International Journal of Geo-Information

MDPI

Article

A PDF Tile Model for Geographic Map Data

Xiaodong Zhou [1,2,3,*], Tinghua Ai [1], Nina Meng [4] and Peng Xie [2,3]

1 School of Resource and Environmental Science, Wuhan University, 129 Luoyu Road, Wuhan 430079, China
2 Xi'an Research Institute of Surveying and Mapping, Xi'an 710054, China
3 State Key Laboratory of Geo-Information Engineering, Xi'an 710054, China
4 College of Geology Engineering and Geomatics, Chang'an University, Xi'an 710054, China
* Correspondence: zhouxiaodong@whu.edu.cn; Tel.: +86-029-8474-9138

Received: 5 August 2019; Accepted: 24 August 2019; Published: 27 August 2019

Abstract: Vector tile mapping is an important issue in web map research. At present, vector tile mapping requires the symbolization of geographic information, as supported by cartographic software, and the development of a corresponding symbolic database when web map services are provided for users. The development of PDF (portable document format) mapping makes it possible to use symbolized map data directly for web map services. This paper presents a PDF tile map model suitable for web map services, in order to provide a new solution for vector tile mapping. This paper details the construction and establishment method of the PDF tile map model and verifies that the model has the characteristics of a vector tile map and that this approach can provide web map services. In addition, this model has some other noticeable characteristics. First, the map service supported by this model does not require the support of cartographic software, and thus has low system software and hardware requirements. Second, the maps displayed in the service system can be directly used for high-quality map publishing. Third, the high popularity of the PDF format can promote the sharing of PDF tile maps and reduce the use threshold of ordinary users.

Keywords: vector tiles; PDF tiles; map tiles; web map; map visualization; clipping; splicing

1. Introduction

As a special form of language, maps are not only a tool for humans to recognize the world but also display the results of changes in the world [1–3]. The internet enhances map usability and combines map printing and distribution. Web mapping can more rapidly and directly provide the user with map services [4–7]. All of the present mainstream web maps—such as Google Maps [8], Apple Maps, Bing Maps, and OpenStreetMap (OSM) [9]—provide map services by publishing images on the client side in the form of raster tiles. Raster tiles take image blocks as visualized representations of map data. As a result, amplification of raster tiles will result in fuzziness of map images, which cannot be clearly displayed. In practical applications, raster tiles of different resolutions can be obtained by rendering map data on different scales in order to continuously present maps of more elaborate resolutions. In consideration of the deficiencies of raster tiles, such as large total data size and lack of interaction ability with geographic elements, vector data-based web maps have gradually become a popular research subject, and vector tiles have consequently appeared [10,11]. MapBox promoted MapBox Streets in 2012, which provided vector tiles of the world and formed standards for MapBox vector tiles [12]. Google started using vector tiles in Google Streets in 2010, and Apple Maps began in 2012. Both gained good results but used exclusive formats which could not be used by the general public.

Vector tile maps are currently generated based on geographic information. They use symbol mapping mechanisms supported by specific symbol libraries. Different types of cartographic software usually use different symbol libraries, different mapping mechanisms, and different map

browsers [13,14]. Thus, web map services using vector tiles are run under their respective system platform, such as Apple Maps and Google Maps. Cartographic models determined that the mainstream use of vector tile maps is related to system software. Is it possible to use published digital symbolic maps as vector tiles if we do not need to consider the cartographic procedure? The development of PDF (portable document format) map technology makes this idea possible.

PDF mapping is a new type of map product that stores geospatial and multimedia information, such as graphics, images, and text, in PDF format. The main characteristics of PDF mapping are summarized as follows. First, PDF maps do not need to be associated with a map symbolic database, or the support of a complicated GIS (geographic information system) or cartographic system. There is no particular system requirement for PDF maps, thereby lowering the threshold for users. The users of PDF maps can use a PDF reader, such as Adobe Acrobat, Adobe Reader, and Foxit Reader, to read layered maps and image products that have a geographic reference. Second, compared to traditional vector maps, PDF maps can reduce data volume by about 80%, which provides a guarantee for its use on portable terminals. Third, PDF maps have the same display effect on any display screen. Furthermore, they can print as a clear and high-quality paper map product on a local printer. Thus, users can obtain a paper map that is the same as that displayed on the screen [15].

In this paper, we adopt the PDF tiling method to realize logically the seamless organization of PDF maps up to the global area. Our PDF tile model is a new vector map tile between raster tiles and traditional vector tiles, which inherits the advantages of the PDF format. We take map publishing data as the data source and PDF map technology as the technical support, with the goal of establishing a new PDF vector tile map model that can be used for web map services. This model not only has the existing ability of the vector tiles, but also other advantages. First, PDF tile mapping provides a map browse function without the need for symbolization of geographic information, and thus needs no specific symbolic database. Second, since PDF tile mapping uses the final publishing-oriented map results, it can be directly used for paper map publishing. Finally, because of the high universality of the PDF format, PDF tile maps are easier to share and can be more easily used by the general public.

The remainder of this paper is structured as follows. Section 2 briefly reviews previous related work. Section 3 details the construction and establishment method of the PDF tile model. Section 4 carefully designs experiments and discusses the experimental results, in which an offline map and online map are carried out to validate the performance of our PDF tile model and compare our PDF tile map with the current raster tiles and vector tiles. This paper ends with conclusions (Section 5).

2. Related Work

The development of the internet has hastened the demand for the use of maps through browsers. The WMS (web map service) solution emerged in 1999, which greatly promoted the progress of web maps [16]. The WMS renders a map into a large image on the server side according to browser window size, transmits the map image to the browser side through the network, and displays the map image directly using the browser. For WMS, back-end rendering and network transport are challenges and inefficiencies. Hence, WMS-C (cached) improves efficiency by caching map tiles. Google Maps appeared online in 2005, allowing global users to easily enjoy online maps through high-efficiency tile map technology, which generates raster tiles from the image. With the emergence of AJAX (asynchronous JavaScript and XML (extensible markup language)) and drawing APIs (application program interface), the application of web maps based on raster tiles has been extensively popularized, and most commercial and open-source public map services adopt a tile-based data transmission method [17]. However, with in-depth application and mobile information of web maps, raster tiles have encountered at least two problems: first, images consume large bandwidth and storage, which is adverse to map applications on mobile devices; and second, users cannot interact with images. Therefore, a solution which uses vector tiles to replace raster tiles on the mobile side appeared. Similar to raster tiles in concept, vector tiles store the spatial position and other attribute data of geographic elements within the scope of the tiles. They constitute a vector representation of geographic elements

rather than pre-generated raster images. With small volumes, vector tiles can be highly compressed, which not only reduces network bandwidth consumption but also makes offline mapping possible. Hence, vector tile maps have been praised as the future for the application of web maps and vector tiles have gradually become a very important technology for web maps on the browser side.

The URL (uniform resource locator) solution of vector tiles has always abided by raster tile patterns, such as those in Google XYZ [18], Microsoft Quad Tree [19] and open-source TMS (tile map service) [20]. The vector tile encoding format determines the data size of vector tiles so as to directly influence transmission efficiency of vector tiles on the web. The traditional vector tile encoding format mainly concentrates on XML and GeoJSON (a geospatial data interchange format based on JavaScript object notation (JSON)) [21]. As a text encoding format, GeoJSON has been extensively accepted in the field of geographic mapping. At present, most vector tile servers support GeoJSON-encoded vector tiles. TopoJSON (an extension of GeoJSON that encodes topology) [22], which can be regarded as an extension of GeoJSON, is actually a delta-encoded version of GeoJSON that eliminates geometric coordinate redundancy. It transmits data differences between sequential data rather than complete files, which reduces data size of vector tiles. Compared with XML and JSON [23], compact binary encoding formats, such as Google Protocol Buffers [24], can greatly reduce data size and improve data transmission efficiency.

The various solutions applied to vector tiles are seen in Table 1 including (i) the open-source programs TileStache, Mapzen, OpenLayers, Kartotherian, Tegola, and (ii) the commercial programs MapBox, Google, and Apple.

Table 1. Vector tile solutions and the corresponding supported tile formats.

Vector Tile Solution	Supported Tile Formats	Language	Open Source	Reference
TileStache	GeoJSON [2], TopoJSON [3], ArcJSON [4], MVT [5]	Python	Yes	http://www.tilestache.org/
Mapzen	GeoJSON, TopoJSON, MVT	JavaScript	Yes	https://www.mapzen.com/
OpenLayers	GeoJSON, TopoJSON, KML [6], GML [7], MVT, etc.	JavaScript	Yes	https://openlayers.org/
Kartotherian	PBF [8], JSON [1]	JavaScript	Yes	https://github.com/kartotherian/kartotherian/
Tegola	MVT	Go	Yes	https://tegola.io/
MapBox	Google Protocol Buffers, MVT	C++, JavaScript, C#	No	https://www.mapbox.com/maps/
Google	Google Protocol Buffers	Android	No	https://mashable.com/category/google-maps/
Apple	exclusive vector tile	iOS	No	https://developer.apple.com/maps/web/

[1] JSON(JavaScript Object Notation); [2] GeoJSON (a geospatial data interchange format based on JSON); [3] TopoJSON (an extension of GeoJSON that encodes topology); [4] ArcJSON (); [5] MVT (MapBox vector tile); [6] KML (keyhole markup language); [7] GML (geography markup language); [8] PBF (protocolbuffer binary format).

TileStache (http://www.tilestache.org/) is a Python-based server application that can serve map tiles based on rendered geographic data. It supports vector tile encoding formats like GeoJSON, TopoJSON, ArcJSON and MVT (MapBox vector tile), however it does not support the Google Protocol Buffers vector tile encoding format. Mapzen (https://www.mapzen.com/) focuses on the core components of map displays, such as search and navigation. Mapzen's vector tile service delivers worldwide coverage of OpenStreetMap base layer data, and is available in GeoJSON, TopoJSON, and MVT binary format. OpenLayers (https://openlayers.org/) makes it easy to put a dynamic map on any web page [25,26]. It can display map tiles, vector data, and markers loaded from any source, such as GeoJSON, TopoJSON, KML (keyhole markup language), GML (geography markup language), MapBox vector tiles, and other formats. Kartotherian (https://github.com/kartotherian/kartotherian/) is a map tile service originally built for the Wikimedia projects. Its supported formats include PBF (protocolbuffer binary format) vectors, and JSON. Tegola (https://tegola.io/) is an open source vector tile server written in Go. Tegola

takes geospatial data and slices it into vector tiles that can be efficiently delivered to any client. It supports the MapBox vector tile encoding format. The MapBox (https://www.mapbox.com/maps/) Tile Server supports the Google Protocol Buffers vector tile encoding format. As one of the largest providers customizing web maps, MapBox defines the standard for MapBox vector tiles.

With a lack of formal standards, studies of vector tiles still concentrate on aspects such as the application advantage analysis of web map vector tiles and correct loading and display of vector tiles [27–29]. At present, both Google (https://mashable.com/category/google-maps/) and Apple (https://developer.apple.com/maps/web/) use exclusive vector tile encoding formats that cannot be used by the general public. In addition, vector tiles have problems such as non-refined rendering, dependence on a specific platform, and poor reading experience. In order to further study the vector tile encoding format, a vector tiles solution—namely tile mapping that uses PDF map technology—is proposed in this paper, with the aim of realizing goals that could not be achieved by existing vector tile technologies. Such goals include: convenient and fast generation, independence of specific platforms for real-time data rendering in map services, conformity with human visual reading effects, small data sizes, and a guarantee of the offline map quality. Moreover, the proposed approach can also use a reader with a high market share and improve the convenience of map usage.

PDF is a document recording format developed and designed by Adobe. PDF documents, which contain data encoding all of the information displayed on the screen and output to a printer, image setter, or computer-to-plate (CTP) imager, are entirely independent of the original application and computer platform that created them. Thus, they have become an ideal document format for electronic document transmission and digitized information communication. In 2004, Layton Graphics, Inc. (LGI, a company in the United States, http://laytongraphics.com/) put forward the notion of using PDF documents to store maps and geographic information, and obtained GeoPDF products. Thereafter, it founded a subsidiary called TerraGo Technologies (a company in the United States founded in 2005, https://terragotech.com/), which specialized in GeoPDF research [30]. GeoPDF has expanded many functions based on the existing PDF format standard, provided free serviceable tools for application terminals, and become an important format for map distribution. Thus, it has gained trust from governmental agencies and business corporations on a global scale, and has been extensively applied in multiple countries, including the United States, Canada, and Japan [31]. With continuous development and growth of the GeoPDF standard, OGC (Open Geospatial Consortium) has taken GeoPDF as a recommended PDF map encoding standard, and Adobe is designing PDF2.0 (ISO 32000-2) to manage geographic information based on the GeoPDF encoding standard [32]. At the present time, mainstream GIS software, such as ArcGIS v10.x, FME2013, Intergraph GeoMedia, Erdas, MapInfo, and GDAL, support GeoPDF output. China has mainly carried out research to store maps and geographic information in PDF documents, such as a study of the data model of PDF maps, research on the conversion method from digital maps into PDF documents, and a study on PDF cartographic technology [15,33,34]. In addition, China has undertaken a significant exploration into PDF map application patterns, including the development of a multimedia electronic atlas based on the PDF format [35]. The team of which this paper's authors are a part began the earliest research into PDF maps in China. This team has developed its own PDF map products with obvious advantages in aspects such as large-page map browsing, and is currently expanding towards the application and service of PDF maps.

