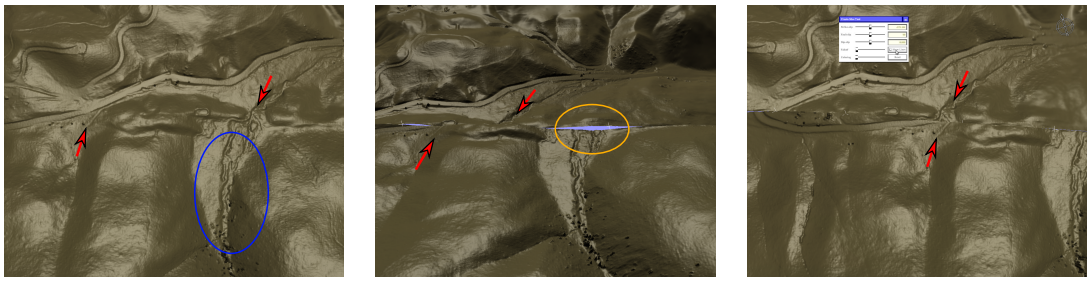
of the fault plane. For each segment of the mesh-plane intersection curve, a triangle is produced which connects the segment with the point on the fault plane closest to the planet's center. This triangle is duplicated with opposite vertex ordering to provide correct, front-facing surfaces on both fault blocks. The triangles are colored, as the exposure of the interface due to translation can provide valuable indicators for the geologist (see Section 5.5). Finally, the strike and dip displacement translation is applied to those triangles associated with the moving block.

To allow for interactive modification of all fault parameters, including the fault line geometry, I have chosen to implement these operations as part of the rendering pipeline by using a geometry shader. Note that a straightforward implementation of these geometric manipulations can easily lead to artifacts, as the large difference in scale between the planet's radius and individual triangle sizes leads to numerical instabilities due to cancellation effects when using single-precision arithmetic. In tile based terrain rendering systems, these problems can be avoided, however, by performing all operations in a coordinate system local to the tile currently being rendered. The final translation which moves the patch to its world space coordinates can in this case be absorbed into the model-view transformation.

To ensure high visual fidelity of those parts of terrain that are shifted into view by the applied translations, it is necessary to adapt the level-of-detail metric used by the underlying terrain rendering scheme to compute the necessary terrain resolution. For schemes which (recursively) evaluate the screen-space projected area of terrain patches, each patch can be tested against the fault plane and if it is found to be contained in the moving block, the translation can be considered in the evaluation of the LoD-metric by applying it to the bounding volume being considered. For patches which are being intersected by the fault plane, the maximum screen area for both the original and translated position of the patch should be used instead.

5.4 Visualizing segmented fault displacement

In contrast to the standard model, naturally occurring fractures are often non-planar and can exhibit slight curvature at large scales. In the following, the kinematic



(a) present-day (b) intermediate (c) reconstructed

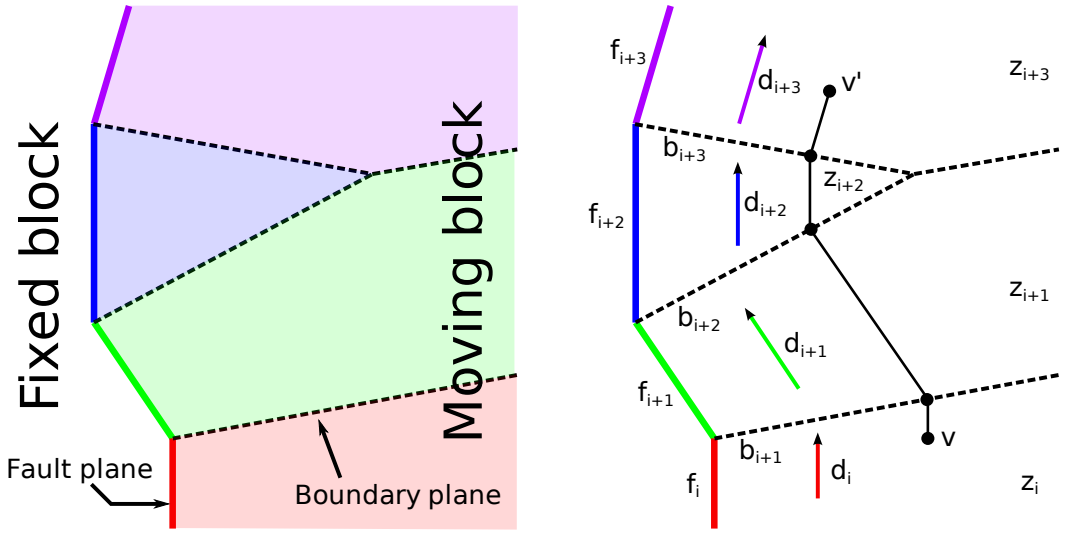
Figure 5.4: Exploring incremental reconstructions of a feature offset by ≈280 m (red arrows). A shutter ridge appears at ≈175 m (circled in orange), implying a formation age of the alluvial fan (circled in blue).

model for planar faults described in Section 5.3 is extended to a piecewise planar formulation to enable simulation of these cases by allowing curved fault traces to be approximated using multiple planar fault segments. Based on the assumption that points close to the fault curve should move tangentially to it, I define a displacement field that is tangential to offset surfaces of each fault plane. This field is constructed by subdividing the moving block into disjunct zones of constant displacement direction. Subsequently, the deformation algorithm is extended to implement the refined model in real-time using a ray-casting scheme.

5.4.1 Definition of influence zones

With each fault plane I associate a zone of influence within which fault displacement occurs strictly parallel to the plane. Between each pair of influence zones, a boundary plane is introduced which explicitly represents the zone interface. These boundary planes bisect the angle between the neighboring fault planes (Figure 5.5(a)).

For the first and last fault plane, the boundary plane corresponding to the missing neighbor is omitted, which implies that the influence zones extend indefinitely in that direction. If only a single fault line segment is specified, its influence zone equals the half-space defined by the fault plane which is equivalent to the planar model.



(a) Segmentation (b) Displacement algorithm

Figure 5.5: (a) The moving block is segmented into influence zones by boundary planes which bisect the angle between fault line segments. (b) A mesh vertex v is advected until it has travelled a distance equal to the desired offset.

5.4.2 Displacement algorithm

In the following I will present a modified displacement scheme (Figure 5.5(b)) which supports segmented fault lines. Let z_i be the influence zone bounded by the fault plane f_i as well as the pair $(b_i,\ b_{i+1})$ of boundary planes. Furthermore, let d_i be a unit vector parallel to the corresponding fault line segment.

The algorithm begins by classifying each mesh vertex according to the influence zone within which it is located. The influence zone is determined by perfoming a signed distance test against the triplet of planes bounding each zone. The mesh is then cut and closed as in the planar case by intersecting each triangle within a zone z_i against the corresponding fault plane f_i.

Vertices within the moving block are then advected parallel to the fault line by the user-specified positive strike-slip offset s. A vertex v within a zone z_i travels in direction d_i until it leaves the zone, which occurs at the intersection with the boundary plane b_{i+1}. To obtain the distance t to this point we compute the intersection of the ray (v, d_i) with b_{i+1}. If this distance is larger than s, the vertex is translated

along d_i by s and the algorithm ends. Otherwise, the vertex v is advanced to the intersection point, the remaining offset distance s is decreased by t and the process is repeated with the next zone z_{i+1}.

Note that since vertices always move parallel to fault line segments, they can never intersect the fault planes f_i. Furthermore, for positive strike-slip offsets, vertices always visit the influence zones in increasing order, therefore only the boundary planes b_{i+1} needs to be considered for intersection. In the case of negative offsets, zones are visited in decreasing order and the planes b_i are tested instead.

The requirement to conditionally generate new geometry again suggests implementing this algorithm using geometry shaders. However, as geometry shaders operate on individual primitives, vertices shared by multiple triangles would be processed redundantly. In order to avoid this overhead, the computation of the initial influence zone as well as the advection step was instead performed once per vertex using a vertex shader, passing the result (the zone index) to the geometry shader. For the majority of triangles that are not intersected by any fault plane, the geometry shader can simply pass these values through to the next pipeline stage. For intersected triangles, newly generated vertices still have to be advected by the geometry shader, however.

5.4.3 Graben reconstruction

The displacement algorithm allows for the simulation of multiple faults, given that the resulting influence zones (compare Figure 5.5(a)) do not intersect. This enables the reconstruction of graben, which are depressed regions bounded by a pair of parallel normal faults (Figure 5.6). This application requires, however, that the surrounding upland is defined as moving blocks while the depressed region is kept fixed.

