Mapping Two-Dimensional Plots to a Spherical Surface using Elliptical Grid Mapping

Naoki Kita
Tokyo University of Agriculture and Technology
Tokyo, 184-8588, Japan
nkita@go.tuat.ac.jp

Grégoire Cliquet
L'École de design Nantes Atlantique
Nantes, 44200, France

Kazunori Miyata
Japan Advanced Institute of Science and Technology
Ishikawa, 923-1292, Japan
miyata@jaist.ac.jp

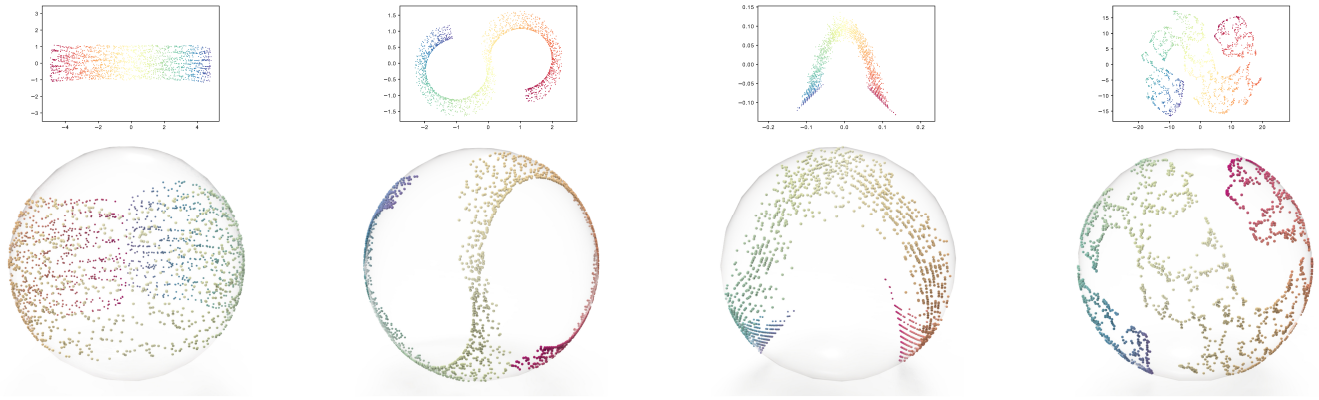
Abstract

In this paper, we present a technique that maps 2D plots to a spherical surface. We project 2D plots as large as possible while minimizing distortions. To accomplish this projection, we employ elliptical grid mapping, which maps points on a cube to those on a sphere. To determine the largest projection from points on the 2D plot to the cube, we find the largest projection inside an unfolded cube pattern. The unfolded cube is then folded to construct a cube, and the projected 2D plot is mapped to a spherical surface. We consider all unfolding patterns of a cube and determine the optimal pattern for the projection. We apply the proposed mapping to high-dimensional data visualizations. Since the technique maps a plot to a spherical surface surrounding the user, and this covers a wider range of the user's field of view, our mapping provides an immersive experience to users.

1. Introduction

Spherical displays are effective mediums for visualizing geospatial data, such as meteorological data or human activities on a globe. In contrast to planar displays, spherical displays can be viewed and sensically interpreted from any direction because of their omnidirectional property. When a user positions at the center of a sphere, he or she can visualize data on the spherical surface surrounding the user in virtual reality (VR) space, which is a kind of immersive VR visualization [31]. In recent years, head-mounted displays (HMDs) such as Oculus Rift, HTC Vive, Sony PlayStation VR, Samsung Gear VR, Google Cardboard, and Daydream View have become easily available and are commonly used in VR games and other applications using HMDs. Although CAVE [5] is also available for immersive display, it requires a large physical space and there are significant costs for maintaining the system. In part for these reasons, HMDs are gaining popularity as the devices of choice for immersive VR environments. When using HMDs, the visualization covers a wider range of the user's field of view (FOV); hence, such a visualization provides a more immersive experience to the user. Although geospatial data can be naturally illustrated on a spherical display, it is not straightforward to display plots defined on a 2D Cartesian coordinate plane on a spherical display. Doing so inevitably introduces distortions from projecting the 2D plots on a spherical surface, and such distortions restrict the potential applications of spherical displays of 2D data. Spherical displays still have uses for 2D data, however; for example, we can visualize various 2D plots such as scatter plots on a spherical surface surrounding a user in an immersive VR environment, which enriches the user's experience. We can also exploit a wider FOV using spherical displays compared to plots that are displayed in a 2D window.

This paper proposes a method for mapping plots defined on a 2D Cartesian coordinate to a spherical surface. Because distortions are inevitably introduced while mapping, we determine a mapping with minimal distortions. Our purpose is to enable displays that cover a wider range of a user's FOV than the 2D plots from which they are constructed by determining a mapping that minimizes distortion when a 2D plot is projected on a sufficiently large spherical surface. To accomplish this, we employ *cube* as an intermediate proxy before mapping the 2D data onto a sphere because we can determine correspondences between points on a cube and those on a sphere analytically via an elliptical grid mapping without parameterization. Thus, we first calculate the largest inscribed projection for a given 2D plot inside an unfolded cube pattern. We then map the plot on



(a) Isomap (b) MDS (c) Laplacian Eigenmaps (d) t-SNE

Figure 1. We propose an elliptical grid mapping technique for projecting 2D plots onto spherical surfaces that are as large as possible with minimal distortion. The images show mapping for 2D plots of dimensionality reduction results of the S-curve dataset including 2000 data points (top) and their spherical mappings (bottom).

the cube constructed from the unfolded pattern to a sphere using the mapping after folding the cube. Another benefit of our method is that cube mapping can be implemented more easily than mappings using other regular polyhedra. The decision to use cube mapping will be further discussed in Sections 4 and 8. Without loss of generality, we focus on scatter plots as the representative example of 2D plots in this paper.

