The World of Music: User Ratings; Spectral and Spherical Embeddings; Map Projections.

David Gleich* Stanford
Matthew Rasmussen MIT
Kevin Lang Yahoo! Research
Leonid Zhukov Yahoo! Inc.

July 24, 2006

Abstract

In this paper we present an algorithm for layout and visualization of music collections based on similarities between musical artists. The core of the algorithm consists of a non-linear low dimensional embedding of a similarity graph constrained to the surface of a hyper-sphere. This approach effectively uses additional dimensions in the embedding. We derive the algorithm using a simple energy minimization procedure and show the relationships to several well known eigenvector based methods.

We also describe a method for constructing a similarity graph from user ratings, as well as procedures for mapping the layout from the hyper-sphere to a 2d display. We demonstrate our techniques on Yahoo! Music user ratings data and a MusicMatch artist similarity graph.

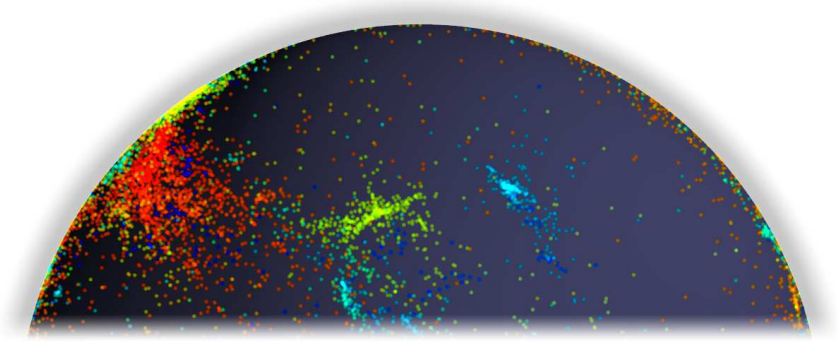


Figure 1: A partial spherical embedding of the MusicMatch similarity graph.

*Email: dgleich@stanford.edu

1 Introduction

Online music services such as Yahoo! Music, MusicMatch, last.fm, and Rhapsody allow service providers to collect an enormous amount of data about the musical tastes of their users. In this paper, we address the question of how to use these datasets to understand and visualize the *world of music* created by users. We do not use any data besides user ratings or metrics derived from user ratings. The goal of this research is purely exploratory. We seek to determine what, if anything, we learn by creating a visual representation of this world.

At a high level, our approach is to view the set of music ratings as a bipartite graph between users and artists. From this graph, we induce a similarity graph between artists, often employing some heuristics to ensure high data quality. To visualize the relationships between music artists, we compute an embedding of the similarity graph and draw the graph as a point cloud along with a subset of largely transparent edges.

This approach is what we used to create figures 1 and 2. In the title picture, we computed a spherical embedding of the MusicMatch dataset introduced below. Each set of similarly colored points represents a set of similar musical artists. The second picture presents a labeled "map" of the LAUNCHcast dataset. Our approach outlined above follows other ideas in dimensionality reduction and data embedding [11, 12, 2]. Although the process combines existing ideas in a straightforward manner, it yields consistent and interesting results on two datasets derived from music. This paper extends the results in [6] by greatly expanding the exposition of the techniques used and demonstrating results for a second dataset.

This paper proceeds as follows. Section 2 briefly describes the two datasets we use. These datasets both come from Yahoo!'s music services. Section 3 details our data processing methodology. In short, we process the data, compute a derived graph, embed the graph onto a sphere (section 4), unroll the spherical data (section 5), and display. The remainder of the sections describe our implementation and results.

2 Data

For our results, we use two datasets. The first dataset is from Yahoo! Music's LAUNCHcast radio service and consists of user ratings. The second dataset is a set of artist similarity scores from Yahoo!'s MusicMatch service.

LAUNCHcast The LAUNCHcast data used in this paper consists of all the ratings made by users on the Yahoo! Music service during a 30 day period. The full dataset contains approximately 250 million ratings on 100,000 artists from 4 million users. The ratings are on a scale from 1 (dislike) to 100 (like). The LAUNCHcast dataset has a power-law distribution in the number of ratings for each musical artist.

MusicMatch In contrast with the LAUNCHcast data, the MusicMatch data only contains artist similarity scores computed with an unknown metric derived from user ratings. The

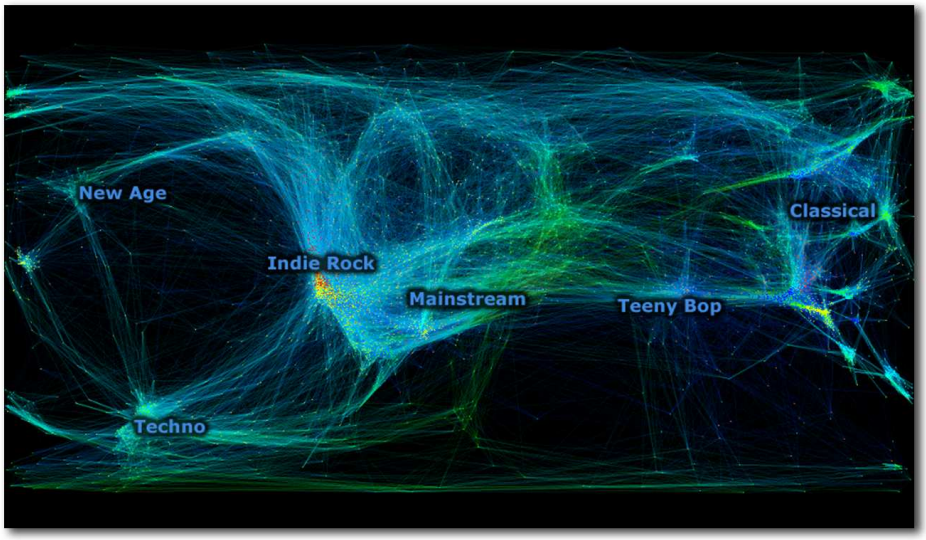


Figure 2: A hand labeled map of the LAUNCHcast data.

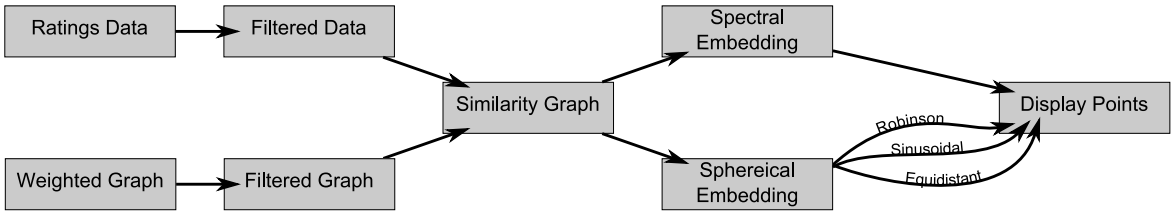


Figure 3: Our data processing pipeline.

scores are between 0.00034 and 50.37 and are not symmetric. Higher scores indicate greater similarity between artists. There are 41,627 artists and 3.4 million similarity scores. The average similarity score is 5.6. For each artist, we have a set of at most 100 similarity scores.

3 Data Pipeline and Filtering

The previous section described the raw datasets. In this section, we describe how we converted those datasets into weighted adjacency matrices for input to the graph embedding algorithms discussed next.

Figure 3 visually displays our data pipeline. There are two entry points, the top left and top right. The LAUNCHcast dataset follows the upper-left pathway and the MusicMatch data is a weighted graph entering on the right.

LAUNCHcast Processing Although we were given approximately 250,000,000 ratings in the LAUNCHcast dataset, our first step was to filter the data. Because our eventual goal was a similarity graph between artists, we wanted to eliminate unnecessary data. Toward

that end, we removed all ratings below the numeric value of 75. The intuition behind this choice was that low ratings are not reliable between users, but that high ratings are reliable; that is, people reliably know when they *love* a musical artist, but make less useful ratings on artists they are less enthusiastic about.

Following the filtering step, we further wanted a set of artists and users that have many ratings. To accomplish this goal, we removed all artists and users with less than 100 ratings. Afterward, the dataset had 9,276 artists, 140,691 users and 25,466,113 ratings.

To convert from the filtered data to a similarity matrix between artists, we used a cosine similarity metric. To compute the cosine distance between two artists, we view each artist as a vector in the space of users and compute the cosine of the angle between the artists. Figure 4 presents a visual depiction of this metric. More rigorously, artist i is the point,

$$a_i = \begin{pmatrix} u_1^i & u_2^i & \dots & u_m^i \end{pmatrix}$$

where u_1^i is the rating user 1 gave to artist i and we have arbitrarily labeled users from 1 to m with $m = 140,691$. With this notation, the cosine of the angle between artists i and j is

$$\cos(\theta_{ij}) = \frac{\sum_{k=1}^m u_k^i u_k^j}{\sqrt{\sum_{k=1}^m \left(u_k^i\right)^2}\sqrt{\sum_{k=1}^m \left(u_k^j\right)^2}}.$$

There are two properties of the cosine metric we use. First, the metric is symmetric, $\cos\theta_{ij} = \cos\theta_{ji}$. Second, assuming that $u_k^i \geq 0$ for all i, k, then $0 \leq \cos\theta_{ij} \leq 1$.

