Chapter 13
Big Data Visualisation

Miguel Ángel Esbrí, Eva Klien, Karel Charvát, Christian Zinke-Wehlmann, Javier Hitado, and Caj Södergård

Abstract In this chapter, we introduce the topic of big data visualization with a focus on the challenges related to geospatial data. We present several efficient techniques to address these challenges. We then provide examples from the DataBio project of visualisation solutions. These examples show that there are many technologies and software components available for big data visualisation, but they also point to limitations and the need for further research and development.

13.1 Advanced Big Data Visualisation

Data visualisation is the graphical representation of information and data. By using visual elements like charts, graphs and maps, data visualisation tools provide an accessible way to see and understand trends, outliers and patterns in data [1]. More

M. Á. Esbrí (✉) · J. Hitado
Atos Spain, Albarracin 25, 28004 Madrid, Spain
e-mail: miguel.esbri@atos.net

J. Hitado
e-mail: javier.hitadosimarro@atos.net

E. Klien
Fraunhofer-Institute for Computer Graphics Research IGD, Fraunhoferstr. 5, Darmstadt, Germany
e-mail: eva.klien@igd.fraunhofer.de

K. Charvát
LESPROJEKT-SLUŽBY Ltd, Martinov 197, 27713 Zaryby, Czech Republic
e-mail: charvat@lesprojekt.cz

C. Zinke-Wehlmann
Institut for Applied Informatics e.V. at the University of Leipzig, Goerdelering 9, Leipzig, Germany
e-mail: zinke@infai.org

C. Södergård
VTT Technical Research Centre of Finland, Espoo, Finland
e-mail: Caj.Sodergard@vtt.fi

C. Södergård et al. (eds.), *Big Data in Bioeconomy*,
https://doi.org/10.1007/978-3-030-71069-9_13

particularly, the defining feature of big data visualisation is scale, in terms of the vast amounts of data to be dealt with.

In that sense, the amount of data created by the private and public sectors around the world is growing every year, skyrocketing with the emergence and popularisation of the Internet of Things and the many open data initiatives that have made available a wealth of datasets (typically owned by the public sector) to the public. The Copernicus programme and the data provided by its Sentinel satellite constellation are a paradigmatic example of this (see Chap. 4).

The underlying problem for decision-makers is that all this data is only useful if valuable insights can be extracted (sometimes in near real-time) from it, and decisions can be made based on them. Big data visualisation is not the only way for decision-makers to analyse data, but big data visualisation techniques offer a fast and effective way to [2]:

- Review large amounts of data—Data presented in graphical form enables decision-makers to take in large amounts of data and gain an understanding of what it means very quickly.
- Spot trends—Time-sequence data often captures trends, but spotting trends hidden in data is notoriously hard to do—especially when the sources are diverse, and the quantity of data is large. The use of appropriate big data visualisation techniques can make it easy to spot these trends and take decisions.
- Identify correlations and unexpected relationships—One of the huge strengths of big data visualisation is that it enables users to explore datasets—not to find answers to specific questions, but to discover what unexpected insights the data can reveal. This can be done by adding or removing datasets, changing scales, removing outliers and changing visualisation types.
- Present the data to others—An often-overlooked feature of big data visualisation is that it provides a highly effective way to communicate any insights that it surfaces to others by conveying meaning very quickly and in an easily understandable way.

Besides, an important aspect of big data visualisation is choosing the most effective way to visualise the data to surface any insights it may contain. In some circumstances, simple graphic tools such as pie charts or histograms may be enough, but with large, numerous and diverse datasets more advanced visualisation techniques may be more appropriate. Various big data visualisation graphics examples include:

- Linear: Lists of items, items sorted by a single feature, text tables, highlight table
- 2D/Planar/geospatial: Cartograms, dot distribution maps, proportional symbol maps, contour maps.
- 3D/Volumetric: 3D computer models, computer simulations.
- Temporal: Timelines, time series charts, connected scatter plots, arc diagrams, circumplex charts.
- Multidimensional: Pie charts, histograms, histogram, matrix, tag clouds, bar charts, tree maps, heat maps, spider charts, area chart, Box-and-whisker Plots, bubble cloud, bullet graph, circle view, Gantt chart, network, polar area, scatter plot (2D or 3D), streamgraph, wedge stack graph.

