Path finding on a spherical SOM using the distance transform and floodplain analysis

Michael Bui and Masahiro Takatsuka
ViSLAB
School of Information Technologies
University of Sydney
email: {mbui,masa}@vislab.usyd.edu.au

Keywords: self-organizing map, distance transformation, floodplain, path planning

***Abstract*— Data visualization has become an important tool for analyzing very complex data. In particular, spatial visualization enables users to view data in a intuitive manner. It has typically been used to externalize clusters and their relationships which exist in highly complex multidimensional data. We envisage that not only cluster formation and relationships but also other types of information, such as temporal changes of datum, can be extracted through the spatialization.

In this paper, we investigate an application of trajectory/path analysis carried out using a Self-Organizing Map as a spatialization method. We propose an application of distance transformations to the Geodesic Self-Organizing Map. This new approach allows a user to visually inspect the trajectory of multidimensional knowledge pieces on a two-dimensional space. The trajectories discovered through this approach are essentially the shortest paths between two points on the Self-Organizing Map. However, those paths might go outside of the input dataspace due to the connectivity of neurons imposed by the grid structure. We also present a method to find the shortest path, which falls within the input dataspace using simple floodplain analysis.

1 Introduction

Our society has for a long period of time used maps as tools for exploration and navigation. Through the use of maps, a simplified representation of a space can be obtained, reducing the complexity of discovering pathways in the unexplored space. However, maps are not just reserved for geography but have also been used for information visualization; the latter having greater relevance to our research.

A popular jargon term, referred to as roadmaps, has even been made to describe plans consisting of many stages. Such roadmaps, like those for software development projects, consist of many intermediate stages that need to be incrementally reached in order to achieve a clearly defined goal. We envisage that if an abstract knowledge space can be represented in the form of a map, transitions of knowledge can be defined as a path on these maps. The discovery of a desirable path could then be translated as a process of knowledge transition.

It would thus be of key interest to our research to study the usage of maps to spatialize multidimensional data onto a two-dimensional or three-dimensional space. Once the data has been spatialized, we can use these maps to track temporal or state changes, such as the movement of a data point. Furthermore, paths in the original high-dimensional space could also be found. If a map is used to describe a knowledge space, the process of decision making can then be viewed as being equivalent to finding paths on the map.

With the recent efforts on applying cartographic perspectives on non-geographic information visualization [1, 2], we believe it would be worthy to investigate the application of techniques which have typically been used on georeferenced data, such as path planning, on non-geographic data. This is also driven by the fact that spatial analysis techniques should, in principle, applicable to spatial representations of data. By using automated computational techniques to perform tasks such as path planning, alternative and better paths may be discovered which would have otherwise remained hidden.

We recently proposed the application of distance transformations to the Geodesic Self-Organizing Map (Geodesic SOM). The Geodesic SOM would be used to visualize complex datasets in the form of a spherical map. Distance transformations are then used to perform multidimensional path planning on the resulting visualization in two dimensions. This computes the shortest path between two points on the Geodesic SOM. In some situations, these paths may traverse through cluster boundaries of large distances even there is no input data covering such area. In such cases, there may be other possible and more desirable paths to travel around these cluster boundaries. Such paths will travel through the dataspace represented by the existing input data. In this paper, we present an application of floodplain analysis to improve the path finding process in order to find the flattest path to solve this problem.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

2 Related work

2.1 The Self-Organizing Map and Visualization

The Self-Organizing Map (SOM) [3] is a widely used artificial neural network which can be used to visualize data. The benefits of using SOMs are that they can (1) preserve the topology of the data, (2) approximate the probability density function, (3) perform multidimensional scaling and (4) unsupervised clustering.

Their topology preservation properties have allowed them to be regarded as roadmaps of the high-dimensional space [4]. Moreover, they could also be used for temporal sequence processing. Temporal information can then be recovered from SOMs, even though time may not be taken into consideration during the training process. Given a temporal sequence of features, their corresponding best matching units on the SOM can then be connected to form trajectories that visualizes the state transitions. These "trajectory-based SOMs" have been used for speech recognition [5], process monitoring [6] and bankruptcy prediction [7]. Financial benchmarking applications [8] have also demonstrated that there is the potential for SOMs, in general, to be used as a tool for strategic management. This could support executives in discovering characteristics that would lead a company to reach and maintain desirable performance levels.

