3D Geomodeling for Europe

Project number: GeoE.171.005

Deliverable 4.3

Authors and affiliation:
Björn Zehner
[BGR]

E-mail of lead author:
Bjoern.Zehner@bgr.de

Requirements of the European Geoscience Data Infrastructure (EGDI) for visualizing uncertainties in geomodels

Version: 19-10-2021

This report is part of a project that has received funding by the European Union's Horizon 2020 research and innovation programme under grant agreement number 731166.

Deliverable Data		
Deliverable number	D4.3	
Dissemination level	Public	
Deliverable name	Uncertainty visualization requirements of EGDI	
Work package	WP4, Uncertainty in Geomodels	
Lead WP/Deliverable beneficiary	BGR	
Deliverable status		
Submitted (Author(s))	19.10.2021	Björn Zehner
Verified (WP leader)	19.10.2021	Björn Zehner
Approved (Project Leader)	20.10.2021	Stefan Knopf

GENERAL INTRODUCTION

When 3D models of the geological subsurface are built with large extents, for example on basin scale, the underlying data are usually distributed unevenly, e.g. clustering in regions with economically interesting reserves, while being very sparse elsewhere. Further the data itself can only be interpreted with a certain degree of uncertainty, and finally the whole process of generating a 3D model from the different data is subject to interpretational issues, thus generating additional uncertainty.

This uncertainty characterizing the 3D models stands in stark contrast to the way in which the 3D modelling results are usually visualized these days. The software packages that are used for 3D geological modelling, such as Skua-Gocad or Petrel, already provide visualization methods that are currently used to communicate the 3D models to the stake holders or the public. Further 3D models are published on the world wide web, using the necessary web-technology to present these models in a browser. The visualization is usually done by rendering stratigraphic interfaces and faults as triangle- or quadrangle-meshes in 3D space and it pretends that the 3D subsurface is known exactly, sometimes giving the position of a mesh's vertices with a precision of up to a millimeter. In reality, however, we often do not know if a certain fault should be moved up or down a hundred meters, if it extends hundred meters more or less, or even if it actually exists at all or has a complete different shape. How do we express the magnitude and different types of uncertainty in our 3D models and how can we estimate and handle the uncertainty? Work package 4, "Uncertainty in Geomodels" which is part of the GeoERA project 3DGEO-EU, will work towards establishing the necessary workflows to provide a visualization of the 3D models, including their uncertainty.

One important question concerns the requirements which the handling and presentation of the uncertainty add to the overall requirements for the EGDI infrastructure. These might involve some requirements regarding the data management but mainly will affect the visualization backend, in order to be able to make the users aware of the uncertainty. The purpose of this report is to sum up and motivate these additional requirements.

TABLE OF CONTENTS

1 INTRODUCTION

When constructing 3D regional models of the subsurface, the geoscientist has to deal with a wide range of different types of uncertainty. As shown in Figure 1, the uncertainty should already be estimated and assessed during the acquisition and interpretation of the data which later form the basis of the 3D model. The location of markers for faults and horizons that are interpreted from borehole data is uncertain, especially when old logs from the archive have to be used, as the tools to determine the borehole path had, and still have, only a limited precision (see e.g. Wolf & Wardt, 1981). When seismic imaging is used, different sources of uncertainty are introduced in the different steps of the seismic processing sequence, especially during the time to depth conversion as often the velocity model can only be estimated with a limited precision (for an overview, see e.g. Thore et al., 2002).

During the next step, namely the geometrical modelling phase during which the 3D geological model is built, the propagation of the uncertainty that comes from the input data must be assessed and its influence on the final model estimated. Sometimes there are insufficient data available for a large area and the modeller has to provide some kind of model-based interpretation in order to fill the void space in the 3D model. So the modellers have to make a decision on which conceptual models they should apply (e.g. the deformation style? flexure or fracture?) which introduces additional uncertainty, often called conceptual uncertainty. The approach commonly used to assess all these uncertainties in the resulting 3D model is the use of Monte-Carlo Simulation (see, e.g., Wellmann & Regenauer-Lieb, 2012 or Schweizer et al., 2017). Different realizations of the 3D model are generated by first sampling into the input data. The depth of a borehole marker might, for example, be given as a Gaussian distribution function and for each realization the depth is randomly drawn from this function (see, e.g., Pakyuz-Charrier et al., 2018). Subsequently, a 3D model is generated for each set of randomly drawn data. These different models are then visualized or ideally could be summarized to be represented as one model which expresses the geology and its uncertainty (see, e.g. Wellmann & Regenauer-Lieb, 2012). When the resulting uncertain structural geological model is subsequently used for process simulation, it has to be propagated with attributes, such as permeability, which also involves uncertainty. Many methods have been developed to treat this uncertainty, especially in the oil & gas and the mining industry to optimize exploitation and minimize risk (see e.g. Pyrcz & Deutsch, 2014). A general overview of these first two steps with respect to the uncertainty of structural models has been given in Deliverable 4.2 of the GeoERA project 3DGEO-EU (Zehner et al., 2021)

The last, but nevertheless important, step in Figure 1 is the visualization. When the 3D models generated are presented to the public and the stakeholders, they should be made aware of these uncertainties in those models. Currently the representation of the geological models as triangle- or quadrangle-meshes often pretends that the position of geological structures is known with a precision of a centimetre. It is one of the primary targets of this work package to find a good visualization which shows the uncertainty in 3D geological subsurface models and where this uncertainty is coming from. The visualization should be easy to understand and intuitive and might vary for different types of viewers, e.g. for experts and novices. A discussion of these different visualization options can be found in Zehner (2021) and Deliverable 4.1 (Zehner, 2019).

The aim of the work package “Uncertainty in Geomodels” is to structure the whole discussion on uncertainty in our 3D geological models and its quantification and visualization from the viewpoint of geological surveys. What is already there and what are the gaps? The work package provides a knowledge base to assist in the future use of the visualization methods already established in geosciences and also establishes the basis for future cooperation with other research disciplines, such as computer graphics, to fill the gaps identified.

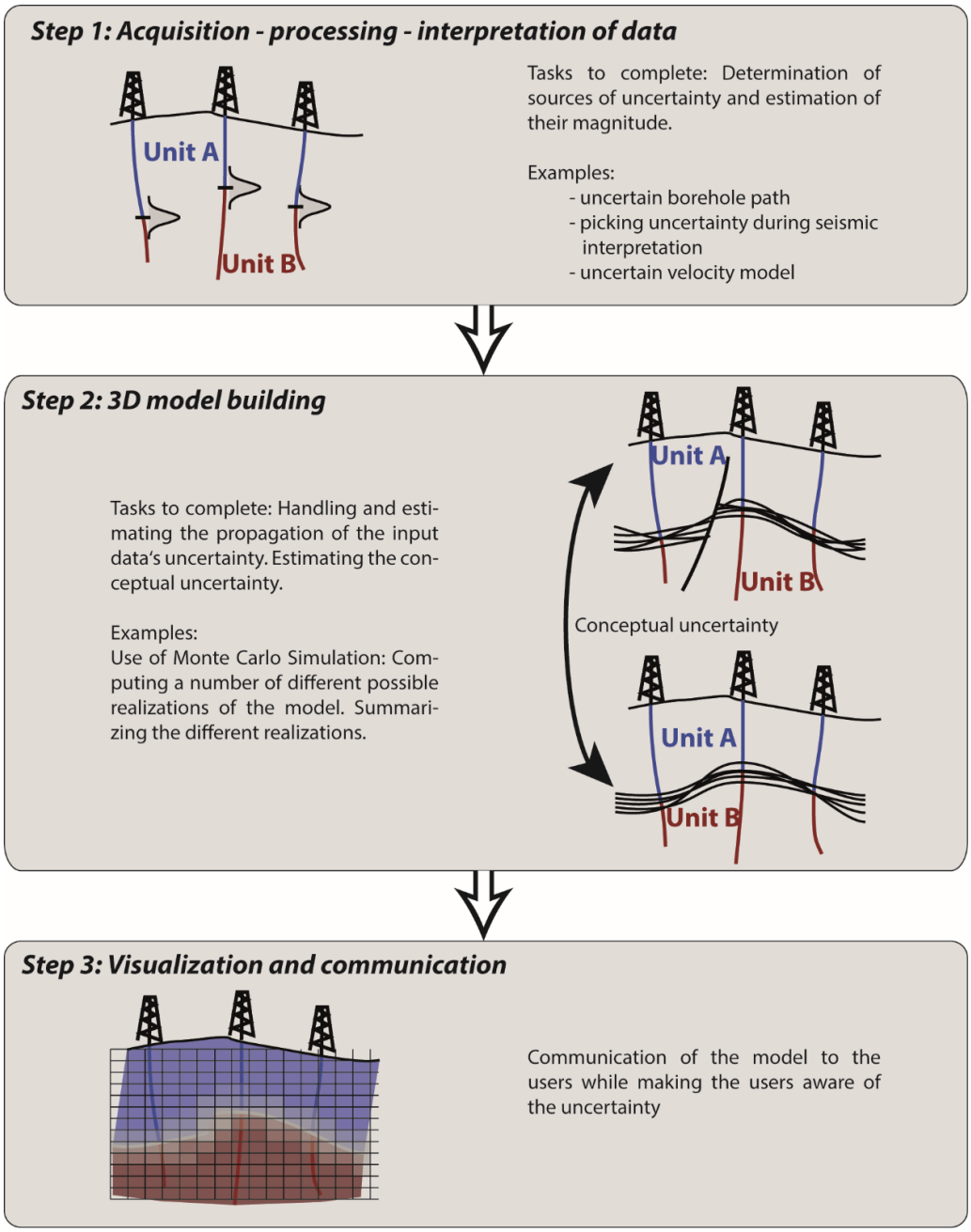

Figure 1: The different general steps to build and display a 3D geological model where the uncertainty has to be assessed.

In order to achieve this, the whole work package is structured in terms of four different tasks (see Figure 2). During the first half of the project, the aim of the first two tasks has been to establish and document the methods and concepts required. Task 1 captured the state of the art in uncertainty visualization (options for step 3 in Figure 1) and in this manner also provides information about which type of data we need to compute in order to be able to display the uncertainty in our models. It thus sheds light on where we might go and what we will need for it. Task 2 discussed the different sources of uncertainty and the methods to propagate this uncertainty through the 3D modelling process (steps one and two in Figure 1). Task 3 and 4 in the second half of the project applied the methods described, in order to test different visualization options, generated some example data sets and discuss the requirements that the assessment and visualization of uncertainty will have for the European Geoscience Data Infrastructure.