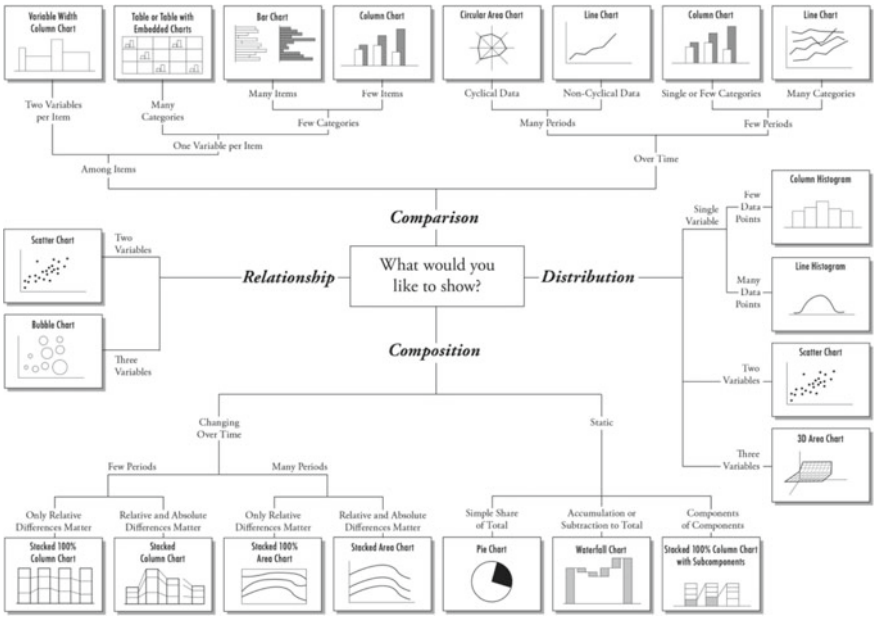

Fig. 13.1 Chart selector guide [3]

- Tree/hierarchical: Dendrograms, radial tree charts, hyperbolic tree charts.
- Any mix-and-match combination in a dashboard.

The following chart selection guide (Fig. 13.1) summarises the selection of the most appropriate chart types depending on what it is intended to be shown:

The variations in the visualisation of geoinformation (GI) are more limited because it is fundamentally linked to spatial context and geographical maps. The first priority of GI visualisation tends to be more geographical than to be informational or graphical. Maps allow us to communicate spatial information effectively. Big data visualisation opens the possibilities of GI visualisation in terms of spatial extent, spatial resolution and density of content. New techniques help mastering the vast amount of information, thus strengthening the spatial context and facilitating the exploration of new meanings and insights through map and other kinds of representations.

13.2 Techniques for Visualising Very Large Amounts of Geospatial Data

Different visualisation charts were presented in the previous section, the selection of which is dependent on the type of information and the goals of the target audience. However, in many occasions, the resulting visualisation requires the use of different

techniques that allow simplifying, aggregating and reducing in various orders of magnitude the information that is finally used in the graphic charts and maps.

The following section presents three different and complementary approaches to deal with the visualisation of large amounts of geospatial data.

13.2.1 Map Generalisation

Cartographic generalisation, or map generalisation, includes all changes in a map that are made when one derives a smaller-scale map from a larger-scale map or map data or vice-versa [4]. Generalisation seeks to abstract spatial information at a high level of detail to information that can be rendered on a map at a lower level of detail. This is of high importance when dealing with massive amounts of data, as it would be prohibitively—in terms of computation, data transfer and user experience (i.e. real-time interactivity)—to try to render the several gigabytes of data "as it is".

In that sense, suitable and useful maps typically have the right balance between the map's purpose and the precise detail of the subject being mapped. Well-generalised maps are those that emphasise the most important map elements while still representing the world in the most faithful and recognisable way [5].

There are many cartographic techniques that may fall into the broad category of generalisation [4, 6]. Among the most commonly used methods, we can find:

- **Simplification**—allowing to reduce the complexity of the geometries (i.e. lines and polygons) by eliminating or merging some of their vertices
- **Aggregation**—allowing to combine or merge some of the geometries (e.g. using the distance between polygons or by common attribute values) and thus resulting in a more reduced set of geometries.
- **Selection/Elimination**—allowing to reduce the number of features in the map by filtering or retaining them according to certain criteria (e.g. attribute values, spatial relations such as overlaps and distance between them).
- **Typification**—This method can be seen as an extreme case of simplification, where a detailed geometry is replaced by a simpler one to represent the feature in the map (e.g. a polygon defining the boundaries of a city is replaced by a point).
- **Exaggeration**—allowing to visually make more prominent some aspects we are interested in presenting in the map (e.g. represent cities with larger or smaller point sizes depending on the number of inhabitants).
- **Classification**—allowing to group into the same category and present in the map features with similar values.
- **Resampling**—which allows to reduce the amount of information provided in a map by changing its spatial resolution (e.g. changing the resolution of a raster dataset where the original pixel size is resampled from 100 m^2 to 1 km^2). This can be seen as a particular case of the aggregation method, involving interpolation techniques for determining the pixel values of the new resulting raster.

