AN ABSTRACT OF THE THESIS OF

Stanislav I. Trubin for the degree of Master of Science in Computer Science presented on June 14, 2006.

Title: Information Space Mapping with Adaptive Multiplicatively Weighted Voronoi Diagrams.

Abstract approved:

_____

E. N. Mortensen R. F. Reitsma

Traditional application of Voronoi diagrams for space partitioning creates Voronoi regions, with areas determined by the generators' relative locations and weights. Especially in the area of information space (re)construction, however, there is a need for inverse solutions; *i.e.*, finding weights that result in regions with predefined areas. In this thesis, an Adaptive Multiplicatively Weighted Voronoi Diagram solution is formulated and a raster-based optimization method for finding the associated weight set is proposed. The basic algorithm is described, and several improvements are explored in detail, followed by algorithm's complexity analysis. The adaptive solution is successfully tested is successfully tested on a series of ideal/pathological cases, as well as using empirical data.

# Information Space Mapping with Adaptive Multiplicatively Weighted Voronoi Diagrams

by

Stanislav I. Trubin

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science

Presented June 14, 2006

Commencement June 2007

Master of Science thesis of Stanislav I. Trubin presented on June 14, 2006.

APPROVED:

---

Co-Major Professor, representing Computer Science

---

Co-Major Professor, representing Computer Science

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

---

Stanislav I. Trubin, Author

# ACKNOWLEDGEMENTS

The author expresses sincere appreciation to the University of Colorado's Integrated Teaching and Learning Program and the US Department of Education's Fund for the Improvement of Post Secondary Education (FIPSE) for partly funding this research.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

# LIST OF FIGURES

Figure Page

LIST OF FIGURES (Continued)

# LIST OF TABLES

# LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| A | Total image size |
| $A_j$ | Goal area for region $r_j$ |
| $a_{i,j}$ | Calculated area of region $r_j$ at iteration $i$ |
| AMWVD | Adaptive Multiplicatively Weighted Voronoi diagram |
| CVD | Centroidal Voronoi diagram |
| D | Dimensionality |
| $E_{i,j}$ | Mean error of region $r_j$ at iteration $i$ |
| $E_i$ | Average error across $N$ regions at iteration $i$ |
| I | Total number of iterations |
| k | Operator that controls amount of changes done to $w_{i,j}$ |
| $k_{mult}$ | Multiplier of factor $k$ |
| MWVD | Multiplicatively Weighted Voronoi diagram |
| N | Number of regions in a Voronoi Diagram |
| r | Correlation |
| $r_j$ | $j^{th}$ Voronoi region |
| S | Number of sub-pixel levels along one of the axis |
| SVD | Standard Voronoi diagram |
| VD | Voronoi diagram |
| $w_{i,j}$ | Weight of region $r_j$ at iteration $i$; $i \in [1, \ldots, N]$ |
| X | Horizontal image resolution in pixels |
| Y | Vertical image resolution in pixel |

# Chapter 1: Introduction

Because of the large and complex volumes of information in today's world, visualization and categorization are two rapidly growing research areas. As the Internet grows, the amount of information available grows too, requiring new methods for data analysis and visualization. Information visualization comprises the display of large amounts of information in a visual format. A good visualization helps the human mind to process large quantities of information; *i.e.*, understand it, and its internal relationships. Some of these visualizations involve region or space partitionings according to some predefined values. Even though numerous visualizations methods that include partitioning have been developed, most of them have shortcomings such as limited applications or lack of certain characteristics. This can be particularly observed in the area of inversed solutions; *i.e.*, finding a set of values that result in regions with predefined areas.

Voronoi diagrams are widely used for space partitioning. Our main goal is to present a new mapping approach by using a new type of Voronoi Diagram: the so-called Adaptive Multiplicatively Weighted Voronoi Diagram. This diagram is an inverse version of Multiplicatively Weighted Voronoi Diagram.

In this thesis we start by reviewing several existing methods that divide information spaces into regions the size of which is proportional to some known value: ET-Map, rectangular space partitioning and the InfoSky power diagram. We demonstrate that

although all of them have some attractive characteristics, they have some conceptual problems as well.

Next, we offer an alternative solution based on an adaptive version of the Multiplicatively Weighted Voronoi Diagram. We give an overview of both the basic and advanced versions of the adaptive algorithm; the advanced version involving speed and precision improvements. The worst-case complexity of the algorithm is discussed followed by several performance measurements.

Finally, we run various test cases, which demonstrate the algorithm's performance under various scenarios and its ability to solve the problem successfully.

## Chapter 2: Background

Today's volumes of information that we have access to may become so large and complex, humans may lose track of it all, e.g. a bookcase with books in random order, or a retail store with many products in stock. Categorization of information is necessary in these situations; libraries group books using the Dewey decimal system, and stores place similar products together on the same shelf giving them a common group name; e.g. vegetables, hardware, kitchen supplies, etc. The Internet faces similar challenges with regards to information presentation. With the growing amount of information available on the Internet [5, 8, 9], new methods for collecting, processing, categorizing data and delineating of information space are needed [1, 8, 9, 17, 19, 22, 23]. Information visualization is a method of presenting large sets of data or information in non-traditional, interactive graphical forms. Visualizations can show the structure of information, allow one to navigate through it, and help evaluate the content. Information interpretation is often challenging because of multidimensionality of the information. Even though information visualization usually involves two or three-dimensional graphics, it can represent information about additional dimensions. For instance, consider displaying a temperature distribution in a volume. This is four-dimensional data (x-location, y-location, z-location and temperature), which can be displayed on a three-dimensional graphic using color to represent temperature. Here, the temperature information (physical attribute) is converted into a visual attribute – color.

## 2.1 Information Space Tessellations

Numerous visualization methods involve partitioning; *i.e.*, the allocation of each point in a space to an information item in that space. These methods are often referred to as 'information space tessellations.' Maps of information space are sometimes called spatializations; spatial representations of a phenomen the spatial characteristics of which cannot be directly observed [11]. The cartographer must decide on the fundamental attributes of the space: its dimensionality, metric, item position, *etc*. Figure 1 shows an implementation of tree maps [13, 20] by Chen *et al*. [6]. The map shows a two-dimensional categorization of approximately 110,000 web links into rectilinear regions. Each region represents links that are similar to each other in terms of content; an artificial intelligence algorithm determines the similarity between the links. The number of links corresponding to a region determines its area. The space is hierarchical; each of the regions can again be subdivided into a map that shows category details.

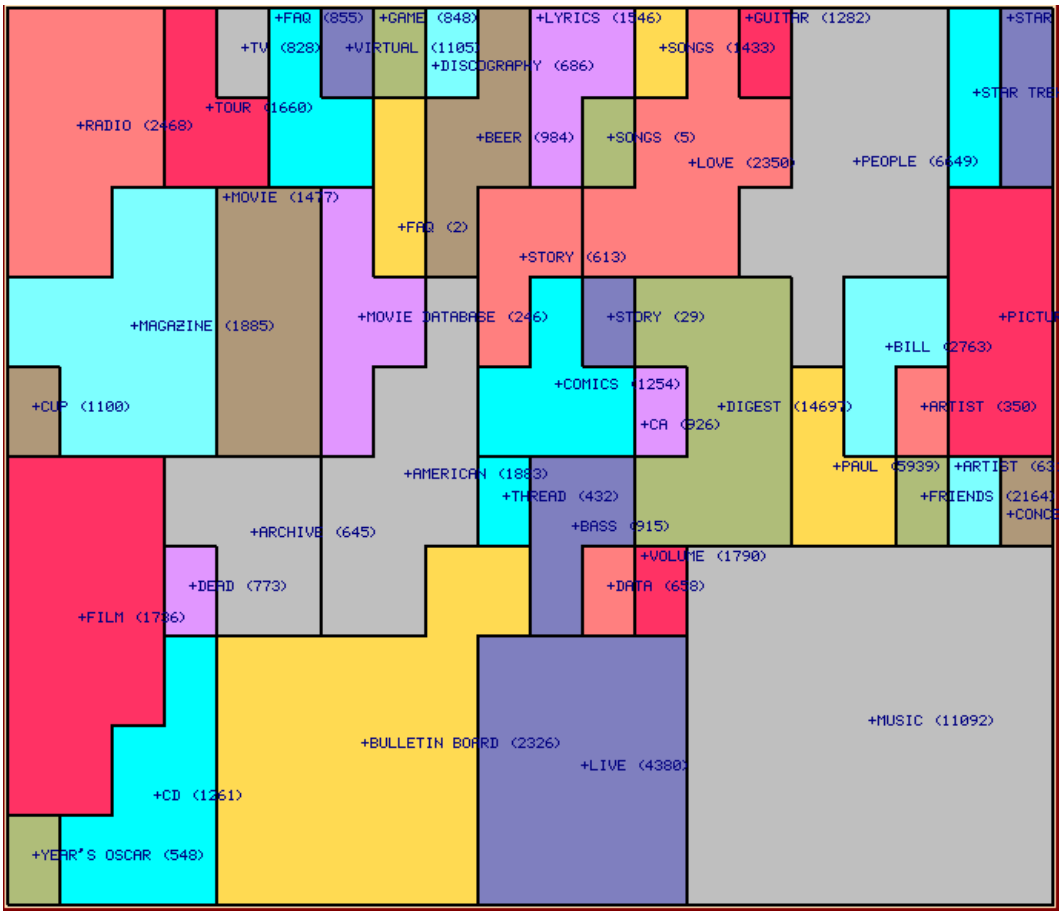



*Figure 1: ET-Map by Chen et al. [6].*

Figure 2 shows a rectilinear partitioning that is made available for commercial licensing by SmartMoney.com [24]. It shows the stock market for about 500 companies over a 26-week period. Each rectangle's area reflects a company's current market capitalization. Price change is indicated by the rectangle's color, which varies from dark blue to bright yellow. Yellow denotes a stock price increase, blue represents a decrease, black represents no change, and gray indicates the unavailability of data (in the case of the gray scale printed copy of this document, yellow regions are bright, blue regions are dark). When the mouse cursor is placed over one of the rectangles, a popup window shows the latest price and price change over the specified period of

time. The advantage of this map is the fast visual analysis of the market it provides. One can quickly look at the map and see that particular types of stocks have been going down, while others have gone up.

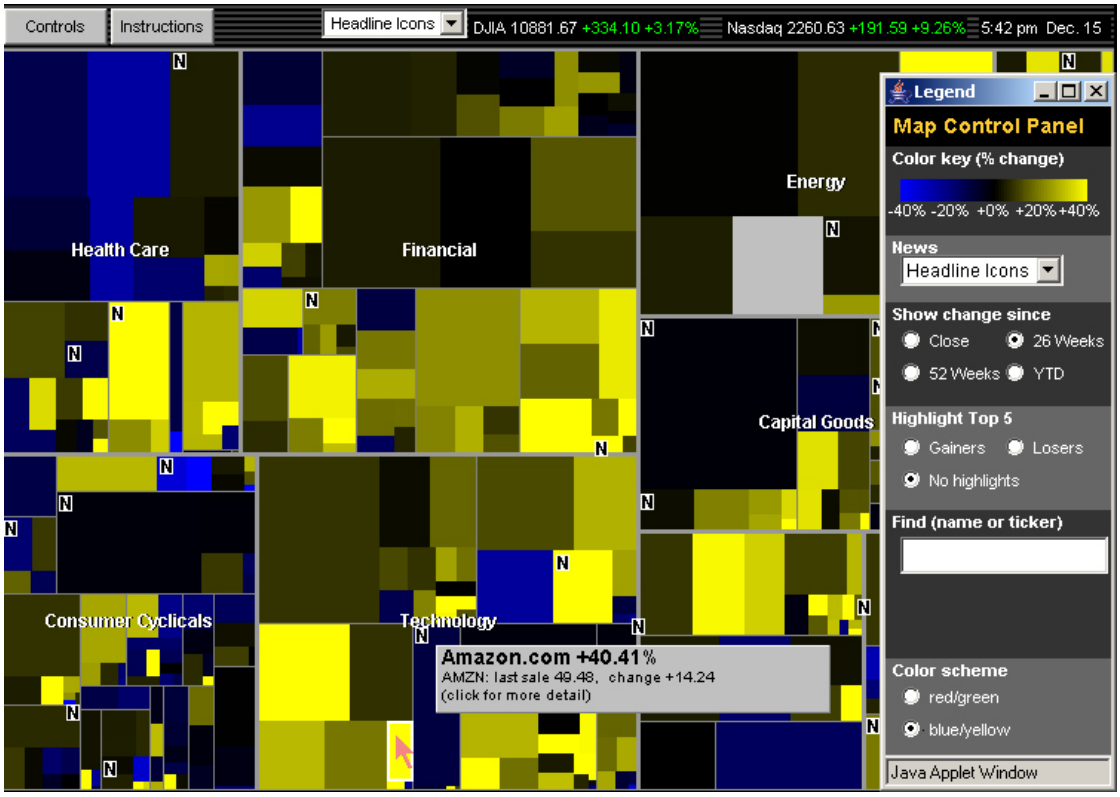



*Figure 2: Rectangular space partitioning of the US stock market by SmartMoney.com [24].*

Figure 3 shows yet another example of rectangular space partitioning made by WebMap Technologies [29]. The idea of a WebMap is very similar to SmartMoney.com's rectangular partitioning. Again, the size of a rectangular region is proportional to its magnitude.

All three cases (Figure 1–3) are based on the spatial metaphor: items that are closely related to each other in an information system are placed close to each other on the resulting spatialization.

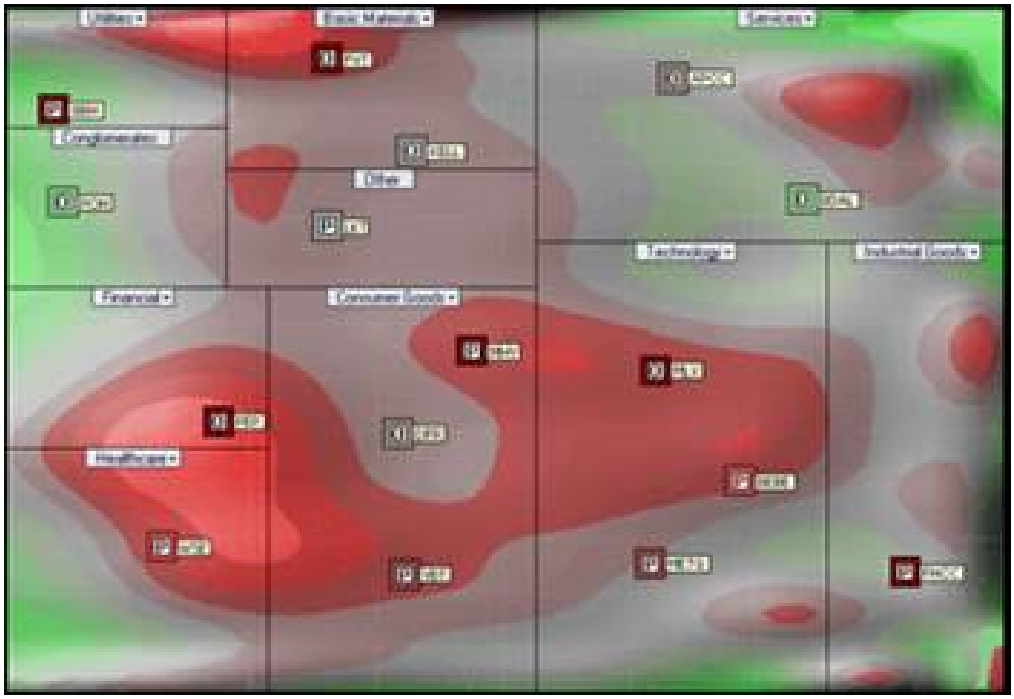


*Figure 3: Rectangular information space partitioning by WebMap Technologies Inc.*

## 2.2 InfoSky Power Diagram

Andrews *et al*. [1] present a tessellation method for newspaper articles into a two-dimensional Euclidean space. Each region represents clusters of news articles and the size of each region is approximately proportional to the number of articles it contains. The tessellation process is performed using a variation of a Voronoi diagram known as a Power Voronoi Diagram (PVD) [16]. The result is the information space visualization shown in Figure 4.

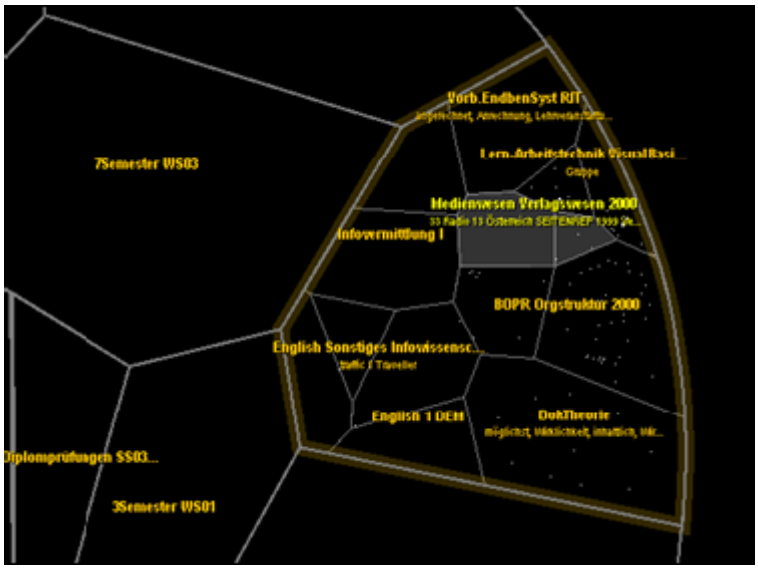



*Figure 4: InfoSky information space visualization by Andrews et al. [1].*

Andrews *et al*. specify the following constraints for a finite Euclidean orthogonal information space $A_D$ with dimensionality $D$ containing $N$ points, and generators $g_i$ with weights $w_i$ into $N$ regions of $r_i$, each with area $a_i$:

- Constraint (1): inclusiveness I: $g_i \in r_i$
- Constraint (2): $r_i$ is convex
- Constraint (3): proportionality: $a_i \propto w_i$
- Constraint (4): inclusiveness II: $a_i >$ threshold

Constraint (1) states that each generator $g_i$ is located inside of a region $r_i$. Constraint (2) states that the regions are convex polygons. Constraint (3) states that the region area $a_i$ must be proportional to its weight $w_i$. Constraint (4) says that the regions should be larger than some user-defined minimum threshold.

