ARRANGING ARBITRARY DATA INTO STRUCTURED LAYOUTS

GRANT STRONG

Arranging Arbitrary Data into Structured Layouts

by

© Grant Strong

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

Department of Computer Science
Memorial University of Newfoundland

May 13, 2013

St. John's Newfoundland

Abstract

Data. It can range from a single item to an exhaustive array of samples. It may be as simple as a binary label or as complex as a set of high dimensional vectors. It may be comparable by distance, by order, or not at all. In general, data is a term that encompasses a wide gamut. While specific data types vary in makeup and comparability, the common denominator is that it is always possible to take two elements of a data set and determine how dissimilar they are to some degree. The purpose of this thesis is to use that dissimilarity to arrange data into structured layouts in which each item occupies a unique cell. These layouts are meant to be simple, minimal, and free of occlusion. Three novel algorithms for this type of data arrangement are proposed. The first is an amalgamation of the Self-Organizing Map and the k-d tree. It builds a semi-structured layout that is later refined. It performs admirably but requires intermediary steps making it lack the speed necessary for scalability. The second algorithm, called the Self-Sorting Map, is new and is inspired by techniques from the field of dimension reduction. It approaches the problem from a sorting perspective and generates minimal, occlusion-free layouts from all types of data directly. It is fast, scalable, and parallel. The third algorithm transforms the problem from minimizing local dissimilarity, like the others, to maximizing global correlation. Entitled the Max Correlation Map, it is simpler and more robust than its counterparts. Throughout this thesis the three algorithms are analyzed and applied. Creative organizations of images, articles, and cities show the practical worth of organizing data into structured layouts.

Acknowledgements

Formally, it is my finest honor to thank my supervisor, advisor, and mentor Dr. Minglun Gong for everything he has done for me over the past five years as a Masters and Doctoral student. None of what we have accomplished together, or what I have accomplished academically and vocationally in my life, would have happened without him and his guidance. I must offer my highest thanks to David Mould, Antonina Kolokolova, and Andrew Vardy for their meticulous review of my work, both on paper and during the defense. I cannot forget Darlene Oliver whom I am indebted to for aligning all the stars so the defense could even happened on schedule. I would like to thank everyone from the students, to the faculty, to the staff in the Computer Science department at Memorial University. I would not trade my experiences here with you for anything. I also have to thank NSERC for their gracious funding throughout my program. A special thanks to Yoones Sekhavat for printing this document in my absence at the 11^{th} hour to allow me to graduate.

That said these formal thanks are not enough. So many astounding, amazing, and phenomenal things have happened in my life, particularly in the last year and a half. There is a big, detailed, informal story to be told and so many more to be thanked in it. For that reason I have written it and placed it in the Appendix. Maybe I am the first person to ever continue their acknowledgements in the form of a story in an appendix, but after five years in grad school there was a lot to say so I did.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

BMU Best Match Unit

LLE Locally Linear Embedding

MCM Max Correlation Map

MDS Multidimensional Scaling

SOM Self-Organizing Map

SNE Stochastic Neighbor Embedding

SSM Self-Sorting Map

Chapter 1 Introduction

Data comes in many forms. Raw high dimensional data is hard for human beings to understand. Our visual and analytical systems are brilliantly engineered to make sense of and extract patterns from complex pictures, yet they are ill-equipped to deal with tables of numbers directly. For this reason researchers have been trying for years to come up with better approaches to visualization to facilitate data understanding [45]. Many powerful algorithms have been designed to reveal the underlying structures of input data [39]. Due to the output of these techniques being unconstrained, complex interfaces are sometimes required for users to explore the results. This often limits the use of these techniques to experienced users only.

There are a host of techniques for displaying data of all kinds. Typically if data is of a hierarchical nature, trees are favored; if data is connected, graphs are used; if data is tabular, grids are favored; and if data is multidimensional it is projected into a space that can be readily visualized, like 2D or 3D, and then displayed in some interactive way. The first three are instances of structured layouts. The idea of organizing data into a structured layout is what is studied here.

Common problems when attempting to visualize data, especially a lot of it, are those of scope, focus, and occlusion. When data spans a broad range, it is hard to know exactly what things to show and what to leave out at different levels of detail. Things need to be summarized at different levels to accommodate users so that they are presented a

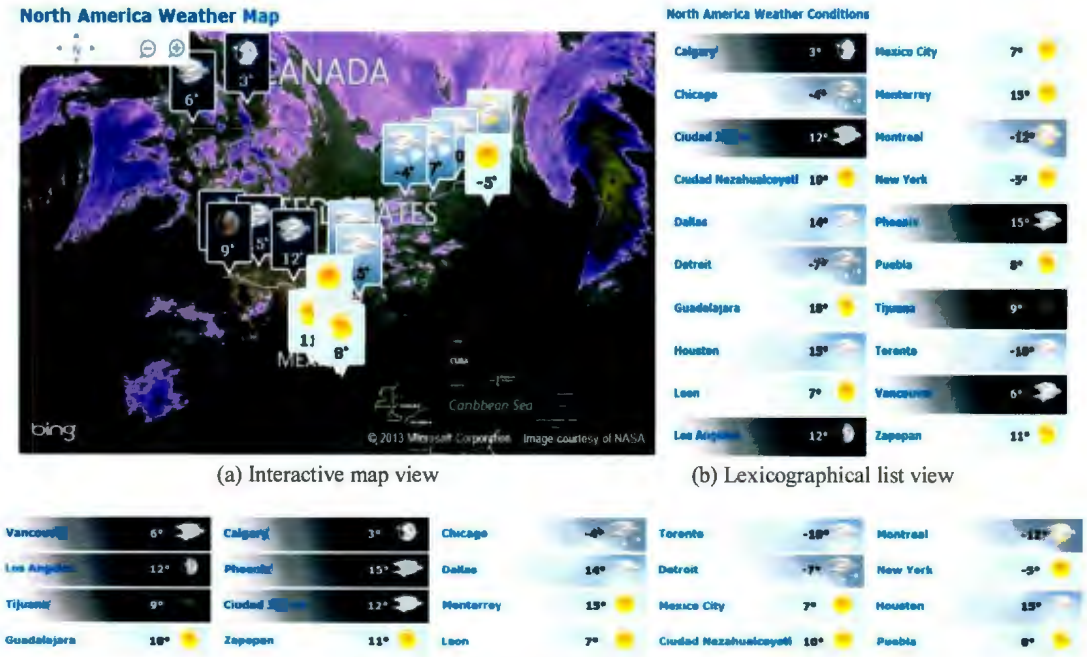
reasonable scope of information. The center of attention and the context surrounding it are key factors in how quickly someone can make connections between data and find what they are looking for, no matter where they start in the visualization. Creating smooth paths between data facilitates easy transition of focus as users guide themselves toward their goals. When a rendering space is saturated with data it is hard for things not to overlap, especially if the data items require a portion of the available screen space to be differentiated from others; for example when labeling is required or the items are images. Normally working around occlusion in these circumstances requires interactivity so that things can be translated, rotated, and scaled. Scaling requires collapsing of regions of data into representative items. If not, and things are zoomed by the size instead, they can shrink out of view or grow over each other.

A simple approach to addressing the issues of scope, focus, and occlusion is presenting data to viewers in a predictably structured layout like a tree, graph, or table. We see tables displaying data often. For instance, search results, photo albums, operating system file browsers, and many more use tables. The 2D grid layout of tables can help users to memorize the location of particular data items to facilitate re-finding. However, without an intuitive way of sorting the data, finding a data item for the first time requires the users to linearly scan through the table, which can be time consuming.

1.1 Motivation

This thesis presents novel ways of automatically laying out data using dissimilarity into structures where each item occupies a unique cell. Layouts with this property are

referred to as structured layouts here. Based on the key constraint of a one to one item to cell mapping, when an organization is complete, the structure of items has the appealing property that it can be visualized directly; see Figure 1. Organizations of data into structured layouts should exhibit the property that proximity reflects similarity, in the sense that like items are close together and unlike items are separated. A 1D grid structure with 1D numerical data items should be sorted comparatively to exhibit this property. As dimensionality increases in either the structure or the data, the processes proposed still strive to maintain this property. Optimally finding a solution to this arrangement problem is intractable, but by using a course-to-fine-grained method and some approximation, near-optimal solutions can be obtained in a fraction of the time required by brute force enumeration of the solution space. The techniques described work with any data over which a dissimilarity measure is defined. This measure should tell how unrelated two data items are, thus giving an idea of how far apart they should appear. For instance, distance measures like Manhattan, Euclidean, cosine similarity, and shortest path are useful for vector data. Using the relative dissimilarities of items, a structured layout can be generated. Using the principle that proximity should reflect similarity, the layouts produced are expected to be simple and intuitive to help ordinary users find the data they desire quickly without any learning curve.



(a) Interactive map view

(b) Lexicographical list view

(c) 2D grid layout generated using similarity

Figure 1: Presenting the weather of 20 key cities in North America. Top: Two presentations used by the AccuWeather (http://www.accuweather.com/en/north-america-weather). Bottom: Cities are arranged into a grid layout by a combination of country membership and geographic location. (a) Directly positioning the cities using their geographic coordinates causes occlusions, forcing users to interact with the map. (b) Sorting the cities lexicographically by their names into a 2D grid allows particular cities to be found quickly, but makes it difficult to grasp regional weather trends. (c) Arranging cites by country and location similarity in a 2D grid structure promotes quick finding without loss of the weather trend information. This layout is also valuable in situations where interaction is difficult or there is none like on a mobile device or a public billboard.

To describe it formally, the data organization problem is transformed into the following labeling problem: Let Ω be a set of data, for which a dissimilarity measure $\delta(\cdot,\cdot)$ is defined. Further assume that the dissimilarity satisfies the following constraints: $\delta(s,s) = 0$, $\delta(s,t) \geq 0$, and $\delta(s,t) = \delta(t,s)$ for $\forall s,t \in \Omega$. Now given a structured layout M with m ($m \geq |\Omega|$) cells, where the distance between cells is defined by the structure, the objective is to assign each data item $s \in \Omega$ a unique cell L_s, such that the following normalized cross-correlation is maximized:

$$\underset{L}{argmax} \sum_{\forall s,t \in \Omega} \frac{(\|P(L_s) - P(L_t)\| - \bar{P})(\delta(s,t) - \bar{\delta})}{\sigma_P \sigma_\delta} \quad (1)$$

where $P(x)$ is the location of cell x in the structured layout. $\bar{P}$ is the mean distance between any two assigned cells and $\bar{\delta}$ is the mean dissimilarity between any two data items. σ_P and σ_δ are the corresponding standard deviation values which normalize the correlation to the range $[-1, 1]$. $\|\cdot\|$ is a distance measure that is layout dependent. For instance, a tabular grid might use Euclidean distance while a tree could use shortest path.

The result of the above maximization places data items that are more related to each other closer together in the layout, and vice versa. Hence, it attempts to organize the data based on the topology defined by δ and within the space and structural constraints of M.

If Ω is sufficiently small then it may suffice to evaluate every possible labeling assignment and determine the one giving the highest correlation. However, since the number of possible assignments is a function of the factorial of $|\Omega|$ the problem soon becomes intractable as $|\Omega|$ increases. In these instances an efficient method that finds a near optimal solution is needed. This work presents the details of some algorithms that achieve that end with varying degrees of speed and accuracy.

1.2 Contributions and Overview

The core contribution of this thesis is a set of novel algorithms for organizing data into structured layouts; in essence these algorithms sort multi-dimensional data into multi-dimensional structures. The mainstay of the layouts generated by these algorithms is the property that proximity should reflect similarity and every item occupies a unique

layout cell. Structured layouts produced with this property can be visualized directly without any occlusion. Applications and evaluations are provided and discussion on the merit of each technique with respect to the results is also given.

The remainder of the thesis is organized as the follows: Chapter 2 discusses foundational work that influences, or is closely related to, the work presented throughout this thesis. It includes sections on dimension reduction and information visualization.

Chapter 3 presents the first algorithm for organizing data into structured layouts. It is a Self-Organizing Map (SOM) based method. This is the preliminary work that inspires the methods that follow.

The next algorithm in Chapter 4 is entitled the Self-Sorting Map (SSM). It is a multi-data, multi-dimensional pseudo-sorting algorithm. If a dissimilarity measure can be defined over the set of input items, the algorithm can run on them. It is flexible and fast, and delivers high quality data organizations in a directly useable form. Chapter 4 includes a theoretical complexity analysis of the algorithm and the speedup achievable through parallelization. To complement the analysis, actual benchmarks are given for quantitative evaluation.

The final algorithm in Chapter 5 is called the Max Correlation Map (MCM). It approaches the organization of data items in a structured layout in the same way as the SSM but optimizes directly on the overall correlation described in Equation (1). It is simpler and more robust than its predecessor and produces better results more

consistently as well. The down side is a loss of parallelism and a longer overall runtime, but often the tradeoff of quality for speed can be warranted depending on the data.

In their respective chapters, the algorithms are each evaluated and discussed. Their benefits and impediments are described with respect to their performances in organizing colors and images. Organizing 3D colors in 2D is the baseline experiment since the results of organizations of colors are relatively easy to visualize and evaluate. The Lab color space is used since the Euclidean distance between Lab color vectors represents the perceived similarity between their colors [23]. Since just arranging colors does not provide grounded real-world results on which to base initial comparative conclusions, arranging images using visual similarities and contextual metadata is the second evaluator in these sections. Organizing images based on visual [53] and conceptual similarities [55] and presenting them in a coherent layout has been proven to be useful and practical with regards to search [54]. Organizing images also introduces new challenges. The first major challenge is acquiring a set of images; the second is transforming those images into data that the algorithms can work with. The transformation process is described in the respective sections of the chapters but the acquisition of images is common to both. The images are obtained by pulling the top results from an expanded query. The "concept information" of each image is considered to be the sub-query the individual images were obtained with. A more elaborate description of the query expansion process can be found in the Appendix.

Chapter 6 is dedicated to some test applications of the algorithms. It contains the results of organizing different data types including artificial data, images, textures, Wikipedia articles, and cities.

Chapter 7 concludes the work with an overview of what the proposed algorithms bring to the field of data organization.

Chapter 2 Background

The techniques developed and examined throughout this thesis have their roots in the field of dimension reduction. They have the ability to take a variety of data types as input and generate structured layouts that provide a means of information visualization. Relevant background in these areas is expanded upon in the sections that follow.

2.1 Dimension Reduction

Dimensionality reduction transforms high-dimensional data into a meaningful representation at a lower dimensionality. What we want is for the reduced representation to correspond to the intrinsic dimensionality, or topology, of the original data. This means we look for the minimum number of parameters needed to account for the observed properties of the data [14]. There are a number of techniques that have approached the dimension reduction problem from a variety of angles. Some emphasize on the global properties of the data and others on the local ones. These techniques have been surveyed extensively [6, 60]. What follows is a general discussion of some of the distinctive methods that define the different styles of reduction.

Principal Components Analysis (PCA) [26] is an established and popular technique for linear data dimension reduction. It attempts to find a linear transformation (a lower dimensional linear basis) using the covariance matrix of the data. Once a transformation matrix is found, the original data is mapped to a low dimensional space via the new basis.

PCA works under the assumption the underlying data is linear so that the needed transformation matrix can be found.

The non-linear counterpart of PCA in terms of age and popularity is the manifold learning technique named Multidimensional Scaling (MDS) [4, 35]. It is often used to map data onto a 2D or 3D space for visualization. Given the function δ, defining the dissimilarity between every pair of data items, the goal of MDS is to find a set of vectors $\{x_1, \ldots, x_{|I|}\} \in \mathbb{R}^N$ for the set of items I that minimizes the cost function:

$$\underset{\{x_1,\ldots,x_{|I|}\}}{argmin} \sum_{\forall s,t \in \Omega} \left(\|x_s - x_t\| - \delta(s,t)\right)^2 \tag{2}$$

Intuitively, the set of vectors should model the relative similarity between the data items as positions in $\mathbb{R}^N$. Since this is posed as a least squares optimization problem, an approximate solution can be found using variety of techniques; generally gradient descent is used. A notable variation of the MDS is Sammon's mapping [42]. It attempts to minimize a different non-linear cost function that aims to preserve small distances:

$$\underset{\{x_1,\ldots,x_{|I|}\}}{argmin} \frac{1}{\sum_{\forall s,t \in \Omega} \delta(s,t)} \sum_{\forall s,t \in \Omega} \frac{\left(\|x_s - x_t\| - \delta(s,t)\right)^2}{\delta(s,t)} \tag{3}$$

The difference between MDS and the Isomap [56], Maximum Variance Unfolding [64], and Diffusion maps [31] techniques is that they modify the data's Euclidean distances in attempts to improve their reductions. Manipulating the distances has the potential to better unfold underlying manifolds in the data [60]. Other global techniques like a non-linear kernel variant of PCA [64] and neural network based approaches [20] are also proposed, which have been applied to dimension reduction with success.

The next family of techniques emphasizes the local properties of data and assumes that if they are represented accurately than the global information will be retained implicitly. Locally Linear Embedding (LLE) [41] and Stochastic Neighbor Embedding (SNE) [19] are two popular methods for local-based high-to-low dimensional embedding. The basic idea is that a set of weights (LLE) or neighborhood probabilities (SNE) that are derived from the high dimensional data points can be used to guide the solution of the low dimensional set of embedded points. LLE does this by solving for the matrix of weights W using a linear system set up from each point's neighbors (closest K points) to itself in the high dimensional space. The least squares error equation for this is the following:

$$\underset{W}{argmin} \sum_i \left(x_i - \sum_j W_{ij} x_j \right)^2 \tag{4}$$

Next, W is fixed and then the embedded low dimensional vectors $\{y_1, \ldots, y_{|I|}\} \in \mathbb{R}^N$ for each data item point are found using the same formulation:

$$\underset{\{y,\ldots,y_{|I|}\}}{argmin} \sum_i \left(y_i - \sum_j W_{ij} y_j \right)^2 \tag{5}$$

The assumption is that locally, in the high dimensional space, the data points can be described as linear combinations of their neighborhood of data points. With this in mind, LLE finds a low dimensional embedding that is a combination of linear clusters. SNE applies the same idea to a set of probabilities that define how likely a given data point will choose another data point as its neighbor. Gradient descent is then used to align the

probabilities of the high and low dimensional data. A common problem with these techniques is so-called "crowding", where distant points in the original high dimensional space with no close neighbors collapse those points that are close together in the center of the low dimensional output. t-SNE [59] is a variant of SNE that addresses this problem by introducing a repulsion factor.

Laplacian Eigenmaps [2] and Local Tangent Space Analysis [67] have the same goal as LLE of producing a local property oriented low dimensional embedding. The first uses a graph neighborhood and eventually the graph Laplacian. The second assumes that if there is local linearity in the data than there is a linear mapping from the high dimensional representation to its local tangent space. Both use their respective matrices to solve for eigenvectors that will eventually become the reduced space.

When input data can be represented as vectors in a high-dimensional space, rather than just a matrix of dissimilarities, the MDS problem degenerates into the so-called Multidimensional Projection (MDP) problem [37]. A well-known MDP technique is the Self-Organizing Map (SOM), which indirectly maps input vectors of N dimensions to positions in the map [29, 30]. An SOM consists of a network of interconnected units, each holds an N-dimensional, randomly initialized, weight vector. The weight vectors at different units are trained by randomly selecting an input vector and computing its Euclidean distance to all of the weight vectors of the SOM units. The closest vector found in terms the distance is called the best match unit (BMU). The weights of the BMU and its neighbors are then adjusted towards the input vector. Once this competitive training procedure is complete, all input vectors are mapped to the location of their BMUs [21,

49]. The SOM is an integral part of the first data organization technique presented in Chapter 3.

The dimension reduction techniques just discussed, with the exception of the SOM, allow any data item s to be placed at an arbitrary location x_s. This differs from the algorithms being proposed in this thesis since they enforce the constraint that s must be assigned to a unique cell L_s in the output structure. This output property transforms what is normally a continuous optimization problem into a discrete one that arranges things within the confines of a given structured layout.

2.2 Information Visualization

One use of dimension reduction is to convert multi-dimensional data into a 2D or 3D format that can be visualized [4, 28, 35, 42, 62]. Standard artificial datasets exist for testing the ability of dimension reduction techniques to unfold low dimensional manifolds that have been embedded in higher dimensional spaces. Some of these datasets include the swissroll, broken swiss, helix, and twinpeaks [60]. These datasets are used in Section 6.1 to compare the techniques proposed by this thesis with other dimension reduction techniques. The resulting structured layouts of the proposed approaches are visualized for inspection. In terms of images, the Brodatz texture set [5] is one of the standards for evaluation involving texture images [38]. In Section 6.2 it too is visualized in a structured layout.

Beyond the process of reducing dimensionality for facilitating visualization, actually how to do the visualization of a set of data items has been discussed and debated in a

plethora of studies and surveys. The reason for this is that visualization can amplify the cognition of analysts to determine, among other things, if items appear separately or in groups and to find relationships between known interest points and new ones [61]. Some of this literature focuses on the visualization of scientific data as a means of visual data mining or relevance feedback [36, 44, 63]. Others focus on alternative visual treatments of abstract data [12, 17, 44]. One such family of visual treatments is graphs or networks [8, 12, 57, 66]. In these, each node represents an item or a cluster of data and edges represent the connections between them. Techniques like edge bundling, radial presentation and edge ghosting [13, 18] have been proposed in an effort to generate readable visualizations when data is complex. Other approaches render hierarchically clustered data in alternative ways such as 3D blobs [48] or tree-maps [25] shown in Figure 2.

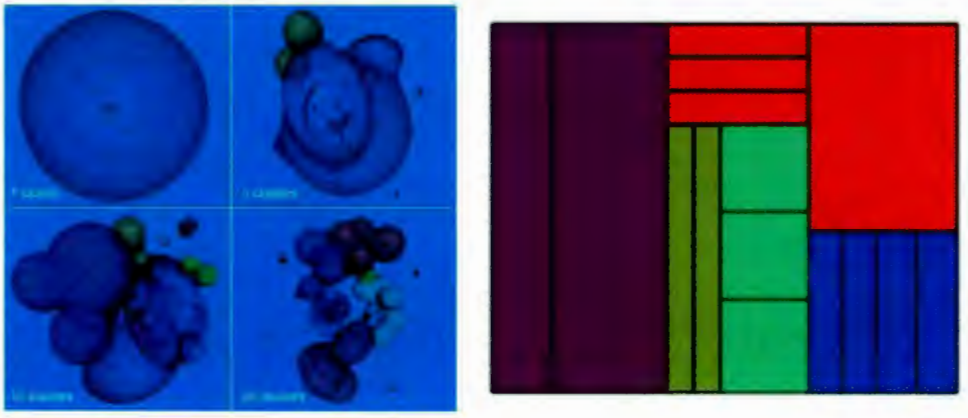


Figure 2: Hierarchical data visualization. Left: A hierarchical clustering of objects from [48] with 20 clusters displayed at different levels of the hierarchical cluster tree. Right: A tree-map from [25] with rectangles where each color belongs to the same level of the tree hierarchy.

