# Conformal Mapping by Computationally Efficient Methods

**Stefan Pintilie** and **Ali Ghodsi**
Department of Statistics and Actuarial Science
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1

## Abstract

Dimensionality reduction is the process by which a set of data points in a higher dimensional space are mapped to a lower dimension while maintaining certain properties of these points relative to each other. One important property is the preservation of the three angles formed by a triangle consisting of three neighboring points in the high dimensional space. If this property is maintained for those same points in the lower dimensional embedding then the result is a conformal map. However, many of the commonly used non-linear dimensionality reduction techniques, such as Locally Linear Embedding (LLE) or Laplacian Eigenmaps (LEM), do not produce conformal maps. Post-processing techniques formulated as instances of semi-definite programming (SDP) problems can be applied to the output of either LLE or LEM to produce a conformal map. However, the effectiveness of this approach is limited by the computational complexity of SDP solvers. This paper will propose an alternative post-processing algorithm that produces a conformal map but does not require a solution to a SDP problem and so is more computationally efficient thus allowing it to be applied to a wider selection of datasets. Using this alternative solution, the paper will also propose a new algorithm for 3D object classification. An interesting feature of the 3D classification algorithm is that it is invariant to the scale and the orientation of the surface.

## Introduction

Manifold learning is a significant problem across a wide variety of information processing fields including pattern recognition, data compression, machine learning, and database navigation. In many problems, the measured data vectors are high-dimensional but we may have reason to believe that the data lies on or near a lower-dimensional manifold. It is possible that the high-dimensional data points are multiple, indirect measurements of an underlying source, which typically cannot be directly measured. Learning a suitable low-dimensional manifold from high-dimensional data provides important information about the underlying source. There exist a class of algorithms that can discover this underling source in non-linear cases. This includes Locally Linear Embedding (LLE) (Roweis and Saul 2000) and Laplacian Eigenmaps (LEM) (Belkin and Niyogi 2003).



However, both LLE and LEM have a particular drawback: they do not produce conformal maps since the lower $m$-dimensional data produced by LLE or LEM, do not preserves the angles from the higher $d$ dimensional data.

Conformal map algorithms are important tools used in many disciplines such as data mining (Burges 2005), computer vision (Wang, Jin, and Gu 2007; Kulpinski 2002), and artificial intelligence (Gu and Yau 2003; Jenkins and Mataric 2002; Pless and Simon 2002).

The paper *Extension of Spectral Methods* by Sha and Saul (Sha and Saul 2005), introduces the interesting idea of using spectral methods like LLE and LEM to find the low dimensional manifold and then further modifying the output to produce a conformal map. The conformal map algorithm they propose finds a linear transformation such that when it is applied to the output from LLE or LEM the result is conformal with respect to the original high dimensional data.

The difficulty with this algorithm lies with the fact that it is computationally intensive. The piece of the algorithm that enables it to find the linear transformation uses Semidefinite Programming (Vandenberghe and Boyd 1996) which becomes slow when the LLE output is high dimensional. It can be shown that the Sha and Saul algorithm has a complexity of $O\left(m^9\right)$ where $m$ is the dimension of the output from LLE. As the value of $m$ increases the runtime required for such an algorithm becomes too large to be practical.

This paper will present an alternative to the Conformal Map algorithm proposed by Sha and Saul (Sha and Saul 2005). Our proposed algorithm maintains the idea of processing the output from LLE (or LEM) but avoids the use of semidefinite programming.

In this paper we will first discuss LLE and *Extension of Spectral Methods* by Sha and Saul. Afterward the paper will present a new method for the efficient conformal mapping and show why it is indeed faster. This paper will then validate the results of the proposed algorithm by comparing them to the results produced by the original Sha and Saul algorithm. Finally, the last part of this paper proposes a new algorithm that effectively classifies 3D surfaces using conformal maps.

## Locally Linear Embedding

Assume that there exists $n$ high dimensional points $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$. The main goal in LLE (Roweis and Saul

2000) is to map these high-dimensional data points to the single global coordinate system of the manifold such that the local geometry in the neighborhood of each data point is preserved. This proceeds in three steps. First, by identifying the neighbors of each data point $x_i$, then by computing the weights that best linearly reconstruct $x_i$ from its neighbors, and finally by finding the low-dimensional embedding vector $\{y_i\}_{i=1}^n \subset \mathbb{R}^p$ which is best reconstructed by the weights determined in the previous step.