Let us define $g_i$ with $i \in [1, ..., N]$ as so-called generators, points in space for which we must compute PVD regions. The Power Voronoi Diagram region $r_i$ for a corresponding generator $g_i$ in a set of $N$ generators is then defined as:

$$r_i = \{x \mid \|x-x_i\|^2 - w_i \leq \|x-x_j\|^2 - w_j, i \neq j\} \quad (2.1)$$

where $x$, $x_i$ and $x_j$ are location vectors with $i$ and $j \in [1, ..., N]$. There exist other types of Voronoi diagrams that have different properties than PVDs. The two most popular types of Voronoi diagrams are the Standard Voronoi Diagram and the Multiplicatively Weighted Voronoi Diagram.

## 2.3 Standard Voronoi Diagram

The Standard Voronoi Diagram (SVD) is also known as the unweighted, ordinary or classic Voronoi diagram. A Voronoi region $r_i$ for a corresponding generator $g_i$ in a set of $N$ generators is defined as:

$$r_i = \{x \mid \|x-x_i\| \leq \|x-x_j\|, i \neq j\} \quad (2.2)$$

where $x$, $x_i$ and $x_j$ are location vectors with $i$ and $j \in [1, ..., N]$. In other words, region $r_i$ is the set of all points $x$ which are closer to generator $g_i$ than to any other generator $g_j$. The set of all regions $r_i$ makes a Voronoi diagram. Figure 5 shows an example of a two-dimensional standard Voronoi diagram with ten randomly positioned generators. The lines of equilibrium (borders) separating the regions are straight lines that orthogonally and equally bisect the imaginary lines connecting neighboring

generators. These imaginary lines together form the so-called ‘Delaunay triangulation.’ The Standard Voronoi Diagram and Delaunay triangulation are inverses of each other; if one of them is known, the other one can be computed.

The Voronoi diagram can also be interpreted in a dynamic way. The dynamic interpretation of an SVD is defined as all generators simultaneously extending their reach in every direction at identical speeds. By their nature, Voronoi diagrams have an infinite extension. While there are ‘internal’ regions whose growth is limited by surrounding regions, the peripheral regions themselves can grow infinitely into one or more directions. Figure 5 has three ‘internal’ regions and seven ‘peripheral’ regions. Two common methods are used to limit the extent of peripheral regions: disposable generators and ‘spatial extent.’ Disposable generators are placed around existing peripheral generators so that they create new peripheral regions; only internal regions are considered in this case. The ‘spatial extent’ method does not modify the set of generators itself, but rather clips the viewing plane of our concern. Figure 4 and 5 are the examples of a Voronoi diagram in a viewing plane.

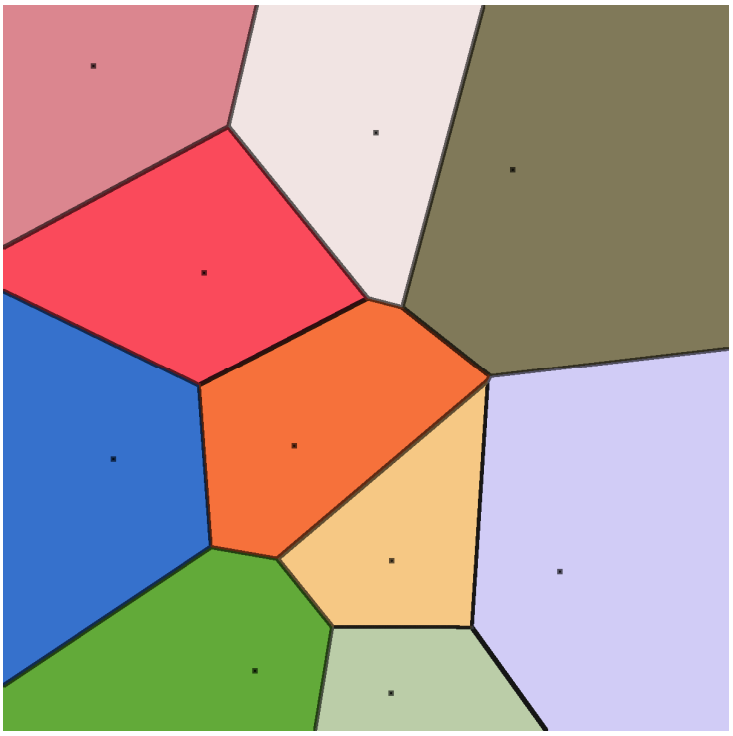


*Figure 5: Standard Voronoi Diagram.*

The concept of a Standard Voronoi Diagram has been discovered in the natural world many times in different sciences. In biology, space boundaries occupied by trees in a forest form in patterns much like an SVD [14]. The same happens in metallurgy with the crystallization of metal films, known as “grain growth” [30]. Even the social sciences show instances of SVDs, such as in psychology, dealing with concepts of personal space [25].

The Standard Voronoi Diagram can be solved analytically. However, other kinds of Voronoi diagrams are solved using iterative methods. One example of an iterative Voronoi diagram is a Centroidal Voronoi Diagram (CVD), which is an SVD, but one which involves moving the generators between solution steps. A single iteration loop consists of two steps: a) solving the SVD, and b) moving generators to the center of

mass of their region. Figure 6 shows two-dimensional Centroidal Voronoi diagrams for a constant probability density function in $[-1, 1]^2$ for $x$- and $y$- positions of the generators. The top row shows diagrams with 64 generators; the bottom row shows diagrams with 256 generators. CVDs have been used in various applications, such as the 'relaxation' of images to create stipple drawings [7], uniform distribution of plants and others.

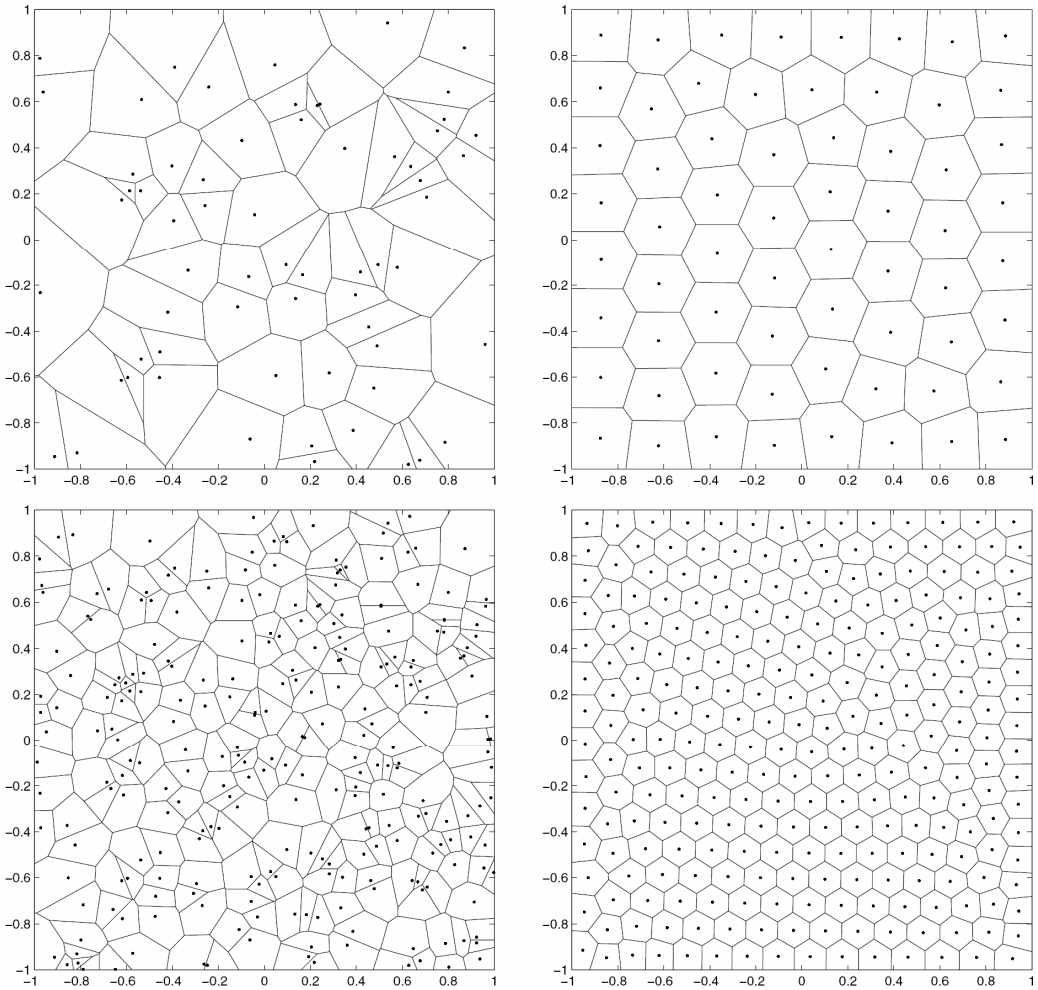


*Figure 6: Centroidal Voronoi Tessellations by Du et al. [10]. Left column: Monte Carlo simulation; right column: Centroidal Voronoi diagrams. Top row: 64 generators; bottom row: 256 generators.*

## 2.4 Multiplicatively Weighted Voronoi Diagram

A Multiplicatively Weighted Voronoi Diagram (MWVD) advances the idea of a Standard Voronoi Diagram by associating a weigh $w$ with each region. An MWVD is defined as:

$$r_i = \{x \mid \frac{\| x - x_i \|}{w_i} \leq \frac{\| x - x_j \|}{w_j}, i \neq j\} \tag{2.3}$$

where $x$, $x_i$ and $x_j$ are location vectors with $i$ and $j \in [1, ..., N]$. $w_i$ and $w_j$ are the weights (magnitudes) [15, 16] of corresponding regions.

An SVD is equal to an MWVD with the weights of all regions equal. The dynamic interpretation of a MWVD is that all generators start simultaneously to grow, but they grow at different rates represented by their weights. If $w_i \neq w_j$ the borders of the Voronoi regions of an MWVD are not straight lines. In fact, they are arcs of so-called *Apollonius circles* [16]. Figure 7 shows an MWVD using the same generators' locations used in Figure 5 but with randomly selected weights.

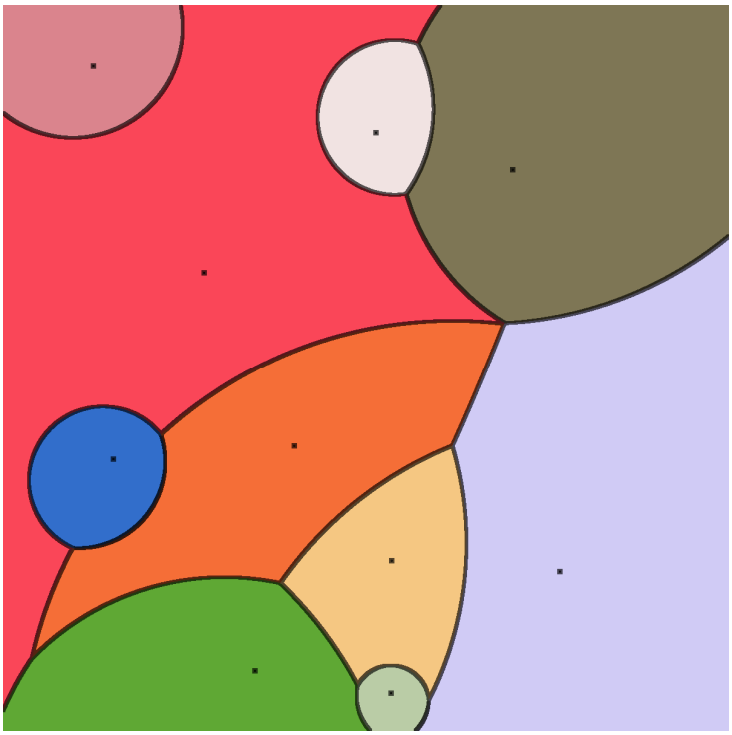


*Figure 7: Multiplicatively Weighted Voronoi Diagram.*

Regions can be disjointed in an MWVD (not the case on Figure 7). If the spatial extent is unbounded, the region with the highest weight 'surrounds' all the other regions and is infinitely large. If two or more have the same highest weight then most or all of them may have an infinite size, depending on their relative location to each other.

Figure 8 shows a top-view picture of homemade bread. Dough pieces of slightly different sizes have been placed on the glass tray before cooking. While baking, the pastries grow uniformly in each direction and their speed of growth is proportional to their initial mass. After forming borders with their neighbors, the dough pieces start to grow up out of the pan. Therefore, Figure 8 shows a good example of an MWVD.

*Figure 8: Multiplicatively Weighted Voronoi Diagram in real life.*

Figure 9 shows a map of the USA divided into urban spheres of influence [12]. Major cities are used as region generators and their populations are used as magnitudes (weights) to computer their service areas.

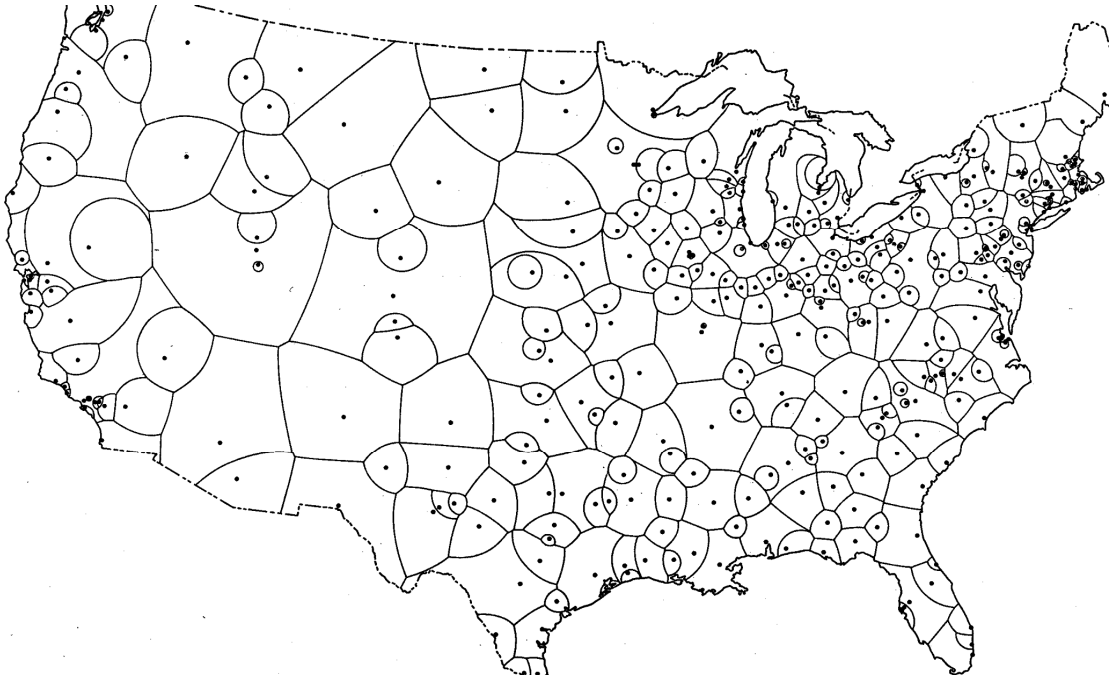


*Figure 9: The US map divided into urban spheres of influences by Huff [12].*

The previously presented Standard, Centroidal and MW Voronoi diagrams are two-dimensional, but conceptually the extension to multidimensional diagrams is straightforward. Peter Gustav Lejeune Dirichlet presented the first formal presentations of standard Voronoi diagrams in 1850 for two- and three-dimensional cases and called them *domaine de Dirichlet* or Dirichlet domains. However, Georgy Fedoseevich Voronoi was the first who examined a general *D*-dimensional space in 1907-09 [26, 27, 28].

## 2.5 Comparing Information Space Tessellations

Constraint (4) forces each region to be larger than some minimum threshold. We suggest replacing this with constraint (4′):

- Constraint (4′): inclusiveness II: $\Sigma a_i = A_D$,

which states that the entire space must be partitioned, not just portions of it. Without constraint (4′), one can always meet the requirements from constraints (1) through (3) by choosing an arbitrarily small threshold.

The method described by Andrews *et al*. may require moving generators. This is due to a property of PVDs, in which a very large weight difference between two generators which are close to one another, can cause the generator with the smaller weight to get located on the wrong side of the bisector and hence outside its own area. To avoid this behavior, Andrews *et al*.'s algorithm moves the generators until they are inside of their region. However, moving generators is unacceptable in many applications, e.g. Huff's [12] studies urban spheres of influences of major US cities (Figure 9). Of course, moving cities on the US map is unacceptable, as it invalidates the original data. Here we can say that our generators' positions are a constraint, while region boundaries are the dependable variable. To reflect that property, we suggest adding the constraint:

- Constraint (5): $\Delta g_i = 0$

which states that the location of a generator cannot change.