Despite the benefits of using SOMs, one of problems associated with them is the "border effect". As neurons on the boundaries of the SOM have fewer neighbours than those within the boundaries, the chance of these neurons being updated is lower. Hence, the map may appear to be less well ordered near the boundaries. One of the various suggestions to eliminate this was to implement the SOM on a spherical lattice so that borders would not exist [9].

Wu and Takatsuka [10] reported that the icosahedron-based geodesic dome is the most suitable spherical lattice for implementing the SOM. Compared to the other platonic polyhedra, the icosahedron has the least variance in edge lengths after tessellation and more closely resembled a sphere. This led to the development of the Geodesic Self-Organizing Map (Geodesic SOM) whose 2D data structure is able to support fast neighbourhood searching for updating neurons in a neighbourhood set. Experimental results were able to show that the Geodesic SOM had more uniform error distribution and potentially better performance when dealing with large datasets. Furthermore, its 2D data structure is able to support fast neighbourhood searching.

2.2 Distance transformations

The original distance transformation algorithm [11] was developed for image processing. Given a binary image I composed of pixels that have a value of either 0 or 1, it is able to compute a corresponding distance map D. This is achieved by propagating distance values from one pixel to another. The pixel value of $D(i, j)$ would then represent the distance of pixel $I(i, j)$ to the nearest zero pixel.

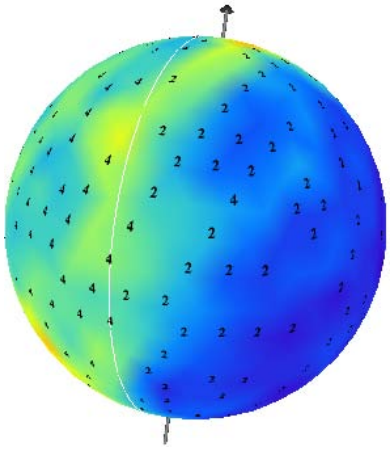

Figure 1: An example of a visualization produced by the Geodesic SOM. The arrow passes through the two extreme points from the south to the north pole. Regions are labelled with numbers to indicate which cluster the points in these regions belong to.

While distance transformations are generally known for their applications in pattern recognition [12], it was discovered that by extending the original algorithm, they could be used to calculate optimal, collision-free paths to solve robot motion planning problems [13]. In this case, a distance wave would be propagated from a source cell through the free space around any obstacles. While A* [14] is commonly used in continuous, real Euclidean space maps, distance transformations are prominent for discretized grid maps. Furthermore, the algorithms are designed to be fast due to the nature of the problems it may be used to solve.

Figure 2 illustrates how the distance transformation algorithm can be applied to compute a distance map to help solve robot motion planning problems. In this example, the goal cell is marked by the letter G. By applying the distance transformation on the map in figure 2 (a), this would compute a distance map as depicted in figure 2 (b). Since the values in each cell represents the minimum amount of cells that need to be travelled to reach the goal, a path to the goal can be found by following the steepest descent (algorithm 1), although any algorithm that calculates the shortest path can be applied to the distance map.

2.3 Path Planning in Multidimensional Data

Trajectory-based SOMs have proven that spatial visualizations can be used to track temporal and state changes in a high-dimensional space [5, 6, 7]. By using the SOM to perform dimensionality reduction, the task of tracking the movement of a data point in high-dimensional space becomes simpler to achieve and perceive. Due to the SOM's

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

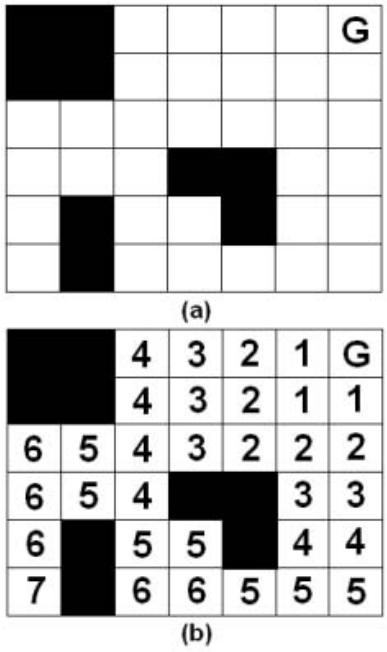

Figure 2: Figure (a) depicts an example of an environment that contains obstacles and is typical of problems in robot motion planning. The goal cell is labelled by the letter G. Applying the distance transformation results in figure (b). Propagation begins from the goal cell and the distance values in each cell represents the shortest distance to the goal cell. By following the steepest descent, a path to the goal can be found.

ability to approximate the data distribution, neurons with no data samples mapped to them are still significant in some way. Therefore, with the appropriate data, if one were to calculate a path between two points on the SOM, the weight vectors of the neurons on the path would represent the intermediate states that need to be reached. To date, no work has been done to take the next step further with trajectory-based SOMs. That is, to progress from utilizing trajectory-based SOMs for tracking the path of an object in high-dimensional space, to calculating/finding the path of an object migh take.