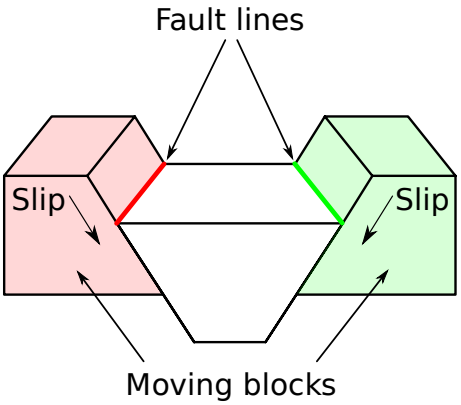


Figure 5.6: Placing multiple fault lines to reconstruct a graben.

5.5 Results

In the following, I will demonstrate the results obtained while applying the system to two different data sets. The first one is a subset of the data gathered by the B4 LiDAR project which represents a ≈ 100 km long segment of the San Andreas Fault. The data was processed to a raster DEM at ≈ 1 m resolution and transformed into a 5.2 GiB hierarchical database for Crusta.

The second example uses the full HRSC [GSP+10] level 4 archival dataset which provides a DEM of one third of the surface of Mars at an average resolution of 90 m (24 GiB) as well as grayscale imagery at a resolution of 16.5 m (386 GiB). These were processed to level-of-detail databases with a size of 121 GiB for the DEM and 410 GiB for the imagery, respectively.

5.5.1 San Andreas Fault

The San Andreas Fault (SAF) is a 1,300 km-long, active, right-slip fault that defines the boundary between the Pacific and North American plates. I will demonstrate the utility of my approach by mapping and retro-deforming a section of the SAF located within the Mojave Desert, north of Los Angeles in southern California.

In particular, the system capabilities mentioned in the introduction will be justified with a set of remote geologic investigations done along the fault section: reconstructing both strike and dip offset of a stream channel; discovering multiple offset reconstructions through effective multi-scale retro-deformation; and inferring the probable age at which an alluvial fan could have formed through 3D exploration of intermediate reconstructions.

3D strike and dip reconstruction (Figure 5.7). 3D visualization of the fault displacement can produce important insights for the interpretation of a hypothetical reconstruction. Panel 5.7(a) shows a plan view of a stream channel, flowing towards the bottom of the image, which has been offset by a fault, as indicated by a sudden right step in the stream channel where it crosses the fault. Panel 5.7(b) presents a hypothetical reconstruction undoing the offset achieved by applying a left-lateral strike-slip displacement of ≈80 m. Note that equivalent results can be obtained using established methods of cutting and re-arranging orthoimages. Using 3D navigation, however, the user can produce an oblique view of the site (Panel 5.7(c)) which reveals that the reconstruction is incomplete. In particular, it reveals a topographic high that is clearly visible due to the exposure of the blue-colored fault plane. The presence of this blue plane indicates that further geological processes must have been involved to explain the height difference. One potential explanation is the presence of a dip-slip component to the fault, compensating for the vertical offset. Applying a dip-slip offset of ≈5 m to compensate yields the result shown in Panel 5.7(d). As the dip-slip component of the reconstructed channel is restored, some sections along the fault might no longer align in terms of vertical displacement. However, this can be considered geologically reasonable for two reasons: 1) there can be localized dip-slip movement along the fault, and 2) sections of the fault adjacent to a retro-deformed channel represent different times in history. Thus, a topographic high further east could be eroded during the time it takes for the slip along the fault to displace it to the reconstructed location. Such considerations are key to geologic studies and can be interactively explored with the proposed system.

Multi-scale retro-deformation. In a typical session with the system, a geologist would: 1) map the trace of the fault along a significant portion of the study area using our multi-segmented representation; 2) identify a single landform that appears offset at a detailed scale; 3) restore that feature which globally retro-deforms the terrain along the entire mapped section; and 4) subsequently explore the landscape, alternating between overview and detailed scales, in search for other markers along the fault which would also have been restored in support of the initial reconstruction. To demonstrate this approach, consider the effects of the previous reconstruction on nearby topography. The top panel in Figure 5.8 gives an overview of a segment of the San Andreas fault with the site previously reconstructed (Figure 5.7) highlighted (orange box). Close examination of the deformed terrain along the fault trace revealed other sites which were reconnected due to the applied offset, two of which (circled) are shown in detail in the bottom panels in their pre- and post-offset state. These observations provide strong visual evidence for a plausible offset distance that was initially only locally justified. Note that these findings would be difficult to reproduce using traditional image editing software because the size of an orthoimage required to adequately represent the terrain at a detailed scale everywhere along the segment would be prohibitively large.

Inferring geologic history (Figure 5.4). For an offset reconstruction to be both reasonable and complete, geologists must be able to describe the sequence of geologic events that occurred during formation and progressive displacement of the offset landform. The ability to interactively evaluate intermediate steps during interactive retro-deformation is critical in establishing such a geologic history. Panels 5.4(a) and 5.4(c) show a right offset channel at present-day and a predicted reconstruction ($\approx$280 m), respectively. Panel 5.4(b) shows an intermediate step of the retro-deformation as observed during interactive exploration. In this view an uphill facing topographic high (blue surface in the figure) is found to block the uphill stream channel. This denotes a time in history at which sediment in the channel was able to pool up against the ridge, thus creating an alluvial fan which can be seen in the present day. By taking into consideration both the average slip rate along the SAF and the offset distance at which this shutter ridge forms, a potential age for

the formation of this alluvial fan can be determined.

5.5.2 Noctis Labyrinthus (Mars)

Products of tectonic activity can be observed on all terrestrial planets in the solar system. Strike-slip faults are not as prominent as they are on Earth, however, where they are often linked to plate tectonics. Faults on Mars [GP10], for example, are almost exclusively of the dip-slip (normal) type, even though evidence for strike-slip faults exists [AHZI08].

The Noctis Labyrinthus region of Mars is located at the western termination of the Vallis Marineris canyon system. It is a complex network of canyons and plateaus formed by grabens and landslides. Figure 5.9 shows a graben structure in which the surface was displaced along a pair of dip-slip faults with sub-parallel orientation. The resulting fault scarps, which dip toward the center of the graben in a V-shape, are marked on each side of the graben (top-left). This site also features a collapse on the right hand side which produces a cliff that intersects the graben. This coincidence allows for a unique cross-sectional view along the graben (top-right), which clearly shows the elevation difference produced by faulting.

To reconstruct this graben, a pair of segmented fault lines is drawn in a top-down view by tracing the foothill of both normal faults (top-right). Using an oblique perspective, the dip-angle of the fault planes is then set to match the slope of the fault scarps (bottom-left). This is supported by visualizing the fault plane orientation (blue) along each fault line segment as the planes are rotated. Going back to the cross-sectional view, the retro-deformation is then performed by applying increasingly larger dip-slip offsets until the valley of the graben is lifted up to match the elevation of the surrounding upland (bottom-center). The top-down view of the reconstructed result (bottom-right) reveals the simultaneous reconnection of another linear fault intersecting the graben (orange), which supports the reconstruction hypothesis. A dip angle of $\approx 36°$ and an offset of $\approx 800\,\mathrm{m}$ along dip was measured, which corresponds to a vertical elevation difference of $\approx 470\,\mathrm{m}$ between valley and upland.

5.5.3 Practical impact

Geoscience experts who applied the system to study segments of the San Andreas fault reported that using the retro-deformation tool to perform geologic reconstructions significantly improves upon the traditional image editing and other 2D approaches such as LaDiCaoz [ZA12] by making it possible to see the full 3D geometry of the reconstructed site during restoration, thus evaluating the feasibility of the reconstruction and understanding the detailed sequence of erosional and depositional events predicted by the reconstruction. The tool has proven essential for identifying and evaluating potential landform offsets by allowing the geoscientists to interactively restore one offset while simultaneously watching for other features to become aligned. Analysis of large swaths of LiDAR data is possible using Crusta because it supports interactive visualization of large (e.g., > 60 GiB) datasets. As such, using the retro-deformation tool with Crusta enables rapid characterization of the range of offset magnitudes permitted at a given site, and thus can be used to quantitatively evaluate the offset uncertainty.

The system was essential for a specific research project along the San Andreas fault, enabling geoscientists to remotely discover, evaluate and reconstruct a total of 60 offsets along a 100 km long section of the fault represented by a ≈ 3.4 GiB DEM dataset. These offsets range from 20 to 261 m and largely infill a number of gaps in previously reported data. The discovery of so many new potential slip-rate sites along this long-studied section of the fault represents a major advance and created important opportunities for resolving several long-standing disputes over the slip rate and mechanics of this important fault system.

