Visualizing Riemannian data with Rie-SNE

Andri Bergsson
PayAnalytics (payanalytics.com)
Reykjavík, Iceland
and.bergsson@gmail.com

Søren Hauberg
Technical University of Denmark
Kgs. Lyngby, Denmark
sohau@dtu.dk

***Abstract*—Faithful visualizations of data residing on manifolds must take the underlying geometry into account when producing a flat planar view of the data. In this paper, we extend the classic *stochastic neighbor embedding (SNE)* algorithm to data on general Riemannian manifolds. We replace standard Gaussian assumptions with Riemannian diffusion counterparts and propose an efficient approximation that only requires access to calculations of Riemannian distances and volumes. We demonstrate that the approach also allows for mapping data from one manifold to another, e.g. from a high-dimensional sphere to a low-dimensional one.**

***Index Terms*—Directional statistics, visualization, machine learning, differential geometry.**

I. Introduction

Visualizations are crucial to investigators trying to make sense of high-dimensional data. The most common output of a visualization is a two-dimensional plot (e.g. on a piece of paper or a computer screen), so we often call on a form of dimensionality reduction when working with high-dimensional data. The vast majority of dimensionality reduction techniques assume that data resides on a Euclidean domain (Sec. II-A), which presents a problem when data is not quite that simple. Data residing on Riemannian manifolds, such as the sphere, appear in many domains where either known constraints or other modeling assumptions impose a Riemannian structure (Sec. II-B). In such settings, how should one visualize data?

There are many concerns and questions when visualizing Riemannian data. The first is generic: all dimensionality reduction tools amplify parts of the signal, while reducing the remainder. This is an inherent limitation, which should always be in mind when interpreting data visualizations. Since some loss of information is inevitable, should we then loosen our grip on the data or its underlying Riemannian structure when such is present? Gauss's *Theorema Egregium* [1] informs us that if the final plot is to be presented on a *flat* screen or piece of paper, then a distortion of the Riemannian structure is inevitable.

In practice, even if one accepts the limitations of a visualization, actual algorithms for visualizing Riemannian data are missing. In this paper, we develop an extension of the *Stochastic Neighbor Embedding* [2] method to Riemannian data and thereby provide one such tool. We call this *Riemannian Stochastic Neighbor Embedding*, or *Rie-SNE* for short. Our approach is quite general as it allows for embedding data observed on one Riemannian manifold to be embedded on another. This allows for mapping data from a Riemannian space to a two-dimensional Euclidean plane (for plotting), but also mapping to a two-dimensional sphere, or similar, when the Euclidean topology is inappropriate. Rie-SNE does not claim to solve the above-mentioned limitations of visualization, but it does provide a working tool, which we demonstrate to have practical merit.

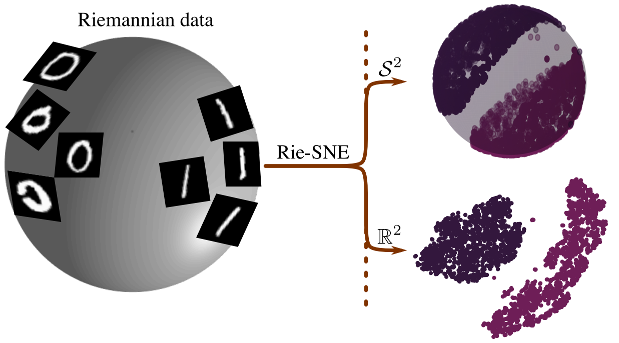

Fig. 1. Rie-SNE visualizes high-dimensional Riemannian data by mapping to a low-dimensional Riemannian (or Euclidean) manifold, which can then be shown. The figure shows high-dimensional spherical data mapped to either a low-dimensional sphere or a low-dimensional plane. The method also supports other manifolds, both as input and output.

II. Background and related work

Before describing Rie-SNE, we provide the relevant context on visualization and geometry.

A. Euclidean visualization

General data visualization is a vast topic, and a complete review is beyond our scope [3]. We here focus on the setting where data is represented as vectors (points) in a Euclidean space of high dimension. When data is two- or three-dimensional a scatter plot can directly reveal its structure, and we focus on the more difficult setting where data dimensions vastly exceed the easily plottable. Here one may explore the data through multiple projections, such as pairwise scatter plots, which is usually manageable for data of up to around 10 dimensions. Eventually the approach tends to become unwieldy and the greater picture is lost. Alternatives include continuously interpolating between dimensions to provide an (interactive) animation [4].