For a long time, multiple countries have accumulated a large number of topographic maps oriented to print (such as China, we call this topographic map for printing as a digital publishing original) [36–38]. These print-oriented digital topographic maps, although of high data quality, the main use of such maps today is to print paper maps. Because of the large amount of data, and the disconnection from the GIS database, high-quality topographic map results are difficult to use for map network sharing and various information systems. Geospatial PDF solutions such as GeoPDF have facilitated the use of this high-quality topographic map data. However, the area represented by a single GeoPDF file cannot be too large. It can be displayed using a PDF browser such as Adobe Reader,

but the hardware requirements are extremely high. GeoPDF does not provide a regional usage model, such as the inability to provide seamless browsing and multi-scale presentation. This paper adopts the PDF tiling method to realize logically the seamless organization of PDF maps up to the global area, so that PDF tiles provide web map services like existing raster and vector tiles. In addition, PDF tiles make high-quality map products possible as an interactive interface for electronic maps and various information systems. The existing Web Map API can be extended to support PDF format, so that PDF vector tile can provide map services directly, like the vector tiles in GeoJSON format. Of course, the function of printing a paper map is certain.

3. Methodology

3.1. Portable Document Format (PDF) Tile Map Pyramid Model Structure

The PDF tile map pyramid model is a model of multiple scales. The relation of halving and doubling are used to establish a model while data continuity is maintained, so as to realize functions such as infinite zooming and seamless roaming of a PDF tile map. It is necessary to determine the tile size of a PDF tile map, the ranges of longitude and latitude, the number of layers in the pyramid model, and the corresponding map scale during the establishment process of the PDF tile map pyramid model.

3.1.1. Establishment of the Coordinate Transformation Object in Web Mercator Projection

For the convenience of calculation in the determination of a PDF tile map partitioning and index coding, a square region is taken for the PDF tile map pyramid model. Therefore, the Earth is regarded as a sphere rather than a spheroid during the Mercator projection process, also called web Mercator projection. This projection takes the equator as the standard parallel, the prime meridian as the central meridian, and their intersection point as the origin of coordinate system. Eastward is the forward direction of the X-axis and northward is the forward direction of the Y-axis. Its projection formula is shown in following equation:

$$\begin{cases} x = R \times \lambda \\ y = R \times \ln[\tan(\pi/4 + \varphi/2)] \end{cases}, \quad (1)$$

where R = 6,378,137 m (the Earth's radius); λ is longitude; and φ is latitude (both in units of radians).

3.1.2. Establishment of the PDF Tile Map Pyramid Rules

Web Mercator projection is adopted for the PDF tile map. The map scope is a square region with side length of 40,075,016.686 m (meter), corresponding to a latitude range of −85.05° to 85.05° and a longitude range of −180° to 180°, as shown in Figure 1.

The PDF tile map pyramid rules for map data on five scales—1:1,000,000; 1:250,000; 1:50,000; 1:10,000; and 1:2,000—are shown in Table 2. The method of tile partitioning is as follows:

(1) The map scope in Figure 1 is equally divided into 4^9 blocks yielding level 1 of the PDF tile map, and the corresponding map scale is 1:1,000,000;

(2) Each block of level 1 is equally divided into 4^2 blocks yielding level 2 of the PDF tile map, and the corresponding map scale is 1:250,000;

(3) Each block of level 2 is equally divided into 4^3 blocks yielding level 3 of the PDF tile map, and the corresponding map scale is 1:50,000;

(4) Each block of level 3 is equally divided into 4^2 blocks yielding level 4 of the PDF tile map, and the corresponding map scale is 1:10,000;

(5) Each block of level 4 is equally divided into 4^2 blocks yielding level 5 of the PDF tile map, and the corresponding map scale is 1:2,000.

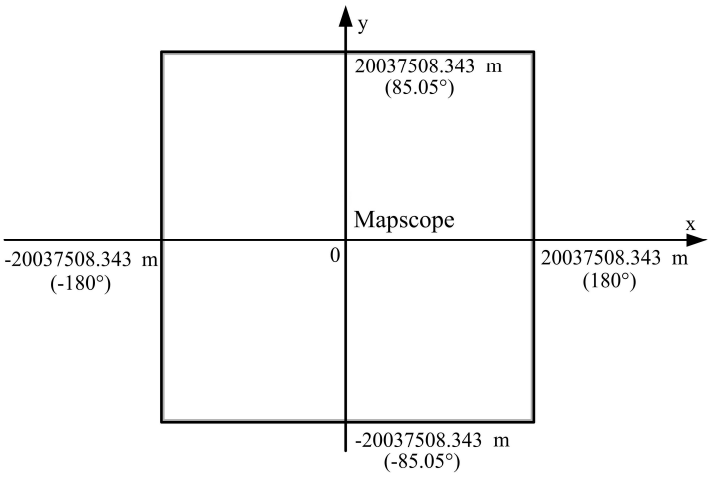

Figure 1. Scope of the square region in the PDF (portable document format) tile map pyramid model. The equator is the x-axis, and the prime meridian is the y-axis. The intersection point of the two is the origin of the coordinate, which is positive to the east and north, and negative to the west and south.

Table 2. PDF tile map pyramid rule list.

Level	Total Number of Tiles	Side Length of Tiles (m)	Tile Page (User Coordinates)	Map Scale
1	4^9	78,271.517	221.872	1:1,000,000
2	4^{11}	19,567.879	221.872	1:250,000
3	4^{14}	2,445.985	138.670	1:50,000
4	4^{16}	611.496	173.337	1:10,000
5	4^{18}	152.874	216.672	1:2,000

The origin of coordinates of the PDF tile map are located at the position with a longitude and latitude of 0, with the lower left point of the tile as the reference point, and a single tile is uniquely represented by the tile code [Z,X,Y], where Z is tile level, for levels 1–5, and X and Y are integers. As shown in Figure 2, tile codes [Z,X,Y] are tile coordinates (X,Y) under a level Z map, where X is the horizontal coordinate and Y is the vertical coordinate. The values are positive when the tiles are upward and rightward with respect to the origin of coordinates and are negative when they are downward and leftward; that is, east longitudes (X) have positive values, west longitudes have negative values, north latitudes (Y) have positive values, and south latitudes have negative values.

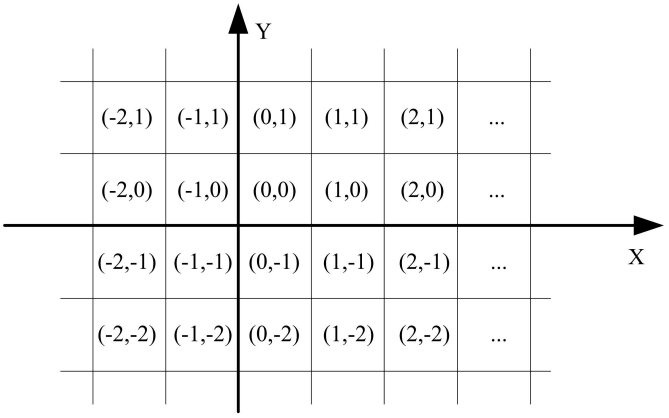

Figure 2. Definition of tile codes [Z,X,Y], where Z is tile level, for levels 1–5, and X and Y are integers. X is the horizontal coordinate and Y is the vertical coordinate. Each tile is labeled according to the coordinate of its lower left corner. Tile coordinates (X,Y) are under a level Z map.

The coordinate transformation of the PDF tile map is shown in Figure 3. The longitude and latitude coordinates and the web Mercator projection coordinates can be mutually transformed through Equation (1) and the relations are unaffected by the tile level. The web Mercator projection coordinates can be transformed into tile coordinates (Equation (2)) and PDF user coordinates (Equation (3)), which depends on the tile level. Meanwhile, the PDF user coordinates can be transformed into web Mercator projection coordinates (Equation (3)).

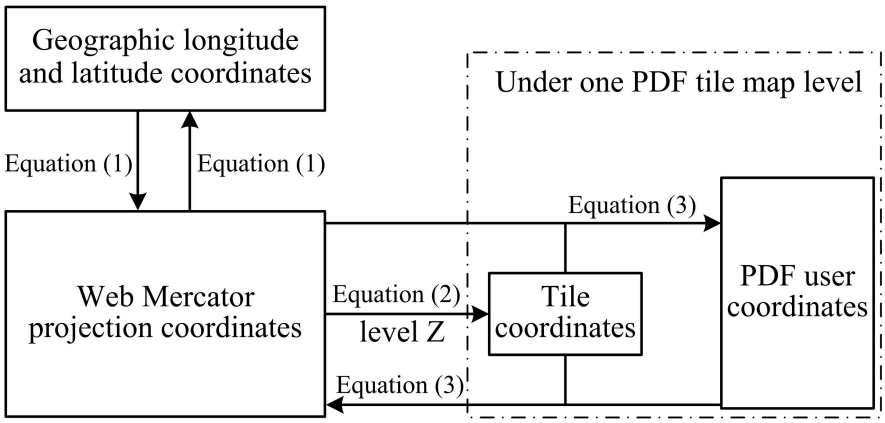

Figure 3. Coordinate transformations in the PDF tile map.

The Web Mercator projection coordinates (x, y) are transformed into tile coordinates (X, Y) for level Z via:

$$\begin{cases} \quad X = INT(\frac{x}{d}) \\ \quad Y = INT(\frac{y}{d}) \\ d = \frac{\pi R}{2^n}, \; where: \begin{cases} n = 2Z + 6, \; (Z = 1, 2) \\ n = 2Z + 7, \; (Z = 3, 4, 5) \end{cases} \end{cases}, \tag{2}$$

where R is the Earth's radius and INT expresses taking the largest integer that does not exceed the numerical value in the bracket.

The web Mercator projection coordinates (x, y) and the tile coordinates (X, Y) for level Z correspond to PDF user coordinates (U_x, U_y) of the PDF tile map via

$$\begin{cases} U_x = \frac{x - d \times X}{s} \\ U_y = \frac{y - d \times Y}{s} \end{cases}, \tag{3}$$

where $s = \frac{2.54K}{7200}$; K is the map scale denominator for level Z (for example, the corresponding map scale is 1:50,000 when Z = 3, thus K = 50,000); and d is defined in Equation (2).

3.2. PDF Tile Map Storage

PDF tile maps consist of symbolic graphic information and geographic entity information, as shown in Figure 4. Symbolic graphic information based on the digital cartographic model (DCM) is mainly composed of cartographic representation information for geographic features. Symbolic graphic information is stored in blocks according to the tile pyramid model and is mainly used for spatial cognitive representation. Each block is stored as a PDF tile file. Geographic entity information based on the digital landscape model (DLM) is mainly used for GIS spatial analysis. Geographic entity information includes geometric and attribute data of geographic features. Geographic entity information is stored for a region using object-oriented modeling technology. Geometric data is not restricted by the region of the PDF tile map pyramid partitioning.

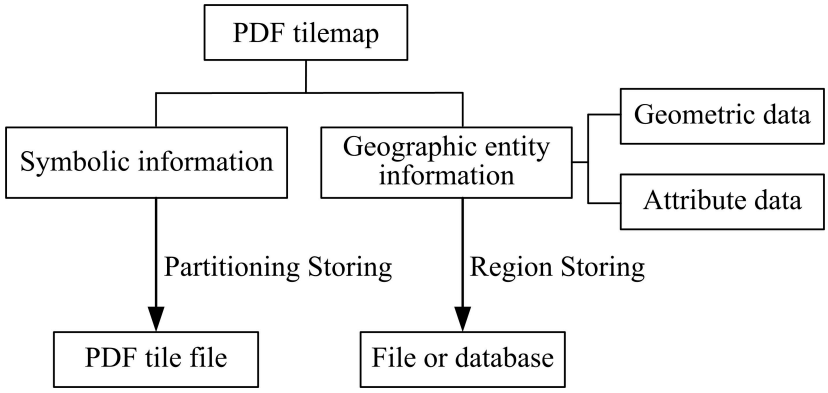

Figure 4. PDF tile map components.

