A Focus Control Method Based on City Blocks for the Focus+Glue+Context Map

Daisuke YAMAMOTO Kohei HUKUHARA Naohisa TAKAHASHI
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555, Japan
E-mail: {daisuke,naohisa}@nitech.ac.jp

Abstract

In order to show both a detailed map (Focus) and a wide-area map (Context) in one map, we have proposed the Focus+Glue+Context map, a type of fisheye view for mobile maps. A problem in the Focus+Glue+Context map is that if the radius of the Focus is large, the Context area is very narrow, and if the radius of the Focus is small, we cannot show the entire target area. To solve this problem, we have to determine the radius of the Focus according to the shape of the target area. In this work, we have proposed the Expand Loop Road algorithm to controls the radius and position of the Focus by city block. Moreover, we applied this method to a Focus+Glue+Context map service. Although the conventional method, which controls the radius of the Focus by a constant scale, has to operate 7.0 times to achieve 80% accuracy, the proposed method operates only 1.07 times to achieve 82.6% accuracy. In the future, the proposed method will enable mobile maps with fisheye views to be controlled easily based on road networks.

1 Introduction

In recent years, advanced Web Map Services (WMSs) such as Google Maps and Yahoo! Maps have become available online. These services can be accessed from not only PCs but also mobile terminals such as cellular phones. When users want to search multiple areas by using existing WMSs, they have to switch between multiple maps with different scales and visualize geographical relations between these maps. Such operations lead to a large cognitive cost for users.

Previous studies [4][7][8][3][1] have proposed a Focus+Context-type fisheye view map that shows both the detailed map (Focus) and the wide-area map (Context) in one map to solve the abovementioned problem. However, since existing fisheye view methods generate the entire map dynamically by using a displacement function, the calculation and communication costs are very high. Therefore, it is difficult to apply these methods to mobile WMSs.

In this light, we have proposed a Focus+Glue+Context map [10] (hereafter referred to as F+G+C map); this is the world's first fisheye view map for WMSs and mobile terminals. This method is much faster because it limits the area that needs to be calculated dynamically. In addition, we proposed an interface based on a posture sensor for use with mobile terminals [11].

Since the F+G+C map has six degrees of freedom, users have to independently control the position (x- and y-axes), size, and scale of the Focus; width of the Glue; and scale of the Context. These operations are particularly difficult when using mobile terminals because these are not equipped with a mouse. Moreover, the F+G+C map has a problem in that if the radius of the Focus is large, the Context area is too narrow, and if the radius of the Focus is small, we cannot view the entire target area. Therefore, a method that can control the radius and center position of the Focus automatically in a manner similar to the autofocus function of a still camera is required. We proposed the *Expand Loop Road algorithm* to control the radius and center position of the Focus automatically based on the roads in the target area. This method enables users to control the Focus with minimal operations and high accuracy.

This method is only the first step toward controlling F+G+C maps based on road networks. In the future, we intend to contribute toward the popularization of fisheye view maps in mobile terminals by improving this method.

2 Focus+Glue+Context

2.1 Focus+Glue+Context maps

We proposed the *Emma* system [9] to enhance cognitive maps by expressing the human's geographical image as the components of *the city image* [5], such as districts, landmarks, and paths. Emma enables users to transform and control a map based on cognitive maps. We proposed the F+G+C map that has a *Focus*, a *Glue*, and a *Context* and made it available publicly as a WMS[1] [10]. As shown in

[1] http://joint.alpslab.jp/fisheye/

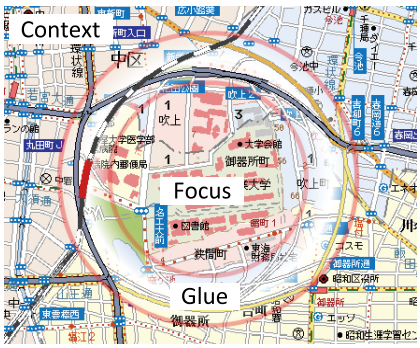

Figure 1. Focus+Glue+Context map.

