Approximated and User Steerable tSNE for Progressive Visual Analytics

Pezzotti, Nicola; Lelieveldt, Boudewijn P.F.; van der Maaten, Laurens; Höllt, Thomas; Eisemann, Elmar; Vilanova, Anna

DOI
10.1109/TVCG.2016.2570755

Publication date
2016

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Visualization and Computer Graphics

Approximated and User Steerable tSNE for Progressive Visual Analytics

Nicola Pezzotti, Boudewijn P.F. Lelieveldt, Laurens van der Maaten,
Thomas Höllt, Elmar Eisemann, and Anna Vilanova

Abstract—Progressive Visual Analytics aims at improving the interactivity in existing analytics techniques by means of visualization as well as interaction with intermediate results. One key method for data analysis is dimensionality reduction, for example, to produce 2D embeddings that can be visualized and analyzed efficiently. t-Distributed Stochastic Neighbor Embedding (tSNE) is a well-suited technique for the visualization of high-dimensional data. tSNE can create meaningful intermediate results but suffers from a slow initialization that constrains its application in Progressive Visual Analytics. We introduce a controllable tSNE approximation (A-tSNE), which trades off speed and accuracy, to enable interactive data exploration. We offer real-time visualization techniques, including a density-based solution and a Magic Lens to inspect the degree of approximation. With this feedback, the user can decide on local refinements and steer the approximation level during the analysis. We demonstrate our technique with several datasets, in a real-world research scenario and for the real-time analysis of high-dimensional streams to illustrate its effectiveness for interactive data analysis.

Index Terms—High Dimensional Data, Dimensionality Reduction, Progressive Visual Analytics, Approximate Computation

1 Introduction

Visual analysis of high dimensional data is a challenging process. Direct visualizations such as parallel coordinates [1] or scatterplot matrices [2] work well for a few dimensions but do not scale to hundreds or thousands of dimensions. Typically indirect visualization is used for these cases. First the dimensionality of the data is reduced, usually to two or three dimensions, then the remaining dimensions are used to lay out the data for visual inspection, for example in a two dimensional scatterplot.

Dimensionality reduction techniques have been an active field of research in the last years, resulting in a number of viable techniques [3]. A variant of tSNE [4], the Barnes Hut SNE [5] has been accepted as the state of the art for non-linear dimensionality reduction applied to visual analysis of high-dimensional space in several application areas, such as life sciences [6], [7], [8], [9]. tSNE produces 2D and 3D embeddings that are meant to preserve local structure in the high-dimensional data. The analyst inspects the embeddings with the goal to identify clusters or patterns that are used to generate new hypothesis on the data, however, the computational complexity of this technique does not allow direct employment in interactive systems. This limitation makes the analytic process a time consuming task that can take hours, or even days, to adjust the parameters and generate the right embedding to be analyzed.

- *N. Pezzotti, T. Höllt, E. Eisemann, and A. Vilanova are with the Computer Graphics and Visualization group, Delft University of Technology, Delft, the Netherlands.*
- *B. P.F. Lelieveldt and L. van der Maaten are with the Pattern Recognition and Bioinformatics group, Delft University of Technology, Delft, the Netherlands.*
- *B. P.F. Lelieveldt is with the Division of Image Processing, Department of Radiology, Leiden University Medical Center, Leiden, the Netherlands.*

Manuscript received August 4, 2015; revised -, -.

Recently Stolper et al. [10], as well as Mühlbacher et al. [11] introduced Progressive Visual Analytics. The idea of Progressive Visual Analytics is to provide the user with meaningful intermediate results, in case computation of the final result is too costly. Based on these intermediate results the user can start with the analysis process. Mühlbacher et al. also provide a set of requirements, which an algorithm needs to fulfill in order to be suitable for Progressive Visual Analytics. Based on these requirements they analyze a series of different algorithms, commonly deployed in visual analytics systems and conclude that, for example, tSNE fulfills all requirements. The reason being that the minimization in tSNE builds up on the iterative gradient descent technique [4] and can therefore be used directly for a per-iteration visualization, as well as interaction with the intermediate results. However, Mühlbacher et al. ignore the fact that the distances in the high-dimensional space need to be precomputed to start the minimization process. In fact this initialization process is dominating the overall performance of tSNE. Even with a per-iteration visualization of the intermediate results [10], [11], [12], [13] the initialization time will force the user to wait minutes, or even hours, before the first intermediate result can be generated on a state-of-the-art desktop computer. Every modification of the data, for example, the addition of data-points or a change in the high-dimensional space, will force the user to wait for the full reinitialization of the algorithm.

In this work, we present A-tSNE, a novel approach to adapt the complete tSNE pipeline, including a distance computation for the Progressive Visual Analytics paradigm. Instead of precomputing precise distances, we propose to approximate the distances using Approximated K-Nearest Neighborhood queries. This allows us to start the computation of the iterative minimization nearly instantly after loading the data. Based on the intermediate results of the tSNE, the user can now start the interpretation process of the

data immediately. Further, we modified the gradient descent of tSNE such that it allows for the incorporation of updated data during the iterative process. This change allows us to continuously refine the approximated neighborhoods in the background, triggering updates of the embedding without restarting the optimization. Eventually, this process arrives at the precise solution. Furthermore, we allow the user to steer the level of approximation by selecting points of interest, such as clusters, which appear in the very early stages of the optimization and enable an interactive exploration of the high-dimensional data.