Next we explore some of these visualization techniques in the context of the similarity-based image organization problem [53]. Several approaches have been devised

for visualizing images. VisualRank [24] is one that generates a similarity graph of images where the vertices are connected by weighted edges. As shown in Figure 3, the size of images changes depending on how connected they are.

Figure 3: The structure of a similarity graph from the VisualRank literature [24]. Notice that the enlarged ones are the most connected.

Torres et al. have defined representations of images as concentric rings or spirals [58]. The concentric representation consists of a series of rings where the position of each image determines its size. The spiral version is similar to its concentric counterpart in terms of image sizing but the placement can vary. Placement order on the spiral depends on rank with respect to a query image, but the placement positions can either be even or depend on the similarity with respect to the center image. Figure 4 shows examples of these visualization structures.

(a) Concentric rings (b) Ranked spiral (c) Similarity spiral

Figure 4: Alternative image presentation structures [58]. Note that changing image sizes are not reflected.

Chen et al. propose a way to visualize the contents of image databases by using pathfinder networks which is a structured modeling technique [7]. It is a branched clustering method based on the similarity between the images (i.e., histogram or texture in this case).

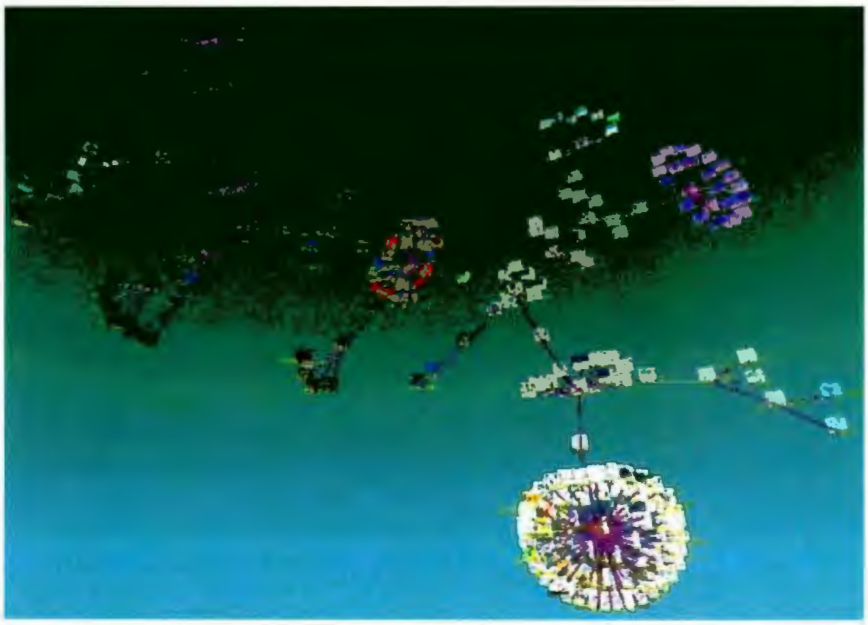


Figure 5: A pathfinder network based on the images' color histograms [7].

Snavely et al. have proposed how to arrange and browse large sets of photos of a scene taken by a community of photographers from different viewpoints [46]. The

proposed method exploits the common underlying 3D geometry in the scene to build a three dimensional rendition of it. This is different from the other methods discussed since its visualization is based on the assumption that it is given a set of images coming from the same region.

Figure 6: A snapshot obtained from Microsoft's Photosynth [33] which is built upon the work of Snavely et al. [46].

From what is presented here in the context of similarity-based image organization we see images being visualized in the standard structures of graphs and trees or in free space. The structured layouts generated by the techniques described in this thesis provide an alternative to these visualizations. The structured layouts are generated in such a way that distances between data reflects similarity, which can be appealing to users [40, 54]. They present things in a regularized format that allows for linear scanning of the organized data. Existing user studies have demonstrated that such a presentation format can

improve users' confidence in their searches [54]. The structured, rather than scattered, arrangement also facilitates re-finding of known data.

An alternative to linear scanning is Rapid Serial Visual Presentation [11]. RSVP, as it is called, is known to improve speed in search tasks particularly in the case of limited screen space on mobile devices [11, 16, 47, 65]. It requires constant attention to not miss desired results which increases the task load [16]. On the other hand, if the output structure is a 2D grid, then the resulting tabular representation is similar to existing interfaces employed today to show search results and hence is familiar to use (see Figure 1). The structured layout can be searched at a user's own pace, whereas in RSVP the pace is rapid as the name suggests. It also exhibits no occlusion which has merit for presenting information to users in situations where interaction might be difficult (mobile devices) or impossible (public billboards).

Chapter 3 Self-Organizing Map based Data Organization

3.1 Method

Here the cornerstone Self-Organizing Map (SOM) algorithm by Kohonen [29] is re-described in the context of this work. The SOM, being a dimension reduction technique, has been used for organizing data onto 2D planes before [49, 50, 52, 53]. However, the SOM cannot be used directly for the organization of data into a structured layout. This chapter discusses how to address this limitation by using the SOM for placing data into a 2D grid layout and then performing a k-d tree [3] post-processing alignment step.

3.1.1 Input Data

All of the data organization algorithms described in this thesis assume that the data input format can be a dissimilarity matrix, or alternatively a computable measure that is not necessarily vector based. If this is the case here, that matrix must be converted into a set of vectors so that the SOM can work with them. The vectors need to model the relationship information between data items as presented in the matrix as closely as possible. That is to say, the distance between any two vectors should be relatively approximate to, if not the same as, the dissimilarity between the corresponding items. This is the same as minimizing the following least-squares function:

$$\{V_1, \ldots, V_N\} = \underset{V_1,\ldots,V_N}{argmin} \sum_{1 \le i,j \le N} \left(\|V_i - V_j\| - \boldsymbol{D}_{i,j}\right)^2 \tag{6}$$

where $\|V_i - V_j\|$ is the Euclidean distance between the two vectors of items i and j, and $\mathbf{D}$ is the given dissimilarity matrix. The task of finding a set of vectors based on $\mathbf{D}$ is the classical multi-dimensional scaling (MDS) problem [4], which can be solved using existing techniques [1].

3.1.2 Self-Organizing Map

A SOM consists of $M \times M$ units, where each unit $\mathbf{x}$ has its own N-dimensional weight vector $\mathbf{W}(\mathbf{x})$. To train the SOM, on every iteration all of the items in the set of data items Ω are shown to the SOM in a random order. When a particular item i is shown to the SOM for training, the goal is to find the best match unit (BMU) and then update weight vectors in the BMU's neighborhood. To find the BMU location for a given i, $\mathbf{B}(i)$, a pass is done over the map to find the unit $\mathbf{x}$ that satisfies the following:

$$\boldsymbol{B}(i) = \underset{\boldsymbol{x}}{argmin}|\boldsymbol{F}(i) - \boldsymbol{W}(\boldsymbol{x})| \tag{7}$$

where $|\mathbf{F}(i) - \mathbf{W}(\mathbf{x})|$ is the Euclidean distance between the feature vector $\mathbf{F}(i)$ for the item and the weight vector $\mathbf{W}(\mathbf{x})$ for the unit $\mathbf{x}$.

Once the location of the BMU for i is found, the weight vectors of the nearby units will be adjusted to match $\mathbf{F}(i)$ more closely. These nearby units can be captured by $\omega_s(\mathbf{B}(i))$, which is defined to be the neighborhood window around the position $\mathbf{B}(i)$ with a radius of s, where s is the neighborhood size which is initialized to $M/2$ in practice. Updating the weight vectors for each unit $\mathbf{x} \in \omega_s(\mathbf{B}(i))$ can be done by applying the following to each:

$$\boldsymbol{W}'(\boldsymbol{x}) = \boldsymbol{W}(\boldsymbol{x}) + \lambda(\boldsymbol{x})\big(\boldsymbol{F}(i) - \boldsymbol{W}(\boldsymbol{x})\big) \tag{8}$$

where $\lambda(\mathbf{x})$ is the neighborhood influence of $\mathbf{x}$. The new weight vector for unit $\mathbf{x}$ is a linear interpolation between the vectors $\mathbf{F}(i)$ and $\mathbf{W}(\mathbf{x})$. The neighborhood influence $\lambda(\mathbf{x})$, which is the interpolation parameter, is computed using the Gaussian function:

$$\lambda(\boldsymbol{x}) = r \cdot e^{-\frac{\|x - \boldsymbol{B}(i)\|^2}{s^2}} \tag{9}$$

where $\|\mathbf{x} - \mathbf{B}(i)\|$ is the Euclidean distance between the units $\mathbf{x}$ and $\mathbf{B}(i)$ in the SOM. s is the neighborhood size and r is the learning rate, both of which decay exponentially over time based on these equations:

$$\begin{aligned} s &= s_0 \cdot e^{-\frac{t \ln s_0}{T}} \\ r &= r_0 \cdot e^{-\frac{t}{T}} \end{aligned} \tag{10}$$

where s_0 and r_0 are the initial neighborhood size and learning rate, t is the iteration number and T is the total number of iterations. T is the key component affecting the rate of decay of the neighborhood size and learning rate. Generally the more iterations the SOM is put through, the slower the decay and the better the final convergence. Figure 9 shows the intermediate and the final states of an SOM obtained during the training.

Once the SOM has converged, the final BMU for each item in Ω is found just as it was during training with Equation (7). The position of any item i in the final organization is simply the coordinate of $\mathbf{B}(i)$. By design the SOM algorithm seeks to preserve data topology as much as it can. This drives items having similar feature vectors to be mapped to locations that are closer to each other, and vice versa.

Since the BMU finding can be done via parallel reduction and the weight vector updates are independent, it is possible to implement the SOM in parallel so that it can run on GPUs [50] and other parallel platforms to yield an impressive 19 times speedup.

3.1.3 K-D Tree Grid Alignment

The final stage is to align the output coordinates into a non-overlapping grid. A grid layout makes it easy to enumerate all items within a given region. In the ideal scenario, training an SOM with the total number of units equal to the number of items would result in every unit being occupied by one item and hence all items lining up on the implicit grid formed by the network of SOM units. However, in practice the uneven distribution of the item feature vectors in the high-dimension feature space they come from will likely cause some units to become the BMUs of multiple items, while other units remain completely unmatched with any item; see Figure 9. For this reason, it is often better to use a SOM with more units than items. In this scenario the items will be able to influence a unique region of SOM units. Then, as shown in Figure 7, given the collection of items and their 2D SOM locations, a k-d tree algorithm [3] can arrange the items into a grid.

The algorithm starts by finding the median value among the horizontal coordinates of all items, and uses this to split the collection into left and right halves. It then computes the median value among the vertical coordinates of items in each half. Each half is further split into top and bottom quarters. These two steps are repeated until each node contains at most one item. In the end, all items are contained in the leaves of a balanced binary

tree. Based on the position of each leaf, a unique location is assigned to its associated item in the final grid layout.

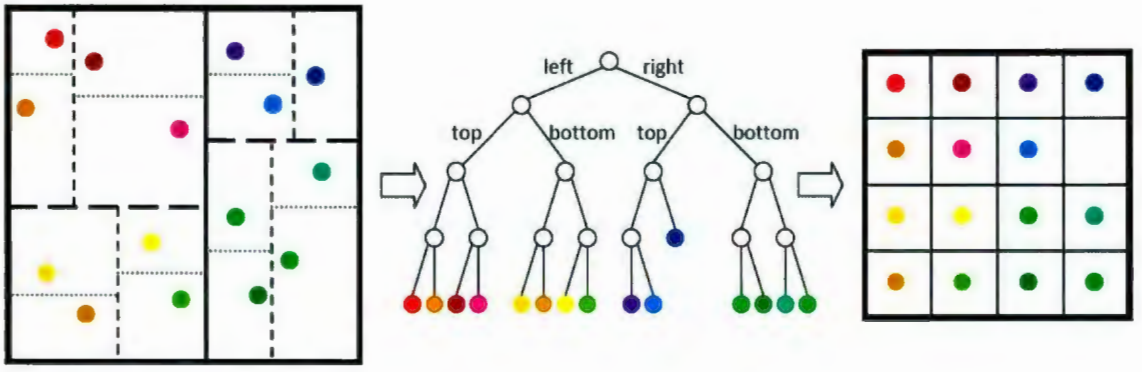


Figure 7: A depiction of the conversion of a continuous set of points to a regular, non-overlapping grid using a k-d tree.

It should be noted that after performing this translation from the SOM to the grid layout, there is some detailed information lost. In an unconstrained SOM layout the similarity between two neighboring items is visually encoded directly into their spatial distance at the resolution of the SOM. The resulting gaps and irregular placement of items provide a good representation of similarity through visual clustering, but makes sequential scanning of the items difficult to a viewer. Once the items have been aligned to a grid, the layout is regularized and easier to follow, but the fine-grained degree of similarity is sacrificed in lieu of this more structured overall appearance. Closeness in the grid still represents similarity, just not with the same accuracy.

At this point, a similarity-based organization of data items is achievable. Figure 8 gives pseudocode for this process of organizing a set of items based on their feature

vectors and aligning their positions to a grid layout. Figure 9 shows the process acting on sample data by breaking it down into images at key stages.

```
Randomly initialize all SOM weight vectors;
for a given number of iterations {
   Compute the new decayed learning rate r and neighborhood size s;
   for each randomly selected and unprocessed item i in Ω {
      Find the BMU B(i) using the item's feature vector F(i);
      for each x in ω_s(B(i)) {
         Compute the neighborhood influence λ(x);
         Update W(x) using linear interpolation with λ(x) and F(i);
      }
   }
}
for each item i in Ω {
   Find the position of the BMU B(i) and set i's position to it;
}
Align the positions of all items in Ω using a k-d tree;
```

Figure 8: Pseudocode for the Self-Organizing Map data organization algorithm with k-d tree grid alignment.

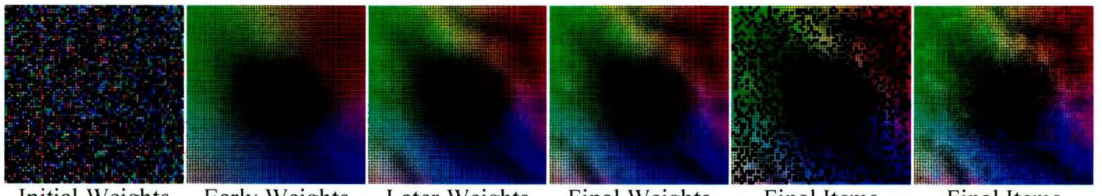


Figure 9: The progression of the organization of 4096 Lab color vectors by a 64 × 64 SOM with k-d tree alignment. The first image on the left shows the initial weight vectors. They are chosen randomly from the ranges of the input item vectors. The next three images show the evolution of the weight vectors at various iterations. The neighborhood and learning rate decay is evident as time passes. The last two images on the right show the items themselves. The first of the two is the unaligned set of items placed directly at the position of their best match units. Note the gaps as some SOM units remain unoccupied while others contain multiple items. The last image is the items at the positions acquired after the application of the k-d tree algorithm for alignment and the correlation score of the organization.

3.1.4 Boundary Conditions

In some applications data might be visualized with wrap around (i.e., overlays on a world map) and the results should be organized and displayed repeatedly without a boundary. To avoid seams among different tiles, the boundary of the SOM can be wrapped around. With wrapping, a BMU near a border will influence units that are close to the opposite border. Doing this also forces data that would normally retreat into a corner to cohabitate with the rest.

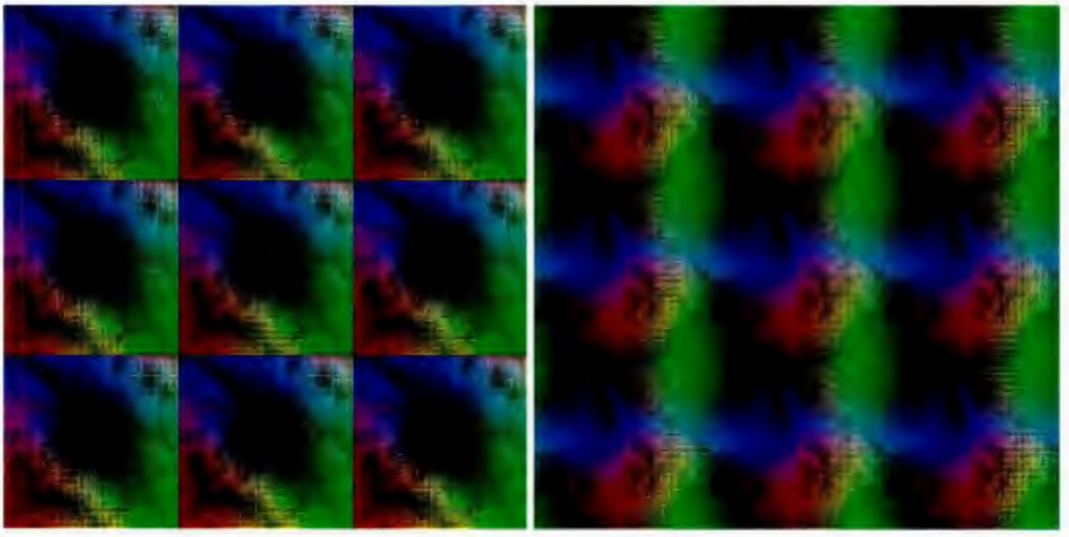


Figure 10: Two organizations of 4096 Lab color vectors repeated in 3 × 3 grids. Left: Generated using an SOM in the standard way. Right: Generated using an SOM with wrapping. Clusters of items form in rounder blotches since there is no way for them to retreat into a corner.

3.2 Results

3.2.1 Colors

In Figure 11 the result of organizing a set of 4096 Lab color vectors into 64×64 cells using an SOM is shown. For comparison, a Sammon's mapping version is also provided.

The Sammon's mapping is obtained using the Sammon projection component of HiSee [22] and the result of the SOM is generated using a previous implementation [21, 50]. Unfortunately, while MDS does admirably with RGB colors [34], it fails to organize the tested sets of random non-linear Lab color data well, collapsing its organization to a single dimensional line. For that reason it is not present here.

Visual inspection of the result of both the SOM and Sammon techniques confirms that data overlapping has occurred. To uncover the occluded data, the k-d tree is applied as described above to align all items. Nevertheless, artifacts (random and isolated dots) and a significant divide show up in the result of Sammon's mapping. This is understandable considering the result is not conducive to a grid layout having been arranged in continuous space. The SOM result on the other hand does exhibit the nice "color wheel" properties of the Sammon result, but without the salt and peppering of light and dark items.

The Sammon result has a higher correlation due to its placement of primary colors around the edges and the fact that the scattered positions of the light and dark items

balance with respect to their closest primaries. That said, the overall visual quality of the SOM result is locally smoother.

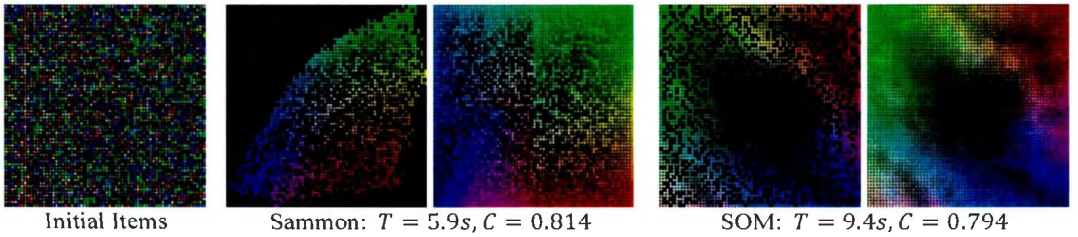


Figure 11: The organizations of 4096 Lab color vectors. The first image in the row is the starting set of items in both cases. Next are the original and aligned results of Sammon's mapping and the SOM. The grid aligned versions for these approaches are generated using k-d trees. Total organization time and final correlation between position and dissimilarity across the grid aligned maps is given below the respective images. All implementations are single threaded Java ones running on an Intel Core 2 Duo P8600 CPU at 2.4 GHz.

3.2.2 Images

The next example organizes a set of images obtained by searching with query expansion [21]. For a given query, the query expansion process finds several related concepts and searches for images using each of these concepts. Hence each image in the search result carries a semantic concept tag. The images can then be organized based on both concept and visual information using an SOM [21, 55]. The process to expand a given query, a Wikipedia one in this case, into concepts and acquire images for each concept is described in [21] and in more detail in the Appendix.

Once a set of concept tags are obtained from the query expansion, the extraction of conceptual vectors from them is not straightforward. To simplify the problem, assume that different images retrieved using the same sub-query (i.e., the same related concept) are conceptually the same and, hence have the same conceptual feature vector. Consequently, a vector $\mathbf{C}_k$ needs to be derived for each concept R_k that was used for

retrieving images. Even though it is difficult to convert concepts into vectors directly, a $N \times N$ semantic distance matrix for the N related concepts can first be computed using the Normalized Google Distance (NGD) from Equation (30) in the Appendix:

$$\boldsymbol{D} = \begin{bmatrix} 0 & NGD(R_1, R_2) & \cdots & NGD(R_1, R_N) \\ NGD(R_2, R_1) & 0 & \cdots & NGD(R_2, R_N) \\ \vdots & \vdots & \ddots & \vdots \\ NGD(R_N, R_1) & NGD(R_N, R_2) & \cdots & 0 \end{bmatrix} \tag{11}$$

where, by definition, we have symmetry since $NGD(R_k, R_k) = 0$ and $NGD(R_j, R_k) = NGD(R_k, R_j)$. Essentially NGD compares the shared hyperlink structure of two webpages, links from the pages and links to the pages, to compute a measure of similarity. Using this matrix which encodes the relatedness information among different concepts (using their respective Wikipedia articles as the pages), a set of m-dimensional vectors $\{\mathbf{C}_1, \ldots, \mathbf{C}_N\}$ is generated using MDS as described in Section 3.1.1 above.