Our contributions can be summarized as follows:

- We propose a method for mapping 2D plots (Cartesian coordinates) to a spherical surface while introducing minimal distortions.
- We solve the largest inscribed projection problem in an unfolded cube pattern.
- We apply the proposed method to high-dimensional data visualization to demonstrate the effectiveness of our method for that application.

2. Related Work

Mappings. In computer graphics, texture mapping is used to mapping images to a 2D shape or the surface of a 3D model. When mapping a texture, texture coordinates are defined on the model. Mesh parameterization defines a mapping between two surfaces with the same topologies, and various parameterization techniques have been proposed [28]. A decaling interface was proposed by Pedersen [23], and other similar approaches have been developed including lapped textures [25], texture sprites [17], and texture mapping based on decals generated using a discrete exponential map approximation [27]. However, decals are small texture patches, and we wish to map plots to large spherical surfaces, so these methods do not serve our purpose. Carlos et al. proposed an area-preserving parametrization for mapping planar rectangles to spherical rectangles using an analytical function [30]. In our case, we also employ an analytical function for our mapping. While Carlos et al. map the unit rectangle to a spherical rectangle, we map 2D plots with arbitrary boundary shapes to a spherical surface. More applicable to our goal are the mappings among spheres, discs [8, 6, 7, 36] and environment mapping [9, 10]. We will briefly describe the mappings from squares to discs and cubes to spheres in Section 4. For a more comprehensive overview, please refer to [14, 36].

Spherical Layouts. Spherical displays are used for information and scientific visualizations. They are especially suitable for the visualization of geospatial data because data such as air flow, distribution of facilities, and earthquakes are already on a globe. Such phenomena are not the only ones well-described by spherical displays. Wu and Takatsuka proposed a method of visualizing multivariate networks using a spherical self-organizing map (GeoSOM) [34]. Kwon et al. studied spherical graph layouts in an immersive VR environment using an HMD [13]. The results of their user study showed that participants performed better using the spherical graph layouts than when they used traditional 2D graph visualizations in an immersive environment, especially for more difficult tasks and larger graphs. In addition to GeoSOM [34], direct spherical embedding techniques have also been proposed [33, 19, 18]. Du et al. proposed the iSphere, focus+context technique to facilitate the exploration of large graphs [4]. They mapped a node-link diagram onto a Riemann sphere and orthogonally projected the sphere onto a 2D plane. Although they visualized graphs (node-link diagrams) using spherical layouts, we can map any 2D plot to a spherical surface. Compared to these techniques, our method requires 2D projection before mapping to a spherical surface; however, our method can be ap-

(a) 2D plot (b) Largest inscribed projection (c) Spherical mapping (d) Immersive visualization

Figure 2. Overview of the proposed technique. Given a 2D plot (a), we first compute the largest inscribed projection inside an unfolded cube pattern (b). Then, the pattern is folded, and the projected 2D plot is on the faces of a cube. Finally, the plot is mapped from the cube to a spherical surface (c). The proposed mapping allows a user with an HMD a more immersive experience when applied to data visualization in a VR environment (d).

plied to inherently planar plots such as existing photographs or drawings.

3. Overview

The overview of the proposed technique is shown in Fig. 2. The input is a plot defined on a 2D Cartesian coordinate plane (Fig. 2(a)). First, we inscribe the plot inside an unfolded cube pattern (Fig. 2(b)), which is then folded to construct a cube. The projected plot is now on the faces of the cube. Finally, the points of the plot that are now on the cube are mapped to points on a sphere (Fig. 2(c)). In Fig. 2(d), we show an application of the mapping for data visualization in an immersive VR environment. The resulting mapped plot covers a wider range of the user's FOV and gives users an immersive experience.

In Section 4, we describe the mapping technique. Section 5 describes the largest inscribed projection problem and its solution. We demonstrate the applications in Section 6, evaluate the mapping in Section 7, and discuss our proposed approach in Section 8, followed by the conclusion and future work in Section 9.

4. Mapping Cube to Sphere

Mapping Square to Circle. In this section, we briefly review mapping techniques used in a 2D case. Namely, to map a point on a square with a side of length 2 (from -1 to 1) to that on a unit circle, one uses the following assignment:

$$\mathbf{v}_{square} = [\, x \;\; y \,]^\top \rightarrow \mathbf{v}_{circle}. \tag{1}$$

The simplest way to map points on a square to those on a circle is to normalize $\mathbf{v}_{square}$:

$$\mathbf{v}_{circle} = \left[\frac{x}{\sqrt{x^2+y^2}} \quad \frac{y}{\sqrt{x^2+y^2}} \right]^\top. \tag{2}$$

Elliptical Grid Mapping. Nowell introduced a mapping that converts horizontal and vertical edges of a square to elliptical arcs inside a circle [22]. The mapping is given by the transformation below:

$$\mathbf{v}_{circle} = \left[x\sqrt{1-\frac{y^2}{2}} \quad y\sqrt{1-\frac{x^2}{2}} \right]^\top. \tag{3}$$

We hereafter refer to Nowell's mapping as *elliptical grid mapping* [7].

Discussion. Lambers provided comparisons of various mapping techniques that map a square to a disk, including the elliptical grid, equal-area, and conformal mapping methods [14]. In any such comparisons, which mapping technique provides a better result to the user depends strongly on the application area. However, since we focus on mapping a 2D plot that is in the interior of a square, elliptical grid mapping is the best available option regardless of the intended application of the projection [14], and therefore, we employ the mapping. In this paper, we do not focus on providing comprehensive comparisons of our choice with alternative mapping techniques, although other methods can certainly be used.