The PDF tile file for a single tile follows the PDF format standard, and its document presents a tree structure. The Document Catalog is taken as the root node, and its child nodes include the Page Tree, Structure Hierarchy, and Configuring Optional Content, as shown in Figure 5.

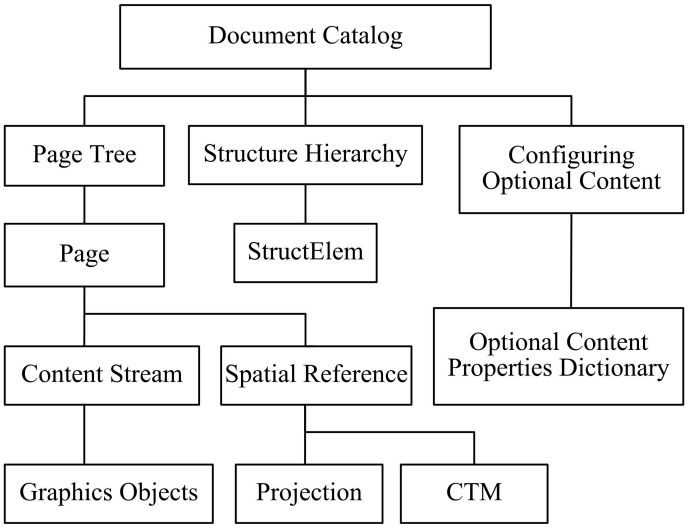

Figure 5. Document structure chart of a single PDF tile. CTM (coordinate transformation matrix), StructElem (structure elements).

The page tree only has one child node, page, which includes the content stream and spatial reference. All symbolic information is stored in the content stream, including a series of graphics objects. A graphics object is a command that draws a graphics element onto the page, including path, text, external, associated image, and shading objects. The spatial reference is used to store conversion parameters for spatial reference so as to realize transformation among geographic coordinates, projection coordinates, and page coordinates. The optional VP (Viewport) entry in a page dictionary, specifies an array of viewport dictionaries that allow different spatial references specified by a measure entry to be used in different areas of the page, if necessary. The main child nodes of spatial reference include the map projection (Projection) and coordinate transformation matrix (CTM).

The logical structure of a document is described by a hierarchy of objects called the structure hierarchy or structure tree. Structure hierarchy is used to store attribute information of geographic elements whose geometric information is stored in the content stream. The *K*th entry specifies the immediate children of the structure tree root, which are the structure elements (StructElem). Any graphics object that corresponds to a structure element may have associated user properties, which can be used to contain the attribute information of geographic elements. The operators of the

graphics object are bracketed as a marked-content sequence between BDC (Begin a Marked-content Sequence) and EMC (End a Marked-content Sequence) operators. The marked-content sequence contains a property list containing the MCID (Marked-content Identifier) entry, which is an integer that uniquely identifies the marked-content sequence within its content stream.

The configuring optional content node includes the optional content properties dictionary (OCProperties), which is used to store graphics layer information of maps and realize the layer display control of symbolic information. According to layer information of map elements, the OCProperties attribute values of the graphics layer resources are set, including graphics layer definition (names and property fields), graphics layer order, and the display switch.

3.3. Organization of PDF Tile Map Data

Although the PDF tile file of a single tile is small, the total data volume of the PDF tile map is very large. For example, the global data volume of a level-3 PDF tile map, with a corresponding map scale of 1:50,000, can reach 4^{14} (268,435,456) PDF tile files, with a data size of 51.2 TB (based on an average of 200 KB for each PDF tile file). Similarly, the global data volume of a level-5 PDF tile map, with a corresponding map scale of 1:10,000, can reach 4^{18} (68,719,476,736) PDF tile files, or a data size of about 12.8 PB. For such large data sizes, if each PDF tile file is independently stored, this will bring about a large quantity of small files. Since the sequential access capabilities of a computer hard disk are far greater than random access capabilities, and a large quantity of scattered PDF tile files will be discontinuously stored on the disk, the read–write operation is subject to random access, which results in very slow file transmission and greatly affects map serviceability when the PDF tile map is accessed, read, distributed, or copied.

In order to solve the above problems, a step-by-step partitioning model is used in this paper to implement data organization of the PDF tile map. This approach is specified as follows: the map scope in Figure 1 is partitioned step by step, and the scope of each region corresponds to 4096 tiles. An SQLite (https://www.sqlite.org/) light-duty database is used to package and store PDF tile files corresponding to tiles contained in each region, as shown in Figure 6.

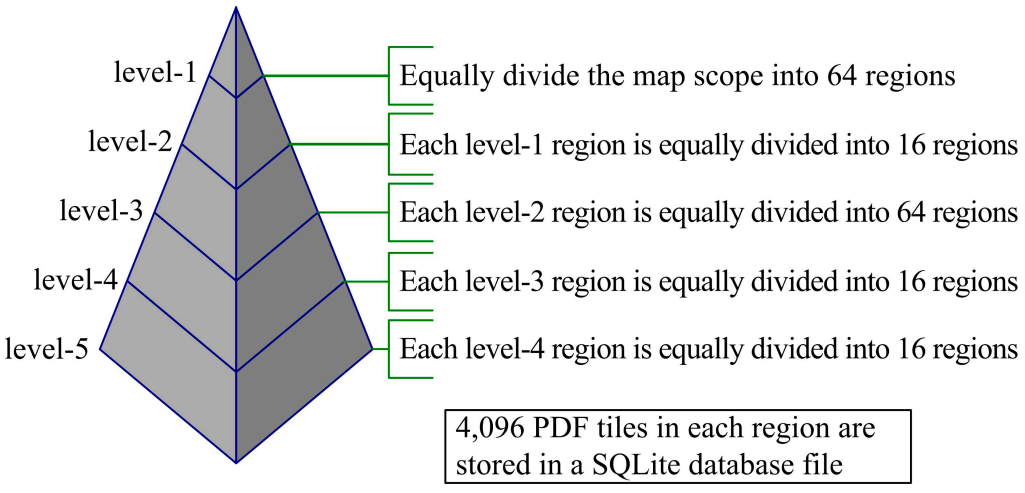

Figure 6. Step-by-step partitioning model of the PDF tile map.

PDF tile map storage includes the storage of PDF tile file data corresponding to tiles at all levels and the storage of the SQLite database files for geographic entity information. SQLite database files include one PDF tile file list and one geographic entity list, where the PDF tile file list stores tile coordinates and the binary data flow of PDF file, while the geographic entity list stores the geometric and property data of the geographic entity. The file catalog organization model of a PDF tile map is shown in Figure 7, and the implementation method of the file catalog organization is as follows:

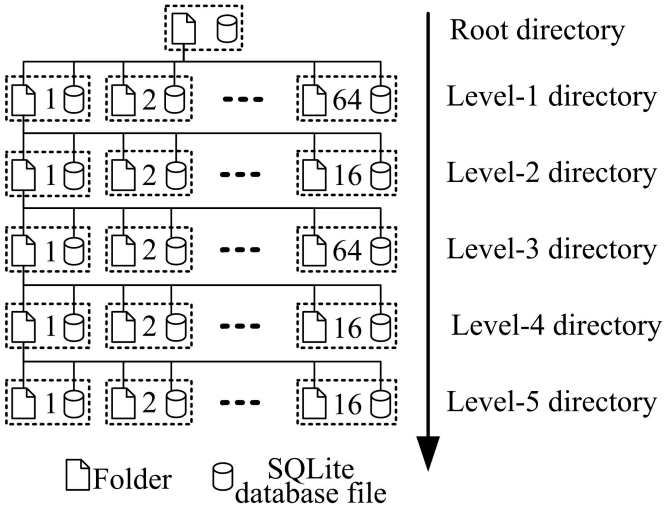

Figure 7. File catalog organization model of a PDF tile map. For Levels 2–5 only the directory structure for the first element is shown and the remaining ones have a similar structure.

(1) Establish a PDF tile map root directory, equally divide the map scope in Figure 1 into 64 regions and establish the index as level-1 regions. For each level-1 region, establish a new folder and an SQLite database file, which are named according to the region, and store them in the root directory as the corresponding level-1 catalog and SQLite database file. Then, store the PDF tile files corresponding to the 4096 level-1 tiles contained in each region in the SQLite database file corresponding to this region.

(2) Equally divide each level-1 region into 16 regions and establish the index for the level-2 regions. For each level-2 region, establish a new folder and an SQLite database file, which are named according to the region, and store them in the level-1 catalog and SQLite database file corresponding to the region. Then, store PDF tile files corresponding to the 4096 level-2 tiles contained in each level-2 region in the SQLite database file corresponding to this region.

(3) Organize level-3, -4 and -5 PDF tile maps in the same way. Store the index information of all regions in the newly established SQLite database file and store them as index files of the PDF tile map in the file catalog.

(4) Divide geographic entity information into different regions step-by-step, and store geometric and property data of the geographic entity at the corresponding scale; at a certain level in the geographic entity data list of the SQLite database corresponding to the appropriate region.

3.4. Dynamic Generation and Updating of the PDF Tile Map

3.4.1. PDF Map Clipping

A PDF page includes five types of graphics objects: a text object (Text), path object (Path), embedded image object (Image), shading object (Shading), and external object (XObject). The path object consists of several linear segments and a cubic Bezier curve, and the path can be used to draw lines and determine the shape of the region to be filled. The main graphics object in the PDF map page is the path object, and a clipping rectangle is used for vector clipping of the path object. The method is as follows:

(1) Clipping of a path consisting of several linear segments.

If the graphic represented by the path is a polyline or polygon and the path only consists of several linear segments, classical algorithms can usually be used for the path clipping, such as the Cyrus–Beck line clipping algorithm [39], the Liang–Barsky linear clipping algorithm [40], the Nicholl–Lee–Nicholl linear clipping algorithm [41], the Weiler–Atherton polygonal clipping algorithm [42], and the Sutherland–Hodgman polygonal clipping algorithm [43]. Schematic diagrams of the clipping of paths consisting of several linear segments are shown in Figure 8.

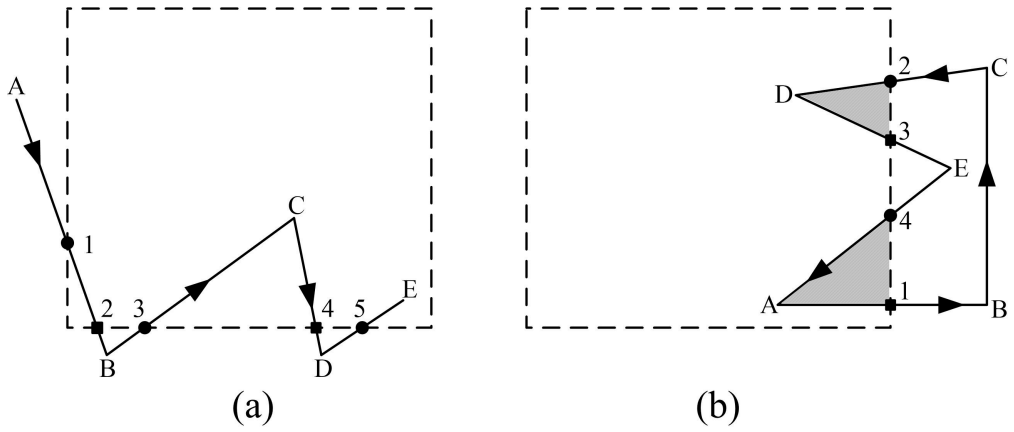

Figure 8. Schematic diagram of the clipping of paths consisting of several linear segments: (**a**) Polyline "A-B-C-D-E" has entry points 1, 3, and 5 and exit points 2 and 4; (**b**) Polygon "A-B-C-D-E-A" has entry points 2 and 4 and exit points 1 and 3.

The path in Figure 8a is a polyline. From the starting point A and along the direction of the path, points 1, 3, and 5 of the polyline mark entry points into the clipping rectangular region, while points 2 and 4 mark exit points of the polyline out of the region. When the path is constructed after clipping, the entry point operator is "m" (path construction operators, begin a new subpath by moving the current point to the entry point), while the exit point operator is "1" (lowercase L, path construction operators, append a straight line segment from the current point to the exit point). The middle parts between the adjacent entry and exit point are reserved, and if the starting point is within the clipping rectangular region, then the starting point is taken as the first entry point. If the end point is inside the clipping rectangular region, then the end point is taken as the last exit point. Thus, the clipped path "1-2, 3-C-4, 5-E" is obtained. The path in Figure 8b is a polygon. From the starting point A, entry points 2 and 4 and exit points 1 and 3 of the polygonal side line from the clipping rectangular region are obtained. If the starting point is inside the clipping rectangular region, it will be taken as the first entry point. With the first entry point as the starting point, one entry point and one exit point constitute one group along the direction of the broken line and the nodes in the middle are reserved. Thus, the polygon is constructed as "A-1-2-D-3-4-A".

