Effective Techniques for Interactive Rendering of Global Terrain Surfaces

Liqiang Zhang, Chongjun Yang, Suhong Liu, Yingchao Ren, Donglin Liu, and Xiaoping Rui

***Abstract*—Global terrain visual systems must support real-time visualization and manipulation of huge multiresolution geographical datasets. In this letter, we focus on certain key techniques, such as construction of three-dimensional (3-D) ellipsoidal models, spatial indexing mechanisms, interactive terrain datasets simplification by the M-band wavelets and triangulation techniques, and surface browsing from different view directions. Finally, an experiment is carried out using different levels of detail data; our results suggest that the methods are effective in enhancing performance of a 3-D global visual system.**

***Index Terms*—Ellipsoidal quadtrees, M-band wavelets, multiresolution, spatial indexing.**

I. Introduction

THE PERFORMANCE and capabilities of inexpensive three-dimensional (3-D) graphics accelerators have shown explosive growth in recent years. Consequently, state-of-the-art home personal computers are now capable of interactive display of complex environments. Three-dimensional visualization has become an important component of GIS.

Our group has developed a web-based 3-D global visualization information system, named *Geobeans3D*. It fuses massive spatial datasets into a seamless 3-D model database of the globe; *Geobeans3D* can also visualize features such as ground cover and trees, buildings and other static objects. The objective of the system is to give a new view of our earth across network depending on system load condition and network bandwidth. This letter reports on our research oriented toward full integration of 3-D earth models and terrain surface rendering.

II. Previous Work

A. View-Dependent Meshes

Lindstrom *et al.* [1] implement terrain patches visualization by introducing a hierarchical quadtree technique. The quadtree recursively divides terrain surfaces into a number of tessellates and then constructs a view-dependent approximate height map. The subdivision operations are fast and simple. However, the method requires much memory, and is not suitable for handling huge datasets. Hoppe [2] first splits the terrain surfaces into many blocks that are merged through independent decimation, and then a set of progressive meshes from the blocks is generated using a bottom-up scheme depending on the viewpoint and the screen-space error metric. However, the method consumes too much memory space and the data structure of the scene retains data redundancy. McArthur *et al.* [3] describe an approach for generating a hierarchical, multiresolution polygonal database from raw elevation data using the wavelet transforms. But he fails to discuss the algorithm performance on frame rates and number of the triangles rendered per frame.

B. Out-of-Core and Streaming Techniques

Lindstrom *et al.* [4] present a method called SOAR for efficient out-of-core management and rendering of continuous adaptive terrain representation. The refinement framework is easy to implement and display huge terrain datasets at high interactive frame rates. Streaming hierarchical level of detail (HLOD) [5] is proposed, which incorporates streaming into a HLOD rendering system that allows for view-dependent refinement. But the scene data structure keeps much data redundancy and it is necessary to enhance the efficiency of data organization. Cignoni *et al.* [6] use a technique called P-BDAM for out-of-core management and interactive rendering of planet-sized terrain surfaces. It is said that it achieves better peak performance than SOAR on the same machine.

C. Multithreading and Clustering Techniques

Roth *et al.* [7] propose an approach for a multithreading and clustering framework that allows to manage the scene contents. The framework uses selective replication of data to give each thread the illusion of owning a full copy of the whole scene. Synchronization is handled via ChangeLists which support cluster-based rendering in sort-first or sort-last fashion. Since not all the scenes being updated are accessed by the thread, certain work should be done to further enhance the display efficiency.

III. Three-Dimensional Global Models

The proper 3-D coordinate system is essential to represent spatial objects on the screen. *Geobeans3D* uses a right-handed, orthogonal coordinate system (see Fig. 1) to model the objects in the 3-D space. Digital elevation models (DEMs) and textures may be regarded as flat-projected surfaces within the given error of this coordinate system. Hence, the view-dependent hierarchical multiresolution terrain databases do not have to be modified, considering the curvature of the earth.

Manuscript received March 15, 2004; revised January 20, 2005. This work was supported by 863 Program and Program of Data Fusion for Flood Analysis and Decision Support (ANFAS).

L. Zhang is with the State Key Laboratory of Remote Sensing Sciences, Jointly Sponsored by Beijing Normal University and the Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing, China, 100875 (e-mail: zihaozhang2003@yahoo.com.cn).