Mapping Cube to Sphere. It is straightforward to extend the elliptical grid mapping to three dimensions, i.e., to map points on a cube to those on a unit sphere [21], rather than points on a square to those on a disk. That is, a point on a cube; $\mathbf{v}_{cube} = [\, x \; y \; z \,]^\top$ is mapped onto a unit sphere $\mathbf{v}_{sphere}$, by the assignment

$$\mathbf{v}_{sphere} = \begin{bmatrix} x\sqrt{1-\frac{y^2}{2}-\frac{z^2}{2}+\frac{y^2z^2}{3}} \\ y\sqrt{1-\frac{x^2}{2}-\frac{z^2}{2}+\frac{x^2z^2}{3}} \\ z\sqrt{1-\frac{x^2}{2}-\frac{y^2}{2}+\frac{x^2y^2}{3}} \end{bmatrix}. \tag{4}$$

For mapping 2D Cartesian coordinates to a spherical surface, we employ a cube mapping-like approach. That is,

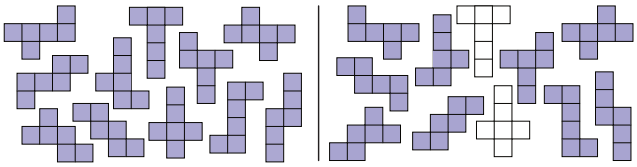


Figure 3. Left: unfolded cubes. Right: flipped patterns. There are twenty patterns in total (excluding the symmetrical patterns with white face color).

given a 2D plot, we first project the plot to the inside of an unfolded cube pattern. The unfolded cube is then folded and an elliptical grid mapping is applied to map points on the cube to a spherical surface. However, there are various ways to unfold a cube, and the unfoldings do not result in identical projections to the sphere. In particular, there are 11 unfolding patterns shown in Fig. 3 (left). By distinguishing two sides of the patterns (Fig. 3 (right)), this gives 20 patterns in total. The shape of the 2D plot we start with determines how it will best fit inside an unfolded 2D cube. To find the unfolding pattern that best inscribes the 2D data, we solve an optimization problem, as is described in the next section.

5. Largest Inscribed Projection

In this section, we describe the means of inscribing a plot of 2D data inside an unfolded cube pattern. Because it is computationally expensive to deal with all data points in the plot, we first compute the concave hull [20] and then use it as a proxy polygon $\mathcal{P}$ for the plot (Fig. 4). We empirically set $k = 60$ for the k-nearest neighbors used in the method described in [20]. If a plot is given as an image (Fig. 5 (left)), we first perform boundary tracking to extract the boundary pixels (Fig. 5 (middle)). Because the boundary may be composed of many pixels, we use the Ramer-Douglas-Peucker algorithm [26, 3] to simplify the polygon (Fig. 5 (right)). In the BUNNY case, although Fig. 5 (middle) contains 2559 vertices, Fig. 5 (right) contains only 72 vertices, which reduces the computational costs associated with checking whether the polygon is contained inside an unfolded cube polygon during the optimization step that follows.

5.1. Problem Formulation

Formally, by denoting the set of the unfolded cube polygons as $\{\mathcal{C}_1, \cdots, \mathcal{C}_{20}\}$, we would like to find the transformation function $\texttt{Proj}(\cdot)$ that projects $\mathcal{P}$ inside an unfolded cube polygon $\mathcal{C}_k$.

$$\begin{aligned} &\underset{s,\mathbf{R},\mathbf{c}}{\text{maximize}} \quad s \\ &\text{subject to} \quad \texttt{Proj}(\mathcal{P}) \text{ is inside } \mathcal{C}_k, \end{aligned} \tag{5}$$

where we decompose the transformation function $\texttt{Proj}(\cdot)$

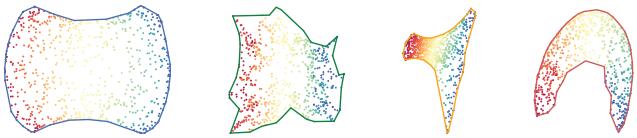


Figure 4. Concave hulls for various plots.

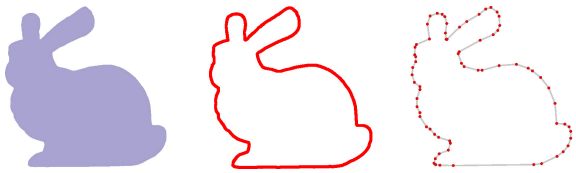


Figure 5. Left:input plot as an image. Middle: boundary tracking. Right: simplified boundary polygon.

to a scaling factor $s \in \mathbb{R}$, rotation matrix $\mathbf{R} \in SO(2)$, and displaced centroid of $\mathcal{P}$, $\mathbf{c} \in \mathbb{R}^2$.