(2) Clipping of a path consisting of a cubic Bezier curve or consisting of linear segments and a cubic Bezier curve.

A cubic Bezier curve is taken as the unit path, and path clipping is implemented when there is no concrete calculation of the intersection point between the cubic Bezier curve and the clipping rectangle. Schematic diagrams of the clipping of paths consisting of a cubic Bezier curve or consisting of linear segments and a cubic Bezier curve are shown in Figure 9.

In Figure 9a, from the starting point A, the path consists of linear segments and cubic Bezier curve units S1–S4. First of all, for any cubic Bezier curve unit in the path, such as S1, its starting, ending, and control points are extracted and their minimum bounding rectangle (MBR) is calculated. If any part of the MBR is located inside the clipping rectangular region, this cubic Bezier curve unit is reserved. If the MBR is outside the clipping rectangular region, this cubic Bezier curve unit will be discarded, and its starting and end points are connected by a linear segment. In Figure 9b, S1 and S2 are reserved, while S3 and S4 are discarded. Then from starting point A, the entry points (points 1 and 3) of the path that enter into the clipping rectangular region on a linear segment, and the exit points (point 2) that leave the region on a linear path are determined. Addition, if the end point of a reserved cubic Bezier curve unit is outside the clipping rectangular region and its next segment is a linear segment, then the end point of this cubic Bezier curve unit is taken as the exit point. If the starting point is inside the region, it will be the first entry point. If the end point is inside the region, it will be the last exit point.

Finally, the entry point operator is "m" and the exit point operator is "1", and the middle parts between the neighboring entry and exit point are reserved. Thus, the clipped path "1-S1-S2-2, 3-E" is obtained.

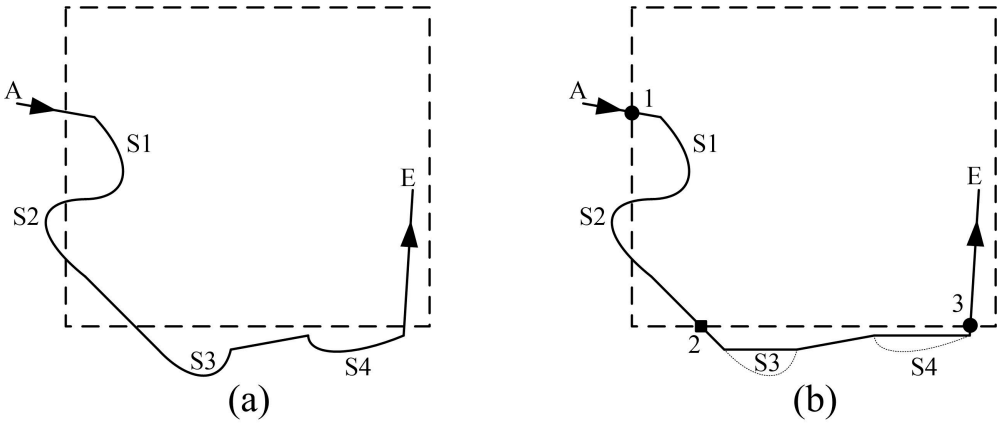

Figure 9. Schematic diagram of the clipping of the path consisting of several linear segments: (**a**) path "A-S1-S2-S3-S4-E"consists of linear segments and cubic Bezier curve units; (**b**) reserved segments S1 and S2, discarded segments S3 and S4, entry points 1 and 3, exit point 2.

3.4.2. PDF Map Splicing

As graphic element contents in one PDF tile file may come from the neighboring two or four PDF maps which are obtained by clipping each PDF map, it is necessary to carry out splicing for same tile-encoded PDF tile files. For instance, as shown in Figure 10, when a tile rectangle is intersected by inner border rectangles of four neighboring PDF maps, and four PDF tile files with the same tile coordinates are obtained after clipping, then splicing treatment will be needed to splice the four PDF tile files and obtain a complete PDF tile file.

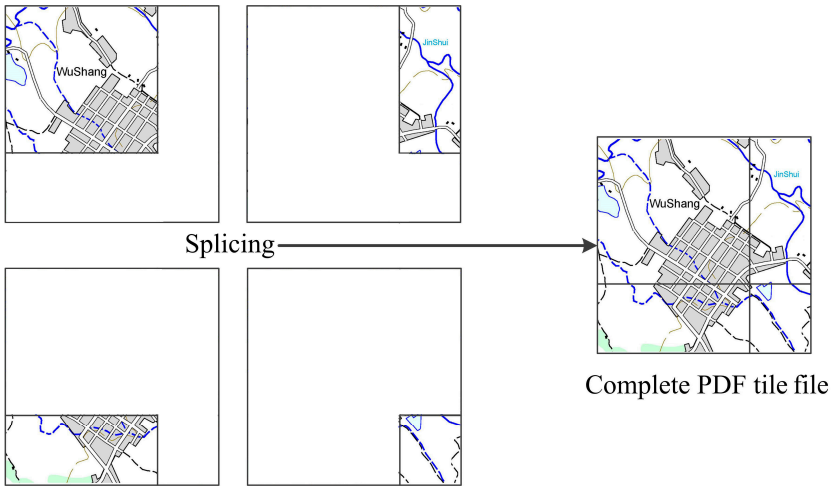

Figure 10. Same tile-encoded PDF tile map splicing.

3.4.3. The Process of Generation and Updating

Topographic maps, projection drawings of surface relief forms with surface feature locations and shapes displayed on a horizontal plane, are indispensable tools for economic construction, national defense, and scientific research. Additionally, topographic maps contain fundamental data for compiling all kinds of ordinary maps, thematic maps, and atlases on small scales. The main data source used to fabricate a PDF tile map is the topographic map. According to the PDF tile map pyramid rules in Table 1, original digitally published topographic maps of scales of 1:1,000,000; 1:250,000; 1:50,000;

1:10,000; and 1:2,000, as well as DLG (digital line graphic) result data are used to realize dynamic generation and updating of the PDF tile map. A schematic of the generation and updating is shown in Figure 11.

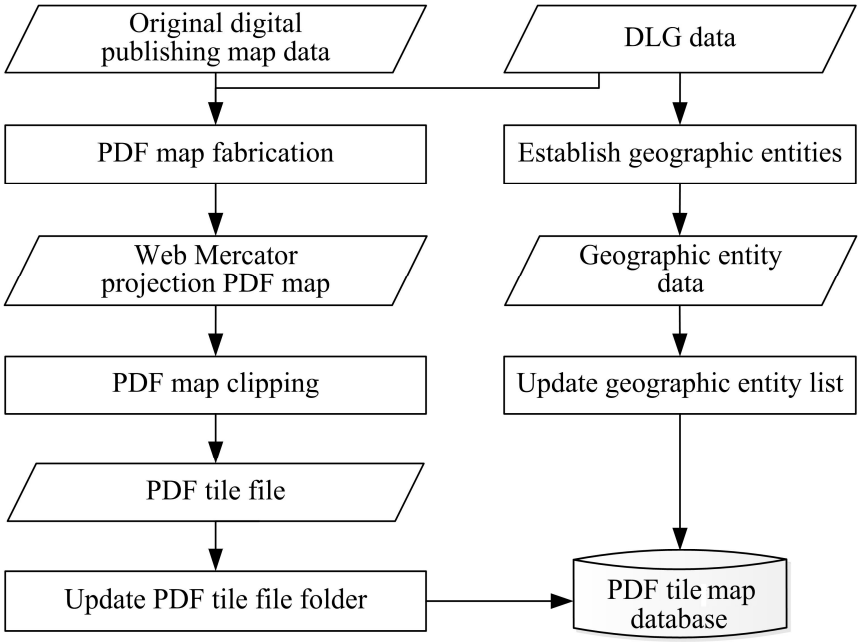

Figure 11. Dynamic generation and updating flowchart of PDF tile map. DLG (digital line graphic). Rectangular boxes denote data-processing operations. Parallelogram boxes denote data-processing results.

The flow shown in Figure 11 is as follows:

(1) Original digitally published topographic maps and DLG result data are used to fabricate the web Mercator projection PDF maps. EPS-format (encapsulated postscript) data of the composite digitally published maps is transformed into PDF-format data, and map orientation is determined from the DLG data, in order to generate a PDF map with spatial reference. The web Mercator projection transformation of graphics objects on the PDF map page is carried out and PDF maps of other map projections are transformed into web Mercator projection PDF maps.

(2) The web Mercator projection PDF map is clipped to generate a PDF tile file. Based on the map scale and sheet range of the PDF maps in the web Mercator projection, the tile level and tiles covered in the map sheet are calculated. For any of these tiles, the tile rectangle is used to clip the web Mercator projection PDF map and to obtain the PDF tile file corresponding to this tile.

(3) PDF tile files are used to update the PDF tile file list in the SQLite database for the PDF tile map catalog. Comparing the positional relationship between the tile rectangle corresponding to the PDF tile file and the rectangle of the web Mercator projection PDF map, allows for the generation of this PDF tile file within the map scope. If the tile rectangle is located inside the rectangle of the map sheet, tile coordinates are used to retrieve the PDF tile file list in the SQLite database and the PDF tile map catalog. Furthermore if the PDF tile file list exists, then the PDF file tile will be used to update the binary data flow of the PDF file in the PDF tile file list. If it does not exist, the PDF file will be stored in the PDF tile file list in the form of a binary data flow. The PDF tile files generated by the neighboring two or four web Mercator projection PDF maps that intersect with this tile rectangle are subjected to splicing to obtain a complete PDF tile file after clipping. This PDF tile file is then used to update the PDF tile file list.

(4) DLG data are used to establish geographic entities and update the geographic entity list in the SQLite database of the PDF tile map catalog.

Geographic entities, such as integrated realms, administrative regions, roads, settlement places, and rivers, are extracted from the DLG data, including unique identification numbers, geometric position

information, and element property information of geographic entities. Coordinated relationships between geographic entities on the same scale and consistency relationships for the same geographic entity on different scales are coordinated; and geographic entity data are used to update the geographic entity list in the SQLite database of the PDF tile map catalog.

4. Experiments and Discussion

4.1. PDF Tile Map Testing Prototype System

4.1.1. Experimental Environment

To evaluate the performance of the model we proposed in the previous section, in a real environment, we set up a test environment and created a verification prototype. Based on this, we carried out a series of experiments. These experiments mainly investigated the performance of our model and compared our PDF tile map with the current raster tiles and vector tiles. Our testing prototype system consists of the PDF tile map production subsystem, the PDF tile map service subsystem, and the offline and online map subsystem, as summarized in Figure 12.

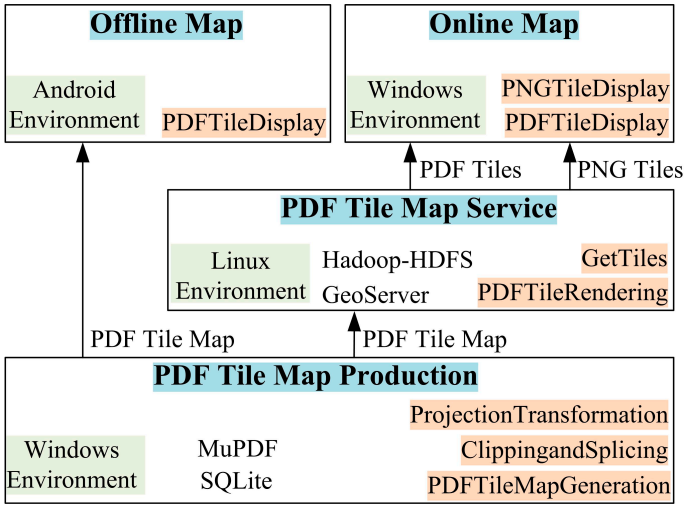

Figure 12. PDF tile map testing prototype system. PNG (portable network graphic format), HDFS (Hadoop Distributed File System). Blue background color denotes subsystem. Red background color denotes access interface. Gray background color denotes operating system.

The PDF tile map production subsystem uses Visual Studio 2015 (Microsoft Corporation, Redmond, WA 98052-7329, USA) as the main development tool and, secondarily, the open source projects MuPDF (1.14.0, https://mupdf.com/) and SQLite (3.20.0, https://www.sqlite.org/). It realizes the projection transformation, clipping and splicing of PDF maps, and generates PDF tile maps in a Windows environment. The PDF tile map service subsystem was implemented in a Linux environment, and installs and deploys the Hortonworks Data Platform (HDP, https://hortonworks.com/), using the Apache Hadoop development tool (HDT) to store and access PDF tile data in the HDFS (Hadoop Distributed File System) on the Hadoop data platform. The PDF tile map service subsystem directly yields PDF tiles through the GetTiles access interface. Moreover, it deploys the WMTS (web map tile service) development environment using the open source GeoServer (Part of a Vibrant Open Source Community, http://geoserver.org/) software, and is used to develop the GetTiles access interface; realizing the WMTS service by rendering PDF tiles to PNG (portable network graphic format) raster tiles. The offline map subsystem in an Android environment that directly stores PDF tile maps in a SQLite database and uses a PDF tile display interface to show maps. The online map subsystem in a Windows environment uses the JavaScript class library package provided by OpenLayers to realize the map display of PDF tiles and PNG tiles released by the WMTS service.