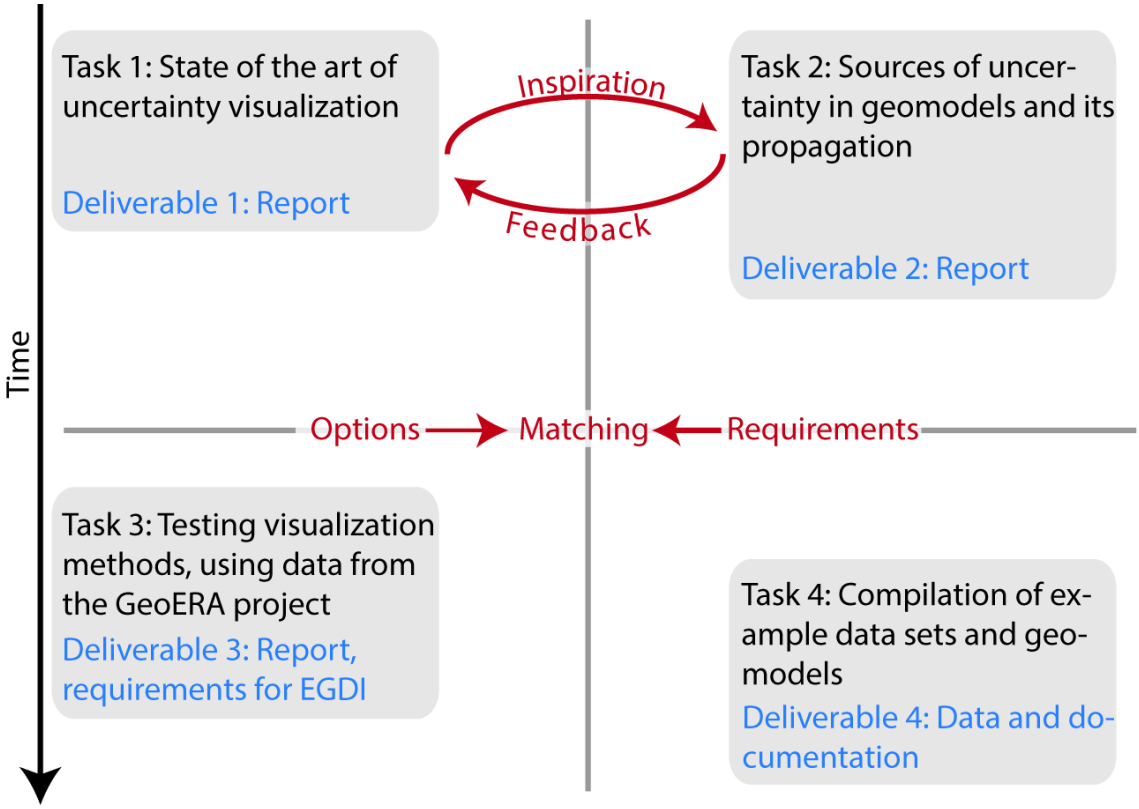

Figure 2: General structure of the 3DGEO-EU work package "Uncertainty in Geomodels".

The overall outcome of the project is a structured and documented overview of what is already available for the treatment and visualization of uncertainty and thus acts as a point of transfer for the necessary knowledge and skills from computer sciences to geosciences. Further it tries to suggest some best practices and workflows for how the visualization of uncertainty could be incorporated into the current standard workflows for 3D geological modelling. Finally the work

package identifies what still needs to be developed and provide the necessary means, gap identification and corresponding example data sets, to give potential outside partners, such as computer graphics groups at universities, the motivation to do research towards developing the methods lacking.

If the European Geoscience Data Infrastructure (EGDI) is to be used in the future for storing and visualizing 3D geomodels that have been analysed for and augmented with uncertainty information, some additional requirements must be met that are beyond the usual requirements for storing and visualizing 3D geological models of the subsurface. This report is the deliverable of Task 3 and captures and motivates these special requirements for the EGDI infrastructure and the information platform. The general processing is usually done by scientists with specialized software. The EGDI infrastructure is mainly used for the dissemination and visualization of the results. For this reason, there are some requirements towards the data management of the 3D models and some requirements for the visualization. Deliverable 4.1, the report on the state of the art in uncertainty visualization (Zehner, 2019) and Zehner (2021), showed that a wide range of visualization methods are available that show the uncertainty within 3D models. However, most of them are quite specialized to show very specific types of uncertainty, for example in flow fields. For the EGDI we should focus on the more general methods that are needed to show the structural uncertainty. The methods available that can be easily implemented are explained in this document.

2 REQUIREMENTS FOR EGDI

Deliverable 4.1 and Zehner (2021) gave an overview of the many methods for visualization of uncertainty that have been published in the past in computer science literature. They have often been developed with data from other scientific disciplines in mind, such as medical sciences or engineering, but could be used for uncertain geoscience data with similar data types and representations. In order to get an overview and to decide which methods are suitable for a certain geoscience problem, they have been classified according to two different criteria which allows us to sort the methods into a 2D matrix:

- The first criterion is in which form or where the data should be presented overall. Should they be presented as a map, or at point locations on a map or in 3D space, or along lines, on polygons or on volumetric 3D cells? This makes an important difference regarding the usability of the methods. To give an example: image-based techniques, such as Line Integral Convolution, will not be suitable to present data or uncertainty at point or line locations. They would require a map or a surface in 3D space onto which they can be projected.
- The second criterion is which type of data we want to show together with its uncertainty. This could be spatially distributed scalar data for which the standard deviation is given, vectors for which the direction and magnitude is uncertain or tensors with an uncertain orientation. Further the actual position and orientation of the object we show might be uncertain or it might be uncertain if a shown object, such as a stratigraphic layer, exists at all at the given location (uncertain presence).

Zehner (2021) sorts the different methods into this 2D Matrix and gives the corresponding references where they are described in more detail. However, usually only the methods have been described algorithmically but no sample implementation is provided. This leads to an additional work load for each visualization method that should be provided and this work load can be considerable when the method is complex or requires further processing of the data to extract the visualized features. So, while in the optimal case, all the different visualization methods above would be implemented, and the corresponding data models would be included in a 3D database, this could not be done in the current time-line. It makes sense to do some kind of cost-benefit consideration as some of the methods outlined in Deliverable 4.1 would possibly be needed very rarely.

2.1 Which methods are needed most frequently and most urgent?

In the first instance it makes sense to look at what the most commonly used data representations and types will be. Partners in the GeoERA project 3DGEO-EU are mostly geological surveys and the European Geoscience Data Infrastructure (EGDI) will most likely serve to present regional geological data and structural models with or without uncertainty and some additional data, such as facies. In Deliverable 4.2 (Zehner et al., 2021) we provide an overview of available workflows and application cases for assessing uncertainty in structural models. Nearly all of them lead to one of the two following representations:

1. Voxel models with uncertain voxel classification: Several workflows that are published lead to voxel models. The TNO-Geological survey of the Netherlands takes a surface-based structural model as input and uses Geostatistics to estimate for each voxel to which

lithological class it belongs with which probability (see Stafleu et al., 2011, and Stafleu & Dubelaar, 2016). De la Varge et al (2019) describes a workflow and corresponding open source implementation that takes uncertain data, such as uncertain locations for the borehole markers, and generates multiple realizations of the geological model. These multiple realizations are then summarized as a voxel model, where each voxel holds, for example, the most likely stratigraphic unit and the information entropy. Pakyuz-Charrier et al (2018) use the software GeoModeller to generate a number of realizations for their example models and display information entropy as a measure of uncertainty. So, for voxel models we want to show two attributes simultaneously – the information about the stratigraphic unit or lithological class to which this voxel belongs, usually as some integer id, and the additional information about the extent to which this information is uncertain.

2. Surfaces with attached geometrical uncertainty: The Structural Uncertainty workflow that is part of the Skua-Gocad software can be used to calculate multiple realizations of a horizon or a fault from given uncertain input data. This multiple horizons can directly be visualized (see Schweizer et al. 2017 for an example) or be used to calculate the standard deviation for each vertex along the normal of the surface. Another example is the uncertainty assessment for the deep geological model of the Netherlands, DGM-Deep, Kombrink et al. (2012). Here the Monte Carlo approach is used in order to assess the uncertainty that is due to the picking of a horizon and further processing, due to structural complexity and due to the error in the velocity model. The result is a 2.5D horizon model, where for each vertex the depth and the standard deviation for the depth are given.

Further, in order to explain to the potential viewers where the uncertainty is coming from, it might be helpful to have the option to show these data and their uncertainty. Examples would be the uncertainty for the borehole markers that could be shown as ellipses that represent the 95% confidence envelope around a point or the uncertainty of the borehole path, rendered with an increasing diameter when it is less well known in deeper sections of the borehole. So, to summarize, the most important methods that should be made available for geoscience visualizations are the visualization of uncertain categorical scalar data on maps, on sections and as volumetric data, and the visualization of geometrical uncertainty of points, lines and surfaces in 3D space.

2.2 Requirements for the data management

EGDI will not be used to support the calculation and processing of the uncertainties. However, the data infrastructure should be able to store and distribute the resulting models and the information on their uncertainty. In general the representation of the models will be the same as for the certain representation. The models will consist of points, lines, polygons or cells, which each carry additional information as attributes. This could be, for example, to which geological unit they belong or what type of geoscience entity they represent (e.g. fault or horizon). Assuming that the EGDI platform will not implement specialized algorithms to further process the uncertainty, there are two ways to present the uncertainty in the database, both of which should be available. The uncertainty could be represented by additional geometry that is stored in the database, or the uncertainty is stored as an additional attribute for the vertices, pixels or primitives of the geometry, and the visualization that is needed to show this uncertainty is created on the fly. It should be mentioned that these requirements already need to be fulfilled

for the data management, independently of the additional need for the handling of uncertainty. The different types of attributes need to be handled anyway, for example to store an Id to represent geological units (integer numbers representing classes), local permeability or porosity (scalar numbers), the flow field in hydrogeology (vector data) or the local stress and strain (tensor data). So it makes sense to support the full set of needed requirements in terms of data management while the implementation of specialized visualization methods could be shifted into the future.

2.2.1 Storage as additional pre-processed geometric representation

For some of the uncertain 3D models, additional geometries could be used to show the uncertainty in the model. An example would be to compute additional surfaces that represent the confidence envelope around a horizon or a fault. This confidence envelope has the same type of representation as the original surface itself and so could be stored in the same way in the database. However, the original representation and the added geometries should be linked, so that it is obvious that the latter augment the former with additional information.

2.2.2 Storage as attributes

The parameters that describe the uncertainty, such as the standard deviation, could be saved for each vertex, pixel, voxel or geometric primitive (that means e.g. triangle, quadrangle in 3D space or a tetra- or hexahedron). In general 4 different types should be distinguished:

1. Uncertain scalar data: This could be an uncertain attribute, such as porosity or thickness of a layer. The data for each vertex or cell would then be described by a pair of scalar numbers – the mean of the attribute and the standard deviation or variance.
2. Geometric uncertainty on a map: some point that is pictured on a map might actually be displaced, as its position is uncertain. This possible displacement can be imagined as an ellipse, described as a 2x2 matrix that can be stored as a 4-component vector. Further texture based methods could be used that turn the point into a fuzzy object to indicate the uncertainty.
3. Geometric uncertainty in 3D space that is measured along a vector. This is usually either the vertical (elevation with standard deviation) or the vector normal to the surface at the vertex position (vertex-normal).
4. Geometric uncertainty in 3D space: A point is possibly displaced in 3D space. The standard deviation can be imagined as a 3D ellipsoid, described as a 3x3 matrix that can be stored as a 9-component vector. This Matrix can be calculated as follows where r_1 to r_3 are the unit vectors describing the orientation of the ellipsoid in 3D space:

$$T = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} * \begin{pmatrix} stddev\_max & 0 & 0 \\ 0 & stddev\_med & 0 \\ 0 & 0 & stddev\_min \end{pmatrix} * \begin{pmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{pmatrix}$$

While the uncertain scalar data could be visualized using colour mapping, the geometric uncertainty needs another representation, e.g. by computing additional geometries as described above or by using glyphs. In both cases the visualization could be pre-computed which would make it easier to fulfil the requirements but comes with drawbacks, as will be indicated below in the corresponding sections.