The next step is to generate visual feature vectors for each image for use with the SOM. The distances between the vectors need to indicate the similarities between the corresponding images. Ways of generating feature vectors from visual information and their performances are studied in [52]. While these types of feature vectors can be used to organize images based on color and/or shape similarities, they cannot group conceptually related, but visually different, images together. To address this problem, a hybrid feature vector is used. The hybrid feature vector for an image contains two parts; a conceptual portion determined using the concept tag carried by the image and a visual portion extracted from pixel intensities and distributions. For the visual portion, a color-gradient correlation [52] approach which counts the combinations of the appearances of colors and

gradient orientations in a histogram is chosen. The reason for the choice is that it is efficient to calculate and offers good organizational performance [52].

To compute the color-gradient correlation histogram of an image I, the gradient magnitude l_p and gradient orientation θ_p for each pixel p is first computed by convolving the image with the Sobel edge operator [15]. The color and gradient orientation spaces are then divided into N_C and N_θ bins, respectively. With functions $C(p)$ and $\Theta(p)$ providing the color and gradient orientation bin indices for pixel p, the sum of gradient magnitudes for all pixels belonging to the kth color/gradient orientation bin of the histogram can be computed using:

$$m_k = \sum_{C(p)\times N_\theta + \theta(p) = k} l_p \tag{12}$$

The visual feature vector $\mathbf{V}(I)$ is then formed using the normalized values of all bins to make the vectors generated from images of different sizes comparable.

$$\boldsymbol{V}(I) = \frac{[m_1, \ldots, m_{N_C \times N_\theta}]}{\sum_{i=1}^{N_C \times N_\theta} m_i} \tag{13}$$

In the end, the hybrid feature vector for a given image I is formed as:

$$\boldsymbol{H}(I) = \langle \boldsymbol{C}_{R(I)}, \boldsymbol{V}(I) \rangle \tag{14}$$

where $\langle \cdot, \cdot \rangle$ concatenates the two vectors into one and $R(I)$ is the concept used to retrieve image I. Since the conceptual portion has m dimensions and the visual part has $N_c \times N_\theta$ dimensions, the total dimensions of a hybrid feature vector is $m + N_c \times N_\theta$, where $m = 4$. In general m should be at least the number of unique eigenvalues that can be

extracted from the matrix to minimize error since MDS relies on having an eigenvalue per dimension in the vectors it generates. In Figure 12 below $N_c = N_\theta = 8$ resulting 64 dimensional visual feature vectors and $m = 4$, which were found empirically.

In order to vary the contribution of the visual or conceptual information in the organization, a weighted average of the distances from each part is computed like so:

$$Dist(\boldsymbol{H}(I), \boldsymbol{H}(J)) = \alpha \left\| \boldsymbol{C}_{R(I)} - \boldsymbol{C}_{R(J)} \right\| + (1 - \alpha) \| \boldsymbol{V}(I) - \boldsymbol{V}(J) \| \quad (15)$$

where I and J are images and the parameter α controls the relative importance of the conceptual and visual distances. Note that the results of the Euclidean vector distances $\|\cdot\|$ are expected to be normalized with respect to the minimum and maximum values of all possible distances among the vectors of the given type. This is so the values fall in the range $[0,1]$ so that the conceptual and visual differences are comparable and the weight factor α has the expected meaning.

In order to make it possible to map distinct feature vectors to unique locations in the SOM settings should ensure that $M \times M \gg |\Omega|$. This is done so that each vector has room to put a region of units, rather than one, in the map under its influence. Even so, there is still no guarantee a balanced spread of influence will happen since the training is data dependent.

Figure 12 shows a small example of the ambiguous query "Washington", the expansion of which is described in the Appendix. Once the query has run through the entire process, a small representative sample of 64 images was organized using a 32×32 SOM run for 15 iterations with an initial learning rate of 0.1. For the distance measure

defined in Equation (15), the weight factor α has been set to 0.75 so that the concept vectors will get the majority of the influence and guide the overall layout while the visual content vectors will only affect the local arrangements. Both the unconstrained SOM layout and the grid aligned layout from k-d tree post-processing are shown. The results show that the colored regions related to the home articles of the query, "Denzel Washington", "Washington State", "Washington, D.C.", and "George Washington", are apparent. Visual similarity is also captured locally when comparing neighbors within regions. The aligned, uncluttered version of the organization displays no occlusion and does not require any extra interface controls to see all of the data.

$T = 0.21s, C = 0.635$ $T = 0.25s, C = 0.543$

Figure 12: Two layouts from a collection of images expanded out of the query "Washington". Left: Images are placed in positions directly from the SOM. Right: Images are placed at k-d tree aligned positions. In both, the images have color-coded borders relative to their category. The regions displaying "Denzel Washington" (red), "Washington State" (yellow), "Washington, D.C." (blue), and "George Washington" (green) are clearly visible. Within the regions, visually similar images also neighbor one another. The organization times and correlations are given. The unaligned SOM correlation is higher since images can be positioned in free space.

3.3 Discussion

There are some unavoidable issues with the SOM data organization approach. Firstly, the SOM is not capable of organizing the images using their concepts based directly on the semantic distance matrix. Converting the matrix to individual vectors for each concept using MDS is required. The vectors are approximates of the true distance matrix; see Figure 13. Next, the SOM often requires having more units than items it is trying to organize so it can produce well-spaced results without overlapping BMUs. This makes an already computationally intensive algorithm even more expensive. Even with all of the units, occlusion is evident when items are rendered since there is no constraint on where they should appear with respect to one another and some will inevitably overlap. The occluded output then needs to be transformed to a grid using a k-d tree. The k-d tree pays no heed the feature vectors of the image, but rather splits based on positions alone, arbitrarily sometimes in the cases where BMUs are shared. The k-d tree then can produce a result but it is not optimized to produce the best one.

	DW	GW	WDC	WS
Denzel Washington	0	0.024	0	0.517
George Washington	0.024	0	0.67	0.914
Washington DC	0	0.67	0	0.687
Washington State	0.517	0.914	0.687	0

Figure 13: The absolute differences between the given concept semantic distance matrix and a distance matrix computed from the MDS generated concept vectors.

Chapter 4 Self-Sorting Map

In this chapter a new data organization algorithm entitled the Self-Sorting Map (SSM) is presented. It is tested using the same data as the SOM-based one in Chapter 3. The SSM approaches data organization from a sorting perspective. It improves upon the SOM since it can work on the final structured layout directly and in doing so gets better speeds and eliminates the need for post-processing. It can also handle more input types than the SOM instead of vector data alone.

4.1 Method

The key idea of this algorithm is that data is swapped within the constraints of a structured layout instead of being placed at arbitrary positions [51]. To facilitate the understanding of the presented algorithm, it will first be explained in terms of positioning a set of numbers in a 1D array. Organizing general datasets into multi-dimensional structures will follow.

4.1.1 Sorting Numbers into a 1D Array

When the objective is to organize a set of numbers into a 1D array so that similar numbers are positioned together, the optimal solution can be obtained by simply sorting all of the numbers in either ascending or descending order. Being a fundamental computer science and mathematics problem, sorting has been extensively studied. There are many classic and efficient algorithms available for the task such as quicksort and

mergesort. Even though the SSM is designed to handle the organization of arbitrary data types in multi-dimensional structures, it can also perform 1D sorting.

As shown in Figure 14, given a set of numbers that are initially placed randomly inside a 1D array, the SSM first splits the cells into 2 blocks. All numbers in the first block are paired up with the numbers at the corresponding cells in the same relative positions of the second block. The two numbers in each pair (s, t) are compared against each other and an exchange is performed if $s > t$. After all pairs are processed in parallel, the first stage is completed.

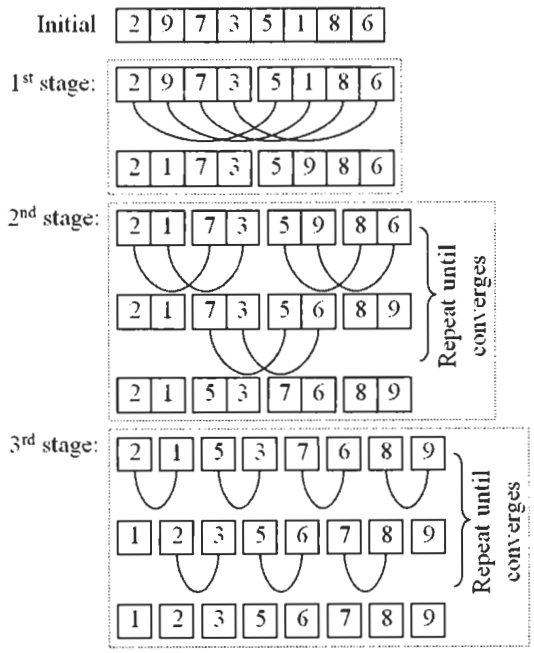


Figure 14: Self-sorting on a 1D array. The algorithm goes through multiple stages with decreasing block sizes.

The second stage further splits each block into two, resulting in four smaller blocks. Corresponding cells in adjacent (even, odd)-numbered blocks are first compared and

swapped if necessary, followed by corresponding cells in adjacent (odd, even)-numbered blocks. The even-odd swap and odd-even swap alternates until the process converges, i.e., all data at the corresponding cells of different blocks are sorted. The process then continues to the next stage by dividing each block into two, until the final stage is reached where all blocks contain only one cell.

The above process guarantees that the final result is a sorted array. If we only look at the final stage where each block contains just one cell, performing even-odd and odd-even swapping until convergence is essentially the odd-even sorting algorithm [32]. However, this approach is more efficient than odd-even sorting since it incorporates the basic idea of shell sort [43], i.e., using an increment sequence to allow the comparison and exchange of elements far apart. The increment sequence of $\{1, 2, 4, 8, \ldots, 2^k\}$ used here is less efficient than other candidates, such as $\{1, 4, 10, 23, 57, \ldots\}$ [10], but it fits well for parallel implementations due to the power of two constraint.

4.1.2 Organizing Multi-Dimensional Data

Multi-dimensional data, such as samples from the RGB color space, cannot be perceptually sorted on all components in a 1D sense. One could sort colors lexicographically based on the red channel first, then the green, and then blue, but such a sorting cannot guarantee that perceptually similar colors will be placed together. To organize multi-dimensional data, the following changes are made to the above baseline algorithm.

If the goal is to bring similar items together then the decision to exchange a given pair of data items (s,t) can be based on whether an exchange will reduce the total difference between the two data items and their neighbors. To perform this evaluation efficiently, we first compute a target vector T_i for each block B_i using:

$$T_i = \frac{1}{|\Omega(B_i)|} \sum_{B_j \in \Omega(B_i)} \frac{\sum_{s \in B_j} s}{|B_j|} \tag{16}$$

where $\Omega(B_i)$ is the neighborhood defined for block B_i; see Figure 15 for an illustration. In essence, this equation computes the target T_i of block B_i as the average of the means of different blocks inside the neighborhood. T_i is an aggregate representative of the items in B_i. Each neighborhood includes 4 blocks and as shown in Figure 15, the neighborhood windows of paired blocks are offset away from each other. The purpose of the offsetting is to encourage diversification among the computed targets.

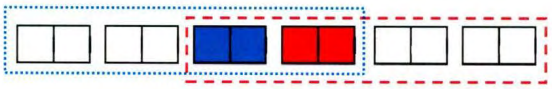


Figure 15: The neighborhoods used for target calculation when swapping data between two paired blocks (color-coded as blue and red). Each neighborhood contains 4 blocks.

Once the targets are calculated, the decision on whether a swap is needed for a given pair (s,t) from adjacent blocks, $s \in B_i$ and $t \in B_{i+1}$, is based on if the following expression can be minimized after swapping:

$$\underset{(s,t)}{argmin}(\|s - T_i\| + \|t - T_{i+1}\|) \tag{17}$$

where $\|a - b\|$ computes the distance between two vectors a and b.

The above test (and swap if necessary) can be performed for all pairs across the map in parallel. The target vectors are then updated based on the new data layout. The processes of computing targets and swapping are alternated until a convergence is reached, i.e., no more swaps are available and all target vectors stay constant.

Notice that there is a similarity between the above iterative process and the k-means clustering algorithm [27]. Both approaches alternate between finding the mean of each cluster and rearranging data into appropriate clusters. As a result, both can properly classify input data and convergence is guaranteed. That said, the k-means algorithm does not impose any constraints on data moving into (or out of) clusters, resulting in different clusters very likely having different numbers of data items. This approach on the other hand, only allows swapping between blocks, which ensures that all cells are occupied by at most one data item.

(a) grayscale color vectors

(b) RGB color vectors

Figure 16: Organizing data without using global comparison operator; only target distance minimization is used. The top row of each subfigure shows the initial (random) arrangement. The remaining 5 rows show the intermediate results after each stage.

Figure 16 shows the results of organizing both grayscale and color vectors into a 1D array using the approach just described. It demonstrates how data is rearranged without

requiring strict ordering information. Note that in Figure 16(a) grayscale vectors are sorted by intensity even though pairwise dissimilarity is used.

4.1.3 Organizing Nominal Data

Some datasets, such as a set of concepts related to a query, are not real-valued and hence the targets cannot be computed based on the above mean-based calculation. Assuming the dissimilarity $\delta(s,t)$ is defined for any given two data items s and t, here we discuss how to handle non-vector nominal data based on only $\delta(\cdot,\cdot)$.

The basic idea is to find a data item that best represents a given block B_i and use it as the target T_i. More formally, we compute the target as the data item inside the neighborhood of B_i that has the minimum total dissimilarity to other data in the neighborhood:

$$T_i = \underset{t \in \Omega(B_i)}{argmin} \left(\sum_{s \in \Omega(B_i)} \delta(s,t) \right) \tag{18}$$

where $\Omega(B_i)$ is the same offset neighborhood as defined above.

It is noteworthy here that the selected target is not limited to come from B_i only. It is allowed to be any data item within the neighborhood $\Omega(B_i)$. This expanded target search area lets more suitable targets be found in all stages of the algorithm including the last where there is only one data item per block. Notice that if the neighborhood target search is not used, in the final stage where there is one single data item to choose from per block, those items will automatically become the targets of their respective blocks. What

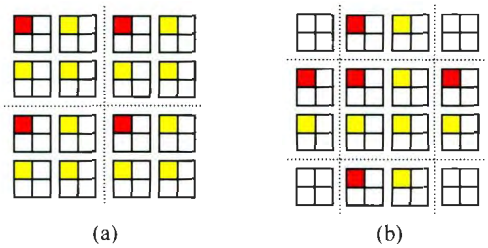
this means is that nothing would swap. Expanding the target search to the neighborhoods overcomes this by allowing items from neighboring blocks to be chosen as targets.

4.1.4 Handling 2D Structural Layouts

The algorithm posed so far can organize an arbitrary dataset into a 1D array, as long as a dissimilarity measure is defined. It can be extended further for organizing data into a 2D grid. The same principles of target finding and exchange can also be applied to organizing data into other lattice structures like a 3D grid or a 2D hexagonal grid.

As shown in Figure 17, when handling a 2D grid, all cells are split into 4×4 blocks first, each of which will be further split into four smaller blocks in the next stage. Even-indexed blocks are then grouped with odd-indexed blocks along both X and Y directions (see Figure 17(a)). This is followed by a shifted grouping in which odd-indexed blocks are grouped with even-indexed blocks along both directions (Figure 17(b)). Alternating between these two block group settings allows a given block to swap data with its four nearest neighbors, facilitating data moving toward the desired cells.

Once the blocks are grouped together, we determine the neighborhood $\Omega(B_i)$ for each block B_i, which is used for computing the target T_i. Similar to the 1D case discussed above, here we use windows that are offset from the centers of the respective block groups; see Figure 18. With the neighborhood determined, the target T_i for each block B_i is computed using either Equation (16) or (18) depending on the type of data to be organized.



(a) (b)

Figure 17: Splitting a 2D map into 4 × 4 blocks and grouping blocks using even-odd (a) and odd-even (b) settings. In both subfigures the red cell is matched to the three yellow cells from the grouped blocks.

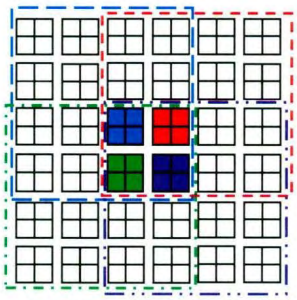


Figure 18: The neighborhoods defined for the four color-coded grouped blocks in the center. Each neighborhood encompasses a 4 × 4 block window that is offset away from the other blocks in the same group.

The next step is to swap data between the grouped blocks using the targets as the guides. Here data items in the corresponding cells of the grouped blocks form quadruples. For example, the data item in the red cell shown in Figure 17(a) is grouped with the ones in the three yellow cells to form a quadruple. The organizations for all quadruples are handled independently in parallel. To place a given quadruple (s, t, u, v) into four cells, there are $4! = 24$ possible alignments. All 24 possibilities are enumerated and find the one that minimizes the following:

$$\underset{(s,t,u,v)}{argmin}\begin{pmatrix}\delta(s,T_{i,j})+\delta(t,T_{i+1,j})+\\ \delta(u,T_{i,j+1})+\delta(v,T_{i+1,j+1})\end{pmatrix} \tag{19}$$

In practice, before each round of swapping the items within each block should be randomly shuffled. No items should leave the blocks they occupy so that the block targets do not change, but rather their positions are jostled so that the items grouped into quadruples are not the same every time. This gives badly positioned items a chance to move out of their block even if the map has converged because the items in their swapping quadruples can change each round.

Figure 19 summarizes the SSM algorithm in the form of pseudocode. Here the size of output grid is assumed to be $N \times N$, where N is power of two. The extension for rectangular and non-power-of-two grids is straightforward. Figure 20 shows the state of a 64×64 map of 4096 Lab color vectors after the algorithm has finished working at each block size. For quantitative evaluation, the cross-correlation values, as defined in Equation (1), are calculated and shown in the figure. As expected, the correlation is close to zero for the initial layout, where the color vectors are randomly placed. The correlation steadily increases as the organization goes through different stages as exchanges performed improve the organization. The final score reaches a high positive correlation between the colors and their positions in the structured layout.

```
Randomly place data into the layout as an initial arrangement;
while block size > 1 do {
   Split each block into 4 smaller blocks;
   do {
      for each of the two block groupings (even-odd and odd-even) {
         for each block B_{i,j} (i,j ≤ n) do
            Update the target T_{i,j} for B_{i,j} using Equation (18);
         Shuffle the items within each block;
         for each set of 4 blocks do {
            Group each cell s in the 1st block with cells
                  in the remaining 3 blocks;
            for every quadruple (s,t,u,v) do {
               Find the arrangement that minimizes the distance
                     between the items and the targets;
               Perform exchange if the arrangement has changed;
            }
         }
      }
   } while exchanges have occurred
         and the maximum number iterations is not reached;
}
```

Figure 19: Pseudocode for the Self-Sorting Map.

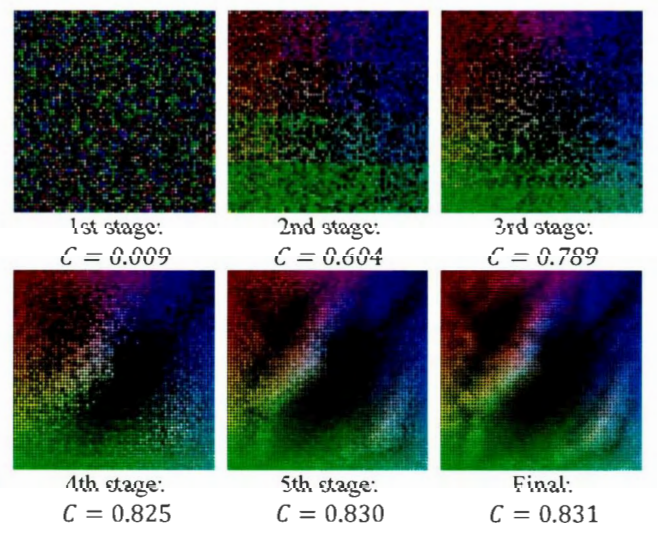


Figure 20: The progression of the organization of 4096 Lab color vectors after different stages. The corresponding correlation scores are given below the images.

4.1.5 Boundary Conditions

The algorithm discussed up until now does not provide any means to control the organization. In some applications, there may be a desire to have data arranged in

particular ways. For example, when positioning different cities in a 2D grid (see Figure 1), it is more familiar to have cities on the northern hemisphere placed on the top of the grid, even though flipping the grid upside down does not affect the cross-correlation score. Next shows how results can be influenced simply by adding boundary conditions to the proposed SSM approach.

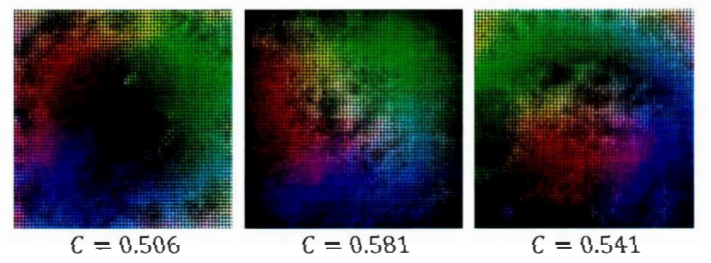


Figure 21: Results generated using additional constraints on the border of the map. Left: Using white color as boundary condition attracts bright colors to the boundaries and pushes dark colors to the center. Middle: Using black as boundary condition has the opposite effect. Right: Using white on top and black on bottom attracts colors of different brightness to different sides. Note that the correlation scores are much lower than the one obtained without any boundary constraints.

To achieve a desired result, users are allowed to specify data items outside the 2D grid. These data items do not participate in the swapping, but they are used during the target search calculation and hence influence the targets generated for blocks near the border. Consequently, input data items that are similar to the specified data items will be attracted toward the corresponding border of the 2D grid. As shown in Figure 21, by enforcing different boundary conditions, the same set of color samples is organized differently.

4.2 Parallel Implementation

The Self-Sorting Map algorithm can be broken down into two major stages; computing a target for each block and then swapping data between blocks based on those targets. Both stages can be performed using parallel kernels on streaming multiprocessors (SMPs) available in GPUs. In this section the design of those parallel kernels is described, as well as analysis of the time complexity of the algorithm in parallel versus its serial counterpart.

4.2.1 Target Generation Kernel

Taking the current data layout as input, the target generation kernel computes a target for each block of data items. The actual calculation involved depends on whether the mean or centroid is used. Here we will discuss how to compute mean targets first, followed by centroid targets.