13.2.2 Rendered Images Versus the "Real" Data

In general, the process of rendering geospatially enabled information into maps is quite costly. Usually, the information, either raster- or vector-based, is stored in files or databases, which must be searched, queried, filtered and then transformed into a georeferenced map that can be integrated in a desktop or web client. This process can take longer the more information we have in our repositories, which can be very inefficient when several concurrent users make requests to the web mapping service asking for different areas or zoom levels. This can lead to unresponsive services due to the large workload imposed to the server.

In order to alleviate this issue, web mapping services offer the possibility to send the maps in the form of a tiled map, which is displayed in the client by joining dozens of individually requested image or vector data files (tiles) over the Internet. The advantage of this approach is that instead of loading the whole map instantly, for each zoom level, the web mapping service divides the imagery into a set of map tiles, which are logically put in an order which the application can understand. When the user scrolls the map to a new location, or to a new zoom level or location, the service decides which tiles are necessary and translates those values into a set of tiles to retrieve.

Concerning the tiling formats, there are two possibilities, each of them with their advantages and drawbacks:

Raster tiles are used to divide raster data into small, manageable areas that are stored as individual files in the filesystem (or BLOBs in a database). The tile-based approach is fundamental for efficient and improved performance for data loading, querying, visualisation and transfer of information over the networks. Thus, for instance, if a user zooms in a map into a small two tile area in a single band image, the underlying management service (e.g. OGC WMS) needs to fetch only two raster tile files from the filesystem instead of the entire raster dataset in order to compose the final image sent to the client.

Raster tiles of 256 × 256 pixel images are a de facto standard; however, 512 × 512 pixel seems to be the usual size of high-resolution tiles. Other sizes are possible depending on the purpose (e.g. 64 × 64 pixel images for mobile use), and in fact, a common approach is to generate a pyramid of different tile sizes that are used depending on the zoom level requested on the client side (Fig. 13.2).

Vector tiles are similar to raster tiles, but instead of raster images, the data returned is a vector representation of the features in the tile [7].

At the client side, it is possible to mix raster tiles with vector tiles and make the best usage of both, e.g. satellite map (raster tiles) with an overlay of streets with labels available in many languages (vector tiles) (Table 13.1).

As it can be seen, it could be possible to mix raster tiles with vector tiles and make the best usage of both, e.g. satellite map (raster tiles) with an overlay of different cartography and thematic layers (vector tiles).

[1] https://wiki.osgeo.org/wiki/Vistsos.

13.2.3 Use of Graphics Processing Units (GPUs)

Large-scale visualisation is an ideal application for graphics processing unit (GPU) computing for several reasons [10]:

- Visualisation is a data-intensive application, particularly as the problem size increases into the petascale. GPUs are well suited for data-intensive tasks.
- Visualisation computations exhibit substantial parallelism, typically both object parallelism (many objects or parts of objects can be computed/viewed in parallel) and image parallelism (visualisations produce large images, and image parts can be computed/viewed in parallel). Parallel computations are necessary for GPUs to be effective.
- Visualisation tasks should be closely coupled to the graphics system; even though much of overall visualisation computation may not be graphics centric, the final stage typically is, and so moving computation closer to the graphics device offers potential benefits in terms of interactivity and computational steering.
- GPUs can offload computation from CPUs, permitting the entire application to run faster when GPUs are involved.

More particularly, the many different functions used to manipulate geospatial data create additional processing workloads ideally fitted to GPU-accelerated solutions. Examples of these functions include:

- Filtering by area, attribute, series, geometry, etc.
- Aggregation, potentially in histograms.
- Geo-fencing based on triggers.
- Generating videos of events.
- Creation of heat maps.

Nowadays, there are big data solutions and frameworks leveraging in GPU capabilities for improving the data processing and rendering (both at server and client side), among others:

- Server side:

Fig. 13.2 Pyramid tile structure[1]

Table 13.1 Comparison raster and vector tiles use (pros/contras) [8, 9]