2.3 Requirements for the 3D visualization

2.3.1 Use of 2D colour maps and corresponding user interface

The use of colour mapping for the representation of scalar data is very common, for example on maps, when representing the geological units at the surface on geological maps (categorical scalar data) and the heights of the terrain using colour codes (continuous scalar data). Usually either a number of bins for the data values and corresponding colour codes are defined (categorical data) or a colour map is defined as a continuously changing set of, for example, 256 or 1024 different colours and the data are mapped to these colours linearly. The final definition of the colours is usually done using the RGB colour model (Red, Green, Blue) where each of the three values is either given as an 8 bit integer with the range [0 : 255] or as a floating point number with the range [0.0 : 1.0]. The RGB colour model is inherently the representation of the colours in most software systems and OpenGL and in many standards, such as VRML and X3D. It is also the representation of colours used by monitors and on graphics boards.

However, in our case we want to represent two data at time:

1. The original data value (e.g. the colour for the stratigraphic unit or the mean of a physical value at a certain point)
2. Its uncertainty (given e.g. as the probability that this point really belongs to the indicated geological unit or as the standard deviation of the physical value)

In this case it is preferable to do the mapping from data to colour in a different colour model first and then to convert the resulting colour into the RGB colour model. One good model for such a task is the HSV (Hue-Saturation-Value) colour model, which is explained in Figure 3.

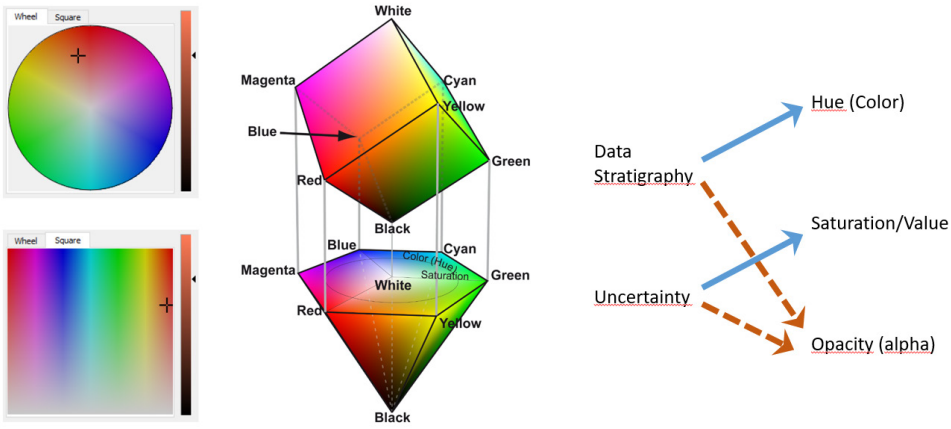

Figure 3: HSV colour model. Top left: representation as a colour wheel (screenshot from Paradigm's Skua-Gocad software). The fully saturated colours are on the circumcircle. The saturation decreases towards the centre of the circle and is zero at its centre. The centre of the circle is actually an axis with colours ranging from white (Saturation is 0 and Value is 1) to black (Saturation and Value are 0). Bottom left: Different representation of the same colour model in Skua-Gocad. The fully saturated colours from the circumcircle were rolled out from left to right while the saturation changes from top to bottom. Right: graphical representation explaining the interrelation of the RGB colour model (cube at the top) and the HSV colour model (up-site-down cone at the bottom), modified from Zehner et al. (2010). For a quantitative explanation and for pseudo-code for HSV to RGB conversion see Foley et al. (1996).

When using the standard representation as a colour wheel (top left of the Figure), one data component is mapped to Hue, which represents a colour on the circumcircle of the HSV colour model. A second component is mapped to the saturation, which means mapping this colour along the radius of the circle. If the second data component is at its lower threshold, it is mapped to a saturation of zero and we are at the centre of the circle (white, grey or black, depending on Value, chosen on the linear slider on the right). When the colour has been determined in the HSV colour model, it can easily be converted to RGB colours, using standard procedures which are, for example, given in Foley at al. (1996), including a code example.

Using colour mapping in order to represent scalar data is a very general method that can be applied on a wide range of geometries and on maps. We could, for example, have a triangle-surface (horizon) that is coloured according to its data value and fades into grey where the data value is increasingly unknown. Figure 4 shows an example of a geological cross section where the colours represent the different geological units, which are uncertain near to the stratigraphic interfaces and to the right of the section. For further examples see Zehner et al (2010) and Hengl (2003). Zehner et al. (2010) also gives an example of how a user-interface that allows for interactive investigation of the data set might look.

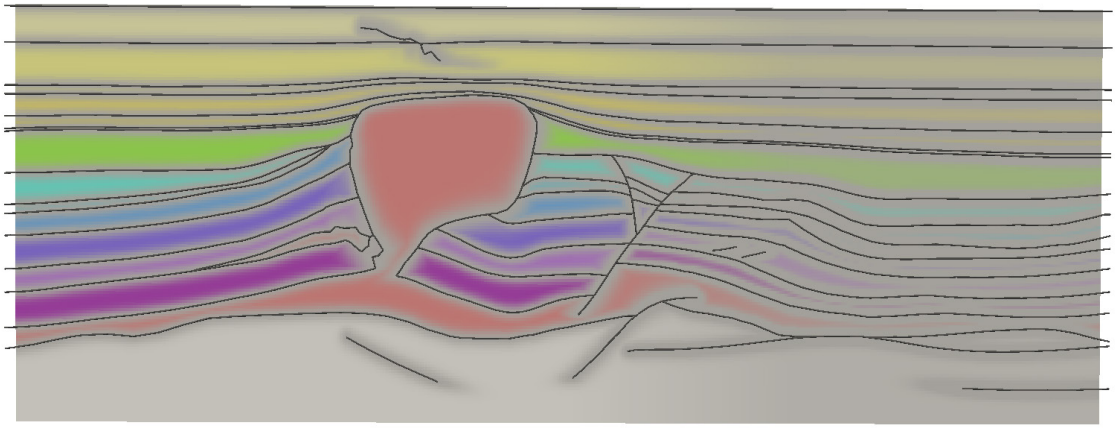

Figure 4: Example of a visualization of a cross section with uncertainty. The uncertainty is higher near to the stratigraphic interfaces and the faults than within the units and overall increases to the right, which, for example, could be due to the lack of data. The cross section shown is an extraction from a voxel data set.

Requirements for the user interface

In order to find the optimal settings for the colour mapping, a user interface is needed that allows us to fine tune the different settings and gives an immediate response. Figure 5 shows an example of how a user interface for a 2D transfer function could look like, inspired by the user interface used in Zehner et al. (2010). Each data value that should be mapped to a unique colour consists of a pair - the actual data value and its uncertainty. The distribution of the data values and of the uncertainties could be shown as histograms to the user, in order to make a more informed decision when adjusting the settings for the colour mapping. Often there are some very high or some very low data values that do not need to be distinguished, so that it is possible to clamp the data range which is mapped to the colour, in order to make the spatial distribution

of the main bulk of the data more visible. The same can be done with the colour mapping of the uncertainty. A certain degree of uncertainty might be acceptable, allowing to clamp the colour mapping for all values with lower uncertainty. The same could be done for high uncertainties. In this way patterns in the distribution of the uncertainty become more visible.

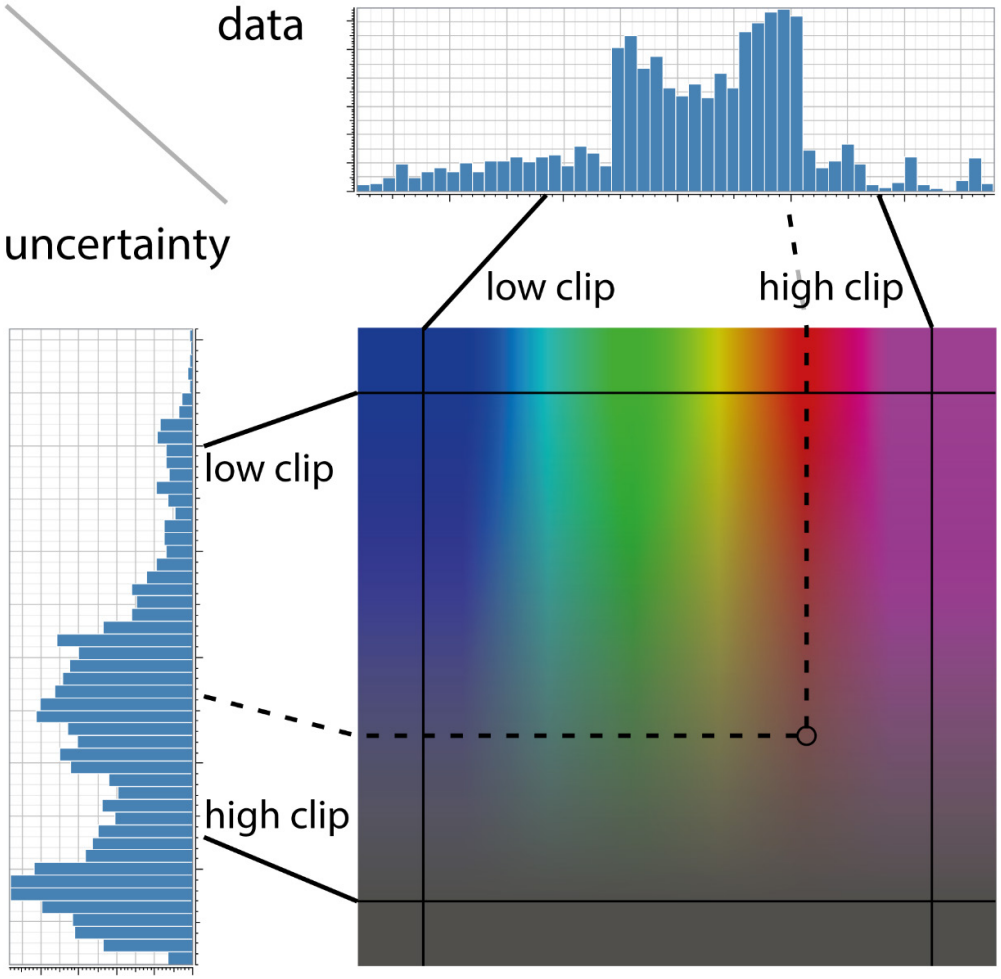

Figure 5: Diagram explaining the different settings that can be used to influence the colour mapping. Each datum is represented by a pair - the actual data value and its uncertainty. Often, a few very high or very low data values do not need to be highlighted to be distinguished from the mass of data, so that the colour mapping can be clamped, making the spatial variation in the bulk of data more visible. The same might be true for very low and very high uncertainties.

The values that the users must be able to set interactively are the low and high clip values for the data and for the uncertainty. The resulting colour-map that depends on the data values is then shown and the visualization can be adjusted accordingly. Further it would be preferable for the user to load predefined colour maps, which would require that they are converted properly into a HSV representation, and that the user can also use categorical data and a simplified colour map.