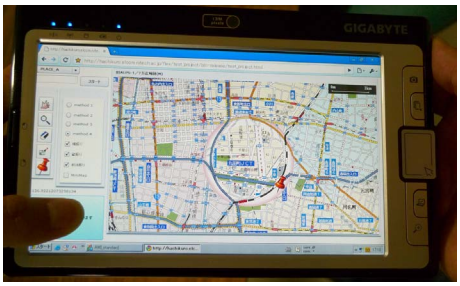

Figure 2. Focus+Glue+Context map system for mobile terminals.

Figure 1, the Focus is an area of a large-scale map that enables users to understand details about the focused area; the Context is an area of a small-scale map that enables users to understand geographical relations; and the Glue shows the routes that connect the Focus with the Context. Unlike existing fisheye views, in the F+G+C map, the Focus and Context have no distortion because the Glue contains all the distortion. Then, the Glue is compressed considerably when moving from the Focus to the Context. Major roads, rails, and the roads that run from an area in the Focus into an area in the Context are drawn selectively in order to reduce the density of roads in the Glue. The F+G+C maps draw only the Glue dynamically. Moreover, in the F+G+C map, the calculation cost is lower than that in existing fisheye views for maps, in which the entire region has to be transformed. Although the Glue must be dynamically generated according to its shape, the advantage of using the Focus and the Context is that dynamic generation is not required. Therefore, by generating the Focus and Context maps in advance, we can generate the F+G+C map in real-time. In fact, the F+G+C method is up to 12 times faster than conventional methods [10].

In addition, we developed an interface based on a posture sensor for use with mobile terminals by taking advantage of these abovementioned characteristics [11], as shown in Figure 2. This system enables users to control the F+G+C maps easily by tilting and shaking the terminal with one hand.

2.2 Problem of determining size of Focus

In the F+G+C map, if the radius of the Focus is smaller than the target area (parks in this case), the entire target area cannot be shown in the Focus, as shown in Figure 3-a. In this case, users cannot view the entire target area, and some roads connected from the Focus to the Context cannot be drawn in the Glue. Conversely, if the radius of the Focus is larger than the target area, as shown in Figure3-b, the Context becomes narrow and the density of roads in the Glue increases. Therefore, we have to adjust the radius of the Focus properly according to the outline of the target area, as shown in Figure 3-c.

If the outline of the target area (for example, landmarks and shopping area) can be acquired, in general, we can easily determine the radius and position of the circle-type Focus that touches this target. However, because many landmarks are associated with a point on the map, we cannot always acquire their outline and/or size. Moreover, some landmarks such as a university and a shopping area consist of some city blocks. Because we have to manually adjust the size of the Focus according to the target, this is particularly difficult when using mobile terminals because these are not equipped with a mouse.

Our previous study [11] adopted a simple method to solve this problem. In this method, the radius of the Focus is enlarged at a constant rate when users shake the terminal. For example, when a user shakes the terminal to the left once, the radius of the Focus increases by $x\%$. When a user shakes the terminal to the right once, the radius of the Focus decreases by $x\%$. Although we have to define x to be as small as possible to enable the Focus to be adjusted precisely, this requires the size to be controlled many times.

Therefore, we propose the Loop Road algorithm to adjust the size of the Focus based on city blocks by using road networks. In general, target areas such as parks and shopping areas are often delimited by streets. Therefore, if we can extract a city block as a small area delimited by streets, we can extract the outline of the target area. Moreover, we propose the Expand Loop Road algorithm to expand city blocks one-by-one in order to deal with targets that consist of multiple contiguous city blocks.

3 Algorithm

In this section, we describe the Expand Loop Road algorithm. First, we propose the Loop Road algorithm to extract a loop road that surrounds the city block that contains the target point. Next, we propose the Expand Loop Road algorithm to extract the roads that surround multiple contiguous city blocks by expanding city blocks one-by-one. By re-

Figure 3. Problem of determining the size of the Focus. a) Because the Focus is very small, we cannot understand the entire target area. b) Because the Focus is very large, the Context and the Glue are very narrow. c) The Focus fits the target area properly.

peating the Expand Loop Road algorithm, we can extract loop roads of any size.