We build a sparsified cosine similarity graph in the following manner. The artists correspond to graph nodes and edges are established by the following procedure: we connect nodes i and j in this graph if j was one of the top N similar artists to i or i was one of the top N similar artists to j, where similarity is defined as $\cos\theta_{ij}$. The weight on the edge is the cosine similarity score. Without the restriction on the top N artists, the resulting graph will be dense with almost all edges. Thus, we term this the sparsified similarity graph.

While cosine is symmetric, the relationship "top N closest using cosine" is not symmetric. The above procedure explicitly symmetrizes the graph using an "or" operation. Thus, an artist may have degree larger than N in this similarity graph. We choose $N = 20$.

The result is a weighted similarity graph between the 9,276 artists. After the sparsification described above, the graph contained 150,292 weighted edges and is connected.

In summary, to process the LAUNCHcast data, we

1. remove *low* ratings (below 75),
2. remove artists and users with insufficient ratings (less than 100), and
3. compute the sparsified cosine similarity matrix.

Note that we did not do any *parameter tuning* on these parameter choices. We selected these numbers before seeing any results.

MusicMatch Processing In contrast with the LAUNCHcast data, for the MusicMatch data, we were already given a set of similarities between artists. In this section, we describe how we converted the provided data into a similarity graph.

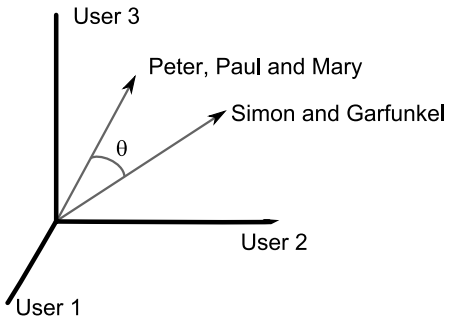


Figure 4: Visual depiction of cosine similarity. We set the similarity between "Peter, Paul, and Mary" and "Simon and Garfunkel" as $\cos(\theta)$, where θ is the angle between the vector describing the artists represented in the space of users.

Recall that for each artist, the data listed as many as 100 similar artists. We interpret this data as a directed weighted graph. As in the LAUNCHcast data we remove all but the top 20 highest weighted links between artists. Following this step, there were 802,826 edges left in the graph. Next, we normalized edge weights to the range $[0, 1]$ by dividing by the largest similarity score. This action was done globally and not for each artist. The intuition behind normalizing the scores to a maximum of 1 is two-fold. First, normalizing to the range $[0, 1]$ gets us "closer" to a cosine-like metric. Second, the numerical algorithms used to embed the resulting graph often exhibit the best precision when the largest floating point number is 1.

The next step is somewhat unintuitive. We remove all weighted edges with weight less than the global mean ranking. After normalization from the previous step, the mean for the MusicMatch data with 802,826 edges was 0.146. This operation reduced the number of edges to 282,907. An intuition for this step follows from the next processing step, which is to remove the direction on each edge. Specifically, we replace each directed edge with an undirected edge weighted with the maximum weight of either directed edge (the weight of a non-existent directed edge is 0). This step yields 240,648 undirected edges.

Returning briefly to the step where we remove weighted edges below the mean, recall that the unknown similarity metric between artists was not symmetric. Because we eventually consider only one of two possible similarity scores, we wish to consider only the strongest data. In some sense, this step is equivalent with dropping all ratings below 75 for the LAUNCHcast data.

The final processing step on the MusicMatch data is to compute the largest strongly connected component of the weighted graph. The largest component has 24,057 artists and 239,984 undirected edges.

To recap, processing the MusicMatch data involved:

1. removing all but the top 20 weighted edges,
2. normalizing weighted edges to the range $[0, 1]$,
3. removing low weighted edges (below the mean score),

Symb.	Meaning
A	the weighted adjacency matrix
δ_{ij}	the Kronecker delta
e	a column vector of ones
E	set of edges
(i, j)	an edge between vertex i and j
L	the Laplacian matrix
n	the number of vertices
U	layout energy
V	set of vertices
x_i	the position of vertex i embedded in $\mathbb{R}^1$
X_{ik}	the kth coordinate of vertex i embedded in $\mathbb{R}^d$
w_{ij}	weight on edge (i, j)

Table 1: The notation used in this paper.

4. symmetrizing each edge by taking the maximum weight of each directed edge, and
5. computing the largest connected component of the resulting graph.

Unlike the LAUNCHcast data, we did do some parameter tuning to determine what cut-off parameter to use in step 3. Other parameters (including removing step 3) yielded subjectively worse results. In actuality, knowledge of the underlying similarity metric or raw rankings would allow us to further tune this processing.

4 Graph Embeddings

Given a weighted undirected graph $G = (V, E, w)$, one way to visualize this graph is to assign coordinates to every node. This process is called *embedding* the graph. We want a set of low-dimensional coordinates that illuminate internal properties of the data. In particular, for our data we would like "similar" artists to be placed "nearby" each other in embedding. This problem is common and many solutions exist [12, 2, 11]. As we will show, our solutions are related to the previous work, but we prefer an alternative derivation to highlight the assumptions implicit in the optimization problem.

To describe our embeddings, we first write both a single and multi-dimensional quadratic energy function and then discuss the problems with this function. The next two sub-sections describe two different "fixes" for these problems. Finally, we provide a section with explicit formulas for the resulting optimization problems in three-dimensions to make our ideas concrete.

This section is greatly expanded in the Appendix, where we discuss motivation for these embeddings. Also, we further elaborate on alternatives to the embeddings which yield poor results.

4.1 One-Dimensional Quadratic Energy

One possibility to embed the graph is a weighted quadratic term over the edges of the graph. Intuitively, this idea gives rise to a quadratic "energy" that "attracts" nodes connected with edges. If we restrict ourselves to a one-dimensional embedding and label all the vertices with integers from 1 to n, we have a coordinate $x_i \in \mathbb{R}^1$ for each vertex $i \in V$. For convenience, we represent the set of all coordinates in a length n vector x. Given an embedding x, then the "energy" of the embedding is

$$\begin{aligned} U(x) &= \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2 = (1/2)\sum_{ij} A_{ij}(x_i - x_j)^2 \\ &= \sum_{ij} L_{ij} x_i x_j, \end{aligned} \tag{1}$$

where A is the weighted symmetric adjacency matrix ($A_{ij} = A_{ji} = w_{ij}$) and L is the Laplacian matrix,

$$L_{ij} = \delta_{ij} \sum_k A_{ik} - A_{ij}.$$

In matrix notation, we have that

$$L = \text{Diag}(Ae) - A \quad \text{and} \quad U(x) = x^T L x,$$

where e is the vector of all ones.

Although the quadratic energy function has significant problems that we will address soon, we first wish to write the multi-dimensional generalization. Let X_{ik} be the kth coordinate of the embedding of vertex i into $\mathbb{R}^d$ and let X represent the $n \times d$ matrix of all the coordinates. The quadratic energy of embedding X is

$$\begin{aligned} U(X) &= \sum_{(i,j)\in E} \sum_{k=1}^{d} (X_{ik} - X_{jk})^2 = \sum_{ij} \sum_{k=1}^{d} L_{ij} X_{ik} X_{jk} \\ &= \text{trace}(X^T L X). \end{aligned} \tag{2}$$

To discuss the problems with these embeddings, we return to one-dimensional embeddings for simplicity. The problems we describe remain for the multi-dimensional function as well.

Ideally, we would like to embed a graph by minimizing $U(x)$, that is, embed the graph with x^* where

$$x^* = \text{argmin}_x \, U(x).$$

However, this optimization problem has a simple minimizer, $x_i^* = 0$. Needless to say, the embedding of a graph to a single point is not good.

The function U has two properties that cause this behavior. First, the minimizer of U is not unique. If x is a minimizer of $U(x)$, then for any scalar γ, $x + \gamma$ is also a minimizer. Second, for any scalar $\gamma < 1$, then $U(\gamma x) = \gamma U(x) < U(x)$. By the second property, $x_i = 0$ for all i is a minimizer and therefore $x_i = \gamma$ for any scalar is also a minimum.

Two solution to these problems which lead to better graph embedding are discussed in the next two sections. The key idea in each solution is to add a set of constraints to the optimization problem to restrict the possible solutions x^*.