As reflected in a PVD, constraint (3) states that the area of a region must be proportional to its weight. Rectangular space partitionings exhibit a strong relationship between region area and magnitude. As a result, we suggest adding a new constraint (6), which is based on constraint (3):

- Constraint (6): proportionality II: $a_i = A_i$

where $A_i$ is a desired area proportion of a total diagram area $A$. Constraint (6) suggests that the size of the regions must be equal to their predefined area proportions.

It is now possible to compare ET-Map, rectangular space partitionings, SVD, MWVD, CVD, PVD and InfoSky space partitioning relative to constraints (1) through (6) as shown in Table 1.

*Table 1: Comparing partitioning methods relative to constraints (1) through (6).*

| Partitioning method | Constraint | | | | | | |
|---|---|---|---|---|---|---|---|
| | (1) | (2) | (3) | (4) | (4′) | (5) | (6) |
| ET-Map | ü | ü | ü | | ü | | |
| Rectangular space partitioning | ü | ü | ü | | ü | | ü |
| SVD | ü | ü | | | ü | ü | |
| MWVD | ü | | ü | | ü | ü | |
| CVD | ü | ü | | | ü | | |
| PVD | | ü | ü | | ü | ü | |
| InfoSky spatialization | ü | ü | ü | ü | | ü | |

We are not aware of an existing approach that would simultaneously satisfy constraints (1), (5) and (6). That is, we know of no method that would take generator locations as constant input data and create a space partitioning with region sizes equal to a predetermined magnitude. The rectangular space partitionings meet constraint (6). However, they put the generators at any location, depending on the algorithm. Therefore, we cannot apply constraint (5) to these types of tessellations. It is our strong belief that if region locations were used as an input to the algorithm, it would not simultaneously satisfy constraints (1), (5), and (6).

As an alternative, we offer a new approach to partitioning problems described by constraints (1) and (3) through (6). However, we omit the convexity constraint (2) for the reason that we are not aware of convincing arguments that a spatialization must consist of convex regions. Limiting regions to convex polygons appears to be over restrictive. For instance, an MWVD can have convex region(s) in a special case described by [16] as "an MW-Voronoi region $r_i$ is convex if and only if the weights of adjacent MW-Voronoi regions are not smaller than $w_i$." As a result, all $N$ regions can be convex if and only if their weights are equal. That is, $w_i = w_j$ for all $i$ and $j$, which is the case of a Standard Voronoi Diagram. Therefore, we reject constraint (2). Like Andrews *et al*., we propose a Voronoi diagram solution. However, unlike Andrews *et al*. we propose an adaptive version of a Multiplicatively Weighted Voronoi Diagram, because it meets constraints (1) through (6), ignoring (2).

# Chapter 3: Adaptive Multiplicatively Weighted Voronoi Diagram

## 3.1 Basic Idea

To solve a Voronoi diagram with predetermined area relationships, we propose an iterative method that creates an Adaptive Multiplicatively Weighted Voronoi Diagram, or AMWVD. In this method, a traditional MWVD is solved repeatedly. The weights are updated in each iteration based on the weights and the error of the preceding iteration. In other words, weights are iteratively adapted based on how well the proportionality requirement is met. This is presented by the initial weight assignment

$$w_{0,j} = A_j, \tag{3.1}$$

and weight update

$$w_{i+1,j} = w_{i,j} + \Delta w_j,\ i \geq 0,\ 1 \leq j \leq N \tag{3.2}$$

where $w_{i,j}$ is the weight of generator $g_j$ at iteration $i$, $w_{i+1,j}$ is the weight of generator $g_j$ at iteration $i$+1, $w_{0,j}$ is the initial weight of generator $g_j$ and $N$ is the number of generators.

We define $a_{i,j}$ as the area allocated to generator $g_j$ after iteration $i$. We define $A_j$ as the target area of generator $g_j$. Both $A_j$ and $a_{i,j}$ are normalized such that

$$0 \leq A_j \leq 1, \text{ and } \sum_{j=1}^{N} A_j = 1 \tag{3.3}$$

and

$$0 \leq a_{i,j} \leq 1 \text{ , and } \sum_{j=1}^{N} a_{i,j} = 1 \tag{3.4}$$

for all $i$.

The constraints $0 \leq A_j \leq 1$ and $0 \leq a_{i,j} \leq 1$ are true if and only if we bound a Voronoi diagram's extent. As mentioned earlier, a general Voronoi diagram does not define any kind of boundaries, and it can grow infinitely in all directions. In the case of unbounded MWVD, a generator with the highest weight produces a region that eventually surrounds all other regions. The area of this surrounding region will be infinite. Using the 'spatial extent' (*i.e.*, bounded) method ensures a finite size of a Voronoi diagram and its regions. In fact, a borderless AMWVD is not solvable because a region with an infinite area makes area calculations and proportions impossible.

$\Delta w_j$ is positive if the allocated area $a_{i,j}$ for generator $g_j$ is less than the goal area $A_j$; $\Delta w_j$ is negative if too much area is allocated. Since the notion of an AMWVD is to continuously adjust weights to minimize the difference $A_j - a_{i,j}$, we suggest making $\Delta w_j$ proportional to three quantities:

1. $w_{i,j}$,

2. $A_j - a_{i,j}$, and

3. a positive, nonzero adjustment parameter $k$.

The resulting weight adjustment is given by

$$\Delta w_j = w_{i,j}\, k(A_j - a_{i,j}). \tag{3.5}$$

Substituting (3.5) into (3.2), we have:

$$w_{i+1,j} = w_{i,j} + w_{i,j}\, k(A_j - a_{i,j}) \tag{3.6}$$

$$= w_{i,j}\, (1 + k(A_j - a_{i,j})). \tag{3.7}$$

Eq. (3.7) is applied iteratively for every generator $g_j$ until one of two stopping criteria is satisfied: 1) the process has reached a maximum number of iterations, $I$, or 2) a 'goodness-of-fit' measure $M$ (described later) achieves some user-defined threshold.

To summarize, a user defines normalized goal areas $A_j$; Eq. (3.1) is always used to set initial weights. The weight $w_{i,j}$ is solved by Eq. (3.7). To prevent the process from becoming unstable, weight(s) cannot be less than or equal to zero. In the case when a single weight becomes negative, say $w_{i,4}$, then the region $r_{i,4}$ would occupy the entire image since in Eq. (2.3) the term $\|x-x_i\|/w_{i,4}$ is negative. Therefore, $\|x-x_i\|/w_{i,4}$ becomes the smallest distance available at any point on the Voronoi diagram. In the case when the weight $w_4$ is zero, $\|x - x_i\|/w_{i,4} = \infty$. Infinite distance leads to a conclusion that this particular region will never be assigned any space on the diagram, which defeats the

purpose of the AMWVD approach. We can also see from Eq. (3.7) that once $w_4 = 0$, it will never change its value.

## 3.2 'Goodness-of-fit' Measure *M*

We need a value of accuracy or test statistic that indicates how well constraint (6) is met at any iteration. Instead of using a predefined number of iterations, one might choose to stop iterating when this 'goodness-of-fit' measure reaches a certain value. The 'goodness-of-fit' measure we use in our tests is the average weighted error value:

$$M = E_{mean} = \frac{\sum_{j=1}^{N} \frac{|a_j - A_j|}{A_j}}{N}, \tag{3.8}$$

which weights the absolute error $|a_j - A_j|$ by the inverse goal area. The weighting factor scales each region's error so that it is relative to the goal area, thus avoiding situations where small regions are underrepresented due to their small area size.

One might use different error measures, such as maximum error $E_{\max} = \max_j \left( \frac{|a_j - A_j|}{A_j} \right)$. In this case, at each iteration we can say that a region $r_j$ has an error of at most $E_{max}$. Another possible error measure is the average squared weighted error

$$E_{squared} = \frac{\sum_{j=1}^{N} \left( \frac{a_j - A_j}{A_j} \right)^2}{N}. \quad (3.9)$$

This method may have certain advantages. $E_{squared}$, as shown in Eq. (3.9), may be especially useful in applications where regions with largest area difference $a_j - A_j$ should have a higher impact on the final solution. These errors should be reduced first, so their contribution to an average value is relatively higher than regions with a smaller difference $a_j - A_j$. Using a minimum error $E_{min}$ as a 'goodness-of-fit' should be avoided, because it does not represent or put a high bound (bounding from top) on all other regions with higher errors.

## 3.3 Adjustment Factor *k*

When measuring error, two extremes that can occur in a Multiplicatively Weighted Voronoi Diagram:

1. A region has a high weight and should occupy most of the image, but it actually occupies very little ($A_j \approx 1$, $a_{i,j} \approx 0$), and

2. A region has a low weight and should occupy very little space, but it in fact covers much of the image ($A_j \approx 0$, $a_{i,j} \approx 1$).

Thus, the difference $A_j - a_{i,j}$ lies somewhere between $-1$ and 1:

$$-1 < A_j - a_{i,j} < 1. \quad (3.10)$$

Multiplying by $k$ and adding 1 gives us

$$1 - k < 1 + k(A_j - a_{i,j}) < 1 + k. \tag{3.11}$$

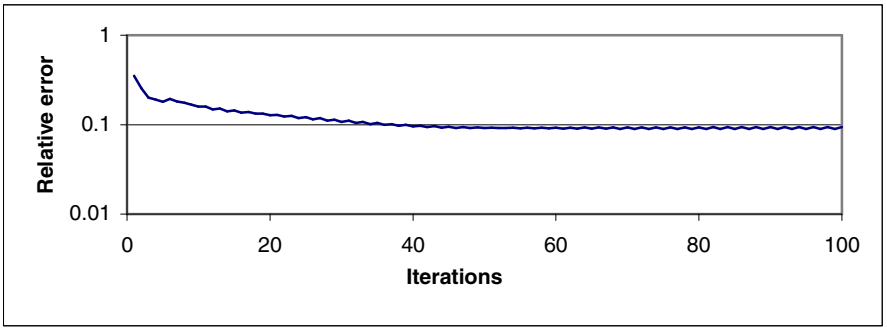
Accordingly, choosing $k=1$ is a safe default, as it will not result in negative weights. If we assign $k$=1 then

$$w_{i+1,j} = w_{i,j}\,(1 + A_j - a_{i,j}). \tag{3.12}$$

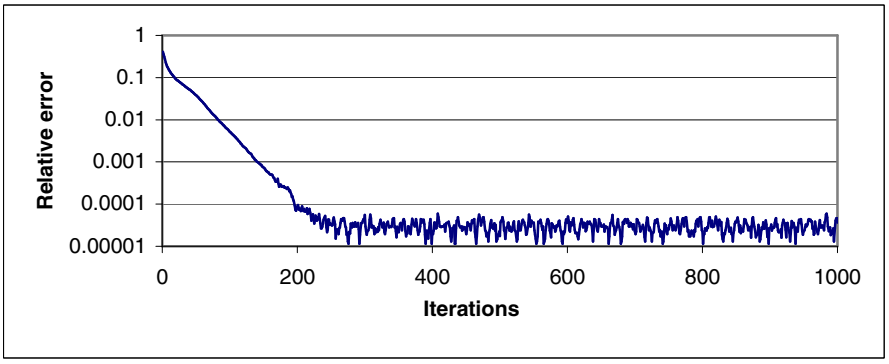
However, since $k$ can be arbitrarily chosen, it may be assigned a value larger than 1. In this case, the difference $A_j - a_{i,j}$ may be minimized in fewer iterations and the algorithm converges more quickly. As noted in Section 3.1, an MWVD cannot have negative weights. Therefore, one has to be careful when assigning large values to $k$. A consequence of Eq. (3.7) is that, in the event $a_{i,j}$ much larger than $A_j$ and $k > 1$, the expression $1 + k(A_j - a_{i,j})$ may become negative, in which case $w_j$ becomes negative and our algorithm fails to minimize the difference $A_j - a_{i,j}$. As shown in Figure 10, we have also found that using a smaller value for $k$ can prevent the algorithm from oscillating its error function. The oscillations happen when the weight adjustments are too large with the current value of $k$ and require smaller adjustments of weights, or when the discrete nature of the pixel grid prevents fine-scale adjustments (described later).

Figure 10 shows three AMWVD tests that were identical to each other except for the value of $k$. Figure 10 displays the y-axes in logarithmic scale to magnify small error values. Figure 10(a) shows a test result where $k$ = 2 with oscillation appearing after

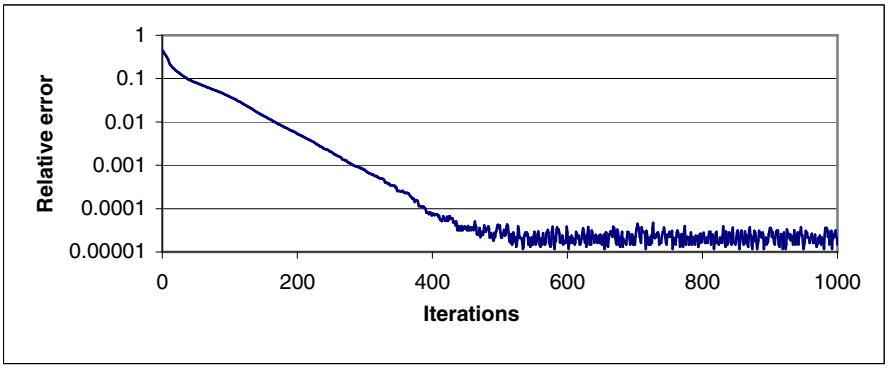
approximately 50 iterations with $E_{mean} \approx 0.1$. Figure 10(b) shows no oscillations at $k$ = 1. Figure 10(c) shows the results of a test with $k = 0.5$; it takes twice as many iterations for 10(c) to achieve the results of Figure 10(b). We observe that after 500 iterations $E_{mean}$ stays slightly below 0.0001. Even though all figures 10(a), 10(b) and 10(c) visually appear to have oscillations, their cause is different. Figure 10(a) oscillations is caused by rapid weight adjustments, while Figure 10(b) and 10(c) oscillations are caused by a pixel size. In fact, it is nearly impossible to achieve $E_{mean}$ = 0 due to our rasterized approach to solve the AMWVD. The reason is that as space is digitized in a raster world. Even very small weight changes may result in a slight over or under-allocation of space to the corresponding Voronoi region.




(a)




(b)




(c)

*Figure 10: Error trajectories $E_{mean}$ for AMWVD with three different values of k: (a) k=2.0, (b) k =1.0, and (c) k=0.5.*

## 3.4 Raster Approach

Several approaches to compute Voronoi diagrams exist: topological overlay, growth simulation, vertex calculation, and a digitized raster approach (as is the method we present) to name a few. Lan Mu [15] describes in detail the first three methods for finding vector solutions for an MWVD. However, an AMWVD places calculating region areas at a higher priority than finding a diagram's topology. As such, a raster-based approach more readily lends itself to easy and efficient computation of region areas than a vector-based approach. Indeed, the greatest advantage of a raster-based approach is its simplicity of area computations, where a vector or topological solution for a diagram's borders or a region's borders may not be known or available. If desired, one of the techniques described in [15] may be used to create a vector-based AMWVD. The raster approach is based on creating a rectangular grid of pixels, which we call an 'image.' A pixel is an atomic element of a grid. It is important to consider the resolution of a grid (or pixel size), because the AMWVD solutions are directly affected by it. This is described in detail in Section 3.5.

The pixels are indexed by an ordered pair $(x,y)$ indicating the column and row of the pixel, respectively. Using zero-based indexing, if an image has $W$ columns and $H$ rows of pixels, the top left element is pixel (0,0) and the bottom-right is pixel $(W-1, H-1)$ as shown in Figure 11.

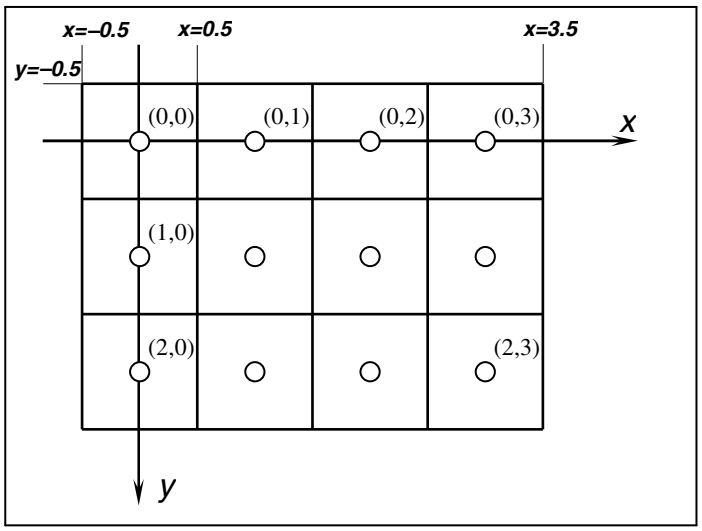



*Figure 11: Sample pixel grid of size 3×4.*

According to Shirley [21], in many applications, the rows are indexed from bottom-to-top. For our experiments, the top-to-bottom approach was found to be simpler for image output and pixel indexing. In our experiment, area is measured in pixels. Thus, a region that occupies *P* pixels has an area of *P*. As shown in Figure 11, pixel centers are located at the intersection of *x* and *y* integer coordinates, and the generators are always placed at these centers. We use two methods for placing generators: manually entered pixel coordinates and randomly generated positions. Since we are using a pixel grid rather than a continuous space approach, round-off errors are common when converting from the continuous domain to rasterized image. Figure 12 shows two different SVDs consisting of two generators and one bisector between them. Figure 12(a) shows an SVD in a continuous space, while Figure 12(b) shows the same diagram in a raster (discrete) form. One can observe that the border between two regions in 12(b) becomes a jagged line instead of a straight one due to the finite pixel

size. Figure 12(c) shows a special case of SVDs when converting to raster form has no changes on the border between regions, as shown in Figure 12(d).

