Organization and Management of Massive Terrain Data Using Block Quadtree

Yang Ying *

LiRen College
Yanshan University
Qinhuangdao, Hebei, 066004, China
ysu_yangying@163.com

***Abstract* — To alleviate the impact on terrain rendering caused by real-time scheduling deformation data, we presented an organization approach of massive terrain data based on block quadtree and Peano encoding non-pointer data index. First, we constructed block quadtree and reorganize blocks according to Peano filling curve, which could ensure location and continuity of block data and improve the speed of terrain rendering. Secondly, non-pointer data index based on block quadtree and Peano encoding was designed to realize the exact orientation of terrain data block. In addition, we adopt GPU minimum box visible detection and terrain data prediction strategy to realize dynamical management and real-time scheduling of terrain data. The experiment results show that the algorithm can improve the efficiency of data scheduling and achieve good performance for massive terrain rendering.**

Keywords - terrain data; massive; block quadtree; non-pointer; data index.

I. INTRODUCTION

Real-time visualization of massive terrain data is an important research topic in computer graphics. It has been widely used in the geographical information system, virtual reality and military game disaster scene simulation. With the development of the technology of sensing satellite, the amount of data based on regular grid terrain has been up to hundreds of megabytes or gigabytes. The contradiction between the scale of massive terrain data and the real-time processing ability of the computer hardware is the main bottleneck of real-time rendering of large-scale terrain. When large scale terrain scene is roaming, the visual area is only a small part of the whole terrain data. Therefore, how to design an effective data organization and realize the real-time scheduling of the visible data is of great significance to the research of real-time visualization of massive terrain data.

In decades, the domestic and foreign scholars have carried out a lot of research on the organization and management of terrain data. Early algorithms based on fine-grained level of detail (LOD)[1], which focused on the reduction of the triangles. With the development of graphics hardware, Graphics Processing Unit (GPU) has been able to deal with a large number of triangles at one time. But the transfer speed of data (especially between external storage and main memory) did not receive the corresponding development. It is difficult to adapt GPU high-speed throughput capacity. GPU is often in a wait state. Therefore the organization management of massive terrain data based on GPU has become the main research hotspot in the current large scale terrain rendering.

In the massive data organization, the predecessors have done a lot of work. Peter Lindstrom designed the interleaved quadtrees[1] on CPU to organize terrain data .The interleaved quadtrees exists data redundancy. With GPU become faster and more powerful, coarse-grained LOD, which processes a set of triangles as a simplified primitive, has replaced fine-grained LOD. Since GPU accelerated, coarse-grained LOD method achieves high performance, and is suitable for large-scale terrain rendering. Block quadtree has became the mainstream method based on GPU coarse-grained LOD with the characteristics of grid LOD and texture LOD corresponding each other, data index simply. In the block data organization, Han presented a Morton compressing method in terrain data block partition based on the surface roughness[3] in order to reduce the amount of reading DEM data and improve the compression speed. Zheng designed block-wise parallel integer wavelet transformation and SPIHT algorithms with CUDA[4] in order to achieve the high compression ratio. The above algorithms are more complicated. For accessing large data sets, Tetsuo Asano presented data layouts based on space filling curves[5] used to guarantee good geometric locality. M. Bader picked out two simple examples both related to data structures and algorithms on computational grids[6] to introduce typical data structures and related properties and problems that can be tackled by using space-filling curves. Tomáš Ježowicz selected space-filling curves to the speed up the original Fruchterman-Reingold graph layout algorithm[7] by computing repulsive forces only between vertices that are near each other. N. Albuquerquea used Peano filling curves[8] to analyze data. Current, many mobile maps and other location softwares use Peano filling curve data for encoding. In this paper, we construct block quadtree, use Peano filling curve to organize the terrain block data based on block quadtree, and design a non-pointer data index for terrain rendering.

In the massive data management, Li constructed the incremental horizon dynamically[9] according to the pre-extracted potential silhouette of each chunk in order to

reduce the out-of-core paging. Peter Lindstrom built in paging mechanisms of the operating system to by virtue of dramatic improvements in multilevel cache coherence[1]. The traditional method is that the intersection between cone of viewpoint and terrain surface has been adopt to the judgement of block visibility on CPU, which can increase CPU computational burden. Liu presented the judgement of block visualization on GPU[10], computed the geographic coordinates of the current scene on GPU, analyzed the visible region through the coordinates. In this paper, we adopt data prefetching strategy based on viewpoint shifting and the visible judgement of terrain block based on GPU minimum box.