4.2 Spectral Embedding

One set of constraints that yields a non-trivial solution results in the following optimization problem

$$\begin{array}{ll} \min\limits_{X} & U(X) \\ \text{s.t.} & \sum_i X_{ik} = 0 \qquad\quad \text{for } 1 \le k \le d \\ & \sum_i X_{ik} X_{il} = n\delta_{kl} \quad \text{for } 1 \le k \le l \le d. \end{array} \tag{3}$$

In matrix notation, we have

$$\begin{array}{ll} \min\limits_{X} & \text{trace}(X^T L X) \\ \text{s.t.} & X^T e = 0 \\ & X^T X = n I_d, \end{array}$$

where I_d is the $d \times d$ identity matrix.

We interpret these constraints as follows. The constraint $\sum_i X_{ik} = 0$ fixes the center of the resulting embedding at the origin. In light of the problems discussed at the end of the previous section, this constraint removes the problem of finding non-unique minimizers by adding scalars. Next, taking $k = l$ the constraint $\sum_i X_{ik}^2 = n$ fixes $X_{ik} \neq 0$ for at least one vertex i in each coordinate k and prevents the minimizer from collapsing to the origin.

When $k \neq l$, the constraint $\sum_i X_{ik} X_{il} = 0$ enforces orthogonality. The implication of this constraint is that the resulting embedding must use all d dimensions available. To understand why this aspect is important, notice that the objective function, equation (2), is independent between dimensions. This constraint correlates the solution X between dimensions. However this constraint also implies that the solution in dimension $k+1$ has a higher energy than dimension k.

In fact, the solution to equation (3) is known analytically. The minimizing matrix X is composed of the eigenvectors of L corresponding to the second through $d+1$th smallest eigenvalues. In fact, in one-dimension, the solution of this optimization problem gives the Fiedler vector of the graph [5]. In higher dimensions, this embedding is well studied under many different names [2].

4.3 Spherical Embedding

Another set of constraints that yields a non-trivial embedding gives the optimization problem

$$\begin{array}{ll} \min\limits_{X} & U(X) \\ \text{s.t.} & \sum_i X_{ik} = 0 \qquad\quad \text{for } 1 \le k \le d \\ & \sum_{k=1}^{d} X_{ik}^2 = 1 \quad \text{for } 1 \le i \le n. \end{array} \tag{4}$$

In matrix notation, the problem is

$$\begin{array}{ll} \min\limits_{X} & \text{trace}(X^T L X) \\ \text{s.t.} & X^T e = 0 \\ & \text{diag}(XX^T) = e. \end{array}$$

As in the previous problem, the constraint $\sum_i X_{ik} = 0$ implies that the center of the embedding is at the origin. The constraint, $\sum_{k=1}^{d} X_{ik}^2 = 1$, however, fixes each point on the surface of a d-dimensional hyper-sphere. A result of this constraint is that the resulting embedding cannot collapse to the origin.

The cost of this second constraint is that there is no longer an analytical solution to the embedding problem. Instead, we must employ non-linear numerical optimization procedures to compute the embedding X that minimizes equation (4). A second consequence of this constraint is that the embedding problem is intractable in one-dimension. The surface of a one-dimensional hyper-sphere is simply the set of points $\{1, -1\}$. Thus, in one-dimension, this constraint yields an integer optimization problem.

The earliest reference (of which we are aware) to the idea of spherical mapping is from Antonio and Metzger in their work on mapping processes to a hypercube multi-processor configuration [1]. Further, this same optimization problem results from examining low-rank solutions to the semi-definite program (SDP) in Goemans and Williamson's approximation algorithm for the minimum bisection problem [4, 7]. Recently, others have used SDP methods for low-dimensional embeddings [14].

4.4 Embeddings in Three-Dimensions

In the previous sections, we wrote the optimization problems for each of the embeddings for an arbitrarily high dimension. While this approach illuminates some of the analytical relationships between the method, it may hide some details. In this section, we restrict ourselves to three-dimensions and explicitly write the optimization programs.

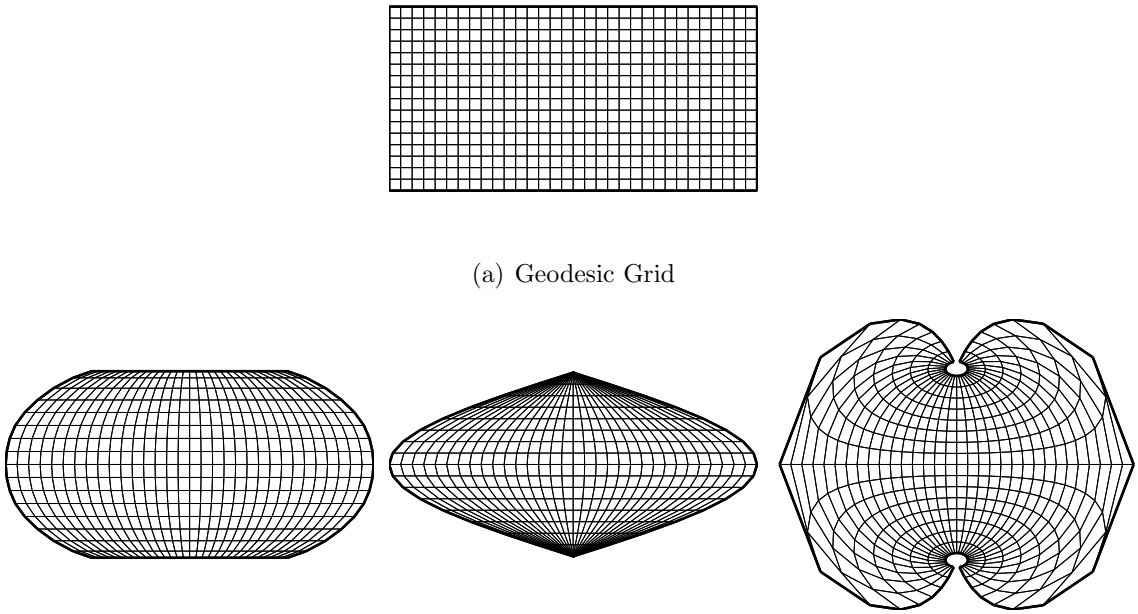
For this section, let x_i, y_i and z_i represent the first, second, and third coordinates of the embedding point for vertex i. The spectral embedding optimization problem is

$$\begin{array}{lll} \min\limits_{x,y,z} & \sum_{ij} L_{ij} x_i x_j + L_{ij} y_i y_j + \sum_{ij} L_{ij} z_i z_j & \\ \text{s.t.} & \sum_i x_i = 0 \quad \sum_i y_i = 0 & \sum_i z_i = 0 \\ & \sum_i x_i^2 = n \quad \sum_i y_i^2 = n & \sum_i z_i^2 = n \\ & \sum_i x_i y_i = 0 \quad \sum_i x_i z_i = 0 & \sum_i y_i z_i = 0. \end{array} \tag{5}$$

As previously mentioned, the solution to this optimization problem is to set x, y and z to be the eigenvector corresponding to the second, third, and fourth smallest eigenvalues of L, respectively.

The spherical embedding problem in 3d is

$$\begin{array}{lll} \min\limits_{x,y,z} & \sum_{ij} L_{ij} x_i x_j + L_{ij} y_i y_j + \sum_{ij} L_{ij} z_i z_j & \\ \text{s.t.} & \sum_i x_i = 0 \quad \sum_i y_i = 0 & \sum_i z_i = 0 \\ & x_i^2 + y_i^2 + z_i^2 = 1 \text{ for all } i. & \end{array} \tag{6}$$



(a) Geodesic Grid

(b) Robinson (c) Sinusoidal (d) Equidistant Azimuthal

Figure 5: In this figure, we plot a series of different map projections. We begin with a simple geodesic (latitude/longitude) layout. Figures (b)-(d) show three different projects each preserving some visual aspect of the geodesic lines. Figure (b) actually draws a slight perturbation of the geodesic grid to better represent the behavior of the projection near the hemisphere edges.

The solution of this problem is a set of coordinates on the unit sphere. Also, to reiterate, there is no known analytical solution to this problem and it must be solved by a numerical optimization routine.

5 Map Projections

At the end of the last section, we determined two methods to embed the graph. Jumping ahead to the results, the most useful embedding is the spherical embedding. In order to use the spherical projections to compute a two-dimensional layout, however, we must compute planar points from locations on the surface of the sphere.

This problem has been studied for over 2,000 years under a different name: drawing a map [13, 3]. The map drawing community produced a plethora of techniques to draw the Earth on a flat map. The large number of techniques exists because there is no perfect map drawing. Loosely speaking, the three types of distortion are area, length, and angles. An area preserving map is called an *equal area* projection, a length preserving map is an *equidistant* projection, and an angle preserving map is a *conformal* projection.

After surveying a large number of map projections using Matlab's Mapping Toolbox [10], we decided to investigate three projections, Robinson, sinusoidal, and equidistant azimuthal. Figure 5 displays the result of each of these projections on the geodesic grid. We report on some properties of each projection below. None of the projections we choose are conformal. For our application, this property is not important.