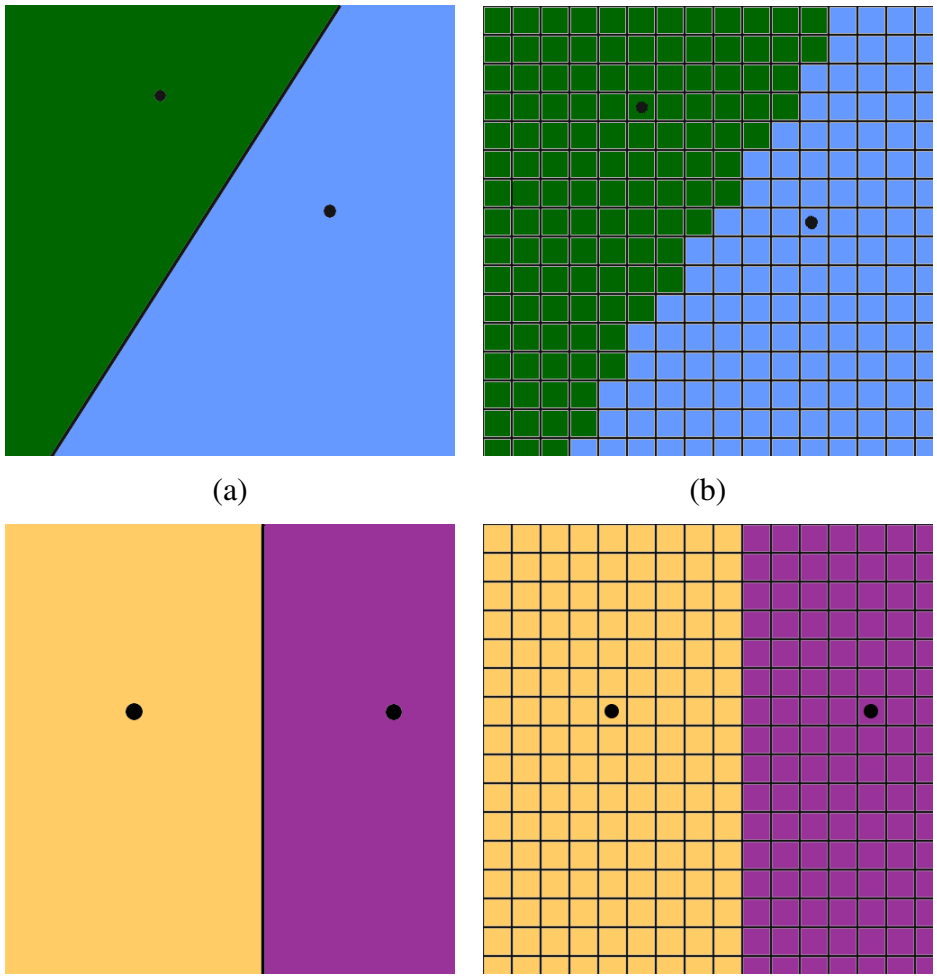
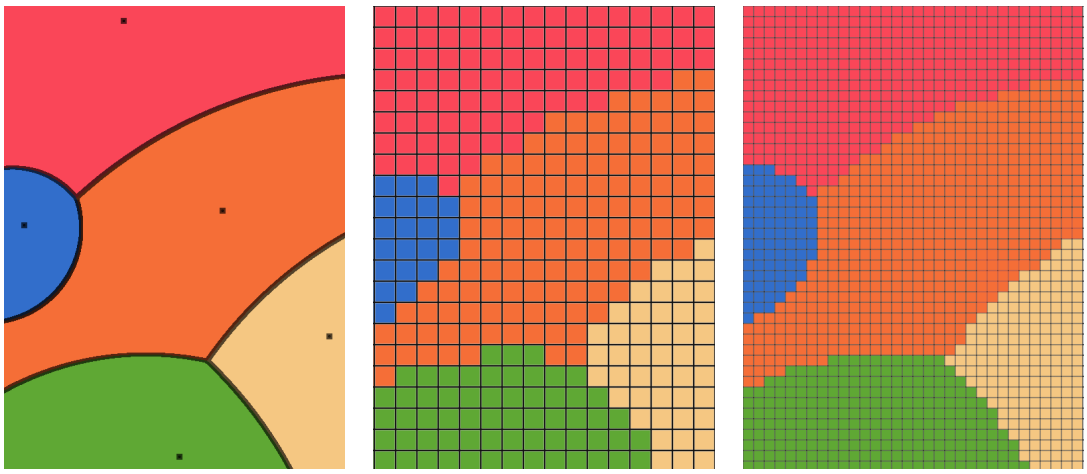



*Figure 12: Converting SVDs from continuous to raster space. (a) SVD in continuous space; (b) rasterized VD with jagged border; (c) another SVD in a continuous space; (d) rasterized VD with a straight-line border.*

For our purposes, we assume that regions occupy at least one pixel. This assumption follows from the simple fact that when considering the pixel where the generator is located, the distance between the pixel center and the region generator is zero. Since zero is the smallest distance permitted, this pixel is guaranteed to belong to the region.

## 3.5 Choosing Image Resolution

Choosing the proper image resolution is crucial when using a raster-based approach to compute VDs, as it directly affects an algorithm's performance and the resulting AMWVD image output. Consider the AMWVD shown in Figure 13, where two diagrams are created with different resolutions and are compared to an ideal continuous solution.



(a) (b) (c)

*Figure 13: MWVD with various resolutions. (a) continuous grid; (b) low resolution; (c) high resolution.*

Due to the discrete quantization inherent in the raster-based approach, the algorithm will eventually reach a point at which relative errors cannot be further reduced by additional iterations. For example, assume that a VD region is created with a goal region size of 120.17 units of area and an actual area of 120 pixels. In this case, the relative percentage error $\delta$ is:

$$\delta = \frac{|120 - 120.17|}{120.17} \times 100\% = 0.14\% .$$

By limiting the smallest unit of area to be a pixel, this is the smallest error achievable on this scale. However, round-off errors can be greatly reduced by increasing image resolution. Take the previous example and increase the image resolution by doubling the resolution along the x- and y-axis. Now, four pixels occupy what was considered one unit of area, and each new pixel occupies 0.25 units of area. Now we can get closer to the value 120.17 by assigning to a region an area of 120.25. The relative percentage error $\delta$ reduces and becomes:

$$\delta = \frac{120.25 - 120.17}{120.17} \times 100\% = 0.07\% .$$

However, choosing a higher resolution has two drawbacks that must be considered before running a solution: reduced performance and increased memory consumption. For example, the above doubling of resolution would require four times as much memory to store an image and four times as much computation time to solve when using a brute force approach. Of course, more sophisticated techniques can be implemented to reduce memory consumption, but often at the cost of using more CPU cycles.

## 3.6 Basic Algorithm Pseudo Code

To simplify the algorithm's pseudo code, we break out portions of pseudo code into separate algorithms, and refer to them as independent procedures with input/output

parameters. An MWVD is easily computed in a single pass that visits every pixel only once, as given in Algorithm 1.

*Algorithm 1: Procedure MWVD(G, N, W, H, P).*

**Input:**

| | |
|---|---|
| $G = \{x_1, \ldots, x_N, y_1, \ldots, y_N, w_1, \ldots, w_N\}$ | {Set of generators.} |
| $N$ | {Number of generators.} |
| $W$ | {Horizontal image resolution.} |
| $H$ | {Vertical image resolution.} |
| $P$ | {Empty image of size $W \times H$.} |

**Data Structures and variables:**

| | |
|---|---|
| $dist(x_1, y_1, x_2, y_2)$ | {Function: returns Euclidean distance between $(x_1,y_1)$ and $(x_2,y_2)$.} |
| $d_{min}$ | {Lowest weighted distance.} |
| $r_{min}$ | {Region number (initialized to 0).} |
| $a = (a_1, \ldots, a_N)$ | {Set of region areas (initialized to 0).} |

**Output:**

| | |
|---|---|
| $P$ | {MWVD on image of size $W \times H$.} |
| $a = (a_1, \ldots, a_N)$ | {Set of calculated region areas.} |

**Algorithm:**

```
1   for each i ∈ [0, ..., H−1]
2       for each j ∈ [0, ..., W−1]
3           d ← dist(j, i, x1, y1) / w1          {Find distance between }
4           r ← 1                                { (x,y) and g1.}
5           for each n ∈ [2, ..., N]
6               d ← dist(j, i, xi, yi)/wi        {Calculate current distance.}
7               if d < dmin then                 {If new distance is less:}
8                   dmin ← d                     {Remember the new }
9                   rmin ← n                     { distance and region.}
10          P(x,y) ← rmin                        {Assign pixel to region.}
11          armin ← armin + 1                    {Increment region area.}
```

The MWVD algorithm computes, for each pixel (lines 1 and 2), the region with the minimum weighted distance (lines 3–9) to that region's generator. The pixel is then labeled with the minimum region label (line 10) and the region area is updated (line 11).

Pseudo code for calculating the basic AMWVD is shown in Algorithm 2. Line 4 provides the algorithm with the areas of each region. These areas are used to update the generators' weights in line 7, which are based on Eq. (3.7). Line 5 is optional and may be used for progress reporting or early termination from the algorithm if $E_{mean}$ becomes less than some user-defined threshold.

*Algorithm 2: Basic AMWVD.*

**Input:**

| | |
|---|---|
| $G = \{x_1, \ldots, x_N, y_1, \ldots, y_N, w_1, \ldots, w_N\}$ | {Set of generators' data.} |
| $A = \{A_1, \ldots, A_N\}$ | {Set of desired area relationships.} |
| $N$ | {Number of generators.} |
| $I$ | {Number of iterations.} |
| $W$ | {Horizontal image resolution.} |
| $H$ | {Vertical image resolution.} |
| $P$ | {Empty image of size $W \times H$.} |
| $k$ | {Weight adjustment factor $k$.} |

**Data Structures and variables:**

| | |
|---|---|
| Error(A, a, N) | {Function: returns averaged error $E_{mean}$ across $N$ regions.} |
| r | {Region number (initialized to 0).} |
| $a = (a_1, \ldots, a_N)$ | {Set of region areas.} |

**Output:**

| | |
|---|---|
| P | {AMWVD on image of size $W \times H$.} |

**Algorithm:**

```
1   for each u ∈ [1, ..., N]
2       w_u = A_u                                   {Initialize the weights.}
3   while i < I
4       (P, a) ← MWVD(G, N, W, H, P)                {Solve MWVD with current data.}
5       E_mean ← Error(A, a, N)                     {Optional 'goodness-of-fit' reporting.}
6       for each n ∈ [1, ..., N]
7           w_n ← w_n(1 + k(A_n − a_n) )            {Update generator's weight.}
8       i ← i + 1
```

## 3.7 Adaptive Weight Factor *k*

We calculate measure $M$ at every iteration. As shown previously in Figure 10, the solution may oscillate and thus fail to improve from the previous iteration. Usually, improvement does not occur after oscillation begins at a specific value of $k$. We

therefore suggest replacing the constant value $k$ with a more adaptive form. If at any given iteration $M$ does not decrease compared to the previous iteration, the value of $k$ is reduced by a factor of $0 < k_{mult} < 1$.

In practice, $0.95 < k_{mult} < 0.98$, so that $k$ does not approach zero too quickly. Figure 14 compares two tests that use the same input data (10 generators, 500×500 pixel grid, $k = 2$). Figure 14(a) uses a constant $k$=2 while Figure 14(b) reduces $k$ by a factor $k_{mult} = 0.95$ after every iteration that $E_{mean}$ no longer decreases. As can be seen in Figure 14(a), $E_{mean}$ fails to improve after about 100 iterations with a constant $k$, whereas with a variable $k$, $E_{mean}$ decreases as far as it is possible with the given image resolution (Figure 14(b)).

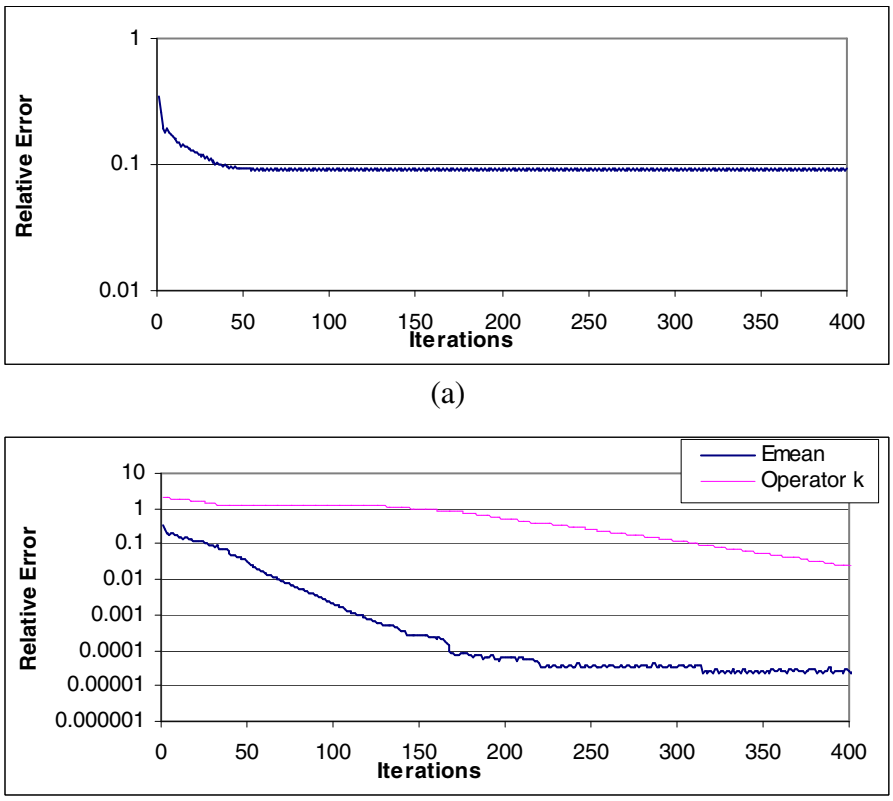


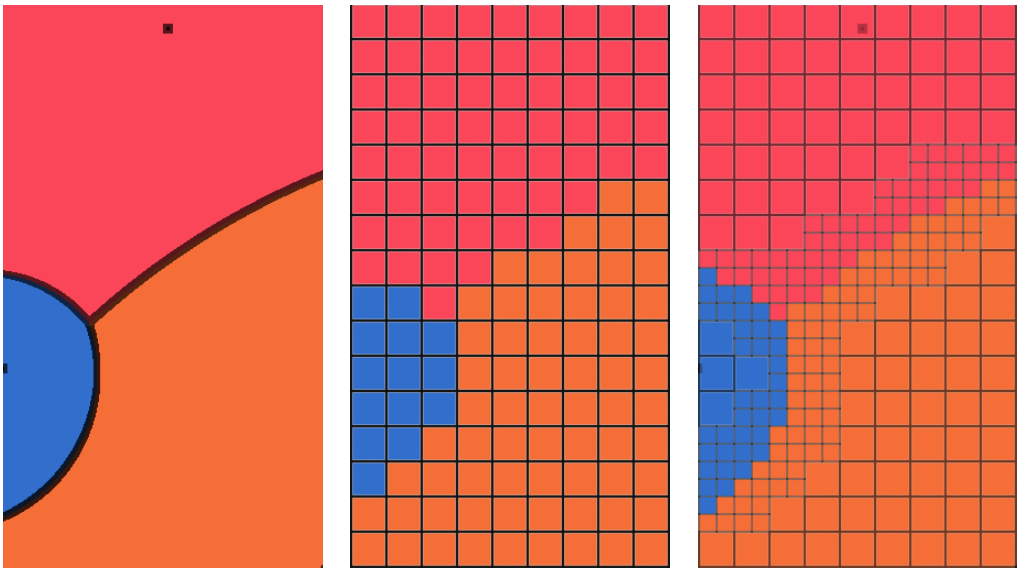

*Figure 14: Effect of k on AMWVD solution. (a) k is a constant and equal to 2; (b) k is a variable.*

## 3.8 Adaptive Pixel Sub-division

As noted earlier, convergence of $E_{mean}$ in a raster-based approach has a theoretical limit directly related to image resolution. An approach to reduce $E_{mean}$ closer to zero is to introduce a location-specific pixel sub-division algorithm that divides all pixels on the border of two regions into *S*-by-*S* square pixel grids, where *S* is the number of rows and columns of the new pixel grid. For instance, a section of an AMWVD with 2-by-2 sub-pixel refinement is shown in Figure 15. Since the sub-pixel area is

$$A_{sub-pixel} = \frac{1}{S^2} \tag{3.13}$$

pixels, sub-pixel refinement will reduce the error due to discrete rasterization by a factor of $S^2$.



(a) (b) (c)

*Figure 15: Effect of using sub-pixel refinement. (a) Continuous solution. (b) Pixel grid with original resolution. (c) Pixel grid with 2-by-2 borderline pixel refinements.*

Algorithm 3 gives the pseudo code for sub-pixel refinement. Every pixel in an image is visited exactly once. Line 3 checks if the pixel (*j*, *i*) has been refined at this iteration, and is on a border; *i.e.*, if one of its eight neighbor pixels belongs to another region. Lines 6 and 7 iterate through all sub-pixels inside of the current pixel, while line 8 assigns each sub-pixel to a closest region generator. Finally, the area of that region is incremented by the sub-pixel area in line 9.