5.5.4 Performance

The system was benchmarked on a workstation equipped with an Intel Xeon E5520 quad-core CPU, 24 GiB of RAM and a NVIDIA Quadro 6000 GPU. All examples were rendered at full-screen resolution of 1920×1200 pixels. Measurement results shown in Table 5.1 demonstrate that interactivity was maintained in both application examples for scene complexities larger than 1 triangle per pixel.

Table 5.1: Rendering performance.

Figure	# triangles (million)	frames/second
SAF, overview	3.3	40
SAF, zoomed in	2.9	50
Mars, top-down	4.6	32
Mars, cross-section	5.1	28

Figure 5.10 shows the scaling behavior with respect to the number of fault line segments, measured in the top-down view of the Mars example by gradually extending a linear fault line. When using up to 16 line segments performance stayed approximately constant. Beyond this number performance decreased in a linear fashion. While the initial plateau seems be to be a hardware dependent artifact, the subsequent linear behavior can be explained by the fact that the algorithm we presented has linear complexity with regard to the number of fault segments. For the examples presented here only a maximum of 14 fault plane segments was ever used, therefore we believe that this decrease in rendering performance is not of much practical relevance.

5.6 Conclusion and future work

I have presented a system for interactive retro-deformation of faulted topography within a 3D terrain visualization system. The technique maintains interactivity by performing all computationally expensive transformations in real-time on the GPU, leveraging geometry shader programs to handle topological changes in the transformed mesh. To simulate nonplanar faults, I have constructed a kinematic model and implemented it on the GPU using a ray-casting approach.

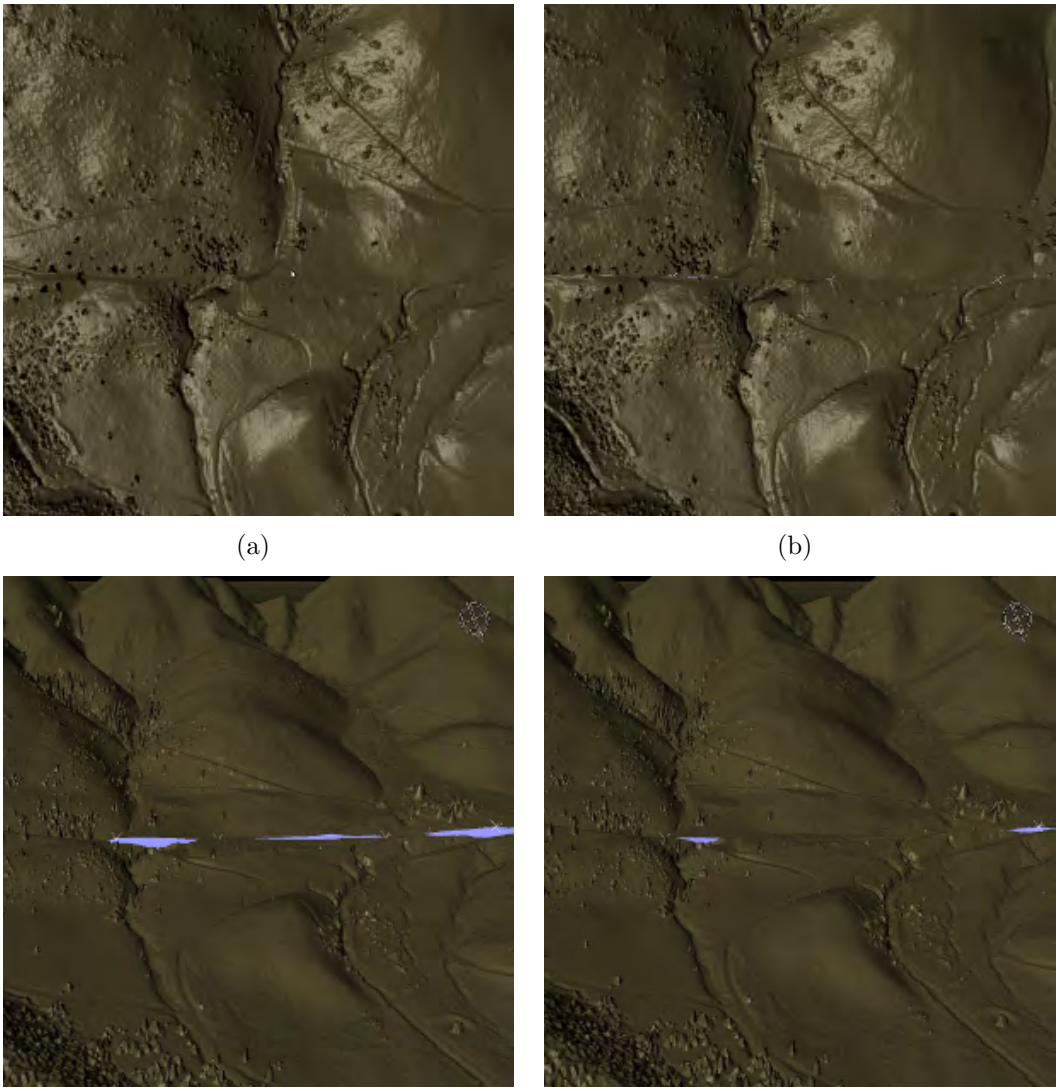
Using the system, fault displacements can be estimated rapidly in a human-in-the-loop scheme, where topography is examined for the alignment of displaced surface features as the terrain is interactively displaced. By showing an oblique view of a feature which has previously been reconstructed in a plan view, I have demonstrated

that 2D approaches are limited as they cannot convey all geometric implications of a given reconstruction.

To show the efficacy of the technique, it was applied to segments of the San Andreas fault, demonstrating the restoration process of a single surface feature displaced by a strike-slip fault. By fixing this offset and inspecting several nearby marker sites, additional visual evidence for a correct reconstruction could be found. Furthermore, I have demonstrated the retro-deformation of a graben on Mars which had been formed by a pair of dip-slip faults. Again, the reconstruction hypothesis was supported by the reconnection of a nearby marker site.

Note that for curved faults, a fault block whose linear motion is obstructed by a fault segment curving towards the block would in reality be expected to produce a vertical component of motion, resulting in crustal shortening as it is pushed upwards over the blocking segment. In the opposite case where the fault curves away from the moving block, crustal extension would be expected. These effects are not currently captured by the kinematic model for curved faults.

Concerning future research possibilities, I would like to refine the kinematic formulation to account for this crustal shortening and extension in response to curves in the trajectory of a slip-strike fault. Furthermore, I want to explore the possibility of simulating large networks of intersecting faults, which is expected to require spatially limiting the influence of individual faults as well as the application of acceleration structures to improve the performance of spatial queries. Furthermore, it could be considered to incorporate the LaDiCaoz [ZA12] approach to semi-automatically provide the user with a set of plausible fault offsets, the consequences of which could then be explored interactively using our system.



(a) (b)

(c) (d)

Figure 5.7: Reconstruction of an offset channel. (a) plan view present day, (b) plan view reconstructed, (c) oblique view reveals obstructing topography, (d) applying dip component to compensate.