5.2. Optimization

We employ particle swarm optimization (PSO) [24] to solve Equation (5), which solves a problem using a population of candidate solutions that are represented as particles. The candidates have positions and velocities, and move around the high-dimensional search space to determine the optimal solution. The positions and velocities are updated for each iteration by calculating the following equations:

$$\begin{aligned} &\mathbf{v}_i(t+1) \leftarrow w\mathbf{v}_i(t) + c_1 r_1(\mathbf{x}^{pbest} - \mathbf{x}_i(t)) + c_2 r_2(\mathbf{x}^{gbest} - \mathbf{x}_i(t)), \\ &\mathbf{x}_i(t+1) \leftarrow \mathbf{x}_i(t) + \mathbf{v}_i(t), \end{aligned} \tag{6}$$

where the position of the i-th particle at time t is represented as $\mathbf{x}_i \in \mathbb{R}^n$ and the velocity is represented as $\mathbf{v}_i \in \mathbb{R}^n$. The variables r_1 and $r_2 \in [0, 1)$ are random values drawn uniformly from $[0, 1)$, and w, c_1, and c_2 are the control parameters. The position $\mathbf{x}^{pbest}$ is the current best position of the particle, and $\mathbf{x}^{gbest}$ is the current best position of all particles. The best positions are determined by calculating an objective function. Using these local and global parameters, the particles communicate with one another and explore possible better solutions. We automatically determine parameters w, c_1, and c_2 by employing a constriction coefficient [2]. In our case, $\mathbf{x} = [s, \mathbf{R}(\theta), \mathbf{c}]$ and the objective function is defined as follows:

$$\texttt{Objective}(\mathbf{x}) = \begin{cases} s & \texttt{Proj}(\mathcal{P}) \text{ is inside } \mathcal{C}_k \\ -\infty & \text{otherwise.} \end{cases} \tag{7}$$

Jacobson employed a hybrid PSO by combining PSO with binary search to solve a problem similar to ours [12]. In our case, however, we found empirically that the hybrid PSO with binary search is computationally more expensive than PSO without binary search because it checks whether $\texttt{Proj}(\mathcal{P})$ is inside $\mathcal{C}_k$ several times. Without the hybridiza-

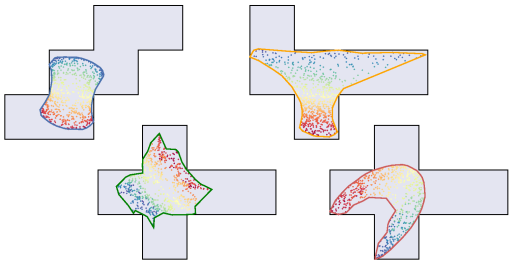


Figure 6. Largest inscribed projections for Fig 4.

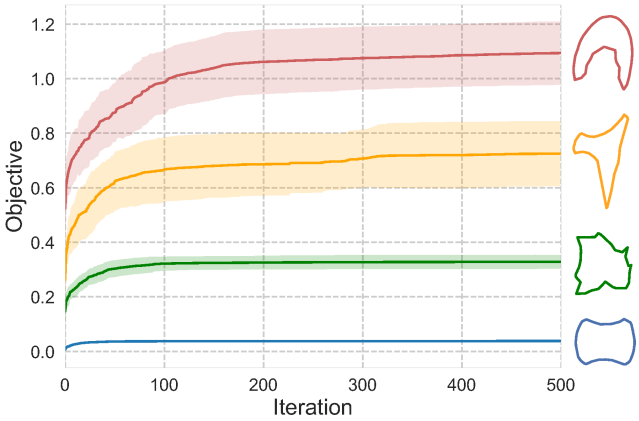


Figure 7. Developments of the mean objective values and standard deviations over iterations. $\#\texttt{particle} = 500$, $\#\texttt{iteration} = 500$.

tion, we can obtain approximately optimal results within hundreds of iterations $(200 - 500)$ in most cases.

5.3. Optimization Results and Discussions

Figure 6 shows the largest inscribed projections found by our method for each of the polygons in Fig. 4. Depending on the shape of the hull polygons, the most appropriate unfolded cube pattern is chosen for the projection.

Optimization Developments. We measure the objective value of each unfolded cube pattern and shows the developments in the mean objective values and standard deviations over iterations in Fig. 7. We set PSO parameters $\#\texttt{particle} = 500$ and $\#\texttt{iteration} = 500$. The curve for each plot shows how fast the optimization converges. We can observe that the blue plot converges faster than the others, and that the red and orange plots converge especially slowly. It seems that the convergence depends on the convexity of the concave hull. Here, the convexity of a polygon can be measured using the ratio of the area of the concave hull and its convex hull. The standard deviations in each plot show the difficulty in finding the optimal values among the unfolded cube patterns because the optimal values are different for each pattern. It also seems that variations in the standard deviation depend on the convexity of

Table 1. Timings. $\#\texttt{particle} = 500$, $\#\texttt{iteration} = 500$.

Hull Polygon				
#vertices	41	30	33	60
Avg. Time	0.796 s	0.510 s	0.510 s	0.875 s

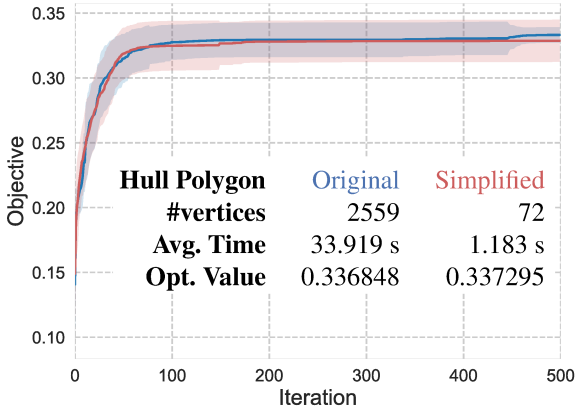


Figure 8. Optimization statistics for BUNNY (Fig. 5).

the polygon as well as the convergence speed.

Here, we have to note that, the above findings are inferences based on the experiments. We have to conduct more dedicated experiments for drawing any conclusions.

Timing. We implement the optimization method in C++, and measure the computation time using an Intel Core i5 @ 2.9 GHz personal computer with 16 GB RAM. Table 1 shows the average time required for the optimization per unfolded cube pattern for each polygon in Fig. 4. It seems that the optimization time depends linearly on the number of vertices of the polygon. Figure 8 shows the optimization statistics for the 2D BUNNY plot in Fig. 5. We compare the original polygon extracted using boundary tracking (Original) and its simplified polygon (Simplified). As we can see, although the obtained optimal scaling values and the developments of the objective values are almost the same, the original polygon required approximately 30 times more time to optimize than the simplified one.