This led to our proposal to apply a distance transform to the Geodesic SOM. In our approach, the Geodesic SOM is utilized to perform dimensionality reduction and for map generation. Each neuron is associated with a distance value, initially -1, which represents the shortest distance from that neuron to the goal. Since the U-Matrix is used to visualize the Geodesic SOM, the calculated U-heights (the local distances in the high-dimensional space) are used as the distance values that would be propagated by the distance transformation algorithm (algorithm 2). This approach helps solve the problem of finding the shortest path between two points on the Geodesic SOM. Note that the neurons on the boundaries on the Geodesic SOM's data structure have duplicates since the data structure is formed by opening up an icosahedron (refer to [10] for more details). Thus, when these neurons need to have their distance values updated, their duplicates also need to be processed.

Algorithm 1 Calculating the shortest path through steepest descent

```
Require: start and goal cell, distance map
  c = start
  while c != goal do
    for all n = c.neighbours do
      if n.distance < c.distance then
        c = n
      end if
    end for
  end while
```

Algorithm 2 Distance transformation on the Geodesic SOM to find the shortest path

```
Require: start and goal and Geodesic SOM
  Initialization
  Queue Q is empty
  for all neuron in neurons do
    neuron.distance = -1
  end for
  goal.distance = 0
  Q.push(goal)
  Start distance transformation
  while Q is not empty do
    c = pop(Q)
    d = avg_diff√2
    neighbours = getNeighbours(current)
    for all n in neighbours do
      if n.distance == -1 then
        n.distance = current.distance + n.avg_diff
        Q.push(n)
      else
        if n is a diagonal neighbour and n.distance >
        c.distance + d then
          n.distance = c.distance + d
          Q.push(n)
        else
          if n.distance > c.distance + n.avg_diff then
            n.distance = c.distance + n.avg_diff
            Q.push(n)
            for all v in n's duplicate neurons do
              v.distance = n.distance
            end for
          end if
        end if
      end if
    end for
  end while
```

Experiments have shown that information about the state/temporal changes of data could be recovered by us-

ing this approach (figure 3). Compared to previous approaches which have prior knowledge of all of the states in a temporal sequence, we rely on a topological mapping of a high-dimensional space to recover temporal information given only an initial and end state. This approach works based on the assumption that consecutive states are similar to each other in the high-dimensional space and would thus be placed near each other on the low-dimensional mapping. Consequently, there is a potential that this could also be used to forecast such changes. Since weight vectors are essentially state vectors, parameters can then be extracted from these vectors for purposes similar to process steering [15]. In fact, it has been suggested in the aforementioned literature that path finding algorithms could help steer a process toward some optimum operating state. Thus, for applications that need to discover the consecutive states that need to be reached from one state to another, our technique may be used to discover such state transitions.

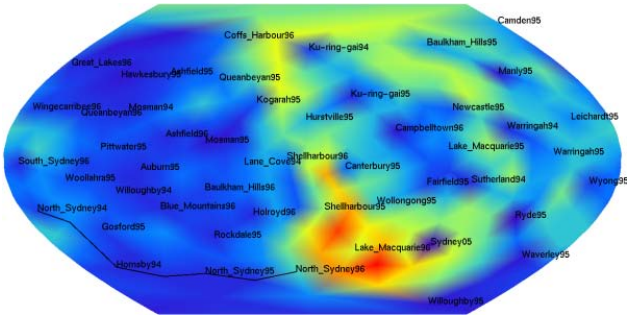

Figure 3: Geodesic SOM trained with the benchmarking data on the 1994/1995, 1995/1996 and 1996/1997 financial year. A five-frequency geodesic dome(252 nodes) was used, with an initial update radius of 9, while the Geodesic SOM was trained for 3000 epochs. The distance transformation was applied to find the path from North Sydney's state in the 1994/1995 financial year, to its state in the 1996/1997 financial year. The path can be seen to pass through Hornsby and North Sydney in the 1995/1996 financial year.