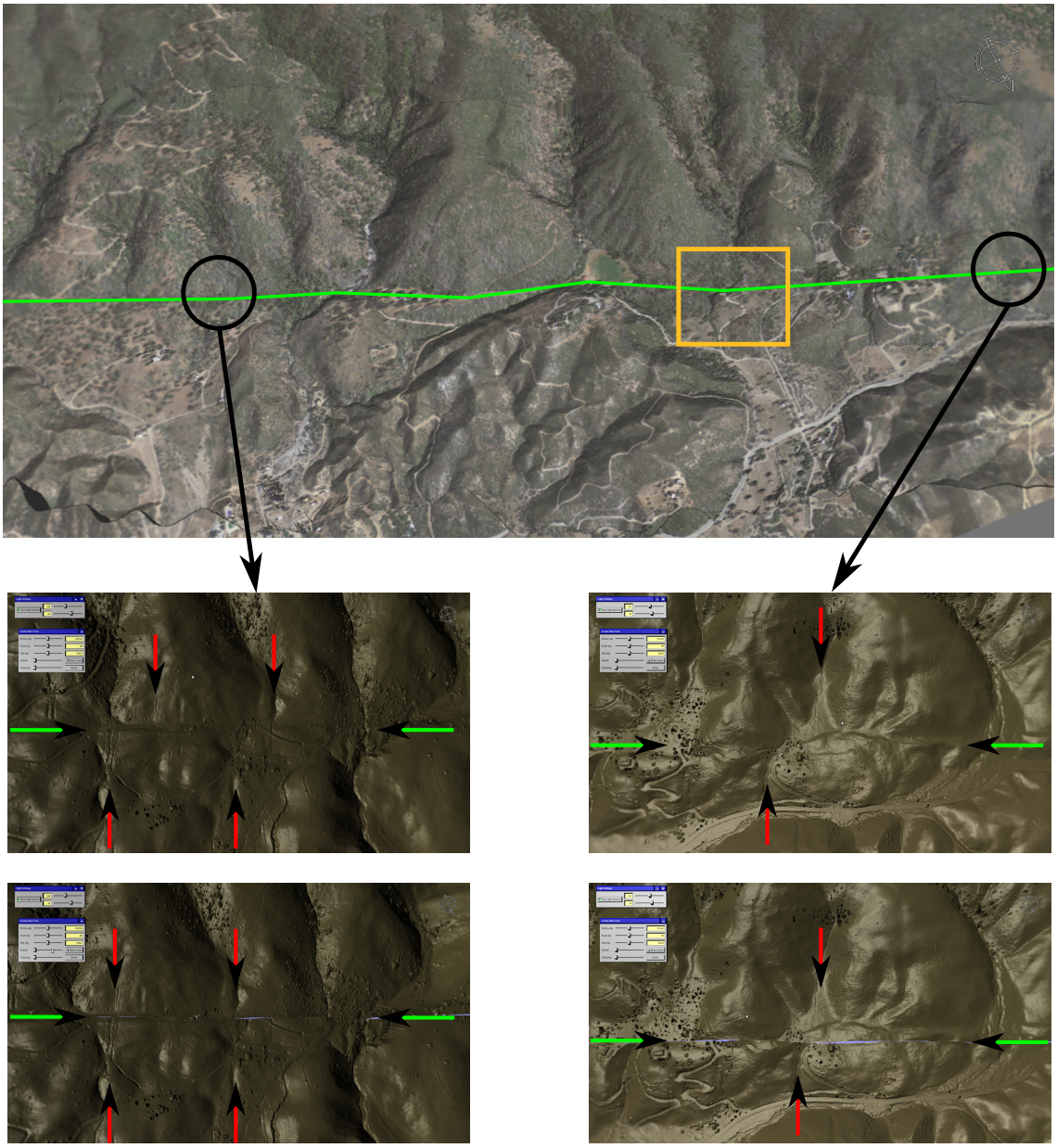


Figure 5.8: Interactively reconstructing a section of the San Andreas fault. Views for all panels are oblique and approximately to the south. Top: present-day overview of the fault section. The orange box highlights the site which was reconstructed in Figure 5.7. The fault trace is indicated by the green polyline. The circles represent additional sites that were found to exhibit connected channels due to the ≈80 m reconstruction. Bottom: top panels for each vertical pair (left and right) show present-day geometry of offset channels (red arrows). The bottom panels of the pair show how the connectivity of the channels is restored after the retro-deformation (indicated by alignment of the red arrows).

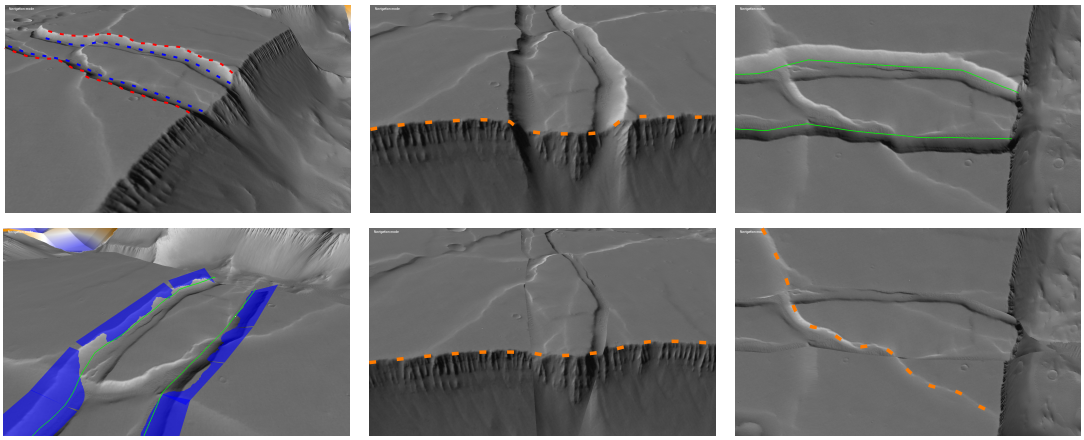


Figure 5.9: Interactive reconstruction of a graben on Mars. Top-left: Fault scarps marked in blue and red. Right side of image shows region where terrain has collapsed. Top-center: Cross-sectional view of the graben from inside the collapse. Height profile (orange) along the rim of the collapse shows elevation difference in graben region. Top-right: Traced the foothill of the fault scarps on both sides of the graben. Bottom-left: Rotating the fault planes (blue) to match the slope of the fault scarps. Bottom-center: Cross-sectional view after reconstruction. Bottom-right: Top-down view after reconstruction. Note reconnection of intersecting fault line (orange).

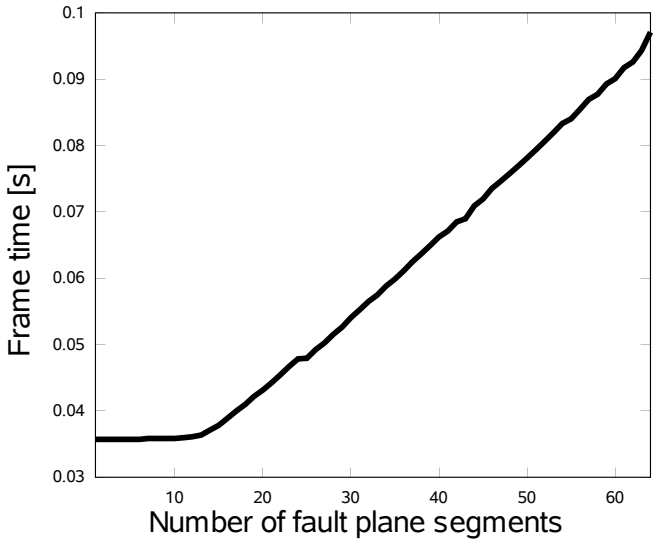


Figure 5.10: Scaling behavior: Relation between number of fault segments and rendering time in Mars top-down view.

Chapter 6

Exploration of Time-dependent Paleoceanographic Flow Data in Virtual Reality

Palaeoceanography is the study of the history of the oceans, including the evolution of ocean currents. Sediments which are transported along these currents eventually settle on the ocean floor, forming a deposition record which can be sampled using drilling or coring techniques. By studying the composition of the extracted samples, scientists attempt to infer the historical flow patterns which have produced these observations, which is a challenging inverse problem with a set of uncertain parameters and boundary conditions. I have developed a stereoscopic visualization system for interactive exploration of global, time-dependent flow datasets in combination with deposition records. The purpose of the software is to enable domain experts to analyze the output of ocean flow simulations and support the refinement of numerical methods and the validation of parameter choices.

6.1 Introduction

The flow patterns of the Earth's oceans change over long time scales. Sediments which record seawater properties fall through the water column and form layered

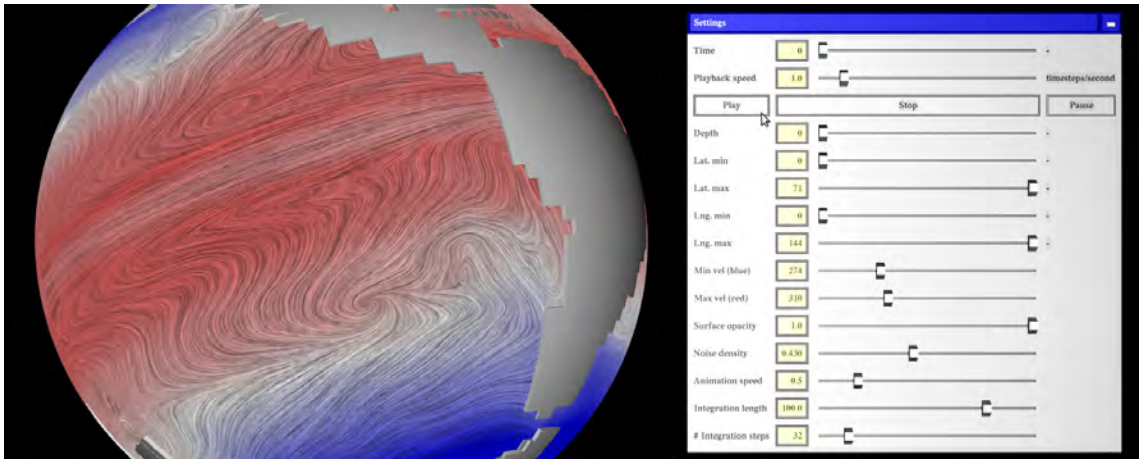


Figure 6.1: Visualization of surface currents using line integral convolution and color-coded temperature. The user interface supports VCR-style controls for time navigation as well as spatial cropping functionality, visualization of deposition records and interactive particle tracing.

depositions on the ocean floor. Scientists use coring techniques to obtain samples at select points and analyze their composition in order to improve our understanding of the involved transport processes. Recovering the history of flow patterns based on these sparse samplings of the deposition record is an underdetermined inverse problem, however. Consequently, there is no unique solution unless one selects values for additional constraints and boundary parameters which are uncertain. Due to this uncertainty, scientists require visualization tools to support the analysis of simulation results to determine whether they agree with the original observations.