The experimental environment comprises four Dell (China) OptiPlex 7060 desktops with Windows 7 (Microsoft Corporation) operating systems (six-core Core 8 generation i7 processor, 16 GB RAM, 1 TB hard drive) and one Huawei (China) tablet M5 with Android 8.0 (4 GB + 128 GB). On three of the desktops, nine CentOS 7 virtual machines are installed and deployed using VMware (a subsidiary of EMC Corporation, California, USA) Workstation 14 software. The PDF tile map production subsystem and the online map subsystem are deployed on the Windows 7 operating system. The offline map subsystem is deployed on the Android operating system. The PDF tile map service subsystem is deployed on the CentOS 7 operating system. One of the virtual machines serves as the master server of the PDF tile storage, and the remaining eight serve as slaves.

4.1.2. Data Description

The experimental data consists of topographic maps of central China. It was provided by the State Bureau of Surveying and Mapping for national special studies. These have four scales, 1:1,000,000; 1:250,000; 1:50,000; and 1:10,000. Two types of data were used, namely, digitally published original PDF maps and shapefile data. The total data volume is 86.1 GB, of which the PDF data is 63.5 GB and he shapefile data is 22.6 GB. Based on the methods of this paper, for the series of scaled topographic map data, the PDF tile files are created from the digitally published original PDF data, and the geographical entity tables for administrative regions, roads, and residents are created from vector shapefile data in the PDF tile map production subsystem. The production of the PDF tile map is as follows: there are four levels of directories from the root directory of the map, corresponding to scales 1:1,000,000; 1:250,000; 1:50,000; and 1:10,000; for a total of 209 SQLite database files. The total amount of data is 92.9 GB.

4.2. Tests

4.2.1. Offline Map

When users are under extremely poor network conditions, even compressed tiles may take too much time to fetch. PDF tile maps stored in the SQLite database at the terminal will help shorten the waiting period and lessen data traffic. The test of the offline map directly uses and accesses the PDF tiles in the SQLite database file through memory data access. The PDF tile is used for map moving, zooming, and display. The spatial query method is used to query the geographic entity data table in the SQLite database corresponding to the scale at each level and is further used to obtain geographic entity data and display its geometry and attribute information. According to the organization form of the PDF tile map file directory, the offline map system adopts the local deployment mode to deploy PDF tile map database files to the Huawei tablet M5 with an Android terminal system directly. The system architecture is shown in Figure 13.

In Figure 13, the offline map browser is responsible for the request, organization, parsing, and display of PDF tiles; the browser cache is responsible for the cache storage management of PDF tiles; and the file system is responsible for the access scheduling of PDF tiles. When the offline map browser needs to display a map, it requests the required PDF tiles from the browser cache (1). If the required PDF tiles exist in the browser cache, the PDF tiles are directly returned (2 and 3). Then the offline map browser parses and displays the PDF tiles to the corresponding location, and the request ends. If the required PDF tiles do not exist in the browser cache, the offline map browser requests the required PDF tiles from the file system (4), which then passes the required PDF tiles to the browser cache for caching (5 and 6) and returns them to the offline map browser for display (5 and 7).

During the zooming and display processes, the user experience mode consists of two types: one is single-level PDF tile infinite zooming, and the other is multi-level PDF tile adaptive zooming.

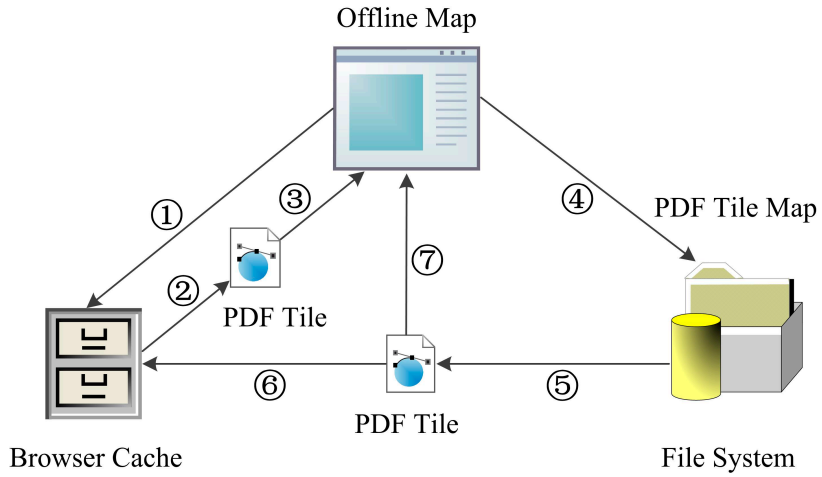

Figure 13. Offline map system architecture. Numbers 1 and 4 denote the process of data request. Numbers 2, 3, 5, 6 and 7 denote the process of data transmission.

(1) Single-level PDF tile infinite zooming: in this mode, map roaming and zooming are based on the same level of PDF tiles, that is, the entire browsing process of the map is based on the same map scale, as shown in Figure 14. In Figure 14, level-3 PDF tiles are used for the map display, corresponding to 1:50,000 map scale and 1:180,000 initial map display ratio. During the map zooming process, the PDF tiles located in the display area are rendered and displayed to the display area directly, according to the new display ratio.

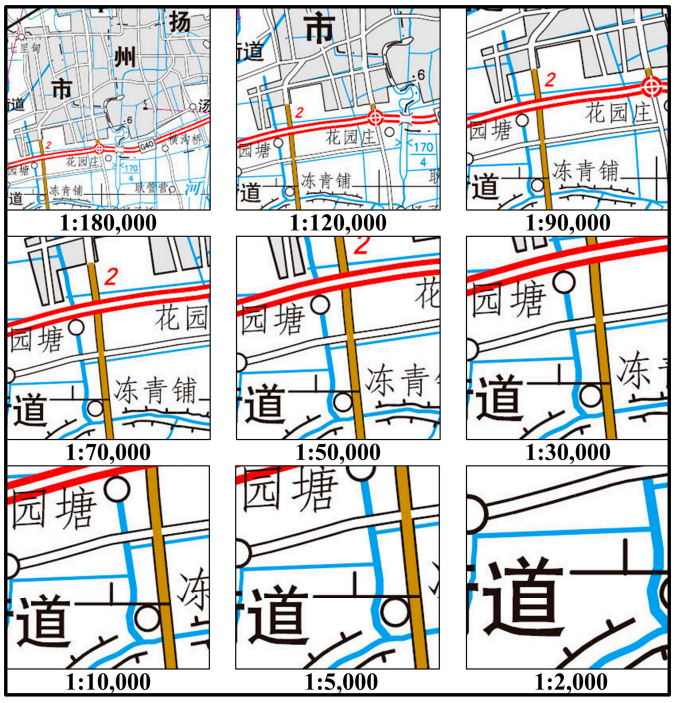

Figure 14. Single-level PDF tile infinite zooming. All ratios from "1:180,000" to "1:2,000" denote the map display ratio.

(2) Multi-level PDF tile adaptive zooming: in this mode, the map zooming operation is for all levels of PDF tiles, and the scale of the map is displayed as a judgment condition to switch adaptively between different levels of PDF tiles, as shown in Figure 15.

Figure 15. Multi-level PDF tile adaptive zooming. All ratios from "1:1,500,000" to "1:30,000" denote the map display ratio. The corresponding map scale of level-1 PDF tile is 1:1,000,000. The corresponding map scale of level-2 PDF tile is 1:250,000. The corresponding map scale of level-3 PDF tile is 1:50,000.

In Figure 15, the first-level PDF file is initially displayed, corresponding to the map scale of 1:1,000,000 and a beginning display ratio of 1:1,500,000 (top left). During the process of map zooming, the first-level PDF tiles located in the display area of the browser cache are directly rendered. Then, the PDF tiles are sent to the display area according to the new display ratio within a certain range, e.g., 1:660,000. When the display ratio of the map is more than a certain value, the level-2 PDF tiles are automatically switched to and cached and scheduled through the browser cache.

Through these experiments, the correspondence between the PDF tile level and the map display ratio range is shown in Figure 16, which further shows the relationship between the full-screen display time of the map on the screen (at the resolution 1920 × 1080) and the ratio of the map display.

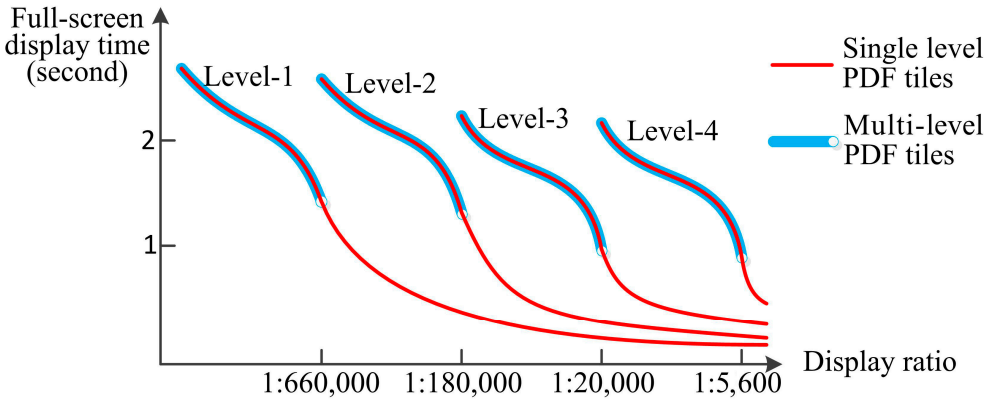

Figure 16. Time curve of the PDF tile map zooming process.

In Figure 16, the red curves indicate the relationships between the full-screen display time and the map display ratio for single-level PDF tile infinite zooming. The blue curves represent the relationships

between the full-screen display time and the map display ratio during multi-level PDF tile adaptive zooming. In the single-level zoom process, the PDF tile level used for display in the screen remains the same regardless of the map display scale. Furthermore, the number of PDF tiles required in the screen decreases as the map display ratio increases. Since the rendering time of a single PDF tile is constant, the time required for full-screen display decreases as the map display ratio increases. In the multi-level adaptive scaling process, as the map display ratio changes, the system adaptively selects the corresponding level of PDF tiles. Therefore, when the PDF tile is switched from a low level to a high level (e.g., switched from level-1 to level-2), the number of PDF tiles required in the screen range is doubled. Since the rendering time of a single PDF tile is constant, when the PDF tile is switched from a low level to a high level, the time required for full screen display increases.

4.2.2. Online Map

The online map uses the distributed storage function of the Hadoop data platform to store all PDF tiles in the SQLite database for the HDFS file system. By rewriting the GetTile access interface in GeoServer, using the background rendering capability of the data server, the vector and raster tile services are supported. There are two types of online maps: ones based on the vector tile service, and ones based on the raster tiles service.

(1) Online map based on vector tile service: The online map based on the vector tile service caches the PDF tiles to the client when acquiring the required PDF tiles directly through the vector tile service. Furthermore, it renders and displays the PDF tiles to the corresponding area using the client browser. The system architecture is shown in Figure 17.

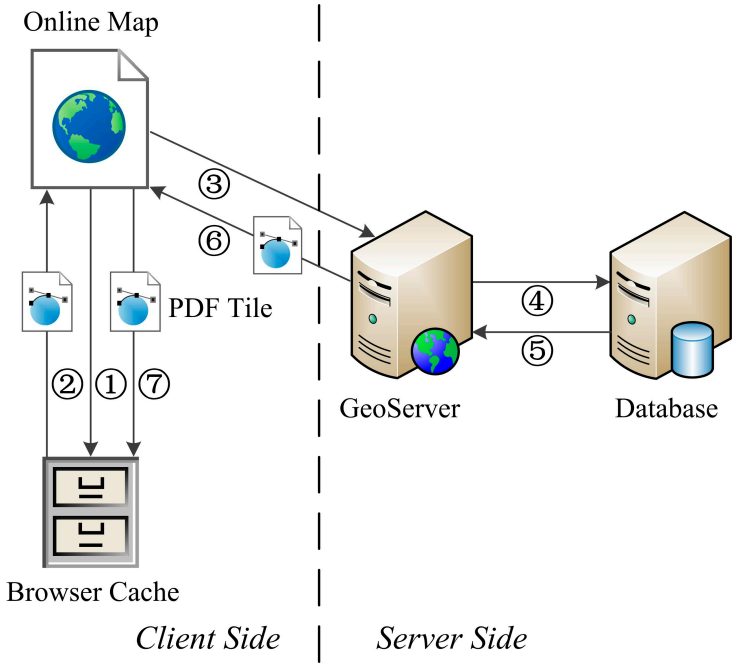

Figure 17. Online map system architecture based on a vector tile service. Numbers 1, 3 and 4 denote the process of data request. Numbers 2, 5, 6 and 7 denote the process of data transmission.