Since the neighborhoods defined for adjacent blocks overlap each other, directly calculating the mean target using Equation (16) will result in redundant computations. A more efficient approach is to pre-compute the means of all blocks using only the data items inside the blocks. The result forms a 2D mean map whose resolution depends on the current block size. The mean map is then used to calculate the average in each block's neighborhood. To better utilize the parallel processing power of SMPs, the mean map is generated using parallel reduction. That is to say, a mean map for blocks of 2×2 cells is calculated first, which is then used to calculate a mean map for 4×4 sized blocks, and so on, until the desired block size is reached.

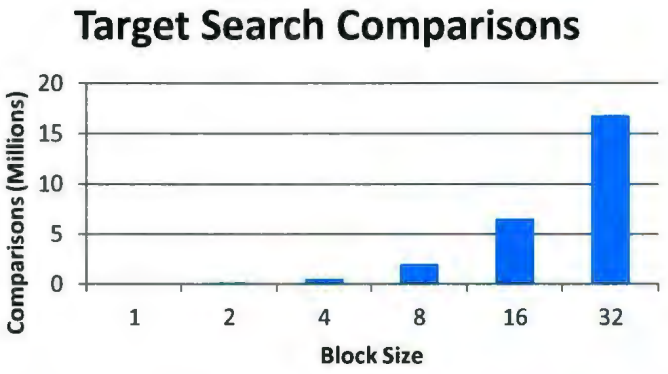


Figure 22: The number of comparisons needed for computing the true centroid targets of different block sizes increases dramatically as the number of data items in each block does.

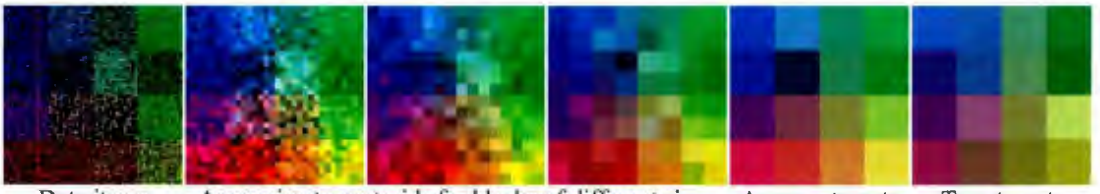


Figure 23: The first subfigure is the original data after items have been swapped into 16×16 blocks. The next four images show the approximate centroid maps obtained using parallel reduction, leading up to the approximate centroids of the 16×16 blocks. The last image shows the true centroids of those blocks calculated directly using Equation (18). The approximate centroids are similar yet more vibrant.

Compared to mean target calculation, centroid target searching is a much more expensive operation. For example, if we are given a map of 64×64 data items and want to find the target for a 16×16 sized block, we have to search all of the items in that block and the neighboring blocks to find the item that has the minimum total dissimilarity among all of the others for the purposes of this algorithm. This leads to millions of comparisons, as shown in Figure 22. Borrowing ideas from the mean target calculation, we can mitigate a lot of this complexity by computing approximate targets rather than the true ones. Since far fewer operations are required to compute the centroids of smaller

blocks than of larger ones, we find the approximate centroid of a given block by repeatedly performing a centroid search on 2×2 sized blocks in a parallel reduction fashion; see Figure 23. The approximate centroids for all blocks form a centroid map, which is then used to find the approximate target of each block, i.e., the centroid of centroids of the neighboring blocks.

4.2.2 Approximate Centroid Initial Block Size

Changing the initial block size at which the approximate centroid parallel reduction starts can affect the speed and quality of the final outcome. Figure 24 shows the same set of data organized by the SSM multiple times. The only independent variable in the experiments is the initial block size from which the parallel reduction starts. It is clear from the time chart in Figure 24 that the search time can be dramatically lowered if the initial block size used is smaller. This stems directly from the fact that the first reduction has the potential to be the biggest bottleneck if the initial block size is large, confirmed by Figure 22. The only downside to approximate centroids is that the final organization quality can suffer depending how small of an initial block size is used. This effect is noted in the correlation graph below which shows a decrease in correlation at smaller initial block sizes. In Figure 24 the middle range initial block size of 4 yields acceptable results in a fraction of the time.

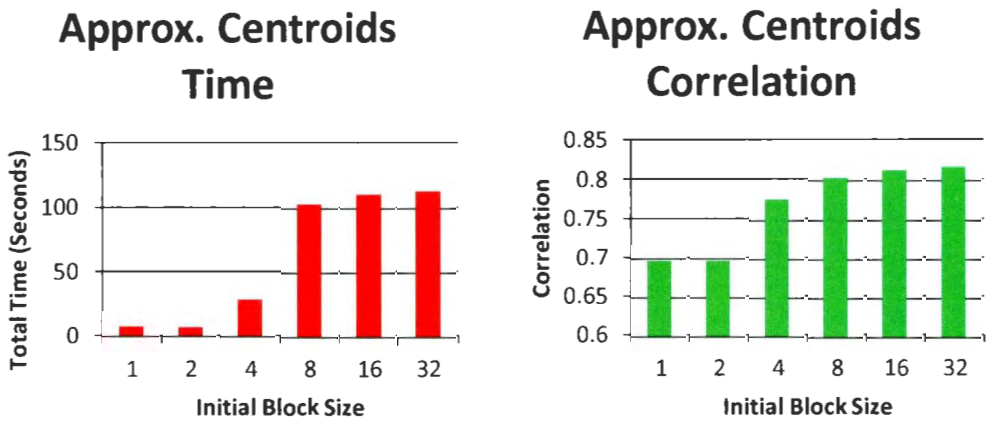


Figure 24: The charts show the total time and correlation respectively. An initial block size of 2 means that centroids for any block size larger than 2×2 would be built by getting the centroids at block size 2×2 and then reducing that map of centroids by half repeatedly until the centroids for the desired block size have been approximated. In this case an initial block size of 4 yields the best trade off of quality versus speedup.

4.2.3 Data Swapping Kernel

Once the targets are determined, the second stage is carried out by a single kernel that swaps items between grouped blocks in such a way that the dissimilarity between the items and the targets of those blocks is minimized. To do this, the kernel looks at each item in parallel in the context of a quadruple of four items formed by taking one item from each of four grouped blocks. Items chosen for each quadruple only ever belong to one such quadruple so as to maintain mutually exclusivity and thus the ability to have items swapped between blocks in parallel without conflict. Once four data items are grouped, they are swapped using the procedure described in Section 4.1.4, i.e., the four items are aligned with the four targets by checking all 24 possible combinations to find the one yielding the lowest total dissimilarity.

4.2.4 Execution

The two stages of finding targets and swapping items occur in that order and are repeated together as the algorithm dictates. Eventually the targets come to represent the items in their blocks well and few swaps need to be done. At this annealing point the algorithm moves on to the next block size where the map is considered to have more blocks that contain fewer items. The kernels continue to work as defined even with changing input sizes. Figure 25 illustrates a simplified version of the parallel kernels executing on some data.

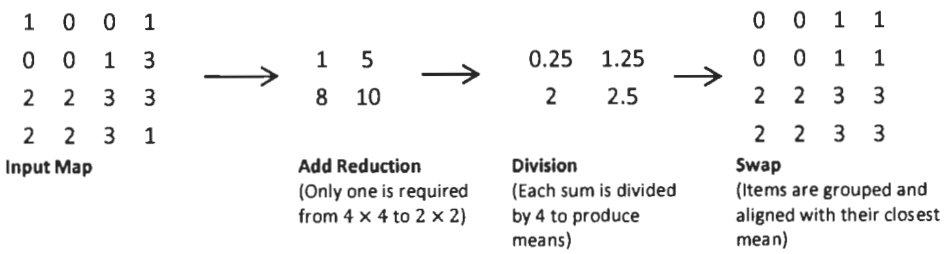


Figure 25: One iteration of kernel execution. Here means are only generated from the items that are contained inside their respective blocks for simplicity.

4.2.5 Performance Considerations

In parallel implementations memory access can be a bottleneck. Random access reads from many threads at once will result in the reads being queued and performed sequentially. This negates a lot of the effort of running things in parallel. For this reason it is important to make use of the faster local shared memory available in parallel hardware like graphics cards. As opposed to each thread reading data into registers from slower global memory where the host process places data initially, chunks of data needed by all threads sharing a local memory bank should be copied into that bank in parallel and

then read from there. To do this, each thread will usually load the element its thread identifier aligns with. Doing this, all data will be loaded by all threads in conjunction in contiguous swathes. This is known as coalesced loading. Once the data is in local memory, random access to it from the group of cooperating threads sharing it is much faster. Based on this, every kernel consists of three major portions: a coalesced data read into shared local memory, computation on the data in local memory, and a coalesced write of data from local to global memory.

In parallel hardware there are generally multiple independent compute units available. These are the groupings of computational resources that share things like local memory banks. Data should be divided up into workable chunks to maximize the concurrent use of these compute units. An optimal data division is usually one that produces enough local workgroups (OpenCL terminology) to activate all of the physical cores (CPU) or stream processers (GPU) available. This is because each local workgroup is executed by a separate physical compute unit. The goal is normally to have something for each unit to be doing at the same time to achieve the best efficiency. In most cases the local workgroup size is constrained by the local memory available for input and output data to fit into. Usually with enough data, the local memory limitations will require the problem to be divided up to activate all compute units. Based on this guidance the determination of the optimal local workgroup size for each kernel is a matter of coming up with a size that fits as much data as possible into the local memory, lower bounded of the number of available compute units.

Most of the data reading and writing stages of the SSM algorithm have a direct mapping of continuous areas of global memory into and out of local memory. The exception to the rule is when the swapping block group size requires more local memory than the device provides. In this case multiple local workgroups work on different areas of the same group of blocks. When this happens four smaller loads of the small areas from each of the four blocks in the group need to be done by each local workgroup. The load happens slower using this scheme but is required for correctness. Figure 26 depicts the different memory access scenarios. Keep in mind that a single local workgroup executing all of the swaps is not an effective use of the hardware if there are multiple compute units available. This example merely demonstrates what is occurring at a small scale.

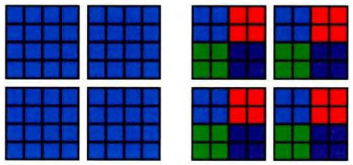


Figure 26: Block group memory access patterns of four 4×4 blocks. (a) is when the block group completely fits into local memory and is loaded by one local workgroup. (b) is the interleaved pattern when the block group cannot fit into local memory and is loaded by multiple local workgroups.

4.2.6 Complexity Analysis

To organize a set of N data items into a $\sqrt{N} \times \sqrt{N}$ map, the SSM algorithm needs to go through $\log_2 \sqrt{N}$ stages. On the k^{th} stage, the data is split into 4^k blocks with each block containing $N/4^k$ cells. Finding the mean for each block requires $O(N/4^k)$ operations since every item in the block needs to be added in. On the other hand, finding

the true centroid requires $O((N/4^k)^2)$ because every item needs to be compared with each other item in the block to determine the one with the lowest total dissimilarity. Hence at the k^{th} stage, the total number of operations needed for target generation of all 4^k blocks is $O(N)$ for mean and $O(N^2/4^k)$ for centroid. Regardless of whether mean or centroid targets are used, the data swapping step requires $O(N)$ operations since each data item is processed only once because it only belongs to one quadruple and quadruples do not share items.

The target generation and data swapping steps are repeated until convergence or the maximum number of iterations, L, is reached. L is a user imposed upper bound on iterations. The number of iterations required may vary per application since different datasets may converge at different times. Hence, we simplify the analysis here and consider the worst case time complexity of the SSM algorithm, where L iterations are used at all stages. Under this scenario, the serial SSM algorithm using mean targets requires $O(L \cdot N \cdot \log N)$ operations to complete all $\log_2 \sqrt{N}$ stages. When centroids are used, the time complexity can be computed as:

$$LN^2 + \frac{LN^2}{4} + \frac{LN^2}{4^2} + \cdots + LN = O(L \cdot N^2) \tag{20}$$

Now assume that we have a parallel computer with N processors. On the k^{th} stage, where each block contains $N/4^k$ cells, the time needed for computing the mean targets or the approximate centroid targets using the aforementioned parallel reduction approach is $O(\log(N/4^k))$, whereas the time required for data swapping is $O(1)$. Hence, the time complexity for the fully parallel version of the algorithm is:

$$L\log(N) + L\log\left(\frac{N}{4}\right) + L\log\left(\frac{N}{4^2}\right) + \ldots + L = O(L \cdot (\log N)^2) \qquad (21)$$

The speedup of the parallelization is $O(N/\log N)$, with the efficiency being $O(1/\log N)$. To see how these theoretical runtimes play out on real hardware, see Section 4.3.2 for some benchmarks.

4.3 Results

In order to exercise the proposed algorithm, the SSM was tested on the same data as the SOM in Section 3.2. Benchmarks from the execution of a serial and a parallel implementation of the algorithm on a CPU and a GPU, respectively, are also given.

4.3.1 Colors

In Figure 27 the results of organizing a set of 4096 Lab color vectors into 64×64 cells using the SSM is shown. This is the same set of colors used in Section 3.2.1 with the SOM. For this example the SSM used mean targets with the maximum number of iterations per stage set to 4. The quality of the SOM result after alignment is similar to the SSM's in terms of the topology, which is likely due to the fact that both techniques incorporate a type of decay into their neighborhoods over time (i.e., the SSM blocks split). In the end though, the SOM requires more computation and k-d tree post-processing to eliminate occlusion.

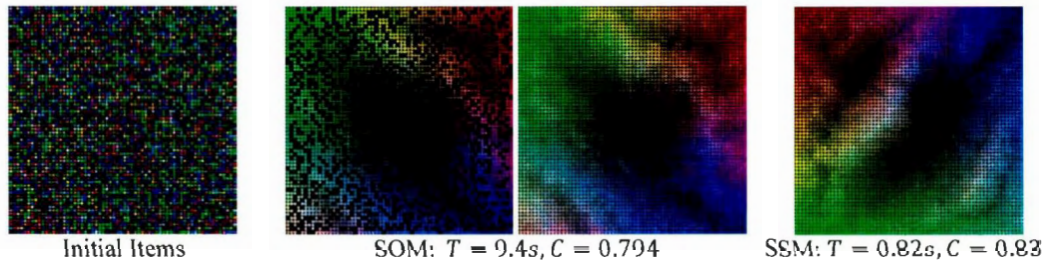


Figure 27: The organizations of 4096 Lab color vectors. The first image in the row is the starting set of items in both cases. For comparison, the SOM result and the k-d tree generated grid version is shown. Total time and final correlation between position and dissimilarity across the grid aligned maps is given below the respective images. All implementations are single threaded Java ones running on a Intel Core 2 Duo P8600 CPU at 2.4 GHz.

Note that the Sammon's mapping result presented previously in Figure 11 also takes more time and produces distinctly different visual appearance. The SSM directly generates the desired result of the highest correlation score of those tested in the shortest time.

4.3.2 Images

Here the same set of images that were organized by an SOM in Section 3.2.2 will be organized by a SSM. Again, concept tags are available for each image which is obtained through a query expansion process that is elaborated upon in the Appendix. The same hybrid vectors used by the SOM can be used here. A change is that the SSM also requires no k-d tree post-processing step like the SOM. This is because items are swapped around directly in the structure they are eventually displayed in.

Use of the SSM does not require all vector data like the SOM if centroids are used. This means that no MDS is needed to generate a set of vectors for the concept tags. Instead, the distance measure used by the SOM in Equation (15) becomes a combination

of the dissimilarities directly from the matrix **D** defined in Equation (11), and the visual feature vectors defined in Equation (13). This new distance measure is:

$$Dist(I,J) = \alpha \boldsymbol{D}_{R_I,R_J} + (1-\alpha)\|\boldsymbol{V}(I) - \boldsymbol{V}(J)\| \tag{22}$$

where I and J are images and the parameter α controls the relative importance of the conceptual distance and visual distance as it does in Equation (15).

Figure 28 shows the same small set of 64 images from the ambiguous query "Washington". This time the images have been organized by an 8×8 SSM run to convergence. Both the mean and centroid target layouts are shown. For the distance measure defined in Equation (22) used in the centroid driven layout, the weight factor α has been set to 0.75 so that the concept vectors will get the majority of influence and guide the overall presentation while the visual content vectors will only affect the local arrangement. In both cases images align to a grid implicitly from the fact that one image will always occupy at least one position in the SSM.

The results show that the images related to the home articles of the query, "Denzel Washington", "Washington State", "Washington DC", and "George Washington". Visual similarity is captured locally when comparing the borders of neighbors within regions. Notice that the mean layout is nicer in terms of the consistency in the shape distributions of the concept regions than the centroid, even though the dissimilarity matrix is approximated by MDS generated vectors. This is also confirmed by the marked improvement in the correlation. Reasons for this are discussed in Section 4.4 below.

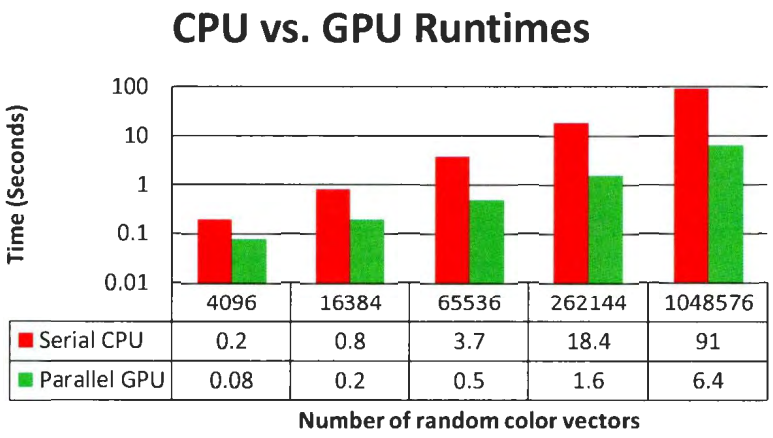
Means: $T = 0.148s, C = 0.679$ Centroids: $T = 0.141s, C = 0.482$

Figure 28: Two layouts from a collection of images expanded out of the query "Washington". Left: Images are organized using a mean driven SSM. Right: Images are organized using a centroid driven SSM. In both, the images have color-coded borders relative to their category. The regions displaying "Denzel Washington" (red), "Washington State" (yellow), "Washington DC" (blue), and "George Washington" (green) are visible. Within the regions, visually similar images also neighbor one another. The times and correlations for each are given.

4.3.3 Benchmarks

Here we measure the processing speeds of both serial and parallel implementations of the SSM algorithm over color vector datasets of different sizes. The serial version is implemented using a single-threaded Java program, whereas in the parallel version the core functions are replaced with OpenCL kernels, which are invoked through JOCL bindings. To make the time measurements comparable across different implementations and datasets, we force the algorithm to go through 5 iterations per stage regardless whether the process converges or not. The CPU used is an Intel Xeon E5540 running at 2.5 GHz. The GPU used is a NVIDIA GeForce GTX 480 which has 480 streaming multiprocessors across 15 compute units running at 1.4 GHz. The timings in Figure 29

show that the parallelism of the SSM allows for a 14 times speedup for the largest dataset tested.



	4096	16384	65536	262144	1048576
Serial CPU	0.2	0.8	3.7	18.4	91
Parallel GPU	0.08	0.2	0.5	1.6	6.4

Figure 29: The running times of the serial implementations versus the parallel GPU implementation on datasets of varying size. The parallel GPU implementation can arrange a million items in 6.4 seconds.

4.4 Discussion

The approach presented is entitled the Self-Sorting Map due the inspiration it garners from the Self-Organizing Map. Given a set of data, both approaches map them to a multi-dimensional structure while trying to preserve the topology of the input data in terms of similar data items being placed near to one another and dissimilar being placed apart. Hence, both can be used to visualize a high-dimensional data space in a lower one. The main difference is that the SSM works to solve the discrete labeling problem of optimal item placement by swapping items between cells. Every item is always guaranteed to have a position in the SSM grid. In contrast, no item has a position assigned to it in the SOM. It does not actually hold the items in its units at all. The units contain only the

weight vectors, and so, finding the position of an item in an SOM is done indirectly by searching for the BMU of that item and using its position. SOM's often return the same BMUs for multiple data items if they are similar. If the results are to be visualized directly without occlusion with one item per cell grid cell like the output obtained directly from the SSM, the SOM's BMU-based positions must be post-processed using something like a k-d tree [55] to generate unique grid cell locations for each item. A second difference is that since the SOM uses weight vectors, input items must be representable by vectors or else they cannot be mixed with the weight vectors of the SOM which are critical to its design. In all, if a minimalistic grid free of occlusion is desired then the SSM can produce the result directly with only one item per grid cell more quickly than the SOM. The SOM often requires a many to one unit to item ratio (i.e., a larger map than the number of items being arranged) to generate quality BMU positions which then have to be post-processed. The SSM can also accept the dissimilarity matrix as is rather than requiring it to be converted into vectors. In saying this, it is clear from the images that have been organized in Figure 28 that when the set of nominal data items is sparse, as the concept dissimilarity matrix does dictate them to be in this case, good centroid items cannot be found to represent blocks of items because the items in those blocks are too varied. As a result the quality of the organization suffers. On the other hand, notice that the mean organization is of good quality even though MDS generated concept vectors, which introduce error in dissimilarity, were used. This is because fully numeric items can be aggregated to form a new mean item that lies between those in each block's neighborhood. In essence, the centroid is a pseudo-mean that approximates the true mean

that must be chosen from the items in a block neighborhood. When there are a lot of items in a block and they lie at opposite ends of the spectrum then the algorithm has trouble converging to a set of good representative centroid items for the blocks, meaning that a high quality global layout will not be established in the early stages.

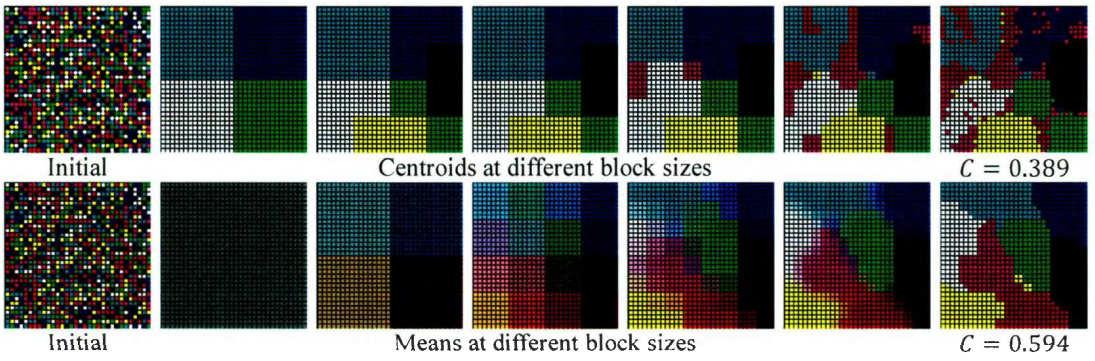


Figure 30: The result from running the SSM on a set of binary RGB color vectors. Every channel of each vector item is only allowed to have full or no color. Each row of images presents the initial items, the centroids (top) or means (bottom) of different block sizes, followed by the final organization of colors and the correlation value. Given the fact that good centroid items cannot be found at different levels, the final result of the centroid SSM suffers. Since the means are generated by blending items, its result converges to a better layout.