C. Yang, Y. Ren, D. Liu, and X. Rui are with the State Key Laboratory of Remote Sensing Sciences, Institute of Remote Sensing Applications, Chinese Academy of Sciences. Beijing, 100101, China.

S. Liu is with the State Key Laboratory of Remote Sensing Sciences, Beijing Normal University, Beijing, 100101, China.

Digital Object Identifier 10.1109/LGRS.2005.

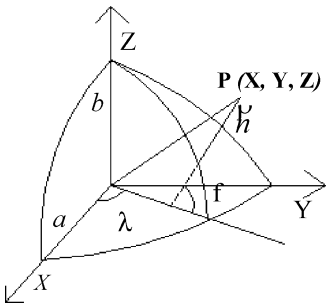

Fig. 1. Definition of the geodetic coordinate system.

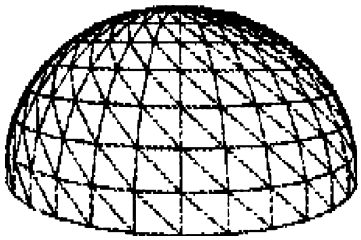

Fig. 2. Triangulation construction of the globe.

The coordinates of the original DEMs are the geodetic coordinate [8] defined by longitude, λ, latitude, φ, and height, H. Before modeling the earth, we should convert geodetic coordinates into right-handed, orthogonal coordinates. To any point (x, y, z) on the earth surface, the transformation formula of the two coordinate systems (see Fig. 1) is as follows:

$$\begin{cases} x = (N+H)\cos\varphi\cos\lambda \\ y = (N+H)\cos\varphi\sin\lambda \\ z = (N - Ne^2 + H)\sin\varphi \end{cases} \tag{1}$$

where N is the prime radius of curvature, H is the ellipsoidal height, e^2 is the first eccentricity squared, $N = a/W$, $W = \sqrt{1 - e^2\sin^2\varphi}$, $e^2 = 2f - f^2$, a, b respectively presents the semimajor radius and semiminor radius, $a = 6378.140$ km, $b = 6356.755$ km, f is the ellipsoidal flattening, $f = (a-b)/a$.

We divide the ellipsoid into two parts: one is the North and South Poles; the other is the part besides the two poles. In the second part, the normal of the patch is defined as outward from the origin. The surface is divided into bands of facets. The sides of each facet are parallel to the meridians and the parallels (see Fig. 2). There is only one point on the poles. Therefore, centered in the poles O, we construct triangles $OAB, OBC, OCD, ODE, OEF, \ldots, OHA$, in clockwise direction finally forming triangulation fans (see Fig. 3). This method takes into account order of the vertices and facilitates large terrain data simplification. The method also computes the normal vector easily.

IV. Efficient Data Access

Many methods have been proposed for indexing global geographic data, e.g., the UTM subdivision of the earth [9], sphere quadtree [10], and ellipsoidal quadtree [11]. In Ottoson's work [11], the latitude difference $\Delta\varphi$ is computed recursively from the index keys, meanwhile the latitudes, φ_n (minimum) and φ_{n+1} (maximum) can only be computed iteratively from the known index key. Thus, computing process is complex and indexing efficiency is not high. In this letter, we improve on the algorithm of the ellipsoidal quadtrees based on Ottoson's work.

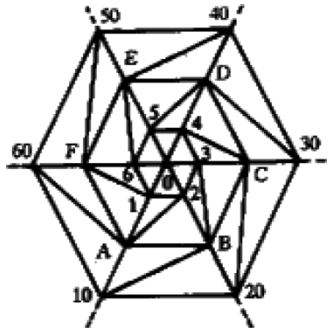

Fig. 3. Triangulation around the North (South) Pole.