Robinson The Robinson projection preserves no property. It distorts area, length, and angles. Instead of a mathematical formula, the Robinson projection is a set of heuristics to construct an appealing map drawing. The National Geographic Society has used the Robinson projection since 1988 for its world map [3].

Sinusoidal The sinusoidal projection, also known as the Sanson-Flamsteed projection, is an equal area projection. At the poles, the projection collapses to a point.

Equidistant Azimuthal The equidistant azimuthal projection is an equidistant projection. The projection unrolls the near hemisphere into a circle such that distances are preserved near the center of the projection. The far hemisphere is highly distorted.

6 Implementation

We used a combination of off-the-shelf and custom software in this project. The implementation is divided into many scripts that accomplish one part of the data processing pipeline. This approach allowed us the flexibility of choosing different languages for many of the different steps. Largely speaking, the implementation nicely divides between preprocessing, embedding, and visualization. Our implementation and processing scale to datasets with millions of artists.[1]

Preprocessing Our preprocessing scripts were written in Matlab, Perl, and C++. We first used custom Perl scripts to convert the raw datafiles provided by Yahoo!'s music services into a sparse matrix representation. For the LAUNCHcast data, we performed the filtering and similarity graph construction using two C++ programs. The program to build the similarity graph used the routine `CLUTO_V_GetGraph` from CLUTO [8]. Preprocessing the MusicMatch data was done entirely in Matlab.

Embedding To compute the spectral embeddings, we used the `eigs` routine in Matlab. For the spherical embedding, we took advantage of the relationship with a low-rank semidefinite program and used the SDP-LR software to compute these embeddings [4]. To unroll the spherical embedding into a planar set of points, we used the routines `eqdazim`, `sinusoid`,

[1] While the computation and visualization tools scale to this level, their utility for such a large dataset becomes questionable. For a dataset with one million points, the points are almost completely dense throughout the sphere and we tend to observe less interesting clustering behavior.

and `robinson` from the Matlab Mapping Toolbox [10]. We manually picked the center of each projection.

Visualization tools We used two custom built visualization tools for this experiment: plot3d, and visgraph. We have used both programs on graphs with over 100,000 vertices and one million edges.

First, we use plot3d to investigate the structure of the three-dimensional spectral and spherical embeddings. The plot3d program takes an edge list and a set of coordinates as input, and optionally, a clustering partition for coloring the nodes. It is a C++ program that uses OpenGL to draw each node and each edge. Users interactively rotate and zoom the embeddings of the graph to gain a perspective on the embedding quality. It includes special code for the spherical embedding to draw a reference sphere at the center.

Once we determine that an embedding is worth pursing, we unroll the three-dimensional spherical points into two dimensions and use the visgraph program to interactively browse a two-dimensional graph layout. Like plot3d, visgraph takes similar input, but can also accept a set of labels for each node in the graph. The nodes (points) and the edges (lines) are alpha-blended to show local density. The program maintains a quad-tree data structure to allow fast label browsing using a circular "brushing cursor" to reveal labels. This technique avoids cluttering the display with all labels simultaneously. Figure 8 demonstrates examples of the visualizations from visgraph.

Publication Tools For publication images, we use a set of perl scripts to convert raw embedding data into POVRAY files for further rendering. Also, the visgraph program has an option to save the current drawing as an SVG file. Both POVRAY and SVG files are resolution independent formats which make them ideal for generating high quality still images of the datasets.

7 Results and Discussion

Figure 9-13 present the main results of the paper. In figures 9 and 10, we compare the embeddings computed by equation (6) and equation (5), respectively. Figures 12 and 13 show the LAUNCHcast and MusicMatch datasets for each of the map projections described in section 5. Throughout the rest of this section, we will use these figures to argue that the spherical embeddings are superior to the spectral embeddings, and that the Robinson projection yields the "best-looking" map projection for our data.

First, the two spherical embedding figures show each of the artists represented as a single point on the surface of the sphere. The color of each point identifies the cluster an artist belongs to from an independent clustering of each similarity graph using CLUTO [8]. The spherical embeddings show a nice grouping of points of similar color, which indicates that the coordinates for the points largely agree with the clustering. Thus, these embeddings are successful and we are representing "similar" artists "nearby" each other. In the two spectral embedding figures, the results are less clear. Here, we have drawn the artists using

the same colors. While the LAUNCHcast spectral embedding shows a reasonable separation between groups of artists, the MusicMatch spectral embedding yields one large heterogeneous "clump" of artists. In fact, for both LAUNCHcast and MusicMatch, the spherical embedding displays a larger number of small dense "clusters" of artists than the spectral embeddings. The failure of the eigenvectors of the Laplacian to yield a good embedding may be related to the power-law structure in the data as was observed by Lang [9].

We have omitted the comparisons in two-dimensions due to space constraints because the results are uniformly worse for the spectral embedding. Briefly, the LAUNCHcast image shows only the approximately $\pm$ 45 degree "arms" of the clustering. The MusicMatch image shows a large mess of points at the origin.

We use the CLUTO clustering for two purposes: first, to evaluate the results, and second, to improve the visualization. CLUTO is a high-quality clustering program with thousands of successful experiments. In this case we use CLUTO to generate a set of 50 clusters for each of the similarity graphs.

Next, we will address the map projections. Figures 12 and 13 show each of the map projections from the display used in visgraph. Each artist is a point colored according to the artist's cluster from CLUTO. All edges longer than distance 2 for LAUNCHcast and 3 for MusicMatch are not drawn — many of these long edges would have "wrapped around" the other side of the sphere. Each edge is drawn with a low alpha-blending value (alpha = 0.025) so "brighter" regions represent higher edge density.

The distortion at the edges in the latitude/ longitude drawing are extreme and visually unappealing for both graphs. The severe distortion at the edge of the sinusoidal projection is also unappealing. Both of these projections either expand (latitude/longitude) or compress (sinusoidal) important groups of points. Subjectively, the Robinson projection looks most appealing and removes the most severe expansion from the latitude/longitude projection while maintaining the a nice expansive layout. Finally, the equidistant azimuthal projection displays the near hemisphere (the center of the picture) with a nice separation, but the far hemisphere is overly distorted into a circular shape.

8 Exploration

In this section, we describe our explorations of the "World of Music." All of the visualizations use the Robinson projection. We do not explain all of the insight we have gained from these visualizations, only a few succinct points that highlight the utility of the visualization.

LAUNCHCast We have explored the LAUNCHcast data extensively. In this section, we note three interesting features. First, in the middle of one of the empty areas ("the oceans"), there is a single artist. Second, there is a strong relationship between "indie" music and "mainstream" music, belying the *independence* of "indie" music. Third, there is a group of "bridge" artists between "mainstream" music and "techno" music.

The artist in the middle of the "ocean" is Austin Powers. Although attributing a cause to the placement of a particular artist is difficult, we can form some hypotheses. If Austin

Powers is an artist, then their position in the layout has been affected by the numerous errant rankings placed on them in reference to Austin Powers, the movie.[2] Instead, if Austin Powers refers to the movie, the position may be influence by the non-standard use of the movie title as the musical artist name. Either way, it indicates an outlier in the data that merits further attention.

Second, we plotted the locations of a few well-known "indie" musical artists, The Shins, Death Cab for Cutie, and The Decemberists, along with the location of more well known "mainstream" musical artists, U2, Soundgarden, Courtney Love, and Stereophonics. The visualization shows that the group of "indie" artists forms a tight cluster near the more extended cluster of mainstream artists. Together, the "indie" and "mainstream" artists form the region that looks like North America in the center of the Robinson projection. While alternate specific analysis would have revealed this result, the visualization makes it immediate.

Finally, the visualization shows a set of "bridge" artists. These artists form a small cluster between "mainstream" music and "techno" music. Bridge artists, and particularly a group of bridge artists, are important sets of musical artists because they relate multiple clusters.[3] Again, the visualization immediately highlights this group due to the pattern of edges entering and existing.

MusicMatch While browsing through the MusicMatch data, we found many interesting clusters and artists. In this section, we highlight a few of our observations. First, we found a cluster of Latter Day Saints musicians. Second, we found a few interesting artists that "connect" various group. Finally, there is a cluster of children's musicians. See figure 7 for supporting visualizations from the visgraph tool.