To replicate the problem broken concept clusters appearing in Figure 28 more clearly, the centroid and mean versions of the SSM can be run on binary RGB color vectors, which are essentially a set of nominal valued colors (red, green, blue, cyan, magenta, etc.) in vector form. The results of these runs are shown in Figure 30. Much like with the images before, the centroid version produces a scattered result with broken regions since there is no good choice for centroids. This is because the centroids must represent whole blocks of heterogeneous items. Take the first centroid image: cyan, blue, white, and green are chosen to represent the four large blocks of the initially random item arrangement. These four colors are close to some items and yet very far from others by

the polarized nature of the binary data. This trend continues until the blocks get small enough that a homogeneous set of items exists in and around the blocks so that good representative centroids can be chosen. At this point it is too late and the damage to the global layout is already done. In the mean example, since items in blocks are blended a good global layout can be established from the beginning. The later stages just refine the edges of it. The mean SSM layout is not without fault though. There are some yellow items that get separated and stranded due to the fact that the mean at the border of the green and red regions does produce a dull yellow. Since the block size is small at this point, and only local neighborhood jumps are possible, those yellow items have no way of swapping out of their positions.

While the SSM does improve upon alternative data organization algorithms for constrained grid layouts in terms of quality and efficiency, Figure 30 shows that its centroid incarnation cannot handle sparse data well and that the algorithm is more greedy overall than the alternatives. Due to the greediness, there is more potential to be trapped by local minima because of the inability for items to jump globally, especially in the later stages where block neighborhoods are small. In many cases these weaknesses are acceptable given the superior speed at which results can be obtained, particularly when using mean targets. Section Chapter 6 presents many applications where this is so.

Chapter 5 Max Correlation Map

This section presents a data organization algorithm that has the same properties and features as the SSM while working in an alternative way. Decisions to move data items are dictated by the overall correlation of the map. The Max Correlation Map (MCM), as this approach is called, is simpler than the SSM in design, while being more robust and able to produce improved results. The strength of this technique is the global optimization based on correlation. The major downside is that correlation is expensive to compute and results in a higher overall computational cost than the other techniques that were previously described.

5.1 Method

The major difference in the design of the MCM versus the SSM or SOM approaches is reformulation of the problem as an optimization that uses the correlation coefficient as an objective function directly. This makes the MCM approach more flexible and able to generate organizations that are of higher quality globally than the other methods. It has the ability to handle all types of data and layouts that the SSM can, with the addition of being able to place items in structured layouts with some items being fixed and/or enlarged or cells being removed.

5.1.1Problem Definition

Here the problem of arranging data items into a structured layout is generalized further in the context of the MCM.

Given a dataset Ω, assume there is a dissimilarity function δ defined such that for any two data items s and t in Ω, if $s = t$ then $\delta(s,t) = 0$, otherwise $\delta(s,t) \geq 0$. Now assume there is a structured layout Γ that contains n cells, where $n = |\Omega|$. We define a mapping $M: \Omega \rightarrow \Gamma$ as a function that assigns each data item in the dataset Ω to a cell in the structured layout Γ. M is occlusion-free if no more than one data item can be assigned to any given cell in Γ. According to this definition, the SSM in Chapter 4 is an occlusion-free map, whereas the SOM in Chapter 3 is not.

The objective is to find an occlusion-free map where the proximity of data items in the structured layout correlates with the similarities among them. More formally, such a map is one that attempts to maximize the following Pearson correlation coefficient:

$$\rho(M) = \frac{1}{\sigma_\psi \sigma_\delta} \cdot \frac{1}{|\Omega|^2} \sum_{\forall s,t \in \Omega} \big(\psi(M(s), M(t)) - \bar{\psi}\big)\big(\delta(s,t) - \bar{\delta}\big) \tag{23}$$

where $\psi(\cdot,\cdot)$ is the distance between two cells, $M(\cdot)$ returns the cell to which a given item is mapped, and where $\bar{\psi}$ and $\bar{\delta}$ are the means and σ_ψ and σ_δ are the standard deviations of the distance and dissimilarity measures, respectively.

Different distance functions, such as Euclidean distance or geodesic distance, can be used to define $\psi(\cdot,\cdot)$. If we fix the structured layout Γ and the dataset Ω, then the means

and standard deviations are constant. Equation (23) can be simplified by defining normalized cell distance and data dissimilarity measures as:

$$\Psi(u,v) = \frac{\psi(u,v) - \bar{\psi}}{\sigma_\psi}, \Delta(s,t) = \frac{\delta(s,t) - \bar{\delta}}{\sigma_\delta}$$

Thus, in simplified form, the equation becomes:

$$\rho(M) = \frac{1}{|\Omega|^2} \sum_{\forall s,t \in \Omega} \Psi\big(M(s), M(t)\big) \cdot \Delta(s,t) \tag{24}$$

5.1.2 Cell Correlation

Here we consider an alternative way to compute ρ. Since the mapping function $M: \Omega \to \Gamma$ is one-to-one for an occlusion-free map, its inverse function $M^{-1}: \Gamma \to \Omega$ exists. Equation (24) can also be expressed as:

$$\rho(M) = \frac{1}{|\Gamma|^2} \sum_{\forall u,v \in \Gamma} \Psi(u,v) \cdot \Delta\big(M^{-1}(u), M^{-1}(v)\big) = \frac{1}{|\Gamma|} \sum_{u \in \Gamma} \rho_u \tag{25}$$

where ρ_u evaluates how well the data item stored in a given cell u correlates with the remaining cells. Here ρ_u is referred to as the cell correlation score and is computed using:

$$\rho_u(M) = \frac{1}{|\Gamma|} \sum_{v \in \Gamma} \Psi(u,v) \cdot \Delta\big(M^{-1}(u), M^{-1}(v)\big) \tag{26}$$

Figure 31 visualizes the correlation scores for different cells. By definition in Equation (25), the overall correlation of a given map is equal to the average of all cell correlation scores.

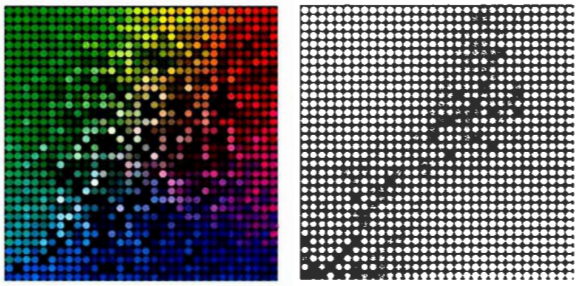


Figure 31: Left: An organization of 1024 random Lab color vectors. Right: Visualization of the correlation scores at different cells, where high intensity represents high correlation score. In this instance the lower left corner is the worst correlated area.

5.1.3 Partial Correlation Updates

Given a mapping M with its correlation $\rho(M)$ evaluated, we first discuss how to efficiently compute the correlation $\rho(M')$ for the updated map M', which is obtained by swapping two data items in M. The key to partially updating ρ lies in the fact that it can be described as the sum of the cell correlations, as shown in Equation (25). If two items, s in cell u and t in cell v, are swapped then every cell correlation needs to be updated to account for the change. This is done in two phases: First, the differences in correlation between the old item placement and the new item placement must be applied to all other cells, and secondly, the cell correlations for u and v need to be recomputed. These operations can be described in the following way:

$$\rho'_w = \begin{cases} \rho_w + \Psi(w,u)\big(\Delta(M^{-1}(w),t) - \Delta(M^{-1}(w),s)\big) \\ \quad +\Psi(w,v)\big(\Delta(M^{-1}(w),s) - \Delta(M^{-1}(w),t)\big) & If\ \ w \in \Gamma\backslash\{u,v\} \\ \sum_{i\in\Gamma} \Psi(w,i)\cdot\Delta\big(M^{-1}(w),M^{-1}(i)\big) & If\ \ w \in \{u,v\} \end{cases} \tag{27}$$

where ρ'_w is the new cell correlation score for cell w.

Equation (25) can then be used as usual to compute the updated correlation $\rho(M')$ after the swap. By doing updates in this way, only the necessary changes to the cell correlations are carried out and the recalculation of all cell correlations on every swap is avoided.

5.1.4 Enumerated MCM Search with Pruning

One way to solve for the best mapping $M: \Omega \rightarrow \Gamma$ is by enumerating all possibilities using "brute force" evaluation. Discussing this approach is useful for evaluation of the optimal solutions of small datasets.

To enumerate all mappings we start with an empty map and fill the cells one by one recursively. Once all of the cells are filled, we have an occlusion-free mapping. The recursive process backtracks and continues forward again with a different items until all possible mappings are evaluated and the one with maximum correlation is found. Even though it is possible to find the true best mapping with this approach, the computational cost is too high for practical use, as shown in Table 1. In the rest of this section we discuss a novel way of reducing the computational cost by pruning the recursive tree of possible mappings being searched.

Table 1: The number of solutions needs to be evaluated with and without pruning. Without pruning, the solution space becomes impractical to search for maps as small as 4×4.

Map Size	# of Solutions Evaluated w/o Pruning	# of Solutions Evaluated w/ Pruning
3×3	362,880	6,043
4×3	479,001,600	66,099
4×4	20,922,789,888,000	21,380,134

Rather than naïvely examining each potential mapping independently, we can consider the above brute force search process as depth-first traversal of a decision tree with all possible mappings represented as leaves in the tree. We can prune a branch during the traversal if we know none of the leaves in the branch has a higher correlation than a known solution. To apply pruning, we need a way to compute an upper bound correlation score $\bar{\rho}$ for a given partially filled map. The upper bound is computed by assuming that each unfilled cell u is occupied by an ideal data item, whose dissimilarity to any other cell v is identical to the distance between u and v. Hence, the upper bound of the cell correlation score for a given cell u is:

$$\bar{\rho}_u(M) = \begin{cases} \sum_{v \in F} \Psi(u,v) \cdot \Delta\left(M^{-1}(u), M^{-1}(v)\right) + \sum_{v \in \Gamma \setminus F} \Psi(u,v)^2 & if\ u \in F \\ \sum_{v \in \Gamma} \Psi(u,v)^2 & if\ u \in \Gamma \setminus F \end{cases} \quad (28)$$

where F is a subset of Γ containing occupied cells.

Summing together $\bar{\rho}_u(M)$ for all cells gives us the upper bound correlation score $\bar{\rho}(N)$ for a partially filled map N. In this case, placing a data item s into a given cell u means replacing the imaginary perfect data in u with the real data s, which will leave or lower the correlation. Hence, if the upper bound $\bar{\rho}(N)$ is already smaller than the correlation $\rho(M)$ of an already found full map M, there will be no reason to conduct any further search beyond the partial solution N. This allows the corresponding subtree to be pruned during the depth first traversal.

In practice, a solution M should be computed using one of the faster techniques like the SSM and its value $\rho(M)$ uses as the initial pruning threshold. Once a better solution M' ($\rho(M') > \rho(M)$) is found during the traversal, the value $\rho(M')$ is used instead. As shown in Table 1, pruning can greatly reduce the number of solutions that need to be evaluated. It is also worth noting that during the traversal when a data item is filled into an existing map N, the upper bound $\bar{\rho}(N')$ for the new map N' can be efficiently updated based on the original $\bar{\rho}(N)$ using the approach discussed in Section 5.1.3.

While pruning does cut off searching to the leaves of the search tree in a lot of cases, building up partial solutions through the inner nodes of the tree still requires a lot of time. In reality, even small problems are impractical to solve with this approach. Figure 34 shows the dramatic increase in search time even for small datasets.

5.1.5 Coarse-to-Fine Swap-based MCM Search

From studying the enumerated MCM search approach, it is clear that the performance cost is too high to find the optimal one every time. The SSM on the other hand is an approach that moves items around at a high level first, and then works its way down to only moving things around in smaller and smaller regions until eventually the region is one item. The hierarchical approach generates quality results very quickly, but since it is primarily a local sorting algorithm problems can arise if a good global layout is not achieved in the early stages.

Here an MCM search method is presented that addresses the locality issues associated with the SSM by sorting based on the correlation score directly. For this

method, data swapping is performed in a coarse-to-fine manner where the cells in the map are initially grouped into a small number of large blocks. Each of the blocks is further split into smaller blocks in the next block level, until the finest block level where each block contains only one cell is reached. Figure 32 shows an example of this coarse-to-fine block splitting scheme. At a given block level, the cell distance function is adjusted to $\psi'(u,v) = \psi\big(C(u), C(v)\big)$, where function $C(\cdot)$ returns the center cell for the block that u belongs to. Once the distance measures are adjusted, the corresponding mean $\overline{\psi'}$, standard derivation $\sigma_{\psi'}$, and normalized distance $\Psi'(u,v)$ are updated as well. Such a change in the distance measure ensures that $\Psi'(u,v) = 0$ as long as u and v belong to the same block. Consequently, the exact cell position where a data item is placed inside a given block does not matter, allowing the swap-based optimization to focus on moving data into the proper block.

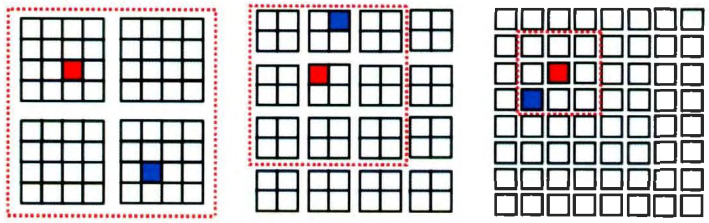


Figure 32: Illustration of how data are swapped in three levels: 2 × 2 blocks (left), 4 × 4 blocks (middle), and 8 × 8 individual cells (right). The neighborhood for the same red cell is marked with a red dashed box, in which its swap partner (shown in blue) is randomly selected.

To move data at a given block level, the algorithm cycles through every cell u in the map and looking for a random cell in u's block neighborhood to swap the data with. This neighborhood is a 3 × 3 block window with the block of u at the center, as seen in Figure 32. After the swap the correlation is updated using Equation (27). If the correlation

increases the change is kept, otherwise it is reverted. Look to Figure 33 for pseudocode of this MCM algorithm.

```
Randomly map the items from Ω to cells in Γ;
while block size b > 1 do {
   Split each block into 4 smaller blocks and modify b;
   Update ψ̄, δ̄, σ_ψ, and σ_δ and then ρ at the new block size;
   do {
      for each cell u in Γ {
         Select a random cell v from one of the 8 neighboring
               blocks adjacent to u's block;
         Swap items M^-1(u) and M^-1(v);
         Partially update ρ;
         if ρ decreased {
            Swap M^-1(u) and M^-1(v) back;
            Partially update ρ;
         }
      }
   } while changes have occurred
         and the maximum number iterations is not reached;
}
```

Figure 33: Pseudocode of the MCM algorithm.

5.1.6 Comparison

Figure 34 contains a comparison between the enumerated Brute Force, MCM, and SSM generated mappings. The implementations were in Java and only the Brute Force one was multi-threaded. An Intel Xeon E5540 CPU with 4 cores at 2.53 GHz was used. Note that while all of techniques place related colors together (i.e., light and dark hues) and distinctly different colors apart (i.e., red, blue, green), the correlation of the SSM is lower because it does not consider the global arrangement in as much detail as the others do. For instance, intense colors should lie in the corners like in the Brute Force arrangements but they do not in all of the SSM's. However, the search time for the Brute Force algorithm at just 4×4 makes it impractical. The relative correlations and

similarity in visual quality between the MCM and Brute Force optimal are much closer than those of the SSM making it the best compromise here.

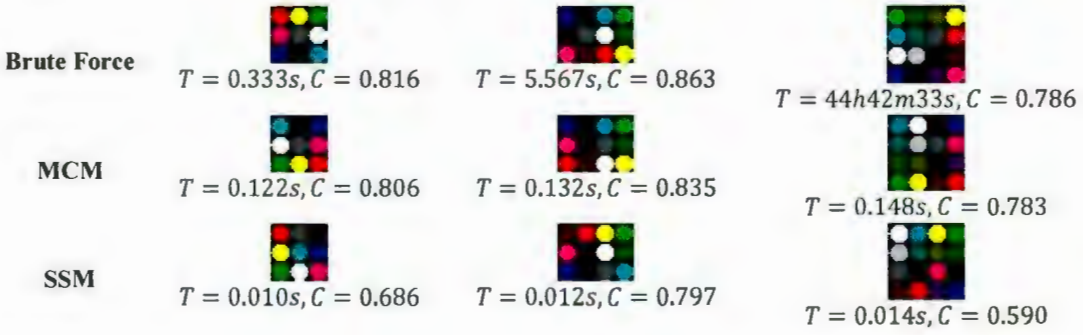


Figure 34: A comparison of the Brute Force, SSM, and MCM algorithms. Times and correlations are given.

5.1.7 Fixed Item Conditions

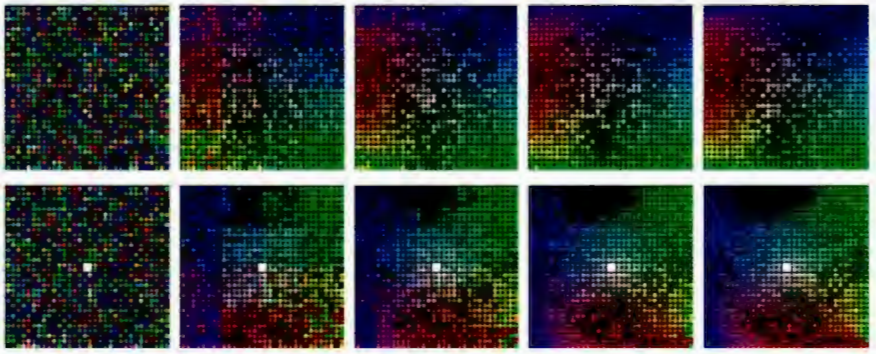


Figure 35: Top row: An organization of Lab color vectors. Bottom row: A fixed item is introduced.

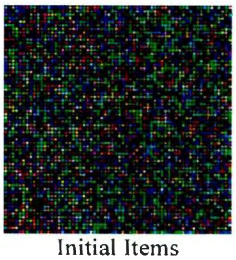
The design of the MCM algorithm makes the implementation of fixed items (or alternatively removed cells from the layout) straightforward. The most significant consideration is that dissimilarities of comparisons with fixed items are weighted in the correlation calculation by a multiplicative factor. The magnitude of the factor changes the strength of the fixed item's influence and is set at the user's discretion. This prioritizes the relationships of the fixed items with other non-fixed ones. Fixed items can also

occupy a larger region than standard items without modification to the fundamental design. Figure 35 shows an example of how a relatively large fixed item can affect the organization of colors.

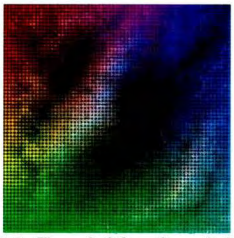
5.2 Results

For consistent comparison, the MCM is evaluated here with the same test data as the SOM in Section 3.2 and SSM in Section 4.3.

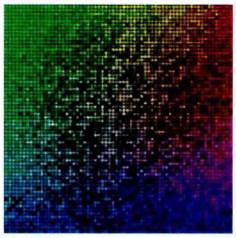
5.2.1 Colors



Initial Items



SSM: $T = 0.82s, C = 0.831$



MCM: $T = 31m, C = 0.816$

Figure 36: The organizations of 4096 Lab color vectors. The first image in the row is the starting set of items in both cases. For comparison, the SSM result is shown. Total time and final correlation between position and dissimilarity across the grid aligned maps is given below the respective images. All implementations are single threaded Java ones running on a Intel Core 2 Duo P8600 CPU at 2.4 GHz. Note that with this number of items the MCM is not as good of a choice to do the organization as the SSM.

In Figure 36 the results of organizing a set of 4096 Lab color vectors into 64×64 cells using the MCM is shown. This is the same set of colors used in Section 4.3.1 with the SSM. For this example the maximum number of iterations per level of the block hierarchy was set to 10 to keep the processing time on this dataset down. Due to the low number of iterations used by the MCM, the SSM generates a result of a higher correlation score in the shortest time. The global layout of the MCM is actually better than the SSM

due to its overall color wheel pattern because of the global correlation optimization being done at the coarse block level. Conversely, at a fine level local noise is not smoothed because small swaps do not have enough positive effect on the global correlation.

5.2.2 Images

The image test deals with a far smaller number of items and introduces a concept dissimilarity matrix, described in Section 3.2.2, which needed to be converted to vectors for both the SOM and SSM. When an MCM is used to organize the same set of images as the SOM and SSM, as seen in Figure 37, the result is slightly better and practically the time is not a factor with this number of items. Visually it appears to be of the same quality, although the correlation is improved slightly, as it should with the MCM, the speed of the organization is slower, as is also expected. The major benefit in this case is simplicity because the MCM achieves the best layout of the three algorithms being compared using the concept dissimilarity matrix directly instead of needing MDS generated concept vectors to get a good layout like the others. Although it has the potential to do the same thing, this is where the SSM with centroid representatives falters since centroid items fail to represent intermediate targets for the polarized concepts used here for the reason given in Section 4.4. The MCM does not exhibit the same concept fragmentation using the dissimilarity matrix as the SSM with this type of data. The reason for this is elaborated on in Section 5.3 below.