After finding the $k$ nearest neighbors in the first step, a sparse matrix of local weights $W_{ij}$ is computed. These weights minimize the local reconstruction error $\sum_{i=1}^n ||x_i - \sum_{j=1}^k W_{ij}x_j||^2$ subject to the constraints that $\sum_j W_{ij} = 1$ and $W_{ij} = 0$ if $x_j$ is not a neighbor of $x_i$. The embedding, $Y = [y_1\ y_2\ \ldots\ y_n]$ a $p$ by $n$ matrix, is then obtained from the lowest $p$ eigenvectors of $M = (I-W)^T(I-W)$, excluding the smallest eigenvector corresponding to eigenvalue 0.

Note that the solution for $Y$ can have an arbitrary origin and orientation. In order to make the problem well-posed, these two degrees of freedom must be removed. Requiring the coordinates to be centered on the origin ($\sum_i^n y_i = 0$), and constraining the embedding vectors to have unit covariance ($Y^TY = I$), removes the first and second degrees of freedom respectively. Discarding the eigenvector associated with eigenvalue 0 satisfies the first of these two constraints.

## The Conformal Map (Sha and Saul 2005)

A conformal map is a low-dimensional embedding where the angles formed by three neighboring points in the original high dimensional dataset are equal to the angles between those same three points in the embedding. Consider the point $x_i$ and its neighbors $x_j$ and $x_k$ in $d$-dimensional space. Also, consider $z_i$, $z_k$ and $z_j$ to be the images of those points in the final embedding. If the transformation were a conformal map then the triangle formed by the $x$ points would have to be similar to that formed by the $z$ points. In the traingle formed by the $x$ points the expression $|x_j - x_k|$ represents the length of one side of the triangle while the expression $|z_j - z_k|$ represents the corresponding side in the embedding. Since the triangles are similar there must exist $\sqrt{s_i}$ such that:

$$\sqrt{s_i} = \frac{|x_j - x_k|}{|z_j - z_k|} = \frac{|x_i - x_k|}{|z_i - z_k|} = \frac{|x_i - x_j|}{|z_i - z_j|} \quad (1)$$

$$s_i = \frac{|x_j - x_k|^2}{|z_j - z_k|^2} = \frac{|x_i - x_k|^2}{|z_i - z_k|^2} = \frac{|x_i - x_j|^2}{|z_i - z_j|^2} \quad (2)$$

It is usually not possible to find a perfect embedding where all of the triangles are exactly similar to each other. Therefore, the goal is to find a set of $z$ coordinates such that the triangles are as similar as possible. This leads to the following minimization:

$$\min_{z,s_i} \sum_{j,k} \left( |z_j - z_k|^2 - s_i|x_j - x_k|^2 \right)^2 \quad (3)$$

Where the $x_i$ represent the initial points and the $z_i$ denote the points in the final embedding. Let $y_i$ represent the points in the embedding produced by LLE (or LEM). The $y_i$ points represent an intermediate step in the algorithm and so go part of the way to solving for $z_i$. Once LLE has produced a result, the goal of the algorithm becomes a search for a transformation matrix $L$ such that $z_i = Ly_i$ where the $z_i$ values satisfy the minimization in Equation (3).

$$\min_{L,s_i} \sum_{j,k} \left( |Ly_j - Ly_k|^2 - s_i|x_j - x_k|^2 \right)^2 \quad (4)$$

This should be done for all points $x_i$ and with the condition that the points $x_j$ and $x_k$ are the neighbors of $x_i$.

$$\min_{L,s_i} \sum_i \sum_{j,k} \eta_{ij}\eta_{ik} \left( |Ly_j - Ly_k|^2 - s_i|x_j - x_k|^2 \right)^2 \quad (5)$$

Where $\eta$ is an indicator variable and $\eta_{ij} = 1$ only if $x_j$ is a neighbor of $x_i$ otherwise $\eta_{ij} = 0$.