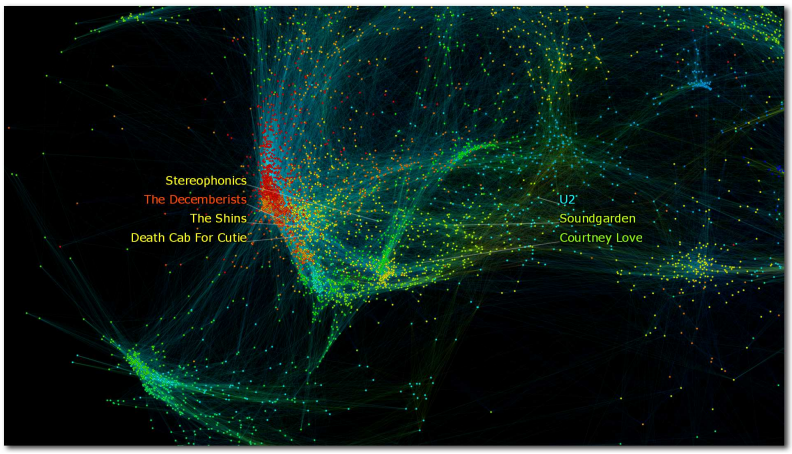
The Latter Day Saints (LDS) cluster is interesting from a few perspectives. Because of the uniform color of all the artists, CLUTO identified this grouping of artists as well. However, finding this group by browsing the CLUTO clusters is onerous. We asked CLUTO for 50 clusters, which it happily provided. Determining the *semantics* behind these clusters, however, is beyond CLUTO and requires human intervention. Globally, the LDS cluster stands out in the visualization due to its unique shape. The shape invites further investigation. Further, once someone sees the artist "Church of Jesus Christ of Latter Day Saints," he or she can easily hypothesize and confirm the nature of the cluster. This process is more engaging and fulfilling than a brute-force human evaluation of all 50 clusters.

Second, while browsing around the graph, some artists stand out with a "connector" pattern. Typically, these artists have multiple (3-4) dense sets of edges emanating from a single point. Visually, these artists break from the regularity of the surrounding region. In particular, Will Downing is an "connector" artist between soul, R&B, and jazz. Producers may like such artists because they influence multiple sets of other artists; consumers may like such artists because they help expand musical tastes.

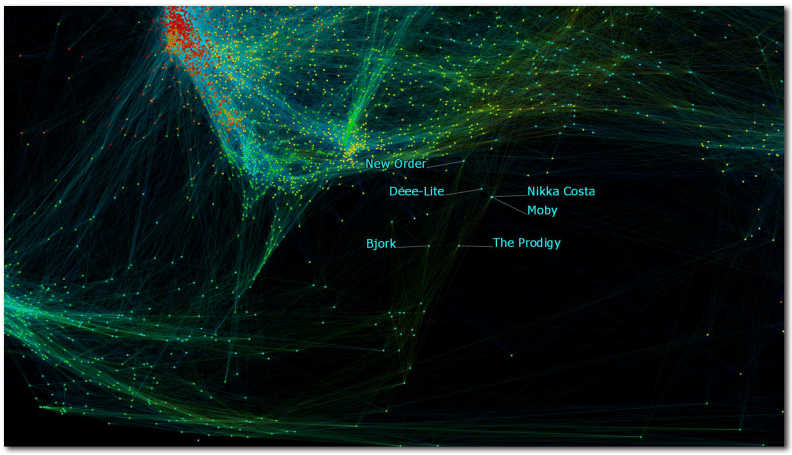
[2]Previously, we were under the impression that Austin Powers was indeed an artist unrelated to the movie. However, recent searches to confirm this fact have not been successful.

[3]We write more on the importance of connector artists below.

(a) Austin Powers in the middle of nowhere.



(b) Indie music is closely related to mainstream music.



(c) The bridge from mainstream to techno music.

Figure 6: Some exploration results from the LAUNCHcast dataset.

Finally, we have highlighted a set of children's musicians. While the particular artists are fairly well known and contain little important information, the surrounding region and the connections are rich with data that should interest musical producers and marketers. Unfortunately, without expertise in the domain, we cannot evaluate this region ourselves. However, the visual presentation of the material is key to identifying the possibility of extracting this data. Without the visual presentation, it would not have occurred to us to seek more information about children's musicians and their relationships to the other musicians.

9 Related Work

[Note: this section is incomplete and lacks appropriate references.]

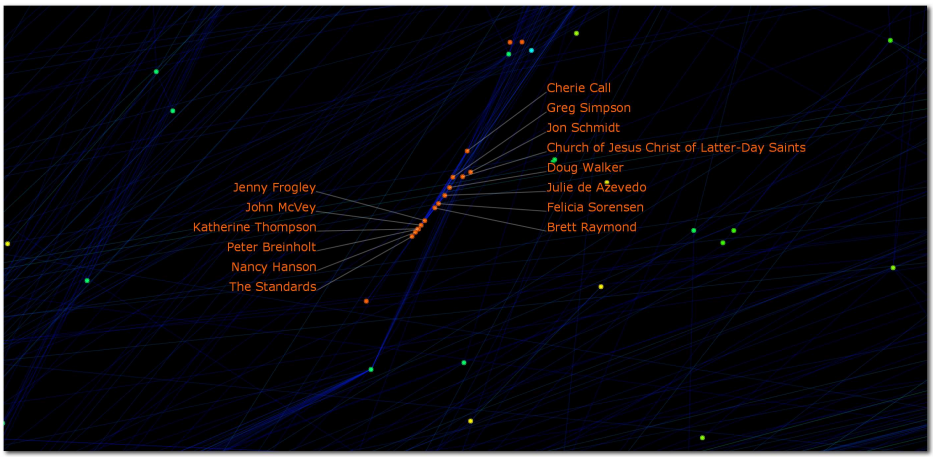
The major contribution of this paper are a more detailed exposition of the techniques used in [6] as well as demonstrating that the techniques continue to work on another dataset as well. To reiterate, we view a data exploration problem as a dimensionality reduction and graph embedding problem. From each of these viewpoints, there exists significant literature.

One of the standard methods for graph embedding is multi-dimensional scaling (MDS). This method is not as computationally scalable as our approach. In order to make MDS techniques effective, they must include a repulsive force which includes a computational cost of $n \log n$ (for a fast multipole/quad tree appraoch) or n^2 (for an exact computation) per iteration. Our procedure does not include a repulsive force and has a linear cost per iteration. Instead, it relies on the constraints of the optimization problem for "sufficient repulsion." To be clear, there is no reason to believe the constraints in the hyper-sphere optimization problem will lead to anything like a repulsive force, but for the music graphs, the final effect is a nice cluster separation and hence, "sufficient repulsion."

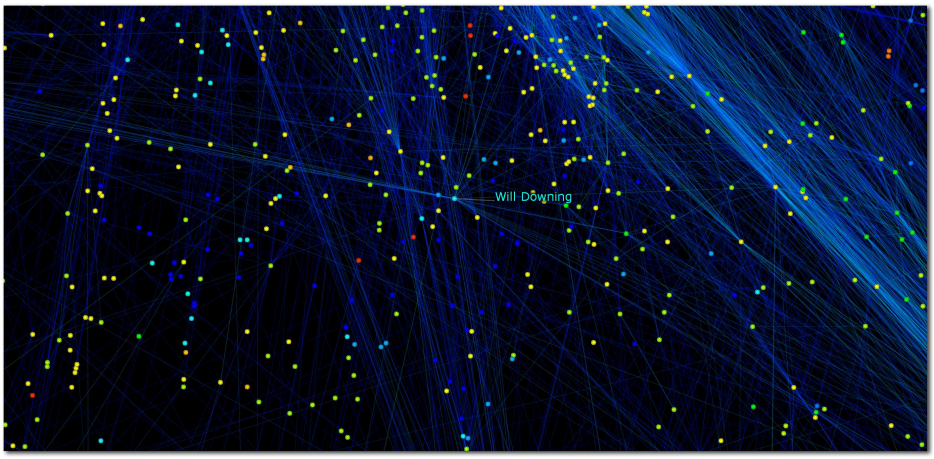
The MDS approach is strongly related to a force-directed layout. Computed an embedding of the similarity graph using the yEd graph layout tool. As exhibited in figure 11, the results are inferior to the embeddings computed with our tools.

10 Future Work

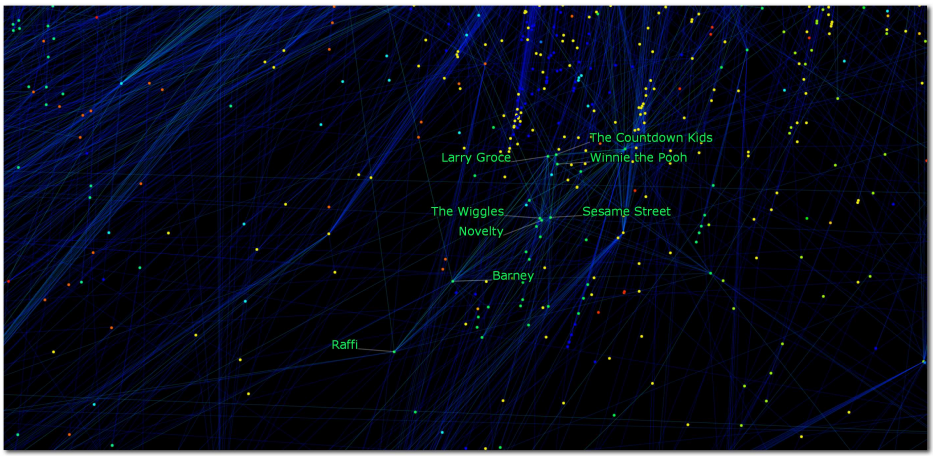
There are a few aspects of this project that merit further investigation. First, the layout of the unrolled maps needs to be hand tuned to determine a good origin and rotation of the sphere. While conceptually simple, this problem has a strong combinatorial aspect which makes straightforward optimization techniques difficult. Using a grid search based approach would be a significant step toward automating this step. Second, the current approach is to project the points on the surface of the sphere to a two-dimensional plane. A useful alternative may be to directly visualize the sphere and avoid the projection to two-dimensions. Finally, this approach works for large datasets (more than 250,000 nodes), but the resulting embeddings are nearly useless due to the density of points throughout the sphere. To be clear, there is structure present in the embedding, but a direct visualization seems of limited use. Our belief is that a two-dimensional space is too restrictive and we need visualization techniques



(a) The cluster of Latter Day Saints musicians.