	Raster tiles	Vector tiles
Pros	• Tiles are generally rendered in advance on the server and streamed to the destination • Detailed tiles can be generated and served • More suitable for the display of imagery and shaded terrain • Lower requirements for end users hardware • Still a bit better support in web JavaScript libraries and desktop GIS software	• Tiles are rendered quickly and are only 20–50 per cent the file size of raster tiles • More tiles can be produced per second • Less bandwidth is needed due to the smaller size of tile packages—making vector tiles a better choice when streaming to devices • Map styles (colour, grey, night mode, etc.) can be changed without needing to download more information or other tile sets • Dynamic labelling allows size and font types to be changed on the fly • Better user experience —smooth zooming • No need for zoom levels—- users zoom and pan throughout all scales • De facto mobile standard
Contras	• Each map style must be created in a separate raster tile set • Labelling is preset and cannot be changed • A bigger size of each tile and data on servers • Takes more time to generate—can be CPU and memory consuming • Not the greatest for real-time rendering. Slower loading disrupts the user experience when moving around the map	• Rendering occurs on the client side, where limited resources can hamper speed • Compromises clarity by reducing display detail • Requires OpenGL/WebGL/DirectX support, which is an issue for some mobile devices • Not suitable for imagery or other raster maps • Vectors are generalised (i.e. not raw data) so they may not be suitable for editing

  - Rasdaman array database (only available in the enterprise version)[2]
  - OmniSci database (formerly MapD)
  - AresDB
  - Apache Spark
  - PostgreSQL and PG-Strom extension[3]
- Client side:
  - Cesium
  - MapD-charting
  - Kepler.gl.

[2] https://rasdaman.com/commercial-free.php.

[3] https://heterodb.github.io/pg-strom/.

Another example of visualisation leveraging on graphical cards is exploratory visualisation. Exploratory visualisation is the process that involves an expert creating maps and other graphics while dealing with relatively unknown geographic data. Generally, these maps serve a single purpose and function as an expedient in the expert's attempt to solve a particular (geo) problem. While working with the data, the expert should be able to rely on cartographic expertise to be able to view data from different perspectives. As such, the resulting maps and graphics are available in an interactive viewing environment that stimulates visual thinking and promotes informed decision-making. WebGLayer[4] is a JavaScript library focused on fast interactive visualisation of big multidimensional spatial data through linked views. The library is based on WebGL and uses GPU for fast rendering and filtering. Using commodity hardware, the library can visualise hundreds of thousands of features with several attributes through heatmap or point symbol map. The library can render data on the map provided by third party libraries (e.g. OpenLayers, Leaflet, GoogleMap API). Figure 13.3 shows an example for the analysis of yield potential [11].

13.3 Examples from DataBio Project

13.3.1 Linked Data Visualisation

Linked data visualisation is about providing graphical representations of interesting aspects within the Semantic web. The high variety of linked data and its types is huge. An example of agriculture linked open data is the FOODIE data model, which

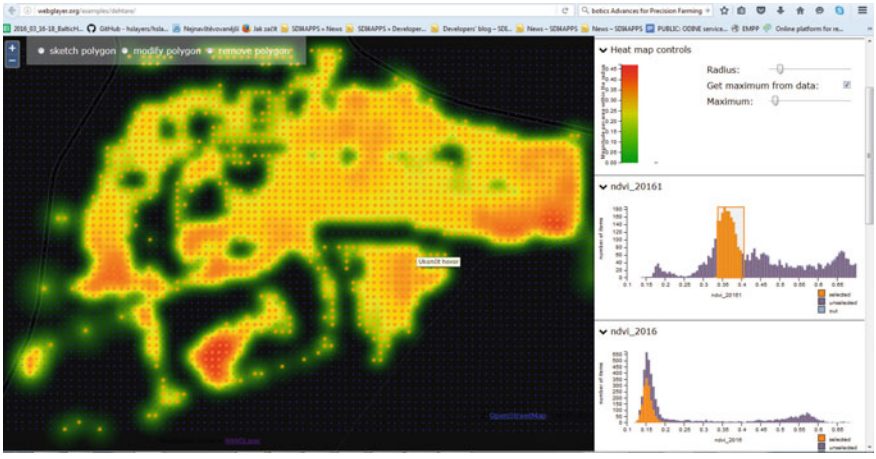

Fig. 13.3 WebGLayer showing yield potential

[4] https://webglayer.org.

was originally developed as part of the FOODIE project and later extended in the DataBio project. The FOODIE data model is based on the generic data models of INSPIRE, especially the data models for agricultural and aquaculture facilities and Land-Parcel information system. The key motivation was to represent a continuous area of agricultural land with one type of crop species, cultivated by one user in one farming mode (conventional vs. transitional vs. organic farming). Additionally, the FOODIE data model includes concepts for crop and soil data, treatments, interventions, agriculture machinery and others. Finally, the model reuses data types defined in ISO standards (ISO 19101, ISO/TS 19103, ISO 8601 and ISO 19115) as well standardisation efforts published under the INSPIRE directive (like structure of unique identifiers). The FOODIE data model was specified in UML (as the INSPIRE models) but can be transformed into an OWL ontology in order to enable the publication of linked data compliant with FOODIE data model [12].