Assume that $(\lambda_{\min}, \varphi_{\min})$ and $(\lambda_{\max}, \varphi_{\max})$ are, respectively, the minimum and maximum geodetic coordinates of the geographical datasets which have $M \times N$ posts. The datasets are divided into many quadrangles. Each of them has $row \times col$ posts, so the number of the quadrangles in west/east directions is $q_\lambda = \text{int}(N/col)$. The number of in north/south ones is $q_\varphi = \text{int}(M/row)$. The longitude difference $\Delta\lambda$ is $\Delta\lambda = (\lambda_{\max} - \lambda_{\min})/q_\lambda$, and latitude difference $\Delta\varphi$ is $\Delta\varphi = (\varphi_{\max} - \varphi_{\min})/q_\varphi$. The index key, for the ith level in the multiple quadtree, can be calculated from (2)

$$\text{key} = \text{int}\left(\frac{\lambda}{(2^i \cdot \Delta\lambda)}\right) + \text{int}\left(\frac{(\varphi - \varphi_{\min})}{(2^i \cdot \Delta\varphi)}\right) + \sum_{j=1}^{i}(q_\lambda \cdot q_\varphi \cdot 2^{-j}). \tag{2}$$

We also need the minimum geodetic coordinates (λ_1, φ_1) and maximum geodetic coordinates (λ_2, φ_2) of the current quadrangle from a known index key in order to construct and search the ellipsoidal quadtree. By integrating with respect to q_φ, q_λ, $\Delta\lambda$, and $\Delta\varphi$, and by using (2), the coordinates (λ_1, φ_1) and (λ_2, φ_2) can be computed

$$\begin{cases} \varphi_1 = \varphi_{\min} + \text{int}(t - q_\lambda) \cdot \Delta\varphi \\ \lambda_1 = \lambda_{\min} + \left(\left(t - \text{int}\left(\frac{t}{q_\lambda}\right)\right) \cdot q_\lambda\right) \cdot \Delta\lambda \\ \varphi_2 = \varphi_1 + \Delta\varphi \\ \lambda_2 = \lambda_1 + \Delta\lambda \end{cases} \tag{3}$$

where $t = \text{key} - \sum_{j=1}^{i}(q_\lambda \cdot q_\varphi \cdot 2^{-j})$.

To accelerate visualizing speed, only the data tiles inside the view frustum are loaded, the ones outside of the view frustum are discarded. When switching among different levels of tiles, the visual system always loads the datasets of the bound $\{fL\min, fB\min, fL\max, fB\max\}$ centered in the reference point. Fig. 4 depicts the index cell traversing the east and west hemispheres. Fig. 5 depicts the index cell only traversing one hemisphere (Western or Eastern). Steps for loading datasets are as follows.

1) Based on the viewpoint and view direction, compute the coordinates of the datasets falling inside the view frustum. Then calculate the index key of the quadtree using (2). The minimum and maximum coordinates of the quadrangles can be obtained from (3).

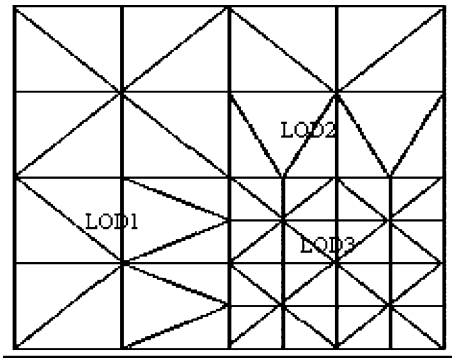

Fig. 4. Sewing the crack at the boundary at the same subdivision.

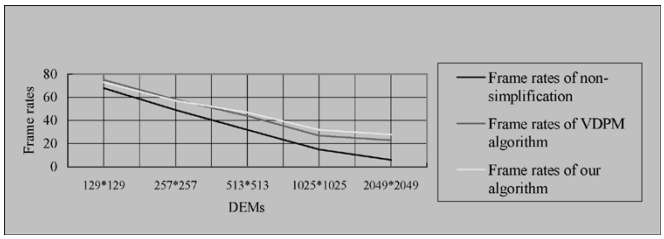

Fig. 5. Frame rates of three different methods.

2) Search the quadtrees for access to the current level of DEM and imagery datasets. Moreover, we allow for some tiles to be located in the east hemisphere, and others in the west hemisphere (see Fig. 4 and Fig. 5).
3) Map a view of the datasets into the address space of the calling process. The system determines whether the datasets have already been defined. If not, the datasets identifiers are allocated and the proper datasets are obtained. In that point, the system generates all mipmap images. The mipmap performs view-dependent textures: the highest resolution images are placed on the surfaces where are the closest to the viewer and progressively lower ones are used for further distances. Each of these stages is known as a Mip map level. MIP mapping can reduce memory bandwidth requirements and ensure that pixels do not get lost at distant places, etc.