Comparison between Patterns. Figure 9 shows the results of optimization for 2D LUCY polygons. In this example, we inscribe a LUCY polygon inside an unfolded cube polygon. Although the LUCY polygons are projected with maximum scaling in both cases, the right one occupies approximately twice as much area in the unfolded cube as the left one. In the case of immersive spherical display applications where a user stands at the center of a surrounding spherical display, the right projection gives a more immer-

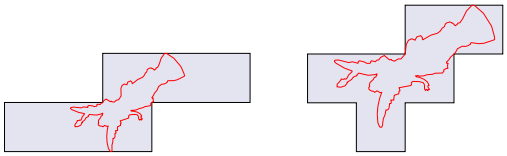


Figure 9. Largest inscribed projections for LUCY polygons.

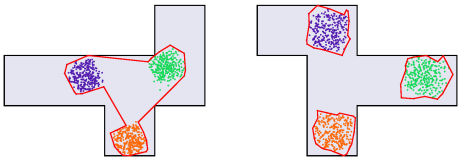


Figure 10. The largest inscribed projections for a single concave hull (left) and multiple concave hulls (right).

sive projection because the projection covers a wider FOV of the user.

Approximating with Multiple Concave Hulls. In the case of a plot composed of clusters that are far apart from one another, a single concave hull does not approximate the plot well (Fig. 10 (left)). In such cases, we first compute k-means clustering, then construct concave hulls for each cluster, and finally, calculate the largest inscribed projection while maintaining the center of the plot in order to retain the relative positional relations (Fig. 10 (right)). In the current implementation, we manually select k depending on input plots. As can be seen in Fig. 10, the projection with multiple concave hulls occupies a larger area than that of the single hull.

5.4. Mapping Cube Patterns to a Spherical Surface

Because we have 20 unfolded cube patterns, we define mappings for each pattern to a spherical surface. That is, we define the LEFT ($-X$), RIGHT ($+X$), UP ($+Y$), DOWN ($-Y$), FRONT ($+Z$), and BACK ($-Z$) directions for each face of an unfolded cube pattern. Then, a coordinate defined on the face is mapped to a corresponding part of the spherical surface using elliptical grid mapping. Figure 11 shows the mapping results of Fig. 6.

6. Applications

6.1. Visualizing High-Dimensional Data

S-curve. High-dimensional data are difficult to understand. Therefore, dimensionality reduction techniques are used to project the data to two or three dimensions. We apply our proposed technique to the visualization of high-dimensional data. We use the S-curve dataset, where the S-curve is a 2D manifold embedded in 3D space. We generate 2000 sample points, and then apply dimensionality reduction techniques to project the data to 2D space. Figure 1 (top) shows the results of Isomap [29], multidimensional scaling (MDS) [16], Laplacian Eigenmaps [1], and t-SNE [32]. We then apply our technique to 2D plots, and the resulting spherical plots are shown in Fig. 1 (bottom).

MNIST. We also visualize the MNIST dataset [15]. MNIST consists of images of handwritten digits. For the visualization, we randomly sample 2500 images and apply t-SNE to project 784-dimensional feature vectors to 2D space (Fig. 12(a)). Here, we empirically determine the number of samples i.e., 2500 samples since it is adequate for demonstrating our visualization. We then apply our technique to map the 2D plot to a spherical surface. To compare the 2D planar plot and the spherical plot, we visualize the plots in an immersive VR environment (Fig. 12(b-c)), where a user is positioned at the center of a unit sphere. Figure 12(b) shows the 2D planar plot. The digit images are used as the texture of a plane representing the data points. The centroid of the plot is at the point of contact between the plane and the sphere. Although we can adjust the perceived distance between the user and the plot and the scaling, it is difficult to locate data points in the surroundings. In contrast, because our spherical mapping maps the data on a unit sphere, the distance between the user and any data point is constant (Fig. 12(c)). Figure 12(c) shows planes constructed to represent the data point billboards, namely those in which all digit images always face the user. These help a user to correctly interpret the size of the digits, which is important if the data points are encoded in variably sized objects. Although we can walk among visualized data in a VR Cartesian 3D space, data visualization in a Cartesian 3D space can occlude data behind another data because of depth. In contrast, our spherical display visualizes data on a sphere that holds images at a constant distant from the user, and therefore, cannot occlude data due to depth.

7. Evaluation

7.1. Comparison of Different Mapping Techniques

Qualitative Comparison. We compare different mapping techniques shown in Fig. 13, and find that cylindrical mapping stretches and distorts the mesh/texture, especially in regions close to the poles (Fig. 13(b)). We also observe that normalization-based mapping distorts the mesh/textures (arms and legs in Fig. 13(c)). In contrast, although some distortions are observed after our mapping, it preserves the mesh better than other techniques.