As mentioned in Chap. 8 "Linked Data Usages in DataBio" the triplestore with linked data has over 1 billion triples—which is organised into named graphs (IRI) and sub-graphs. For example, the LPIS-Poland dataset (Land-Parcel identification in Poland) can be identified by the graph <https://w3id.org/foodie/open/pl/LPIS/> and contains 727,517,039 triples with a subgraph <https://w3id.org/foodie/open/pl/LPIS/lubelskie#> , referring to the data with the Lublin Voivodeship. Thus, querying and pre-processing, including link discovery, are very important for an efficient way to visualise linked data. Depending on the size of linked datasets (amount, distributed etc.) and the linkages between the data, there are different ways to visualise them. In DataBio, metaphactory, a linked data exploitation platform, has been used to query, browse and navigate linked data—for example, the catch records data from Norway (see Fig. 13.4).

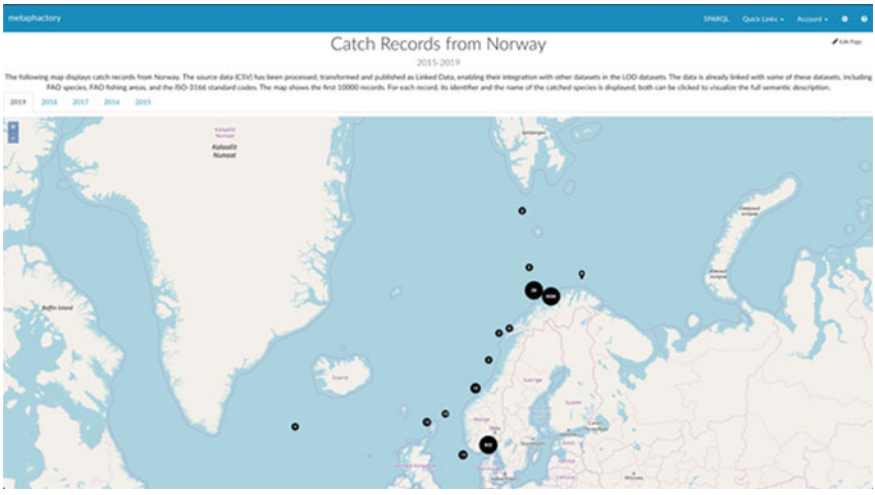

Fig. 13.4 DataBio metaphactory custom view (map with catch records from Norway)

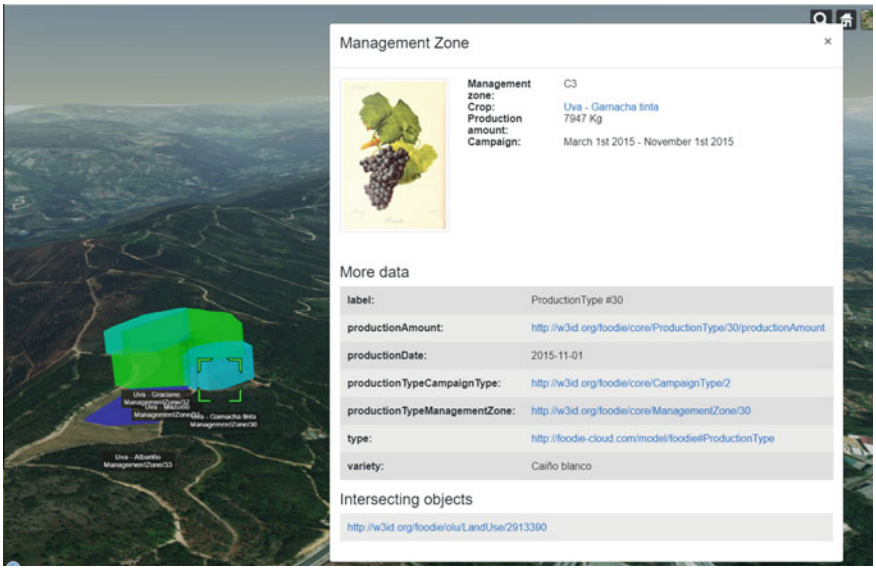

Fig. 13.5 Screenshot of the application showing result of use case crops types based on linked data

The second way to visualise is to query the SPARQL endpoint(s) (using GeoSPARQL[5]) and get RDF or JSON-LD.[6] There is also the possibility to discover more data (types and links) and put them together. Finally, the results can be transformed into the form of JSON resp. GeoJSON, which are easily processed by most visualisation clients. Leading technology providers are aware of this need and plan to develop some features to do so automatically. Figure 13.5 shows an example for visualising different crop types based on information from linked data.[7]

13.3.2 Complex Integrated Data Visualisation

Complex integrated data visualisation was an important part of the Czech agriculture pilots, and the technology was also tested for fishery pilots. The technology used was HSlayers NG. Hlayers NG (https://ng.hslayers.org/) is a web mapping library written in JavaScript. It extends OpenLayers 4 functionality and takes basic ideas from the previous HSlayers library but uses modern JS frameworks instead of ExtSJS 3 at the

[5] https://www.opengeospatial.org/standards/geosparql.