In Figure 17, the online map browser and the browser cache are on the client side, while the GeoServer and the database are on the server side. Among these, the online map browser is responsible for the request, organization, parsing, and display of PDF tiles; the browser cache is responsible for the cache storage management of PDF tiles; the GeoServer is responsible for the vector tile service of PDF tiles; and the database is used to store the Hadoop data platform of the PDF tile map. When the online map browser needs to display a map, it requests the required PDF tiles from the browser cache (1). If these exist in the browser cache, the PDF tiles are directly returned, the online map browser parses

and displays the PDF tile for the corresponding location, and the request ends (2). If the required PDF tiles do not exist in the browser cache, the online map browser requests the required PDF tiles from the GeoServer (3–5), which then passes the required PDF tiles to the online map browser for display (6) and caches them to the browser cache (7).

The map zoom display mode of the online map based on a vector tile service is the same as the offline map, that is, single level PDF tile map infinite zooming and multi-level PDF tile adaptive zooming are possible.

(2) Online map based on a raster tile service: the online map based on a raster tile service obtains the required raster tile directly through the raster tile service, and directly displays the raster tiles by using the client browser. The system architecture is shown in Figure 18.

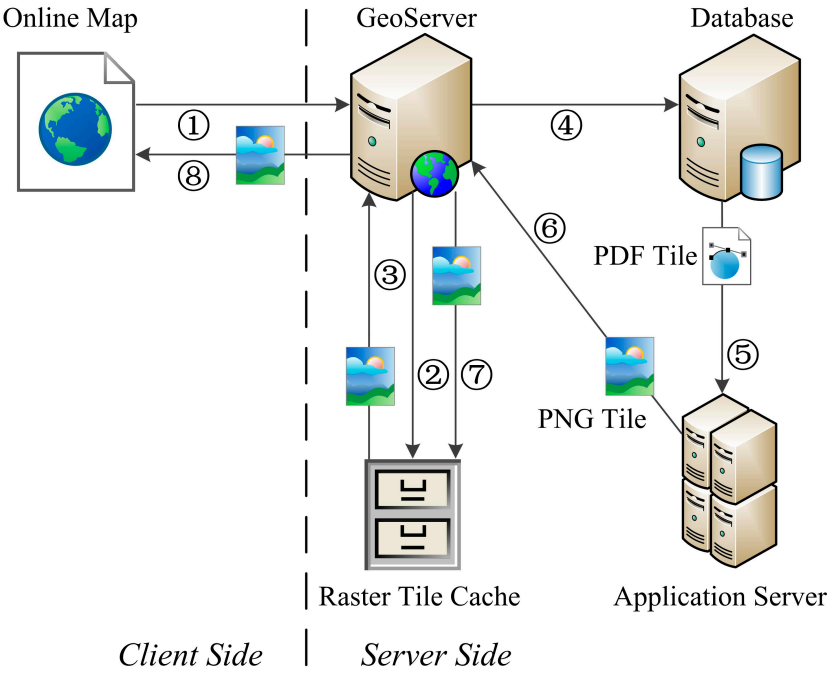

Figure 18. Online map system architecture base on raster tile service. Numbers 1, 2 and 4 denote the process of data request. Numbers 3, 5, 6, 7 and 8 denote the process of data transmission.

In Figure 18, the online map browser is on the client side, while the GeoServer, the raster tile cache, the application server, and the database are on the server side. Among these, the online map browser is responsible for the display of the raster tile map; the GeoServer is responsible for the services of raster tiles; the raster tile cache is responsible for cache storage management of raster tiles; the application server is responsible for rendering the PDF tiles into raster tiles; and the database is used to store the Hadoop data platform of PDF tile maps. When the online map browser needs to display a map, it requests the required raster tiles from the GeoServer directly (1), which then checks whether the required raster tiles exist in the raster tile cache (2). If the required raster tiles exist in the cache, the GeoServer return the raster tiles directly to the online map browser for display (3). If not, the application server gets the PDF tiles corresponding to the raster tiles directly from the database (4 and 5). When the PDF tiles are rendered into raster tiles, the application server provides the raster tiles directly to the GeoServer (6). The GeoServer simultaneously caches the raster tiles to the raster tile cache (7) and returns them to the online map browser for display (8).

The PDF tile is rendered into a 256 × 256-pixel sized PNG image as the raster tile of this level, or 4 256 × 256-pixel sized PNG images as the raster tiles of the next level. When the same PDF tile data source provides a higher level of raster tiles, its efficiency is nearly four times higher.

The application server directly renders the PDF tile map into raster tiles of the PNG image format, as shown in Figure 19, specifically, by rendering the level-1 PDF tile into a 256 × 256-pixel sized PNG image as the ninth-level raster tile, and four 256 × 256-pixel sized PNG images as the tenth-level raster tile. Similarly, the level-2 PDF tile is rendered into the 11th, 12th, and 13th level raster tiles; the level-3 PDF tile is rendered into the 14th and 15th level raster tiles; the level-4 PDF tile is rendered as the 16th and 17th level raster tiles; and the level-5 PDF tile is rendered into the 18th and 19th level raster tiles.

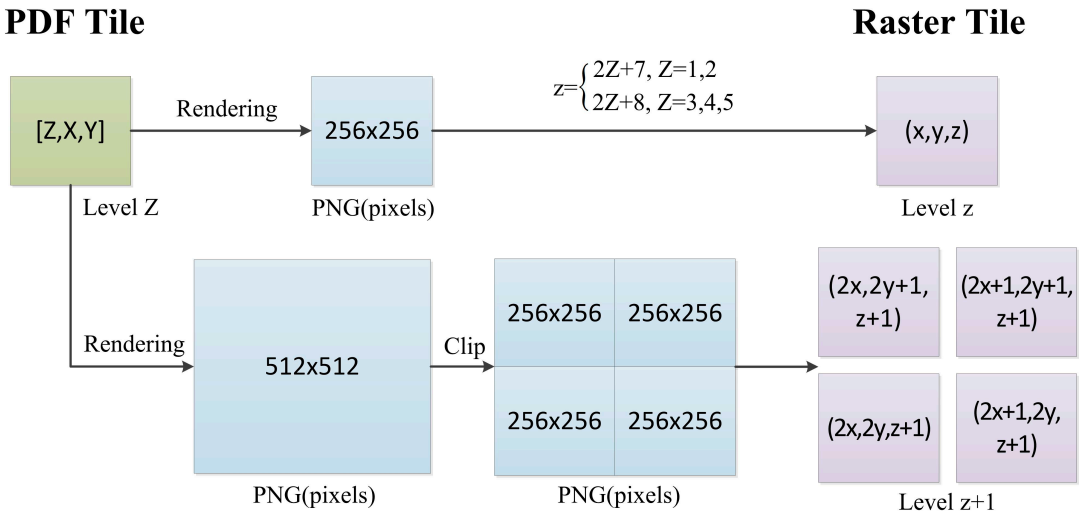

Figure 19. PDF tiles providing raster tiles. Tile codes [Z,X,Y] are tile coordinates (X,Y) under a level Z PDF tile. Tile codes (x,y,z) are tile coordinates (x,y) under a level z raster tile.

For example, if a level-1 PDF tile is rendered into a PNG image of 256 × 256 pixels, the 9th level raster tiles can be provided. If a level-1 PDF tile is rendered into a PNG image of size 512 × 512 pixels, and then divided into four PNG images of 256 × 256 pixels, the 10th level raster tiles can be provided, as further shown in Figure 20. In addition, we found that the time to render PDF tiles into PNG images of size 256 × 256 pixels and PNG images of size 512 × 512 pixels was the same; that is, the same PDF tiles have nearly a four-fold improvement in efficiency when they provide a higher level of raster tile services.

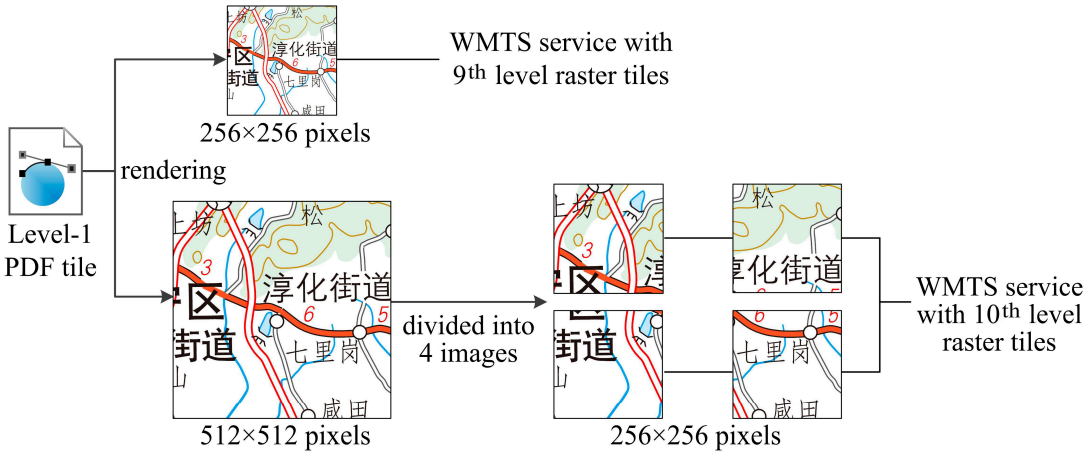

Figure 20. Rendering a level-1 PDF tile into 9th and 10th level raster tiles. WMTS (web map tile service).

The PDF tile rendering time is only related to the size of the PDF tile file, regardless of the size of the image it is rendered into. When the same level of PDF tile provides different levels of raster tile services, the higher the raster tile level, the shorter the time required to render the PDF tile into the

raster tile. Thus, it is possible to pre-render the PDF tile, which costs more rendering time, into a raster tile of a given level for caching, such as raster tile levels 9, 11, 14, and 16, to improve raster tile service efficiency. Other raster tile levels can be rendered into PNG images using PDF tiles in real time, greatly reducing the data volume.

4.3. Results and Discussion

4.3.1. Tile Size

In this section, we compare the sizes of three different tile types, that is, MapBox vector tiles, PNG raster tiles, and our PDF tile map format. Figure 21 presents the average tile size at different tile levels. MapBox vector tiles and PNG raster tiles, which are the publicly available tile types, are included as the comparison baselines. MapBox vector tiles, which have the same area as the PDF tile map, are produced with DLG data. PNG raster tiles are produced with the application server by rendering the PDF tiles into PNG images. We sampled 100 tiles of each kind and at each tile level.

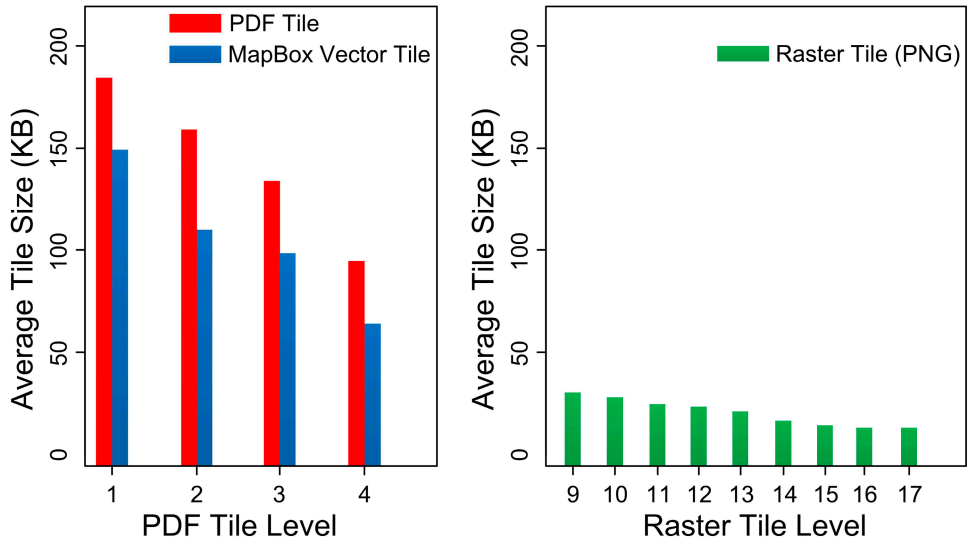

Figure 21. The average tile size of three different types at different levels.

