Adaptive Tessellation Mapping (ATM) for Spatial Data Mining

Ting Wang, Senior *Member*, *IACSIT*

***Abstract*—In the research of spatial data mining, gridding/tessellation mapping is a common technique to aggregate the locational data points in smaller regions (namely grids or tiles) so that properties of those data points can be observed. It is a natural way to study spatial related information because such information is dependent to the locational proximity in most of the cases, and it significantly reduces the effort needed to learn useful insights from the data of the entire area. In this work, we propose an adaptive tessellation mapping (ATM) method to decompose the entire area of interest to tiles with variable sizes so that spatial data mining can be carried out more efficiently, purposefully and dynamically. In particular, we show that human behavior can be understood better with ATM with some examples.**

***Index Terms*—Spatial data mining, data structure, adaptive tessellation mapping, behavioral analysis.**

I. Introduction

Over the last few decades, with the increasingly accurate positioning services (e.g. GPS, AIS, Mobile Phone Triangulation, RFID/Wi-Fi tracking etc.) and the decreasing price of their deployment, data that identifies the geographic locations of objects (namely *spatial data*) are becoming pervasive in our daily lives and scientific researches. Either indoor or outdoor, it is not difficult to obtain the trace, the velocity, and even the acceleration of any moving entity of our interest, providing proper equipment and infrastructure. As part of the "big data regime", interests in spatial data have recently grown even more rapidly thanks to the new database technology and data mining techniques.

In general, *spatial data mining*, or knowledge discovery in spatial databases, is the extraction of implicit knowledge, spatial relations and discovery of interesting characteristics and patterns that are not explicitly represented in the databases. These techniques can play an important role in understanding spatial data and in capturing intrinsic relationships between spatial and non-spatial data. Moreover, such discovered relationships can be used to present data in a concise manner and to reorganize spatial databases to accommodate data semantics and achieve high performance.

Spatial data are widely used in a variety of applications, such as traffic modeling, supply chain management and human behavior analytics. These efforts are being hampered by the sparse nature of data collection strategies, the sheer volume of the data, and technical issues associated with the use of the data. The enormous volume of data can easily overwhelm analysis. This motivates the need for automated methods to split them to smaller pieces so that it becomes feasible to design and apply algorithms to the data.

One of the most common practice is to define geographical boundaries and divide the entire area of interest to smaller regions, so that data points inside the same region can be aggregated to one group, and treated by some algorithm. Statistical properties or similarities could then observed from each single region, and compare to others. Moreover, usually the same algorithm would be applied to each of the regions till all data points in the entire area are covered. This is particularly important when the algorithm is less scalable and it is impractical to apply it directly to all the data points at once.

Mathematically, this process could be understood as a *tessellation* process [1], which is defined in Section II. For the sake of simplicity, most of the existing works in spatial data mining use fixed tessellation, in which the area is divided to uniform regions with the same shape and size. However, in Section III this work we argue that an *Adaptive Tessellation Mapping (ATM)* scheme, which splits the area to regions of different sizes with a hierarchical structure, could be more efficient and offer us more and better insights. We present some use cases of ATM in behavioral analytic in Section IV. Section V concludes the paper with our findings and future works.

II. Background

In this section, some preliminary concepts and requirements will be discussed, before we go into the detail mechanism of ATM.

A. Tessellation Mapping

A *tessellation* of a flat surface is the tiling of a plane using one or more geometric shapes, called *tiles*, with no overlaps and no gaps. In the spatial data mining, *tessellation mapping* is used to overlay the tiles on top of existing spatial data, so that the locational data points could be identified by the tiles they belong to. Moreover, data points mapped in the same tile could be considered as one cluster, which forms the basic unit of algorithms or visualization techniques. Tessellation is also studied as a way to define coordinates in a Geospatial Information System (GIS), so that the spatial data could be systematically addressed. In this way, tessellation is directly related to the data structure that host spatial data so that they can be grouped or aggregated.

In some works, tessellation is also referred to as *gridding*

Manuscript received April 10, 2014; revised June 25, 2014. This work was supported in part by the Singapore Economic Development Board (EDB) and Singapore National Research Foundation (NRF).