[6] An extension of JSON for Linked Data is JSON-LD (JavaScript Object Notation for Linked Data), which is a method of encoding Linked Data using JSON. This allows data to be serialised in a way that is like traditional JSON. JSON-LD is designed around the concept of a "context" to provide additional mappings from JSON to an RDF model.

[7] Further examples for integrated data visualisation on maps from DataBio can be explored under the following link: https://app.hslayers.org/project-databio/land/.

frontend and provides better adaptability. That is why the NG ("Next Generation") is added to its name. It is still under development and provided as open-source software. HSLayers is built in a modular way which enables the modules to be freely attached and removed as far as the dependencies for each of them are satisfied. The dependency checking is done automatically. The core of the framework is developed using AngularJS, requireJS and Bootstrap. This combination of frameworks was chosen mainly for providing fast and scalable development and for providing a modern responsive layout for the application. Figure 13.6 gives an example for a complex integrated data visualisation.

The most important modules are:

- The map functionality is provided by OpenLayers4 and extended by some controls.
-

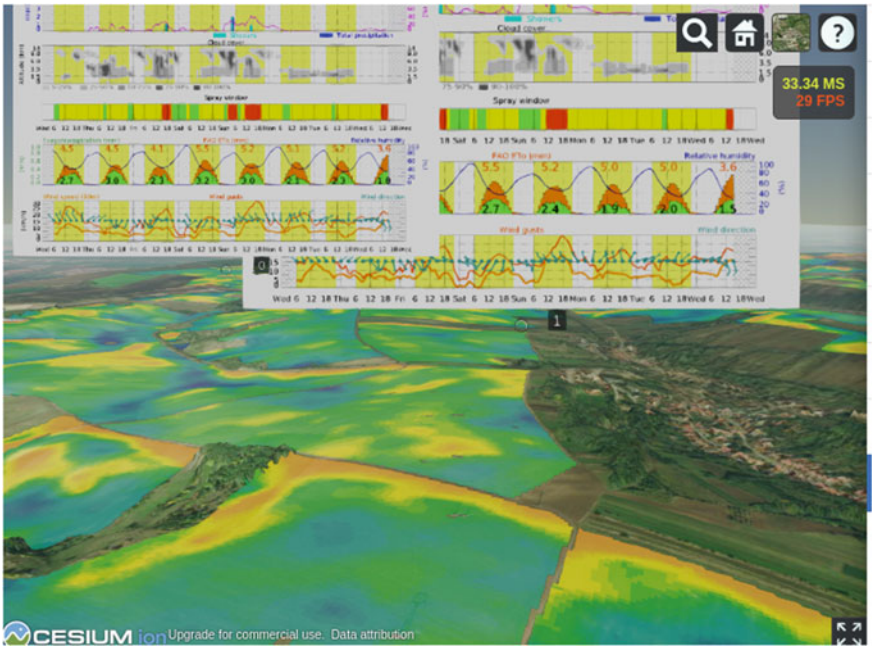

Fig. 13.6 Integration of yield potential data (3D maps) with meteorological data (time series) [11–13]

Layer manager is used for listing all the map layers, displaying or hiding them and setting the transparency.
- OGC web services parser is used for GetCapabilities requests to different map servers and parsing the response.
- Linked Open Data explorer: Eurostat explorer is a demo application (module) which queries Semantic web data sources via SPARQL endpoints.
- HSlayers visualises geographical data in a 3D environment.
- Support for visualisation of sensors and agrometeorological data for farmers can help with forecast of weather and better planning of operations.

13.3.3 Web-Based Visualisation of Big Geospatial Vector Data

Chapter 15 introduces various pilots on smart farming for sustainable agricultural production in Greece. In these applications, information about growing crops in millions of parcels spread over the country needs to be visualised. The information about the growing plants, trees and grain types is updated periodically, which makes the data dynamic. Providing a map interface that supports end users to explore this amount of dense data using a vector-based approach is a big challenge to the implementation.

In order to address this challenge, an approach to visualise huge sets of geospatial data in modern web browsers along with maintaining a dynamic tile tree was developed in the DataBio project and successfully applied to the pilot application [14]. The approach makes it possible to render over one million polygons integrated in a modern web application by using 2D vector tiles (see Sect. 13.2.2). Figure 13.7 shows an example for an in-depth parcel assessment with vegetation index colour coding for Greece.