Let us define some terms. A *city block* is the smallest area that is surrounded by streets. A *loop road* is a series of roads that surrounds a city block, as shown in Figure 4. A *node* is an intersection. A *link* is a road that connects two nodes without another intersection. A link may include some curves and turning points.

3.1 Loop Road algorithm

We propose the Loop Road algorithm to extract a loop road surrounding point C. Here, we remove orphan links such as straight roads that do not form a loop road beforehand.

First, we search the nearest link from point C. Then, we follow the link in a counterclockwise direction. When the link connects to the tail of the first link, the path consisting of these links is a loop road. However, when the link comes to a dead end, we follow the neighboring link by the depth-first search algorithm, as shown in Figure 4.

Algorithm 1 shows the Loop Road algorithm in the counterclockwise direction. Here, *start* and *goal* are the front and tail nodes, respectively, of the nearest link from point C. *setVisitedLink(node1, node2)* is a function that sets a *visited flag* to the link that consists of node1 and node2. *node.UnvisitedLeftChild()* is a function that returns the unvisited and leftmost link that is connected from the *node*. In the case of Figure 5, when node B is the current node and all links (P1, P2, P3) are not set to visited flags, this function returns node P3. In the same manner, *node.UnvisitedRightChild()* is a function that returns the unvisited and rightmost link that is connected from the current node.

However, in the case of Figure 6, we cannot extract a correct loop road by using this algorithm. Then, if the loop road cannot surround point C, we delete the loop road, and retry the Loop Road algorithm. After the execution of this algorithm, the route that consists of the node stored in the Stack is a loop road.

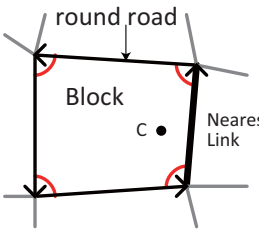

Figure 4. An example of a loop road.

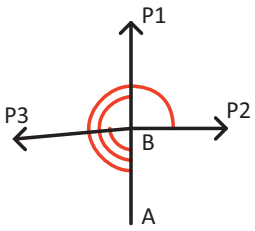

Figure 5. Selection of a loop road.

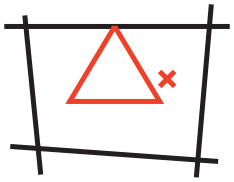

Figure 6. Sample of Loop Road algorithm that cannot generate a loop road in one try.

3.2 Expand Loop Road algorithm

The Expand Loop Road algorithm expands city blocks surrounded by a loop road along the radial direction block-by-block. We define the smallest loop road that surrounds point C as the level 1 loop road. We define the loop road expanded n times by the Expand Loop Road algorithm as the level n+1 loop road.

We describe the Expand Loop Road algorithm as follows. For each link in the level n loop road, apply the Loop Road algorithm in the clockwise direction, as shown in Figure 7. This gives us the level n+1 loop road. This loop road expands city blocks along the radial direction. However, in the case of complex road networks, we have to apply the Loop Road algorithm in the counterclockwise direction for some links in order to expand city blocks. Therefore, we have to apply the Loop Road algorithm in not only the

Algorithm 1 Loop Road algorithm (counterclockwise)

```
Require: start, goal
stack=new Stack()
setVisitedLink(goal,start)
stack.push(start)
while not stack.empty() do
    node ⇐ stack.top()
    if node ∈ stack without top then
        return stack
    else
        child ⇐ node.UnvisitedLeftChild()
        if child = none then
            stack.pop()
        else
            setVisitedLink(node,child)
            stack.push(child)
        end if
    end if
end while
```

clockwise but also the counterclockwise direction for each link.

The algorithm is as given below.

STEP1 Initialize List A.

STEP2 Store links of an initial loop road (Figure 7-a) in List A.

STEP3 For each link stored in List A, apply the Loop Road algorithm in the clockwise and counterclockwise directions, as shown in Figure 7-b.

STEP4 Remove links stored in List A from links generated in STEP3.

STEP5 Apply the Loop Road algorithm in the counterclockwise direction by using the links generated in STEP4, as shown in Figure 7-c. Finally, we obtain an expanded loop road.