It has been shown in (Sha and Saul 2005) that the value for $s_i$ can be calculated via least squares and the initial minimization (5) can be rewritten as:

$$\text{minimize} \quad t \quad (6)$$

$$\text{such that} \quad P \succeq 0 \quad (7)$$

$$trace\,(P) = 1 \quad (8)$$

$$\begin{pmatrix} I & RVec\,(P) \\ (RVec\,(P))^T & t \end{pmatrix} \succeq 0 \quad (9)$$

Where $P = L^TL$, $I$ is the identity matrix, $R$ is a $m^2 \times m^2$ matrix computed from $\{x_i, y_i\}_{i=1}^n$, and $t$ is an unknown scaler. The condition $trace\,(P) = 1$ is added to avoid the trivial solution where $P = 0$. The optimization is an instance of semidefinite programming problem(SDP) over elements of unknown matrix $P$. After solving the SDP, the matrix $P$ can be decomposed back into $L^TL$ and the final embedding can be found by $z_i = Ly_i$ for all $i$.

## Alternative Solution

The solution presented in this paper will begin with the initial optimization problem (5) which has two unknown variables, $s_i$ and $L$. The value for the unknown $s_i$ parameters can be found through least squares. [1] Finding $s_i$ values and substitute them back into the initial equation (5) the result can be written as:

$$D\,(L) = \min_L Vec\,(P)^T\, QVec\,(P)$$

where $Vec$ operator represents the vectorization of a matrix such that a $m^2$ by $m^2$ matrix is converted into a column vector of length $m^4$, and the matrix $Q$ is known and can be computed from $\{x_i, y_i\}_{i=1}^n$.

In this case there exists the trivial solution where $L^TL = P = 0$. To remove this possibility, a constraint is added such that the trace of $P$ must be one. To add this constraint we use the Lagrange multiplier $\lambda_1$. The new optimization becomes:

$$D\,(L) = \min_L Vec\,(P)^T\, QVec\,(P) + \lambda_1\,(trace\,(P) - 1)^2 \quad (10)$$

$$= \min_L Vec\left(L^TL\right)^T QVec\left(L^TL\right) + \lambda_1\left(trace\left(L^TL\right) - 1\right)^2 \quad (11)$$

[1] In this section details are omitted due to lack of space.

The minimization equation is difficult to solve for $L$ primarily because the polynomial in the equation is of degree four.

To solve for $L$ we reduced the degree of the polynomial. We will rewrite $P = L^T L$ as $P = B^T C$ and add the constraint that $Vec(B) = Vec(C)$ for some unknown matrices $B$ and $C$. This is the key to the new algorithm since it reduces the degree of the polynomial to 2 in terms of $B$ and 2 in terms of $C$. Of course there are now two unknowns and there is an extra constraint to consider. To include the constraint into the equation we can form the Lagrangian by adding a new parameter $\lambda_2$ and adding $||Vec(B) - Vec(C)||^2$ to the minimization. Then Equation (12) can be written as:

$$\begin{aligned} D(B,C) &= \min_{B,C} Vec\left(B^T C\right)^T Q Vec\left(B^T C\right) \\ &\quad + \lambda_1 \left(trace\left(B^T C\right) - 1\right)^2 \\ &\quad + \lambda_2 ||Vec(B) - Vec(C)||^2 \end{aligned} \tag{12}$$

Now we can assume that one of the variables is known and then solve for the other one. Note that $trace\left(B^T C\right) = Vec(B)^T Vec(C) = Vec\left(B^T\right)^T Vec\left(C^T\right)$. And therefore:

$$\begin{aligned} D(B,C) &= \min_{B,C} Vec\left(B^T C\right)^T Q Vec\left(B^T C\right) \\ &\quad + \lambda_1 \left(Vec(B)^T Vec(C) - 1\right)^2 \\ &\quad + \lambda_2 ||Vec(B) - Vec(C)||^2 \end{aligned} \tag{13}$$

To develop the solution further, note that $Vec(ABC) = \left(C^T \otimes A\right) Vec(B)$ (Lutkepohl 1997) where $\otimes$ is the Kronecker product. Therefore the following can be substituted back into Equation (13):

$$Vec\left(B^T C\right) = \begin{cases} Vec\left(IB^T C\right) = \left(C^T \otimes I\right) Vec\left(B^T\right) \\ Vec\left(B^T CI\right) = Vec(C)^T (I \otimes B) \end{cases}$$