The approach we here explore is to find a non-linear mapping from the high-dimensional observation space into a two- or three-dimensional space, which is suitable for plotting. Many variants of this approach exists, and we only touch upon a few. *Principal component analysis (PCA)* [5] is perhaps the most commonly used approach to dimensionality reduction. This seeks a low-dimensional representation of data that preserves as much variance as possible. The restriction to spanning a linear subspace of the observation space, however, often implies that the low-dimensional view reveal little structure. The *Gaussian process latent variable model (GP-LVM)* [6] provides a nonlinear probabilistic extension of PCA that places a Gaussian process prior on the unknown mapping that reduce dimensionality, and marginalize this accordingly. The approach carries intrinsic elegance, but its optimization can be brittle [7]. Classic 'manifold learning' techniques avoid optimization issues by phrasing optimization tasks, where an optimum is available through spectral decompositions [8]–[10]. These rely on constructions of neighborhood graphs, which, however, can be brittle, so in practice the resulting visualizations are sensitive to parameter choices. The *stochastic neighbor embedding (SNE)* [2] replaces the 'hard' graph construction with a softer construction. A modern variant of this method [11] is one of the currently most popular algorithms, and is the one we here extend. We cover this in Secs. II-D and II-E.

B. Riemannian data

Data is often equipped with additional knowledge such as constraints or given smooth structures. In many scenarios this additional knowledge gives the data a Riemannian structure. For example, knowing that all observations have unit norm, places them on the unit sphere, which has a well-studied Riemannian structure. The a priori available knowledge giving rise to a Riemannian data interpretation differs between domains, and we here only name a few. The most prominent example is that of *directional statistics* [12]–[14] where data resides on the unit sphere. Other examples include *shape data* [15]–[19], *DTI images* [20]–[22], *image features* [23]–[25], *motion models* [26], [27], *human poses* [28]–[31], *robotics* [32], [33], *social networks* [34] and more.

Even if Riemannian data is becoming increasingly common, tools for visualization have not followed. The most common approach for visualizing Riemannian data is to locate a point on the manifold, e.g. the intrinsic mean [35], and map the data to the tangent space of this point. That gives a Euclidean view of the data, which can then be visualized with one of the many methods for this domain (Sec. II-A). In particular, applying PCA tangentially is the gold standard [36]. Unless data is concentrated around the point of tangency this approach is bound to give a highly distorted view of the data, and in practice most knowledge reflected in the geometry will be lost by the linearization [37]. Several extensions of PCA to Riemannian manifolds exist, e.g. [38]–[40]. These focus on generalizing the classic linear models, and less work has been done on extending nonlinear methods. Two notable extensions are a 'wrapped' extension of the GP-LVM [41], and an extension of the classic principal curves model [42].

C. Measuring on Riemannian manifolds

In order to engage with data residing on a manifold, we need a collection operators that can be applied to data. Here we only cover the most basic as that is all required by Rie-SNE. A detailed exposition can be found elsewhere [35], [43].

A Riemannian manifold is a space which is locally Euclidean. This imply that in a neighborhood around a point $\boldsymbol{\mu}$ on the manifold, we can get a Euclidean view of the manifold in the form of a tangent space, which is equipped with an inner product,

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\boldsymbol{\mu}} = \mathbf{x}_i^\top G_{\boldsymbol{\mu}} \mathbf{x}_j, \tag{1}$$

where $G_{\boldsymbol{\mu}}$ is a symmetric positive definite matrix that reflects the inner product at $\boldsymbol{\mu}$. This inner product is allowed to change smoothly between tangent spaces, in order to compensates for the approximation error induced by linear view of the manifold. This distortion can be characterized by the change-in-volume between the manifold and its tangent, which follows $\sqrt{\det G_{\boldsymbol{\mu}}}$ [35].

The inner product allow us to define local distances, which can be integrated to provide a notion of *curve length*. That is, given a curve $\mathbf{c}$ on a manifold, we may compute its length as

$$\text{Length}[\mathbf{c}] = \int \sqrt{\langle \dot{\mathbf{c}}_t, \dot{\mathbf{c}}_t \rangle_{\mathbf{c}_t}} \mathrm{d}t, \tag{2}$$

where we assume the curve to be parametrized by $t \in [0, 1]$, and use $\mathbf{c}_t$ and $\dot{\mathbf{c}}_t$ to denote the position and velocity of the curve, respectively. From the notion of a curve length, it is trivial to define the distance between two points as the length of the shortest curve,

$$\text{dist} = \min_{\mathbf{c}} \text{Length}[\mathbf{c}]. \tag{3}$$

The shortest curve commonly goes under the name *geodesic*. The distance function can be differentiated using the relation

$$\partial_{\mathbf{x}_i} \text{dist}^2(\mathbf{x}_i, \mathbf{x}_j) = 2\text{Log}_{\mathbf{x}_i}(\mathbf{x}_j), \tag{4}$$

where $\text{Log}_{\mathbf{x}_i}(\mathbf{x}_j)$ is the Riemannian *logarithm map* [35]. If $\mathbf{c}$ is a constant-speed geodesic connecting $\mathbf{x}_i$ and $\mathbf{x}_j$, then the logarithm map is merely the initial velocity $\dot{\mathbf{c}}_0$ of said curve.