STEP6 If the city blocks need to be further expanded, repeat from STEP2, as shown in Figure 7-d,e.

Here, List A is the list structure that stores the loop road. Finally, we can obtain the radius and center point of the Focus that contacts to the loop road for each level.

Figure 8 shows an example of the Expand Loop Road algorithm for each level. In this case, it changes the size of the Focus centered at the Nagoya Institute of Technology. Figure 8-a shows a level 1 loop road; Figure 8-b, a level 2 loop road; and Figure 8-c, a level 3 loop road. Although this campus consists of some city blocks, we can expand the Focus properly by considering these city blocks. Moreover, the loop roads including each level are stored in the database by associating them with landmarks.

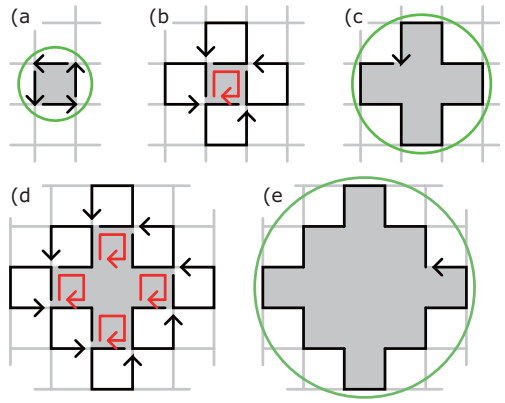

Figure 7. Procedure of Expand Loop Road algorithm. Gray lines, black arrows, red arrows, and the green circle indicate roads, links, previous links, and the Focus area, respectively.

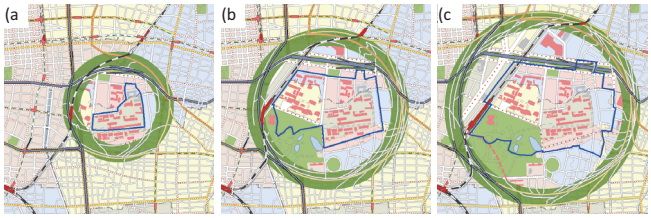

Figure 8. Example of loop road with the Focus. a) level 1 loop road, b) level 2 loop road, and c) level 3 loop road.

4 Prototype System

The vector map data adopted by the prototype system are Standard Road Map 2007 and Navigation Road Map 2007 obtained from Yahoo! Japan Corporation. In addition, we generate static raster maps by using ProAtlas Enterprise Server Development Kit obtained from Yahoo! Japan Corporation. Because these maps are generated from the same map data, there is no misalignment when they are overlapped. The server system was developed using Java Servlets and MySQL, and the client system, using Adobe Flex3.

Figure 9 shows the result generated by using the prototype system without the Focus. Our system achieved not only an area consisting of orderly city blocks but also the downtown area consisting of an intricate system of roads, as shown in Figure 9-a. Although the proposed algorithm can achieve an area that consists of long and thin blocks such as a river terrace, the result of the loop road is too long and thin and the Focus becomes too large, as shown in Figure 9-c. Moreover, in an area that has only a few roads, such

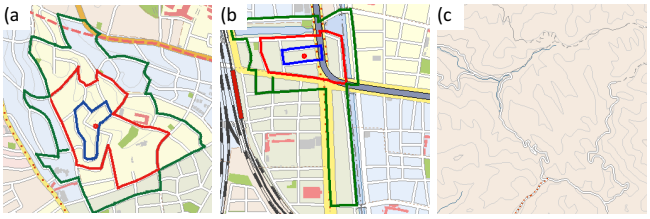

Figure 9. Example of Expand Loop Road algorithm. A level 1, level 2, and level 3 loop road are indicated in blue, red, and green, respectively.