This novel approach to build and maintain the tile tree database provides an interface to import new data and a more flexible and responsive way to request vector tiles. There are three essential steps involved [14]:

1. Data storage is re-organised in a way to have efficient access to geospatial vector tiles. This is achieved by using a geospatial index along with the fast and scalable distributed file system GeoRocket.[8] GeoRocket uses MongoDB to persist data and Elasticsearch to build a spatial index for data query and aggregation tasks. GeoJSON can be imported directly without conversion.
2. Secondly, it is essential to speed up the vector tile creation process, which is important for both, the initial creation of the tile tree and serving tiles. For this, a new tiling algorithm was implemented. The tiling implementation is a server component itself and provides a REST interface. It can be configured using different file storage backend technologies for persisting the tiles. The configuration includes a range of zoom levels in which the tiles are created,

[8] GeoRocket—https://georocket.io.

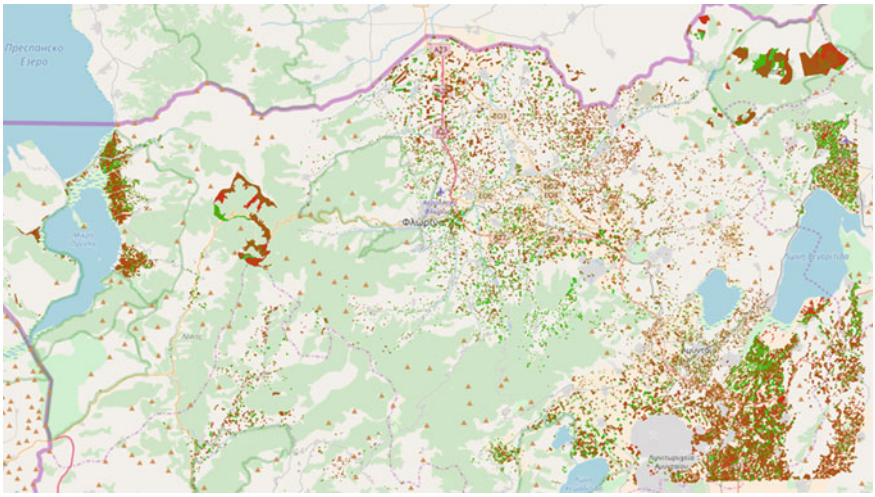

Fig. 13.7 In-depth parcel assessment with vegetation index colour coding

which is 2 to 15 by default. These are enough for most users' map interface experience, but for a more detailed view, it is also possible to build tiles on higher zoom levels.

3. Finally, data must be transmitted to a web application running in modern web browsers. The geometries are rendered using a WebGL map application framework. It is possible to add interaction concepts such as filters and user-defined styling. The most common and stable frameworks are OpenLayers and MapboxGL JS. The young vector tile implementation in OpenLayers has many issues, most critical a memory leak, no data-driven styling and no WebGL support for vector tiles. Therefore, Mapbox GL JS was used in the pilot application and evaluation.

13.3.4 Visualisation of Historical Earth Observation

Earth observation measurements provided by satellites from the Sentinel and Landsat programmes are one of the largest sources of big geospatial data, which are not only challenging in terms of data storage and access management (as presented in Chap. 4 Remote Sensing) but also for filtering, processing and visualising due to the large size of the files. Figure 13.8 shows an example from the DataBio fisheries pilot, where a web client is used for 3D visualisation of oceanic historical datasets, such as ocean salinity, temperature, concentration of chlorophyll.), in the whole Indian Ocean region where the fishery vessels operate.

The satellite imagery time series is served through the Rasdaman service via the OGC WMS-T and WMST interfaces and integrated with the HSLayers and Cesium JS library, which allow to display geospatial data available in various raster and vector

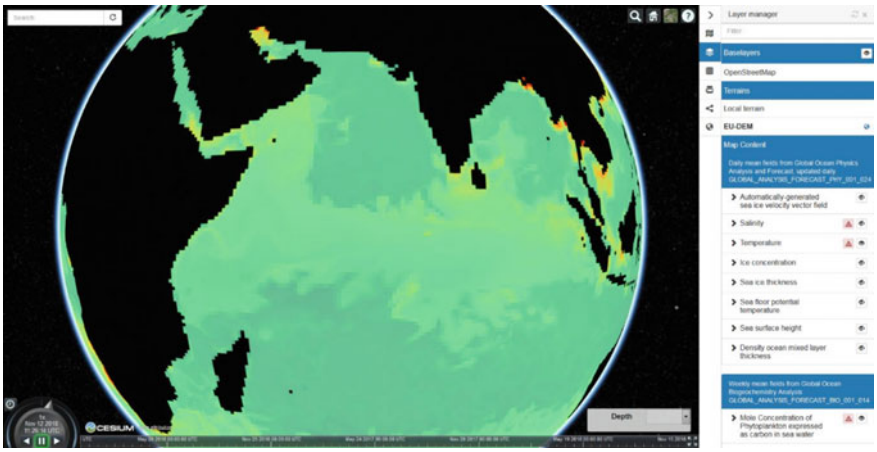

Fig. 13.8 3D web visualisation of historical oceanic measurements using HS layers and Rasdaman