(b) An artist that "connects" a set of clusters



(c) The cluster of children's musicians.

Figure 7: Some exploration results from the MusicMatch dataset.

for higher dimensional point sets.

11 Conclusions

In this paper, we built a world of music based on ratings from users. We used two datasets derived from user ratings to compute a similarity graph between musical artists. To embed the graph, we examined two embeddings based on quadratic energy functions. Empirically, we found the spherical embedding to be superior to the spectral embedding. To compute a set of planar points from the spherical embedding, we considered three map projections, of which the Robinson was the most appealing. By combining these embeddings with a set of visualization tools, we developed a system to interactively browse the underlying dataset and help build an understanding for the world of music.

12 Acknowledgments

Todd Beaupre deserves special acknowledgment for giving us the LAUNCHCast dataset and spending his time explaining the data and what Yahoo! Music would like to understand about the data. Chris Staszak provided the MusicMatch dataset.

References

[1] John K. Antonio and Richard C. Metzger. Hypersphere mapper: a nonlinear programming approach to the hypercube embedding problem. *J. Parallel Distrib. Comput.*, 19(3):262–270, 1993.

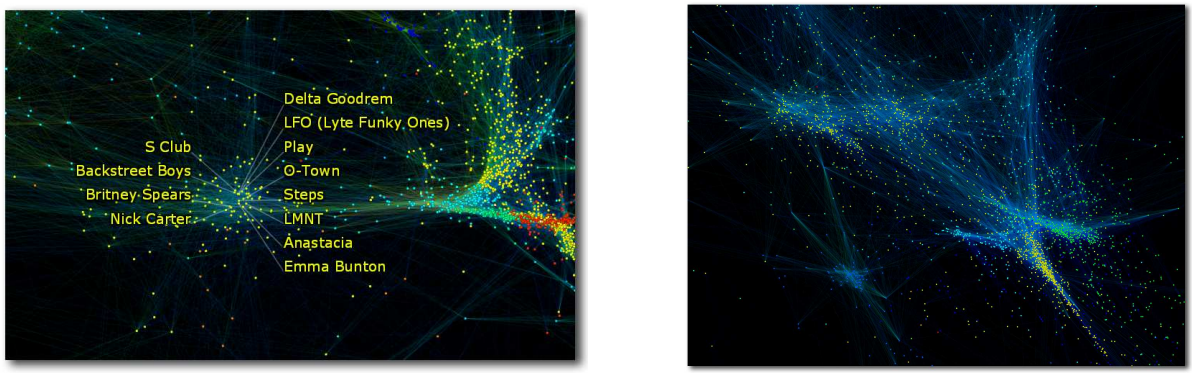
[2] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comp.*, 15(6):1373–1396, 2003.

[3] Lev M. Bugayevskiy and John P. Synder. *Map Projections: A Reference Manual*. Taylor and Francis, 1995.

[4] Samuel Burer and Renato D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (series B)*, 95(2):329–357, 2003.

[5] Miroslav Fiedler. Algebraic connectivity of graphs. *Czech. Math J.*, 23:298–305, 1973.

[6] David Gleich, Kevin Lang, Matt Rasmussen, and Leonid Zhukov. The world of music: Sdp embedding of high-dimensional data. In *Information Visualization*, Minneapolis, Minnesota, 2005. Interactive Poster.



(a) visgraph showing labels on LAUNCHcast.

(b) visgraph showing zoomed edge on MusicMatch.

Figure 8: visgraph usage shots.

[7] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. Assoc. Comput. Mach.*, 42:1115–1145, 1995.

[8] George Karypis. Cluto – a clustering toolkit. Technical Report 02-017, University of Minnesota, Department of Computer Science, 2002.

[9] Kevin Lang. Fixing two weaknesses of the spectral method. In *Advances in Neural Information Processing Systems*, 2005.

[10] MathWorks. Matlab mapping toolbox: Version 2.0.2, May 2004.

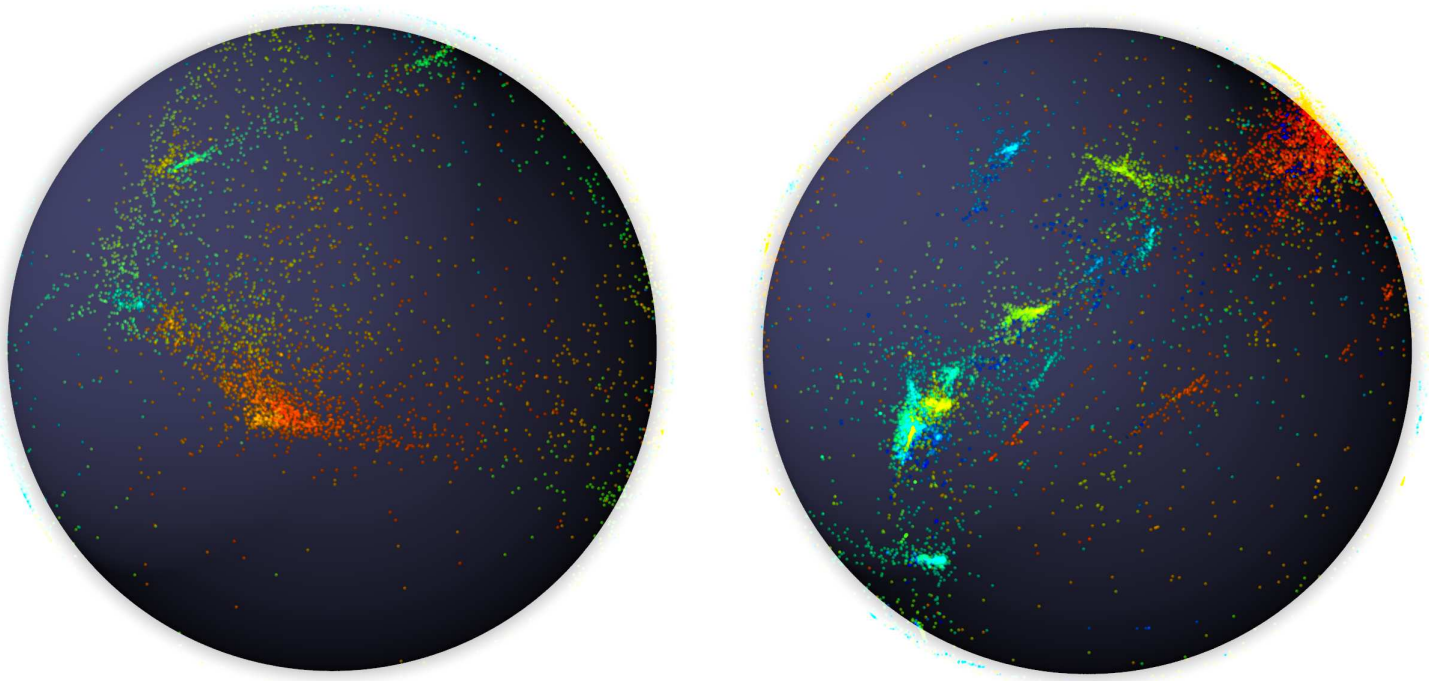
[11] J. C. Platt. Fast embedding of sparse music similarity graphs. In *Advances in Neural Information Processing Systems*, volume 16, pages 571–578, 2004.

[12] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2626, December 2000.

[13] John P. Snyder. *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press, 1993.