Now assume that $C$ is known and solve for $Vec\left(B^T\right)$. This can be solved in closed-form which leads to:

$$\begin{aligned} Vec\left(B^T\right) &= 2(\lambda_1 + \lambda_2) Vec\left(C^T\right) \\ &\quad \left((C \otimes I)\left(Q + Q^T\right)\left(C^T \otimes I\right)\right. \\ &\quad \left. + 2\lambda_1 Vec\left(C^T\right) Vec\left(C^T\right)^T + 2\lambda_2 I\right)^{-1} \end{aligned} \tag{14}$$

Similarly, if we assume that $B$ is known we can find the solution for $Vec(C)$.

$$\begin{aligned} Vec(C) &= 2(\lambda_1 + \lambda_2) Vec(B) \\ &\quad \left((I \otimes B)\left(Q + Q^T\right)\left(I \otimes B^T\right)\right. \\ &\quad \left. + 2\lambda_1 Vec(B) Vec(B)^T + 2\lambda_2 I\right)^{-1} \end{aligned} \tag{15}$$

## Algorithm

The algorithm is based on the theoretical solution presented in the previous section. The basic concept is to iteratively compute matrices $B$ and $C$ such that successive estimations will bring the result closer to the real solution. However it must be noted that not all problems posed have an exact solution. A perfect conformal map may not exist for a particular dataset. To address this concern an error margin was introduced: `ErrorMargin`. The algorithm will iterate by successively esimating $B$ and $C$ until the difference between the two matrices, as defined by an `ErrorFunction`, is smaller than the error margin that was defined. The algorithm will eventually converge because at each step the cost monotonically decreases. When that algorithm has converged, the matrices $B$ and $C$ will be equal to each other, or sufficiently similar with respect to the predefined margin of error. This will represent the solution for the unknown matrix $L$: $L = B \approx C$. The outline for the algorithm is:

```
X = GetInitialHighDimensionalPoints()
Y = RunLLE(X)
Q = ComputeQMatrix(X,Y)
B,C = RandomMatrix()
While (ErrorFunction(B,C) > ErrorMargin)
      C = EstimateC (B, Q)
      B = EstimateB (C, Q)
EndWhile

L = B or L = C since B≈C

Z = LY to find the final embedding
```

The functions `EstimateB` and `EstimateC` are based on the calculations in Equations (14) and (15) respectively. The function `ComputeQMatrix` depends only on known elements $\{x_i, y_i\}_{i=1}^n$. Details are omitted due to lack of space, but the derivation is easy to understand at a high level.

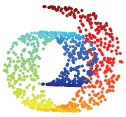
The `ErrorFunction` is composed of the minimization goal in Equation (12). The allowed error margin is a value set by the user and can be used as a tuning parameter. Other tuning parameters used in the algorithm are the $\lambda_1$ and $\lambda_2$ regularization parameters which must both be greater than zero since they decide how important each of the two restrictions are. For all of the tests performed in this paper the values for both $\lambda_1$ and $\lambda_2$ were set to 1.

## Experimental results

In this section, some results of the proposed algorithm are presented for a number of synthetic datasets as well as a few natural image datasets. In the first part of this section, the purpose is to show that the algorithm presented in this paper creates conformal maps of the same quality as those produced by the algorithm introduced by Sha and Saul. In the second part, the paper will examine some interesting results obtained by applying the conformal map algorithm to datasets composed of images of handwritten digits and to datasets of images of faces.

### Synthetic Datasets

Consider the Swiss Roll example in Figure 1. The image shows, in order from left to right, the initial 3D dataset, the result produced by LLE, the result produced by the new algorithm, and in the rightmost panel the result produced by the Sha and Saul algorithm. The third and fourth panels show that the two algorithms produce nearly identical embeddings. Both algorithms use LLE as the base dimensionality reduction algorithm. The result shown is produced consistently at this level of quality for both algorithms.



(a) Swiss Roll 3D Data

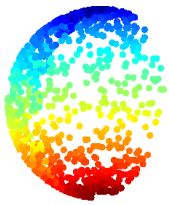
(b) LLE Result

(c) New Algorithm

(d) Sha and Saul Algorithm

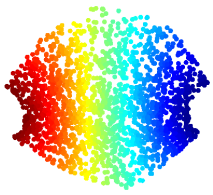
Figure 1: Swiss Roll with Embedding using LLE as base.