As mentioned earlier, there is a problem with the proposed approach. In figure 4, we can see an example of a dataspace that may be used to train the Geodesic SOM. After training the Geodesic SOM, contraction may occur whereby two dissimilar points may be placed near each other on the SOM and there may be neurons representing points outside of the dataspace. Consequently, if we applied the distance transformation to find the shortest path (dashed line in the figure) between two points within this dataspace, the path may actually travel outside of the dataspace. Figure 5 demonstrates what this may look like on the SOM where this may result in an incorrect path being found. To ensure that a path travels within the dataspace, we propose to find the flattest, shortest path instead.

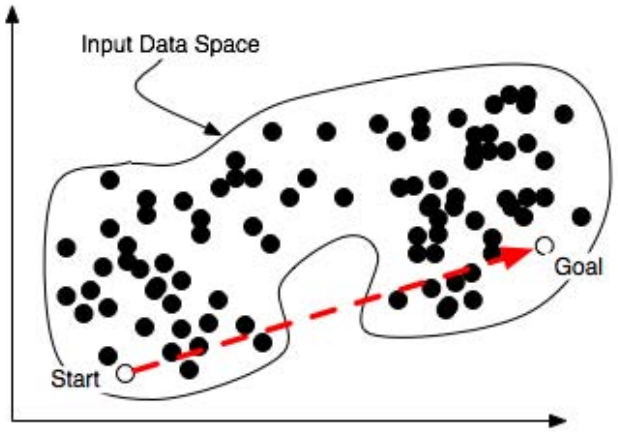

Figure 4: An example of a dataspace. The dashed line depicts a path thats move outside of the dataspace, which is the region enclosed by the curved shape.

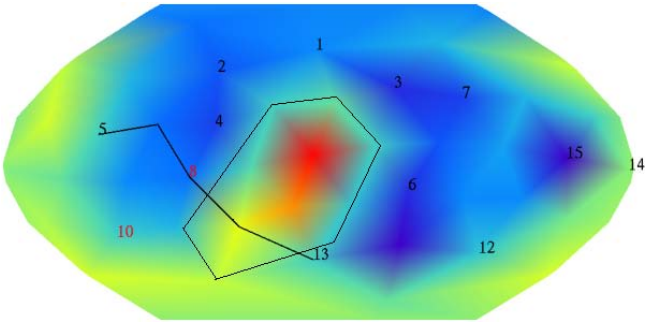

Figure 5: The path (5-8-13) from node 5 to 13 on the Geodesic SOM. The path is incorrect as rather than travelling through the flat region on the SOM (blue region), it travels through the mountain range enclosed by the hexagonal shape.

3 Calculating the Flattest Path in Multidimensional Data

Robot motion/path planning applications operate in environments where obstacles may exist. Here, path planning is typically performed to calculate collision-free paths (which are usually optimal) from a starting location to a goal location. We initially applied distance transformations to the Geodesic SOM to perform path planning with multidimensional data in two dimensions. The result would then correspond to the shortest path from one point to another on the Geodesic SOM. However, visual inspection revealed that these paths may pass through large cluster boundaries on the SOM, despite the fact that alternate paths around them may exist. If one were to use a landscape metaphor to describe the U-Matrix, these regions would appear as mountain ranges on the landscape. The best route would clearly be to go around these mountains if possible.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

Furthermore, one of the properties of the SOM is that it will approximate the probability density function of the input data. If we view the input data as a cloud, or even multiple clouds of points, then the SOM would function like an elastic net that tries to fit itself around all of these points. This results in a surface representing the probability density function. The surface, however, might cover regions where there are no samples. This is because the structure of this surface is governed by the neighbourhood connectivities of neurons. In this scenario, when a path passes through a cluster boundary on the SOM, this may correspond to the path navigating in a region where no training samples exist. Since the SOM may have neurons that correspond to these regions, finding the shortest path between two neurons may reveal a path that goes through these neurons. Calculating the flattest path instead will help limit paths such that they will only travel along the region representing by the input dataset's probability density function, and create a sense of continuity.

3.1 Approach

The approach we use to find the flattest path between two points on the Geodesic SOM is a modification to the approach that we have initially presented (a simple application of distance transformations), and is described below by algorithm 3. Hence, distance transformations are still applied to the Geodesic SOM. The difference here is that the neurons with a U-height (denoted by u-height) above a certain threshold (max_u-height) will be ignored. Given the start and goal neuron, the threshold's value will be initialized as the largest U-height value out of these two neurons. The threshold max_u-height defines a floodplain-like landscape. In other words, all neurons whose u-height is smaller than max_u-height belong to the floodplain, and we would like to find the shortest path on this floodplain by applying the distance transformation.