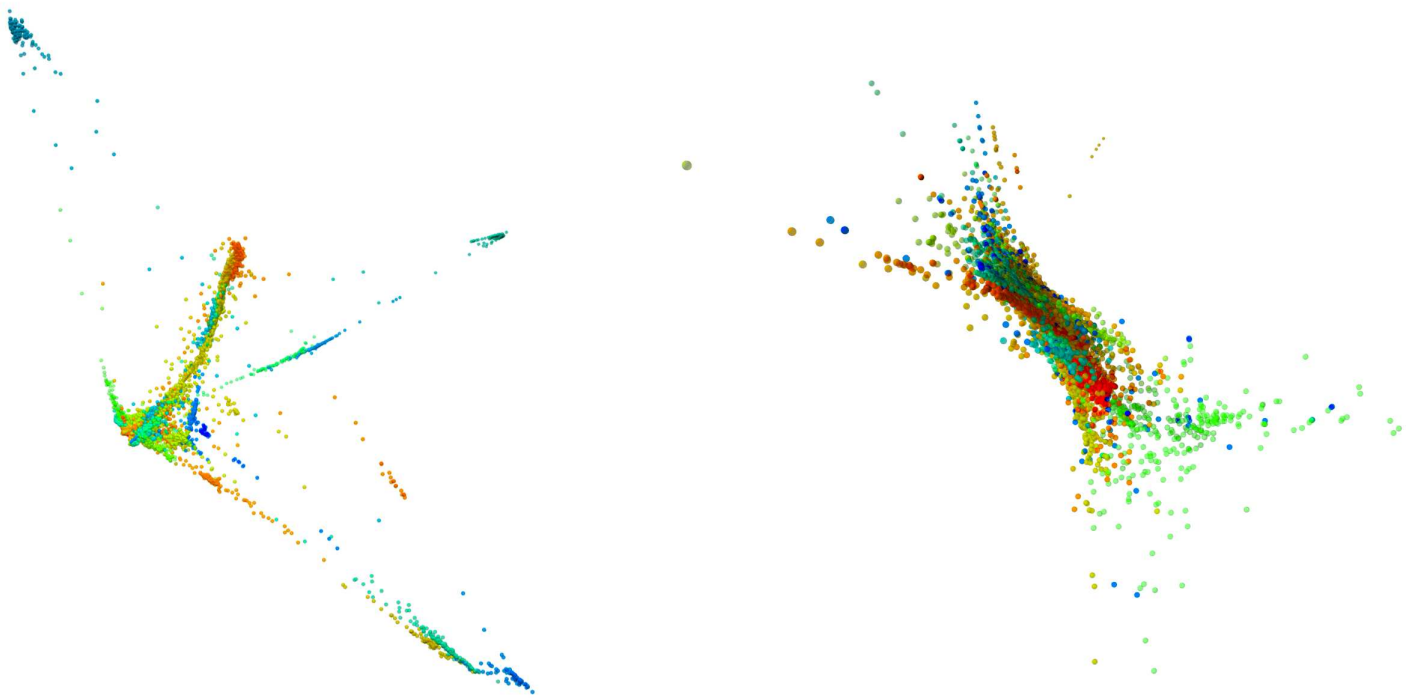
[14] Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[15] Brian Whitman and Steve Lawrence. Similiarty for music from community metadata. In *International Computer Music Conference*, pages 591–598, Goeteborg, Sweden, September 2002.



(a) LAUNCHcast (b) MusicMatch

Figure 9: Spherical embeddings of the similarity graph from both datasets displayed as point-clouds.



(a) LAUNCHcast (b) MusicMatch

Figure 10: Spectral embeddings of the similarity graph from both datasets displayed as point-clouds. These pictures represent 3d point clouds. For the MusicMatch picture, this figure excludes a few points that were outside the dense point-cloud near the origin.

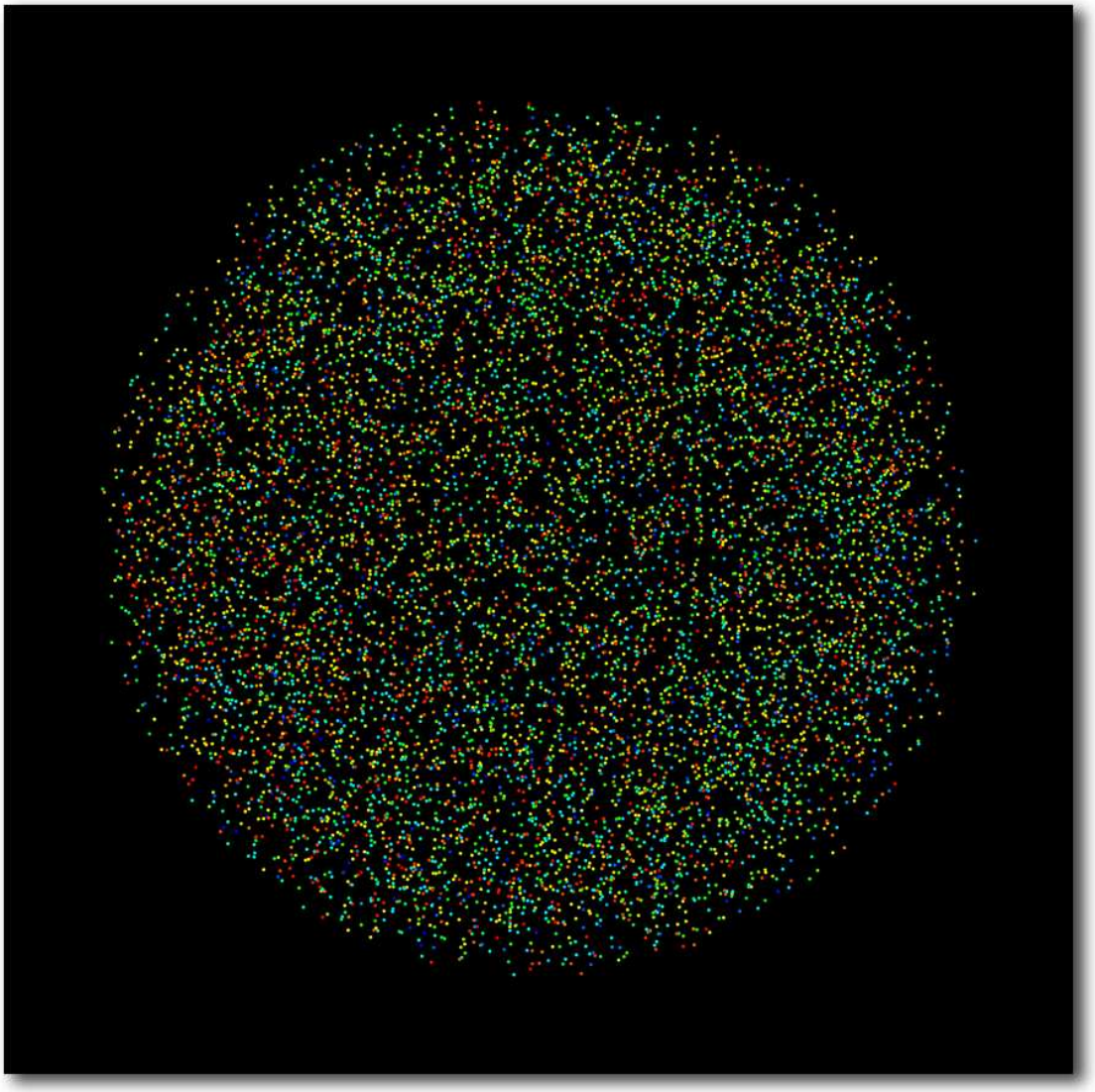
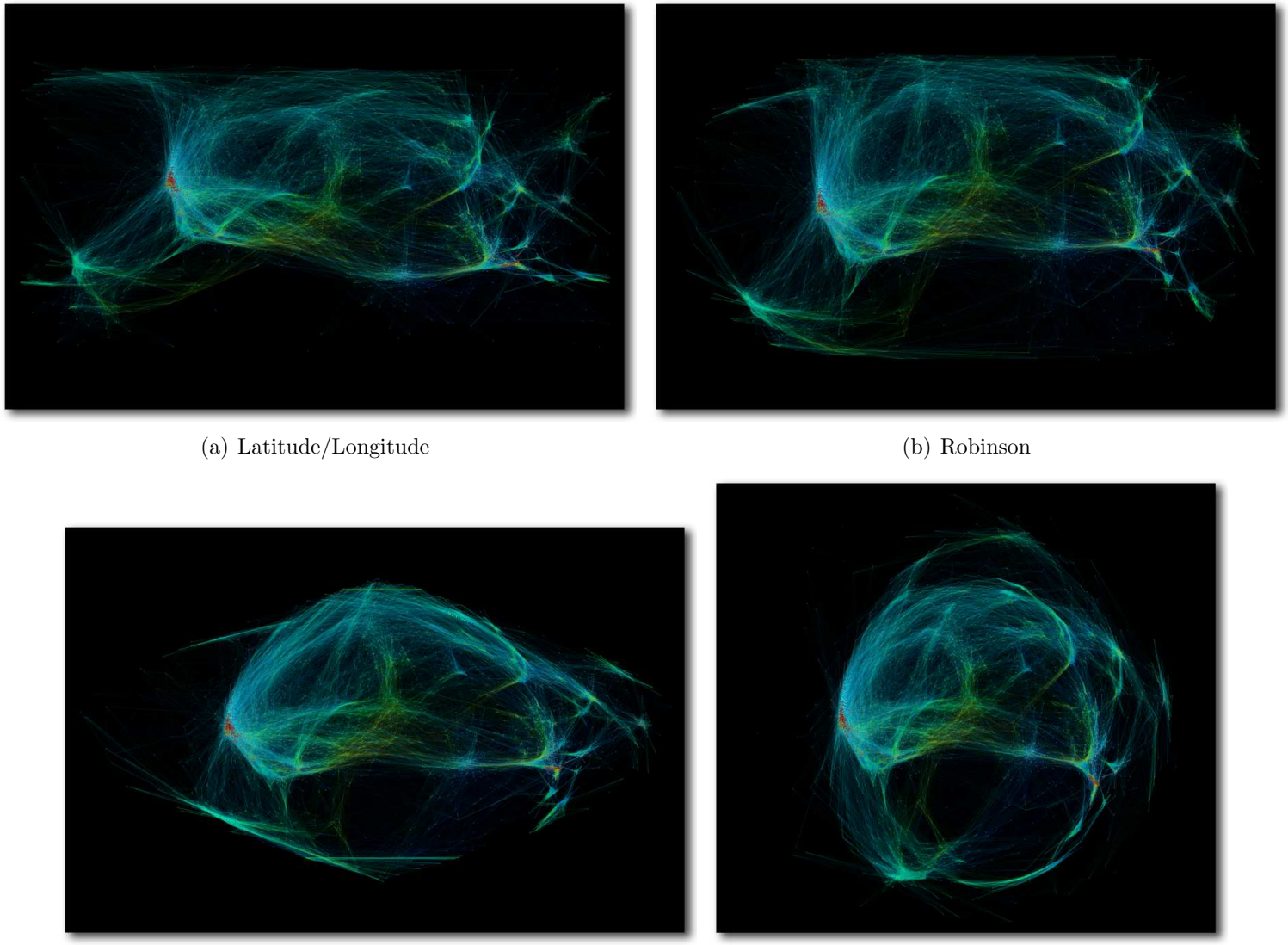