In most of the examples in this paper LLE is used as the base dimensionality reduction algorithm. However, the new conformal map algorithm as well as the Sha and Saul algorithm can also use LEM to reduce the dimensionality of the original data. The images in Figure 2 show that the algorithm can also be used for other types of manifolds. The Half Sphere in Figure 2 is a fairly difficult case since it cannot be unfolded properly without forcing some of the points together in the final embedding. However, the algorithm does a good job by clustering a number of points together at either end indicating correctly that those points are also close together in the 3D surface. LLE does not manage to cluster these end points correctly for the Half Sphere.

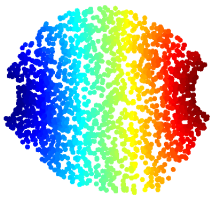


(a) Half Sphere 3D

(b) LLE



(c) New Algorithm

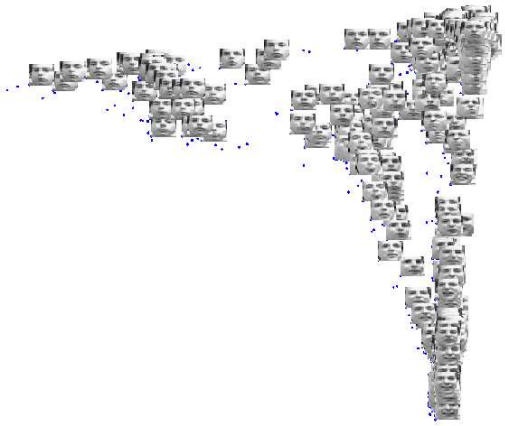


(d) Sha and Saul Result

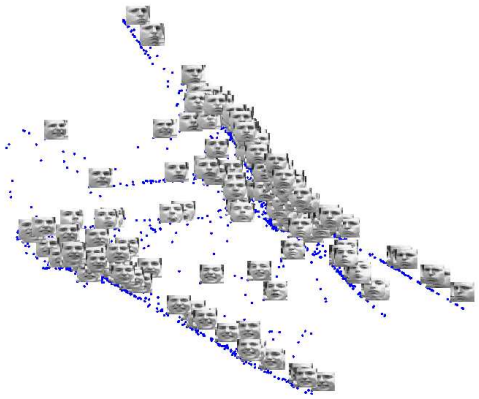
Figure 2: Half Sphere with Embeddings

## Real Datasets

While synthetic datasets are useful in showing that the new algorithm produces similar results to the ones presented by Sha and Saul, it is more interesting to apply the new conformal map algorithm to a number of real datasets. Consider the dataset composed of 1965 face images. Each face is a gray scale image of 20 by 28 pixels and so represents a point in a 560 dimensional space. The embedding produced by the first two dimensions of LLE is in the left panel of Figure 3 while the embedding of the first two dimensions of our algorithm is in the right panel of the same figure. LLE produces a figure where the happy faces are mostly at the bottom and the sad faces are mostly at the top. However, there is no clear separation between the two groups. In the conformal map embedding the points in the data are clearly separated into two sets. In the top set the facial expressions are unhappy while in the bottom set the facial expressions are happy. Each set is arranged in a line. The faces that are on the left side of the plot are facing to the right while the faces that are on the right side of the plot are facing to the left. This is an example of where the conformal map algorithm has been able to establish the proper position of each point relative to its neighbors.



(a) LLE

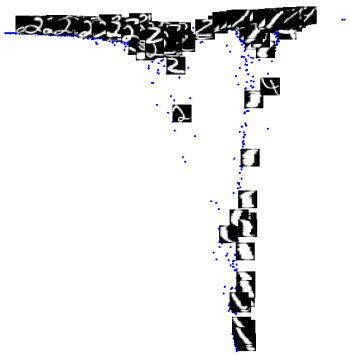


(b) Conformal Map

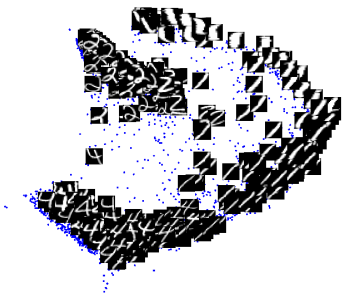
Figure 3: Brendan Frey Face Dataset

Finally, in Figure 4 the dataset consists of 1100 images of hand drawn digits each of *One*, *Two*, and *Four*. Again, LLE can group similar images together but lacks transition and in this case some of the images of *One* are superimposed on the images of *Four*. In contrast, the conformal map can detect the difference between the images with *One* and those