*Algorithm 3: Sub-pixel refinement*

**Input:**

```
G = {x1, ..., xN, y1, ..., yN, w1, ..., wN}      {Set of generators' data.}
N                                                {Number of generators.}
S                                                {Number of split levels S.}
W                                                {Horizontal image resolution.}
H                                                {Vertical image resolution.}
P                                                {AMWVD image of size W×H. Each pixel stores region number.}
a = (a1, ..., aN)                                {Set of region areas.}
```

**Data Structures and variables:**

```
sub                 {Area of a sub-pixel (initialized to 1/S^2).}
border(x,y)         {Boolean function: true if (x, y) neighbors with another region.}
refined(x,y)        {Boolean function: true if (x, y) has been refined, false otherwise.}
region(x,y)         {Function: returns generator # with shortest weighted distance to (x, y).}
```

**Output:**

```
a = (a1, ..., aN)                                {Updated set of region areas.}
```

**Algorithm:**

```
1  for each i ∈ [0, ..., H-1]
2      for each j ∈ [0, ..., W-1]
3          if border(j,i) and not refined(j,i) then
4              refined(j,i) ← true
5              a_P(j,i) ← a_P(j,i) - 1                {Decrement area of the region.}
6              for each u where -S+1 ≤ u < S with step 2
7                  for each v where -S+1 ≤ v < S with step 2
8                      r ← region( j + v/(2*S), i + u/(2*S) )
9                      a_r ← a_r + sub                {Increment area of sub-pixel's region.}
```

Adaptive boundary refinement introduces a potential gain in computational efficiency since it implies that the resolution of the non-boundary pixels can lowered. In general, solving a problem of $N$ generators on an $W \times H$ grid with a boundary resolution increase factor of $S$ gives the same resolution as solving for those generators on $SW \times SH$ grid, but with far fewer computations. The precise computational gain depends on the length of all region boundaries.

## 3.9 Update-on-Improvement-Only Scheme

As can be seen in Figures 10 and 14, the error $E_{mean}$ can increase, decrease, or stay the same from one iteration to the next. Situations where $E_{mean}$ increases are unfavorable. Ideally, weights should change only when the overall 'goodness-of-fit' delineates. Thus, we introduce $E_{old}$ and $E_{new}$, where $E_{old}$ is the previous error value of $E_{mean}$ at iteration $i$ and $E_{new}$ is the error $E_{mean}$ in the current iteration $i$+1. In this way, we can check if $E_{new}$ decreases relative to $E_{old}$ and adjust the algorithm slightly to prevent the error $E_{mean}$ from ever increasing. Prior to each new iteration, we save the current error in $E_{old}$ and the set of region weights and areas in $w_{old}$, and $a_{old}$, respectively. We then compute the new weights, areas and error, $w_{new}$, $a_{new}$, and $E_{new}$ and compare the new error to the previous error. If $E_{new} < E_{old}$, (indicating that the new weights have produced a 'better' AMWVD) then we save the new values assigning them to $E_{old}$, $w_{old}$, $a_{old}$ and continue on. If, however, $E_{new} \geq E_{old}$, then the new weights have not resulted in an improvement and we restore the old weights, adjust the weight adjustment factor $k$, and try again. To accomplish this, the following steps are added to the AMWVD algorithm as shown in Algorithm 2:

- Compute the initial input weights and assign them to $w_{old}$,
- Solve the MWVD,
- Calculate the area of each region and put it into a set $a_{old}$, and
- Calculate the error $E_{mean}$ and put it into a set $E_{old}$.

The algorithm's iterative loop will contain these steps:

- Create a new set of weights $w_{new}$ according to Eq. (3.7),
- Solve the MWVD with weights $w_{new}$,
- Calculate areas of the regions and put them into the set $a_{new}$, and
- Calculate error $E_{mean}$ and put it into the set $E_{new}$,
- If $E_{new} < E_{old}$, save the current state: $E_{old} = E_{new}$, $w_{old} = w_{new}$, and $a_{old} = a_{new}$; otherwise, restore the previous state ($E_{new} = E_{old}$, $w_{new} = w_{old}$, and $a_{new} = a_{old}$), reduce parameter $k = k_{mult} * k$, and continue to next iteration.

## 3.10 Final Algorithm

Combining the original AMWVD algorithm (Algorithm 2) with sub-pixel refinement, an adaptive weight adjustment factor $k$, and the update-on-improvement method, we obtain the final algorithm as given in Algorithm 4. Since the sub-pixel method is not used in every test, it is isolated with an *if* statement (line 4). If it is known that sub-pixels will always be calculated, it may be more efficient to combine the MWVD and sub-pixel procedures together.

*Algorithm 4: Final version of the AMWVD algorithm.*

**Input:**

| | |
|---|---|
| $G = \{x_1, \ldots, x_N, y_1, \ldots, y_N, w_1, \ldots, w_N\}$ | {Set of generators' data.} |
| $A = \{A_1, \ldots, A_N\}$ | {Set of desired area relationships.} |
| $N$ | {Number of generators.} |
| $I$ | {Number of iterations.} |
| $W$ | {Horizontal image resolution.} |
| $H$ | {Vertical image resolution.} |
| $P$ | {Empty image of size $W \times H$.} |
| $S$ | {Number of split levels $S$.} |
| $k$ | {Weight adjustment factor $k$.} |
| $k_{mult}$ | {Multiplier for factor $k$.} |

**Data Structures and variables:**

| | |
|---|---|
| `Error(A, a, N)` | {Function: returns averaged error $E_{mean}$ across $N$ regions.} |
| $E_{old}$ | {Old value of $E_{mean}$ (initialized to 0).} |
| $E_{new}$ | {New value of $E_{mean}$ (initialized to 0).} |
| $r$ | {Region number (initialized to 0).} |
| $a = (a_1, \ldots, a_N)$ | {Set of region areas.} |

**Output:**

| | |
|---|---|
| $P$ | {AMWVD on image of size $W \times H$.} |

**Algorithm:**

```
1  for each u ∈ [1, ..., N]
2      w_u = A_u                                  {Initialize the weights.}
3  (P, a) ← MWVD(G, N, W, H, P)                   {Solve MWVD with initial data.}
4  if S>1 then                                    {Check if sub-pixels must be used.}
5      a ← SubPixel(G, W, H, P, a, S)             {Refine area values.}
6  end if
7  E_old ← Error(A, a, N)                         {Save the current state.}
8  w_old ← {w_1, ..., w_N}                        {Save the current state.}
9  a_old ← a                                      {Save the current state.}
10 while i < I
11     for each n ∈ [1, ..., N]
12         w_i ← w_i(1 + k(A_i − a_i) )            {Update generator's weight.}
13     (P, a) ← MWVD(G, N, W, H, P)               {Solve MWVD with current data.}
14     if S>1 then
15         a ← SubPixel(G, W, H, P, a, S)         {Refine area values.}
16     E_new ← Error(A, a, N)                     {Optional 'goodness-of-fit' reporting.}
17     if E_new < E_old then                      {Save the current state.}
18         w_old ← {w_1, ..., w_N}
19         a_old ← a
20         E_old ← E_new
21     else                                       {Revert to previous state.}
22         {w_1, ..., w_N} ← w_old
23         a ← a_old
24         k ← k * k_mult
25     i ← i + 1
```

## 3.11 Algorithm Analysis and Efficiency

The AMWVD algorithm is a modified simple fixed-point iteration optimization [4]. The classic simple fixed-point iteration finds the root of the function $f(x) = 0$ by rearranging terms such that

$$x = g(x)\,. \tag{3.14}$$

Eq. (3.14) can, in turn, be converted into an iterative formula,

$$x_{i+1} = g(x_i)\,, \tag{3.15}$$

to predict a new value of $x$ as a function of an previous value of $x$.

In the case of an AMWVD, we are looking for the root of $A_j - f(w_j)$ (*i.e.*, where $f(w_j) = A_j$), whose closed-form analytical solution or representation is not known, but can be computed numerically. Function $f(w_j)$ takes region weight $w_j$ as an input and returns region area $a_j$. Due to complexity of simultaneously considering the interaction of all the regions and their borders in the AMWVD diagram, we simplify the problem by considering only the relationship between a single weight and its associated area for a particular region and treat each region independent of the other regions in iteration $i$. Following the development of Eq. (3.15), we obtain

$$w_{i+1,j} = g(w_{i,j}). \tag{3.16}$$

Substituting Eq (3.7) for $g(w_{i,j})$, we have

$$w_{i+1,j} = g(w_{i,j}) = w_{i,j}\,(1 + k(A_j - a_{i,j})) \tag{3.17}$$

where $a_{i,j}$ is the result of our AMWVD function $f(x)$ mentioned earlier. Finally, replacing $a_{i,j}$ with $f(x)$ gives us

$$w_{i+1,j} = w_{i,j} (1 + k(A_j - f(w_{i,j}))). \quad (3.18)$$

which has the same form as Eq. (3.16) in terms of variable dependency and corresponds to the basic AMWVD algorithm in Algorithm 2.

### 3.11.1 Complexity Analysis

The AMWVD algorithm worst-case ('big-oh') complexity analysis is considered from several perspectives: running time, memory consumption, and actual measured running time in seconds. The complexity of the MWVD algorithm (Algorithm 1) is shown by the following expression:

$$O(N\ W\ H). \quad (3.19)$$

Complexity of the sub-pixel algorithm shown in Algorithm 3 is

$$O(N\ W\ H\ S^2) \quad (3.20)$$

where each pixel is divided into $S$-by-$S$ sub-pixels. However, it is possible to divide a square pixel into rectangular sub-pixels (i.e. 1x4, 2x3, 7x4, etc.). In the case of dividing a square pixel into $S_{horizontal}$ columns and $S_{vertical}$ rows of sub-pixels, the complexity of procedure sub-pixel becomes

$$O(N\ W\ H\ S_{horizontal}\ S_{vertical}). \quad (3.21)$$

The big-oh estimate for a basic AMWVD algorithm is:

$$O(I * O(\text{MWVD procedure}) ). \tag{3.22}$$

Substituting Equation (3.19) into (3.22), we obtain the following complexity for the basic AMWVD algorithm without sub-pixel refinements:

$$O(I\,N\,W\,H). \tag{3.23}$$

Running time of the final AMWVD algorithm with sub-pixel accuracy:

$$O(I * O(\text{MWVD procedure}) + O(\text{Sub-pixel procedure}) ), \tag{3.24}$$

which becomes

$$O(I\,N\,W\,H + I\,N\,W\,H\,S^2) ). \tag{3.25}$$

Eq. (3.25) becomes Eq. (3.26) after smaller terms elimination:

$$O(I\,N\,W\,H\,S^2). \tag{3.26}$$

If sub-pixel refinement is not used in Algorithm 4 (lines 14 and 15), running time for the final AMWVD algorithm remains equal to (3.23).

### 3.11.2 Memory Consumption

Let us now look at the algorithm's memory consumption. At run time, we store the following information in memory:

1. Generators: *x*-position, *y*-position, and weight *w*. This is $O(N)$.

2. Image. Every pixel in the image stores the number of the generator the pixel belongs to. The size of the image is $O(W\,H)$.

All other variables and data structures are either constant in memory size or bounded by $O(N)$. Thus, the total memory consumption is $O(N + W\,H)$. Since we assumed that every region must occupy at least one pixel, there are at most $W \times H$ regions. The final memory consumption becomes:

$$O(W\,H). \tag{3.27}$$

### 3.11.3 Measured Running Time

We performed a series of tests with the goal to measure the AMWVD algorithm's performance. Our test machine was a 1.81 GHz AMD Athlon64 3000 with 512 MB of 400 MHz DDR RAM.

Table 2 shows running times for several tests that involved varying the number of regions $N$, the number of iterations $I$, and image resolution $W \times H$. We used the same generators weights $w_1 ... w_N$ for all tests and generator $x$- and $y$-positions have been created only once using a random number generator in a range [0,1), after which, the generators' positions were scaled accordingly to match the image resolution of the test. Table 3 shows running times for the same tests as shown in Table 2, with the only difference being the use of different image resolutions and 4-by-4 sub-pixel refinements. Since an image with a resolution of 500×500 pixels is equivalent, in terms of measurement error due to rasterization, to a resolution of 125×125 pixels with

4-by-4 sub-pixel refinements, the corresponding results in each table can be compared, since relative pixel area coverage (in terms of total image percentage) has been preserved. For instance, pixel (100,48) at resolution 500×500 is equivalent to pixel (25,12) at resolution 125×125. Table 4 shows time advantage of using sub-pixel refinement method. The value of each cell is obtained by dividing the values of corresponding cells in Table 2 and Table 3. It can be concluded that a test with sub-pixel refinements, 50 regions, and 100 iterations at 1000×1000 resolution runs about 5.1 times faster than a similar test without sub-pixel refinements.

*Table 2: Running time for different test cases in seconds.*

| **Number of Regions (*N*)** | **Iterations (*I*)** | **Resolution (*W*×*H*)** | |
|---|---|---|---|
| | | 500×500 | 1000×1000 |
| 10 | 100 | 10 | 38 |
| | 500 | 46 | 185 |
| 50 | 100 | 39 | 157 |
| | 500 | 195 | 781 |

*Table 3: Running times with 4-by-4 sub-pixel refinement in seconds.*

| **Number of Regions (*N*)** | **Iterations (*I*)** | **Resolution (*W*×*H*)** | |
|---|---|---|---|
| | | 125×125 | 250×250 |
| 10 | 100 | 2 | 5 |
| | 500 | 9 | 24 |
| 50 | 100 | 13 | 31 |
| | 500 | 66 | 162 |

*Table 4: Time advantage of using sub-pixel refinement method.*

| **Number of Regions (*N*)** | **Iterations (*I*)** | **Resolution (*W*×*H*)** | |
|---|---|---|---|
| | | 500×500 | 1000×1000 |
| 10 | 100 | 5.0 | 7.6 |
| | 500 | 5.1 | 7.7 |
| 50 | 100 | 3.0 | 5.1 |
| | 500 | 3.0 | 4.8 |

The results show that although the worst case of the AMWVD algorithm is $O(I\,N\,W\,H\,S^2)$ with sub-pixel refinements and $O(I\,N\,W\,H)$ without, the actual running times of tests listed in Table 3 are 3 to 7 times shorter of those listed in Table 2. We estimated that in the case of sub-pixel refinements the worst case occurs when all pixels must be sub-divided, which is very unlikely in an average test. As a result, sub-pixel refinement allows the AMWVD to be effectively computed using lower image resolution while only having to process a small number of pixels located on the borders with other Voronoi regions. Of course, the actual gains depend entirely on the complexity of the solution.

# Chapter 4: Results

In this chapter, we present the results of several tests that show the ability of the algorithm to create an AMWVD using the final version of the algorithm described in section 3.10. The results include three pathological tests and two space partitioning tests involving empirical data.

## 4.1 Pathological Cases

Figures 16 and 17 show results for the SVD (a), the MWVD (b), and the AMWVD (c) solutions. Figure 18 shows only the AMWVD solution.

- Figure 16 represents the problem of nine uniformly placed generators, each with a randomly selected weight $1 \leq w_j \leq 9$.
- Figure 17 shows the degenerate case of ten in-line generators spaced at equal intervals with weights increasing linearly from 1 on the left to 10 on the right.
- Figure 18 shows an AMWVD for 100 randomly placed generators. The weights are randomly selected integers between 1 and 100.

To assess weight-area proportionality, we rely on the previously defined 'goodness-of-fit' measure $E_{mean}$ (3.8). For the two types of multiplicatively weighted Voronoi diagrams in Figures 16 and 17, we show the following 'goodness-of-fit' measures:

- Minimum error $E_{min}$,

- Mean error $E_{mean}$,
- Maximum error $E_{max}$, and
- Correlation $r$ between accomplished area $a_{i,j}$ and desired area $A_j$.

It is important to note that resolution (1,000×1,000), number of regions, number of iterations (1,000), and initial value of $k$ were the same in the first two tests. The only differences are generator locations and weights.