My contribution is an interactive visualization system (Figure 6.1) which produces spherical renderings of time-varying simulated global flow data in a desktop or virtual reality (VR) environment. Line integral convolution (LIC) is used to visualize flow direction on surfaces of constant depth, latitude or longitude or any combination thereof, allowing users to obtain cross-sectional views of flow behavior.

Sediment or isotope concentration data from coring or drilling experiments can be overlaid using a billboard metaphor. Massless virtual particles can be interactively injected at any point in the flow field to, for example, verify the trajectories predicted by the flow field against observed sediment concentrations. Scalar quantities

such as temperature or salinity can be visualized simultaneously using color-coding (heatmaps).

The system was developed to support validation of future prehistoric flow models in a project supported by the U.S. National Science Foundation. For evaluation purposes, we used simulation results produced by [TT07] which represent the last 21,000 years at a resolution of 10 years per timestep. The flow field as well as a scalar temperature field at each timestep is given on a uniform latitude-longitude grid with a resolution of 2.5 degrees and 20 non-uniformly spaced depth layers at each grid point.

6.2 Related work

[NMG+96] presented a VR-integrated in-situ visualization system for ocean flow simulations. Specifically, their system generates a low-resolution 2D model of the Sea of Japan, allowing the user to change simulation parameters on a front-end VR system and obtain an updated solution on the fly from a cluster. In the application presented here, however, any change of parameters would require a re-computation of the entire time-dependent 3D flow field, making interactive parameter space exploration infeasible.

LIC, introduced by [CL93], is a well-established technique for visualizing flow on surfaces. By integrating a noise texture along the flow direction, a dense texture is computed which indicates flow direction at any point of the surface. I have implemented a GPU-based version of the algorithm, as described by [HWSE99], using 3D textures to represent the flow field. Since static LIC does not convey flow orientation, we apply an animated variant by convolving the grayscale values with a phase-shifted sinusoidal kernel as described in [WGP97].

[SBK07] demonstrated interactive particle tracing using GPGPU techniques. Even though this approach would be applicable here, I have chosen to implement a simpler CPU-based integration scheme as the performance proved to be sufficient.

The CAVE, introduced by [CNSD+92], is a highly immersive virtual reality environment consisting of a room whose walls serve as projection surfaces for interactive

stereoscopic imagery. My system is designed to be usable both in desktop and CAVE environments.

6.3 Virtual reality integration

Development of visualization applications for VR environments can be challenging due to the vast number of possible hardware configurations, from a simple desktop workstation up to a CAVE environment, a room whose three to six walls serve as projection surfaces for stereoscopic images. A similar range exists for input devices. In a desktop setup, a 3D space-mouse is often found, which is functionally similar to a joystick but is able to capture translation and rotation on three axes each, resulting in six degrees of freedom (a so-called 6-DOF device). Such table-top devices are not practical in larger environments where users are standing and walking around, however. Here, so-called flysticks are frequently employed. These handheld devices capture the positions and the orientation of the user's hand, again supporting six degrees of freedom.

Software development is complicated further by the fact that most large display systems use multiple front-end rendering nodes to produce output images. This implies that an efficient visualization system should be a distributed program which executes in parallel on these nodes. To help manage the complexity of interfacing with a wide range of input and output devices as well as handle state synchronisation between distributed application instances, so-called VR toolkits were developed. These hide the specific hardware configuration of the target environment from application developers by managing display transformations and providing abstraction layers for input devices. I have chosen the VRUI toolkit (Virtual Reality User Interface), developed by [Kre08], which also contains functionality for creating graphical user interfaces (GUIs) to control visualization parameters.

Figure 6.2: Interaction with the parameter dialog in a CAVE using a pick-ray.

6.4 User interface

The user interface of the application consists of a parameter dialog as well as intuitive metaphors for direct interaction with a virtual globe. In a desktop environment, the parameter dialog appears as a regular, 2D window using the mouse cursor as an interaction device. Within a CAVE, however, the dialog is projected into a plane which can be placed anywhere in the virtual space. A so-called pick-ray extending from a handheld flystick is used to point at and manipulate dialog elements (Figure 6.2).

At the top of the parameter dialog, a slider is provided to browse and select individual timesteps and VCR-style controls can be used to toggle automatic playback at a selectable speed. Using a set of five sliders, the flow visualization can be cropped to a specific box-shaped region of interest. Color-coding of surface textures is controlled by a drop-down menu which selects a data source as well as a pair of sliders to chose the scalar value interval which is mapped to blue and red, respectively.

If tabular data from drilling or coring is available, drop-down menus are provided to select a set of up to four columns which are visualized simultaneously.

Apart from manipulating controls in the parameter dialog, the flystick can also be used to directly interact with the virtual globe. When the user presses and holds a button while pointing at the globe, it becomes attached to the flystick, following any movement and rotation of the user's hand. This is intuitively described as a 'fork' metaphor, due to the manipulated object behaving as if the user had stuck a fork into it.

Particle seeding is another functionality which exploits the unique visualization and input capabilities of the CAVE environment. As the user shares the 3D virtual world with the virtual globe, it is possible to position the globe at the center of the room. The user can walk around the globe and reach into the water volume to inject particles at any point in the 3D flow field.

6.5 Data visualization

Our system assumes that all input data (vector and scalar field data) are given in NetCDF format and represented on a latitude-longitude-depth grid with uniform spacing for the geographical coordinates and arbitrary spacing for depth. All computations are performed in this spherical coordinate system and geometry is transformed to Euclidean coordinates just prior to rendering to obtain a spherical virtual globe representation. Depth coordinates are exaggerated by a factor of 300 to compensate for the large difference between the Earth's radius and the depth of the ocean and to obtain some visual separation between adjacent depth layers. Those simulation grid cells which do not contain any data are displayed as solid geometry to provide a coarse approximation of landmasses and bathymetry for orientation purposes.

Coring or drilling data is expected to be in tabular CSV format with columns specifying latitude and longitude coordinates of the drill site and any number of additional columns containing measured scalar concentration values.

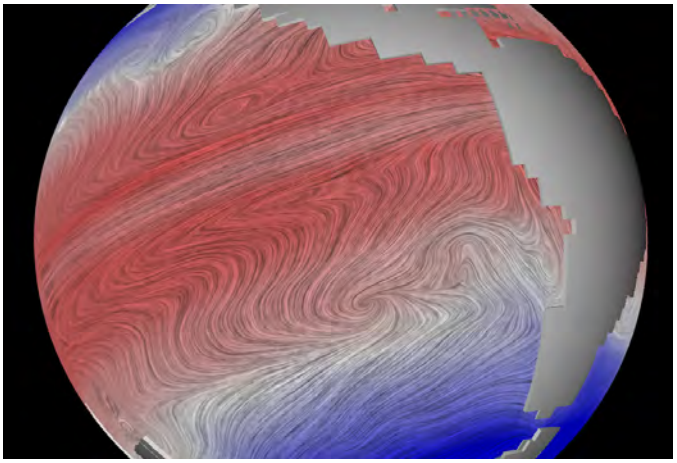


Figure 6.3: Visualization of flow at the ocean surface (depth zero) using LIC. Flow orientation is indicated by animation while color indicates velocity magnitude (blue < white < red).

6.5.1 Flow visualization

Two-dimensional LIC was implemented as the primary method to visualize the flow field. Given an arbitrary surface geometry, this technique can visualize the flow component tangential to the surface by convolving ('smearing') a grayscale noise texture along the flow. The resulting texture can be colored in order to convey additional scalar data, such as flow velocity or temperature.

LIC computation is performed in a GPU shader program, which allows all visualization parameters to be changed interactively. For this purpose, the flow and scalar fields for the current time step are maintained within GPU memory as a 3D texture. Field data at non-integer time steps is approximated by linear interpolation to provide smooth animation during playback at any speed.