D. Stochastic neighbor embedding

Stochastic neighbor embedding (SNE) [2] is a dimensionality reduction tool, which aims to preserve similarity between neighboring points when mapped to a low-dimensional representation. Assume for now, that we have access to function s_{high} and s_{low}, which measure the similarity between observation pairs in the high-dimensional observation space and the low-dimensional representation space, respectively. Now define the conditional probability, $p_{j|i}$ that $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as its neighbor [11]

$$p_{j|i} = \frac{s_{\text{high}}(\mathbf{x}_j | \mathbf{x}_i)}{\sum_{k \neq i} s_{\text{high}}(\mathbf{x}_k | \mathbf{x}_i)}. \tag{5}$$

Common convention is to define $pi|i = 0$. Further note that $\sum_j p_{j|i} = 1$. We can renormalize this to form a distribution over all observations as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}, \tag{6}$$

where n is the number of observations.

To learn a low-dimensional representation, the key idea is to repeat the above over the low-dimensional space to form

$$q_{j|i} = \frac{s_{\text{low}}(\mathbf{y}_j|\mathbf{y}_i)}{\sum_{k\neq i} s_{\text{low}}(\mathbf{y}_k|\mathbf{y}_i)}, \tag{7}$$

$$q_{ij} = \frac{q_{j|i} + q_{i|j}}{2n}. \tag{8}$$

We can now compare the similarity of our data and the representation by computing the Kullback-Leibler divergence between p_{ij} and q_{ij},

$$C = \text{KL}\left(P||Q\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{9}$$

This can then be minimized using gradient descent with respect to the low-dimensional representation $\{\mathbf{y}_i\}_{i=1}^n$.

In its classic form, SNE picks the measures of similarity as Gaussian functions

$$\begin{aligned} s_{\text{high}}(\mathbf{x}_j|\mathbf{x}_i) &= \left(2\pi\sigma_i^2\right)^{-D/2} \exp\left(-\frac{\|\mathbf{x}_j - \mathbf{x}_i\|^2}{2\sigma_i^2}\right), \\ s_{\text{low}}(\mathbf{y}_j|\mathbf{y}_i) &= (2\pi)^{-d/2} \exp\left(-\frac{\|\mathbf{y}_j - \mathbf{y}_i\|^2}{2}\right). \end{aligned} \tag{10}$$

With this choice, the normalization constants of Eq. 10 cancel out when computing $p_{j|i}$ and $q_{j|i}$. Note that this approach gives a per-observation variance σ_i^2, such that different points effectively can have different sizes of neighborhoods. To determine the σ_i^2 parameters, the user specifies a *perplexity* parameter, which can be thought of as a measure of the effective number of neighbors [11]. This is defined as

$$\text{perplexity} = 2^{H(P_i)}, \tag{11}$$

where $H\left(P_i\right)$ is the Shannon entropy [44] of P_i in bits:

$$H\left(P_i\right) = -\sum_j p_{j|i} \log_2 p_{j|i}. \tag{12}$$

For a specific user-provided value of the perplexity parameter, we can perform a binary search over σ_i^2 such that Eq. 11 holds. In practice, the user experiments with different choices of perplexities to see which reveals a pattern.

E. The t-distributed stochastic neighbor embedding

The most popular variant of SNE is the *t-distributed SNE* [11]. This is motivated by the so-called 'crowding problem' often observed in SNE, where the low-dimensional representations significantly overlap without revealing much underlying structure. The idea is to use a similarity in representation space with more heavy tails than the Gaussian. Specifically, s_{low} is chosen as a t-distribution with one degree of freedom centered around one representation, i.e.

$$s_{\text{low}}(\mathbf{y}_j|\mathbf{y}_i) = \pi^{-1}\left(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2\right)^{-1}. \tag{13}$$

As is evident, t-SNE needs to compute all pairwise distances between data points and therefore has quadratic complexity. Using approximation techniques, such as vantage-point trees or the *Barnes-Hut approximation*, the running time can be lowered down to having $\mathcal{O}(n \log n)$ complexity [45].

III. Method

The inner workings of Rie-SNE, or *Riemannian Stochastic Neighbor Embedding*, are now examined.

A. Brownian motion on a Riemannian manifold

The key building block for generalizing SNE to Riemannian manifolds is a suitable generalization of the Gaussian distribution. Here we consider a density derived from a Brownian motion on a Riemannian manifold for high-dimensional probability computations.

Given a Brownian motion in Euclidean space, the probability that the random walk will end in point $\mathbf{x}$ can be computed by using the Gaussian density. If the increments of the Brownian motion are sufficiently small, each increment can be projected onto the tangent space of corresponding points on a Riemannian manifold without error. Then, a Brownian motion starting at point $\boldsymbol{\lambda}$ running for some time t can be projected onto a D-dimensional Riemannian manifold and there it will give rise to a random variable, a random variable that can be interpreted as the probability that a Brownian motion starting at $\boldsymbol{\lambda}$ will end in point $\mathbf{x}$ on the manifold. Since now the Brownian motion has been projected onto a Riemannian manifold the density will be different, and it can be approximated with a paramatrix expansion [46], [47]:

$$\mathcal{BM}(\mathbf{x}|\boldsymbol{\lambda}, t) \approx (2\pi t)^{-\frac{D}{2}} H_0 \exp\left(-\frac{\text{dist}^2(\mathbf{x}, \boldsymbol{\lambda})}{2t}\right) \tag{14}$$

where

- $t \in \mathbb{R}_+$ is the duration of the Brownian motion and corresponds to variance in Euclidean space.
- $\boldsymbol{\lambda}$ is the starting point of the Brownian motion.
- H_0 is the ratio of Riemannian volume measures evaluated at points $\mathbf{x}$ and $\boldsymbol{\lambda}$ respectively, i.e.:

$$H_0 = \left(\frac{\det G_{\mathbf{x}}}{\det G_{\boldsymbol{\lambda}}}\right)^{\frac{1}{2}} \tag{15}$$

with again $G_{\mathbf{p}}$ being the metric evaluated at $\mathbf{p}$.