From Figure 21, we can clearly see that the size of the PDF tile is comparable with that of the MapBox vector tile and is significantly larger than that of the PNG tile. The tile size of the PDF tile and MapBox vector tile decreases as the tile level goes up because the scope of each tile becomes smaller, resulting in less spatial objects being included.

All levels of PDF tiles can provide raster tile services at different zoom levels, e.g., Level-1 PDF tiles can provide level 9 and higher raster tiles, and Level-2 PDF tiles can provide level 11 and higher raster tiles. The PDF tile map has a lightweight advantage when providing the same WMTS service. The data volume of the PDF tile map and raster tiles is shown in Table 3. We found that the same level of PDF tile mapping can provide WMTS services equivalent to two or three levels of raster tiles, and the volume of PDF tile data is significantly smaller than the volume of raster tile data. The higher the level, the more obvious this situation will be. For example, the data volume of the Level-4 PDF tiles is 82.18 GB, while the sum of the data volume of the 16^{th} and 17^{th} level raster tiles is 469.81 GB.

Table 3. PDF tile and raster tile data volume statistics.

PDF Tiles			Raster Tiles		
Level of PDF tile	Tile amount (pieces)	Data volume of tile (GB)	Level of raster tile	Tile amount (pieces)	Data volume of tile (GB)
1	72	0.03	9	72	0.01
			10	288	0.03
2	1152	0.26	11	1152	0.09
			12	4608	0.38
			13	18,432	1.47
3	73,728	13.82	14	73,728	5.89
			15	294,912	23.59
4	1,179,648	82.18	16	1,179,648	94.32
			17	4,718,592	375.49

4.3.2. Rendering Performance

In this section, we compare the rendering performance of the PDF tile map with a single GeoPDF file map and MapBox vector tile map. We selected a single GeoPDF file, MapBox vector tiles, and PDF tiles covering the same area. The size of a single GeoPDF file is 39.5 MB. There are 298 single-level MapBox vector tiles, and the total amount of data is 32.5 MB. There are 298 single-level PDF tiles with a total data size of 45.3 MB. The rendering performances of the PDF tile map, a single GeoPDF file map, and a MapBox vector tile map are shown in Figure 22. Figure 22a shows the time-consumption of the initial map loading, map move, map zoom in, and map zoom out features of the three types of data. Figure 22b shows the rendering time comparison for the MapBox vector tiles and PDF tiles as a function of file size.

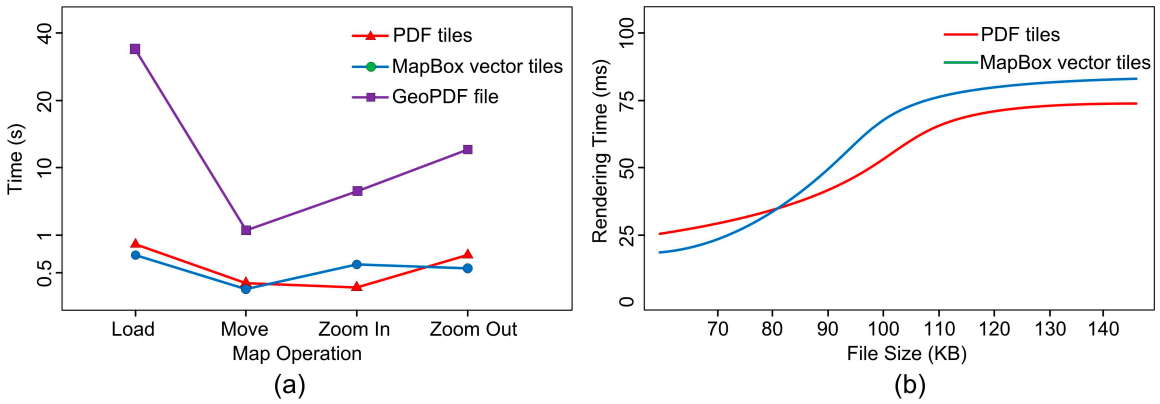

Figure 22. The rendering performance of three types of map data: (**a**) the time-consumption of the initial map loading, map move, map zoom in, and map zoom out; (**b**) the rendering time comparison for the MapBox vector tiles and PDF tiles as a function of file size.

As shown in Figure 22a, we found that a single GeoPDF file map has the longest loading time, and the loading times of a MapBox vector tile map and PDF tile map are on the same order. A single GeoPDF file map loading and display requires the processing of PDF file IO (Input/Output), PDF document parsing, and PDF page rendering. Due to the large amount of data in a single GeoPDF file, it takes a long time to parse the PDF document, which affects the map loading efficiency. During the map loading of a MapBox vector tile, only several blocks within the screen need to be loaded. In the experiment, the screen resolution was 1920 × 1080, and the initial process required loading 40 MapBox vector tiles. As the data size of a single MapBox vector tile is about 100 KB, which can be loaded in milliseconds, the full-screen map loading time is very small. Similarly, the map loading process of a PDF tile map is similar to the map loading process of the MapBox vector tile map. The data

size of a single PDF tile is also about 100 KB. Such a small PDF file has a document parsing time of milliseconds, which makes the full-screen map loading time very small. In Figure 22b, the rendering times of a MapBox vector tiles and PDF tiles are almost similar. When the data size is small, such as less than 80 KB, the MapBox vector tile rendering time is less than the PDF tile rendering time. When the data size is larger than 80 KB, the MapBox vector tile rendering time is greater than the PDF tile rendering time.

4.3.3. Visual Effect

In this section, we selected the vector tiles and raster tiles in the same position as the PDF tiles for infinite zooming, and then observed the display at different zoom ratios, as shown in Figure 23.

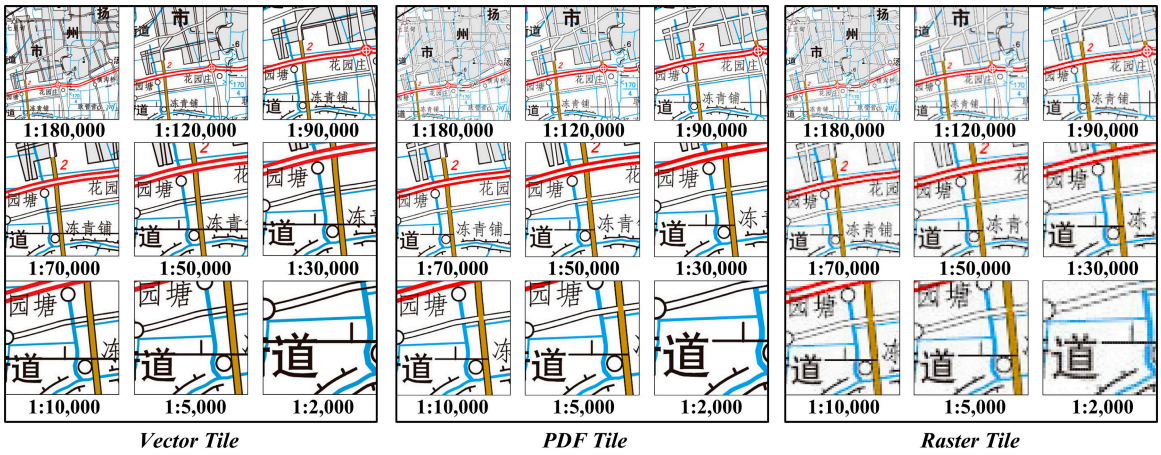

Figure 23. Visual effect of vector tiles, PDF tiles, and raster tiles at different zoom ratios.

As shown in Figure 23, we found that vector tiles visual effects are graphical representations implemented by symbolic collocations under the support of the symbol library. PDF tiles are a detailed map that satisfies paper map publishing requirements, and maintains vector characteristics during the infinite zooming process. Raster tiles are composed of PNG images, which blur during infinite map zooming.

4.3.4. Map Service Performance

In the vector tile-based web map service model, the PDF format serves as a carrier for data transfer. The client receives the required PDF tiles directly from the server, and parses and renders the PDF page into a bitmap for display on the screen. In the raster tile-based web map service model, the PNG format serves as a carrier for data transfer. The server side needs to render the PDF tiles into PNG format raster tiles, and the client displays the PNG format raster tiles directly to the screen. The performance of the map service in an online map environment depends on several technological factors. Among these are the internet connection, traffic intensity, data efficiency, and capacity of the client and server machines. Since the patterns of map services are different, and the data formats sent to the client are different, the response time for PDF tile online maps differs for vector tile and raster tile services. Online mapping based on the vector tile service requires more work by the client, while less strain is put on the server. Online mapping based on the raster tile service requires more work by the server, and less work by the client. As we can see in the results shown in Figure 24, there are large differences in the performance of the map at different zoom levels.

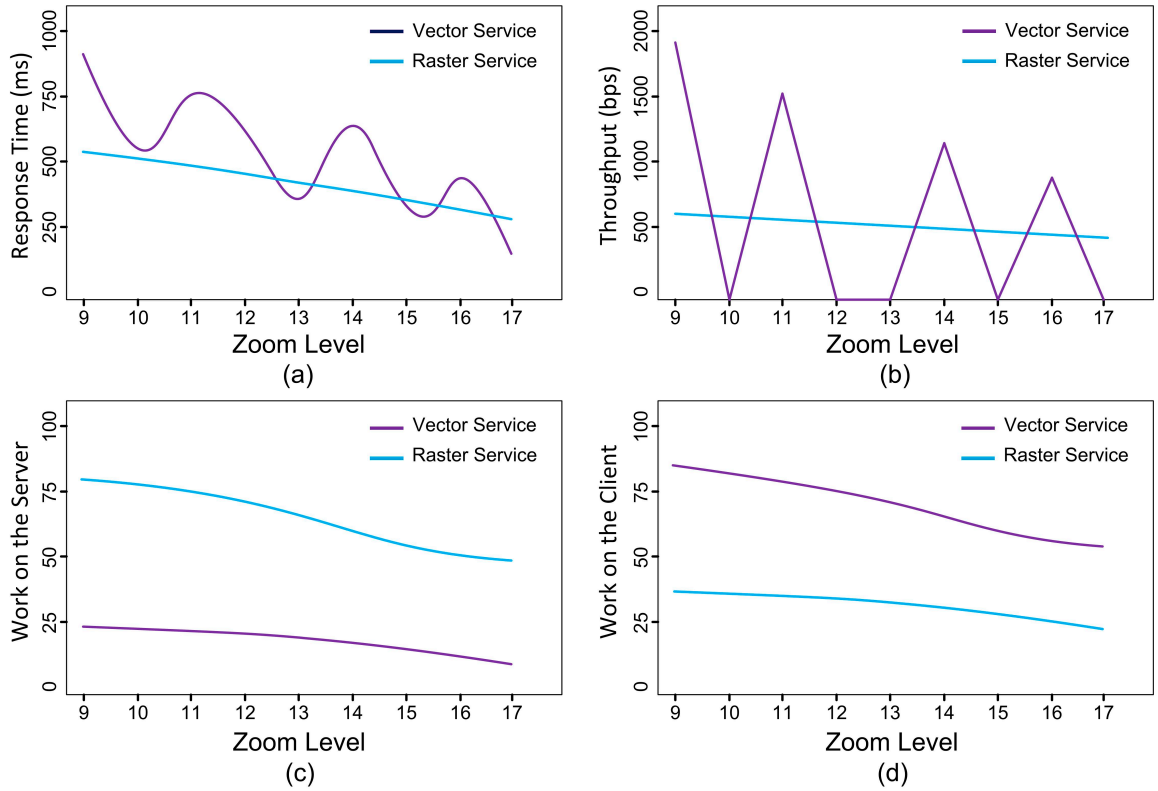

Figure 24. The performance of two map service patterns: (**a**) the total response time for the map display based on vector tiles service and raster tiles service; (**b**) the data throughput for vector tiles and raster tiles; (**c**) the amount of processing work on the server side; (**d**) the amount of processing work on the client side.

In Figure 24a, the total response time for the map display based on vector tiles is slightly larger than that based on raster tiles. The total response time mainly comprises network delay time, data processing time, data transmission time, and map display time. Furthermore, the network delay time and data transmission time of the two cases are basically equal. Thus, the data processing time and map display time will directly affect the total response time of the two maps. For the map service based on vector tiles, the server side does not need data processing, and directly transfers PDF tiles to the client. For the map service based on the raster tiles, the server side needs to render the PDF tiles into PNG format and then transfers the PNG tiles to the client. Due to the high performance of the server and the support for distributed processing, the time to render PDF tiles into PNG tiles on the server side is negligible. For map services based on vector tiles, the client side renders the PDF tile into a bitmap in real time and displays it on the screen through a single thread. This rendering process takes a long time due to the low performance of the client hardware. For map services based on raster tiles, the client side directly displays the PNG tiles to the screen, a process which takes almost no time.