User Study. We conduct a user study to perceptually compare the mapping techniques. We develop an immersive VR data visualization demo for displaying *S-curve* data visualized by t-SNE (Fig. 1(d)) using three mapping techniques: cylindrical, normalization-based (norm-based), and

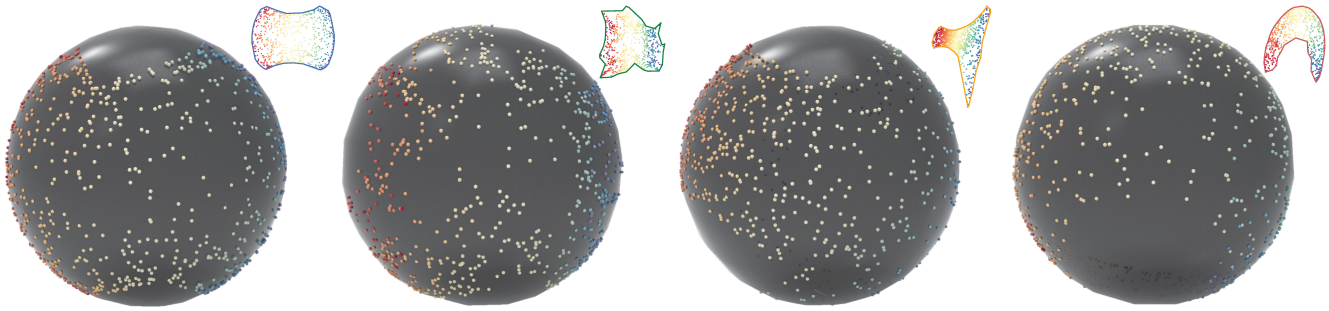
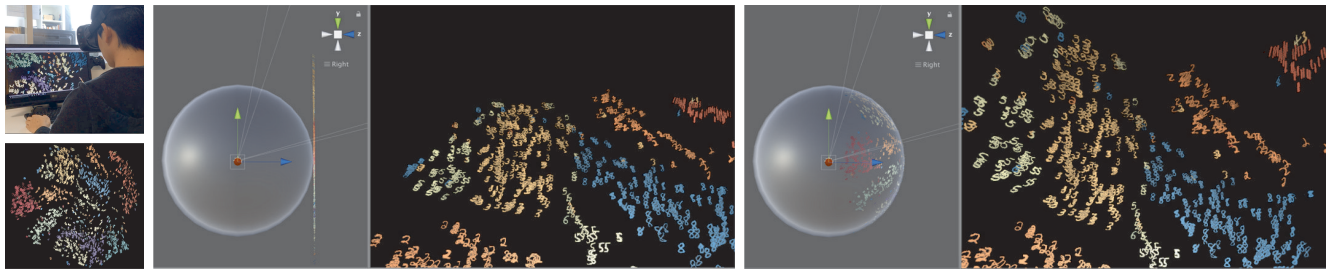


Figure 11. Results of spherical mapping for the projections listed in Fig. 6.



(a) 2D plot (b) 2D plot in VR environment (c) Our spherical mapping in VR environment

Figure 12. Visualization of the MNIST dataset in a VR environment. (a) Visualization of 2500 digit images by t-SNE. (b) Viewing of (a) in a VR environment. (c) Viewing of (a) mapped on a spherical surface by the proposed method.

(a) Original (b) Cylindrical (c) Normalization-based (d) Ours

Figure 13. Comparison of mapping techniques.

the mapping that is the subject of this paper. We ask the participants to answer the following questions:

Q1 *Which mapping is more immersive?*

Q2 *Which mapping preserves the inter-element distances and neighbors well?*

In the demos, we display the original 2D plot and the spherical plots generated by the three mapping techniques. A participant can switch the plots at any time to compare them. We first explain each participant the ways in which they can

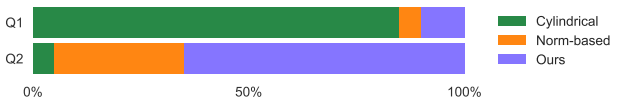


Figure 14. User study results.

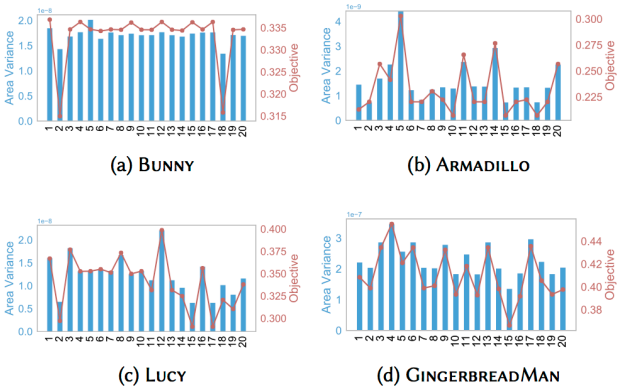
interact with the plots. Each user is then asked to sit in front of a desk wearing an Oculus Rift DK2 HMD. Because the spherical layouts do not require positional tracking, we only use the head tracking. We include 20 participants (twelve males and eight females) with normal color vision in the user study. The age of the participants ranges from 22 to 54. The mean age is 24.65 and the median is 22 years old (standard deviation: 7.15). Most of them are graduate students at a design school.

The results of the user survey are shown in Fig. 14. Most of the participants answer that *cylindrical* mapping is the most immersive in *Q1*. We believe this is because the mapping result of cylindrical mapping covers almost the full range of the hemisphere, whereas the norm-based map and our mapping cover narrower FOVs. In *Q2*, our mapping technique outperforms other mapping techniques by best preserving the characteristics of the original 2D plot. This is an important property for a mapping technique when it is applied to data visualization applications as it helps to avoid misleading users and generating invalid interpretations. Although the *cylindrical* mapping technique heavily distorts the original plot, *norm-based mapping* and *ours* cause less distortion and better preserve the inter-element distance and neighbors. This makes it more difficult to distinguish between the two mapping techniques. Nevertheless, according to the result of *Q2*, the participants regard *ours* to be more distance-preserving than the *norm-based* mapping technique, which shows that *ours* outperforms *norm-based* mapping in the aspects measured.