Superficially, Eq. 14 looks like the density of the normal distribution, with the Euclidean distance being replaced by its Riemannian counterpart. However, here it is worth noting that the normalization factor H_0 is different from the usual Euclidean distribution.

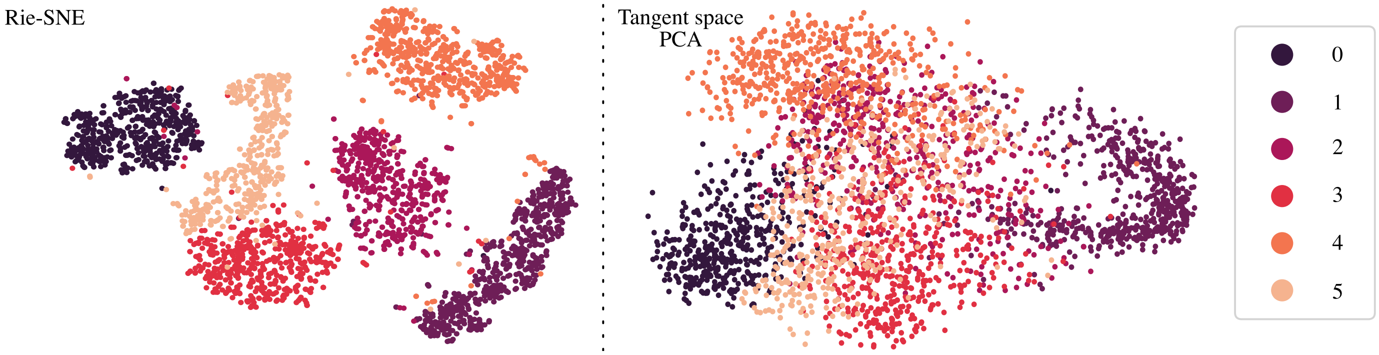

Fig. 2. Two-dimensional Euclidean embeddings of spherical MNIST. The left panel show the embedding obtained by Rie-SNE while the right panel show the gold standard of tangent space PCA. Note how Rie-SNE discovers structure, which is lost on tangent space PCA.

B. *Rie-SNE*

Rie-SNE works in a similar manner as SNE and t-SNE, i.e. it will also produce two probability distributions P and Q from the data and aim to make them as similar as possible and in the process capture some underlying structure in the produced low dimensional embedding. However, computing the high-dimensional probability distribution P comes with an added cost. To preserve the Riemannian nature of the data, a different density is used when computing high-dimensional probabilities belonging to P, namely the approximate density induced by the heat kernel of a Brownian motion on a Riemannian manifold given in Eq. 14, i.e. we pick

$$s_{\text{high}}(\mathbf{x}_j|\mathbf{x}_i) = \mathcal{BM}\mathbf{x}_j|\mathbf{x}_i, t_i. \tag{16}$$

The added computational cost is that the evaluation of $\mathcal{BM}(\cdot|\cdot)$ is more demanding than the conventional Gaussian similarity (10). Specifically, the normalization H_0 and the geodesic distance may be demanding, depending on the manifold on which the data resides.

With the Browninan motion model we get

$$p_{j|i} \approx \frac{H_0[i,j] \cdot \exp\left(-\frac{\text{dist}^2[i,j]}{2t_i}\right)}{\sum\limits_{k \neq i} H_0[i,k] \cdot \exp\left(-\frac{\text{dist}^2[i,k]}{2t_i}\right)}, \tag{17}$$

where we use the notations $\text{dist}^2[i,j] = \text{dist}^2(\mathbf{x}_i, \mathbf{x}_j)$ and $H_0[i,j] = \sqrt{\det G_{\mathbf{x}_i} / \det G_{\mathbf{x}_j}}$ to emphasize that these quantities can be pre-computed. As with SNE, we can optimize t_i to match a pre-specified perplexity using a binary search.

C. *Choice of representation*

As mentioned in Sec. I, Gauss's *Theorema Egregium* [1] inform us that we cannot isometrically embed data from a curved space into a space of different curvature without introducing distortion. Specifically, if we embed data from a nonlinear manifold onto a *flat* two-dimensional representation (for plotting) then the curvature mismatch between spaces induces a distortion. This is a fundamental limitation that any visualization of Riemannian data will face, but we may nonetheless try to limit its impact. One approach is to embed the data onto a manifold of similar curvature as that of the manifold on which the data resides. For example, if the data resides on a high-dimensional sphere, it is perhaps more prudent to embed onto a two-dimensional sphere for plotting, rather than a Euclidean space.