Data at the surface are visualized by using a spherical shell corresponding to a depth of zero as input for the LIC algorithm (Figure 6.3). The radius of this sphere can be changed using a slider to explore the flow at different depths. When a region of interest is selected, the sphere surface is cropped to a sector corresponding to the selected longitude-latitude intervals. The boundary of this sector is closed by inserting four bounding rectangles. All five exposed surfaces are textured using the

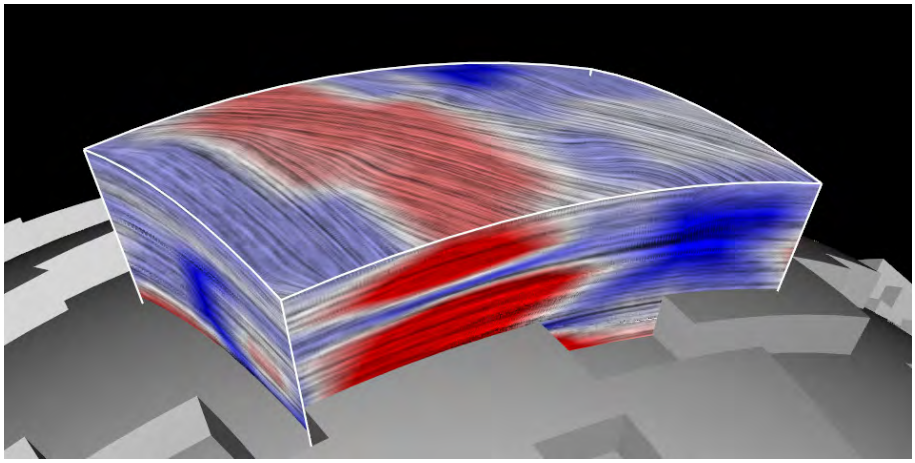


Figure 6.4: Constraining the visualization to a region of interest reveals flow behavior in depth. Note the coarse approximation of bathymetry, which is generated from the set of those simulation grid cells which are marked as having no data.

LIC algorithm which results in a cross-sectional view, revealing 3D flow behavior in depth (Figure 6.4).

6.5.2 Particle tracing

While the LIC method provides a good overview of the flow field topology at the current time step, it produces too much visual clutter to follow individual ocean circulation pathways. To enable users to isolate pathways of interest, we have implemented real-time particle tracing. Massless particles can be injected anywhere within the 3D volume and their trajectory can be observed as they are advected by the flow field (Figure 6.5). At any point in time, particles are advected by the flow field at the currently visible time step. This implies that the meaning of the resulting particle trajectories depends on whether time animation is enabled or not. When paused, the particles travel through a static flow field and form so-called streamlines. This is a valid approximation when assuming that particle travel time is small relative to the time interval between two time-steps (10 years in the reference dataset). Otherwise, when animation is enabled, the particle trails are called pathlines and record the trajectory of a particle being advected as the ocean flow patterns change.

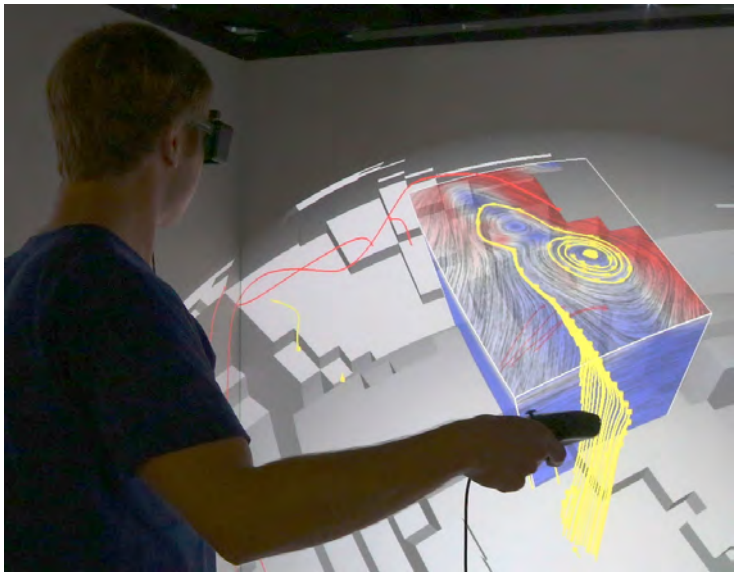


Figure 6.5: Interactive particle injection in the vicinity of an eddy (vortex). As the particles are advected by the flow, their trails (pathlines) are visualized in yellow.

6.5.3 Visualization of coring data

Particle tracing can be used to visualize sediment transport by seeding particles at a source site and observing their trajectory. To help verify the transport pathways against deposition records, sediment concentrations from core sample databases can be quantitatively visualized using a billboard technique. Bar charts representing up to four columns of the data are rendered at the location of each drill site. Each chart is projected onto a rectangle, a so-called billboard, which is dynamically oriented towards the user for maximum legibility (Figure 6.6). Billboards for sites which are beyond the horizon are culled (omitted from rendering), using smooth blending of transparency to avoid disturbing popping artifacts.

6.6 Results

Figure 6.7 illustrates how the different visualization components work together. The flow field at the ocean surface is visualized using LIC and color-coded according to velocity. The coring data shown represents isotope concentrations within core

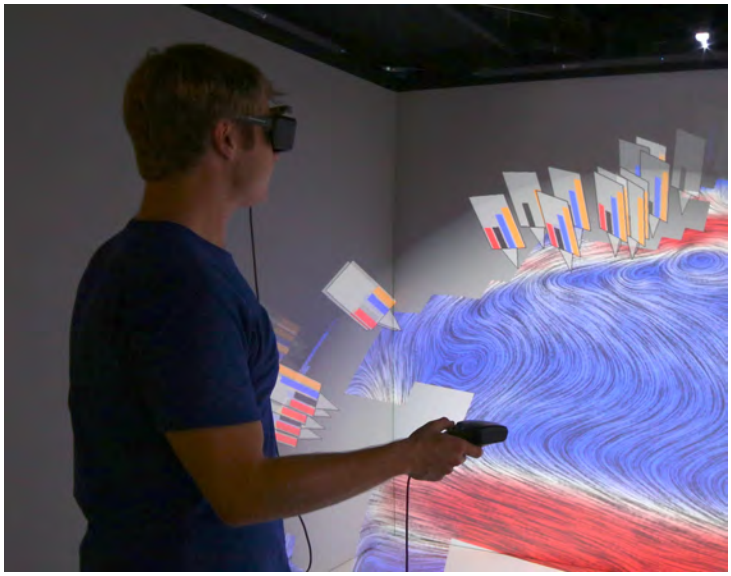


Figure 6.6: Billboard visualization of deposition records.

samples, which are dominated by shells of benthic foraminifera which live along the seafloor. These are buried by falling sediments, providing a timeline in form of a layered deposition. The given data captures concentrations of δ^{13}C and δ^{18}O in the Holocene and during the last glacial maximum (LGM) period.

Particles were injected in the vicinity of two coring sites which exhibit a similar isotope distribution. The particle trails (yellow) converge and become parallel, passing through two other sites which also show a similar distribution fingerprint. This observation suggests a potential correlation between the simulated flow field and measured isotope concentrations.

6.7 Future work

The system presented here is a work in progress which evolves with the needs of domain scientists working on numerical models of the prehistoric ocean. In the context of this project, work is being done to create a forward-simulation model for sediment transport which advects scalar quantities along a flow field, producing a

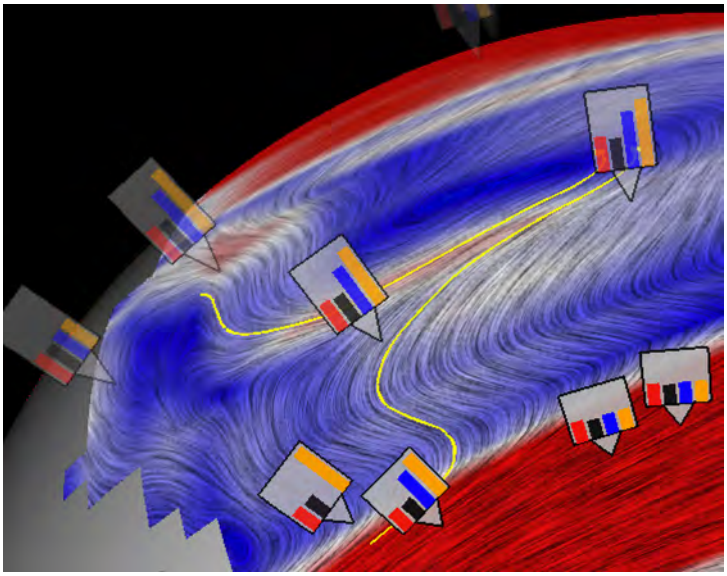


Figure 6.7: Analysis of conservation of seawater properties along the simulated flow using particle tracing. Billboards indicate isotope concentrations for Holocene δ^{13}C (red), LGM δ^{13}C (black), Holocene δ^{18}O (blue) and LGM δ^{18}O (yellow).

global scalar field of predicted sediment concentrations. I expect that the presented system will be useful to verify results generated by this model.