Wang Ting is with SAP Asia Pte Ltd, Singapore (e-mail: dean.wang@sap.com).

[2], *sectioning* [3], *partitioning* [4], or *meshing* [5]. In fact, they are all special cases of tessellation:

- Gridding and sectioning are tessellation with only square tiles.
- Partitioning is mostly used in aggregation of wireless sensors. Even in [4], main focus of the paper was on the addressing issue, instead of making the partitions adaptive.
- Meshing is tessellation with tiles of uniform size.

In our work, we use the more general term tessellation, because the tile size may vary in our scheme, and we will also discuss the use of tiles with different shapes.

Moreover, tessellation is the precise term we should use when we want to cover the area with no gaps nor overlaps, *i.e.* to not miss out any spatial data point in the area.

B. Spatial Data Mining

Spatial data mining is the application of data mining methods to spatial data. The end objective of spatial data mining is to find patterns in data with respect to geography.

One of the major challenges in spatial data mining is that geospatial data repositories tend to be very large, due to the overwhelming amount of features and attributes. Algorithmic analysis is needed to obtain insights from such data. However, due to the random way of how nature crafts the geospatial landscapes, and stochastic human behavior, spatial data could never be uniformly distributed across any given area. Thus, if we aggregate spatial data to uniform tiles with same size and shape, it usually reduces the efficiency of the algorithms, and could waste large amount of computational power and resources.

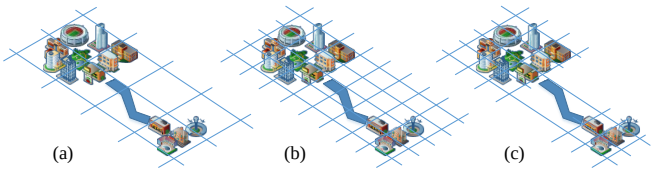


Fig. 1. Fixed gridding vs. adaptive tessellation.

If we use only large fixed grids, as in Fig. 1(a), details in the city region cannot be observed closely. In contrast, if we use small fixed grids as in Fig. 1(b), although we have more details in the city region, effort will be wasted when we study those grids with only road or empty lands. Therefore, the ideal case is to use adaptive tessellation, as in Fig. 1(c), where the tiles can be adaptive to the geospatial features, meaning smaller for places with more details, larger when there are less interesting information.

It is clear that uniform or fixed tessellation mapping is not suitable for spatial data mining, and it calls for the need of an adaptive solution. This motivates us to invent the *Adaptive Tessellation Mapping (ATM)* scheme.

III. Adaptive Tessellation Method (ATM)

The basic idea of ATM is to split the area into tiles with same shape but variable sizes. In particular, we start by splitting the area into *base tiles*, which are the tiles with the maximum size, defined as *base size*. Base tiles are further split into *sub-tiles*, which are smaller in size, but the same shape as base tiles. A sub-tiles can be further divided into smaller sub-tiles, until certain requirement is satisfied.

To present the ATM scheme clearly, we will start with the basic geometric requirements and properties, to the algorithm that constructs the adaptive tiles, and then move on to some challenging issues with ATM in this section.

A. Basic Geometry Requirement

Being adaptive is different from being random. In fact, split the area into random regions could be even more costly than uniform tessellation. There need to be some basic principles that guide the design of ATM — in particular, the tile shape.

1) Simplicity

The purpose of ATM is to simplify the spatial data mining process and make the knowledge extraction more efficient. Therefore the tessellation process itself must be simple. Moreover, the outcome tiles needs to be simple, too, so that the boundaries and regions of the tiles could be easily defined in computer programs.

Thus we restrict our discussion of tile shapes to *regular polygons*, which is defined as a polygon that is equiangular (all angles are equal in measure) and equilateral (all sides have the same length).

2) Similarity

In order to apply same algorithm to different tiles, we need the tiles to be *similar*. In geometry, similarity is defined when Two geometrical objects both have the same shape, or one has the same shape as the mirror image of the other. More precisely, one can be obtained from the other by uniformly scaling (enlarging or shrinking), possibly with additional translation, rotation and reflection. This means that either object can be rescaled, repositioned, and reflected, so as to coincide precisely with the other object.