Figure 11: A force directed layout of the LAUNCHcast similarity graph using the yEd graph editor and layout program. We exported the layout from yEd and used visgraph to generate the image above. As in the other pictures, the colors come from the CLUTO clustering. This image shows that the force-directed approach used in yEd is not able to separate the clusters like the energy-minimization approach we use.



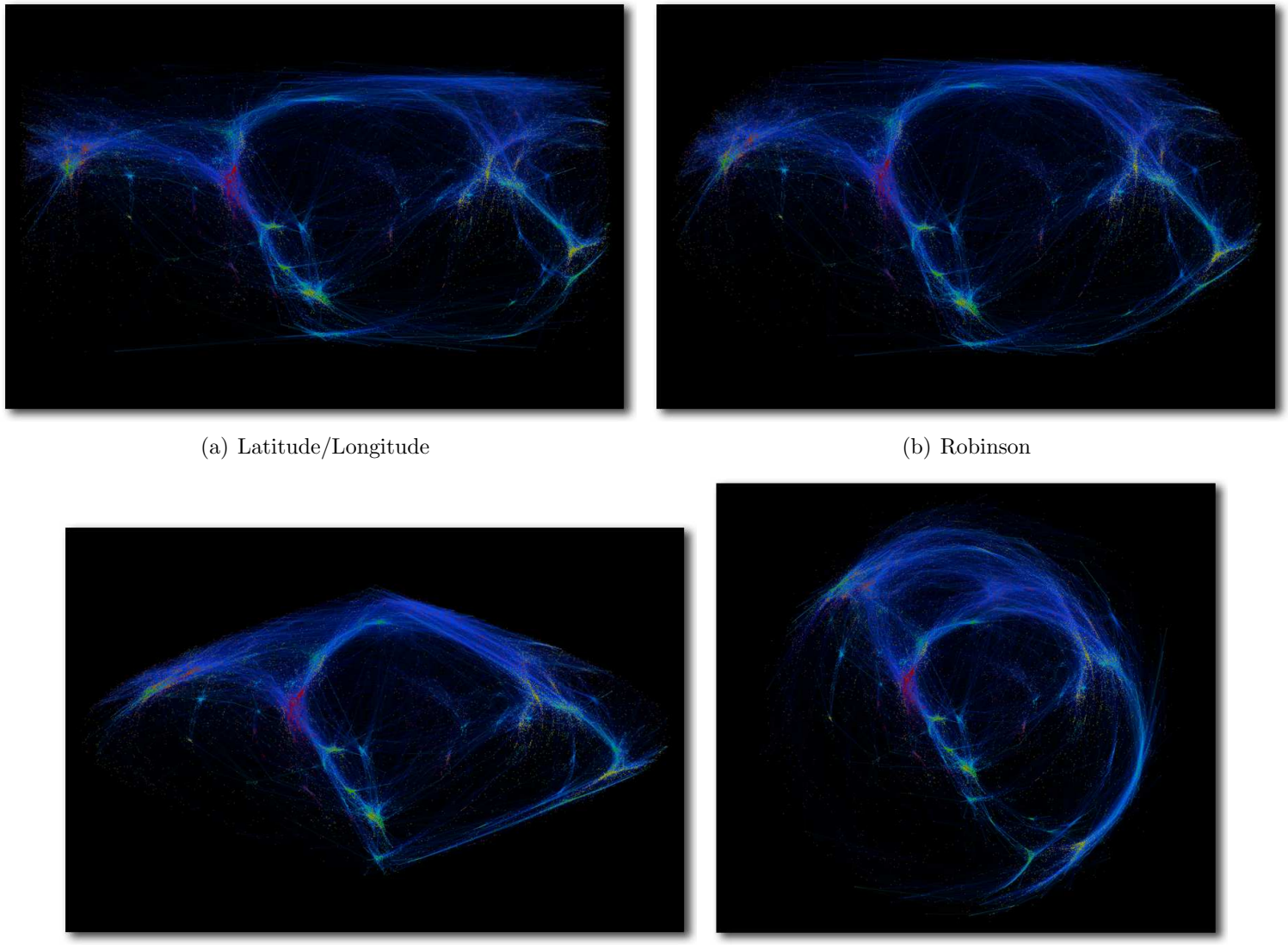
(a) Latitude/Longitude

(b) Robinson

(c) Sinusoidal

(d) Equidistant Azimuthal

Figure 12: Map projections of the LAUNCHcast spherical embedding.



(a) Latitude/Longitude

(b) Robinson

(c) Sinusoidal

(d) Equidistant Azimuthal

Figure 13: Map projections of the MusicMatch spherical embedding.

A Quadratic Energy Embeddings

As we stated in section 4, one way to visualize an undirected graph $G = (V, E, w)$ is to assign coordinates to every node. This process is called *embedding* the graph. Ideally, we want a set of low-dimensional (ideally two-dimensional) coordinates with properties "like:"

- for each edge $(i, j) \in E$, vertex i and vertex j should be "close" in the embedding,
- for vertex pairs $(i, j) \notin E$ with a large number of edges on the shortest path between vertex i and vertex j, the vertices should be "far" in the embedding, and
- the embedding "reveals" non-trivial information about the graph.

We indicate these terms with scare-quotes to emphasize that these are not (yet) rigorous mathematical properties of the embeddings. Instead, these are subjective evaluation metrics we can use to compare and contrast different embeddings of the same data.

If we restrict ourselves to a one-dimensional embedding and label all the vertices with integers from 1 to n, we have a coordinate $x_i \in \mathbb{R}^1$ for each vertex $i \in V$. For convenience, we use the vector x to represent the set of all coordinates. Throughout this paper and in particular throughout this appendix, we will look at quadratic energy embeddings. A quadratic energy embedding is a quadratic difference term over the edges of the graph.

In the quadratic energy formula, each edge $e = (i, j)$ contributes "energy"

$$w_{ij}(x_i - x_j).$$

This term has the effect of pulling nodes connected with edges close because the minimum of this term is when $x_i = x_j$. For the entire graph G, the quadratic "energy" of the embedding is

$$U_G(x) = \sum_{(i,j)\in E} w_{ij}(x_i - x_j)^2. \tag{7}$$

We call this function an "energy" because of the relationship with the energy of a mass-spring system. Section **??** elaborates on this relationship.

An alternative graph embedding is to use a set of springs on the edges of the graph. For a linear spring with constant k, the potential energy stored in the spring with scalar extension x is

$$U(x) = \frac{1}{2}kx^2.$$

In our case, we have $k_{ij} = w_{ij}$ and $x = (x_i - x_j)$. This expression is why we call our quadratic energy embedding an "energy" embedding, if each edge is a spring, then we minimize the total potential energy in the system given by $U_G(x)$.