In Figure 24b, when the zoom levels are 9, 11, 14, or 16, the data throughput for vector tiles is larger than that for raster tiles. At other zoom levels, the data throughput for vector tiles is zero. This is because the map services based on vector tiles need to load the corresponding PDF tile levels of level-1, level-2, level-3, and level-4 in real time when at zoom levels of 9, 11, 14, and 16, respectively. At other zoom levels, this approach directly utilizes the previous level of PDF tiles in memory and does not require loading of new PDF tiles. As shown in Figures 24c and 24d, the map services based on vector tiles require a small amount of processing work on the server side and a large amount of processing work on the client side. On the other hand, the map services based on the raster tiles requires a large amount of processing work on the server side and a small amount of processing work on the client side.

5. Conclusions

This paper proposed a PDF tile map model combined with PDF mapping technology that is suitable for a web map service with the aim of providing a new solution for vector tiles. During the methodology description, this paper introduced a PDF tile map pyramid structure, designed an organization model of PDF tile map data, and elaborated on a dynamic generation and updating method of PDF tile maps. Then, we verified the accuracy, feasibility, and properties of the map services using the proposed model through a series of experiments.

To achieve our goal, we created a PDF tile map pyramid model and successfully generated a PDF tile map based on DLG data and series scale digitally published topographic maps within a map sheet range of 1:1,000,000. The following conclusions can be drawn. First, the PDF tile map has the good characteristics of a vector map. The amount of PDF tile data for the same area is much smaller than the amount of raster tile data. The amplification of the PDF tile map retains the clarity of the vector graphics, while the amplification of raster tiles results in a reduction of resolution. Second, the PDF tile map model can seamlessly manage a series of scaled PDF maps, which provides multi-level adaptive display abilities. It can not only provide map services in the form of vector tiles, but also render the PDF tiles into PNG images at the server-terminal to provide map services in the form of raster tiles. The same level of PDF tiles can be rendered into different levels of raster tiles, and the rendering time of a single vector tile decreases as the tile level increases. Third, PDF tile mapping does not require the support of cartographic software or a symbol database when building the map service system. It simplifies the workflow of symbolization compared to a traditional vector tile map, and simultaneously reduces the requirement for the software and hardware of the system. Fourth, PDF tile map data is organized by the geometric data, attribute data, and symbolic data of a geographic entity. The symbolic information guarantees the ability of the PDF tile map to display the map, while the geometry and attribute information guarantees the interaction ability at the element level.

It can be seen that the PDF tile map model proposed in this paper has the characteristics of a vector tile map and has the ability to provide a web map service in the form of vector tiles. In the future, we will combine automatic cartographic generalization technology to realize the multi-representation of a PDF tile map. This improvement would enable the PDF tile map to automatically select and generalize map features with the zoom operation of the map.

Author Contributions: Conceptualization, Xiaodong Zhou; Methodology, Xiaodong Zhou and Nina Meng; Software, Xiaodong Zhou and Nina Meng; Validation, Xiaodong Zhou and Tinghua Ai; Formal Analysis, Xiaodong Zhou and Tinghua Ai; Data Curation, Peng Xie; Writing-Original Draft Preparation, Xiaodong Zhou and Nina Meng; Writing-Review and Editing, Tinghua Ai; Visualization, Nina Meng; Supervision, Xiaodong Zhou; Project Administration, Tinghua Ai; Funding Acquisition, Nina Meng and Tinghua Ai.

Funding: This research was funded by the National Natural Science Foundation of China, grant number [41501498], the National Key Research and Development Program of China, grant number [2017YFB0503500], the Fundamental Research Funds for the Central Universities, grant number [300102268207], and the State Key Laboratory of Geo-information Engineering, grant number [SKLGIE2017-M-4-2].

Acknowledgments: We would like to thank the editors and the anonymous reviewers for their valuable comments and Chuncheng Yang for valuable input.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gao, J. The 60 Anniversary and Prospect of Acta Geodaetica et Cartographica Sinica. *Acta Geod. Cartogr. Sin.* **2017**, *46*, 1219–1225. [CrossRef]
2. Wang, J.; Cheng, Y. Discussions on Properties of Cartography and the Value of Map. *Acta Geod. Cartogr. Sin.* **2015**, *44*, 237–241. [CrossRef]
3. Guo, R.; Ying, S. The Rejuvenation of Cartography in ICT Era. *Acta Geod. Cartogr. Sin.* **2017**, *46*, 1274–1283. [CrossRef]
4. Menno-Jan, K. The role of the map in a Web-GIS environment. *J. Geogr. Syst.* **2004**, *6*, 83–93. [CrossRef]

5. Michael, P.P. Web-Mapping Services. In *International Encyclopedia of Geography: People, the Earth, Environment and Technology*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2017; wbieg0890. [CrossRef]
6. Rosatti, G.; Zorzi, N.; Zugliani, D.; Piffer, S.; Rizzi, A. A Web Service ecosystem for high-quality, cost-effective debris-flow hazard assessment. *Environ. Model. Softw.* **2018**, *100*, 33–47. [CrossRef]
7. Pingel, T.J. Using Web Maps to Analyze the Construction of Global Scale Cognitive Maps. *J. Geogr.* **2017**, *117*, 153–164. [CrossRef]
8. Taher, H.A. Mapping Environmental Sounds Using Google Map (Acoustic Maps). In Proceedings of the 2018 International Conference on Advanced Science and Engineering (ICOASE), Duhok, Iraq, 9–11 October 2018; pp. 431–436. [CrossRef]
9. Lin, W. Volunteered Geographic Information constructions in a contested terrain: A case of OpenStreetMap in China. *Geoforum* **2018**, *89*, 73–82. [CrossRef]
10. Antoniou, V.; Morley, J.; Haklay, M.M. Tiled vectors: A method for vector transmission over the Web. In *International Symposium on Web and Wireless Geographical Information Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 56–71.
11. Gaffuri, J. Toward Web mapping with vector data. In *Geographic Information Science*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 87–101.
12. Li, C.; Lu, H.; Xiang, Y.; Liu, Z.; Yang, W.; Liu, R. Bringing Geospatial Data Closer to Mobile Users: A Caching Approach Based on Vector Tiles for Wireless Multihop Scenarios. *Mob. Inf. Syst.* **2018**, *2018*, 5186495. [CrossRef]
13. Chang, K.T. *Introduction to Geographic Information Systems*; McGraw-Hill Higher Education: Boston, MA, USA, 2006.
14. LI, L.; Xu, M.; Yin, Z.; Zhu, H. Development of Map-based Visualization of Geographical Information. *Eng. Surv. Mapp.* **2006**, *15*, 11–14.
15. Zhou, X.; Yang, C.; Meng, N. A New Digital Map Product—PDF Map. *Bull. Surv. Mapp.* **2016**, *1*, 108–110. [CrossRef]
16. Wagemann, J.; Clements, O.; Marco Figuera, R.; Rossi, A.P.; Mantovani, S. Geospatial web services pave new ways for server-based on-demand access and processing of Big Earth Data. *Int. J. Digit. Earth* **2017**, *11*, 7–25. [CrossRef]
17. Li, S. HTML5 Based Ajax and SSE Technology with Server Intelligent Instant Messaging to Meet the Needs of Transactional Websites. *ITM Web Conf.* **2019**, *25*, 01008. [CrossRef]
18. Utomo, S.B. Mapping to predict and prioritize branches affected by natural disasters based on web applications and Google Maps geocoding methods in the group audit and risk advisory. (case study: PT. Astra International, tbk.). *J. Phys. Conf. Ser.* **2019**, *1165*, 012003. [CrossRef]
19. Mehrotra, S.; Chaddha, N.; Gray, R.M. Predictive Hierarchical Table-Lookup Vector Quantization with Quadtree Encoding. In Proceedings of the 3rd IEEE International Conference on Image Processing, Lausanne, Switzerland, 19 September 1996; Volume 3, pp. 407–410.
20. Nie, Y.F.; Xu, H.; Liu, H.L. The Design and Implementation of Tile Map Service. *Adv. Mater. Res.* **2010**, *159*, 714–719. [CrossRef]
21. Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Hagen, S.; Schaub, T. The Geojson Format, No. RFC 7946. 2016. Available online: https://tools.ietf.org/html/rfc7946 (accessed on 25 July 2019).
22. Bacik, V. Possibilitties of TopoJSON format and D3 library by the visualization of geodata in the Internet. *Geogr. Cassoviensis* **2015**, *9*, 5–16.
23. Friesen, J. Introducing JSON. In *Java XML and JSON*; Apress: Berkeley, CA, USA, 2019; pp. 187–203.
24. Popić, S.; Pezer, D.; Mrazovac, B.; Teslić, N. Performance evaluation of using Protocol Buffers in the Internet of Things communication. In Proceedings of the 2016 International Conference on Smart Systems and Technologies (SST), Osijek, Croatia, 12–14 October 2016; pp. 261–265. [CrossRef]
25. Langley, P.J.; Perez, A.S. *OpenLayers 3. x Cookbook*; Packt Publishing Ltd.: Birmingham, UK, 2016.
26. Maiellaro, N.; Varasano, A. One-Page Multimedia Interactive Map. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 34. [CrossRef]
27. Wan, L.; Huang, Z.; Peng, X. An Effective NoSQL-Based Vector Map Tile Management Approach. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 215. [CrossRef]
28. Zouhar, F.; Senner, I. Web-Based Visualization of Big Geospatial Vector Data. In *The Annual International Conference on Geographic Information Science*; Springer: Cham, Switzerland, 2019; pp. 59–74. [CrossRef]

29. Hu, W.; Li, L.; Wu, C.; Zhang, H.; Zhu, H. A parallel method for accelerating visualization and interactivity for vector tiles. *PLoS ONE* **2019**, *14*, e0221075. [CrossRef]
30. Pardue, J. *The GeoPDF File: A New Solution for the Digital Publication, Distribution, and Collaboration of Geospatial Data*; US Geological Survey: Reston, VA, USA, 2008; p. 30.
31. Caputa, R.G. The GeoPDF Project: Creating Maps for Non-Mapper. *Engineer* **2010**, *1*, 36–37.
32. OGC. GeoPDF Encoding Best Practice. Version 2.2. 2009. Available online: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwiW_p7kmaLkAhUDYlAKHQr2C2QQFjAAegQIAhAC&url=https%3A%2F%2Fportal.opengeospatial.org%2Ffiles%2F%3Fartifact_id%3D40537&usg=AOvVaw3zPjRg61IbVFL78Iw0dOyf (accessed on 25 July 2019).
33. Deng, S.; Liu, Z.; Wang, X. A Method to Translate Digital Map to PDF File. *J. Zhengzhou Inst. Surv. Mapp.* **2001**, *2*, 150–153.
34. Hu, H. *Research on Critical Technique of Geo-PDF Map Cartography*; PLA Information Engineering University: Zhengzhou, China, 2011.
35. Zhang, D.; OU, R.; Wang, J.; Liu, Y. Discussion on the Development of Multimedia Electronic Atlas Based on PDF Format Document. *Geomat. Spat. Inf. Technol.* **2012**, *35*, 61–63.
36. Ning, J.; Wang, Z. Progresses from Surveying and Mapping to Geomatics. *Acta Geod. Cartogr. Sin.* **2017**, *46*, 1213–1218. [CrossRef]
37. Liao, K. Retrospect and Prospect of the Development of Chinese Cartography. *Acta Geod. Cartogr. Sin.* **2017**, *46*, 1517–1525. [CrossRef]
38. Meng, L. The Constancy and Volatility in Cartography. *Acta Geod. Cartogr. Sin.* **2017**, *46*, 1637–1644. [CrossRef]
39. Cyrus, M.; Beck, J. Generalized Two and Three Dimensional Clipping. *Comput. Graph.* **1979**, *3*, 23–28. [CrossRef]
40. Liang, Y.; Barsky, B. A New Concept and Method for Line Clipping. *ACM Trans. Graph.* **1984**, *3*, 1–22.
41. Nichoil, T.M.; Lee, D.T.; Nicholl, R.A. *An Efficient New Algorithm for 2-D Line Clipping: Its Development and Analysis*; ACM: New York, NY, USA, 1987; Volume 21, pp. 253–262.
42. Weiler, K.; Atherton, P. Hidden surface removal using polygon area sorting. *Comput. Graph.* **1977**, *11*, 214–222. [CrossRef]
43. Sutherland, I.E.; Hodgman, G.W. Reentrant polygon clipping. *Commun. ACM* **1974**, *17*, 32–42. [CrossRef]