With this in mind, we choose different distributions over the low-dimensional representation, depending on user preference.

- **Euclidean.** If the user prefers a Euclidean low-dimensional representation, we opt to use a student-t as in regular t-SNE,
$$s_{\text{low}}(\mathbf{y}_j|\mathbf{y}_i) = \pi^{-1}\left(1 + \|\mathbf{y}_j - \mathbf{y}_i\|^2\right)^{-1}. \tag{18}$$
- **Spherical.** If the data manifold has positive curvature it may be beneficial to embed on a sphere, in which case we opt to use a von Mises-Fisher distribution [12],
$$s_{\text{low}}(\mathbf{y}_j|\mathbf{y}_i) = \left(\sqrt{2\pi} I_{d/2-1}(1)\right)^{-1} \exp\left(\mathbf{y}_j^\top \mathbf{y}_i\right), \tag{19}$$
where I_v is a modified Bessel function of the first kind of order v. In practise the normalization constant cancels out and can be ignored.
- **Other.** The user may have other prior knowledge about the manifold on which the data resides, which may suggest embedding on some other low-dimensional manifold. In this case, we suggest to also use the Riemannian Brownian over the low-dimensional representation, i.e.,
$$s_{\text{low}}(\mathbf{y}_j|\mathbf{y}_i) = \mathcal{BM}(\mathbf{y}_j|\mathbf{y}_i, 1). \tag{20}$$

Once we have defined both s_{high} and s_{low}, we can estimate the representations using gradient descent just as regular SNE. Having performed T iterations (with a sufficiently large value for T) of the gradient descent, the two probability distributions P and Q will have a minimal KL-divergence resulting in near-optimal positions of the points in the low-dimensional embedding. The key elements of the resulting computations are provided in Alg. 1 on page 7.

D. *Implementation details*

Our implementation of Rie-SNE relies on two approximation techniques that are traditionally also used when performing regular t-SNE.

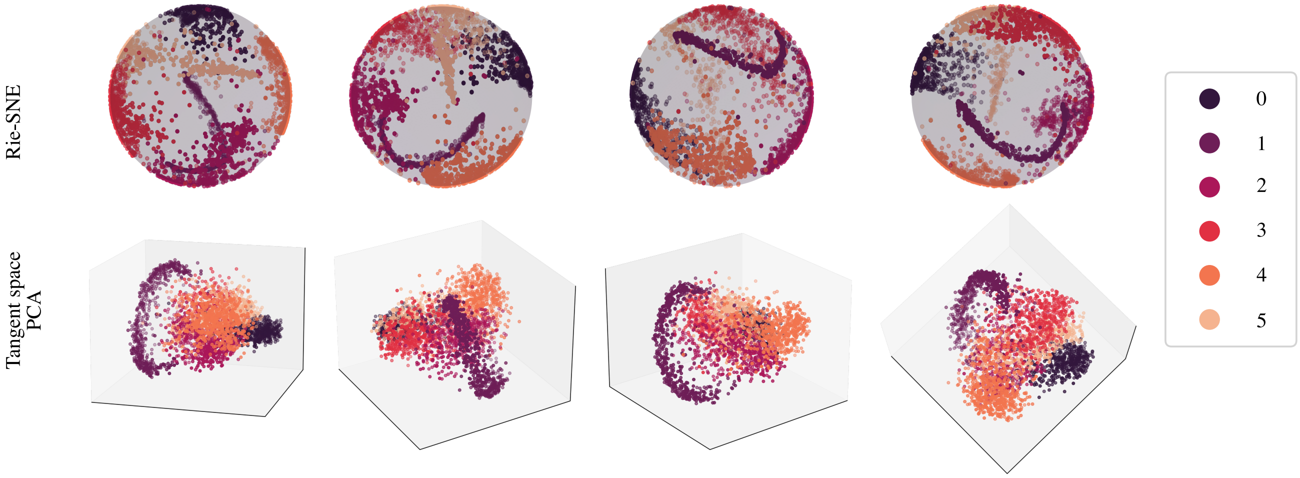

Fig. 3. Four different views of embeddings of spherical MNIST. The top row show spherical embeddings obtained using Rie-SNE, while the bottom row show three-dimensional embeddings using the gold standard tangent space PCA. Note that the clustering structure is significantly more evident in Rie-SNE.

First, we compute the high-dimensional probabilities in P by using a sparse nearest neighbor-based approximation technique [45]. This means we compute a sparse approximate P distribution, where far-away points are given a probability of zero. This can be realized with a nearest neighbor search. Empirical results from van der Maaten [45] suggest a value of $\tau = \lfloor 3 \cdot \text{perplexity} \rfloor$ nearest neighbors will give sufficiently good approximations of P, which we also use here. Finding the τ nearest neighbors of each point can be done by constructing a vantage-point tree [48] over the data and performing a nearest neighbors search on the resulting tree. Constructing the tree, performing the nearest neighbor search and computing the relevant values of P has time complexity $\mathcal{O}(\tau n \log n)$.