II. MASSIVE TERRAIN DATA ORGANIZATION

The size of massive terrain data can reach hundreds of megabytes or gigabytes. It is impossible for loading the whole terrain data sets at one time. As a means to combat this problem, it is necessary to design terrain data organization and management. Block quadtree has became the mainstream method based on GPU coarse-grained LOD with the characteristics of grid LOD and texture LOD corresponding each other, data index simply and selection quickly. In this paper, block quadtree is constructed in order to optimize the tree modeling and improve the rendering speed.

Block Quadtree Construction. The regular grid terrain data is divided into blocks. The block quadtree is constructed. The whole terrain is divided into several levels. The root node of block quadtree represents a lowest resolution of the whole terrain. The size of the lower layer is a quarter the size of the upper layer. The lower layer is a higher resolution representation of the upper layer. Quarter the root node into four children nodes, northwest (NW), northeast(NE), southwest(SW) and southeast(SE). An example of block quadtree of four levels is shown in Figure. 1.

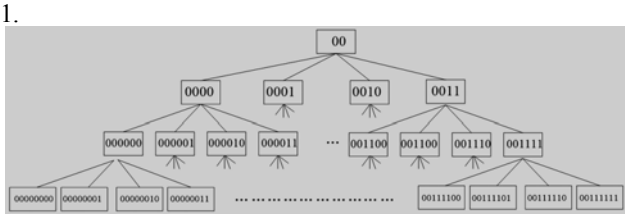

Figure. 1 Block Quadtree Construction.

In this paper, the whole terrain data is organized by block quadtree and rendered by the coarse-grained LOD, which can meet the requirement of real-time rendering of massive terrain data in a certain degree. Through quartering the terrain data, we can improve the search speed of terrain rendering. The hierarchical continuity and local continuity of the block data in block quadtree can reduce the number of data between external storage and main memory, so as to improve the scheduling efficiency of the data.

Block Data Organization By Peano filling curve. Current, many mobile maps and other location software use Peano filling curve data for encoding. In this paper, we use Peano filling curve to organize the terrain block data. According to the order of the Peano fill curve, reorganize blocks of block quadtree. The encoding regulation of blocks data in block quadtree is defined as follow. The code of the root of the block quadtree is 00. The codes of the four children of the root are defined, in this order: SW is 0000, NW is 0001, SE is 0010, NE is 0011. The codes of the four children of the note(code:0010) are defined, in this order: SW is 001000, NW is 001001, SE is 001010, NE is 001011. When the level is added 1, the codes of the children are added two bits binary encoding after their fathers. The added two bits binary encoding are defined in the same order: SW is 00, NE is 01, SW is 10, SE is 11. Repeat the steps until generate the left nodes of the block quadtree. The codes of block quadtree of four levels is shown in Figure 1.Set the number of level of block quadtree from the beginning of 1, we can get the level of the block quadtree according to encoding's length, level = the length of encoding/2. Every block in block quadtree has own code. An example of codes and organization of level is 2, 3, 4 of block quadtree is shown in Figure. 2.

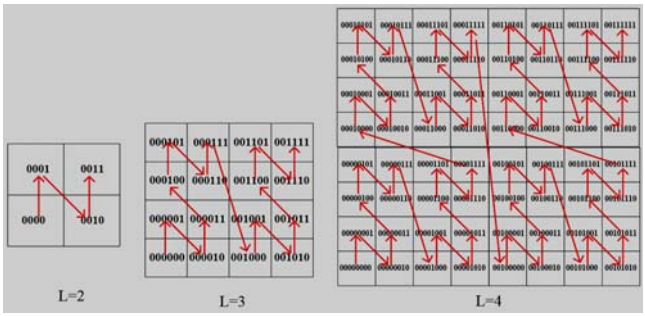

Figure. 2 Block Data Organization using Peano Filling Curve.

Block Non-pointer Data Index Computation. Reorganize blocks according to Peano filling curve could meet the physical address of the adjacent data with the geometry of local continuity. Terrain block boundary coordinates information can be needed in terrain rendering. How to exactly orientate the terrain block boundary coordinates according to Peano filling curve is an important study in terrain rendering. It is necessary to design an efficient data index computation.