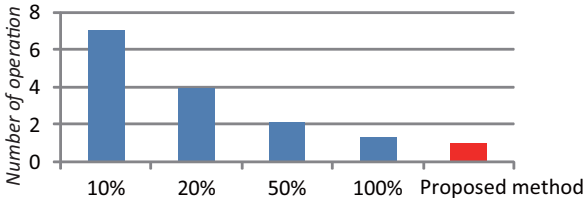

Figure 10. Comparison between number of operations in conventional and proposed method. 10%, 20%, 50%, and 100% scales for the conventional method.

as a mountainous district, the proposed system generates a loop road that is very large, as shown in Figure 9-c. In a future work, we intend to improve the Expand Loop Road algorithm in order to adapt to such areas by considering the shape of blocks.

5 Experimental Results

In this section, we evaluate the Focus controlling method based on the Expand Loop Road algorithm. The target landmarks are universities and parks in Japan. Some of these consist of some city blocks. We compare the proposed method with the conventional method from the viewpoint of the number of operations and fit ratios. The two methods are described as follows.

Method 1 (proposed method) Enlarge and narrow the Focus based on city blocks by using the Expand Loop Road algorithm.

Method 2 (conventional method) Enlarge and narrow the Focus by constant scales (10%, 20%, 50%, 100%).

In each method, we enlarge the Focus from the initial size until it covers the entire target area. Then, we measure the number of operations and the fit ratio, defined as follows.

Number of operations Number of operations that enlarge the Focus from the initial size.

Fit ratio The ratio of the dimension of the Focus to that of the circle that contacts the outline of the target area.

In addition, the initial radius of the Focus is 80 pixels. This size can cover most small parks in Nagoya city. Moreover, the positions (latitude and longitude) of the targets are acquired by Yahoo! Geocoder.

Figure 10 and Figure 11 show the results.

The conventional method has a disadvantage in that although the fit ratio is high (76.5%) when the Focus expands by 10%, the number of operations is also large (7.0 times). Although the number of operations is small (1.33 times) when the Focus expands by 100%, the fit ratio is not high (50.5%). Therefore, we cannot simultaneously realize both a high fit ratio and a small number of operations. On the other hand, the proposed method can achieve a high fit ratio (82.6%) with a small number operations (1.07 times); in other words, it can simultaneously realize a high fit ratio and a small number of operations. These results suggest that the proposed method can generate a Focus with higher accuracy and a lesser number of operations than the conventional method.

The proposed method has the following two advantages. First, it determines the radius of the Focus by considering the outline of the target based on city blocks. In the conventional method, since the radius is increased by a constant value such as 10%, users cannot set the radius to an intermediate value. In contrast, the proposed method can set the radius of the Focus to an appropriate value according to the outline of city blocks. Second, the proposed method can adjust the position of the Focus properly. In the conventional method, since the center position of the Focus is the fixed point acquired from the geocoder, the fit ratio is not high if this point is out of alignment from the center of the target. Since the proposed method can adjust the center position of the Focus based on the outline of the target area, the Focus can fit the target area properly.

Next, Figure 12 shows the relation between the number of operations that shows the entire target area and the fit ratio in this case. Although the fit ratio is 68% on average in the conventional method when the number of operations is 0, the fit ratio is 100% in the proposed method. This is because the level 1 loop road always corresponds to the outline of the target city blocks. This suggests that the proposed method has a significant advantage when the target consists of one city block. Moreover, this suggests that the fit ratio of the proposed method is always better than that of the conventional method irrespective of the number of operations.

6 Related Work

In some studies, a map was controlled in mobile terminals using posture sensors. Rekimoto [6] proposed a mech-

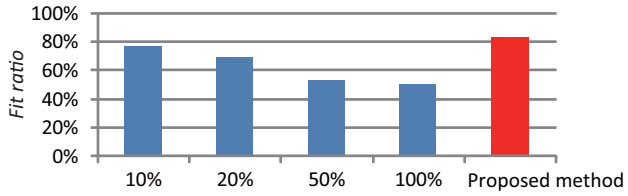

Figure 11. Comparison between fit ratio of conventional and proposed method. 10%, 20%, 50%, and 100% are scales for the conventional method.