As a result, the base tile needs to be of the same shape, and the tessellation must be a *regular tessellation*: a highly symmetric, edge-to-edge tiling made up of regular polygons, all of the same shape. There are only three regular tessellations: those made up of equilateral *triangles*, *squares*, or *regular hexagons*. In another word, the base tiles (as well as all the sub-tiles) must be of one of these three shapes.

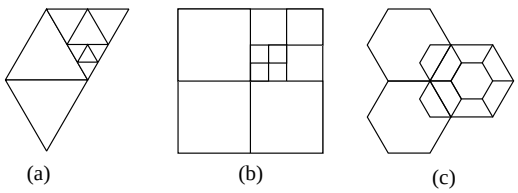


Fig. 2. Triangle, square and hexagon tessellation.

Fig. 2 shows the three types of regular tessellation.

3) Hierarchy and inheritance

We design ATM in a way that the smaller tiles *inherit* shape from the bigger ones, and a hierarchical structure is preserved when we zoom in or zoom out.

In Fig. 2(c), we can see that a hexagon tile cannot be split into a group of hexagons, meaning that it is not possible to maintain a hierarchical structure with hexagonal tessellation. Therefore we will only discuss about triangular and square

tessellation in this paper.

The main reason we use a hierarchical structure is to enable the use of recursive algorithm. When tiles are of the same shape but different size, the same algorithm can be applied to all the tiles. When we split the area into tiles, recursive algorithm can be used to further split big tile to smaller once till certain criteria is met. We will demonstrate the algorithm itself in next Section.

Also, we found that when tiles of the same shape is used in the tessellation, it is visually clear and pleasant. Consistency can be maintained when we focus on different part of the area.

In this paper, we only discuss the case where each tile is split into 4 sub-tiles. It is possible to split a square tile for more sub-tiles, such as 9, 16 or any square number. The same principle discussed in this papers can be extended to those scenarios easily.

B. Algorithm

The algorithmic challenges in ATM include three parts: how to address the tiles, how to create them, and how to split them into sub-tiles.

1) Data model and structure

To efficiently represent the hierarchical structure between a base tile and its sub-tiles, a *tree* data structure is the most suitable. Each sub-tile will be represented as a child of its parent, as shown in Fig. 3. Each base tile can be represented as the root (level 1) of a tree, and the tessellation of the entire area can be seen as a *forest* of such trees, as in Fig. 3(a).

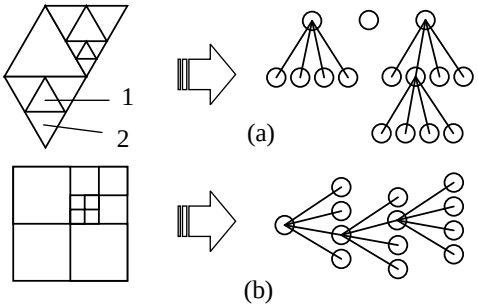


Fig. 3. Representing tiles as tree structure.

In most existing spatial data schema, there are at least two parts: the locations of data points, and the properties of the data points. Usually, the locations are addressed by the longitude (*x* coordinate) and the latitude (*y* coordinate). In some modern data base with spatial API, such location can be stored as a Point data type in one column. In this paper, without losing generality, we use two columns to store the (*x*, *y*) tuple. The properties of the data points, such as temperature, building type, population *etc.* are depicted by additional columns. When they are aggregated to tiles, extra columns need to be created to address which tile they belong to. We refer to such columns as *tile info columns.*

For square tiles, three additional columns will be needed: the *centroid* coordinate (two columns), and the *level* of the tile in the tree. Given the size of base tile, it is relatively easy to directly calculate the boundaries of the tile.

For triangular tiles, one more column will be needed to denote the orientation: whether the triangle is point upwards (as triangle 1 in Fig. 3(a)) or downwards (as triangle 2 in Fig. 3(a)).

2) Base tile creation

Creating base tiles, is actually equivalent to dividing the given area into regions of same size and shape (square or triangle). In the sense of data manipulation, it is to define the tile info columns based on the tile location columns, *i.e.* to define centroid coordinates, denoted as (x_c, y_c) and orientation (for triangular tiles only) of the tiles. The level of a base tile will always be 1, as they are always the root of a tree; and the size will always be base size, denoted as S_b. We need to design algorithm to solve the following problem:

Base Tile Creation Problem: Given coordinates of a given data point (x, y) and base size S_b, find the centroid coordinate (x_c, y_c) and orientation z (for triangular tiles) of the tile that his data point belongs to.