Set the left lower corner point of the terrain block as the origin of the coordinates as shown in Figure. 3.

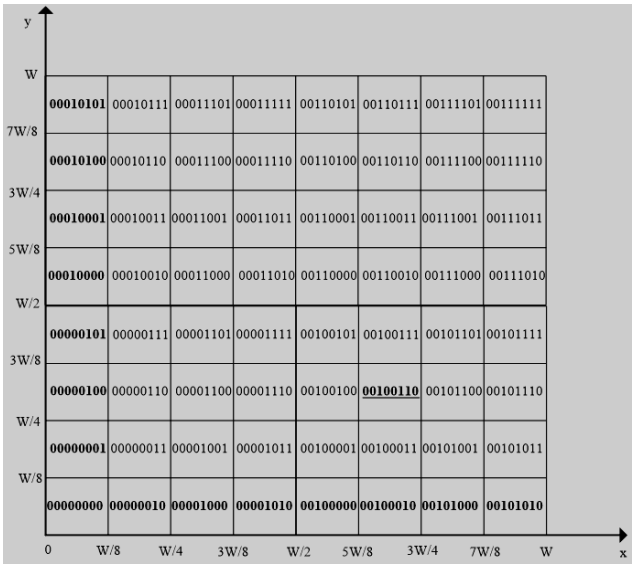

Figure. 3 l=4 Block Data Organization using Peano Space Filling Curve.

The length of the whole terrain is W. Set the level of block quadtree as l. So the length of each terrain block is shown in Eq. 1.

$$block_l = W >> (l-1) \tag{1}$$

Design a non-pointer data index according to Peano filling curve encoding. The law obtained by Peano filling curve is used to compute the terrain block boundary coordinates. The terrain block boundary coordinates contain left boundary, right boundary, bottom boundary, top boundary. The expression of right boundary and left boundary is shown in Eq. 2. The expression of top boundary and bottom boundary is shown Eq. 3.

$$right = left + block_L \tag{2}$$

$$top = bottom + block_L \tag{3}$$

L is the level of block quadtree in Eq. 2 and f Eq. 3. Therefore, if left boundary and right boundary has been computed, the four boundary coordinates can be obtained.

Firstly, compute the left boundary coordinate. As an example, extract the first line of encoding in Fig. 3. The relationship between encoding and horizontal coordinate is shown in TABLE I.

TABLE I THE RELATIONSHIP BETWEEN ENCODING AND HORIZONTAL COORDINATE

encoding	00000000	00000010	00001000	00001010	00100000	00100010	00101000	00101010
coordinate	0	W/8	W/4	3W/8	W/2	5W/8	3W/4	7W/8

As seen from Table 1, if the second bit from the right is 1, the length can represent W/8. If the forth bit from the right is 1, the length can represent W/4. If the sixth bit from the right is 1, the length can represent W/2. When computing the data index, it is only necessary to extract the data from the various even bits of encoding. The left boundary coordinate can be obtained by accumulating the length of even bits of encoding represented.

Define block quadtree encoding as ZINDEX. The second bit from the right is obtained by right shifting one bit, ZINDEX>>1. The forth bit from the right is obtained by right shifting three bits, ZINDEX>>3.The sixth bit from the right is obtained by right shifting five bits, ZINDEX>>5.

When the level is l, the number of encoding bits can reach 2*l. Define $level_l = l-1$. If the current level is L, each even bit of encoding can be obtained by Eq. 4.

$$Zindexx_l = ZINDEX >> \lfloor 2\times(L - level_l) + 1 \rfloor \& 1 \tag{4}$$

l=1,2,…,L in Eq. 4.

The left boundary coordinate of the block is expressed in Eq. 5.

$$BLOCK.left = \sum_{l=1}^{L} block_l (Zindexx_l \& 1) \tag{5}$$

The right boundary coordinate of the block is expressed in Eq. 6.

$$BLOCK.right = \sum_{l=1}^{L} block_l (Zindexx_l \& 1) + block_L \tag{6}$$

Secondly, in the same way, compute the bottom boundary coordinate. As an example, extract the first column of encoding in Fig. 3. The relationship between encoding and longitudinal coordinates is shown in TABLE II.