V. Methodology for Terrain Simplification

The implementation of our process for simplifying terrain surfaces is accomplished in two basic phases. First, we partition the terrain into several square tiles based on distance to the viewpoint, and then creates filtered models ranging from a coarse to a fine approximation of each tile falling inside the view frustum by the M-band (M is an integer and $M \geq 2$) wavelet transforms. Second, a level of detail triangulation technique is applied to build view-dependent continuous, multiresolution terrain surfaces using the filtered data.

Wavelet transforms theory has spurred new interest in geosciences field, and has provided it a more rigorous mathematical framework. The standard dyadic wavelets are not suitable for analysis of high-frequency signals with relatively narrow bandwidth [12]. To overcome this disadvantage, M-band wavelet transforms are developed as a direct generalization of the two-band orthogonal wavelets of Daubechies. M-band wavelets are set of $M-1$ basis functions that scaled and translated versions form a tight frame for a set of square integrable functions defined over the set of real numbers $(L^2(R))$ [13]. They are able to zoom onto narrowband and high-frequency components of a signal, and can give a better energy compaction than two-band wavelets.

In our work, M-band wavelet transforms are applied to remove vertices from high-resolution terrain datasets, and generate a set of coarse to finer triangles depending on the viewpoint and the roughness of the terrain. We conclude that the low-frequency component $a_{j,k,l}$ of the M-band wavelet transform [14] is

$$a_{j+1,k,l} = \sum_{n1}\sum_{n2} c_{n_1-Mk} c_{n_2-Ml} a_{j,n1,n2} \tag{4}$$

where j, k, and l are integers; $n1 = 0, 1, 2, \ldots,$ rows, $n2 = 0, 1, 2, \ldots,$ cols; rows is the number of the rows of DEM, and cols is the number of the columns of DEM.

The high-frequency component of the M-band wavelet transform is

$$b_{j,k,l}^{s_1,s_2} = \begin{cases} \sum_{n1}\sum_{n2} c_{n_1-Mk} d_{n_2-Ml}^{s_2} \\ \quad \times a_{j+1,n1,n2} & s_1 = 0,\ 0 < s_2 < M \\ \sum_{n1}\sum_{n2} d_{n1-Mk}^{s_2} c_{n_2-Mk} \\ \quad \times a_{j+1,n_1,n_2} & 0 < s_1 < M,\ s_2 = 0 \\ \sum_{n1}\sum_{n2} d_{n1-Mk}^{s_1} d_{n_2-Mk}^{s_2} \\ \quad \times a_{j+1,n_1,n_2} & 0 < s_1,\ s_2 < M \end{cases} \tag{5}$$

where $\{a_{j+1,k,l}\}$ is the low-frequency portion of the $j+1$ level M-band wavelet decomposition of image $\{a_{0,k,l}\}$, and $\{b_{j+1,k,l}^{t,s}\}$ is the high-frequency portion of the $j+1$ level.

The principle of the wavelet transforms is to use wavelet filters along each row and column of the datasets to create low-frequency $\{a_{j+1,k,l}\}$ and high-frequency $\{b_{j+1,k,l}^{t,s}\}$ wavelet coefficients. The high-frequency components are stored; the low-frequency ones will continue to be divided into new high- and low-frequency components using the same filters. This process is repeated until the coarsest level is reached. We see the low-frequency $\{a_{j+1,k,l}\}$ of the $j+1$ level terrain datasets can maintain the important features of the j level data $\{a_{j,k,l}\}$. So the dataset $\{a_{j+1,k,l}\}$ is an approximation of the original data.

In the subsequent steps, the hierarchical triangulation technique uses the filtered datasets to generate multiresolution terrain. The whole terrain scene is built by a root triangulation, and then the triangulation is continuously divided until the height error is smaller than the given height error. On the one hand, the rough areas obviously have more details than those flat areas, so we choose an error called Static Error to control this merging operation. On the other hand, the regions far away from the viewpoint have fewer triangles than those close to the viewpoint. In this step, another splitting error is called Dynamic Error.