Second, we use the *Barnes-Hut approximation* [49], whenever we opt to use a student's t-distribution over the low-dimensional representation. Minimizing the KL-divergence between the two probability distributions P and Q requires using gradient descent, and in each step of the gradient descent we need to compute all pairwise q_{ij} of Q, which has quadratic complexity. The Barnes-Hut approximation of the gradient instead has complexity $\mathcal{O}(n \log n)$. In short, this approximation split the low-dimensional representation into quadrants (via tree structures named quadtrees/octtrees), such that points in far-away and small enough quadrants can be approximated as the same point appearing multiple times. For each low-dimensional point, a depth-first search with complexity $\mathcal{O}(\log n)$ is done to mark quadrants as approximate quadrants or not. A total of n depth-first searches are carried out, yielding the $\mathcal{O}(n \log n)$ time complexity.

IV. Results

The performance of Rie-SNE is shown by comparing it to the gold-standard of visualizing non-euclidean data. This amounts to first computing the intrinsic mean, mapping all data to the tangent space at this point, and performing PCA over the tangential data.

A. *Spherical MNIST*

We start with the classic MNIST dataset [50] consiting of 24×24 dimensional gray-scale images. We consider digits 0–5 to reduce clutter. To induce a non-Euclidean data geometry, we project the data onto the unit sphere of $\mathbb{R}^{24\times 24}$ and denote the resulting data *spherical MNIST*. We visualize the resulting data using both Rie-SNE and tangent space PCA. First, we embed the data onto the plane, $\mathbb{R}^2$, and show the resulting plots in Fig. 2. Here it can be seen that Rie-SNE captures well the underlying relationship between the data points (same digits are grouped together), while tangent space PCA produces a cluttered view which does not reveal the underlying structure. Since the data has a spherical geometry, it may be beneficial to embed onto a low-dimensional sphere to better preserve topology and curvature. Figure 3 show the Rie-SNE embedding onto $\mathcal{S}^2$, where the clustering is again evident. As a baseline, the figure also shows a tangent space PCA embedding on $\mathbb{R}^3$. Although some structure can be captured with tangent space PCA here, Rie-SNE still gives better separation of the digit classes.

B. *Crypto-tensors*

Following Mallasto et al. [41] we consider the price of 10 popular crypto-currencies[1] over the time period *2.12.2014 — 15.5.2018*. As is common in economy [51] the relationship between prices is captured by a 10×10 covariance matrix constructed from the past 20 days. This gives rise to a time series of covariance matrices, each of which reside on the cone of symmetric positive definite matrices. We provide visualizations of the data in Fig. 4. Rie-SNE is used to produce visualizations on both the plane $\mathbb{R}^2$ and the sphere $\mathcal{S}^2$, showing a one-dimensional structure capturing the time-evolution behind the data. In contrast, tangent space PCA produce $\mathbb{R}^2$ and $\mathbb{R}^3$ visualizations showing little to no structure in the embeddings.

[1] Bitcoin, Dash, Digibyte, Dogecoin, Litecoin, Vertcoin, Stellar, Monero, Ripple, and Verge.

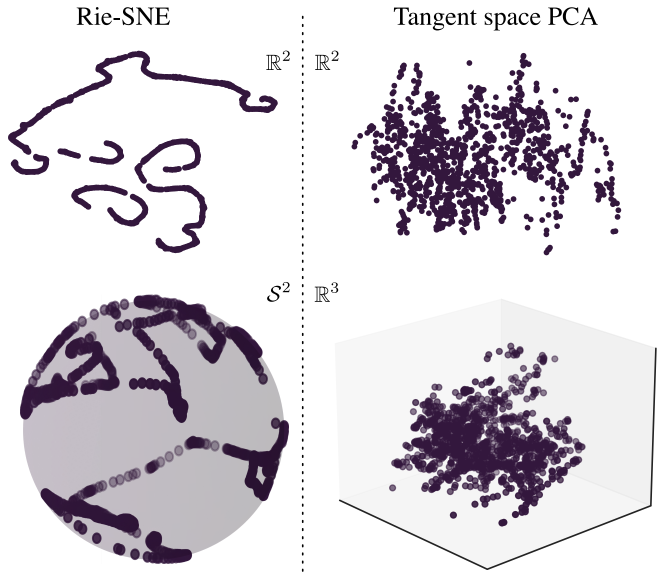

Fig. 4. Embeddings of symmetric positive definite matrices using Rie-SNE (left) and tangent space PCA (right). The top row show two-dimensional Euclidean embeddings, while bottom row show spherical and $\mathbb{R}^3$ embeddings, respectively. In both cases Rie-SNE recovers a one-dimensional signal matching the underlying time series, while tangent space PCA does not.