When visualizing core samples using the billboard technique, clusters of nearby drill sites currently result in visual clutter due to their billboards obscuring each other. This could be prevented by 'exploding' these clusters of billboards and connecting each to the location of the drill site by a line.

The current requirement to load the entire data set to be visualized into RAM when starting the system leads to a limitation in the size of data that can be handled. Future flow will certainly be too large to be handled in this fashion and a data streaming approach where only data within an interval around the currently selected time step is maintained in memory will be required.

Chapter 7

Conclusion

In the presented work I have documented my research into the application of virtual reality (VR) methods to support geoscientific research. This includes both the development and implementation of suitable visualization and interaction metaphors as well as the evaluation of their impact in the application domain.

I have demonstrated a novel terrain visualization algorithm using a quad-tree tiling of the HEALPix coordinate system, whose properties allow it to support not only efficient rendering but also measurements of spatial properties. A shader-based implementation of the projection between HEALPix and euclidean space guarantees a simple structure of the high-level rendering algorithm by restricting most geometric operations to the 2D plane. The system was applied to a 500GiB dataset of Mars and studies of both performance and image quality at varying quality levels were performed.

A distributed pre-processing system was then introduced which performs efficient on-the-fly data conversion between standard map formats and the hierarchical Level-of-Detail representations required for interactive terrain visualization. The conversion algorithm executes on a backend cluster of machines equipped with GPUs and produces tile data required by a rendering frontend on demand. Potential applications of the presented system include future disaster management scenarios in which sensor platforms provide near-live three-dimensional coverage of the situation.

To provide geologists with virtual tools for quantification of topographic features discovered within a VR session, a pick-ray based interaction scheme was implemented. Scientists can either measure height profiles by specifying a line on the ground or determine volumes of features (e.g. mountains or craters) by specifying a polygonal footprint indicating the integration domain. The equal-area property of the HEALPix hierarchical grid and the locality-preserving nature of the quadtree scheme are exploited to obtain simple and efficient query algorithms.

Furthermore, I have presented a human-in-the-loop system for reconstructing geological fault offsets which uses a deformation model to visualize the reversal of fault processes within an existing terrain rendering system. The approach is interactive with respect to all parameters, which allows users to explore the continuum of translations leading up to a reconstruction. The neighborhood of a fault trace can be inspected at any scale to search for surface features providing evidence for or against a successful reconstruction. It was shown that by performing these reconstructions in a 3D terrain visualization as opposed to the traditional top-down perspective, ambiguous interpretations can be avoided as the deformed topography can be examined from any perspective.

The last application domain addressed within this work is palaeoceanography. I have developed a VR-integrated visualization system for time-dependent, global 3D flow fields as produced by models of the prehistoric ocean. The system's purpose is to help oceanographers validate those numerical models by providing simultaneous visualization of the primary data from coring experiments as well as the flow fields predicted from that data through simulation. Using a globe representation within a CAVE-environment, sediment concentrations obtain through coring are visualized using a billboard technique while the flow vector field is illustrated using line-integral-convolution (LIC). To examine transportation pathways and their correlation with measured sediment concentrations, particles can be interactively injected into the 3D flow field at any point using a flystick input device.

In summary, my contributions presented in this work include immersive, interactive VR solutions for visualization, exploration and analysis of geoscientific datasets. For this purpose, I have introduced data structures suitable both for interactive visualization and analysis and have presented methods for accelerated construction of

these structures using GPU technology and distributed processing. I have demonstrated that, in contrast to desktop interfaces, VR techniques can lead to more accurate interpretation of features due to stereoscopic 3D visualization combined with free navigation.

When considering how VR techniques could be established as part of everyday geoscientific workflows, it quickly becomes clear that not every problem can be efficiently solved in VR. A number of common tasks such as creation of vector maps based on arial photography would certainly not benefit from 3D visualization and are better executed in a traditional 2D desktop environment. Key to the acceptance of VR methods, then, is the seamless interoperability between (existing) desktop systems and VR environments. To give an example scenario, assume that a user identifies an ambiguous terrain feature within a desktop GIS system. Ideally, he or she could then, with the press of a button, load the data into a CAVE environment and meet there with other colleagues to analyse and discuss the finding in a collaborative setting. Enabling this kind of impromptu access to VR technology is an open technical challenge in most application fields.

Bibliography

[AHZI08] Jeffrey C. Andrews-Hanna, Maria T. Zuber, and Steven A. Hauck II. Strike-slip faults on mars: Observations and implications for global tectonics and geodynamics. *J. Geophys. Res.*, 113:E08002, 2008.

[BCK+11] Tony Bernardin, Eric Cowgill, Oliver Kreylos, Christopher Bowles, Peter Gold, Bernd Hamann, and Louise Kellogg. Crusta: A new virtual globe for real-time visualization of sub-meter digital topography at planetary scales. *Computers & Geosciences*, 37(1):75–85, 2011. Virtual Globes in Science.

[Bir78] Peter Bird. Finite element modeling of lithosphere deformation: The zagros collision orogeny. *Tectonophysics*, 50(2-3):307–336, 1978.

[BK05] Mario Botsch and Leif Kobbelt. Real-time shape editing using radial basis functions. In *Computer Graphics Forum*, pages 611–621, 2005.

[BSM+02] Cynthia D. Bruyns, Steven Senger, Anil Menon, Kevin Montgomery, Simon Wildermuth, and Richard Boyle. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The Journal of Visualization and Computer Animation*, 13(1):21–42, 2002.

[CGG+03] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Planet–sized batched dynamic adaptive meshes (p-bdam). In *Proceedings IEEE Visualization*, pages 147–155, Conference held in Seattle, WA, USA, October 2003. IEEE Computer Society Press.

[CH06] Malte Clasen and Hans-Christian Hege. Terrain rendering using spherical clipmaps. In *EuroVis*, pages 91–98, 2006.

[CL93] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on*

Computer graphics and interactive techniques, SIGGRAPH '93, pages 263–270, New York, NY, USA, 1993. ACM.

[CNSD+92] Carolina Cruz-Neira, Daniel J Sandin, Thomas A DeFanti, Robert V Kenyon, and John C Hart. The cave: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.

[Coq90] Sabine Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. *SIGGRAPH Comput. Graph.*, 24:187–196, September 1990.

[CS07] Carlos D. Correa and Deborah Silver. Programmable shaders for deformation rendering. In *GH '07: Proceedings of the 2007 ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 89–96, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[DWS+97] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Mille, Charles Aldrich, and Mark B. Mineev-weinstein. Roaming terrain: Real-time optimally adapting meshes. In *IEEE Visualization*, pages 81–88, 1997.

[EKB+99] S. S. Egan, S. Kane, T. S. Buddin, G. D. Williams, and D. Hodgetts. Computer modelling and visualisation of the structural deformation caused by movement along geological faults. *Computers & Geosciences*, 25(3):283–297, 1999.

[FBD+07] Kurt L. Frankel, Katherine S. Brantley, James F. Dolan, Robert C. Finkel, Ralph E. Klinger, Jeffrey R. Knott, Michael N. Machette, Lewis A. Owen, Fred M. Phillips, Janet L. Slate, and Brian P. Wernicke. Cosmogenic 10be and 36cl geochronology of offset alluvial fans along the northern death valley fault zone: Implications for transient strain in the eastern california shear zone. *J. Geophys. Res.*, 112(B6):B06407, 2007.

[GHB+05] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759, 2005.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA, 1996. ACM.

[GP10] M. P. Golombek and R. J. Phillips. Mars tectonics. In *Planetary Tectonics*, pages 183–232. Cambridge University Press, 2010.

[GSP+10] K. Gwinner, F. Scholten, F. Preusker, S. Elgner, T. Roatsch, M. Spiegel, R. Schmidt, J. Oberst, R. Jaumann, and C. Heipke. Topography of mars from global mapping by hrsc high-resolution digital terrain models and orthoimages: Characteristics and performance. *Earth and Planetary Science Letters*, 294(3-4):506 – 519, 2010.

[HWSE99] Wolfgang Heidrich, Rüdiger Westermann, Hans-Peter Seidel, and Thomas Ertl. Applications of pixel textures in visualization and realistic image synthesis. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 127–134. ACM, 1999.