Quantitative Comparison. Rather than relying exclusively on user reports, we compare the distortions quantitatively. We measure the area variances of the triangles of the 2D polygonal mesh of BUNNY, ARMADILLO, LUCY, and GINGERBREADMAN. The area of a triangle on a unit sphere is equal to the spherical excess of the triangle. The original 2D meshes are generated using capacity-constrained Delaunay triangulation [35] to ensure that the triangles of the meshes have a minimal capacity variance. The result is shown in Table 2. Cylindrical mapping has a larger area variance than the others, as is especially pronounced in the LUCY example. Our mapping has approximately ten times less area variance than normalization-based mapping.

Table 2. Area variance comparison of different mapping techniques.

	Area Variance		
Polygon	Cylindrical	Norm-based	Ours
BUNNY	9.28×10^{-7}	4.08×10^{-7}	1.77×10^{-8}
ARMADILLO	2.37×10^{-8}	2.83×10^{-8}	1.45×10^{-9}
LUCY	1.63×10^{-2}	1.72×10^{-7}	1.60×10^{-8}
GINGERBREADMAN	6.26×10^{-6}	5.39×10^{-6}	3.45×10^{-7}



(a) BUNNY (b) ARMADILLO (c) LUCY (d) GINGERBREADMAN

Figure 15. Area variance and objective value for each cube pattern. x-labels represent cube pattern indices.

7.2. Comparison between Unfolded Cube Patterns

We also compare the area variances (less is better) and objective values (larger is better) of the proposed mapping for each cube pattern (Fig. 15). Since BUNNY is more convex than the others, the graph is different from those of the others. In contrast, the graphs of ARMADILLO, LUCY, and GINGERBREADMAN show that the area variances and objective values vary depending on the cube pattern indices. In all these cases, the graphs show that there is a tradeoff between area variance and objective value. Although we select the one with the largest objective value in our current implementation, we can select the one with the least area variance or the maximum balance for other applications, as appropriate.

8. Discussion

We have proposed a two-step approach for mapping 2D planar plots to a spherical surface. First, we project a 2D plot to an unfolded *cube* pattern. Then, the projected plot is mapped to a spherical surface using *elliptical grid mapping*. We could have used another regular polyhedra (e.g., octahedron or icosahedron) as an alternative for a *cube*, and such a choice has the potential to reduce distortions when the polyhedral is mapped to a spherical surface. However, we employ *cube* since we can find a correspondence between a point on a cube to that on a sphere analytically via *elliptical grid mapping*. We do not have to pre-compute parameterizations to define a mapping from a 2D domain to a unit

sphere. In addition, the number of comparisons needed for optimization of the initial projection into the unfolded polyhedral increases significantly when we move from the cube to another polyhedral: while a cube has 11 essentially different unfoldings, a dodecahedron and an icosahedron have 43380 essentially different unfoldings [11]. If we can determine the most appropriate unfolding pattern and define a mapping between the polyhedra to a unit sphere, it would be a better alternative to the proposed approach, but it is computationally very expensive. On the other side, since an octahedron has 11 essentially different unfoldings, it might be a reasonable alternative to our choice (cube).

As we discussed in Section 4, we compare the three selected mapping techniques, but this is not a comprehensive comparison between our method and every possible well-suited mapping technique, as such choices depend on the area of application. As such, it is difficult to provide a general solution. Although we focus on mapping a 2D plot to a spherical surface as large as possible while minimizing distortions, trying to use as much of the sphere as possible seems to be unhelpful since our FOV is limited to 120° in the horizontal direction with less in the vertical direction. Rather than replacing 2D planar displays with spherical displays for data visualization and exploration, we aim to combine a 2D window and a spherical layout for data exploration. For example, we can use a 2D window to overview the data, and then dive into an immersive visualization using the proposed spherical mapping.

9. Conclusion and Future Work

This paper proposes a mapping technique that maps 2D plots to a spherical surface. To minimize distortions, we employ elliptical grid mapping to map points on a cube to a spherical surface. To fill as much of a user's FOV as possible, we solve the largest inscribed projection problem for a polygon approximating a given plot and an unfolded cube pattern. We apply the proposed technique to a high-dimensional data visualization problem.

Constructing a conformal map between a sphere and a cube does not seems to be simple, however, if such mapping would be constructed, it can derive a better result than that in the proposed method in terms of distortion, and is more suited for immersive data visualization applications.

Limitation and Future Work. As reported in Section 5.3, solving the largest inscribed projection problem is not a computationally demanding task, but may not be suited for realtime mapping. In that case, direct mapping can be better regarding the computational performance.

In addition, although we quantitatively evaluate distortions in Section 7, the distortions are noticeable if we apply the mapping technique for textures (Fig. 13 (bottom)).

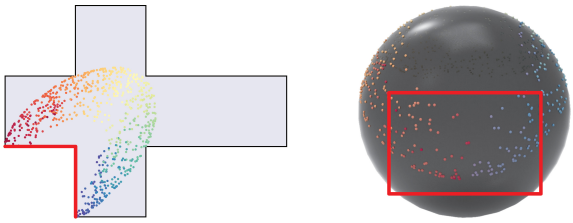


Figure 16. Since the red edges of the unfolded cube pattern shown in the left figure are merged when constructing a cube, the mapping places both ends of the plot close (see the area enclosed by the red rectangle in the right figure).

Therefore, further optimization is necessary when one is applying the technique described here to such applications. Our method also may not preserve neighbors for points close to the edges of the cube because we first project a 2D plot to an unfolded cube pattern, fold it to construct a cube, and then map points on the cube to a spherical surface. For example, in Fig. 16 and Fig. 1(a), while both ends of the plot are far apart in the 2D plot, they are close together in the spherical plots. Therefore, we must take care not to mislead viewers if applying the technique for data visualization. To address such problems, we can scale the plot slightly smaller to avoid edges. Because mapping distortions are inevitable, we would like to further optimize the mapping, and conduct a task-based user study to measure the performance of our method in applications with sensitivity to this feature.