Precomputed colours versus on the fly colour mapping

It would be possible to apply the colour to the data directly and to store a geo-referenced colour image instead of a 3D model with colour on its vertices or a model or a map with data. This would allow us to use standards that are targeted at visualization purposes, such as image formats (for maps) or X3D for 3D models. However, this approach has two disadvantages. Firstly the users who retrieve the data only obtain a representation of the model. They cannot sample the attribute's values from the model, as the model does not know the original attribute – it only knows the colours. Secondly the colour map that has been chosen must be chosen in a way that is optimal for looking at the whole model but when looking at details or at sub-regions, another colour map would possibly be preferable. To give an example, let us assume we have a large field with the data and its uncertainty clustering around one mean in a region in the north of the field and around another mean in the south of the field (the histogram has a bivariate distribution). If we would investigate the north region for data distribution, we would use a different colour table for the north region than for the south region in order to spread the colour range that corresponds to each of the data ranges and thus shows more detail. For this reason it makes sense to transmit the original data and to create the visualization (colour mapping) on the fly using lookup-tables or transfer functions. The users could then interactively play with the data and the colour table to find the best settings in order to extract the important information in their region of interest.

2.3.2 Use of translucency

Translucency could, for example, be used to indicate the uncertain presence of a geologic feature, such as a horizon. When the uncertainty with respect to the occurrence of a horizon at a certain position increases, it becomes more and more translucent and so fades away. Translucency can usually be defined for each vertex or primitive (triangle or quadrangle) and is then part of the colour definition. Instead of a three component vector with the colours (RGB), a four component vector is given where the last component defines the translucency (RGBA, where A stands for alpha value). Ideally the translucency is defined interactively by a transfer function on an attribute that could represent the uncertainty, such as variance or standard deviation. Translucency can be used very nicely to indicate that the presence of some geologic feature, e.g. a horizon or a fault, is uncertain (which means that we are unsure if it even exists at the given location).

2.3.3 Usage of glyphs and additional geometry

Glyphs and additional geometry can be used to indicate within which range a point, a vertex of a horizon, a line or a surface might be displaced in 3D space. Examples could be:

- Several semi-translucent spheres or ellipsoids that are rendered as hulls around points, such as borehole markers, to indicate different confidence intervals.
- Several semi-transparent tubes that are rendered as hulls around lines, such as a borehole path, to indicate different confidence intervals.
- Several semi-transparent surfaces that act as a confidence envelope around a surface in 3D space representing the equivalent of a confidence interval.

Figure 6, Figure 7 and Figure 8 show examples of this type of uncertainty visualization using additional geometry. They are explained in more detail in the state of the art report on

uncertainty visualization (Zehner, 2019). Some of these geometries could be precomputed and stored in the database as additional geometries. This could be done, for example, when the generation does not only depend on local data and thus is too complicated, so that it cannot be expected to be done by the 3D viewer.

Glyphs can easily be computed locally and applied to represent the uncertainty when it is given for each vertex. The glyph is given as a small triangulated surface, for example a triangulated sphere with a radius of 1 and its centre at the origin. When the geometrical uncertainty is given along a line or along the vertical the glyph must be rotated and properly aligned and subsequently scaled by a multiple of the standard deviation, depending on which confidence interval should be shown. Finally it is transformed to the position of the vertex. When the uncertainty is given as a real 3x3 matrix, the vertices of the glyph can be transformed with the 3x3 matrix that describes the standard deviation, using matrix multiplication. Then the sphere (now an ellipsoid) can be scaled to describe a certain confidence interval and finally be translated to the location of the point. The same technique could be applied to place and orient 2D glyphs on a map, using a 2x2 matrix.

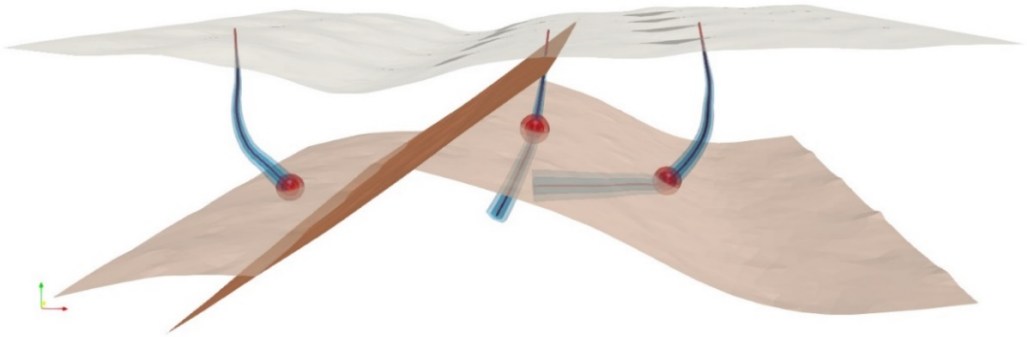

Figure 6: Visualization of points (here borehole markers) and lines (here the borehole paths) with geometric uncertainty assuming that the uncertainty is isotropic.

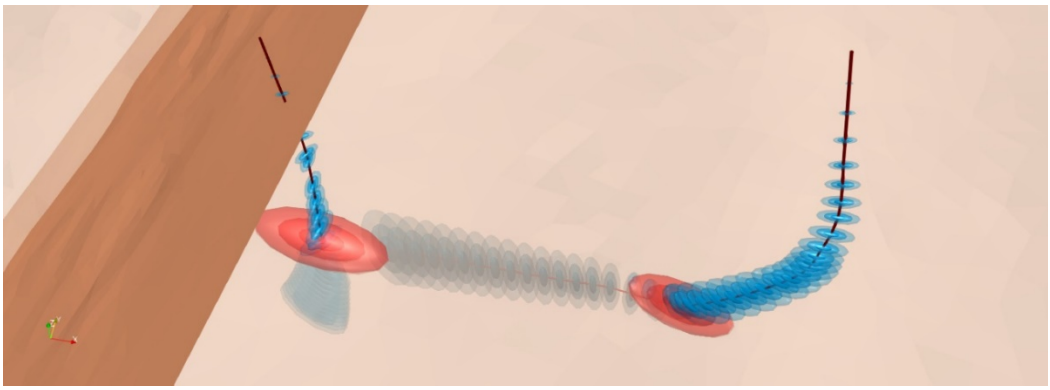

Figure 7: Visualization of points (here borehole markers) and lines (here the borehole paths) with the uncertainty of the locations described as a tensor (anisotropic uncertainty).

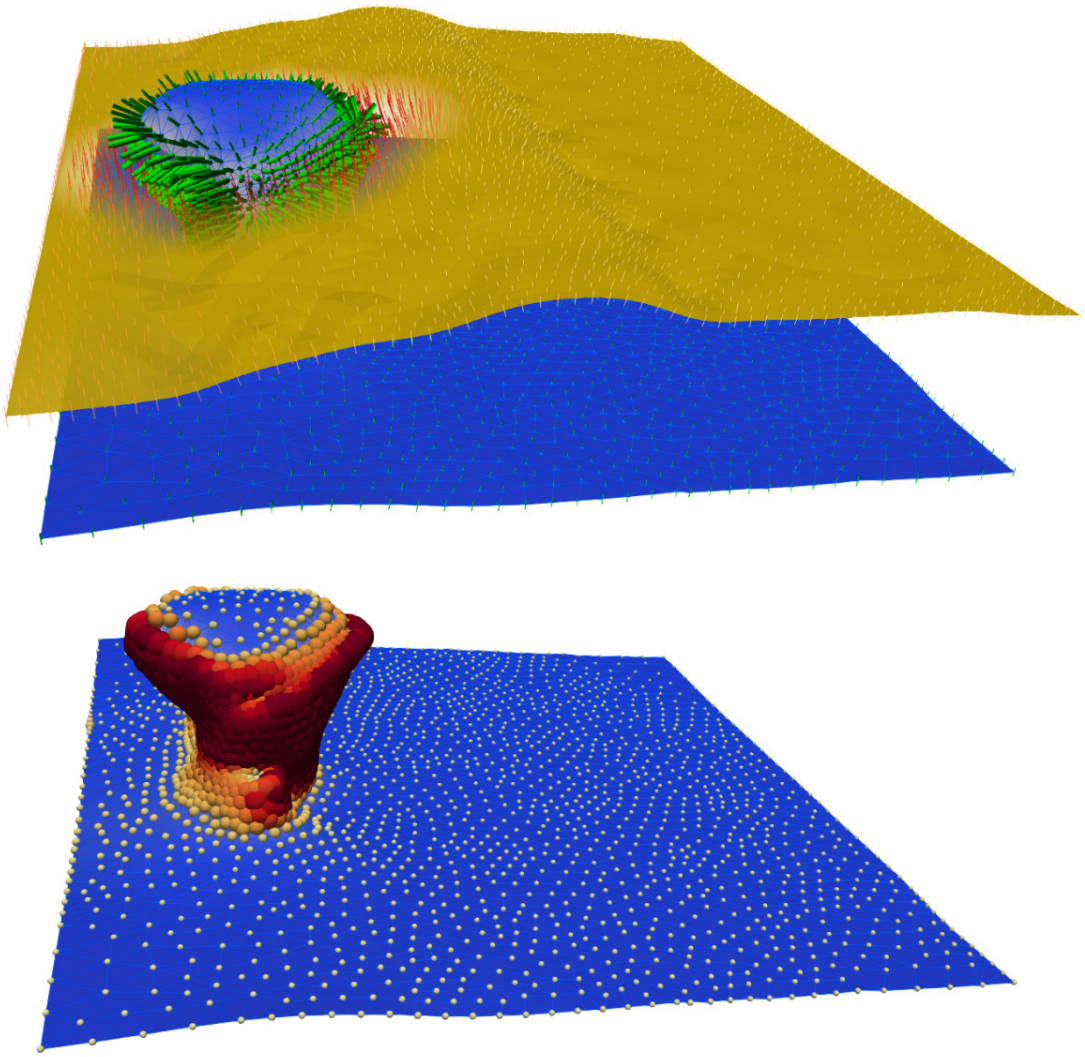

Figure 8: Examples of the visualization of geometric uncertainty on surfaces for a synthetic geoscience data set. The assumption is that the position of the boundary between the different units is less well known when it is steep or vertical, because it is often not visible in seismics in this case. Top picture: The upper horizon becomes increasingly translucent when its presence is less secure. Further the possible displacement is indicated by lines (needles). For the lower horizon (salt) the colour becomes less saturated (grey) with decreasing knowledge of its position. The possible displacement is indicated by the green cylinder. Bottom picture: The possible displacement of the horizon is indicated by the size and colouring of sphere glyphs.

It should be noted that the use of glyphs for data representation is not specific for the representation of uncertainty. In particular the use of glyphs that are specifically set at each

vertex location can be used for a wide range of applications, such as the visualization of a vector field that indicates fluid flow patterns.

Requirements for the user interface

For the user interface we have to distinguish if the geometries that represent the uncertainty have been precomputed or not.

For precomputed geometries, e.g. the envelopes around surfaces or lines, the different geometries should be grouped as one object in the user interface. To give an example: If the users select an object, e.g. a borehole, they are shown several check boxes that allow them to switch the representations of the borehole path and the 50%, 75% and 90% confidence envelope on and off. Another option would be to use a hierarchical tree structure where choosing an object (e.g. the borehole) flips open the next hierarchy allowing to switch on and off the path and the confidence intervals.