A. Crack Remedy

For flexibility, the terrain visualization system should integrate the terrain surfaces into a common coordinate system

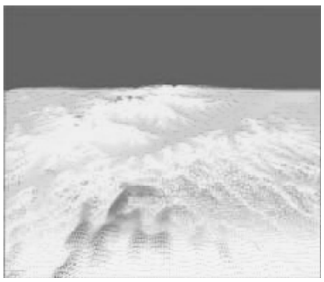

Fig. 6. Terrain models of original data (375 000 triangles, 6 fps).

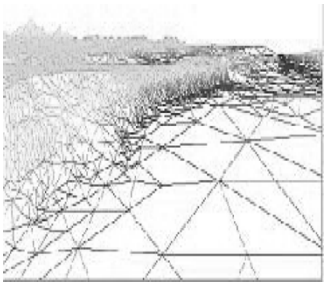

Fig. 7. Multiresolution terrain surface (12 570 triangles, 40 fps).

Fig. 8. Multiresolution terrain surface (40 893 triangles, 32 fps).

without seams or gaps. If the neighboring datasets are simplified by the same band wavelet transform, cracks are not created at the boundary of the tiles. Otherwise cracks and shading discontinuities will be generated. A common method is to sew the tiles as shown in Fig. 4.

After the above steps, a set of multiresolution terrain models is created. We compare the performance of our approach with that of VDPM presented by Hoppe [2] with use of many DEMs under the same environment conditions. As evidenced by Fig. 9, our method sustains a minimum of 26 frames per second (fps), varying with the number of the polygons rendered in view frustum. The efficiency of our algorithm is higher than VDPM algorithm.

Figs. 6–9 show the triangulation models, triangle count, and frame rates in the view frustum.

Table I shows time to download the data from the server, the number of triangles rendered per frame, and frame rates on each client. The table indicates that frame rates of our method exceed 26 frames per second with only a minor loss in image quality. Figs. 10–13 show the global scene.

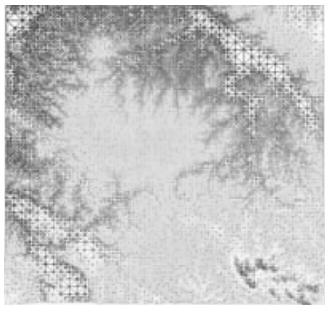

Fig. 9. Multiresolution terrain surface (64 905 triangles, 27 fps).

TABLE I
TEST RESULTS OF SEVERAL DIFFERENT RESOLUTION GEOGRAPHIC DATA DOWNLOADED FROM THE SERVER AND VISUALIZATION ON THE CLIENT

Frame rate (fps)	Transmission time(s)	Triangle number per frame
30.8	0.9	50128
28.5	1.1	56621
26.8	1.3	60155

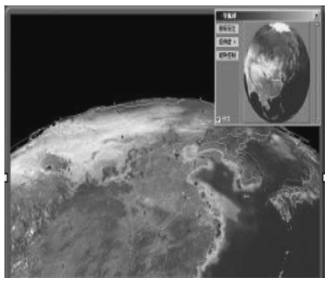

Fig. 10. Three-dimensional visualization of the earth.

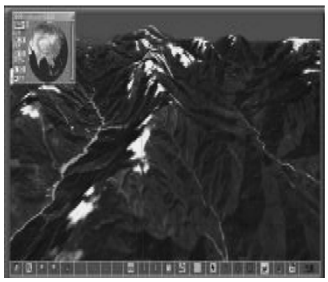

Fig. 11. Visualization of Himalayas area.

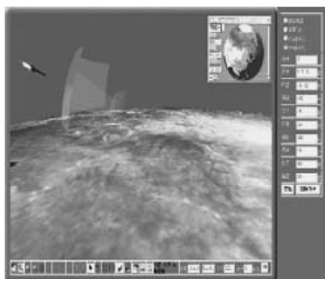

Fig. 12. Simulation of a fight battle.

VI. EXPERIMENTAL RESULTS

Our application has been implemented in VC++ and OpenGL, and integrated into a COM component. JavaScript together with HTML are used for implementing the graphical

Fig. 13. View of Beijing city data with 3-D models.

user interface. An element of the 3-D scene contains multiple different data types such as DEMs, textures, vector datasets, symbols, and 3-D geometric models. The application renders different levels of datasets in light of the system load condition, the user's view position, and capabilities of the computers. It provides an integrated environment for spatial data visualization, analysis, and interaction.