Acknowledgements

We thank all reviewers for their helpful comments. This work was supported by JSPS KAKENHI grant number 19K24338.

Appendix

2D Polygonal Meshes used for Quantitative Evaluation

Figure 17. Input 2D polygonal meshes for Table 2 and Fig. 15.

References

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

[2] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, Feb 2002. 4

[3] D. H. Douglas and T. K. Peucker. Algorithms for the Reduction of the Number of Points Required To Represent a Digitized Line or Its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. 4

[4] F. Du, N. Cao, Y.-R. Lin, P. Xu, and H. Tong. isphere: Focus+context sphere visualization for interactive large graph exploration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 2916–2927, New York, NY, USA, 2017. ACM. 2

[5] A. Febretti, A. Nishimoto, T. Thigpen, J. Talandis, L. Long, J. D. Pirtle, T. Peterka, A. Verlo, M. Brown, D. Plepys, D. Sandin, L. Renambot, A. Johnson, and J. Leigh. Cave2: a hybrid reality environment for immersive simulation and information analysis, 2013. 1

[6] C. Fong. An indoor alternative to stereographic spherical panoramas. In G. H. Gary Greenfield and R. Sarhangi, editors, *Proceedings of Bridges 2014: Mathematics, Music, Art, Architecture, Culture*, pages 103–110, Phoenix, Arizona, 2014. Tessellations Publishing. 2

[7] C. Fong. Analytical Methods for Squaring the Disc. *ArXiv e-prints*, Sept. 2015. 2, 3

[8] D. M. German, P. D'Angelo, M. Gross, and B. Postle. New methods to project panoramas for practical and aesthetic purposes. In *Proceedings of the Third Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics'07, pages 15–22, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 2

[9] N. Greene. Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, 6(11):21–29, Nov. 1986. 2

[10] W. Heidrich and H.-P. Seidel. View-independent environment maps. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, HWWS '98, pages 39–ff., New York, NY, USA, 1998. ACM. 2

[11] T. Horiyama and W. Shoji. The number of different unfoldings of polyhedra. In L. Cai, S.-W. Cheng, and T.-W. Lam, editors, *Algorithms and Computation*, pages 623–633, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 9

[12] A. Jacobson. Generalized matryoshka: Computational design of nesting objects. *Computer Graphics Forum*, 2017. 4

[13] O. H. Kwon, C. Muelder, K. Lee, and K. L. Ma. A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(7):1802–1815, July 2016. 2

[14] M. Lambers. Mappings between sphere, disc, and square. *Journal of Computer Graphics Techniques (JCGT)*, 5(2):1–21, April 2016. 2, 3

[15] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. 6

[16] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer Publishing Company, Incorporated, 1st edition, 2007. 6

[17] S. Lefebvre, S. Hornus, and F. Neyret. Texture sprites: Texture elements splatted on surfaces. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, I3D '05, pages 163–170, New York, NY, USA, 2005. ACM. 2

[18] Y. Lu, Z. Yang, and J. Corander. Doubly stochastic neighbor embedding on spheres. *CoRR*, abs/1609.01977, 2016. 2

[19] D. Lunga and O. Ersoy. Spherical stochastic neighbor embedding of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 51(2):857–871, Feb 2013. 2

[20] A. J. C. Moreira and M. Y. Santos. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of point. In J. Braz, P.-P. Vázquez, and J. M. Pereira, editors, *GRAPP (GM/R)*, pages 61–68. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007. 4

[21] P. Nowell. Mapping a cube to a sphere, 2005. http://mathproofs.blogspot.com/2005/07/mapping-cube-to-sphere.html Accessed: 2019-09-12. 3

[22] P. Nowell. Mapping a square to a circle, 2005. http://mathproofs.blogspot.com/2005/07/mapping-square-to-circle.html Accessed: 2019-09-12. 3

[23] H. K. Pedersen. A framework for interactive texturing on curved surfaces. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 295–302, New York, NY, USA, 1996. ACM. 2

[24] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, Jun 2007. 4

[25] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 465–470, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 2

[26] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244 – 256, 1972. 4

[27] R. Schmidt, C. Grimm, and B. Wyvill. Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph.*, 25(3):605–613, July 2006. 2

[28] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, Jan. 2006. 2

[29] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000. 6

[30] C. Ureña, M. Fajardo, and A. King. An area-preserving parametrization for spherical rectangles. *Computer Graphics Forum*, 32(4):59–66. 2

[31] A. van Dam, A. S. Forsberg, D. H. Laidlaw, J. J. LaViola, and R. M. Simpson. Immersive vr for scientific visualization: a progress report. *IEEE Computer Graphics and Applications*, 20(6):26–52, Nov 2000. 1

[32] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 6

[33] R. C. Wilson, E. R. Hancock, E. Pękalska, and R. P. W. Duin. Spherical embeddings for non-euclidean dissimilarities. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1903–1910, June 2010. 2

[34] Y. Wu and M. Takatsuka. Visualizing multivariate network on the surface of a sphere. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, APVis '06, pages 77–83, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. 2

[35] Y. Xu, L. Liu, C. Gotsman, and S. J. Gortler. Capacity-constrained delaunay triangulation for point distributions. *Computers & Graphics*, 35(3):510 – 516, 2011. Shape Modeling International (SMI) Conference 2011. 8

[36] M. Zucker and Y. Higashi. Cube-to-sphere projections for procedural texturing and beyond. *Journal of Computer Graphics Techniques (JCGT)*, 7(2):1–22, June 2018. 2