When the uncertainty is given per vertex, the geometries to visualize it could be computed on the fly. The user interface should then allow the users to set several parameters:

- Choose the shown glyph (e.g. line, arrow, sphere …) from a list of glyphs.
- Set a global scale factor that is applied to all glyphs.
- Choose the vector attribute that is used to align the glyph, e.g. the surface normal.
- Choose the attribute which is used to define the transformation for each individual glyph (the attribute containing the 3x3 or 2x2 matrix or a scalar attribute that is used for isotropic scaling).

One example of a software that allows us to set these parameters and so could serve as an example is the software Paraview (www.paraview.org). When uncertainty is isotropic or aligned to a certain vector attribute, such as the surface normal, Paraview's glyph filter can be used and two attributes can be chosen - a vector attribute for the alignment and a scalar attribute for the scaling. When the uncertainty is anisotropic (given as a 3x3 matrix), Paraview's tensor-glyph filter can be employed which directly uses the matrix and can be scaled additionally to represent the desired confidence interval.

Precomputed versus on the fly computation of geometry

This question is only relevant for glyphs that are computed locally for each vertex, as the computation of geometry that does not only depend on the local uncertainty is quite complex and would exceed what could be expected from a 3D viewer. In the case of glyphs that should be added for each vertex, the computation could be implemented quite easily. This would allow the users to analyse interactively the data on a more local level. They could, for example, interactively choose an appropriate scale factor, so that glyphs are rendered in a size that shows the uncertainty pattern in a certain region especially well. However, the advantage is less obvious than in the case of colour mapping.

2.4 Application Programming Interface (API)

Many of the visualization methods that have been described in Zehner (2021) and Zehner (2019) would require too much effort to be implemented in the 3D viewer for the EGDI. Examples are

the use of volume visualization to represent the uncertainty in volumetric models or the visualization of vector fields with uncertainty which is not relevant within the GeoERA project 3DGEO-EU, but might become relevant in the future. When the data model can handle attributes with multiple components, e.g. vectors, and does support the above mentioned types of uncertainty, it could already handle these requirements that might become interesting in the future.

In this case it would be beneficial for the 3D viewer to also be extended to render this new uncertain information. This extensibility could either be given in the form of a plugin mechanism or by developing the 3D viewer in an object oriented manner as open source software.

3 EXAMPLE VISUALIZATIONS USING WEB-TECHNOLOGIES

Several example visualizations have been created in order to demonstrate different visualization options for uncertain geological models in conjunction with the EU report on the state of the art in uncertainty visualization (see Zehner, 2019). While these visualizations are based on the publicly available software Paraview, they require that the users install Paraview and acquire some minimum knowledge of how to use it. As Paraview is a general software for scientific visualization, this can already become quite confusing for people who neither have experience with 3D modeling nor with 3D visualization, and the level of reluctance can be assumed to be quite high. Further, the models need to be converted into VTK format which is, thus far, not a standard format for geoscience applications. Another problem with Paraview is that it does not support interactive manipulation of 2D colour tables or transfer functions, which would be a plus when visualizing uncertain data.

So, as a next step towards making these visualization methods available in the EGDI, some of the example visualizations have been implemented as web applications in javascript and based on the scenegraph Three.js (www.threejs.org). These applications run interactively in a browser and allow for the interactive manipulation of the 2D colour map. The implementation has been done in such a way that the functionality is encapsulated in classes that are mostly derived from Three's standard classes. Instead of the Geometry class of Three.js we implemented our own classes to display different data types. They are derived from Geometry but hold not only the usual vertices and triangle lists but also additional data that describe attributes and their uncertainty for each vertex or voxel. Attributes can then be used to colour the vertices with the help of a 2D colour table or to compute additional glyphs or envelope surfaces in order to express the geometrical uncertainty of the surface shown. This implementation should make it easy and efficient to incorporate the visualization method into other 3D software that has been created based on the same scenegraph. However, this might not hold for the user interface required to interact with the objects. In the following, we describe the generated classes in more detail.

2D colour tables

Two classes have been implemented for 2D colour mapping. The first class, *ScalarColorTable2D*, maps a scalar attribute, that needs to be shown, to a saturated colour and fades the colour into grey as the value of another attribute increases that is usually a measure of the uncertainty of the first data value (such as standard deviation, variance or information entropy). In this way, the data is mapped to a saturated colour, where it is certain, and appears greyish/dirty, where it is uncertain (see Figure 9 for an example). In the first example (top) the histogram for the uncertainty (aligned vertically to the left of the table) shows that there are few points with a very high uncertainty which are responsible for the fact that the intermediate uncertainties near to the faults are less visible. In such cases it makes sense to clip the high end of the uncertainty mapping to lower values, so that the regions with intermediate uncertainty become more visible (bottom).

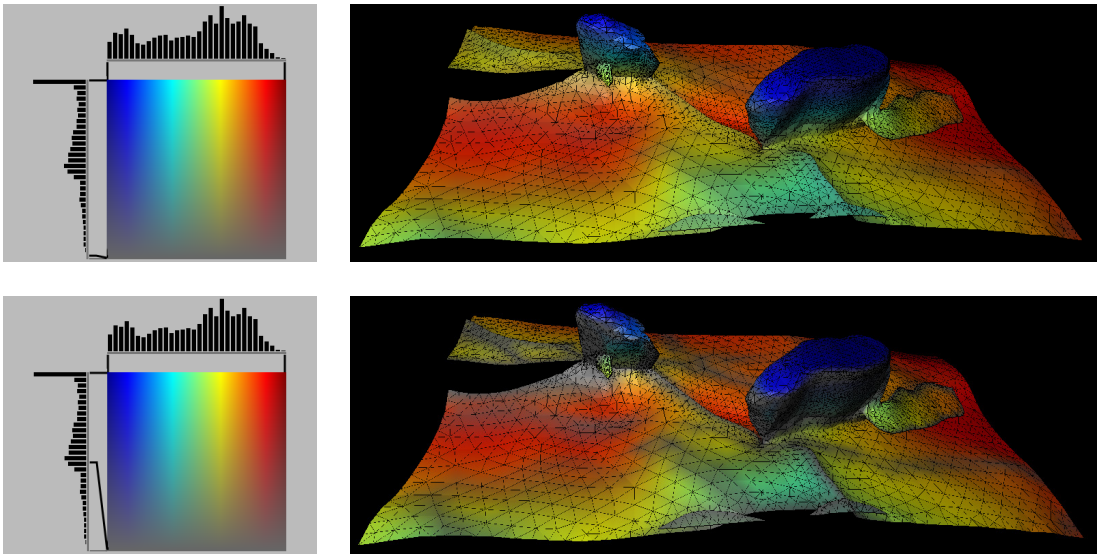

Figure 9: Mapping of data/uncertainty pairs as colour onto the vertices, using a 2D colour table.

The second colour-table-class, *CategoricalColorTable2D*, maps Integer numbers that represent Ids, for example for the stratigraphy or the facies, to pre-defined colours. These colours are faded into grey with increasing uncertainty (see Figure 10).

Both tables can be used in two different ways. The first one uses the tables directly via their data values. The clip values for the data attribute and the uncertainty attribute are set. Then the 2D colour table can be used to obtain a colour for a given [data value, uncertainty value] pair and to set this colour for the corresponding vertex. The other option is to use texture mapping. Instead of setting the vertex colour for each vertex of the object, the colour map is set as texture for the object, and the data value und uncertainty value are mapped to texture coordinates. One advantage of the texture mapping way would be that alpha mapping could be used to make the object semi-translucent in dependency of the data value. Using Three.js this could not be done when colouring the surface using vertex or face colours, as Three.js only supports 3-component colours for this purpose (no RGBA colours with alpha value).

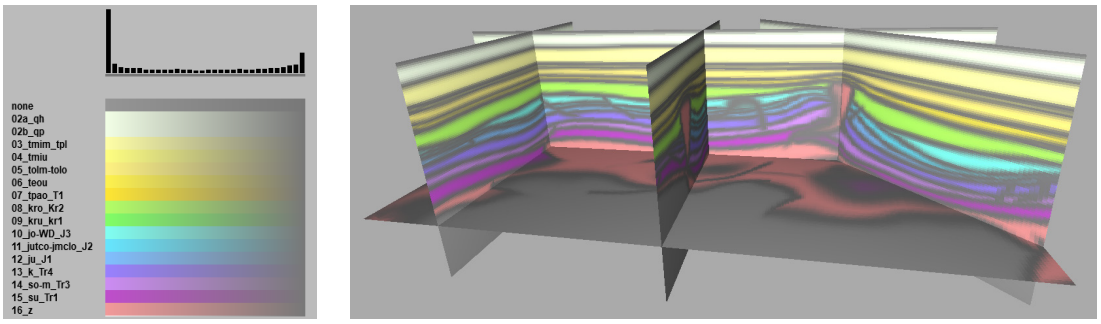

Figure 10: Mapping of categorical uncertain data (stratigraphy), using a 2D colour table.

The colour tables store the different settings that a user has chosen, for example the values for which high uncertainties are clipped, and so one table is generated for each variable. The generation is done automatically by using a global instance of the class *ColorTableFactory* which returns a colour table for a given variable name. If no colour table has been predefined for a given variable, the *ColorTableFactory* returns the default colour table.

3.1 Visualization of triangle meshes with uncertain geometry and data

This example shows how the typical 3D triangular meshes that are often generated using the software Skua-Gocad could be visualized when they have additional data that should be shown together with their uncertainty or when the geometry itself is uncertain. The visualization is mainly based on two classes, the class TrglGeometryWithData, derived from THREE.Geometry and the class TrglGeometryWithDataGroup, derived from the Three.Group class.

TrglGeometryWithData

This class represents a triangular mesh in 3D space and is derived from the Geometry class of Three.js. In addition to the Geometry class it handles additional attributes (currently continuous scalar data as float numbers) which belong to each vertex. The data are read from a Skua-Gocad TSurf ASCII file or directly from the EGDI database (see below). Using the additional data, an object of the above described ColorTable class can be used to colour the surface accordingly, either only by using it as a 1D colour table for the data values or by incorporating a second attribute that describes the uncertainty of the data values.

When the uncertainty data represent geometric uncertainty, for example as a standard deviation that defines how much a vertex might be displaced perpendicular to the surface, the uncertainty could be shown in the form of additional geometry. The TrglGeometryWithData class provides two functions that calculate such a geometrical representation and returns it as an object of the type Three.Geometry. The first one computes glyphs (conical cylinders) that point along the surface normal and indicate with their length how much the surface might be displaced with a certain confidence. The second one calculates two surfaces (the envelope), one above and one below the original surface (or one outside and one inside the salt dome in this case) by displacing the surface along the surface normals with a magnitude that depends on the given standard deviation per vertex. In the example, the length of the glyphs and the distance of the envelope surface to the original surface are set to 1.96 * standard deviation which corresponds to the 95% confidence interval. See Figure 11 for screenshots.

For horizons it may furthermore be necessary to indicate the geometric uncertainty but also to indicate that the occurrence of the horizon is unsure at a specific location. When a horizon is, for example, in contact with a fault or with a salt diapir for which the extent is unknown, then the extent of the horizon is unknown too. To indicate this, a variable called pr_presence can be defined that specifies the probability that a horizon occurs at a specific location. Generally a good way to display this uncertainty would be to fade the horizon into translucency with decreasing probability of occurring. However, as Three.js does not support 4 component (RGBA) colours on the vertex level, we instead fade the horizon to white where the probability of occurrence is low (see Figure 12).