The distance value of the goal neuron will be initialized to 0, while the rest of the neurons will have a distance value of -1. This distance value corresponds to the shortest distance to the goal neuron, with a value of -1 meaning that no path exists from the corresponding neuron to the goal. The goal neuron is then added to a queue. It is then removed from the queue and its distance value will be propagated to its direct neighbours so that their distance values can be updated. The direct neighbours are then added to the queue so that their distance values may also be propagated, and this process is repeated until the queue is empty. However, only neurons with a U-height value below or equal to the threshold will be added to the queue. If no path exists from the start to the goal, that is, the start neuron's distance value is -1, then the threshold is increased to be the next highest U-height that was found during the distance transformation. In this situation the distance transformation will be repeated, taking the threshold into consideration, until a valid path is found from the start neuron to the goal neuron.

Algorithm 3 Distance transformation on the Geodesic SOM to find the flattest, shortest path

Require: start and goal and Geodesic SOM
  max_u-height = max(start.u-height, goal.u-height)
  while start.distance $=$ - 1 **do**
    next_max_u-height = ∞
    Queue Q is empty
    for all neuron in neurons **do**
      neuron.distance = -1
    end for
    goal.distance = 0
    Q.push(goal)
    Start distance transformation
    while Q is not empty **do**
      c = pop(Q)
      d = u-height$\sqrt{2}$
      neighbours = getNeighbours(current)
      for all n in neighbours **do**
        if n.u-height $>$ max_u-height **then**
          if n.u-height $<$ next_max_u-height **then**
            next_max_u-height = n.u-height
          end if
          ignore n
        end if
        if n.distance == -1 **then**
          n.distance = current.distance + n.u-height
          Q.push(n)
        else
          if n is a diagonal neighbour and n.distance $>$ c.distance + d **then**
            n.distance = c.distance + d
            Q.push(n)
          else
            if n.distance $>$ c.distance + n.u-height **then**
              n.distance = c.distance + n.u-height
              Q.push(n)
              for all v in n's duplicate neurons **do**
                v.distance = n.distance
              end for
            end if
          end if
        end if
      end for
    end while
    max_u-height = next_max_u-height
  end while

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

4 Results

Our experiments were conducted on a Pentium 4 3.2GHz PC with 1 GB of memory and the Geodesic SOM was implemented in Java. Here, the initial learning rate is always 0.8 and the update radius is set such that it covers approximately 80% of the network. Due to the fact that the Geodesic SOM visualizes the data in the form of a spherical dome, it is difficult to get a full view of the visualization and generated paths. Consequently, the Wagner III pseudocylindrical projection was used to project the Geodesic SOM to a two-dimensional map [10]. Our experiments involve both synthetic and real data. Note that as the colours on the Geodesic SOM are associated with the local distances in high-dimensional space, visual inspection could be used to see if they navigate around the mountain ranges.

4.1 Synthetic data

In this experiment, we constructed a synthetic dataset representing a binary tree of depth 3 that contains 15 nodes(refer to figure 6. The dataset itself is essentially a 15 x 15 distance matrix, that is, it consists of 15 data points described by 15 attributes. If we denote this matrix as M, the value of $M(i, j)$ represents the graph distance from node i to node j, where a value of zero would mean node i and node j are actually the same node.

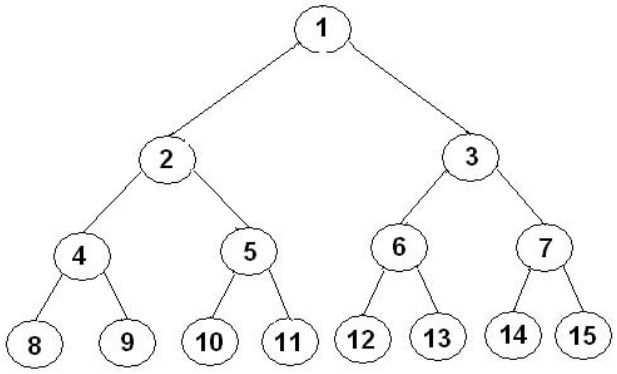

Figure 6: The binary tree used to construct the dataset.