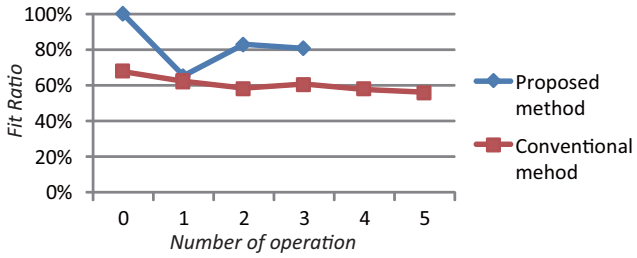

Figure 12. Relation between number of operations and fit ratio.

anism that enables users to select and zoom a target area in a bird's-eye-view map by tilting a mobile terminal by pushing a button. In contrast to the fisheye map or F+G+C map, in this mechanism, a user has to search for a target area using a wide-area map before he/she views a detailed map of the target many times. Gutwin [2] proposed a method that enables users to view Web pages by using the Focus+Context method in a mobile terminal. Because ordinary Web pages viewed on a PC are not optimized for small displays, in this method, the zoom in the focus area increases and that in the context area decreases. Although users can view Web maps by using this mechanism, this interface does not consider the characteristics of maps.

7 Conclusion

In this paper, we proposed an Expand Loop Road algorithm in order to control the size and position of the Focus in the Focus+Glue+Context map. In addition, we developed a prototype system based on this algorithm. Moreover, we evaluated the prototype system. The obtained results suggest that the proposed method can determine the size of the Focus more appropriately with fewer operations than the conventional method. In fact, although the existing method has to operate 7.0 times to achieve 80% accuracy, the proposed method operates only 1.07 times to achieve 82.6% accuracy. By developing a mobile map system with fisheye views that employs the proposed method, we will be able to control a map easily.

In future work, we intend to solve the following problems. When the target area includes long and thin city blocks, the loop road generated by the proposed algorithm is very long. Therefore, we have to improve the Expand Loop Road algorithm in order to adapt to these areas by considering the shape of blocks. We will propose a new method for scrolling or turning the map by considering the components of a cognitive map, such as a district. By considering city blocks, we will be able to control the maps using fisheye views. In addition, we will develop a system that can operate on common cellular phones such as an iPhone, and we will make this system available publicly. Our system will serve as a novel mobile WMS with fisheye views for mobile terminals.

Acknowledgment We would like to thank the Yahoo! Japan Corporation for supporting us during the development of the prototype system. This work was also supported by JSPS KAKENHI 20509003.

References

[1] C. Gutwin and C. Fedak. A comparison of fisheye lenses for interactive layout tasks. In *Proc. Graphics Interface 2004*, pages 213–220, 2004.

[2] C. Gutwin and C. Fedak. Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. In *Proc. Graphics Interface 2004*, pages 145–152. ACM Press, 2004.

[3] C. Gutwin and A. Skopik. Fisheye views are good for large steering tasks. In *Proc. SIGCHI 2003*, pages 5–10, 2003.

[4] L. Harrie, L. T. Sarjakoski, and L. Lehto. A variable-scale map for small-display cartography. In *Proc. Symp. on GeoSpatial Theory, Processing, and Applications*, pages 8–12, 2002.

[5] K. Lynch. *The image of the city*. MIT Press, 1960.

[6] J. Rekimoto. Tilting operations for small screen interfaces. In *Proc. 9th ACM Symp. on User Interface Software and Technology*, pages 167–168, 1996.

[7] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proc. SIGCHI 1992*, pages 83–91. 1992.

[8] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In *Proc. 6th ACM Symp. on User Interface Software and Technology*, pages 81–91. 1993.

[9] N. Takahashi. An elastic map system with cognitive map-based operations. *Int'l Perspectives on Maps and the Internet, Michel P. Peterson (Ed.), Lecture Notes in Geoinformation and Cartography*, pages 73–87, 2008.

[10] D. Yamamoto, S. Ozeki, and N. Takahashi. Focus+glue+context: An improved fisheye approach for web map services. In *Proc. ACM GIS 2009*, pages 101-110, 2009.

[11] D. Yamamoto, S. Ozeki, and N. Takahashi. Wired fisheye lens: A motion-based improved fisheye interface for mobile web map services. In *Proc. W2GIS 2009)*, 2009. (to appear).