TABLE II THE RELATIONSHIP BETWEEN ENCODING AND LONGITUDINAL COORDINATES

encoding	00000000	00000001	00000100	00000101	00010000	00010001	00010100	00010101
coordinate	0	W/8	W/4	3W/8	W/2	5W/8	3W/4	7W/8

As seen from Table 2, if the first bit from the right is 1, the length can represent W/8. If the third bit from the right is 1, the length can represent W/4. If the fifth bit from the right is 1, the length can represent W/2. When computing the data index, it is only necessary to extract the data from the various odd bits of encoding. The left boundary coordinate can be obtained by accumulating the length of odd bits of encoding represented.

Define block quadtree encoding as ZINDEX. The first bit from the right is obtained by right shifting zero bit, ZINDEX>>0. The third bit from the right is obtained by right shifting two bits, ZINDEX>>2. The fifth bit from the right is obtained by right shifting four bits, ZINDEX>>4.

If the current level is L, each odd bit of encoding can be obtained by Eq. 7.

$$Zindexy_l = ZINDEX >> \lfloor 2\times(L - level_l) \rfloor \& 1 \tag{7}$$

l=1,2,…,L in Eq. 7.

The bottom boundary coordinate of the block is expressed in Eq. 8.

$$BLOCK.bottom = \sum_{l=1}^{L} block_l (Zindexy_l \& 1) \tag{8}$$

The top boundary coordinate of the block is expressed in Eq. 9.

$$BLOCK.top = \sum_{l=1}^{L} block_l(Zindexy_l \& 1) + block_L \quad (9)$$

The computation of the four boundary coordinates has been completed in the preprocessing. The boundary coordinates are used to CPU selects the visible blocks, when visible region coordinates obtain by GPU. Reorganize blocks according to Peano filling curve could enhance data access locality and improve blocks' query efficiency. The non-pointer data index only needs the block encoding to realize exact orientation of the four boundary coordinates, in order to achieve the partial reading of the whole terrain data and simplify the operation of massive data. The computation of boundary coordinate, according to Peano encoding, contains only additive and bit operations, the computational complexity is not high.

III. MASSIVE TERRAIN DATA DYNAMIC MANAGEMENT

In the process of terrain rendering based on view point, it is necessary to adjust the main memory data in real-time. In this paper, we study the dynamic management of massive terrain data based on GPU visibility judgment and data prefetching strategy.

GPU Visibility Judgment. When terrain is rendering, only visible terrain areas are needed to enter the graphics pipeline. Visible terrain data is only a small part of the entire terrain data. How to judge the visual data is an important problem in dynamic management and real-time scheduling of massive terrain data.

In the traditional method, simplify the CPU view of the body as a column. The bottom surface of the column is a triangular. When visibility is judged, project of the column and intersect with the horizontal plane. This will cause a lot of stress on the CPU's calculation.

In this paper, we judge block visualization on GPU, similar to [10]. We use "Render To Texture" to map every terrain block into a texture on GPU. So the texture can save every block's coordinates(x, y). Then put the texture read back to CPU. Calculate the visible region through the cycle of judgment of minimum value of geographical coordinates. According to principle of minimum bounding box, the minimum box can judge terrain visible region is shown in Figure. 4. The minimum box, which represents the visible region, is made up four coordinates (min x, min y), (max x, min y), (max x, max y), (min x, max y).

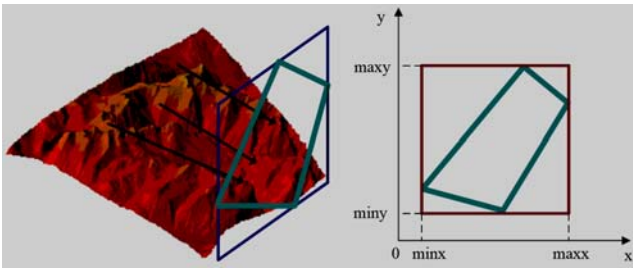

Figure. 4. Visual Region Judgement.

The visibility judgement by GPU the minimum box makes full use of GPU and reduces the CPU computation burden. Because the terrain frustum culling rendering, redundant data are not drawn, which can not affect the speed of terrain rendering.