V. Conclusions

In this paper, we presented a new type of visualization technique, *Rie-SNE*, that is aimed at data residing on Riemannian manifolds, such as spheres. It is a SNE-based technique that can additionally produce different kinds of low-dimensional embeddings depending on user preference and the curvature of the original data manifold. We compare Rie-SNE to a standard technique when it comes to visualizing non-euclidean data, which is to perform PCA on the data mapped to tangent space. The results are promising, and we believe this technique could have some merit. For future work we would like to see this taken further: it would be interesting to see more visualizations of data mapped to well-known manifolds, other than those that were used in this paper, and it would especially be interesting to try this out on some non-standard manifolds. In the case of non-standard manifolds, computing geodesics is likely non-trivial, such that some approximation techniques might need to be developed. We hope that the present work also paves the way for other visualization tools for Riemannian data in order to support investigators relying on geometric models.

Acknowledgements

SH was supported by research grants (15334, 42062) from VILLUM FONDEN. This project has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n$^{\text{o}}$ 757360). This work was funded in part by the Novo Nordisk Foundation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606).

References

[1] Karl Friedrich Gauss and Peter Pesic. *General investigations of curved surfaces*. Courier Corporation, 2005.

[2] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 857–864. MIT Press, 2003.

[3] Jake VanderPlas. *Python data science handbook: Essential tools for working with data.* " O'Reilly Media, Inc.", 2016.

[4] Daniel Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM journal on scientific and statistical computing*, 6(1):128–143, 1985.

[5] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

[6] Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.

[7] Cilie W. Feldager, Søren Hauberg, and Lars Kai Hansen. Spontaneous symmetry breaking in data visualization. In *International Conference on Artificial Neural Networks (ICANN)*, 2021.

[8] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[9] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[10] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[12] Kanti V Mardia, Peter E Jupp, and KV Mardia. *Directional statistics*, volume 2. Wiley Online Library, 2000.

[13] Gerhard Kurz, Igor Gilitschenski, and Uwe D Hanebeck. Recursive nonlinear filtering for angular data based on circular distributions. In *2013 American Control Conference*, pages 5439–5445. IEEE, 2013.

[14] Søren Hauberg. Directional statistics with the spherical normal distribution. In *Proceedings of FUSION 2018*, 2018.

[15] D.G. Kendall. Shape manifolds, Procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984.

[16] O. Freifeld and M.J. Black. Lie bodies: A manifold representation of 3D human shape. In A. Fitzgibbon et al. (Eds.), editor, *European Conference on Computer Vision (ECCV)*, Part I, LNCS 7572, pages 1–14. Springer-Verlag, 2012.

[17] Anuj Srivastava, Shantanu H Joshi, Washington Mio, and Xiuwen Liu. Statistical shape analysis: Clustering, learning, and testing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(4):590–602, 2005.

[18] Laurent Younes. Spaces and manifolds of shapes in computer vision: An overview. *Image and Vision Computing*, 30(6):389–397, 2012.

[19] Sebastian Kurtek, Eric Klassen, John C Gore, Zhaohua Ding, and Anuj Srivastava. Elastic geodesic paths in shape space of parameterized surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(9):1717–1730, 2012.

[20] C. Lenglet, R. Deriche, and O.D. Faugeras. Inferring white matter geometry from diffusion tensor MRI: Application to connectivity mapping. In *European Conference on Computer Vision (ECCV)*, volume 3024 of *Lecture Notes in Computer Science*, pages 127–140, 2004.

[21] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision (IJCV)*, 66(1):41–66, 2006.

[22] Yuanxiang Wang, Hesamoddin Salehian, Guang Cheng, and Baba C Vemuri. Tracking on the product manifold of shape and orientation for tractography from diffusion MRI. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3051–3056, 2014.

[23] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *European Conference on Computer Vision (ECCV)*, pages 589–600. Springer, 2006.

[24] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on Lie algebra. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 728–735, 2006.

[25] Oren Freifeld, Søren Hauberg, and Michael J. Black. Model transport: Towards scalable transfer learning on manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[26] Pavan K. Turaga, Ashok Veeraraghavan, Anuj Srivastava, and Rama Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(11):2273–2286, 2011.

[27] H.E. Çetingul and R. Vidal. Intrinsic mean shift for clustering on stiefel and grassmann manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1896–1902, 2009.

[28] Salem Said, Nicolas Courty, Nicolas Le Bihan, and Stephen J Sangwine. Exact principal geodesic analysis for data on SO(3). In *Proceedings of the 15th European Signal Processing Conference*, pages 1700–1705, 2007.

[29] Søren Hauberg, Stefan Sommer, and Kim S. Pedersen. Natural metrics and least-committed priors for articulated tracking. *Image and Vision Computing*, 30(6-7):453–461, 2012.

[30] Søren Hauberg, Stefan Sommer, and Kim Steenstrup Pedersen. Gaussian-like Spatial Priors for Articulated Tracking. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *ECCV*, volume 6311 of *LNCS*, pages 425–437. Springer, 2010.

[31] Søren Hauberg, François Lauze, and Kim Steenstrup Pedersen. Unscented Kalman Filtering on Riemannian Manifolds. *Journal of Mathematical Imaging and Vision*, 2011.