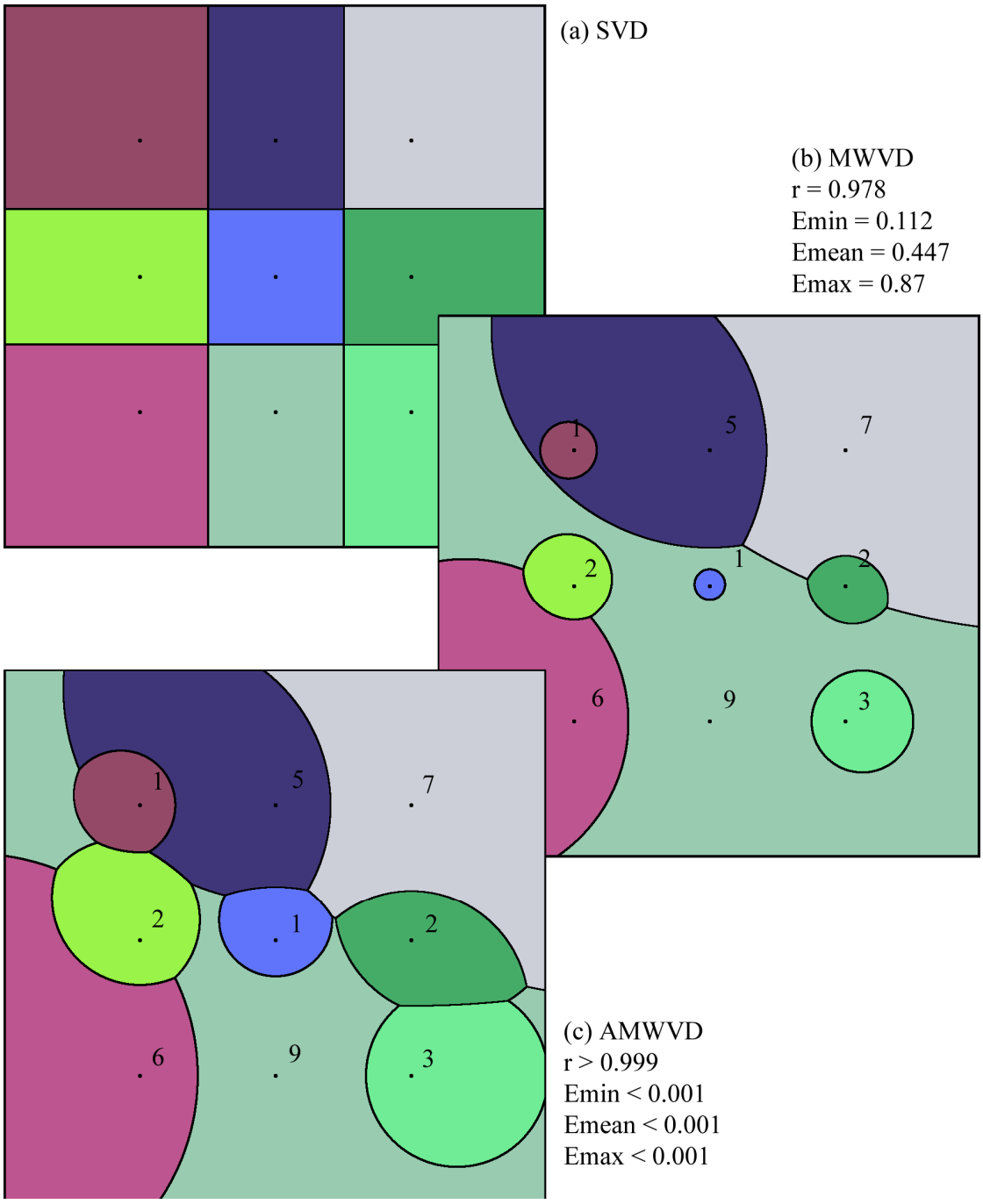



*Figure 16: AMWVD solution for nine uniformly placed generators. Integer weights randomly selected between 1 and 9. (a) SVD; (b) MWVD; (c) AMWVD.*

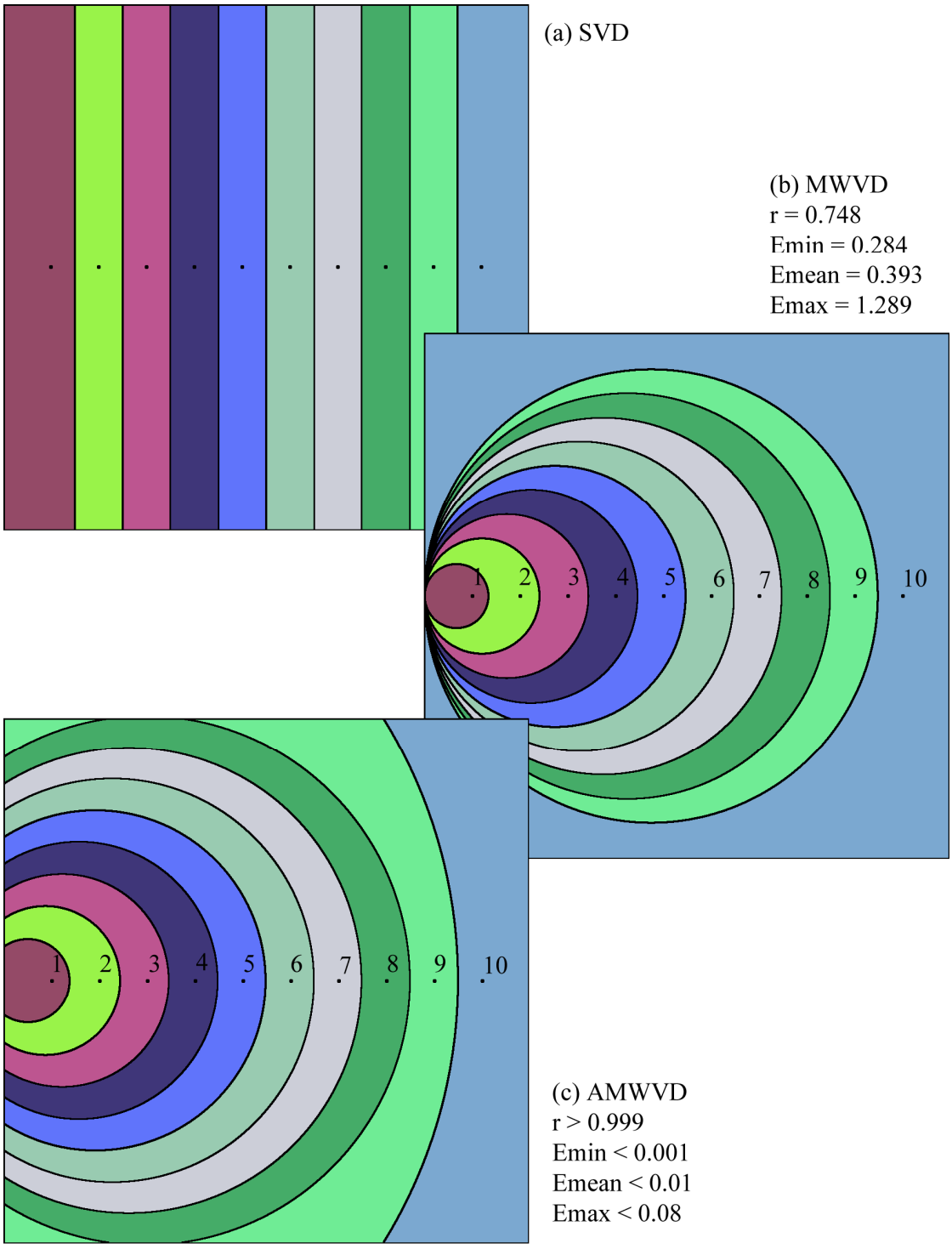



*Figure 17: AMWVD solution for ten in-line generators with weights linearly increasing from 1 to 10. (a) SVD; (b) MWVD); (c) AMWVD.*

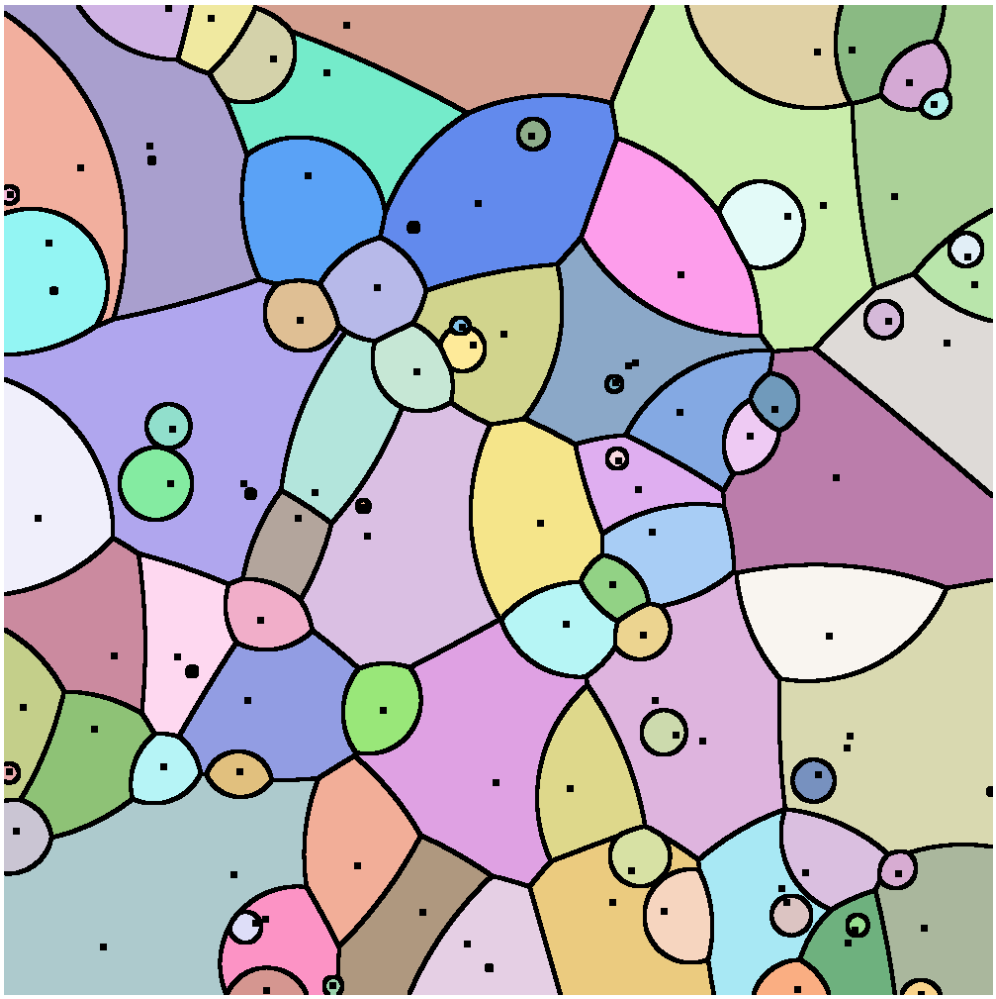


*Figure 18: AMWVD solution for 100 randomly distributed generators. Integer weights selected randomly from 1 to 100. Very small regions display as points.*

In the first two pathologies, the MWVD clearly outperforms the SVD. However, the Adaptive Multiplicatively Weighted Voronoi Diagram clearly outperforms its non-adaptive form. Weighting by distance generates better results than not weighting at all, but adaptively weighting works much better still. The observations show the MWVD assigns too little area to generators with low weights and too much area to those with high weights. The adaptive version solves this problem by iteratively adjusting the weight set in a direction that results in the required area-weight proportionality.

Figures 19, 20, and 21 show the trajectories of the various 'goodness-of-fit' measures for each of the three pathological cases using the final AMWVD algorithm shown in Algorithm 4. All values were increased by $10^{-7}$ in order to plot zero values on a logarithmic scale and $\log_{10}(0)$ is undefined. We show solutions for figure 16 and 17 using three spatial resolutions: 1000×1000, 2000×2000 and 4000×4000. The solution for Figure 18 is shown only in 4000×4000. Whereas the 1000×1000 grid was used as the base resolution for all three cases, the higher 2000×2000 and 4000×4000 resolutions were implemented using the boundary resolution refinement technique discussed above. *k* adjustments were implemented as described in Section 3.7. Although all three trajectories indicate conversion toward ideal proportionality, the convergence rate is neither guaranteed nor unproblematic. For instance, solution for the first two cases (Figure 16 and 17) with a *k* of 1, using the final algorithm shown in Algorithm 4, works differently. The uniform pattern case starts to converge immediately allowing *k* to stay high for about 200 iterations before having to be reduced, while the linear pattern requires reducing *k* first to about 0.1 before the

algorithm starts converging. One can also notice that the number of iterations required for the uniform pattern to converge is between 150 and 200 (depending on resolution), whereas, the in-line pattern takes 800 to 1200 iterations.

The test with in-line generators shown in Figure 20 has exposed another case in which the convergence rate is unpredictable. Even though the errors $E_{min}$, $E_{mean}$, and $E_{max}$ are less than 0.01, they are still approximately one magnitude worse than the corresponding error values in the uniformly distributed generators test (Figure 19). We hypothesize that in-line generators, with weights as in Figure 17, create $N$ full circles that are stacked inside of each other. As a result, the total boundary length of each region becomes very long. Most of the changes made to the region's weight affect the region more than in any other test.

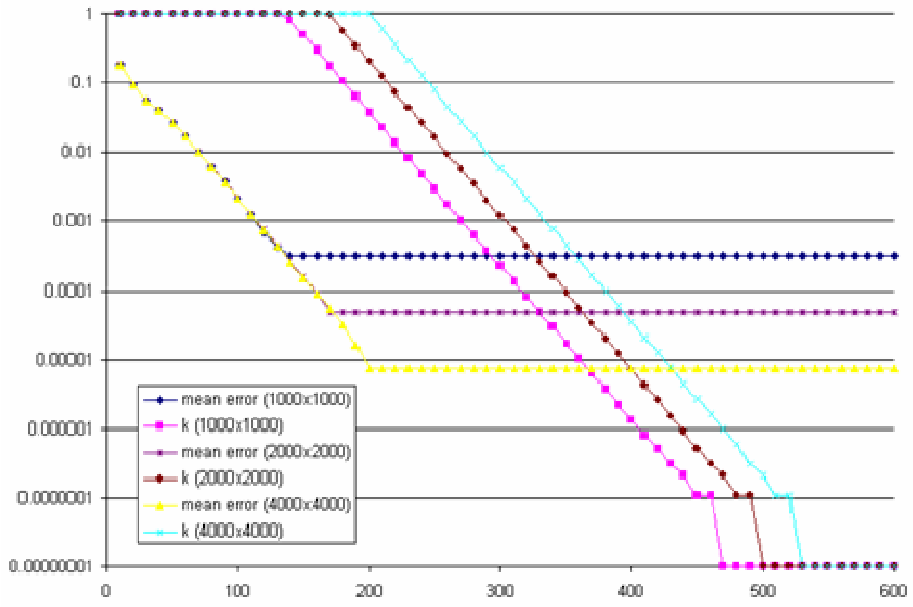



*Figure 19: Mean error and k trajectories for the uniform pattern. (Figure 16(c); every tenth iteration plotted).*

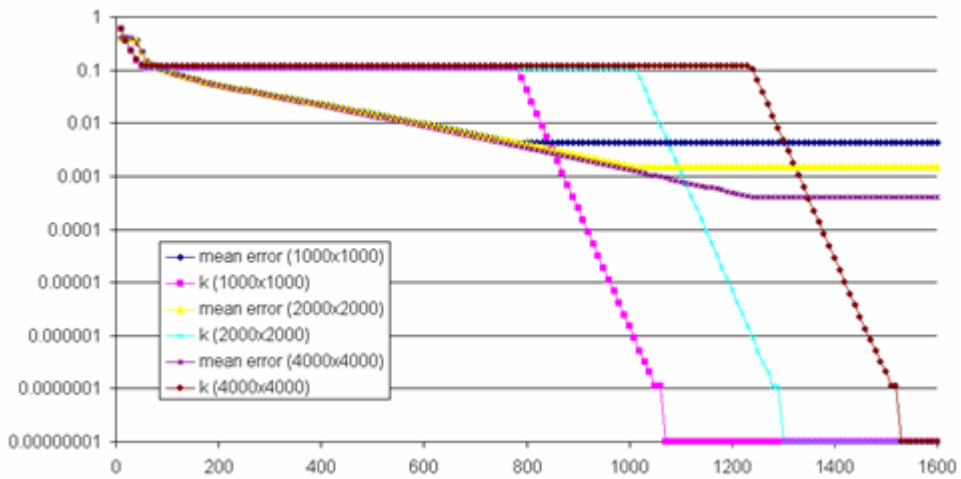



*Figure 20: Mean error and k trajectories for the linear pattern. (Figure 17(c); every tenth iteration plotted).*

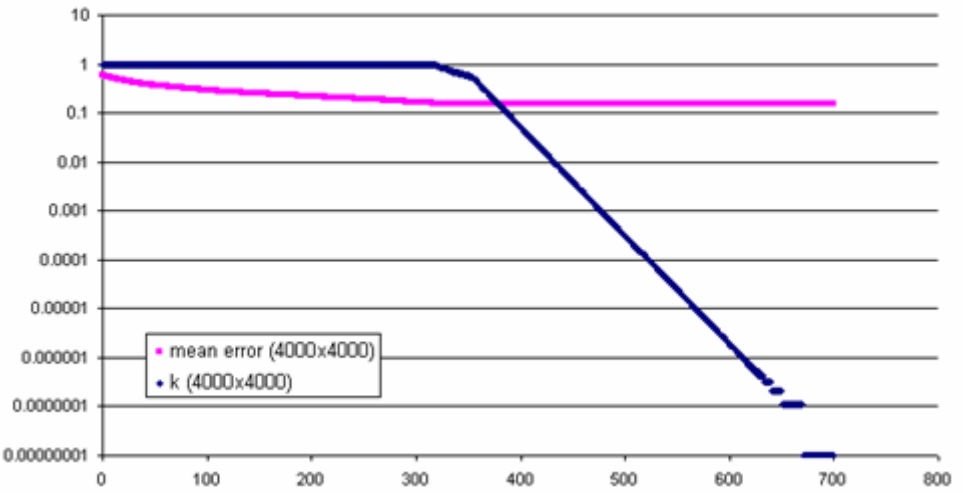



*Figure 21: Mean error and k trajectories for an AMWVD of 100 randomly distributed generators. (Figure18).*

It was described in section 3.3 that it is practically impossible to reach complete conversion when using a raster-based spatial model. This can be seen at the tail end of Figures 19, 20 and 21. At this point in the convergence process, even a single pixel

allocation change will change the error. As a result, even when $k$ is reduced, the error no longer reduces.

Figure 21 shows the worst convergence results with $E_{mean} \approx 0.2$ for 100 randomly placed generators (Figure 18). This test exposes the influence of two major factors: a) image resolution relative to the smallest region, and b) relative weights between two regions. In particular, one can notice that Figure 18 shows quite a few regions as dots, because corresponding generator weights are small compared to other regions. As a result, these regions are assigned only a small number of pixels. Again, each pixel here makes a big difference in error for that particular region, and, therefore, negatively affects the average value of $E_{mean}$ in overall solution across all regions.