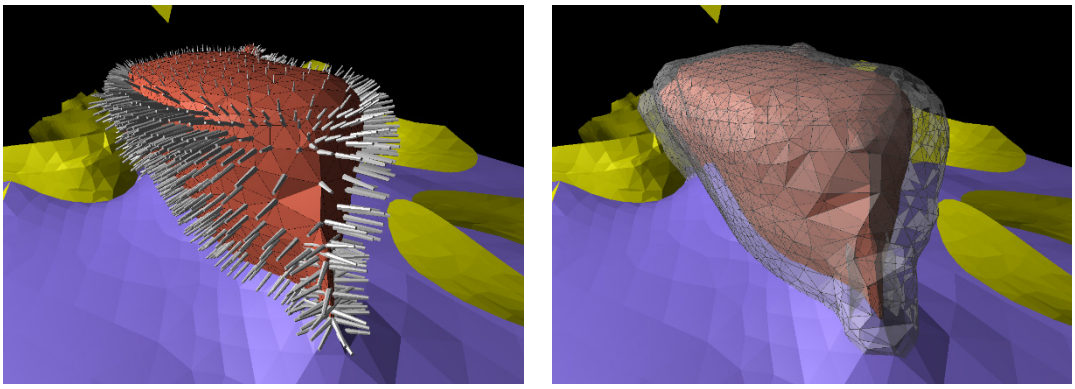

Figure 11: Visualization of geometrical uncertainty, using glyphs (left) and a translucent envelope surface (right), both indicating the 95% confidence interval.

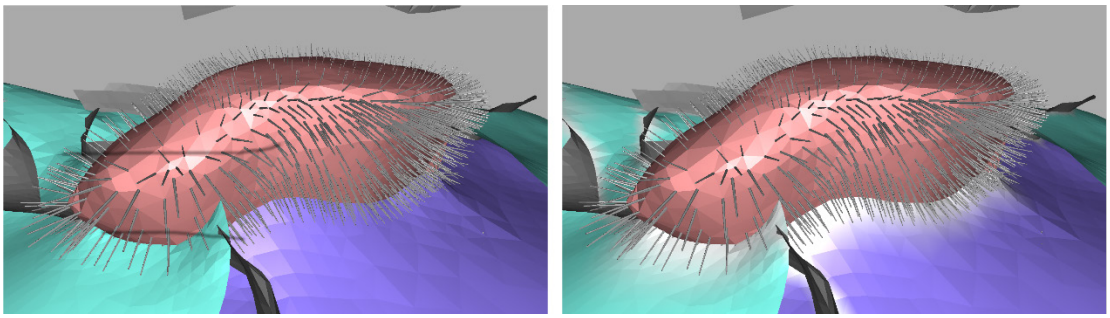

Figure 12: The occurrence of the horizons is unsure near to the salt diapirs as the extent of the diaper is unknown (indicated by the glyphs). For comparison the left image shows the original visualization while the right image indicates this uncertainty by fading the horizons to white near to the salt diapir.

Currently the TrglGeometryWithData class only handles data that are attached to the vertices and uses vertex colours to show these data, but it could easily be extended to also handle data that are attached to the faces (primitives, here the triangles).

TrglGeometryWithDataGroup

This class is derived from the Group class of Three.js and can be directly included in its standard scenegraph. It works as some kind of wrapper or container around the TrglGeometryWithData class – all function-calls should be directed to this class which will advance these calls to the TrglGeometryWithData class where necessary. When, for example, the glyphs should be shown, the corresponding function should be called on this class. It will then check if they can be shown directly or if they first need to be computed. It further keeps track of additional geometries, such as the envelopes and glyphs that have been described above.

3.2 Visualization of voxel data with uncertain scalar data

This example shows how the voxel data sets which contain data for each voxel, for example an integer id that defines the class to which this voxel belongs (e.g. geological unit) or some physical attribute (e.g. porosity or permeability) and the corresponding uncertainty (e.g. probability or information entropy) can be visualized. The visualization is mainly based on two classes, the class VoxelGeometryWithData which is derived from THREE.Geometry and handles the data and the visualization using sections through the data set, and the class VoxelGeometryWithDataGroup which is derived from the Three.Group class and provides the interface to the main application.

VoxelGeometryWithData

This class reads the scalar data from a file in an xml based vtkStructuredPoints format (a format to describe voxel data from the Visualization Toolkit, Schroeder et al., 1997) and stores them in a 2D array. At this stage no difference is made between the attribute data themselves and the corresponding uncertainty data. Different functions have been implemented to obtain information on the data, such as attribute names, minimum and maximum values and the histogram for each attribute.

In order to visualize the data, they are converted into unsigned byte format (Uint8, 8 bit) and stored in a 3D texture (basically a 3D image). If only the data value needs to be visualized with a 1D colour table, a 3D texture with one component is used. When, additionally, the uncertainty has to be shown, using a 2D colour table, a 3D texture with two components is used (Luminance-Alpha format). The 3D texture can be visualized by slicing sections that are made of geometric primitives (triangles) through the 3D texture (see Figure 10 for an example). Several functions are provided, which define either vertical sections that can be rotated around the z-axis and shifted along x and y axis, or horizontal slices that can be shifted along the z axis.

Unfortunately Three.js does not support 3D texture coordinates and for this reason the actual visualization must be calculated in a shader, defined as *VolumeSectionShader*, using OpenGL shading language (see e.g. Rost, 2006). The texture coordinates are computed from the vertex coordinates on the fly within the vertex shader and then set as varying variable, so that they are interpolated for each pixel in image space that “sees” the current primitive. The texture with the data is accessed in the fragment shader, using the interpolated texture coordinate. Dependent of the data value (and the uncertainty in the case of a 2D colour table) a second texture coordinate is computed which points into the colour table as a 2D texture. For the resulting colour value the lighting calculations are done subsequently in the fragment shader on a per-fragment basis.

VoxelGeometryWithDataGroup

This class is derived from the Group class of Three.js and can be directly included in its standard scenegraph. It works as some kind of wrapper or container around the TrglGeometryWithData class – all function-calls should be directed to this class, which will advance these calls to the VoxelGeometryWithData class where necessary. The material that is used by this class to render the geometries is a THREE.Shadermaterial that uses the *uniform variables* and *textures* from the corresponding *VoxelGeometryWithData* instance. Principally any geometry that has the vertex

coordinates defined in the same coordinate system as the *VoxelGeometryWithData* instance can be used to visualize the data on its surface.

3.3 SceneViewer and global variables

Several variables are defined globally to make them available from different parts of the implementation. Normally, in object oriented programming, these would likely be implemented using the Singleton design pattern (e.g. Gamma et al., 1995).

- **allGeoDataGroups:** is a global variable that holds all the different geoscientific data objects, that are derived from *THREE.Group*, such as *TrglGeometryWithDataGroup* and *VoxelGeometryWithDataGroup.* The variable *allGeoDataGroups* itself is of type *THREE.Group* itself and is included in the scenegraph. Basically it constitutes the root of the geoscientific scene objects.
- **colorTableFactory:** is a global variable of the class *ColorTableFactory* and is used to generate colour tables for the different attributes that are loaded with the geoscientific geometries. If no pre-defined colour table exists for the given attribute name, an instance with the standard colour table will be created and returned.
- **sceneViewer:** is an instance of the class *SceneViewer* and encapsulates the 3D visualization canvas. It provides the functionality to steer the camera (e.g. view from top, view from east), to handle changes in the window size and to adjust the camera and scale the whole scene to get a global view.

3.4 Data-loaders

The data-loader classes implement the functionality to load the test data in order to showcase some of the suggested visualization methods from files or from the official data base.

EGDILoader

Within the EGDI 3D database, the vertices and the triangle indices that indicate which three of all the vertices form a triangle, are stored in dedicated places of the data base and are accessed from there using the object-id that is issued for each 3D object in the 3D database. In order to find these data there, the URL must be set first. It differs for the standard data sets which have no additional data on their vertices and for the experimental data sets that carry additional information, such as standard deviation and the probability that the surface exists at the vertices location.

- Vertices for standard models: “https://geusegdi01.geus.dk/geom3d/data/nodes/”
- Vertices for uncertain models: “https://geusegdi01.geus.dk/geom3d/data/probnodes/”
- Indices for all models: “https://geusegdi01.geus.dk/geom3d/data/triangles/”

These URLs must be handed to the EGDILoader before loading the model, using its member functions *setEGDIPointUrl()* and *setEGDIIndexUrl()*;

Generally, loading the data is a two-step process. First the meta-data are loaded from the data base, given a general model ID that is unique for each model. They contain information about the type and name of the different objects, also called features, that belong to a model, the

colour with which they should be displayed and their geometry-ID under which the vertices and the indices can be retrieved from the EGDI 3D database. Subsequently the geometry for each feature (here the triangulated surface) is pulled from the data base, using the information and the geometry-ID. Then a new object of type TrglGeometryWithDataGroup that encapsulates the surface is created, and appended to all other data (allGeoDataGroups, see above).

As things currently stand, the metadata in the EGDI 3D database do not carry information about additional variables for each vertex of a feature exist, such as standard deviation and the probability that it occurs at the vertices positions, and of which type they are. So currently this must be handled by the application in advance. When no additional data need to be loaded, then another function-call is used which loads the model in a standard way. When additional data are present, the function must be given the number of data per vertex and the names of the data, in order to later present it properly. Future versions of the EGDI 3D database need to solve this and to provide this information in the metadata, so that a universal loader can be written.

4 EXAMPLE DATA SETS

In order to showcase the different visualization methods and in order to test and discuss them, example data are needed. For this reason we have created two example data sets. The first one, a 10x20 km pilot region in the German North Sea (Entenschnabel), has been taken from another project and augmented with uncertainty making some general assumptions instead of a detailed uncertainty assessment. This data set has been taken throughout the project, in order to generate example renderings, such as for Deliverable 4.1, the report on the state of the art in uncertainty visualization. The second example data set is the result of an assessment of uncertainty for the 3D depth model that has been generated from the cross border Dutch, German and Danish sector around the Entenschnabel within work package 3 of the 3DGEO-EU project. The uncertainty assessment for this model has been done at a later stage of the 3DGEO-EU project and has therefore not been used to create illustrations for the reports. Both data sets are stored in the EGDI 3D database and together form Deliverable 4.4 of this 3DGEO-EU work package on the uncertainty in geomodels. Currently EGDI only stores triangle-meshes with additional attributes that represent scalar values. Additional example data sets that have been used throughout the project, such as lines and points with standard deviations given as 3x3 matrices or voxel data sets, could not be stored in EGDI. For this reason, they have not been included here – but it can be assumed that they will be stored in EGDI in the future.

4.1 Example data: 10x20 km Pilotregion Entenschnabel

The small data set is based on the 3D geological model of the 10x20 km large pilot region in the German North Sea sector (see Zehner, 2018 for detailed information). The model consists of 16 horizons, marking the base of the corresponding units, 43 faults and two salt-diapirs, called Berta and Bella and is shown in Figure 13. Further a visualization of a part of this model showing the uncertainty and using the javascript implementation described above is shown in Figure 14.