Due to the page limit constraint, the derivation of following results are omitted.

a) Square tiles

It is relatively easy to obtain the centroid of the square tile.

$$x_c = \lfloor x/\sqrt{S_b} \rfloor \sqrt{S_b} + \sqrt{S_b}/2$$
$$y_c = \lfloor y/\sqrt{S_b} \rfloor \sqrt{S_b} + \sqrt{S_b}/2$$

b) Triangular tile

For triangular tiles we use a to represent the edge length of the tile. We have

$$S_b = \sqrt{3}a \Rightarrow a = 3S_b/\sqrt{3}$$

It is easier to calculate the centroid of the tile if we do a *coordinate transformation* first, as shown in Fig. 4:

$$x^{\Delta} = x - \sqrt{3}y/3$$
$$y^{\Delta} = 2\sqrt{3}y/3$$

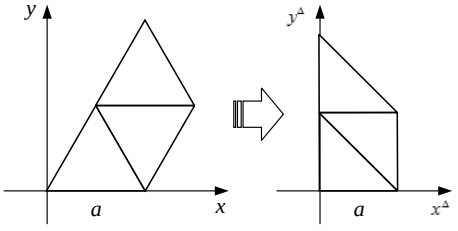


Fig. 4. Coordination transformation for triangular tessellation.

We calculate z, the orientation of the tile first

$$z = \begin{cases} 1 & \text{if } \left(x^{\Delta} - \lfloor x^{\Delta}/a \rfloor a\right) + \left(y^{\Delta} - \lfloor y^{\Delta}/a \rfloor a\right) \le a \\ -1 & \text{if } \left(x^{\Delta} - \lfloor x^{\Delta}/a \rfloor a\right) + \left(y^{\Delta} - \lfloor y^{\Delta}/a \rfloor a\right) > a \end{cases}$$

The centroid of the tile in the transformed coordinate could then be obtained

$$x_c^{\Delta} = \lfloor x^{\Delta}/a \rfloor a + a/2 - az/6 = \left(\lfloor x^{\Delta}/a \rfloor + 1/2 - z/6\right)a$$
$$y_c^{\Delta} = \lfloor y^{\Delta}/a \rfloor a + a/2 - az/6 = \left(\lfloor y^{\Delta}/a \rfloor + 1/2 - z/6\right)a$$

Finally we transform the coordinate back to the normal:

$$x_c = x_c^{\Delta} + \sqrt{3}y_c/3 = x_c^{\Delta} + y_c^{\Delta}/2$$
$$y_c = \sqrt{3}y_c^{\Delta}/2$$

3) Sub-tile split

The decision of whether or not to split a tile to sub-tiles depends on certain property that we could observe from the tile, referred to as the *feature.* For example, if we want to restrict that there should be less than or equal to 100 buildings in a single tile, the building count will be the feature, and any tile covering area with more than 100 buildings, being *lack of feature*, will be split into sub-tiles. This split process can be recursive, as depicted in Algorithm 1 with pseudo code.

Algorithm 1. Create Adaptive Tessellation

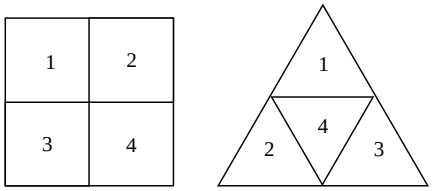
```
procedure CREATE_AT(area A)
    split A into base tiles
    for each base tile T
        if T is lack of feature
            CREATE_SUBTILE(T)
        end if
    end for
end procedure

procedure CREATE_SUBTILE(tile T)        \\a recursive procedure
    if T is lack of feature
        split T into sub-tiles sT[1…n]
        for each sub-tile sT[i]
            CREATE_SUBTILE(sT[i])       \\recursion
        end for
    end if
end procedure
```

To split a tile of size S to sub-tiles of size S', we always have

$$S' = S/4 = S_b/4^{l-1}$$

where l is the level of the parent tile. The level of sub-tile will be l+1.

a) Square tile



(a) (b)

Fig. 5. Sub-tiles.