Data Prefetching Strategy. We use data prediction method to solve unsmooth problem caused by transferring the deformation visible data from the external storage in the terrain roaming. According to the position of the viewpoint and the change direction of eye sight in the terrain roam, predict the location of the point of view, and determine whether import the deformation visible data into main memory or not. Schedule the possible data according to the forecast position can reduce the waiting time to load the deformation data, and ensure the terrain smooth roaming in a certain extent.

GPU Terrain Rendering Baesd on Block Quadtree. In terrain rendering, we make full use of the characteristics of GPU multi-channel parallel processing and texture mapping. Only load visible block quadtree from CPU to GPU, and perform two-dimensional texture mapping and error metric computation on GPU to realize GPU terrain real-time rendering. All blocks in visible block quadtree can be seen as text to compute on GPU. In this article, the R channel respectively stores the level of the whole block quadtree, the G and B channels respectively store coordinate BLOCK left and BLOCK bottom. The level, BLOCK left and BLOCK bottom. could be used to compute the orientation of the block. Visible terrain data orientation, visible terrain data scheduling and texture mapping have been adopt to realize GPU terrain rendering based on block quadtree.

IV. RESULTS AND DISCUSSION

According to the above algorithms and principles, we use a data set over the Puget Sound area in Washington provided by Peter Lindstrom personal home page, which is made up of 16385*16385 vertexes at 10 meter horizontal and 0.1 meter vertical resolution, 650M. The length of each block is 513. We use a 3.0GHz Intel Pentium IV PC, with 2G DDR2 of RAM, GeForce 9500GT graphics with 512M of graphics RAM, and SATA 500G disk in our experiments. We implement prototype program using Windows XP32 and Visual Studio C++ 6.0, OpenGL and OpenGL Shading Language (GLSL). The prototype program is used to verify effectiveness of the proposed algorithm in this article.

In order to test the block quadtree data organization and Peano non-pointer data index, we organize the terrain data as the following three forms: CPU interleaved quadtrees[1], GPU block row column linear organization, GPU block quadtree Peano organization (GPU block quadtree Peano non-pointer data index organization in this paper). Statistic access data number per frame of different data organizations in data scheduling, as shown in Fig. 5.

As seen from Figure. 5, the data scheduling efficiency on GPU is higher than the data scheduling efficiency on CPU. The data scheduling efficiency of GPU block quadtree Peano organization is highest. The access data number per frame is about 0.38M. Compared to CPU interleaved quadtrees, the access data number per frame of our algorithm can reduce

53.1%. The algorithm in this paper can improve the speed of terrain rendering efficiently.

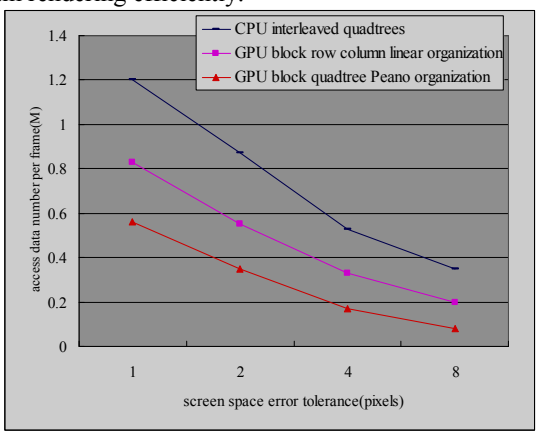

Figure. 5. access data number of different data organizations.

Statistic scheduling data number and scheduling time of our algorithm based on GPU block quadtree Peano organization in this paper, as shown in TABLE III.

TABLE III STATISTICAL TABLE SCHEDULING DATA NUMBER AND SCHEDULING TIME

screen space error tolerance (pixels)	initial scheduling data number (M)	real-time scheduling data number (M)	scheduling time (ms)
1	1.8	0.56	2.7
2	1.2	0.35	2.1
4	1.1	0.17	1.6
8	0.9	0.08	1.1

As seen from Table 3, the initial scheduling data number and real-time scheduling data number of our algorithm are relatively low. The scheduling time of our algorithm is between 1ms and 2.7ms. Compared to Li[9], whose access time one time is about 2.3ms, our algorithm could meet the requirement of real-time rendering of massive terrain data rendering.