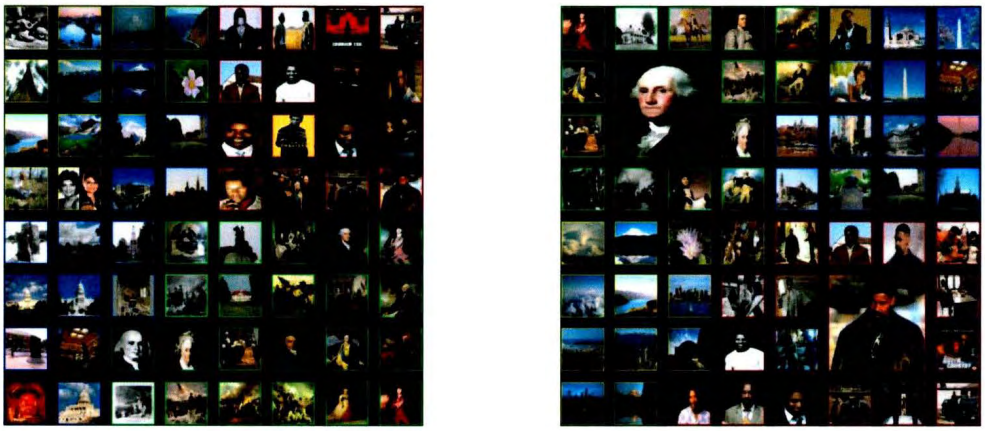


Figure 37: The MCM layout of a collection of images expanded out of the query "Washington". The images have color-coded borders relative to their category. The regions displaying "Denzel Washington" (red), "Washington State" (yellow), "Washington DC" (blue), and "George Washington" (green) are visible. Within the regions, visually similar images also neighbor one another. Left is the standard arrangement while the right shows one with enlarged fixed items. The collage is fitted to the fixed items.

Section 5.1.7 described the introduction of fixed item conditions in the context of the MCM. Figure 37 shows an example of this in action with two double sized fixed items. The result is interesting and shows the flexibility of the MCM in dealing with "obstacles" in the layout.

5.3 Discussion

From looking at the results, the MCM bests the other methods except in terms of time. This is because the calculation of the correlation coefficient, which is integral to the swapping process, also slows it down, even if partial updates are used. There is also a loss of parallelism due to the partner selection regions overlapping. Even in light of the losses, the quality improvement for small maps where performance is not an issue shows worthwhile gains. Beyond that, the algorithm is significantly simpler than the SSM since the need for the use of block representatives is eliminated as well as the need for shifted

block groupings during the alignment stage. In essence, the MCM only consists of a single straightforward alignment stage that uses the correlation score as the guide for swapping items.

In Section 4.4, the issue of bad organizations resulting from an SSM using centroid representatives with nominal data was illustrated in Figure 30. The SSM works by moving items around at a high level and then in smaller and smaller regions until eventually the regions are one item. This hierarchical approach generates high quality results very quickly, but the problem with polarized data like binary vectors is that centroids cannot capture the important facets of the data at the top level. Due to this, the global layout, which is a critical building block of the organizational process, suffers. The subsequent levels of the organizational hierarchy do not recover because the SSM approach is primarily a local sorting one that uses large blocks to do its global sorting early on. The MCM rectifies this issue by eliminating the need for representative targets altogether and organizing things globally.

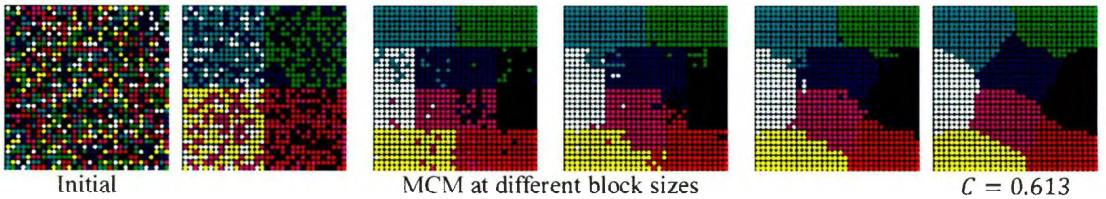


Figure 38: MCM running on binary RGB colors. This is the same set of data the SSM had trouble with in Figure 30. In the case of the MCM, there is a logical progression to a result that is of higher correlation and visual quality than the SSM even with means. The MCM code did require significantly more time at 1 minute.

Figure 38 shows the stages of the MCM arranging the same dataset. Its arrangement is more correlated and more symmetrical. Also, the mixed segmentation after the first

split when there are 4 large blocks is logical given the relationships between the primary and secondary colors. The tradeoff is speed. The correlation updates, even partial updates, take time. This forces the single-threaded Java implementation of the MCM tested to take 1 minute to complete this organization of 32×32 vectors, as opposed to the original SSM which takes less than a second to generate its version.

Overall, when it comes to quality structured layouts the MCM is generally the first choice of the methods presented, especially if constraints like fixed items and removed cells are to be imposed. A major issue is that the MCM is only worth waiting for when it is organizing relatively quantities of items, otherwise, for time critical applications the SSM should be considered as the alternative. A final drawback with the MCM's coarse-to-fine grained greedy approach is shown in Figure 36 where the global arrangement is done well (color wheel) while the local arrangement is noisy. This makes sense given the design of the algorithm since a greedy search is used. At a coarse level, the choice is small and it is possible to find a good solution greedily. At a finer level there are too many possible solutions so the greedy search becomes trapped in a local minimum. Local minima are more evident in balanced datasets like random colors then many real world datasets that contain more distinct items like images.

Chapter 6 Applications

This section shows the results of the organizations of some varying data types. Included types are artificial datasets, images and textures with varying feature vectors, Wikipedia articles, and cities. The figures and discussion showcase the features of structured layout style data organization under more practical circumstances than the results sections of the previous chapters. All figures were generated using single threaded Java implementations running on an Intel Core 2 Duo P8600 CPU at 2.4 GHz.

6.1 Artificial Data

The datasets presented in Figure 39 are commonly used to evaluate dimension reduction techniques because their underlying topological structures are known [60]. The swissroll is a 2D manifold embedded in 3D space. The broken swiss is similar but exhibits discontinuities. The helix and twinpeaks datasets are non-linear. They are all rendered in 3D in the figure to help realize the underlying structures in them.

Figure 39 presents the results of MDS, SOM, SSM, and MCM run on these datasets. Multiple outcomes are demonstrated in the figure. The most notable one is that none of the techniques are able to organize the swissroll and broken swiss based on the underlying topological structure when the Euclidean distance is used. An overlap between the two layers of the 2D manifold manifests in the form of peppering (MDS and MCM) and tangled regions (SOM and SSM), which is due the small Euclidean distance between the two layers between overlapping layers.

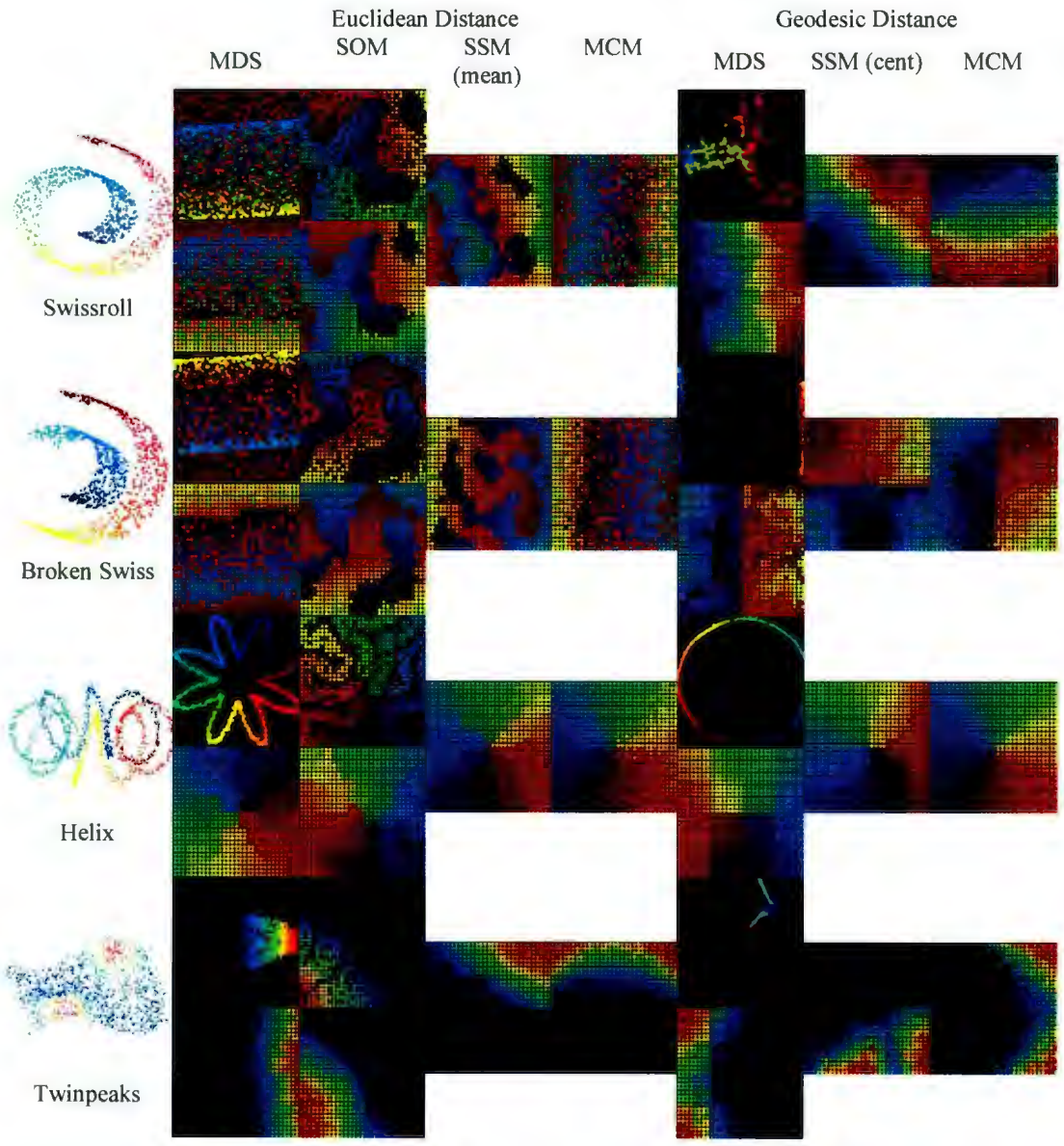


Figure 39: Results of different approaches on four artificial datasets, each of which contains 1024 color-coded 3D vectors. The original datasets are shown in the left column, followed by organizations generated using existing dimension reduction methods and the SSM under two different distance measures. For the existing approaches, both the layouts before (top) and after (bottom) applying k-d tree alignment are shown.

One way to address the overlapping issue is to organize the data using geodesic distance instead of Euclidean distance. The geodesic distance between two data items is

calculated along the shortest path between the two items that passes through other data items. Hence, it measures the distance between data items along the 2D manifold and forces the rolls to unwind. The geodesic distances must be represented using a distance matrix so the SOM method cannot be applied due to the nature of the weight vector updates it requires. The MCM, MDS and the SSM with centroid representatives can work directly on the distance matrix and generate layouts that respect the underlying topological structures. For the non-linear helix and twinpeaks datasets all techniques generate logical results. The MDS results show blocky artifacts due to the k-d tree post-processing used to adapt their raw organizations to the structured layout of the grid. The geodesic MCM result is the smoothest of the helix ones with only a small number of isolated items. The MCM is the only technique that separates the twinpeaks under both distance measures. From the figure we can see that changing the distance measure can have a significant effect on the outcome.

6.2 Images

In Section 2.2, previous research is discussed on how organizing images based on visual similarities can improve the users' browsing experience [40, 54]. Although a variety of techniques have been proposed for organizing images into complex structures, such as clusters or networks [7], popular image search engines still present image search results in a grid layout due to its simplicity.

Figure 40 illustrates how the SSM can rearrange image search results based on visual similarities while still maintaining the conventional 2D grid presentation. Here the input

is 64 images retrieved from Google Image Search using the query "Eiffel Tower". To extract visual information, the RGB color space is quantized into 64 bins to compute a 64-dimensional histogram for each image [52].

Figure 40: An SSM organized set of 64 Eiffel Tower images using color histogram vectors. Not only are dark and light images placed together but ones with other properties like fireworks are also clustered. Duplicated images (marked in green and red) retrieved from different sources are also placed adjacent to each other as well.

Besides organizing image search results obtained online, the SSM can also rearrange existing image collections. Figure 41 shows the results obtained for the Brodatz texture collection[1].

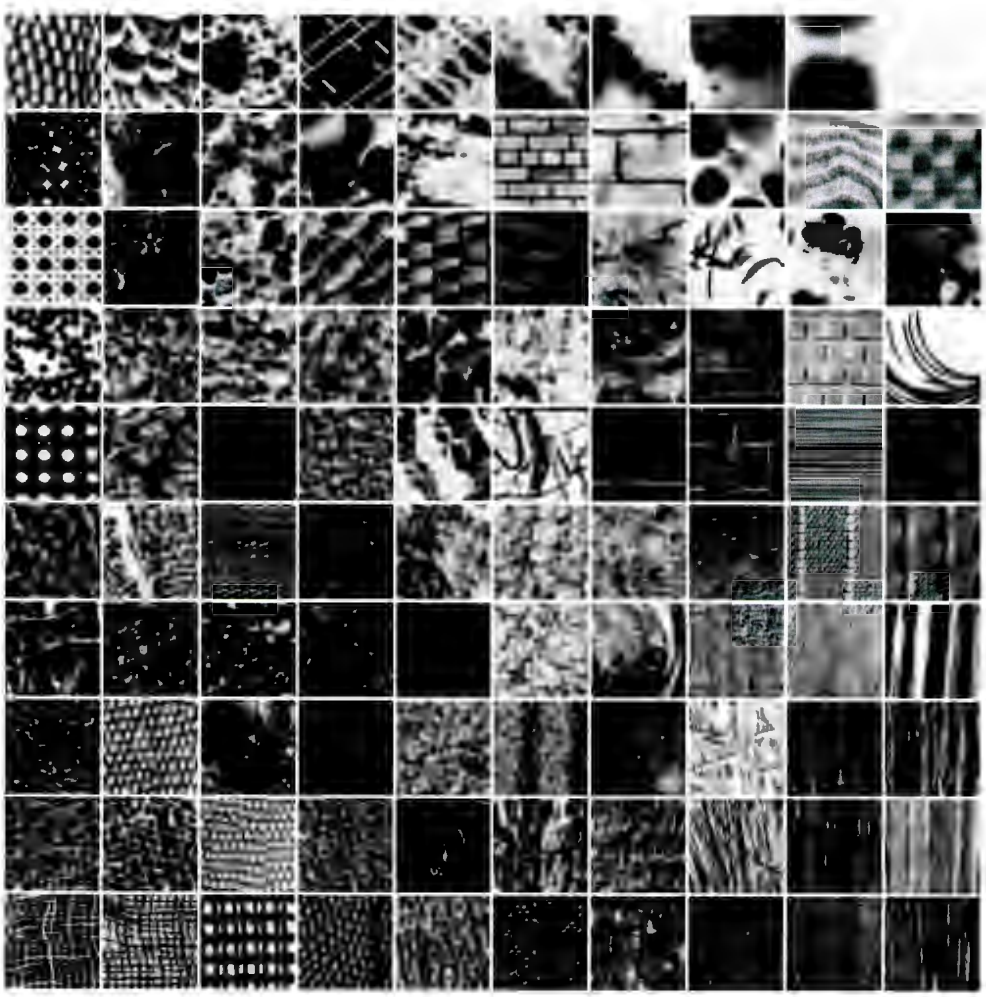


Figure 41: An SSM organized set of 100 textures using gist vectors. Notice how regions of structurally similarity (i.e., round structures, cross hatching, or vertical lines) form.

[1] Downloaded from http://www.ux.uis.no/~tranden/brodatz.html

For the texture collection a 320-dimensional gist [24] feature vector is computed for each image [68]. Gist vectors are low dimensional representations of scenes and are created by measuring edge responses from a series of convolutions on images using a filter at different orientations. The distances between gist vectors can be used to measure perceptual similarity [17]. Hence, the SSM result based on gist vectors effectively separates texture images based on overall edge and structure patterns.

The inputs to the SSM in the two image experiments are high-dimensional vectors. These vectors can be combined in the same way the hybrid concept and content vectors were created in Section 3.2.2 to generate a feature descriptor that carries both color and structure information. The dissimilarity between any two images is then measured by computing a weighted sum of the normalized distances between their gist vectors and histograms. Figure 42 presents a set of 1024 images obtained from Flickr using a variety of queries. There are trees, roadways, waterfalls, pyramids, lightning, and so on. The images are varied and have distinct color and structural signatures. The SSM separates these images based on their overall gist scene feel and color. Even though from a high level there are evident clusters resulting from the color signatures of the histograms, since gist is given 75% of the weight in this arrangement the edge and structure based description of the scenes is most influential. Figure 42 gives closer views of regions to illustrate what the SSM does with these types of vectors. The first cutout shows images with high frequency vertical line scenes like forests and waterfalls. The second shows images with smooth features like desert dunes and pedals of flowers. The organization that the SSM is generating exhibits the mixed properties of the hybrid vectors being used.

Figure 42: An SSM organized set of 1024 Flickr photos using a combination of gist and histogram vectors. Regions of structurally similarity form because of gist, like the noisy forests and rivers, and the smooth beaches and flowers marked and shown. Gist is favored in the result with 75% of the weight, but the color histogram's influence helps maintain smooth transitions between regions across the map. The total organizational time was 11 seconds at 50 iterations per block size.

6.3 Articles

The next experiment uses an SSM to organize the abstract non-vector data that is the set of Wikipedia articles retrieved using the query "Washington". Due to the ambiguity of the query, a diverse set of articles is retrieved, ranging from persons, to movies, to places. The semantic relatedness between any two articles is computed using Wiki Miner [34]. As shown in Table 2, it is difficult to grasp relatedness information from the Wiki Miner similarity scores directly. Hence, an effective organization method is desirable in this circumstance. This is a typical nominal dataset where the mean operation cannot be applied without vectorization of the matrix data like in Section 3.2.2. To organize this data, the semantic relatedness values in Table 2 are negated to be used as dissimilarity. Since this is sparse data, centroid selection will not work well (as shown in Section 4.4) and since it is a small amount of data, an MCM (as described in Chapter 5) is used to get a high quality result. The result shows the relation between things corresponding to Denzel Washington and his movies, Washington DC and its landmarks, Washington State and its neighboring states, as well as George Washington and the persons and places related to him. Other data-driven properties emerge from the organization like the Washington DC cluster and the George Washington cluster being placed adjacent to one another because of their relatively high relatedness and the article Mount Vernon being placed at the center because overall it is the most related to every other article based on the matrix in Table 2.

Table 2: The semantic relatedness among different Wikipedia articles related to the query "Washington". Cells in the table are shaded based on the values for better visualization. The shading helps to identify the cluster of articles related to Denzel Washington and his movies, but the relations among the rest are unclear since they are somewhat interconnected.