As shown in Fig. 5(a) the centroid (x'_c, y'_c) of the sub-tiles i (where i=1, 2, 3, 4) can be derived:

$$i = \begin{cases} 1 & x < x_c, y > y_c \\ 2 & x > x_c, y > y_c \\ 3 & x < x_c, y < y_c \\ 4 & x > x_c, y < y_c \end{cases} \quad x'_c = \begin{cases} x_c - \sqrt{S}/4 & i = 1,3 \\ x_c + \sqrt{S}/4 & i = 2,4 \end{cases} \quad y'_c = \begin{cases} y_c - \sqrt{S}/4 & i = 1,2 \\ y_c + \sqrt{S}/4 & i = 3,4 \end{cases}$$

If a point falls on the boundary, we randomly assign it to either tile, for simplicity.

b) Triangular tile

Again we use a to represent the length of the tile edge

$$a = 3S/\sqrt{3}$$

To determine which sub-tile each data point (x, y) falls into, we calculate i, as shown in Fig. 5(b).

$$i = \begin{cases} 1 & y > y_c + \sqrt{3}a/12 \\ 2 & \sqrt{3}x + y < \sqrt{3}x_c + y_c - \sqrt{3}a/6 \\ 3 & \sqrt{3}x - y < \sqrt{3}x_c - y_c - \sqrt{3}a/6_c \\ 4 & \text{otherwise.} \end{cases}$$

The centroid (x'_c, y'_c) of the sub-tiles i (where i=1, 2, 3, 4) can be derived

$$x'_c = \begin{cases} x_c & i = 1 \\ x_c - a/4 & i = 2 \\ x_c - a/4 & i = 3 \\ x_c & i = 4 \end{cases}$$

$$y'_c = \begin{cases} y_c + \sqrt{3}a/6 & i = 1 \\ y_c - \sqrt{3}a/12 & i = 2 \\ y_c - \sqrt{3}a/12 & i = 3 \\ y_c & i = 4 \end{cases}$$

The orientation z is straight forward as shown in Fig. 5(b).

$$z = \begin{cases} 1 & i = 1,2,3 \\ -1 & i = 4 \end{cases}$$

C. Challenges

There are several challenges in ATM we want to present here as some future work direction.

1) Base size

Base size determines the "worst" resolution of ATM. Smaller base size offers more details to the entire area, but if they are too small to be further split down, it becomes equivalent to fixed size gridding or meshing.

Big base size can always be split to small ones depends on the desired features. Therefore, the most adaptive way is to have a base tile cover the entire area, and split it down to sub-tiles. However, this could also produce too many levels in the tree structure, and make the ATM algorithm less efficient.

Therefore it is challenge to determine just the optimal size of the base tile, especially when we start without any knowledge about the data set.

2) Query

We have adopted the tree structure from [4] to address the tiles. However, given any random location. It is computationally costly to find out which tile it belongs to.

It will need either go through the entire tiles list to find the

tile covering this location, or go through the tree structure to identify the corresponding tile. The complexities of these two solutions depend on the number of tiles and the levels of the trees.

IV. Use Cases

As we have discussed before, ATM is a new and useful way to aggregate spatial data for mining purpose. In this Section, we present two very simple use of ATM, for visualization and behavior classification.

Both examples have been implemented as working solutions. However, due to the page limit, we could not go into the details of these case studies. We will try to publish them as stand-alone papers.

A. Visualization

Display two features on the same heat map. Using ATM to show density, and another feature as an overlay top of the tiles.

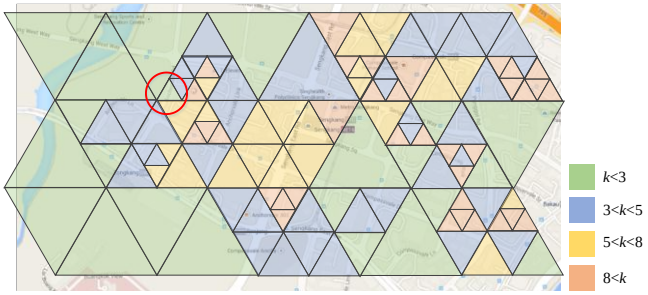


Fig. 6. Density of kids vs. children facilities.