Finally, Figures 16 also reveals a fundamental weakness of multiplicatively weighted Voronoi diagrams which is reminiscent of the dislocation problem associated with Andrews *et al.*'s [1] use of power diagrams, namely that a region can become discontinuous. For instance, the region with a weight of 9 in Figure 16 is split into three separate areas by other generators, each with their own region. Strictly speaking, this does not violate constraint (1) (the generator must be located in its region) yet it might be considered an undesirable effect in certain applications. However, according to a usability test conducted by Reitsma and Trubin [18], no evidence was found that "the estimation of discontinuous areas was more difficult, or more erroneous than that of continuous regions."

## 4.2 Comparing ET-Map with AMWVD

In this test, we compare the results of ET-Map shown in Figure 22 and an AMWVD that uses ET-Map data as input. The reason for this test is to compare the AMWVD algorithm with an existing space partitioning method that also tries to establish area-magnitude relationship using empirical data (as compared to the pathological tests shown in Section 4.1). The color of the regions on the ET-Map serves no other goal than to visually separate them. Chen *et al.*'s ET-Map space partitioning has a resolution of 20×10. As a result, the pixels are not square on the map and have 1:2 width-to-height ratios. Table 4 shows the ET-Map's data as reengineered from Chen *et al.*'s publication [6]. Generator centers have been defined as the closest pixel to a '+' character on the map. The number of links of each ET-Map's region has been used as target area and initial weight for an AMWVD. It appears that an ET-Map can occasionally produce discontinuous regions. For instance, *FAQ*, *Songs*, *Story*, and *Artist* categories are split into two regions each. In these cases, each region divided into two sub-regions has been treated as one region; the area of a region has been defined as a combined sum of the two sub-regions, and the generator's location is set to the larger of the two regions.

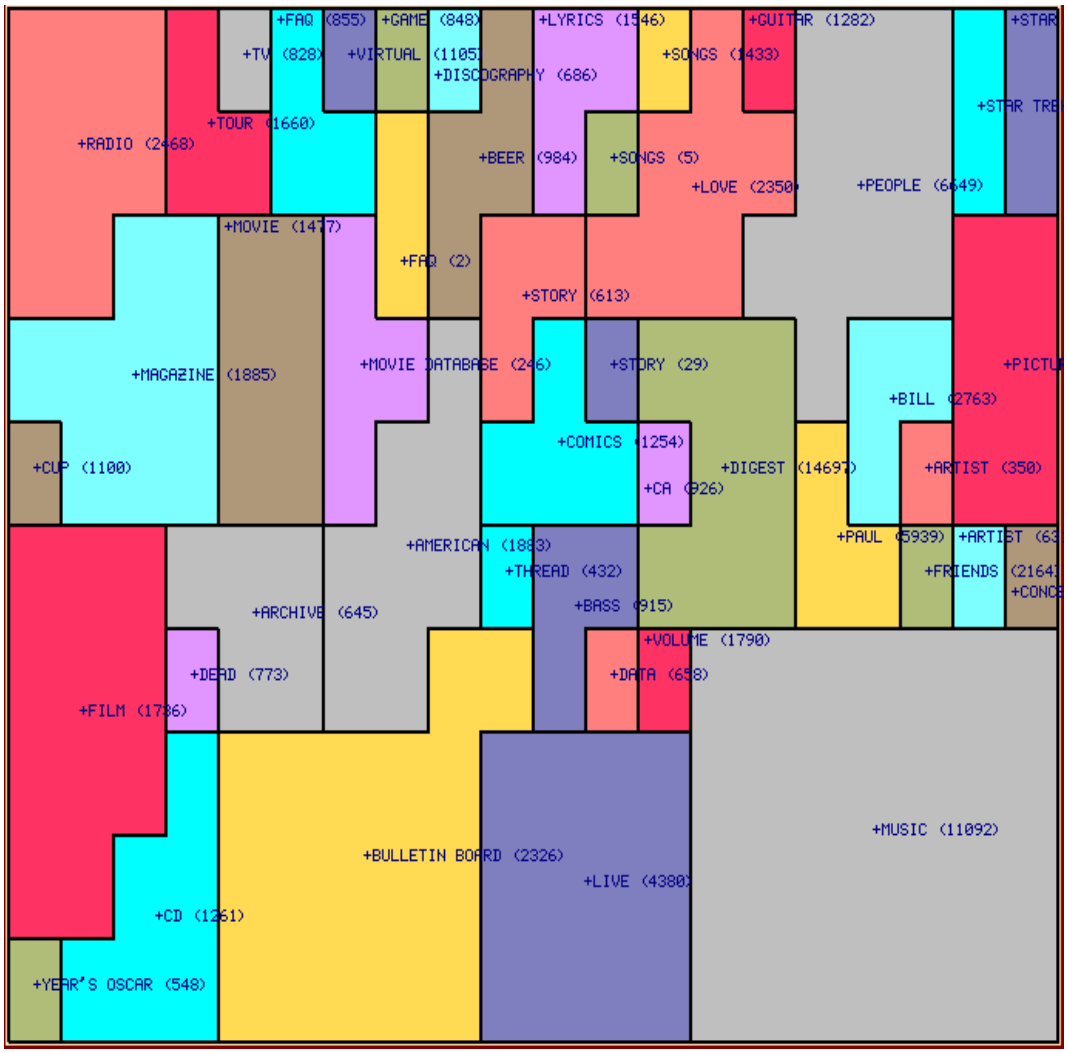



*Figure 22: ET-Map.*

The X and Y columns in Table 4 show the coordinates of the regions on a 20×10 resolution, while the X′ and Y′ columns show coordinates of the regions scaled to fit an alternative 1200×1200 resolution.

*Table 5: ET-Map data.*

| **Name** | **Number of links** | **Area** | **X** | **Y** | **X′** | **Y′** |
|---|---|---|---|---|---|---|
| American | 1883 | 8 | 7 | 5 | 460 | 619 |
| Archive | 645 | 5 | 4 | 5 | 285 | 698 |
| Artist | 981 | 2 | 17 | 4 | 1047 | 532 |
| Bass | 915 | 3 | 10 | 5 | 650 | 690 |
| Beer | 984 | 4 | 9 | 1 | 542 | 175 |
| Bill | 2763 | 3 | 16 | 3 | 1006 | 453 |
| Bulletin Board | 2326 | 17 | 6 | 8 | 410 | 976 |
| CA | 926 | 1 | 12 | 4 | 729 | 555 |
| CD | 1261 | 6 | 2 | 8 | 175 | 1046 |
| Comics | 1254 | 4 | 10 | 4 | 631 | 501 |
| Concert | 1884 | 1 | 19 | 5 | 1144 | 674 |
| Cup | 1100 | 1 | 0 | 4 | 37 | 532 |
| Data | 650 | 1 | 11 | 6 | 690 | 769 |
| Dead | 773 | 1 | 3 | 6 | 215 | 769 |
| Digest | 14697 | 8 | 13 | 4 | 816 | 531 |
| Discography | 686 | 1 | 8 | 0 | 491 | 80 |
| FAQ | 857 | 5 | 5 | 0 | 313 | 17 |
| Film | 1736 | 11 | 1 | 6 | 88 | 810 |
| Friends | 2164 | 1 | 17 | 5 | 1047 | 650 |
| Game | 848 | 1 | 7 | 0 | 432 | 16 |
| Guitar | 1282 | 1 | 14 | 0 | 847 | 16 |
| Live | 4380 | 12 | 11 | 8 | 661 | 1006 |
| Love | 2350 | 7 | 13 | 1 | 783 | 209 |
| Lyrics | 1546 | 3 | 10 | 0 | 610 | 17 |
| Magazine | 1885 | 9 | 2 | 4 | 149 | 424 |
| Movie | 1477 | 6 | 4 | 2 | 254 | 254 |
| Movie Database | 246 | 4 | 6 | 3 | 408 | 413 |
| Music | 11092 | 28 | 16 | 7 | 988 | 947 |
| Paul | 5939 | 3 | 15 | 5 | 947 | 611 |
| People | 6649 | 11 | 16 | 1 | 970 | 207 |
| Picture | 2181 | 6 | 19 | 3 | 1135 | 413 |
| Radio | 2468 | 8 | 1 | 1 | 87 | 159 |
| Songs | 1433 | 2 | 12 | 0 | 750 | 56 |
| Star Trek | 346 | 2 | 18 | 0 | 1106 | 115 |
| Star Wars | 1282 | 2 | 19 | 0 | 1144 | 16 |
| Story | 642 | 4 | 9 | 2 | 591 | 334 |
| Thread | 432 | 1 | 9 | 5 | 571 | 651 |

*Table 5 (Continued).*

| | | | | | | |
|---|---|---|---|---|---|---|
| Tour | 1660 | 3 | 3 | 1 | 234 | 135 |
| TV | 828 | 1 | 4 | 0 | 275 | 56 |
| Volume | 1790 | 1 | 12 | 6 | 729 | 729 |
| Virtual | 1105 | 1 | 6 | 0 | 393 | 56 |
| Year's Oscar | 548 | 1 | 0 | 9 | 37 | 1125 |

Figure 23 shows the resulting AMWVD for the ET-Map data. The image has been rescaled from 20×10 pixels to 20×20 pixels. The results are better than that of an ET-Map in terms of area–number of links relationships, but the error values $E_{mean}$ and $E_{max}$ remain too high and the correlation coefficient $r$ is too low. For this experiment, the AMWVD algorithm performs poorly, because the image has a very low resolution. The weights of the regions have to increase by a significant amount in order to grab a neighboring pixel. In order to eliminate the effect presented by low image resolution we reran the AMWVD test using a 1200×1200 resolution. The resulting diagram is shown in Figure 24.

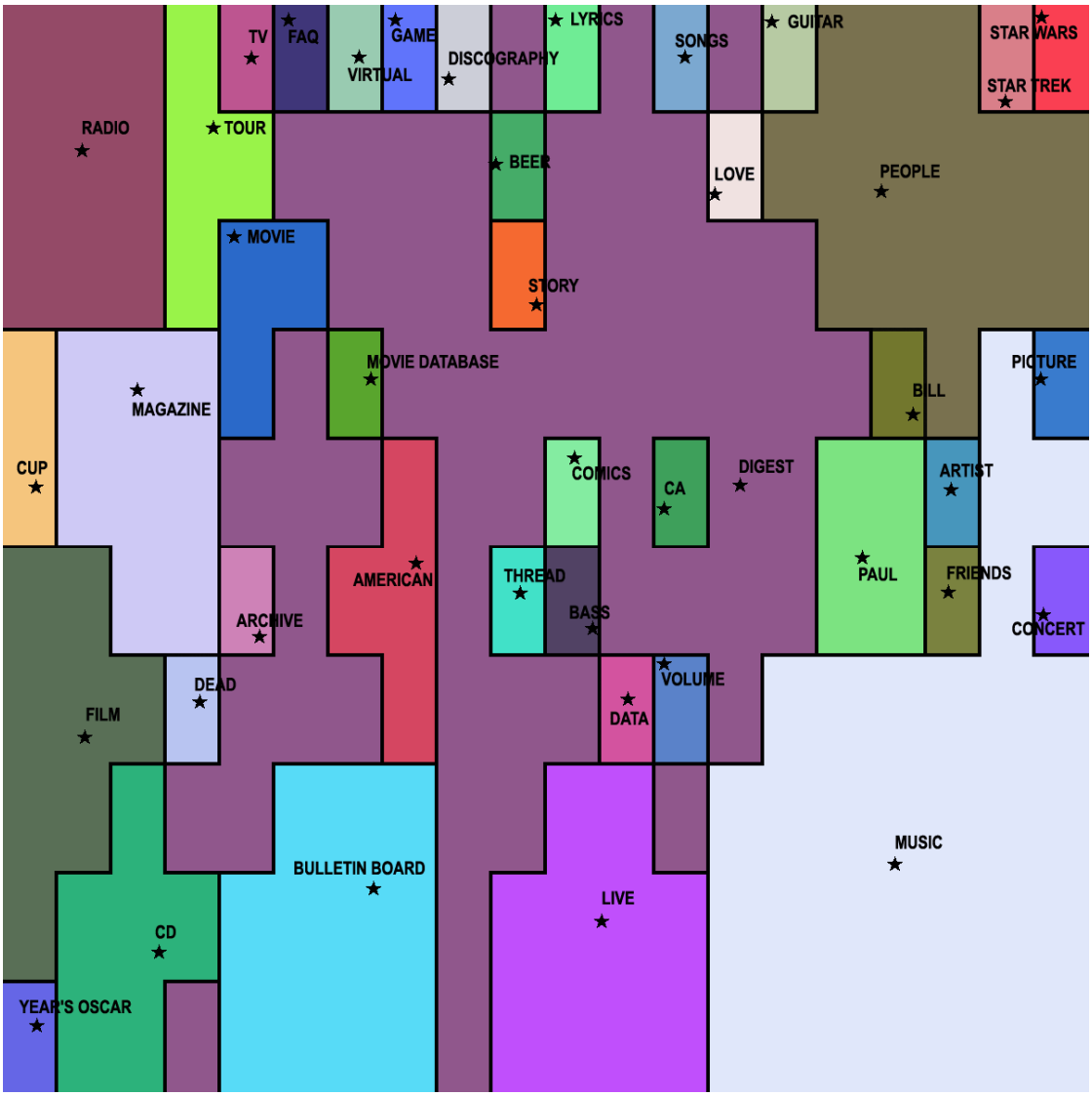



*Figure 23: AMWVD for ET-Map data using 20×10 pixel grid.*

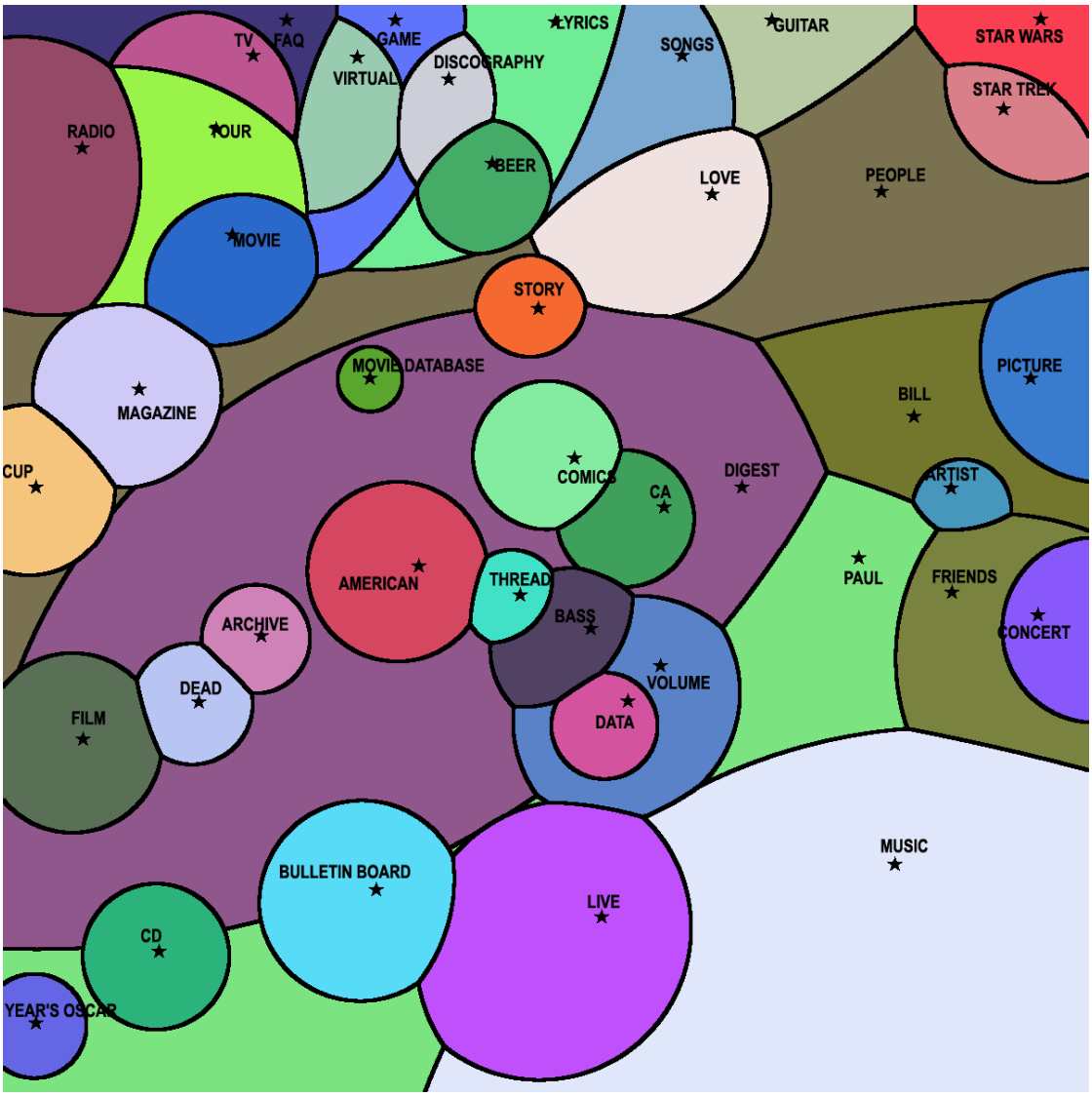



*Figure 24: AMWVD for ET-Map data using 1200×1200 pixel grid.*

Table 5 compares the results of the ET-map tessellation, the AMWVD with 20×10 resolution and the AMWVD with 1200×1200 resolution using four different error measures. It can be observed that both AMWVD diagrams create better area proportionality with the number of links in a category than the original ET-Map.