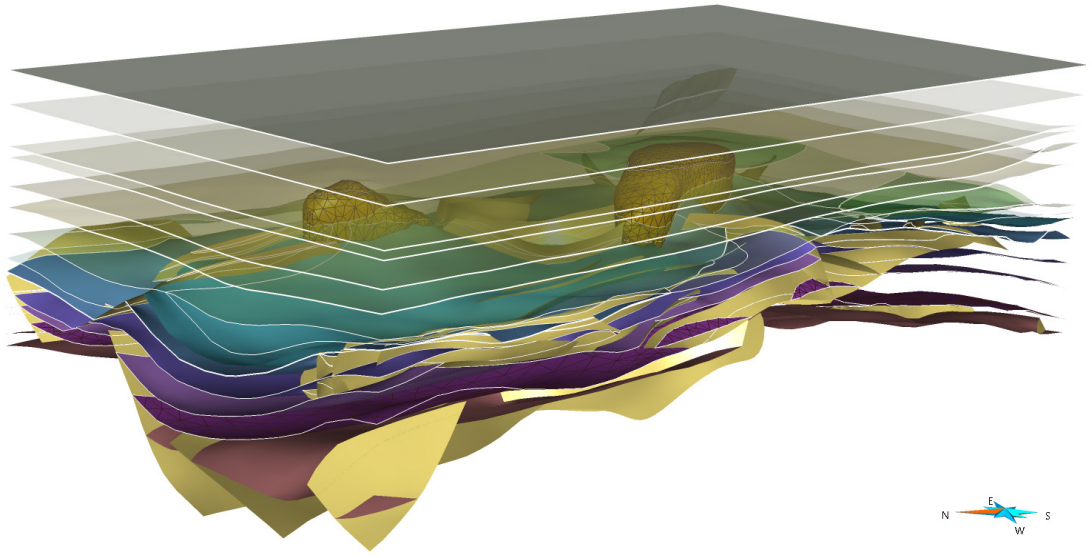

Figure 13: 3D geological model from Zehner (2018) that has been used as example data set.

In reality, the construction of the original model did not involve an uncertainty assessment. For demonstration purposes, information on uncertainty was artificially generated and calculated by using general assumptions. For the horizons the uncertainty is assumed to be much higher in the vicinity of the salt diapirs, as has been shown in Deliverable 4.2 (Zehner et al., 2021) by statistically analyzing misties of crossing 2D seismic sections. Further the vicinity of faults is assumed to add some uncertainty and a small general increase of uncertainty with depth is assumed. The relative uncertainty is then multiplied by an assumed maximal standard deviation of 200m for the horizons to generate the standard deviation for each vertex. Further a variable pr_presence has been created that assumes that the probability for the horizon to be pictured on the right side of the fault or existing at this location outside of the salt diapir to fall from 1 at a distance of 150m (for faults) or 250m (for the flanks of the diapirs) to 0 at the position of the faults and diapirs (due to the exact position of the faults and the exact extension of the diapir to be unknown).

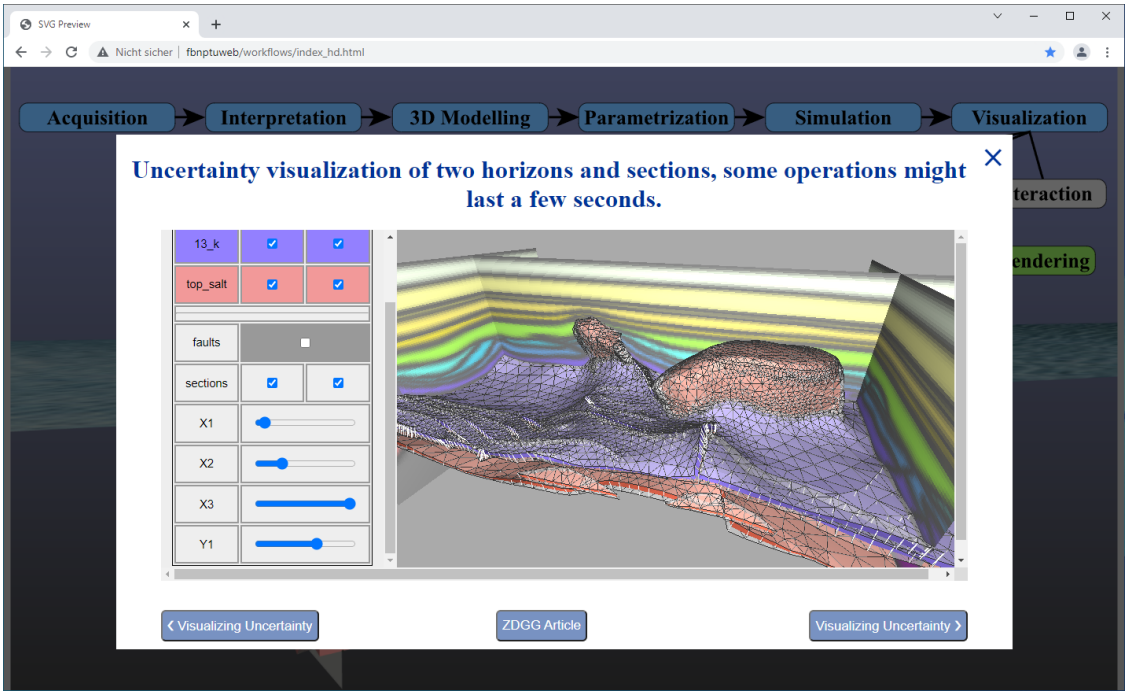

Figure 14: Visualization of the geological model with uncertainty using the Javascript prototype that has been developed as part of the work package (embedded in a webpage).

The uncertainty of the salt diapirs has been assumed to be mainly proportional to the steepness of the flanks, as the vertical walls of the diapir can usually not be imaged in seismics properly. The geometric standard deviation of the extension of the diapirs is assumed to vary between 40m on the horizontal interfaces (e.g. top) and 200m on the vertical flanks. For the faults the geometric standard deviation has been assumed to be 100m.

The model has been used as example data for visualizations using Paraview and for testing how a model could be stored in the EGDI 3D database and subsequently directly could be drawn from there into our web based prototype for uncertainty visualization.

4.2 Example data: Model of the Entenschnabel from WP3

The original (certain) data represent an offshore cross border North Sea area between the Netherlands, Germany and Denmark, and are one of the results of work package 3 of the 3DGEO-EU project. The modeling and harmonization of the horizons in the time domain is described in Deliverable 3.6 (Thöle et al., 2021), the time to depth conversion has been done based on the velocity model described in Deliverable 3.7 (Doornenbal et al., 2021) and the harmonized depth model, together with the report, constitutes Deliverable 3.8 (Thöle et al, 2021).

These data have been used by TNO, Maryke den Dulk, to apply the workflow for uncertainty assessment as described by her in Section 3.2 of Deliverable 4.2 (Zehner et al., 2021). Due to data availability and time constraints during the project, the assessment could only be done for the Dutch and the German part of the model. The original model was given as 2.5D elevation grid with a high resolution. The data were converted to Skua-Gocad's TSurf format and are given in two resolutions - a high resolution version that reflects the original resolution but results in a large number of vertices, and a low resolution version that has been generated in Skua-Gocad by resampling the surfaces while paying respect to their curvature. Both model versions carry two variables on the vertices:

- *z_stddev*: This variable holds the standard deviation for the z-values.
- *z_stddev_is_known*: This variable holds the information if the standard deviation has been estimated. If its value is 0 (false), uncertainty has not been assessed and the value for z_stddev is set to 0.0. If its value is 1 (true), the uncertainty has been assessed and the variable *z_stddev* holds the calculated value.

Please note that this means that the whole Danish part of the model is given a standard deviation of 0.0 as it has not been assessed in this region (*z_stddev_is_know* is 0 / false).

Table 1 lists the ASCII file names, their colour code and the name of the geological units to which the horizons constitute the base.

Filename	Colour	Geological unit
01_NU_V3_TSurf_***.ts		Upper North Sea Group
02_N_V3_TSurf_***.ts		Lower and Middle North Sea Groups
03_CK_V3_TSurf_***.ts		Chalk Group
04_KN_V3_TSurf_***.ts		Rijnland Group
05_S_V3_TSurf_***.ts		Schieland, Scruff and Niedersachsen Groups
06_AT_V3_TSurf_***.ts		Altena Group
07_TR_V3_TSurf_***.ts		Lower Germanic Triassic Group

*Table 1: File names, colour codes and the names of the base horizons / geological units for which the uncertainty has been assessed. The wildcards *** can either be highres for the high resolution model (original resolution) or lowres for the resampled version with lower resolution*

Resampling the surfaces in order to provide the model in a form that uses fewer resources clearly has an effect on the distribution of the data, especially on the standard deviation, as the remeshing has been done by respecting curvature. This primarily makes a difference when looking at the detail, but leaves the overall trends and feature intact. See Figure 15 for a comparison.

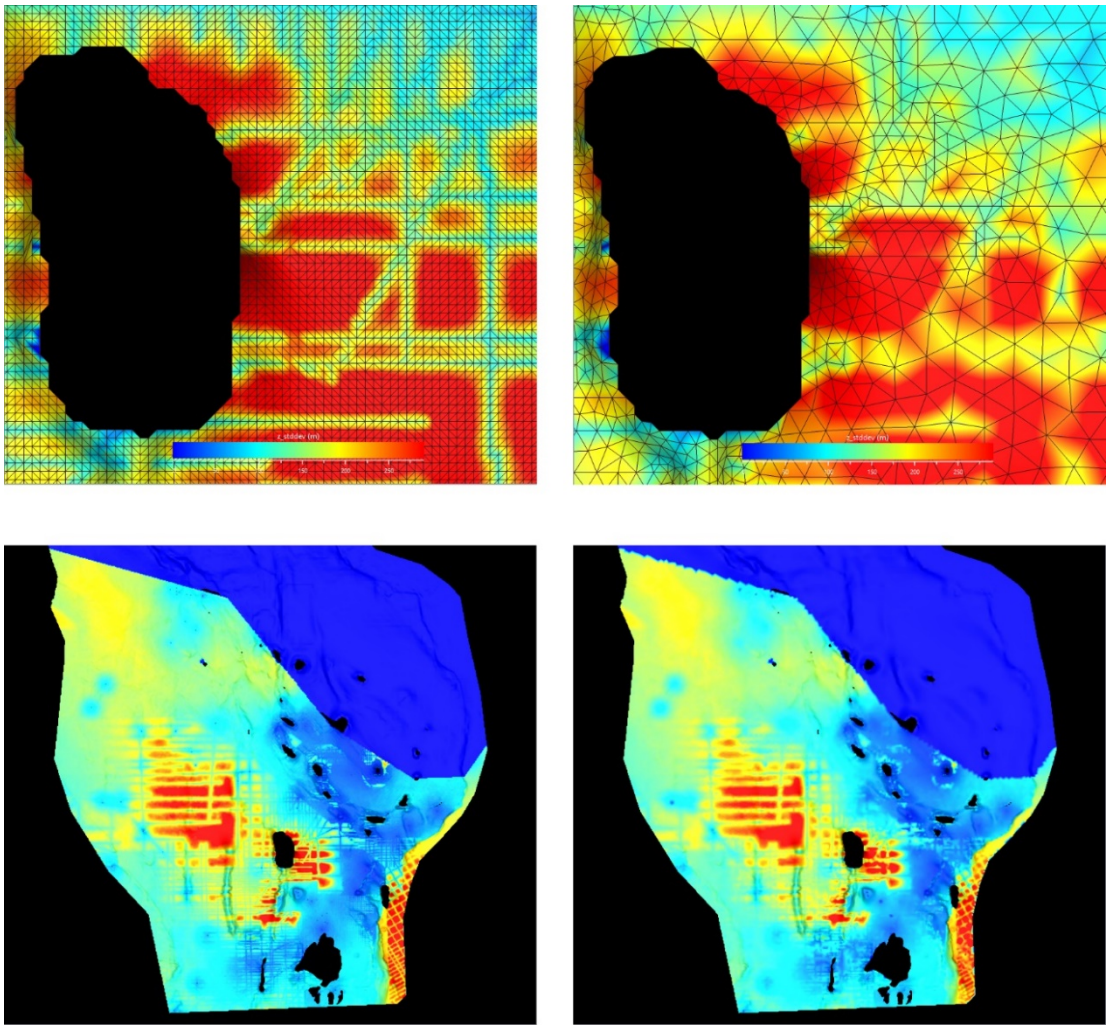

Figure 15: Effects on the spatial distribution due to remeshing the model with a lower resolution. Blue colours indicate low, and red colours indicate hight standard deviation. Top: close up to show the effects of resampling the mesh with the original resolution (left) to one with much lower resolution (right). As can be seen in the bottom of the figure these effects are hardly visible when looking from the distance at the whole model. The large blue area in the North indicates the Danish part of the model where the uncertainty has not been assessed. Therefore the standard deviation has been set to zero.