Fig. 6 shows a map with number of kids and number of children facilities (including kindergarten, playgrounds and kids' clinics).

The feature of ATM in Fig. 6 is the number of kids. We split the sub-tiles in a way that each triangular tile will have maximum 10 children younger than 10 years old registered as residents. The smaller the tile is, the higher the density of kids is.

The color (heat) of the map indicates the number (k) of children facilities. By right, the higher the children density, the more number of facilities should be in the tile and vice versa. However, we can see there is one small tile on the upper left (circled out) with green color, which could indicate in sufficient facilities for children. Other than this part, we should say this area is planned quite well because most of the places have facilities that proportional to the density of children.

Using traditional heat maps, it will be more difficult to compare these two features in parallel. Especially when the tile with issue is so small compare to the entire map. With ATM, it adds one more dimension to the heat map so that two features can be displayed simultaneously and the problematic place just pops out.

B. Behavior Classification

We can also use ATM to differentiate the places where people move similarly (like on a long straight pedestrian way) and a places where people move differently (like a crossroad). As shown in Fig. 7.

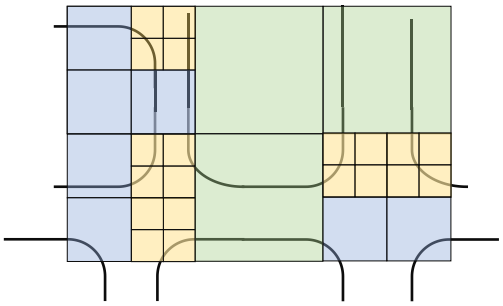


Fig. 7. Using ATM to classify pedestrian behavior.

The feature in this ATM map is the variance of people's behavior (walking straight or making turns). Tiles are split to sub-tiles until people behave similarly in one tile (*i.e.* various is lower than a threshold).

With ATM we can ignore those large tiles when we want to classify people. Because in those tiles people all behave similarly and could not differentiate themselves. One the other hand, we should focus on those smaller tiles (marked with yellow in Fig. 7). We can identify different people (or groups of people) by the sequence that they visit the yellow tiles, which forms their distinct movement patter that different with others.

V. Conclusion

In this paper, we propose the *Adaptive Tessellation Mapping (ATM)* method, which allow us to aggregate the spatial data into adaptive tiles and enable more straight forward visualization and flexible data mining. We have derived coordinates of tile given any data point, and designed a recursive algorithm to split tile to sub-tiles based on certain desired feature. Two use cases are also briefly discussed to show that ATM works effectively with real life data.

References

[1] I. Lee, R. Pershouse, and K. Lee, "Spatial cluster tessellation through the complete order-k Voronoi diagrams," *Spatial Information Theory*, Springer Berlin Heidelberg, 2007, pp. 321-336.

[2] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," *VLDB*, vol. 97, 1997.

[3] C. A. White, *A history of the rectangular survey system*, US Department of the Interior, Bureau of Land Management, 1983.

[4] J. Hershberger, N. Shrivastava, S. Suri, and C. D. Toth, "Adaptive spatial partitioning for multidimensional data streams," *Algorithmica*, vol. 46, no. 1, pp. 97-117, 2006.

[5] G. Dutton, "Encoding and handling spatial data with hierarchical triangular meshes," in *Proc. 7th International Symposium on Spatial Data Handling*, vol. 43, Netherlands: Talor & Francis, 1996.

Wang Ting was born in 1983 in Chengdu, Sichuan, China. He came to Singapore for his undergraduate studies under Singapore Government-Linked Company SM2 scholarship in 2001. He obtained his bachelor's degree with honors in 2005 and subsequently Ph.D in 2011 both at Nanyang Technological University (NTU), Singapore.

He worked as a demand planner at Apple South Asia and joined SAP as a data scientist in 2012. His research interests include data mining, mathematical modeling and algorithmic optimization.

Dr. Wang loves soccer. He considers family as his greatest award. He has a son, and he is a good cook — said his wife.