The original method of applying distance transformations to the Geodesic SOM was able to reveal information about the structure of the data. The majority of the discovered paths on the Geodesic SOM (from the root, that is node 1) were also valid paths in the original binary tree (see figure 7 for an example). After analyzing the incorrect results, we were able to see cases where the paths would cross through the mountain ranges on the Geodesic SOM like in figure 8. If these paths were to go around these mountain ranges instead, then the paths may be valid in the original binary tree. In the following figures, the mountain ranges of interest is enclosed by a hexagon. We can see that such regions have no data samples whatsoever. Therefore, these regions could be said represent regions outside of the dataspace.

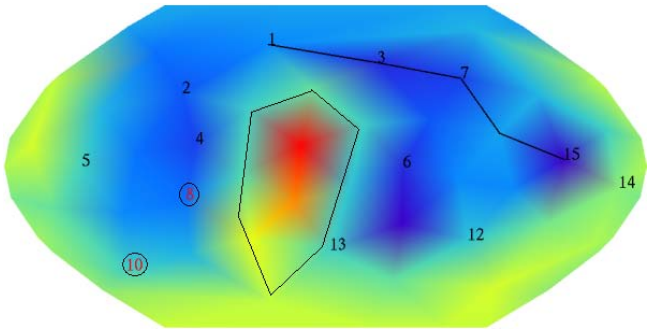

Figure 7: The path (1-3-7-15) from node 1 to 15 on the Geodesic SOM. The path is correct.

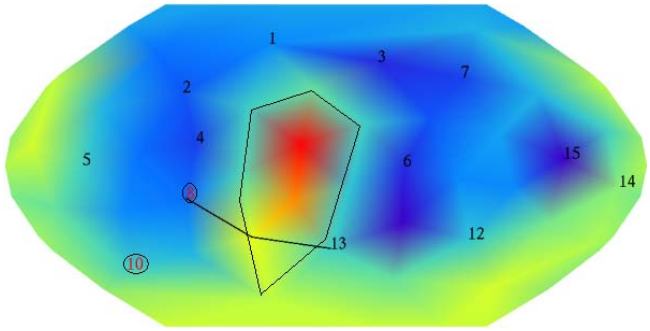

Figure 8: The path (8-13) from node 8 to 13 on the Geodesic SOM. The path is incorrect and also passes through the mountain range.

Our aim for this experiment is to see if using the modified approach to compute the flattest shortest path would actually travel around these regions and discover valid paths within the dataspace. A two-frequency geodesic dome was used (42 nodes), while the initial update radius is set to 3, and the Geodesic SOM was trained for 150 epochs. Note that there are two circled labels which are 8 and 10. This indicates that there was another node mapped to that same neuron as well, which are 9 and 11 respectively. Since these clustered nodes both share the same parent, this is not a problem.

Compared to figure 8, we can see that in figure 9, using the flattest, shortest path approach solves the problem where the path travels through the mountain range on the Geodesic SOM. Furthermore, the discovered path was indeed valid in the original binary tree. Therefore, in cases where the shortest path approach fails, the flattest path approach is a valuable alternative that is worth trying out to generate paths that may follow the structure of a dataset more closely.

Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007)
Published by the Neuroinformatics Group, Bielefeld University, Germany, ISBN 978-3-00-022473-7
All contributions to WSOM 2007 are available online at: http://biecoll.ub.uni-bielefeld.de

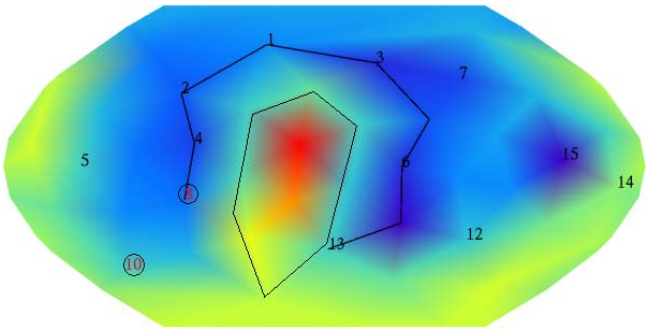

Figure 9: The path (8-4-2-1-3-6-13) from node 8 to 13 on the Geodesic SOM. The path is correct and travels around the mountain range enclosed in the rectangle unlike the path in figure 8