[32] Igor Gilitschenski, Gerhard Kurz, Simon J Julier, and Uwe D Hanebeck. Unscented orientation estimation based on the bingham distribution. *IEEE Transactions on Automatic Control*, 61(1):172–177, 2015.

[33] Noémie Jaquier, Viacheslav Borovitskiy, Andrei Smolensky, Alexander Terenin, Tamim Asfour, and Leonel Rozo. Geometry-aware bayesian optimization in robotics using riemannian matérn kernels. In *Conference on Robot Learning*, pages 794–805. PMLR, 2022.

[34] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincaré variational auto-encoders. *Advances in neural information processing systems*, 32, 2019.

[35] Xavier Pennec. Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006.

[36] P. Thomas Fletcher, Conglin Lu, Stephen M. Pizer, and Sarang Joshi. Principal Geodesic Analysis for the study of Nonlinear Statistics of Shape. *IEEE Transactions on Medical Imaging (TMI)*, 23(8):995–1005, 2004.

[37] Stefan Sommer, François Lauze, Søren Hauberg, and Mads Nielsen. Manifold Valued Statistics, Exact Principal Geodesic Analysis and the Effect of Linear Approximations. In K. Daniilidis, P. Maragos, , and N. Paragios, editors, *ECCV '10: Proceedings of the 11th European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 43–56. Springer, Heidelberg, September 2010.

[38] S. Huckemann, T. Hotz, and A. Munk. Intrinsic shape analysis: geodesic PCA for Riemannian manifolds modulo isometric Lie group actions. *Statistica Sinica*, 20(1):1–58, 2010.

[39] Victor M. Panaretos, Tung Pham, and Zhigang Yao. Principal flows. *Journal of the American Statistical Association (JASA)*, 109(505):424–436, 2014.

[40] Sungkyu Jung, Ian L Dryden, and JS Marron. Analysis of principal nested spheres. *Biometrika*, 99(3):551–568, 2012.

[41] Anton Mallasto, Søren Hauberg, and Aasa Feragen. Probabilistic riemannian submanifold learning with wrapped gaussian process latent variable models. In *Proceedings of the 19th international Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[42] Søren Hauberg. Principal curves on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[43] M.P. do Carmo. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992.

[44] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

[45] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, January 2014.

[46] E.P. Hsu and American Mathematical Society. *Stochastic Analysis on Manifolds*. Graduate studies in mathematics. American Mathematical Society, 2002.

[47] Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with riemannian brownian motion priors, 2020.

[48] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, pages 311–321, 1993.

[49] J. E. Barnes and P. Hut. A hierarchical O(n-log-n) force calculation algorithm. *Nature*, 324:446, 1986.

[50] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *Online*, 2010.

[51] Andrew Gordon Wilson and Zoubin Ghahramani. Generalised wishart processes. *arXiv preprint arXiv:1101.0240*, 2010.

Algorithm 1: Computing conditional probabilities $p_{j|i}$

```
Data: Geodesic distance matrix: $D$,
      Riemannian volume measure ratio matrix: $\mathcal{H}_0$,
      Data space dimensionality: $dim$,
      Desired perplexity: $desired\_perplexity$
Result: $n \times n$ matrix of conditional probabilities: $P$
begin
    $P \leftarrow zeros(n, n)$
    $binary\_search\_steps \leftarrow 100$
    $perplexity\_tolerance \leftarrow 10^{-5}$
    for i=0 to $n$ do
        $t\_min \leftarrow -\infty$
        $t\_max \leftarrow \infty$
        $t \leftarrow 1.0$
        for l=0 to $binary\_search\_steps$ do
            $row\_sum \leftarrow 0.0$
            for j=0 to $n$ do
                $P[i,j] \leftarrow$
                    $(2\pi t)^{-\frac{dim}{2}} \cdot \mathcal{H}_0[i,j] \cdot \exp\left(-\frac{D[i,j]^2}{2t}\right)$
                $row\_sum \leftarrow row\_sum + P[i,j]$
            end
            $entropy \leftarrow 0.0$
            for j=0 to $n$ do
                $P[i,j] \leftarrow \frac{P[i,j]}{row\_sum}$
                if $P[i,j] \neq 0.0$ then
                    $entropy \leftarrow$
                        $entropy + P[i,j] \cdot \log_2(P[i,j])$
                end
            end
            $entropy\_diff \leftarrow$
                $-entropy - \log_2(desired\_perplexity)$
            if $|entropy\_diff| \leq perplexity\_tolerance$
              then
                $break$
            end
            if $entropy\_diff < 0.0$ then
                $t\_min \leftarrow t$
                if $t\_max == \infty$ then
                    $t \leftarrow 2 \cdot t$
                else
                    $t \leftarrow \frac{t + t\_max}{2.0}$
                end
            else
                $t\_max \leftarrow t$
                if $t\_min == -\infty$ then
                    $t \leftarrow \frac{t}{2}$
                else
                    $t \leftarrow \frac{t + t\_min}{2.0}$
                end
            end
        end
    end
    return $P$
end
```