Performance measurements are made on a computer with an Intel 800-MHz Pentium III processor, 512 MB RAM, and TNT2 M64 graphic card under Microsoft Window 2000. DEMs consist of GTOPO30 (http://edcdaac.usgs.gov/gtopo30/gtopo30.asp) that is a global DEM with a horizontal grid spacing of 30 arc-seconds (about 1 km), and the elevation data of Yangtze River region with a scale of 1 : 10 000. Imagery datasets include "blue marble" image containing Western Hemisphere (21 600 wide $\times$ 21 600 high) and Eastern Hemisphere (21 600 wide $\times$ 21 600 high), and TM images of the whole China with resolution is 30 m.

VII. Conclusion and Future Work

This letter has depicted certain effective methods for building the 3-D global scene. The experimental results indicate that the approaches proposed allow for fast access to huge datasets, and real-time visualizing global multiresolution geographical data. The ellipsoidal model is also accurate.

The application is an ongoing project with more upcoming contents and features. In the future work, 3-D models database will be expanded. We also plan to use lifting-schemes for M-band wavelet decomposition to accelerate display speed, and to interactively render the 3-D scenes at even higher frame rates with no cracks or thin triangles. We will also enhance capabilities such as editing and analysis functions of complex environments online.

References

[1] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner, "Real-time continuous level of detail rendering of height fields," in *Proc. ACM SIGGRAPH Conf.*, Aug. 1996, pp. 109–118.

[2] H. Hoppe, "Smooth view-dependent level-of-detail control and its application to terrain rendering," in *Proc. IEEE Visualization Conf.*, 1998, pp. 35–42.

[3] D. E. McArthur, R. Fuentes, and V. Devarajan, "Generation of hierarchical multiresolution terrain databases using wavelet filtering," *Photogramm. Eng. Remote Sens.*, vol. 66, no. 3, pp. 287–295, Mar. 2000.

[4] P. Lindstrom and V. Pascucci, "Terrain simplification simplified: A general framework for view-dependent out-of-core visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 3, pp. 239–254, Jul.-Sep. 2002.

[5] M. Guthe and R. Klein, "Streaming HLODs: An out-of-core viewer for network visualization of huge polygon models," *Comput. Graph.*, vol. 28, pp. 43–50, 2004.

[6] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "Planet-sized batched dynamic adaptive meshes (P-BDAM)," in *Proc. IEEE Visualization Conf.*, 2003, pp. 147–155.

[7] M. Roth, G. Vossb, and D. Reiners, "Multi-threading and clustering for scene graph systems," *Comput. Graph.*, vol. 28, pp. 63–66, 2004.

[8] I. Ussisoo, "Map projections," Nat. Land Survey Sweden, Gävle, Sweden, Professional Papers LMV-Rep. 1977:6, 1977.

[9] M. Mark and J. P. Lauzon, "Approaches for quadtree-based geographic information systems at continental or global scales," in *Proc. Auto-Carto 7*, Bethesda, MD, 1985, pp. 355–365.

[10] G. Fekete, "Rendering and managing spherical data with sphere quadtrees," in *Proc. IEEE Visualization Conf.*, 1990, pp. 176–186.

[11] P. Ottoson and H. Hauska, "Ellipsoidal quadtrees for indexing of global geographical data," *Int. J. Geograph. Inf. Sci.*, vol. 16, pp. 213–226, 2002.

[12] J. T. Bjørke and S. Nilsen, "Wavelets applied to simplification of digital terrain models," *Int. J. Geograph. Inf. Sci.*, vol. 17, pp. 475–501, 2003.

[13] R. A. Gopinath, J. E. Odegard, and C. S. Burrus, "Optimal wavelet representation and of signals and the wavelet sampling theorem," *IEEE Trans. Circuits Syst., II: Analog Digit. Signal Process.*, vol. 41, no. 4, pp. 262–277, Apr. 1994.

[14] G. Wan and C. Q. Zhu, "Application of multi-band wavelet on simplifying DEM with lose of feature information," *Acta Geodaetic. Cartograph. Sinica*, vol. 28, pp. 36–40, 1999.