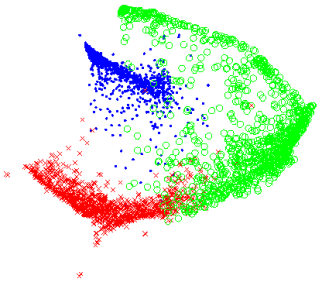
with *Four* since it can create a smooth transition between the two. The plot in the last panel shows that the conformal map algorithm may also be useful as pre-processing tool for classification applications.



(a) LLE



(b) Conformal Map



(c) Conformal Map by Class

Figure 4: Digits 1, 2, 4

## Application of Conformal Map in Classification of 3D Objects

The classification of three dimensional surfaces is an important task in artificial intelligence and computer vision. It can be used for three dimensional object classification and three dimensional face recognition.

In general, the classification of surfaces or 3D objects is a difficult problem because objects can be translated, rotated and scaled. As a result, the classical classifiers, when applied to the raw data, fail to group the objects correctly (Robert Osada and Dobkin 2001). In this section we show how conformal mapping can be used to solve that problem. Five different surfaces (Swiss Roll, S-Curve, Half Sphere,Half Ellipsoid, Cone) will be used in the classification.

The main idea in this algorithm is to classify the transformations which map a 3D object to a conformal 2D map instead of classifying the raw data directly.

Consider a rectangular piece of paper. This paper can be folded into any number of different shapes including the Swiss Roll and the S-Curve. However, the set of actions that are required to fold one shape are different than the set of actions required to fold another. Likewise, if we have a shape that is already folded like a Swiss Roll the actions that will unfold that paper are different from the actions required to unfold the S-Curve. This is the basic intuition that drives this classification algorithm. More precisely we propose to model 3D objects by characterizing the transformations that preserve the invariants they encode.

The transformation that produces the conformal map will be used to classify the objects. An immediate benefit that arises from this approach is that the classification is invariant to scale, position and orientation (intuitively the same actions must be taken to unfold a large figure as a small).

Unfortunately, there is no single transformation that can be applied to the initial three dimensional data points to produce the two dimensional embedding. The conformal map algorithm is actually a two step process beginning with LLE and then adjusting that output to create a conformal map. Both the transformation performed by LLE, and the adjustment through the algorithm presented in this paper, must be captured in order to uniquely identify the full transformation.

The first step is LLE. The LLE algorithm produces a weight matrix $W$ that uniquely defines the transformation. However, the matrix $W$ cannot be explicitly used in classification. $W$ is the matrix that represents the weights of the neighbours for each point. It is of size $m$ *by* $n$ where $m$ is the dimensionality of the output of LLE and $n$ is the number of sampled points. However, the points are not sampled from the 3D surface in any specific order. Therefore, two sets of points drawn from the same surface may have completely different $W$ matrices because the sampling order of the points is different from one set of points to the next. Consider the singular value decomposition of $W = U\Sigma V^T$. Row permutation of $W$ will change $U$ and column permutation will change $V$. However, the diagonal matrix $\Sigma$, which represents the singular values of $W$ is independent of row or column permutations. This implies that if $W_1$ and $W_2$ are equal up to a permutation of rows or columns then their corresponding singular values will be equal. Having equal singular values is necessary but not sufficient to show that $W_1$ and $W_2$ are equal. These singular values will be used to characterize the transformation.

The post processing step of the Conformal Map algorithm produces a transformation matrix $L$. That matrix, and the output from LLE $Y$, are used to compute the final embedding: $Z = LY$. Therefore, this matrix uniquely identifies the entire post processing step. The whole matrix will be used to help characterize the transformation. For a matrix $L$

of size $n_1$ by $n_1$ there are a total of $n_1^2$ values that will be used to identify the post processing step.

When comparing two surfaces, both the singular values of $W$, say there are $n_2$ of them, and the full matrix $L$ will be combined to form a vector. This vector is a point in $n_1^2 + n_2$ dimensional space. To judge similarity between transformations, the Euclidean distance between all pairs of points will be calculated. Points that are close together will be of the same type of 3D surface while the points that are further away will belong to a different 3D surface. 1.8 A simple classification algorithm will be used. The training set will consist of a number of samples from each class which will define a centroid for that class. A centroid is defined as the average of all of the points in a particular class. A point that needs to be classified will be placed in the same class as the nearest centroid measured by Euclidean distance.