[KBB+06] Oliver Kreylos, Gerald Bawden, Tony Bernardin, Magali I. Billen, Eric S. Cowgill, Ryan D. Gold, Bernd Hamann, Margarete Jadamec, Louise H. Kellogg, Oliver G. Staadt, and Dawn Y. Sumner. Enabling scientific workflows in virtual reality. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, VRCIA '06, pages 155–162, New York, NY, USA, 2006. ACM.

[KLJ+09] R. Kooima, J. Leigh, A. Johnson, D. Roberts, M. SubbaRao, and T.A. DeFanti. Planetary-scale terrain composition. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):719–733, sept.-oct. 2009.

[Kre08] Oliver Kreylos. Environment-independent VR development. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, ISVC '08, pages 901–912, Berlin, Heidelberg, 2008. Springer.

[Lev02] Joshua Levenberg. Fast view-dependent level-of-detail rendering using cached geometry. In *Proceedings of the conference on Visualization '02*, VIS '02, pages 259–266, Washington, DC, USA, 2002. IEEE Computer Society.

[LH04] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 769–776, New York, NY, USA, 2004. ACM.

[LK10] Martin Lambers and Andreas Kolb. Dynamic terrain rendering. *3D Research*, 1:1–8, 2010. 10.1007/3DRes.04(2010)01.

[LL06] Q. Li and M. Liu. Geometrical impact of the san andreas fault on stress and seismicity in california. *Geophysical Research Letters*, 33, 2006.

[NMG+96] S. Nations, R. Moorhead, K. Gaither, S. Aukstakalnis, R. Vickery, Jr. Couvillion, W.C., D.N. Fox, P. Flynn, A. Wallcraft, P. Hogan, and O.M. Smedstad. Interactive visualization of ocean circulation models. In *Visualization '96. Proceedings.*, pages 429–432, Nov 1996.

[SBK07] M. Schirski, C. Bischof, and T. Kuhlen. Interactive exploration of large data in hybrid visualization environments. In *Eurographics Symposium on Virtual Environments*, pages 69–76. The Eurographics Association, 2007.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20:151–160, August 1986.

[SSE04] Guy Sela, Sagi Schein, and Gershon Elber. Real-time incision simulation using discontinuous free form deformation. In Stéphane Cotin and Dimitris N. Metaxas, editors, *Medical Simulation*, volume 3078 of *Lecture Notes in Computer Science*, pages 114–123. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-25968-8_13.

[SZF+01] D. E. Smith, M. T. Zuber, H. V. Frey, J. B. Garvin, J. W. Head, D. O. Muhleman, G. H. Pettengill, R. J. Phillips, S. C. Solomon, H. J. Zwally, W. B. Banerdt, T. C. Duxbury, M. P. Golombek, F. G. Lemoine, G. A. Neumann, D. D. Rowlands, O. Aharonson, P. G. Ford, A. B. Ivanov, C. L. Johnson, P. J. McGovern, J. B. Abshire, R. S. Afzal, and X. Sun. Mars Orbiter Laser Altimeter: Experiment summary after the first year of global mapping of Mars. *Journal of Geophysical Research*, 106:23689–23722, October 2001.

[TKR+08] U. Thomas, F. Kurz, D. Rosenbaum, R. Mueller, and P. Reinartz. GPU-based orthorectification of digital airborne camera images in real time. In *Proceedings of the XXI ISPRS Congress*, 2008.

[TM92] R.J. Twiss and E. M. Moores. *Structural geology.* W. H. Freeman, 1992.

[TRAW12] Marc Treib, Florian Reichl, Stefan Auer, and Rüdiger Westermann. Interactive editing of gigasample terrain fields. *Computer Graphics Forum*, 31(2.2):383–392, 2012.

[TT07] O. Timm and A. Timmermann. Simulation of the last 21000 years using accelerated transient boundary conditions. *Journal of Climate*, 20(17):4377–4401, 2007.

[VAYK+10] Jordan Van Aalsburg, M. Burak Yikilmaz, Oliver Kreylos, Louise H. Kellogg, and John B. Rundle. Interactive editing of digital fault models. *Concurrency and Computation: Practice and Experience*, 22(12):1720–1731, 2010.

[vFTS06] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1118–1125, New York, NY, USA, 2006. ACM.

[WGH11] Rolf Westerteiger, Andreas Gerndt, and Bernd Hamann. Spherical Terrain Rendering using the hierarchical HEALPix grid. In *VLUDS'11*, pages 13–23, 2011.

[WGJS81] S. S. C. Wu, P. A. Garcia, R. Jordan, and F. J. Schafer. Topographic map of Olympus Mons. *LPI Contributions*, 441:287, 1981.

[WGP97] R. Wegenkittl, E. Groller, and W. Purgathofer. Animating flow fields: rendering of oriented line integral convolution. In *Computer Animation '97*, pages 15–21, Jun 1997.

[XMM07] H. L. Xing, A. Makinouchi, and P. Mora. Finite element modeling of interacting fault systems. *Physics of The Earth and Planetary Interiors*, 163(1-4):106–121, 2007. Computational Challenges in the Earth Sciences.

[ZA12] Olaf Zielke and J Ramon Arrowsmith. LaDiCaoz and LiDARimager–MATLAB GUIs for LiDAR data handling and lateral displacement measurement. *Geosphere*, 8(1):206–221, 2012.

BIOGRAPHICAL INFORMATION

PERSONAL DETAILS

Name	Rolf Westerteiger
Date of birth	August 23, 1981
Place of birth	Koblenz, Germany
Nationality	German

EDUCATION

04/2008 – present	PhD student at the Department of Computer Science, Computer Graphics and HCI Group, University of Kaiserslautern, Germany
04/2008 – present	Member of the International Research Training Group (IRTG) "Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling, and Engineering"
08/2009 – 03/2013	Dissertation project at DLR (German Aerospace Center), Braunschweig, Germany
10/2012 – 12/2012	Research visit at the University of California, Davis, U.S.A.
02/2011 – 04/2011	Research visit at the University of California, Davis, U.S.A.
09/2010 – 11/2010	Research visit at the University of California, Davis, U.S.A.
10/2001 – 04/2008	Study of Computer Science at the University of Kaiserslautern, Germany Degree: Diplom-Informatiker (Dipl.-Inf.) (Grade point average 1.2) Master Thesis: "“Extraction and Tracking of PAR3-EGFP marked Cell Nuclei of the Eye of Danio Rerio”"
1992 – 2001	Max-von-Laue Gymnasium, Koblenz, Germany

WORK EXPERIENCE

04/2008 – 03/2009	Internship at "Procaess GmbH", Landau, Germany
07/2007 – 09/2007	Bachelor thesis project: "Distributed Implementation of an Explicit-Jump Solver for Effective Heat Conductivity" at Fraunhofer ITWM, Kaiserslautern, Germany
12/2003 – 12/2007	Research assistant at Fraunhofer ITWM, Kaiserslautern

List of publications

[**WCB+12**] Rolf Westerteiger, Tracy Compton, Tony Bernadin, Eric Cowgill, Klaus Gwinner, Bernd Hamann, Andreas Gerndt, and Hans Hagen. **Interactive Retro-Deformation of Terrain for Reconstructing 3D Fault Displacements**. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2208–2215, 2012.

[**WCG+12**] Rolf Westerteiger, Fang Chen, Andreas Gerndt, Bernd Hamann, and Hans Hagen. **Remote GPU-Accelerated Online Pre-processing of Raster Maps for Terrain Rendering.** In *VR/AR*, pages 143–153, 2012.

[**Wes10**] Rolf Westerteiger. **Cartography of mars in a virtual reality environment.** In *VLUDS'10*, pages 100–110, 2010.

[**WGH11**] Rolf Westerteiger, Andreas Gerndt, and Bernd Hamann. **Spherical Terrain Rendering using the hierarchical HEALPix grid**. In *VLUDS'11*, pages 13–23, 2011.

[**CCWB11**] Tracy Compton, Eric Cowgill, Rolf Westerteiger, and Tony Bernardin. **Offset Landforms Record the Holocene History of Fault Slip Along the Mojave Section of the San Andreas Fault**. In *AGU Fall Meeting Abstracts*, volume 1, page 2325, 2011.

[**KHW+09**] Aaron Knoll, Younis Hijazi, Rolf Westerteiger, Mathias Schott, Charles Hansen, and Hans Hagen. **Volume ray casting with peak finding and differential sampling**. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1571–1578, 2009.

[**KWH12**] Aaron Knoll, Rolf Westerteiger, and Hans Hagen. **An Evaluation of Peak Finding for DVR Classification of Biological Data**. In *Visualization in Medicine and Life Sciences II*, pages 91–106. Springer, 2012.