	TD	TGD	CF	FQC	DW	OT	IM	TBC	RC	GH	MV	MW	BF	GW	JM	CWP	AR	VF	FB	WC	USC	WM	WH	VC	GU	GWU	HU	WDC	Sea	MR	Ore	Mon	CR	ONP	CRi	WS
Training Day	1	0.71	0.7	0.56	0.52	0.54	0.49	0.52	0.48	0.44	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.32	0.1	0.09	0	0	0	0	0	0	0
The Great Debaters	0.71	1	0.62	0.65	0.75	0.74	0.71	0.68	0.52	0.47	0.52	0	0	0	0	0	0	0	0	0	0	0	0	0	0.31	0.22	0.45	0.19	0	0	0	0	0	0	0	0
Cry Freedom	0.7	0.62	1	0.64	0.58	0.54	0.59	0	0.5	0.62	0.5	0	0	0	0	0	0	0	0	0	0.38	0	0	0	0	0	0	0.15	0	0	0	0	0	0	0	0
For Queen and Country	0.56	0.65	0.64	1	0.67	0.73	0.63	0	0.48	0	0.54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Denzel Washington	0.52	0.65	0.58	0.67	1	0.87	0.87	0.77	0.65	0	0.51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Out of Time	0.54	0.74	0.54	0.73	0.87	1	0.77	0.67	0.48	0	0.54	0	0.38	0	0	0	0	0	0	0	0	0	0.29	0	0.35	0	0	0.22	0	0	0	0	0	0	0	0
Inside Man	0.49	0.71	0.59	0.63	0.77	0.77	1	0.69	0.44	0.49	0	0	0	0	0	0	0.23	0	0	0	0	0	0.21	0	0	0	0	0	0.13	0	0	0	0	0	0	0
The Bone Collector	0.52	0.68	0	0	0.71	0.67	0.75	1	0.45	0.46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Russell Crowe	0.57	0.52	0.5	0.48	0.55	0.48	0.58	0.45	1	0.74	0	0	0.17	0	0	0	0	0	0	0	0.22	0.35	0.06	0	0.13	0.02	0	0.21	0.09	0	0.11	0.07	0	0	0	0
Gene Hackman	0.54	0.47	0.62	0	0	0	0.44	0.46	0.74	1	0	0	0.26	0	0	0	0	0	0	0	0.31	0.36	0.32	0	0.27	0.11	0	0.22	0.06	0	0.15	0	0	0	0	0
Mount Vernon	0.54	0.52	0.5	0.64	0.51	0.54	0.49	0	0	0	1	0.78	0.51	0.6	0.59	0	0.43	0.63	0.7	0	0.66	0.58	0.45	0.46	0.34	0	0.46	0.3	0.48	0.58	0.05	0.47	0.53	0	0.42	0.48
Martha Washington	0	0	0	0	0	0	0	0	0	0	0.78	1	0.54	0.68	0.61	0.53	0.46	0.64	0.57	0	0.54	0.44	0.43	0	0	0	0	0.27	0.08	0	0.16	0	0.31	0	0	0
Benjamin Franklin	0	0	0	0	0	0.38	0	0	0.17	0.26	0.51	0.54	1	0.57	0.55	0.48	0.42	0.56	0.37	0.41	0.47	0.29	0.31	0	0.28	0.25	0.18	0.38	0.14	0	0.15	0.22	0.09	0	0.29	0
George Washington	0	0	0	0	0	0	0	0	0	0	0.6	0.68	0.57	1	0.54	0.74	0.49	0.48	0	0	0.65	0	0.29	0	0	0.15	0	0.33	0.1	0	0.19	0.19	0.33	0	0	0
James Madison	0	0	0	0	0	0	0	0	0	0	0.59	0.61	0.65	0.54	1	0.46	0.68	0.54	0	0	0.6	0.49	0.48	0.51	0.51	0.36	0.54	0.5	0.11	0	0.57	0.8	0	0	0.34	0
Charles Willson Peale	0	0	0	0	0	0	0	0	0	0	0	0.53	0.48	0.74	0.46	1	0.45	0.8	0	0	0.35	0	0.28	0	0	0.18	0	0.14	0	0	0	0	0	0	0	0
American Revolution	0	0	0	0	0	0	0.23	0	0	0	0.43	0.46	0.62	0.49	0.58	0.45	1	0.54	0.39	0	0.41	0.32	0.35	0.19	0.26	0.24	0.24	0.44	0.05	0.11	0.18	0.19	0.07	0	0.15	0
Valley Forge	0	0	0	0	0	0	0	0	0	0	0.63	0.68	0.66	0.48	0.54	0.8	0.54	1	0.54	0.57	0.49	0.39	0.34	0	0	0	0.28	0.35	0	0.48	0.06	0.21	0	0.55	0	0.24
Fort Le Boeuf	0	0	0	0	0	0	0	0	0	0	0.7	0.67	0.37	0	0	0	0.39	0.54	1	0	0	0.52	0	0	0	0	0	0	0	0	0.22	0	0	0	0	0
Washington Circle	0	0	0	0	0	0	0	0	0	0.41	0	0	0	0	0	0	0	0.57	0	1	0.56	0.6	0.44	0	0.44	0.3	0	0.32	0	0	0	0	0	0.6	0	0
United States Capitol	0	0	0.36	0	0	0	0	0	0.22	0.31	0.68	0.64	0.47	0.65	0.6	0.35	0.41	0.49	0	0.58	1	0.69	0.67	0.5	0.44	0.17	0.41	0.52	0.27	0.41	0.31	0.36	0.15	0.31	0.25	0.3
Washington Monument	0	0	0	0	0	0	0	0	0.35	0.36	0.59	0.44	0.29	0	0.49	0	0.32	0.53	0.52	0.6	0.76	1	0.51	0.58	0.33	0.15	0.34	0.41	0.28	0.52	0.26	0.31	0	0.51	0.31	0.34
White House	0	0	0	0	0	0.29	0.21	0	0.06	0.32	0.45	0.43	0.31	0.29	0.48	0.28	0.35	0.34	0	0.44	0.61	0.51	1	0.33	0.45	0.14	0.41	0.58	0.27	0.17	0.24	0.25	0	0.17	0.11	0.25
Verizon Center	0	0	0	0	0	0	0	0	0	0	0.46	0	0	0	0.51	0	0.19	0	0	0	0.5	0.68	0.33	1	0.54	0	0.46	0.21	0.6	0	0.24	0.48	0	0	0	0.42
Georgetown University	0	0.31	0	0	0	0.35	0	0	0.13	0.27	0.34	0	0.28	0	0.51	0	0.26	0	0	0	0.44	0.44	0.33	0.45	0.54	1	0.34	0.52	0.36	0.42	0.46	0.45	0.47	0	0	0	0.14
George Washington U	0	0.22	0	0	0	0	0	0	0.02	0.11	0	0	0.25	0.15	0.17	0.18	0.24	0	0	0	0.3	0.17	0.15	0.14	0	0.34	1	0.24	0.21	0.14	0.14	0.07	0.1	0.1	0	0	0.19
Howard University	0.32	0.45	0	0	0	0	0	0	0	0	0.68	0	0.18	0	0.64	0	0.24	0.28	0	0	0.41	0.34	0.41	0.46	0.52	0.24	1	0.37	0.12	0.4	0.43	0.51	0	0	0	0.1
Washington DC	0.1	0.19	0.15	0	0	0	0	0	0.21	0.22	0.3	0.27	0.38	0.33	0.5	0.14	0.44	0.35	0	0.32	0.52	0.41	0.55	0.21	0.36	0.21	0.37	1	0.27	0.24	0.43	0.41	0.31	0.23	0.35	0.22
Seattle	0.09	0	0	0	0	0.22	0.13	0	0.09	0.06	0.45	0.08	0.14	0.1	0.11	0	0.05	0	0	0	0.27	0.28	0.27	0.6	0.42	0.32	0.12	0.27	1	0.47	0.45	0.41	0.46	0.34	0.42	0.51
Mount Rainier	0	0	0	0	0	0	0	0	0	0	0.52	0	0	0	0	0	0.11	0.46	0	0	0.41	0.52	0.17	0	0.46	0.14	0.4	0.24	0.47	1	0.43	0.39	0.7	0.53	0.56	0.61
Oregon	0	0	0	0	0	0	0	0	0.11	0.15	0.05	0.16	0.15	0.19	0.67	0	0.18	0.06	0.22	0	0.31	0.26	0.24	0.24	0.45	0.11	0.43	0.43	0.45	0.43	1	0.8	0.62	0.38	0.42	0.42
Montana	0	0	0	0	0	0	0	0	0.07	0	0.47	0	0.22	0.19	0.67	0	0.19	0.21	0	0	0.36	0.31	0.25	0.48	0.47	0.1	0.51	0.41	0.41	0.39	0.8	1	0.48	0.43	0.52	0.41
Cascade Range	0	0	0	0	0	0	0	0	0	0	0.51	0.31	0.09	0.33	0	0	0.07	0	0	0	0.15	0	0	0	0	0.1	0	0.31	0.46	0.7	0.52	0.48	1	0.54	0.74	0.58
Olympic National Park	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.65	0	0.6	0.31	0.51	0.17	0	0	0	0	0.23	0.34	0.6	0.38	0.43	0.54	1	0.52	0.38
Columbia River	0	0	0	0	0	0	0	0	0	0	0.42	0	0.29	0	0.34	0	0.15	0	0	0	0.25	0.31	0.11	0	0	0	0	0.35	0.42	0.66	0.52	0.52	0.74	0.52	1	0.65
Washington State	0	0	0	0	0	0	0	0	0	0	0.68	0	0	0	0	0	0	0.24	0	0	0.3	0.34	0.25	0.42	0.14	0.19	0.1	0.22	0.61	0.61	0.42	0.41	0.56	0.38	0.55	1

George Washington U.	Howard University	Verizon Center	Oregon	Olympic National Park	Cascade Range
Washington Circle	Georgetown University	**Washington DC**	Montana	Mount Rainier	**Washington State**
American Revolution	James Madison	United States Capitol	Washington Monument	Seattle	Columbia River
Benjamin Franklin	Valley Forge	White House	Mount Vernon	Russell Crowe	Gene Hackman
George Washington	Martha Washington	Out of Time	The Great Debaters	Inside Man	Training Day
Fort Le Boeuf	Charles Willson Peale	For Queen and Country	Cry Freedom	**Denzel Washington**	The Bone Collector

Figure 43: This is an organization of the "Washington" Wikipedia articles based on their relatedness produced using an SSM. For better visualization, cells are shaded based on the cluster that the article belongs to. Note that the cluster information is not available to the SSM algorithm, yet it was able to group articles in the same cluster together using individual relatedness. Mount Vernon is in the center because it is the article that is most similar to every other (see Table 2). The organization time was 0.7 of a second and the correlation is 0.668.

6.4 Cities

Tests using a dataset consisting of the major cities of the world are shown here. They are represented using a data structure containing both real and nominal values. Each city item A carries three properties: Its latitude A_{lat}, longitude A_{long}, and the name of its nation A_{nation}. The objective is to place cities based on geological proximity, while at the same time encouraging cities from the same nation to be clustered together. To achieve this goal, we define the dissimilarity between two cities A and B to be based on both the great-circle distance between the respective latitudes and longitudes, and their country membership:

$$\begin{aligned}&\delta(A,B)\\&=(A_{nation}==B_{nation}\ ?\ \alpha:0)\\&+(1\\&-\alpha)\left(2r\sin^{-1}\sqrt{\sin^2\frac{B_{lat}-A_{lat}}{2}+\cos A_{lat}\cos B_{lat}\sin^2\frac{B_{long}-A_{long}}{2}}\right)\end{aligned} \tag{29}$$

where the first term represents the country influence, the second computes the great-circle distance between the two cities (r being the earth's radius), and the α varies the weight between the two terms.

Here boundary conditions are used so that the SSM generates something that is not only accurate in terms city versus city dissimilarity, but also that viewers are familiar with, namely the common world map; see Figure 44. The constraints across the northern border of the map vary from 90°N, 180°W to 90°N,0° to 90°N,180°E from the top left to top middle to top right points. The other borders are given suitable constraints based on their relative positions as well. The rest of the layout is then guided by the dissimilarity

between cities. Note that the dissimilarity measure given in Equation (29) is merely a proof of concept. Presumably, encoding different information about the relationships between places (e.g., trade, lending, conflict, or aid) could produce other interesting results describing political landscapes.

Figure 44: The world's largest 512 cities arranged by an SSM. The top shows the country flags of the cities of the entire 32×16 organization followed by a zoomed in view, in which the city names are marked. The total organization time was 3.3 seconds and the correlation is 0.836.

Figure 45 shows how an MCM can be used to arrange items of varying size. In it a grid of world cities is produced where the proximity among the cities is based on their geographical distance and the sizes of the cities indicate their importance. All cities are

organized simultaneously. Alpha and Alpha+ cities, as per Figure 45, occupy four times the area in the grid than the less important ones. For simplicity, a single cell city is not allowed to move into the region occupied by an alpha city. For this reason we enlarge the neighborhood window to 5×5 so that they have a better choice of places to move. Even so, compromises have to be made so that the single cell cities can fit around the alpha city layout. For instance, Oslo and Istanbul are placed close together under Moscow which then forces The Hague further from Amsterdam. Another compromise is Singapore being higher than it should be with respect to Kuala Lumpur and Bangkok, yet with limited space for a large city in that area the layout is sufficient.

Figure 45: A MCM organization for a set of global cities. More important ones occupy four cells in the grid.

Once structured layout of cities is given it is possible to build an interactive pyramid by halving the map repeatedly until one cell remains at the top level of the pyramid. Every time a group of four cells collapses into one, a criterion can be used to choose the city to place in the new cell; highest population for instance. This is a style of hierarchical

visualization is similar to the 3D blob approach described in Section 2.2 and can be applied to other data types by modifying the comparison criterion.

Another use of city organization with a different objective is shown in Figure 46. A table of links to cheap hotels in popular destination cities from the travel website Expedia is taken as input. The cities are arranged by geographical distance only in this case. The output from the SSM is a table that facilitates searching by proximity since destinations that are close to one another and can be found by checking their neighborhoods.

Cheap Hotels in Las Vegas	Cheap Hotels in Ottawa	Cheap Hotels in Honolulu	Cheap Hotels in Chicago
Cheap Hotels in Toronto	Cheap Hotels in London	Cheap Hotels in Paris	Cheap Hotels in Rome
Cheap Hotels in Montreal	Cheap Hotels in Niagara Falls	Cheap Hotels in Quebec	Cheap Hotels in Richmond
Cheap Hotels in New York	Cheap Hotels in Victoria	Cheap Hotels in Orlando	Cheap Hotels in Seattle
Cheap Hotels in Vancouver	Cheap Hotels in Calgary	Cheap Hotels in Los Angeles	Cheap Hotels in San Diego
Cheap Hotels in Edmonton	Cheap Hotels in Anaheim	Cheap Hotels in Banff	Cheap Hotels in Halifax

(a) Layout used by Expedia (extracted from www.expedia.ca/hotels)

Cheap Hotels in Rome	Cheap Hotels in London	Cheap Hotels in Quebec	Cheap Hotels in Ottawa
Cheap Hotels in Paris	Cheap Hotels in Halifax	Cheap Hotels in Montreal	Cheap Hotels in Toronto
Cheap Hotels in Banff	Cheap Hotels in Edmonton	Cheap Hotels in Calgary	Cheap Hotels in Niagara Falls
Cheap Hotels in Vancouver	Cheap Hotels in Richmond	Cheap Hotels in Victoria	Cheap Hotels in New York
Cheap Hotels in Los Angeles	Cheap Hotels in Seattle	Cheap Hotels in Las Vegas	Cheap Hotels in Chicago
Cheap Hotels in Honolulu	Cheap Hotels in San Diego	Cheap Hotels in Anaheim	Cheap Hotels in Orlando

(b) SSM layout based on distances

Figure 46: Different organizations for a small set of cities. Even for such a small set, finding destinations in the vicinity of a given city, say Ottawa, from the Expedia layout can be difficult. The SSM layout allows users to limit their search within a smaller neighborhood. Note that no boundary conditions were used in this case.

Figure 46 and Figure 1, which presents weather data attached to cities, are generated in the same way by a SSM. Note that it is the cities and their metadata of location and country membership that drive the organizations, while it is the data attached to the subsequent results that change the use case of the visualizations. In the first case it is showing links and in the second examining weather.

Chapter 7 Conclusion

Dimension reduction is a well-studied field. The problem of reducing high dimensional data to 2 or 3 salient dimensions has been approached from countless mathematical angles. Some approaches require linear input data (e.g., MDS), and some deal with non-linear (e.g., SOM, Sammon, LLE). Some approaches output to continuous space (e.g., MDS, Sammon, LLE), while others output to an unconstrained grid (e.g., SOM). This thesis diverges from past approaches in that it considers the problem of data organization from a sorting perspective with structured layouts for easy, balanced, occlusion-free visualization as the end goal. Data can have many facets; it can be numeric, multi-dimensional, linear, non-linear, non-numeric, or a combination of one or more of these. The only stipulation is that a measure of pairwise dissimilarity be defined over the set of input. The layout can vary in shape, dimension, and constraint; it might be square, rectangular, circular, or tree-like; it might be 1D, 2D, 3D, or more; it can have boundary conditions or arbitrarily removed or fixed positions. The only stipulation is that there are enough locations for each data item to uniquely occupy one of them. The reason for studying this alternative approach to data dimensionality reduction and subsequent organization is that structured arrangements of this type are suited for direct renderings and intuitive presentation.

Three novel algorithms have been presented throughout the body of this work. All three organize data on the principle that proximity should reflect similarity. With this principle in mind, results from the algorithms are evaluated using the Pearson correlation

coefficient where the more that distance and dissimilarity positively correlate across all pairings of items, the higher the quality of the result overall. The first algorithm is an adaptation of existing techniques. It uses a Self-Organizing Map (SOM) [29] to first arrange data in a semi-structured way and then a k-d tree [3] to fit the result to a fully structured layout. The second algorithm draws inspiration from the first, as well as other dimension reduction, sorting, and clustering techniques, but it is designed around the specific requirements of organizing arbitrary data into structured layouts. It is called the Self-Sorting Map (SSM) [51] due to its pseudo-sorting of data in the layouts. The third algorithm is a significant alteration of the SSM. It shares the same input and output characteristics of the SSM but differs from the SSM in that does not primarily focusing on local adjustments to the layout from coarse to fine levels. It considers movements globally by optimizing directly on the correlation coefficient that is used to evaluate the results of all of the techniques presented. It is called the Max Correlation Map (MCM).

On the whole, the SOM technique performs well with vector data but is limited to that. Since the SOM has no constraint on how many items can be placed in a location, the results have to be post-processed in a structured way by a k-d tree. Overall the results tend to exhibit the balancing property of the SOM, which the alternative dimension reduction techniques presented lack, even after structuring. In the end, this method can generate adequate results but lacks the speed and flexibility needed to achieve the goal of this thesis.

The SSM is a novel algorithm for organizing and visualizing data. It consolidates many ideas and features from some established dimension reduction, clustering, and

sorting techniques like the flexible dissimilarity matrix input of MDS, the balanced and structured output of the SOM, the speed and parallelism of shell sort, and the target converging nature of k-means. It blends the ideas of these and yet it has an alternative objective. Instead of solving the continuous optimizing problem, as many other dimension reduction approaches do, the SSM transforms it into a discrete labeling problem. As a result, it can organize a set of data into a structured layout guaranteeing no overlap. The SSM is more flexible than the SOM-based approach in terms of input. It can organize numeric data, like its counterpart, or non-numeric data using a provided dissimilarity matrix. The key improvement of the SSM over the SOM-based approach with respect to the goal here is that it avoids organizing data indirectly through weight vectors like the SOM, instead it works on the actual data directly. This makes it possible to organize non-numeric nominal data without computing a contrived numeric average between items. The SSM is fully parallel by design and as such can run efficiently on high-performance hardware. It is the most scalable structured layout generator of those presented. As shown, using a GPU tens of thousands of data items can be organized in a fraction of a second, and over a million in close to six seconds. These are excellent speeds for integration into interactive search and visualization applications.

The final algorithm, the MCM, improves upon the SSM in terms of quality. The reason for the quality boost is that it actively seeks to maximize the correlation score holistically whereas the SSM bases decisions on minimizing local dissimilarity. The MCM algorithm is considerably simpler in implementation and more robust in execution than the SSM but it is also considerably slower. At this stage it is only useful for

relatively small numbers of items. The overhead of updating the correlation is too computationally expensive, even when using the more efficient incremental method derived and presented.

To prove the usefulness of structured data layouts, and particularly these techniques for generating them, experiments on different types of data show that these algorithms can be successfully applied to a variety of applications. The applications range from rearranging image search results based on visual similarities, to visualizing semantic relatedness between articles, to the generation of alternative yet intuitive world maps. Limitations of certain techniques are uncovered throughout the tests, and solutions are posed for them.

Overall, the results show how these novel algorithms can produce high quality organizations that strive for topology preservation of input data within the confines of a given output structure. In particular, the SSM technique is quite scalable to larger datasets than the others due to the efficient parallel implementation that its design lends itself to. The MCM, on the other hand, can be effective at handling awkward polarized datasets, varied item sizes, and modified fixed item layouts that the other techniques have trouble, or the inability, to work with. As a body of work, this thesis makes inroads into the demonstration of the utility the proposed algorithms in the area of structured layout generation from arbitrary data as well as the utility of structured layouts for visualization in general.

References

[1] Algorithmics Group, "MDSJ: Java Library for Multidimensional Scaling (Version 0.2)," 2009.

[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in neural information processing systems*, vol. 14, no., pp. 585-591, 2001.

[3] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, 1975.

[4] I. Borg and P. Groenen, *Modern Multidimensional Scaling: theory and applications*, 2nd ed. New York: Springer-Verlag, 2005.

[5] P. Brodatz, *Textures: a photographic album for artists and designers*, vol. 66: Dover New York, 1966.

[6] C. J. Burges, "Dimension reduction: A guided tour," *Foundations and Trends® in Machine Learning*, vol. 2, no. 4, pp. 275-364, 2009.

[7] C. Chen, G. Gagaudakis, and P. Rosin, "Similarity-based image browsing," *Proc. IFIP International Conference on Intelligent Information Processing*, pp. 206-213, 2000.

[8] T. T. Chen and L. C. Hsieh, "The Visualization of Relatedness," *Proc. International Conference on Information Visualisation*, pp. 415-420, 2008.

[9] R. L. Cilibrasi and P. M. Vitanyi, "The google similarity distance," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 3, pp. 370-383, 2007.

[10] M. Ciura, "Best Increments for the Average Case of Shellsort," *Proc. International Symposium on Fundamentals of Computation Theory*, pp. 106-117, 2001.

[11] O. de Bruijn and R. Spence, "Rapid serial visual presentation: a space-time trade-off in information presentation," *Proc. AVI: Proceedings of the working conference on Advanced visual interfaces*, pp. 189-192, 2000.

[12] G. di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*: Prentice Hall, 1999.

[13] G. M. Draper, Y. Livnat, and R. F. Riesenfeld, "A survey of radial methods for information visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 5, pp. 759-776, 2009.

[14] K. Fukunaga, *Introduction to statistical pattern recognition*: Academic press, 1990.

[15] R. C. Gonzales and R. Woods, *Digital image processing*: Addison-Wesley Publishing Company, 1993.

[16] Gustav, #214, quist, and M. Goldstein, "Towards an Improved Readability on Mobile Devices: Evaluating Adaptive Rapid Serial Visual Presentation," in *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*: Springer-Verlag, 2002, pp. 225-240.

[17] D. Heesch, "A survey of browsing models for content based image retrieval," *Multimedia Tools and Applications*, vol. 40, no. 2, pp. 261-284, 2008.
[18] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 6, no. 1, pp. 24-43, 2000.
[19] G. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, no. 1, pp. 833-840, 2002.
[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504-507, 2006.
[21] E. Hoque, G. Strong, O. Hoeber, and M. Gong, "Conceptual query expansion and visual search results exploration for Web image retrieval," *Proc. Atlantic Web Intelligence Conference*, pp. 73-82, 2011.
[22] S. Hotton and J. Yoshimi, "HiSee," 1.0.0 ed, 2004.
[23] A. K. Jain, *Fundamentals of digital image processing*: Prentice-Hall, Inc., 1989.
[24] Y. Jing and S. Baluja, "VisualRank: Applying PageRank to large-scale image search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1877-1890, 2008.
[25] B. Johnson and B. Shneiderman, "Tree-Maps: a space-filling approach to the visualization of hierarchical information structures," in *Proceedings of the 2nd conference on Visualization '91*. San Diego, California: IEEE Computer Society Press, 1991, pp. 284-291.
[26] I. T. Jolliffe, *Principal component analysis*, vol. 487: Springer-Verlag New York, 1986.
[27] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881-892, 2002.
[28] S. Kaski and J. Peltonen, "Dimensionality Reduction for Data Visualization," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 100-104, 2011.
[29] T. Kohonen, *Self-Organization Maps*: Springer-Verlag, 1995.
[30] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, V. Paatero, and A. Saarela, "Self Organization of a Massive Document Collection," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 574-585, 2000.
[31] S. Lafon and A. B. Lee, "Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 9, pp. 1393-1403, 2006.
[32] S. Lakshmivarahan, S. K. Dhall, and L. L. Miller, "Parallel sorting algorithms," *Advances in Computers*, vol. 23, no., pp. 295-354, 1984.
[33] Microsoft and Schn828, "Photosynth: Kiyomizu-dera "Pure Water Temple" Kyoto Japan," vol. 2009, 2009.
[34] D. Milne and I. H. Witten, "An effective, low-cost measure of semantic relatedness obtained from Wikipedia links," *Proc. AAAI Workshop on Wikipedia and Artificial Intelligence*, pp. 25-30, 2008.

[35] A. Morrison, G. Ross, and M. Chalmers, "Fast multidimensional scaling through sampling, springs and interpolation," *Information Visualization*, vol. 2, no. 1, pp. 68-77, 2003.

[36] M. C. F. d. Oliveira and H. Levkowitz, "From visual data exploration to visual data mining: a survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 378-394, 2003.

[37] F. V. Paulovich, C. T. Silva, and L. G. Nonato, "Two-Phase Mapping for Projecting Massive Data Sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1281-1290, 2010.

[38] R. W. Picard, T. Kabir, and F. Liu, "Real-time recognition with the entire Brodatz texture database," *Proc. Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pp. 638-639, 1993.