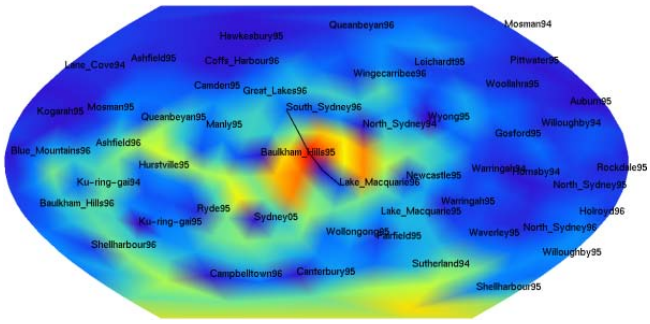

Figure 10: Geodesic SOM trained with the benchmarking data on the 1994/1995, 1995/1996 and 1996/1997 financial year. A five-frequency geodesic dome (252 nodes) was used, with an initial update radius of 9, while the Geodesic SOM was trained for 3000 epochs. The distance transformation was applied to find the path from South Sydney's state in the 1996/1997 financial year, to Lake Macquarie's state in the 1996/1997 financial year. The path passes through a region where there is no data.

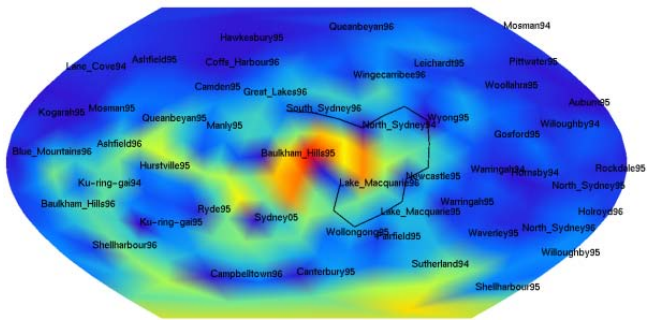

Figure 11: Geodesic SOM trained with the benchmarking data on the 1994/1995, 1995/1996 and 1996/1997 financial year. A five-frequency geodesic dome(252 nodes) was used, with an initial update radius of 9, while the Geodesic SOM was trained for 3000 epochs. The distance transformation was applied to find the path from South Sydney's state in the 1996/1997 financial year, to Lake Macquarie's state in the 1996/1997 financial year. The path passes through neurons with data samples which may help provide more meaningful results.

4.2 Financial benchmarking data

For this dataset, we used real public financial benchmarking data obtained from the State Library of New South Wales, Australia web site [16]. The data was taken from the income/expenditure statements of a number of cities in the metropolitan area for the 1994/1995, 1995/1996 and 1996/1997 financial years. This data was combined together to form a single dataset. Each library in the dataset that described by 9 attributes: the expenditure costs for administration, employee costs, library collections and overhead expenses, as well as the total income, capital expenditure, capital income, total operating expenses and deficit. By performing path planning on this data, we could use the results to determine the financial states that a library would need to achieve over a period of time in order to reach the same level as another library.

When performing financial benchmarking in the real world, one presumption that needs to be made is that the training data adequately covers the whole input space. This would allow the SOM to approximate all the realistic financial states. However, when there are not enough data samples, there may be neurons corresponding to financial states outside of the scope of the data, which may not even be realistic. It may be more desirable to use the data that does exist, and use distance transformations in a way that the paths will travel within the dataspace corresponding to the dataset. Figure 10 illustrates an example where there is a cluster boundary with no data samples. This region could correspond to a region outside of the dataspace. Thus, the application of distance transformations for path planning may not produce useful results.

By applying our new method to the same dataset, this can force the paths to travel along regions with data samples. In figure 11, the path between South Sydney and Lake Macquarie is observed to pass through a number of neurons containing data samples corresponding to states like Lake Macquarie, Newcastle and Wyong in the 1995/1996 financial year. It can also be observed that this method is able to discover a link between the states of Lake Macquarie's libraries in 1995/1996 and 1996/1997 that did not appear using the original method we proposed. Using these results may thus more beneficial as the results would be based on data that does exist.

5 Conclusions

It has been demonstrated through the SOM that spatial visualizations may allow users to track temporal changes of data in high-dimensional space. This process is similar to tracking the movement of an object, such as a car, on a ge-

ographic map. Such maps have been extensively used to plan paths to get from one point to another. Despite the fact that cartographic perspectives have been applied to the visualization of non-geographic data, not much effort has been made to attempt to plan paths on maps representing non-geographic data

Previously, we proposed to use the Geodesic SOM to visualize multidimensional data in the form of a map, due to its ability to effectively remove the border effect and reduce data distortion. Distance transformations, which have frequently been used for robot motion planning, were then applied to perform path planning in high-dimensional space in two dimensions through the Geodesic SOM's 2D data structure. However, this on some occasions would generate paths that pass through mountain ranges on the Geodesic SOM. These regions may correspond to points outside of the dataspace. Consequently, we have presented in this work, a modification of the original application of distance transformations in order to solve this problem. This involves ignoring neurons whose U-height are above a certain threshold when performing the distance transformation. If no valid path can be discovered, the threshold is increased and the distance transformation is done again. This process is repeated until a valid path is discovered. This is a rather simple method to find the flattest path. Future work on this would concentrate on using a better approach to find the threshold for which a valid path exists.