*Table 6: Comparing ET-Map results and AMWVD.*

| **Error measures** | **ET-Map 20×10 grid** | **AMWVD 20×10 grid** | **AMWVD 1200×1200 grid** |
|---|---|---|---|
| $E_{min}$ | 0.028 | 0.001 | < 0.001 |
| $E_{mean}$ | 0.825 | 0.510 | 0.002 |
| $E_{max}$ | 6.218 | 1.112 | 0.034 |
| Correlation *r* | 0.603 | 0.931 | 0.999 |

## 4.3 Partitioning of an Information Space with an AMWVD

This is yet another example where AMWVD is believed to bring the advantages of introducing area-magnitude relationships to the diagram. Out of many existing methods for finding the proper positional information of the generators, we use the approach proposed by Buttenfield and Reitsma [3]. Our input data come from the *Building as a Learning Tool* (BLT) Web-based information system implemented at the University of Colorado [2]. This is a real-time building control information system used in undergraduate engineering teaching and learning. The system monitors and stores information from over 300 sensors embedded in the building's structural and control systems. Students access the data when studying building control theory, heat transfer, material sciences, and a variety of other engineering disciplines. Figure 25 shows a sample of data delivered to a user's web browser, as measured by an array of thermistors embedded in one of the building's walls.

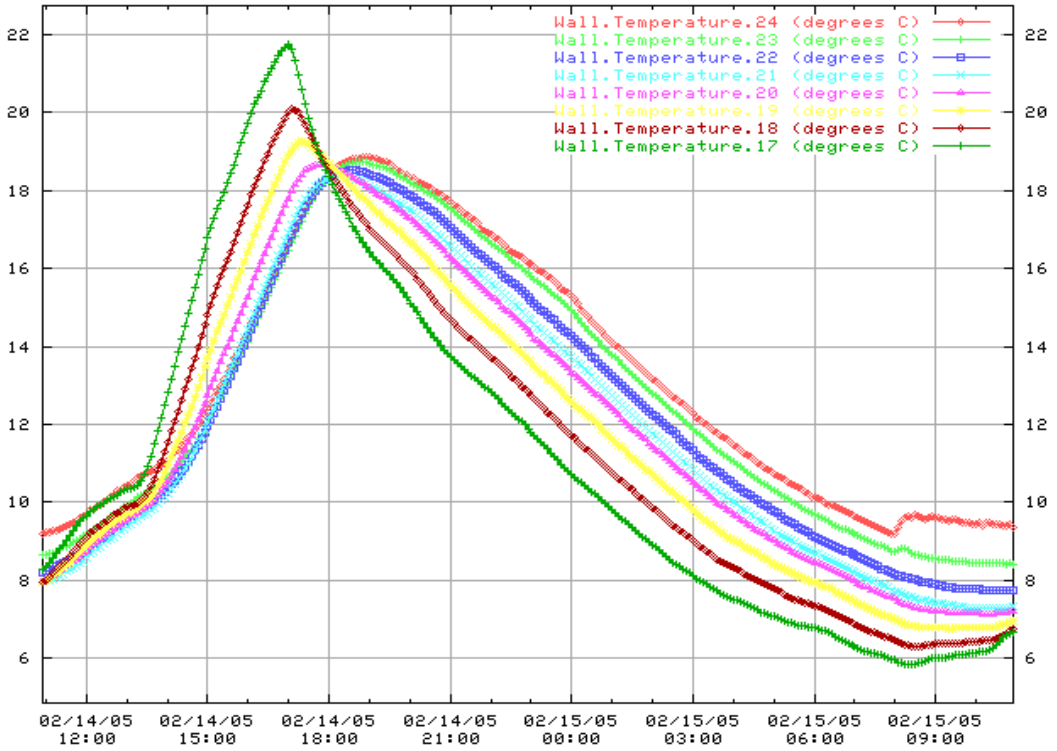



*Figure 25: BLT data display for the last 24 hours.*

The method described in [3] is based on two assumptions:

- If two pages are associated with one another frequently, as in the case of hyperlinking, they are placed in close proximity in the information space.
- Only the statistically significant usage patterns are used for information space reconstruction.

The actual method used is that of loglinear modeling, followed by multidimensional scaling.

Adopting this method, a year's worth of BLT usage data yields a four-dimensional, orthogonal information space. Into this space, ten groups of BLT Web pages are fitted as points. The space dimensions are interpreted as follows:

- *Dimension 1: Relative Task Priority*. This dimension represents the priority of web page groups. High priority tasks in the BLT Website are the formulation of precise data requests, access of technical sensor information and access of information taken from the various building systems and subsystems in which sensors are embedded.

- *Dimension 2: Active vs. Passive*. The second dimension separates active pages from passive pages. Passive pages do not allow for user interaction; they only contain explanatory information. On the other hand, active pages require explicit user input.

- *Dimension 3: Granularity of Information*. Information granularity represents the depth or level of detail offered by pages. General overview pages offer very little detail and do not include detailed user tasks. The detailed end of the dimension, however, contains groups of pages, which incorporate specific, narrative, and technically detailed information.

- *Dimension 4: Data vs. Meta Data*. The fourth dimension scales the nature of the data in terms of data collected by the sensors versus data about the BLT system. Web pages describing the BLT system are located at one end of the dimension, while groups representing measured data are clustered at the other end.

For testing the performance of the AMWVD method outlined above, only the first two dimensions of the BLT information space are used, since at this point we are interested

only in two-dimensional AMWVD solutions. As for the magnitude criterion for allocating space to the Voronoi regions, we chose the overall likelihood that users request the web pages. This likelihood is computed as part of the procedure proposed by Buttenfield and Reitsma [3]. The resultant partitioning then constitutes a sort of gravitational field map with each Web page group as the center of its basin of attraction. The higher the likelihood of visits from the Web, the larger its region.

Figure 26 shows the partitioning of the BLT information space using the SVD (a), the MWVD (b), and the AMWVD (c). As can be seen from the error measure values, the proportionality constraint is well matched in the adaptive version. Conversion occurs at about 900 iterations. However, the weight and location distributions are such that two of the ten regions, *List Sensor* and *Locations*, become discontinuous.

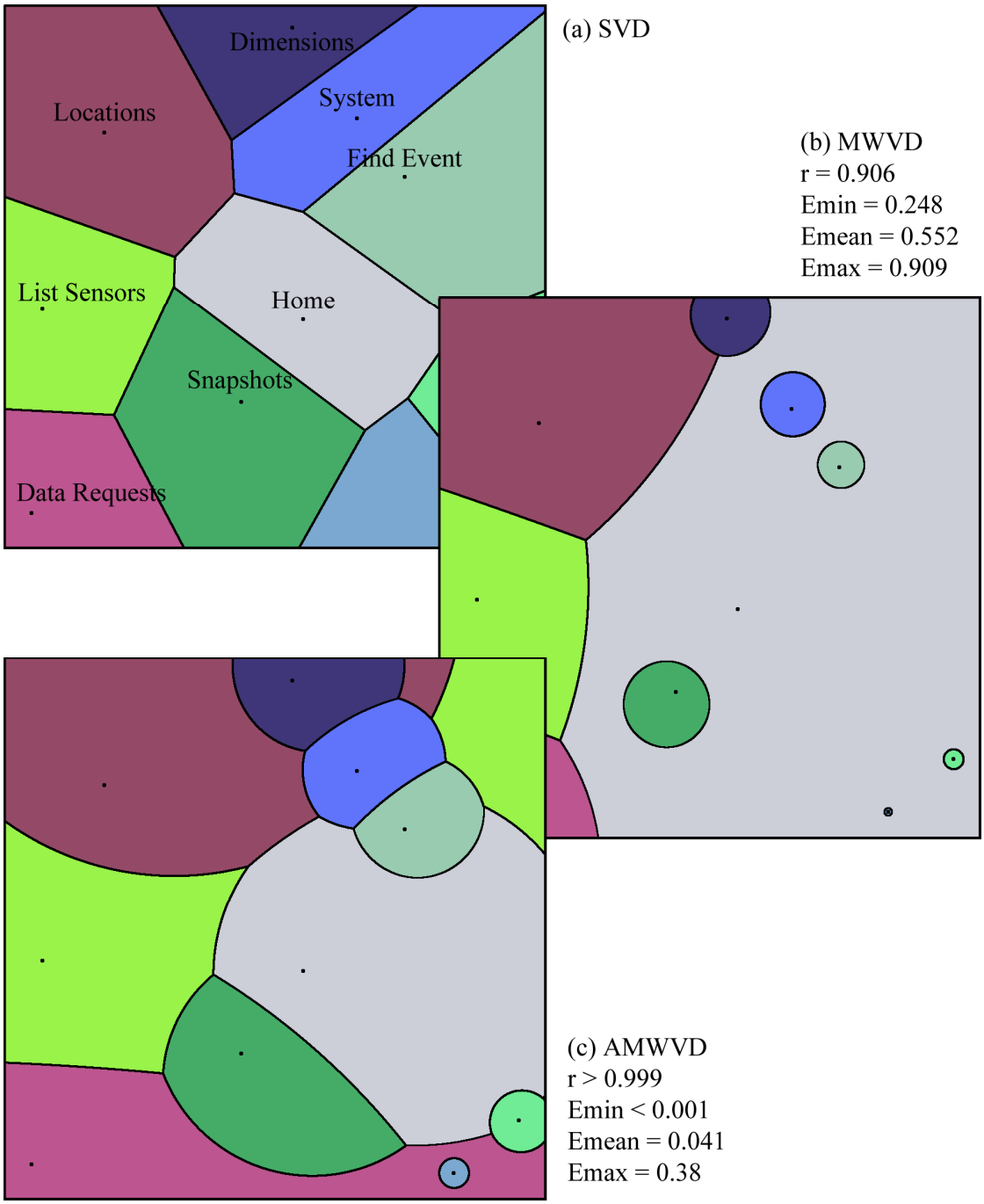



*Figure 26: AMWVD solution for BLT web page data. (a) SVD; (b) MWVD; and (c) AMWVD.*

## Chapter 5: Conclusion and Discussion

We presented the Adaptive Multiplicative Weighted Voronoi Diagram (AMWVD) as a method for partitioning an information space. The objective of the method is to simultaneously preserve both the location of the generators in the information space, and the proportionality between their magnitudes and areas. The basic algorithm of an AMWVD with key components was described, after which we introduced three methods to improve convergence: a variable weight adjustment factor *k*, updating of weights on improvement only, and sub-pixel resolution refinements. Time measurements show that tests with sub-pixel refinements produce better area-magnitude relationships than those without while consuming less time.

We conducted a series of tests to assess the algorithm's ability to produce an AMWVD diagram with pre-defined area relationships. These tests included three pathological (degenerated) cases and two empirical information space-partitioning applications. All of the tests showed successful convergence. However, it was noticed that because of our raster space model, specific problems may require increased spatial resolution.

The version of the AMWVD algorithm described in Chapter 3 uses Euclidean distance metric ($L_2$ metric) with further weighting. Experimenting with other metrics, such as $L_1$ may reveal new applications for an AMWVD. In our tests, we used the same factor *k* for all generators. However, one region might converge faster with a high value of *k*, while another region might converge faster with a small value of *k*.

The sub-pixel refinement method included only one level of pixel refinements. However, it is possible to reduce round-off errors even further with a recursive version of the sub-pixel algorithm, where pixels and sub-pixels at the border can be refined to arbitrarily deep levels. This would also mean that we could start a solution at rather low levels of spatial resolution. After all, we would recursively increase the resolution only in those places where it would be necessary. This, again, would contribute to the efficiency of the computations.

Finally, we could consider developing vector solutions similar to the ones proposed by Lan Mu [15]. In a vector space model spatial resolution does not apply. Hence, the entire operation of sub-pixel refinements would no longer be required. In fact, only one step of our algorithm needs to be changed in order to adopt vector-based VD calculations.

## Bibliography

[1] Andrews, K., Kienreich, W., Sabol, V., Becker, J., Droschl, G., Kappe, F., Granitzer, M., Auer, P., Tochtermann, K. (2002) The InfoSky Visual Explorer: Exploiting Hierarchical Structure and Document Similarities. Information Visualization. 1. 166–181.

[2] Buiding as a Learning Tool (2005) http://blt.colorado.edu. Accessed: February 15, 2005.

[3] Buttenfield, B.P., Reitsma, R.F. (2002) Loglinear and Multidimensional Scaling Models of Internet Transactions; International Journal of Human-Computer Studies. 57. 101–119.

[4] Chapra, S., Canale, R. (1998) Numerical Methods for Engineers with Programming Software Applications. Third edition. WCB McGraw-Hill.

[5] Chen, C. (1999) Information Visualization and Virtual Environments. Springer Verlag. London, UK.

[6] Chen, H., Houston, A.L., Sewell, R.R., Schatz, B.R. (1998) Internet Browsing and Searching; User Evaluations of Category Map and Concept Space Techniques; Journal of the American Society for Information Science. 49. 582–608.

[7] Deussen, O., Hiller, S., van Overveld, C., Strothotte, T. (2000) Floating Points: A Method for Computing Stipple Drawings. Computer Graphics Forum. 19(3). 40–51.

[8] Dodge, M. (2001) Atlas of Cyberspace. Addison-Wesley. Harlow, UK.

[9] Dodge, M., Kitchin, R. (2001) Mapping Cyberspace. Routledge. London, UK.

[10] Du, Q., Faber, V., Gunzburger, M. (1999) Centroidal Voronoi Tessellations: Applications and Algorithms. Society for Industrial and Applied Mathematics. 41(4). 637–676.

[11] Fabrikant, S.I., Buttenfield, B.P. (2001) Formalizing Semantic Spaces for Information Access; Annals of the Association of American Geographers. 91. 263–280.

[12] Huff, D.L. (1973) The Delineation of a National System of Planning Regions on the Basis of Urban Spheres of Influence. Regional Studies. 7. 323–329.

[13] Johnson, B., Shneiderman, B. (1991) Tree-Maps: a Space-filling Approach to the Visualization of Hierarchical Information Structures. Proceedings of IEEE Visualization 91, IEEE. 284–91.

[14] Mercier, F., Baujard O. (1997) Voronoi diagrams to model forest dynamics in French Guiana. Proceedings of the 2nd International Conference on GeoComputation. 161–171.

[15] Mu, L. (2004) Polygon Characterization With the Multiplicatively Weighted Voronoi Diagram. Professional Geographer. 56. 223–239.

[16] Okabe, A., Boots, B., Sugihara, K., Chiu, S.N. (2000) Spatial Tesselations. Concepts and Applications of Voronoi Diagrams. Wiley & Sons; Chichester, UK.

[17] Peuquet , D. J., Kraak, M.J. (2002) Geobrowsing: creative thinking and knowledge discovery using geographic visualization. Information Visualization. 1. 80–91.

[18] Reitsma, R., Trubin, (2005) S. Area-proportional Space Partitioning Using Inverse Voronoi Diagrams. Working Paper; College of Business, Oregon State University, Corvallis, OR.

[19] Reitsma, R., Trubin, S., Sethia, S. (2004) Information Space Regionalization Using Adaptive Multiplicatively Weighted Voronoi Diagrams. Proceedings Eight International Conference on Information Visualization IV 2004. IEEE Computer Society. Los Alamitos, CA. 290–293.

[20] Schvaneveldt, R. W., Dearholt, D. W., Durso, F. T. (1989) Graph theoretic foundations of Pathfinder networks. Computers and Mathematics with Applications. 15. 337–345.

[21] Shirley, P. (2002) Fundamentals of Computer Graphics. A K Peters; Natick, Massachusetts. 49–51.

[22] Skupin, A. (2000) From Metaphor to Method: Cartographic Perspectives on Information Visualization. In: Roth, S.F., and Keim, D.A. (Eds.) Proceedings IEEE Symposium on Information Visualization (InfoVis 2000). IEEE Computer Society. Los Alamitos, CA. 91–97.

[23] Skupin, A. (2002) A Cartographic Approach to Visualizing Conference Abstracts. IEEE Computer Graphics and Application. 22. 50–58.

[24] SmartMoney.com (2005) http://www.smartmoney.com/marketmap/. Accessed: December 16, 2005.

[25] Snibbe, S. (1998) Boundary Functions. http://www.snibbe.com/scott/bf/index.htm. Accessed: May 8, 2006.

[26] Voronoi, G.F. (1907) Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier Mémoire: Sur quelques propriétés des formes quadratiques positives parfaits. Journal für die Reine und Angewandte Mathematik. 133. 97–178.

[27] Voronoi, G.F. (1908) Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième Mémoire: Recherches sur les paralléllοèdres primitijs. Journal für die Reine und Angewandte Mathematik. 134. 198 – 287.

[28] Voronoi, G.F. (1909) Nouvelles applications des paramètres wntinus à la théorie des formes quadratiques. Deuxième Memoire: Recherches sur les

parallélloèdres primitijs. Second partie. Domaines de formes quadratiques correspondant aux différent types de parallélloèdres primitives. Journal für die Reine und Angewandte Mathematik. 136. 67–181.

[29] WebMap.com (2004) http://www.webmap.com. Accessed: October 5, 2004.

[30] Zhao, J., Su, P., Ding, M., Chopin, S., Ho, P.S. (2005) Microstructure-based stress modeling of tin whisker growth. Proceedings of 55th Electronic Components and Technology Conference. 1. 137–144.