Figure 16 further shows the visualization of the KN horizon using the prototype and the HSV colour model for 2D colour mapping, in order to show depth and uncertainty simultaneously. The Hue (fully saturated colour) indicates the depth, while the uncertainty is mapped to colour

saturation by fading the colour saturation in areas of high uncertainty (greyish areas). Due to the large regional extent, it does not make sense to indicate the standard deviation by using glyphs or by rendering confidence envelopes, as the size of the glyphs and the distance from envelope to mean horizon would be very small in comparison to the spatial extents.

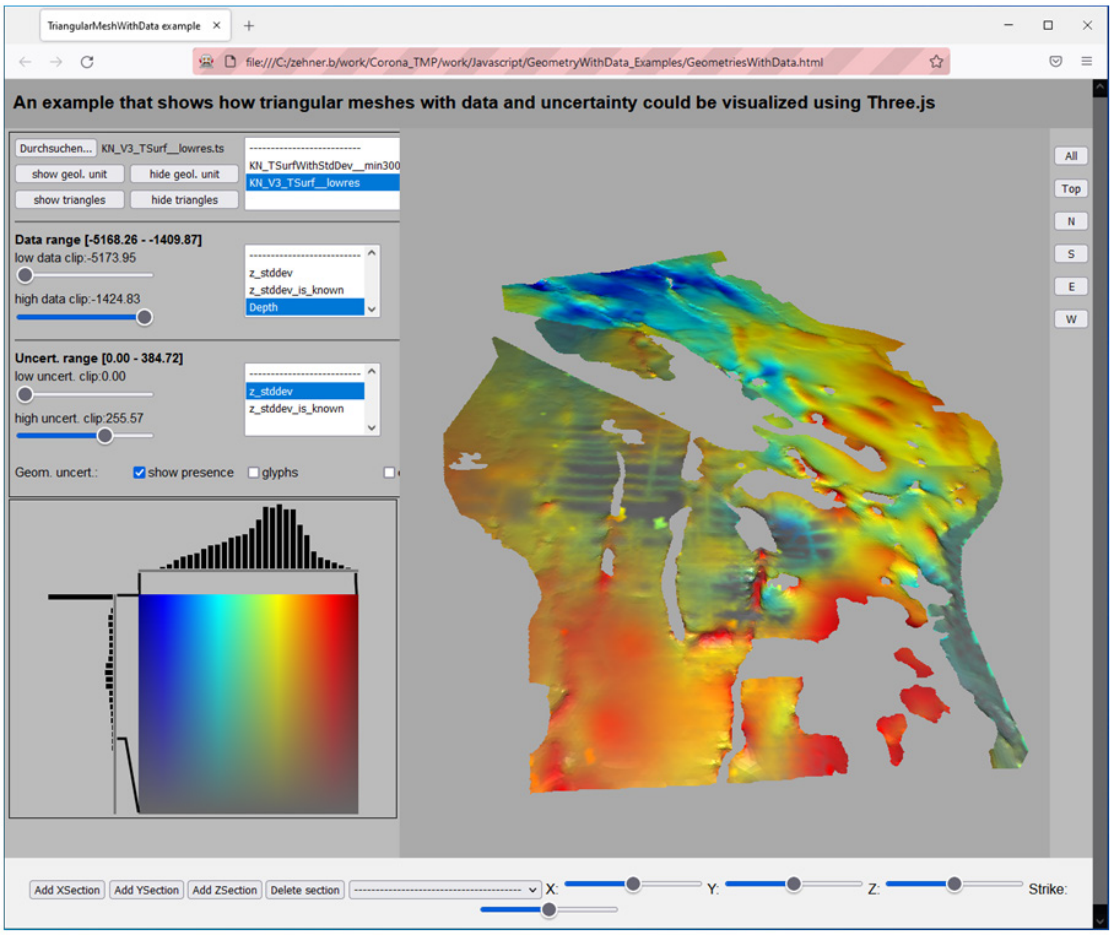

Figure 16: Visualization of the KN horizon using the Javascript prototype to display the horizon in a browser. Colour is mapped to elevation and its saturation is mapped to standard deviation.

5 REFERENCES

DE LA VARGA, M., SCHAAF, A., WELLMANN, F. (2019): GemPY 1.0: open source stochastic geological modelling and inversion, *Geoscientific Model Development* 12, Copernicus Publications / European Geosciences Union, pp. 1-32. DOI: 10.5194/gmd-12-1-2019

DOORNENBAL, H., DEN DULK, M., THÖLE, H., JÄHNE-KLINGBERG, F., BRITZE, P., JAKOBSEN, F. (2021): Deliverable 3.7 – A harmonized cross-border velocity model, Report of the GeoERA project 3DGEO-EU. Available: http://geoera.eu/projects/3DGEO-EU

FOLEY, J. D., VAN DAM, A., FEINER, S. K., HUGHES, J. F. (1996): Computer Graphics: Principles and Practice – Second Edition in C, Addison-Wesley, 1174p.

GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. (1995): Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 395p.

KOMBRINK, H., DOORNENBAL, J.C., DUIN, E.J.T., DEN DULK, M., VAN GESSEL, S.F., TEN VEEN, J.H., WITMANS, N. (2012): New insights into the geological structure of the Netherlands; results of a detailed mapping project, *Netherlands Journal of Geosciences 91(4)*, Cambridge University Press, DOI: https://doi.org/10.1017/S0016774600000329

PAKYUZ-CHARRIER, E., LINDSAY, M., OGARKO, V., GIRAUD, J., JESSELL, M. (2018): Monte Carlo simulation for uncertainty estimation on structural data in implicit 3-D geological modelling, a guide for disturbance distribution selection and parameterization, Solid Earth 9, pp. 385-402, DOI: https://doi.org/10.5194/se-9-385-2018

PYRCZ, M. J., DEUTSCH, C. V. (2014): Geostatistical Reservoir Modeling – Second Edition, *Oxford University Press*, 433 p.

ROST, R. J. (2006): OpenGL Shading Language. Addison Wesley, 740p.

SCHROEDER, W., MARTIN, K., LORENSEN, B. (1997): The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, Prentice Hall, 645p.

SCHWEIZER, D., BLUM, P., BUTSCHER, C. (2017): Uncertainty assessment in 3-D geological models of increasing complexity, *Solid Earth 8*, EGU, pp. 517-530, doi: 10.5194/se-8-515-2017

STAFLEU, J., MALJERS, D., GUNNINK, J.L., MENKOVIC, A., & BUSSCHERS, F.S. (2011): 3D modelling of the shallow subsurface of Zeeland, The Netherlands. *Netherlands Journal of Geosciences 90*, Cambridge University Press, pp. 293–310.

STAFLEU, J., DUBELAAR, C.W. (2016): Product Specification Subsurface model GeoTOP. TNO report 2016, R10133, TNO. Available from: https://www.dinoloket.nl/en/want-know-more

THÖLE, H., JÄHNE-KLINGBERG, F., DOORNENBAL, H., DEN DULK, M., BRITZE, P., JAKOBSEN, F. (2021): Deliverable 3.6 – Harmonized time model of the Entenschnabel region, Report of the GeoERA project 3DGEO-EU. Available: http://geoera.eu/projects/3DGEO-EU

THÖLE, H., JÄHNE-KLINGBERG, F., DOORNENBAL, H., DEN DULK, M., BRITZE, P., JAKOBSEN, F. (2021): Deliverable 3.8 – Harmonized depth models and structural framework of the NL-GER-DK North Sea, Report of the GeoERA project 3DGEO-EU. Available: http://geoera.eu/projects/3DGEO-EU

THORE, P., SHTUKA, A., LECOUR, M., AIT-ETTAJER, T., COGNOT, R. (2002): Structural uncertainties: Determination, management and applications, *Geophysics 67 (3)*, pp. 840-852.

WELLMANN, J. F., REGENAUER-LIEB, K. (2012): Uncertainties have a meaning: Information entropy as a quality measure for 3-D geological models, *Tectonophysics 526-529,* Elsevier, p.207-216.

WOLF, C.J.M., WARDT, J.P. (1981): Borehole Position Uncertainty - Analysis of Measuring Methods and Derivation of Systematic Error Model, *Journal of Petroleum Technology 33(12),* Society of Petroleum Engineers (SPE), pp.2339-2350. DOI: https://doi.org/10.2118/9223-PA.

ZEHNER, B., WATANABE, N., KOLDITZ, O. (2010): Visualization of gridded scalar data with uncertainty in geosciences, *Computers & Geosciences, Vol. 36,* Elsevier, p. 1268-1275. DOI: 10.1016/j.cageo.2010.02.010

ZEHNER, B. (2018): Constructing a volumetric model from a complex 3D structural pilot area in the German North Sea Sector, Proceedings of RING Meeting 2018 in Nancy, France. Can be downloaded from https://www.researchgate.net/profile/Bjoern-Zehner/research

ZEHNER, B. (2019): Deliverable 4.1 – State of the art in uncertainty visualization, Report of the GeoERA project 3DGEO-EU. Available from http://geoera.eu/projects/3DGEO-EU

ZEHNER, B. (2021): On the visualization of 3D geological models and their uncertainty, *ZDGG-Journal of Applied and Regional Geology), Vol. 172 (1),* Schweizerbart Science Publishers, p. 83-89. https://dx.doi.org/10.1127/zdgg/2020/0251 (Open Access).

ZEHNER, B., AYALA, C., BENSE, F., DABEKAUSSEN, W., DEN DULK, M., FRANEK, J., PUEYO, E. L., MALZ, A., MÜLLER, C. O., STÜCK, H. AND CONTRIBUTORS (2021): Deliverable 4.2 - Sources of uncertainty in 3D geological modeling, Report of the GeoERA project 3DGEO-EU. Available from http://geoera.eu/projects/3DGEO-EU