formats. The web client component allows to control the visualisation by additional (non-spatial) dimensions of the data. In this specific case, the web client enables the user of the application to choose the time and depth level parameters, which are then used to query the Rasdaman service, returning the rendered map in the form of a series of raster tiled images.

13.3.5 Dashboard for Machinery Maintenance

Visualisation is important when informing the user about the status of technical processes, e.g. in machine maintenance. Especially, it is central to show alerts about critical events, like too high temperature, pressure and so on. Use of colours and visual effects, like blinking, must be considered with great care. In Fig. 13.9 is an example from DataBio, where we designed a visual dashboard for showing information about the status of the engines of fishing vessels.

Fig. 13.9 Visual dashboard from a DataBio pilot on fishery. The dashboard shows information and alerts about the status of the fishing vessel's engines

References

1. https://www.tableau.com/learn/articles/data-visualization (Accessed September 2019).
2. Rubens, P. (2017). *Big data visualization.* Available at: https://www.datamation.com/big-data/big-data-visualization.html (Accessed September 2019).
3. Abela, A. V. (2013). *Advanced presentations by design—creating communication that drives action* (2nd ed.). Wiley.
4. Li, Z. (2007). Digital map generalization at the age of the enlightenment: A review of the first forty years. *The Cartographic Journal, 44*(1), 80–93. https://doi.org/10.1179/000870407x173913.
5. 'Cartographic generalization'. (2020). *Wikipedia.* Available at: .https://en.wikipedia.org/wiki/Cartographic_generalization (Accessed: 24 September 2020).
6. Raposo, P. (2017). Scale and Generalization. In J. P. Wilson (Ed.), *The geographic information science & technology body of knowledge* (4th Quarter 2017 Edn.), https://doi.org/10.22224/gistbok/2017.4.3. Available at: https://gistbok.ucgis.org/bok-topics/scale-and-generalization-1
7. 'Vector Tiles'. (2020). Openstreetmap Wiki. Available at: https://wiki.openstreetmap.org/wiki/Vector_tiles (Accessed: 24 September 2020).
8. 'Vector tiles vs raster tiles—the pros and cons'. (2017). From TechBlog of MapData Services. Available at: https://mapdataservices.wordpress.com/2017/02/22/vector-tiles-vs-raster-tiles-the-pros-and-cons/ (Accessed: 24 September 2020).
9. Janak, D. (2019). 'What are vector tiles and why you should care'. Available at: https://www.maptiler.com/blog/2019/02/what-are-vector-tiles-and-why-you-should-care.html
10. Owens, J. D. (2007). Towards multi-GPU support for visualisation. *Journal of Physics Conference Series (Online), 78*(1), 012055. https://doi.org/10.1088/1742-6596/78/1/012055.
11. Řezník, Tomáš et al. (2016). MONITORING OF IN-FIELD VARIABILITY FOR SITE SPECIFIC CROP MANAGEMENT THROUGH OPEN GEOSPATIAL INFORMATION. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XLI-B8. 1023–1028. https://doi.org/10.5194/isprsarchives-XLI-B8-1023-2016.
12. Charvat, K., Junior, K. C., Reznik, T., Lukas, V., Jedlicka, K., Palma, R., & Berzins, R. (2018, July). Advanced visualisation of big data for agriculture as part of databio development. In

IGARSS 2018–2018 IEEE International Geoscience and Remote Sensing Symposium (pp. 415–418). IEEE.
13. Jedlička, K., & Charvát, K. (2018, May). Visualisation of Big Data in Agriculture and Rural Development. In *2018 IST-Africa Week Conference (IST-Africa)* (pp. Page-1). IEEE.
14. Zouhar F., Senner I. (2020) Web-Based visualisation of Big Geospatial Vector Data. In: Kyriakidis P., Hadjimitsis D., Skarlatos D., Mansourian A. (eds) Geospatial Technologies for Local and Regional Development. AGILE 2019. Lecture Notes in Geoinformation and Cartography. Springer, Cham. https://doi.org/10.1007/978-3-030-14745-7