While we have not been able to secure more useful data at this stage and evaluate how this would perform in the real world (which further work would concentrate on), our results so far are promising. The paths that our technique generates clearly attempts to avoid regions that would be undesirable to traverse through as much as possible. By limiting the neurons that are to used during the distance transformation, this restricts the paths so that they will only travel on the surface representing the probability density function. This creates a sense of continuity and the results may be more meaningful to users as the paths would only travel within the dataspace.

Acknowledgements

The authors wish to thank Yingxin Wu for her permission to use images related to the Geodesic SOM, and her Geodesic SOM work.

References

[1] A. Skupin, "From Metaphor to Method: Cartographic Perspectives on Information Visualization", *Proceedings of the IEEE Symposium on Information Vizualization 2000*, pp. 91, Washington, DC, USA, 2000. IEEE Computer Society.

[2] A. Skupin and S. Fabrikant, "Spatialization methods: A Cartographic Research Agenda for Non-geographic Information Visualization", *Cartography and Geographic Information Science*, Vol. 30, No. 2, pp. 95119, 2003.

[3] T. Kohonen, "Self-Organizing Maps, Third Edition", Springer-Verlag, Berlin Heidelberg, 2001.

[4] A. Ultsch, "U*-matrix: A Tool to Visualize Clusters in High Dimensional Data", *Technical Report 36*, University of Marburg, 2003.

[5] T. Kohonen, "The Neural Phonetic Typewriter" *Computer*, Vol. 21, No. 3, pp. 1122, 1988.

[6] M. Kasslin, J. Kangas, and O. Simula, "Process state monitoring using self-organizing maps", I., Aleksander and J. Taylor, editors, *Artificial Neural Networks*, Vol. 2, pp. 1531-1534, North-Holland, 1992.

[7] C. Serrano-Cinca, "Let financial data speak for themselves", G. J. Deboeck, T. K. Kohonen, and T. K. Kohonen, editors, *Visual Explorations in Finance*, chapter 1, pp. 323. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[8] B. Back, M. Irjala, K. Sere and V. Vanharanta, "Competitive Financial Benchmarking Using Self-Organizing Maps", *Åbo Akademi, Reports on Computer Science and Mathematics*, Series A, No. 169, 1995.

[9] H. Ritter, "Self-Organizing Maps on Non-Euclidean spaces", *Kohonen Maps*, pp. 97108, Amsterdam: Elsevier.

[10] Y. Wu and M. Takatsuka, "Spherical Self-Organizing Map using Efficient Indexed Geodesic Data Structure", *Neural Networks*, Vol. 19, No. 67, pp. 900910, July August 2006.

[11] A. Rosenfeld and J. L. Pfaltz, "Sequential Operations in Digital Picture Processing", *Journal of the ACM*, Vol. 13, No. 4, pp. 471494, 1966.

[12] K. K. Lau, P. C. Yuen, and Y. Y. Tang, "EDT Based Tracing Maximum Thinning Algorithm on Grey Scale Images", *Proceedings of the Fifthteenth International Conference on Pattern Recognition*, pp. 28632866, Los Alamitos, CA, USA, September 2000. IEEE Computer Society.

[13] R. A. Jarvis, "Collision-Free Trajectory Planning Using Distance Transforms", *Mechanical Engineering Transactions of the Institution of Engineers*, Vol. ME10, No. 3, pp. 187-191, 1985.

[14] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100-107, 1968.

[15] V. Tryba and K. Goser, "Self-organizing feature maps for process control in chemistry", T. Kohonen, K. Mkisara, O. Simula, and J. Kangas, editors, *Proceedings of the International Conference on Artificial Neural Networks*, pp. 847-852, North-Holland, Amsterdam, 1991.

[16] State Library of New South Wales, Accessed 27th Feb 2007, http://www.sl.nsw.gov.au/pls/benchmark/