[39] F. H. Post, G. M. Nielson, and G.-P. Bonneau, *Data Visualization: The State of the Art*: Springer, 2002.

[40] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood, "Does organisation by similarity assist image browsing?," *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 190-197, 2001.

[41] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323-2326, 2000.

[42] J. W. Sammon, "A Nonlinear Mapping for Data Structure Analysis," *IEEE Transactions on Computing*, vol. 18, no. 5, pp. 401-409, 1969.

[43] R. Sedgewick, "Analysis of Shellsort and related algorithms," in *Algorithms—ESA'96*: Springer, 1996, pp. 1-11.

[44] S. F. Silva and T. Catarci, "Visualization of linear time-oriented data: a survey," *Proc. Web Information Systems Engineering, 2000. Proceedings of the First International Conference on*, pp. 310-319, 2000.

[45] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349-1380, 2000.

[46] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," *Proc. SIGGRAPH*, pp. 835-846, 2006.

[47] R. Spence, "Rapid, serial and visual: a presentation technique with potential," *Information Visualization*, vol. 1, no. 1, pp. 13-19, 2002.

[48] T. C. Sprenger, R. Brunella, and M. H. Gross, "H-BLOB: a hierarchical visual clustering method using implicit surfaces," *Proc. Proceedings of the conference on Visualization'00*, pp. 61-68, 2000.

[49] G. Strong, "Similarity-Based Image Organization and Browsing," in *Computer Science*, vol. M.Sc. St. John's, NL, Canada: Memorial University, 2009.

[50] G. Strong and M. Gong, "Browsing a Large Collection of Community Photos Based on Similarity on GPU," in *Advances in Visual Computing*, vol. 5359, *Lecture Notes In Computer Science*. Las Vegas, NV, USA: Springer Berlin, 2008, pp. 390-399.

[51] G. Strong and M. Gong, "Data organization and visualization using self-sorting map," pp. 199-206, 2011.

[52] G. Strong and M. Gong, "Organizing and Browsing Photos Using Different Feature Vectors and Their Evaluations," *Proc. International Conference on Image and Video Retrieval*, pp. 1-8, 2009.

[53] G. Strong and M. Gong, "Similarity-based image organization and browsing using multi-resolution self-organizing map," *Image and Vision Computing*, vol. 29, no. 11, pp. 774-786, 2011.

[54] G. Strong, O. Hoeber, and M. Gong, "Visual image browsing and exploration (vibe): user evaluations of image search tasks," *Proc. International Conference on Active Media Technology*, pp. 424-435, 2010.

[55] G. Strong, E. Hoque, M. Gong, and O. Hoeber, "Organizing and browsing image search results based on conceptual and visual similarities," *Proc. International Symposium on Visual Computing*, pp. 481-490, 2010.

[56] J. B. Tenenbaum, "Mapping a manifold of perceptual observations," *Advances in neural information processing systems*, no., pp. 682-688, 1998.

[57] A. Tikhonova and K.-L. Ma, "A Scalable Parallel Force-Directed Graph Layout Algorithm," *Proc. Eurographics Parallel Graphics and Visualization Symposium*, pp. 25-32, 2008.

[58] R. S. Torres, C. G. Silva, C. B. Medeiros, and H. V. Rocha, "Visual structures for image browsing," *Proc. Conference on Information and Knowledge Management*, pp. 49-55, 2003.

[59] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *The Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579-2605, 2008.

[60] L. van der Maaten, E. Postma, and H. van den Herik, "Dimensionality reduction: A comparative review," *Technical Report TiCC TR 2009-005*, no., 2009.

[61] J. Venna, *Dimensionality reduction for visual exploration of similarity structures*: Helsinki University of Technology, 2007.

[62] J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski, "Information retrieval perspective to nonlinear dimensionality reduction for data visualization," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 451-490, 2010.

[63] J. Vesanto, "SOM-based data visualization methods," *Intelligent Data Analysis*, vol. 3, no. 2, pp. 111-126, 1999.

[64] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," *Proc. Proceedings of the twenty-first international conference on Machine learning*, pp. 106, 2004.

[65] K. Wittenburg, C. Forlines, T. Lanning, A. Esenther, S. Harada, and T. Miyachi, "Rapid serial visual presentation techniques for consumer digital video devices," in *Proceedings of the 16th annual ACM symposium on User interface software and technology*. Vancouver, Canada: ACM, 2003, pp. 115-124.

[66] J. Zhang, C. Chen, and J. Li, "Visualizing the Intellectual Structure with Paper-Reference Matrices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1153-1160, 2009.

[67] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," *arXiv preprint cs/0212008*, no., 2002.

[68] X. S. Zhou and T. S. Huang, "Relevance feedback in image retrieval: A comprehensive review," *Multimedia Systems*, vol. 8, no. 6, pp. 1432-1882, 2003.

Appendix

Image Query Expansion

The process of obtaining query-expanded image search results warrants elaboration since it is not straightforward.

Images retrieved using a user specified query do not carry metadata with conceptual information. To address this problem, as well as to obtain a set of conceptually diverse images, conceptual query expansion is applied to discover different concepts related to the input query. The concepts are then used to diversify the image search results, as well as for generating conceptual feature vectors for the images found.

Wikipedia is used as the knowledge base for the query expansion process in this case because it is an excellent source of information for the purposes of image search since it includes a large number of diverse articles describing people, places, and things. It is densely structured, with hundreds of millions of links between articles within the knowledge base. Most of the articles also contain various representative images and associated textual captions.

The query expansion mechanism [21] works by finding the article that best matches the user-supplied query Q (referred to as the home article) using Wikipedia's search feature. In the case where query Q is ambiguous and Wikipedia suggests multiple links, the ones with higher commonness values are used as home articles. Here the commonness value of an article is calculated based on how often it is linked to by other

articles. In analyzing Wikipedia, it can be observed that the in-link articles (ones having links to a home article) and out-link articles (ones to which a home article links) often provide meaningful information that is closely related to one of the home articles, and hence the query. As a result of this, these linked articles are located and their titles are extracted as candidates for related concepts.

For some queries, the total number of linked articles found might be very high and some of them may not be closely related to the query. To mitigate this, a filtering step is necessary to ensure the quality of the concepts that are extracted. The filtering is performed using the semantic distance between each linked article and its corresponding home article. The so-called Wikipedia Link Measure (WLM) [34] is used for this purpose, which applies Normalized Google Distance (NGD) [9] to the domain of Wikipedia articles. The NGD between any two articles a and b is calculated using the hyperlink structure of the associated articles to determine how much they share in common. That is:

$$NGD(a,b) = \frac{log(max(|A|,|B|)) - log(|A \cap B|)}{log(|W|) - log\big(min(|A|,|B|)\big)} \tag{30}$$

where A and B are the sets of all articles that link to the article of a and b, respectively, W is the set of all articles on Wikipedia, and operator $|\cdot|$ computes the number of articles in the set. The distance obtained lies in the range $[0,1]$, with a smaller value indicating a higher degree of relatedness.

Once the semantic distance measures for all linked articles are calculated, they are sorted in ascending order. The titles of the top N articles are then selected as related

concepts. Given a query Q and the corresponding related concepts $\{R_k | 1 \leq k \leq N\}$, a set of N sub-queries is generated by combining the query with each related concept. Each sub-query $\langle Q, R_k \rangle$ (where $\langle \cdot, \cdot \rangle$ is a concatenation operator that appends a space between the two arguments) is then used to retrieve a set of images from the Web. To avoid duplicate images returned for different sub-queries, a union operation is performed when combining the result sets. All images retrieved are tagged with the corresponding concept R_k. The final set of results contains a diversified concept tagged collection of images that all stem from the common point of the original query Q.

To get a picture of what occurs during query expansion in an ambiguous case where the original query exhibits many facets, the expansion of the query "Washington" will be outlined. This single word query can commonly be considered as a political figure, state, capital, and/or actor. As a result, when given the query "Washington", a Wikipedia search will return multiple articles related to "Washington", but under each of the different perspectives. Articles having higher commonness scores, such as "George Washington", "Washington State", "Washington DC", and "Denzel Washington" are then selected as home articles. All articles linking to or linked from one of the home articles are considered as candidates for related concepts. The top N candidates with smallest semantic distances are then used for generating sub-queries (e.g., "Washington Monument" and "White House" for the home article "Washington DC"; "Martha Washington" and "Benjamin Franklin" for the home article "George Washington", and so on). In the end a tree of expanded queries is produced that is rooted at "Washington", then branches into "George Washington", "Washington State", "Washington DC", and

"Denzel Washington", and then from those branches into sub-queries related to each. For the purposes of this thesis the expansion is halted at this first level of sub-queries after the initial home article generation.

Acknowledgements Unabridged

The first person that deserves bigger thanks than I could ever give is my supervisor Dr. Minglun Gong. When the Head of the Computer Science Department at the time, Dr. Wolfgang Banzhaf, received an application for a Master's student addressed to the Faculty of Engineering (courtesy of the remarkable administrative mess I had myself in at that time) and called me instead of slipping into the recycling bin, little did I know that my life course was about to altered forever.

Originally, when I had finished my undergraduate degree in Computer Science I had decided that instead of leaving school I would stay there forever. Not in university per say, but I heard that across the road from engineering if you did a year with the education crowd they would give you this add-on degree and I could go teach young up and comers in the high school system about computers until the cows come home. I did that. One year, B.Ed., and I was off to the races...or was I? No, I was not. I was still not ready to leave the hallowed halls of MUN just yet. I had just figured it all out. Every nook and cranny; the broken vending machines, the CS printing scam, the water fountains that taste like a stream, the water fountains that taste like disease, the times to go to the UC, and the times when the first years were there (i.e., when not to go). It felt like the place was running by my watch. Why, I was so comfortable I would even go for walks in the residence tunnels at night...in groups of two...armed to the teeth...but still! So I decided, again, that instead of leaving, I would head back across the road, again; back to my beloved Engineering building. I would apply to do a graduate degree in something in

realm of computers and then go into public school teaching with the added bonus of a plumper paycheck. Boy was I wrong.

I picked up the phone and it was Dr. Banzhaf, much to my surprise. He tells me that even though my M.Sc. application should have ended up in the Computer Engineering Department, as I had planned, it came across his desk for some odd reason. I had expected that to be that and if this was my first test of grad school I had failed it. That was not the case. In fact, Dr. Banzhaf did not say he would just forward my application blindly on to the correct people, instead he offered me a potential position with a new faculty member coming on stream in the next semester. I said yes, half because I was way past the deadlines for entrance into either of the Engineering or Computer Science departments, and half because this just felt right. Dr. Banzhaf went ahead and set up a meeting with this professor that was currently in Alberta. I got a call in the next couple days from a calm voice on the other end of the receiver that kindly introduced himself as Minglun Gong! Gong and Strong, think of the conference paper bylines! It was meant to be! P.S. I had no idea grad students wrote conference papers at that time, nor that they got paid. Yes, I got paid! I was a professional student in every sense of the word. Life was good. I was happy. I was also the minority in all of my classes, in fact since all of the students and professors were Chinese the first semester they probably would have been speaking Mandarin had I not been in the room; so they were a little less happy than me I guess, but life was good nonetheless. A year and a half past, Tao Chen and Zheng Chen got me through my courses (I never forgot you guys!), Dr. Gong and I wrote some papers, we had some laughs, I went to Vegas for a conference on the university dime

("You mean they pay for me to go to this thing?!"), and then when I was about a term and a half out from being all done another critical character in the story of my life came into the picture. Elaine Boone. Basically, think of her as a female version of Gandalf the Grey in the department with a candy dish instead of a big stick. Come to think of it, she probably was the one that often enough was tasked with telling grad students, "You shall not pass!" I am not saying she enjoyed that...but I digress. Elaine contacted me and told me that I should apply to NSERC (the big federal pot of money for research in Canada) to see if I could get funding for a doctoral degree. I told her that I was not really considering a Ph.D. since my plan was to go back to school (high school) and teach teenagers there was more to computers than MSN and Myspace (at that time, yes, MSN and Myspace). She insisted that even if I did not take up the award I should at least apply and not close the door. Honestly I felt obligated to at that point. Who says no to Gandalf? Obviously not the Hobbits or there would not have been so many books, movies, or merchandising deals! Also, Dr. Gong had expressed interest in having my back as a Ph.D. student. So I set about writing up the most captivating research proposal and self-promoting, self-proclaiming drivel I could stomach. I have a weak stomach for that sort of thing so it took me a solid week. I did not really want to hear back from them because in reality, I did not want to have to make a decision on whether to stay to do a Ph.D. or simply become a teacher and badger kids about leaving their monitors on all day after class. I also had no idea what I would do for 3+ years during another graduate degree. So on the application I just put down what I was already doing for my M.Sc. with some sort

of Star Trek Next Generation add-ons. Little did I know that NSERC's favorite starship is the Enterprise…

I got money, NSERC money, and plenty of it. So much in fact that considering I did not know I was going to be paid during my Master's a year and a half before, I did not even know what do with myself now. Should I stay or should I go? I was thinking that at that point I would have done a bit better as a teacher financially. A few days later NSERC trampled over that thought when I received another letter in the mail from them. More money! What were they trying to do to me? Drive my blood pleasure through the roof with stress over this decision? At this point I could not say no. Being a grad student? Working on neat stuff with little to no real life responsibility? How could I not take them up on this?! Thank you Elaine!

Flash forward a couple years. In that time, I had been a student representative on all of the committees and boards directly connected to the department for at least some amount of time. The new crop of graduate students had reinvigorated the social life of the department in a big way. You are all awesome and I would not trade our soccer games for the world. Messi...I mean Wasiq, I will probably feel those vicious cutbacks in my ankles for the rest of my life! And Anastasia, the next time I see you winding up to kick a ball, forget defense, I am diving out of sight. During that time I found friends and collaborators in M.Sc. student Enamul Hoque and Dr. Orland Hoeber (a.k.a. the reference referee, with reluctant thanks I submit that you did make my paper writing better). Hoque, Hoeber, Gong, and Strong had a good run there for an awhile. Probably the most memorable thing of all over this period of time had nothing to do with academics. It was

when the arcade machine I bought to rebuild for the grad student lounge mysteriously went missing one Halloween night. Even though the door was normally locked, the huge, heavy, awkward cabinet was nowhere in sight the next day. With the help of Nolan White (who puts up with my left-field tech requests left, right, and center), Mike Rayment (the faculty's own walking, talking almanac of computer science extraordinaire), Paul Price and Aaron Casey (Nerf warriors!), Marian Wissink, Renesa Nizamee, and Nasir, we successfully tracked down and recovered the cabinet. It was full on CSI style, tracing drag marks on the tiles out into a lab, back out and down the hall, up the elevator four floors, down another hall, and into a stairwell. What happened next is insane and best told in person. Ask me about it sometime if you need a laugh...hey, Alejandro? Also, the undergrads (Robert, Justin, Simon, etc.) of this era deserve a shout out too. You guys appreciated "The Brooklyn Heist" as much or more than any of us.

May 2011: I am the lead volunteer at the GI/AI/CRV conference that happened to be local to St. John's that year. During those few days I met lots of great people I will keep in contact with forever (Malika) and became better friends with ones I knew (Jason Gedge, Hamed). I also met someone that would eventually offer me an opportunity that would rock my world in the months and years to come.

One of the event organizers runs out to me in a panic and tells me that the Head of Research for Google NY is upstairs poised to give a keynote talk and her laptop will absolutely not connect to the wireless network. He says that she must be on the Internet as of two seconds ago. I scamper upstairs and burst through the door to find a short blonde lady in what appears to be running clothes in front of a Macbook scratching her

head. I step forward and state my purpose. She allows me to access her laptop and I try to figure out what our dear friends at Computing & Communications have set up for these guests to access the internet. After fumbling around a little while I got the thing online by using my credentials. The lady thanks me kindly and proceeds to bring up her talk. I turn to leave to continue my duties downstairs, as this was the AI portion of the conference and not my thing, but then decided to stay to see how it was going to go having just met the speaker. She proceeds to talk about all sorts of Googley things and then brings up something that I actually referenced in my Master's thesis! Well, after the talk I waited around to mention this interesting fact and comment on my past research. She was very kind and genuinely interested in seeing how it worked the next day. Over the next couple of days I fixed her Internet again (surprise, surprise from C&C), demoed my old proof of concept image organizer, and we proceeded to talk about fun stuff that had nothing to do with computers whatsoever; which is exactly what any respectable Computer Scientists do at Computer Science conferences isn't it? We parted ways and that was that. But it wasn't...

About a week and a half later I got an email from the lady offering me an internship in New York with Google. I said I was too busy at the moment and that...no I didn't! I said yes of course. I had to do a couple interviews (which in my opinion were not stellar) but given whose name was on my reference at the company I guess that got me through. I eventually got an offer. Thank you Corrina Cortes! You may be short on stature but you are big on heart!

Before leaving for New York City I set wheels in motion to do an exchange at NTNU University in Trondheim, Norway after the internship. By wheels I mean begged for money to do the trip from NSERC with another application about the life-altering, world-changing, dynamic, buzzword-filled vision I had for the work I would accomplish while studying aboard. In reality I wanted to do all of these things, plus my brother had just moved there for a job so that narrowed down my choice of location to the "few places in the world to do work like this", which happened to be Trondheim.

During the fall I spent at Google in NYC I frankly had a ball. I met tons of great people inside the company walls and out. Tomasz Zurkowski (lunchtime burrito fanatic), Yuriy Znovyak (sushi connoisseur), and Nemanja Petrovic (my mentor), if it was not for you guys I would still be debugging memory leaks in C++. My ice cream adventuring partners Andreea Gane and Gillian Chin; nobody worked as hard as us to eat so bad. Special thanks to Stephanie Chan for guiding us interns around the city on adventures courtesy of the company, and especially for introducing us to the street meat cart on 53rd Street. Sketchy food never tasted so good! There were way too many good times to list here so I will just sum it up in one word: scooters.

Near the end of my time at Google I received a mass email from some graduate student conference organizers in the company that were looking for referrals for underrepresented grad students in the academic world that met a certain criteria. I read the criteria, realized I fit it, convinced myself I was underrepresented, and submitted my own name.

A little while after that I received word from NSERC that I would be getting the money for the trip to Norway I had set the wheels moving on before I went to Google. My face hurt from smiling for the next hour or so. I booked the flights for the end of January.

A little while after that I got word back that I was invited to come to the Google Grad Student Forum in Silicon Valley that I had referred my own self to. The point of the thing was to visit the Google HQ and hang out with a bunch of other grad students simply to network. I replied I was too busy and...no I didn't! I obviously accepted the invitation. The grinning continued. It actually turned out that the Google conference slotted perfectly into the time frame of when I would be able to go a couple days before I had to leave for Norway. All was well in the world.

Before leaving Google I did interviews for a full time position, like you would. I left for home two days before Christmas.

Late January I went to San Francisco. I enjoyed the conference, meeting fantastic folks and had a hilarious time sight-seeing around SF after it was over with Omid Mola and Daniel Orozco whom I had just met at the conference. Thanks for having my back around the city guys, especially in that "non-legit" hotel we decided to stay in on the cheap; and thanks for not taking up too much room in the king-sized bed we three were forced to share in said hotel room!

Two days later I was on a plane to Trondheim, Norway. After some harrowing wind shears and what felt like nearly crashing a couple times on the way into Værnes airport I

arrived safe and sound. Let me tell you though, there were shrieks and barf bags were being exercised all over the place on that plane. I just sat and clenched everything. I spent a life-changing four months in Norway. I fell in love with the place and the people...everything except the prices. Driving the crazy roads in my brother's under-powered, over-sized van, hiking up glorious untouched mountains and snowboarding down, and generally hanging around with some of the most active and fun people I will ever meet. Det er fantastisk! That was just the fun stuff. At NTNU I met creative thinker and debugging master Rune Jensen. I am indebted to you for your help with my research! Also for wacky knowledge about alternative space stations and ahead of the curve 27 inch 120Hz monitors. I have to thank Ivar Nikolaisen for allowing me to invade his office for four months. I hope you finally got your dream machine set up and purring nicely! My stomach thanks Gunnar Inge for the introduction to grøt (Norwegian rice porridge)...every Friday. And last but not absolutely, positively, not least, none of the trip would have ever been possible without the response from and subsequent gracious hosting of Anne Elster. You were my point of contact at NTNU the first and the last. Thank you so much for opening up your research group and home to me. I will visit for the best chili ever any time you are making and offering it. Texas must have rubbed off on you a bit in that respect! Do you think the government will give me a visa to come "study" chili?

Before leaving Norway, while I was on an extreme snowshoe excursion in the woods (basically jumping off whatever it was possible to climb up) and I received a call from a Google recruiter Cyndi Tran (another thanks here) confirming that based on the

internship and interviews and everything else, they were offering me a full time position in Mountain View in Silicon Valley. Funny enough that happened to be the same office I had visited during the Google Grad Student Forum I had attended months prior. I said no I was too busy and...no I didn't! Cyndi got a resounding yes. When I hung up I was speechless in thought at the rollercoaster, nay, rocket ship, my life had been on up to that point. It was hard to believe how much I had been in the right place at the right time and how well everything had clicked together perfectly. As my mind wandered I snapped back to the sound of my brother's voice coaxing me across the log I was on, in snowshoes, over a frigid cold and running river. I promptly tripped over a couple of the log's branches, caught myself at the other end, and proceeded to barrel up the mountain through thigh-high snow in my jeans (a little ill-prepared I admit).

I left Norway at the limit of my stay time. After returning home I resumed my duties as a MUN Ph.D. Candidate and set a course to clue up all loose ends of my research and finish writing this thesis. Upon my re-arrival to MUN I noticed that while some old friends had finished and moved on, and new friends had started and moved in. During the past months I have had the pleasure of meeting and greeting all of them. For instance, Dominick Feininger, Germany exchange student and all around bicycling fiend. Thanks for the brake-less fixie by the way! I am not dead yet! Another addition to the crew was Esteban Ricalde, who has assumed the role of CS grad student representative. And there are so many more. You guys and girls are the heart and soul of the department as far as I am concerned. You are some of the most genuine people I have ever met and look out for each other like brothers and sisters. I am better having known you.

So that is it. Maybe the longest acknowledgements section in the history of theses? After all of everything I just went through and the thanks I passed out, I would like to end with this. Regardless of what you believe, there is no doubt in my mind that the favor on my life comes from somewhere, or more specifically someone. He deserves the final and everlasting thanks.

Let not mercy and truth forsake you; Bind them around your neck, Write them on the tablet of your heart, And so find favor and high esteem in the sight of God and man. – Proverbs 3:3-4