## Experimental results for classification

Five synthetic classes of 3D surfaces are considered to test the classification algorithm. In order to create a training set and a test set we repeatedly sampled from these surfaces. Each training and test sample is represented by 1000 points drawn randomly from the equation that represents the surface.

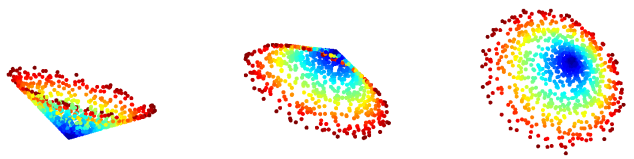
The results presented in this section will show that this classification algorithm can properly distinguish between the five surfaces. They will also show that this algorithm is scale and rotation invariant. In other words, the algorithm will correctly classify an object even if it is much larger or smaller than any of the training surfaces and rotated a random amount around the x, y and z-axis.

The training set is composed of 250 randomly sampled surfaces, 50 from each of the 5 types of surfaces. All of the objects are of the same size and orientation. The training set is then used to create five different centroids, one for each class.

A random sample of 25 surfaces from each class were selected to create a test set. In this case the classification error rates for Swiss Roll, S-Curve, Half Sphere, Half Ellipsoid, and cone are 8%, 12%, 0%, 4%, and 0% respectively.

One of the interesting properties of this algorithm is that it classifies surfaces based on a transformation and so it is invariant to scale. To test this property, the Half Sphere surface is sampled 150 times where the first 50 samples have a radius of 1, the next 50 have a radius of 10 and the last 50 have a radius of 100. The training set of 250 surfaces contains 50 samples of the Half Sphere where the radius was 12, and 50 samples of each of the other shapes. The results show that the classification error rates for Half Spheres with $r = 1$, $r = 10$, and $r = 100$ are 4 %, 4% and 2% respectively.

Another interesting property of this classification algorithm is that it is invariant to the orientation of the surface. To test this idea the cone surface was sampled 50 times and then each sample was rotated a random amount around the x, y, and z axis. Three of those random rotations are shown in Figure 5. Each of the 50 cones was rotated in a different way. The training sample was composed of 50 samples of each type of surface and all surfaces were in their original orientation. Of the 50 cones in the test all but one were classified correctly giving an error rate of 2%.



(a) Orientation 1 (b) Orientation 2 (c) Orientation 3

Figure 5: Cone In Different Orientations

## Conclusion

This paper has introduced a computationally efficient algorithm that produces Conformal Maps. The algorithm's usefulness was demonstrated through both synthetic and real datasets. Finally, this paper has shown that the conformal map algorithm has the potential to classify three dimensional objects.

## References


Belkin, M., and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396.

Burges, C. J. C. 2005. Geometric methods for feature extraction and dimensional reduction.

Gu, X., and Yau, S.-T. 2003. Surface classification using conformal structures. *IEEE* 1:701–708.

Jenkins, O. C., and Mataric, M. J. 2002. Deriving action and behavior primitives from human motion data. *In the IEEE/RSJ Internation Conference on Intelligent Robots and Systems* 2551–2556.

Kulpinski, D. 2002. Lle and isomap analysis of spectra and colour images.

Lutkepohl, H. 1997. Handbook of matrices.

Pless, R., and Simon, I. 2002. Image similarity based image analysis.

Robert Osada, Thomas Funkhouser, B. C., and Dobkin, D. 2001. Matching 3d models with shape distributions. *Shape Modeling International*.

Roweis, S., and Saul, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323–2326.

Sha, F., and Saul, L. K. 2005. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 784–791. New York, NY, USA: ACM.

Vandenberghe, L., and Boyd, S. P. 1996. Semidefinite programming. *SIAM Review* 38(1):49–95.

Wang, S.; Jin, M.; and Gu, X. D. 2007. Conformal geometry and its applications on 3d shape matching, recognition, and stitching. *IEEE Trans. Pattern Anal. Mach. Intell.* 29(7):1209–1220. Member-Yang Wang and Member-Dimitris Samaras.