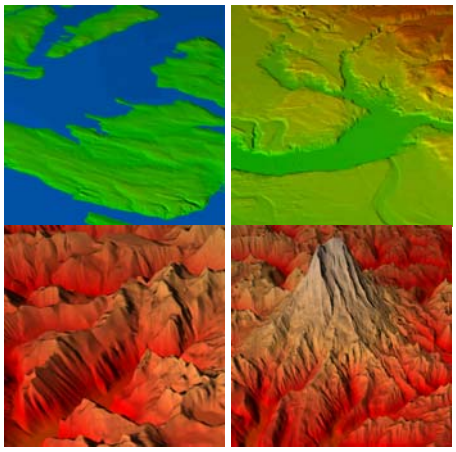

Figure. 6. screenshots of terrain textures.

Real-time rendering screenshots of terrain textures are shown as Figure. 6.

Experiments' results can verify that our algorithm can greatly improve the speed of the terrain rendering and guarantee terrain rendering smoothly.

V. CONCLUSIONS

According to massive terrain data organization and management, block quadtree is constructed to reorganize terrain data blocks, which could meet hierarchical continuity and local continuity of terrain data and improve the scheduling efficiency of the data. Reorganize blocks Non-pointer data index according to Peano filling curve could realize exact orientation of the four boundary coordinates and improve blocks' query efficiency. GPU the minimum box data prefetching strategy could access speedily visible data and reduce access data number per frame. Experiments prove that our algorithm can improve the speed of the terrain rendering and achieve a good visual effect. Future work includes terrain error metric and researching eliminating cracks efficiently to meet the massive complexity of terrain rendering.

ACKNOWLEDGEMENTS

This work is supported by The National Natural Science Foundation, China (No.60970073), HeBei National Natural Science Foundation, China(No.F2012203084), HeBei National Natural Science Foundation, China (No.A2012203124).

REFERENCES

[1] E. P.Lindstrom, V.Pascucci. Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization, IEEE Transaction on Visualization and Computer Graphics, 2002, 8, 3, p.p. 239-254.

[2] H.Y. Kang, H. Jang, C.S. Cho, J.H. Han. Multi-resolution terrain rendering with GPU tessellation, The Visual Computer, 2014, 31, 4, p.p. 455-469.

[3] Han M in, Chen Hongbo, Zheng Danchen. Terrain Simplification Algorithm Based on Morton Compressing in Block and Hybrid Generating Rule, Journal of Computer-Aided Design& Computer Graphics, 2014, 26, 2, p.p. 293-301.

[4] Zheng Xin, Liu Wei, Lu Chenlei, Guo Ping. Real-Time Dynamic Storing and Rendering of Massive Terrain with GPU, Journal of Computer-Aided Design& Computer Graphics, 2013, 25, 8, p.p.1146-1152.

[5] Tetsuo Asano, Desh Ranjan, Thomas Roos, Emo Welzl, Peter Widmayer. Space-filling curves and their use in the design of geometric data structures, Theoretical Computer Science, 1995, 181, 96, p.p. 3-15.

[6] M. Bader. Two motivating examples: sequential orders on quadtrees and multidimensional data structures, An Introduction with Applications in Scientific Computing, 2013, 9, p.p. 1-14.

[7] Tomáš Ježowicz , Petr Gajdoš, Eliška Ochodková, Václav Snášel. A New Iterative Approach for Finding Nearest Neighbors Using Space-Filling Curves for Fast Graphs Visualization, International Joint Conference SOCO'14-CISIS'14-ICEUTE'14, Bilbao, Spain, 2014, pp.11-20.

[8] N. Albuquerquea, L. Bernal-Gonzálezb, D. Pellegrinoa, J.B. Seoane-Sepúlvedac. Peano curves on topological vector spaces, Linear Algebra and its Applications, 2014, 460, p.p.81–96.

[9] S. Li, J. Ji, X. Liu, E.H. Wu, "High performance navigation of very large-scale terrain environment," Journal of Software, 2006, 17, 3, p.p. 535-545.

[10] Sung-Eui Yoon, Peter Lindstrom. Mesh Layouts for Block-Based Caches, IEEE Transaction on Visualization and Computer Graphics, 2006,12,5p.p.1213-1220.

[11] H. Liu, W. Cao, and W. Zhao, "Dynamic scheduling and real-time rendering method for large-scale terrain," Compute Science, 2013, 40, 6, p.p. 120-124.