Our contributions are as follows:

1) We present A-tSNE, a twofold evolution of the tSNE algorithm, which
   a) minimizes initialization time and as such enables immediate inspection of preliminary computation results.
   b) allows for interactive modification, removal or addition of high-dimensional data, without disrupting the visual analysis process.
2) Using a set of standard benchmark data sets, we show large computational performance improvements of A-tSNE compared to the state of the art while maintaining high precision.
3) We developed an interactive system for the visual analysis of high dimensional data, allowing the user to inspect and steer the level of approximation. Finally, we illustrate the benefits of exploratory possibilities in a real-world research scenario and for the real-time analysis of high-dimensional streams.

2 Related work

The tSNE [4] algorithm builds the foundation of this work. As described above, tSNE is used for visualization of high-dimensional data in a wide field of applications, from life sciences to the analysis of deep-learning algorithms [6], [7], [8], [9], [14], [15], [16]. tSNE is a non-linear dimensionality-reduction algorithm that aims at preserving local structures in the embedding, whilst showing global information, such as the presence of clusters at several scales. Most of the user tasks associated with the visualization of high-dimensional data embeddings are based on identifying relationships between data points. Typical tasks comprises the identification of visual clusters and their verification based on detail visualization of the high-dimensional data, e.g., using parallel coordinate plots. For a complete description of such tasks we refer to Brehmer et al. [17].

tSNE's computational and memory complexity is $O(N^2)$, where N is the number of data-points, which constrains the application of the technique. An evolution of the algorithm, called Barnes-Hut-SNE (BH-SNE) [5], reduces the computational complexity to $O(N \log(N))$ and the memory complexity to $O(N)$. This approach was also developed in parallel by Yang et al. [18]. However, despite the increased performance, it still cannot be used to interactively explore the data in a desktop environment.

Interactive performance is at the center of the latest developments in Visual Analytics. New analytical tools and algorithms, which are able to trade accuracy for speed and offer the possibility to interactively refine results [19], [20], are needed to deal with the scalability issues of existing analytics algorithms like tSNE. Mühlbacher et al. [11] defined different strategies to increase the user involvement in existing algorithms. They provide an in-depth analysis on how the interconnection between the visualization and the analytic modules can be achieved. Stolper et al. [10] defined the term *Progressive Visual Analytics*, describing techniques that allow the analyst to directly interact with the analytics process. Visualization of intermediate results is used to help the user, for example, to find optimal parameter settings or filter the data [10]. For the design of our Progressive Visual Analytics approach, we used the guidelines presented by Stolper et al. [10], see section 4. Many algorithms are not suited right away for Progressive Visual Analytics since the production of intermediate results is computationally too intensive or they do not generate useful intermediate results at all. tSNE is an example of such an algorithm because of its initialization process.

To overcome this problem, we propose to compute an approximation of tSNE's initialization stage, followed by a user steerable [21] refinement of the level of approximation. To compute the conditional probabilities needed by BH-SNE, a K-Nearest Neighborhood (KNN) search must be evaluated for each point in the high-dimensional space. Under these conditions, a traditional algorithm and data structure, such as a KD-Tree [22], will not perform well. In the BH-SNE [5] algorithm, a Vantage-Point Tree [23] is used for the KNN search, but it is slow to query. In this work, we propose to use an approximated computation of the KNN in the initialization stage to start the analysis as soon as possible. The level of approximation is then refined on the fly during the analytics process.

Other dimensionality-reduction algorithms implement approximation and steerability to increase performance as well. For example MDSteer [24] works on a subset of the data and allows the user to control the insertion of points by selecting areas in the reduced space. Yang et al. [25] present a dimensionality-reduction technique using a dissimilarity matrix as input. By means of a divide-and-conquer approach, the computational complexity of the algorithm can be reduced. Multiple other techniques provide steerability by means of guiding the dimensionality reduction via user input. Joja et al. [26] and Paulovich et al. [27] let the user place a small number of control points. In other work, Paulovich et al. [28], propose the use of a non-linear dimensionality-reduction algorithm on a small number of automatically-selected control points. For these techniques the position of the data points is finally obtained by linear-interpolation schemes that make use of the control points. However, they all limit the non-linear dimensionality reduction to a subset of the dataset limiting the insights that can be obtained from the data. In this work, we provide a way to directly use the complete data allowing the analyst to immediately start the analysis on all data points.

Ingram and Munzner's Q-SNE [29] is based on a similar idea as our approach, using Approximated KNN queries for the computation of the high-dimensional similarities. However, they use the APQ algorithm [29] that is designed to exploit the sparse structure of high-dimensional spaces obtained from document collections, limiting its application

to such a context. A-tSNE improves Q-SNE in the direction of providing a fast but approximated algorithm for the analysis of traditional dense high-dimensional spaces. For this reason it can be used right away in contexts where BH-SNE is applied and Q-SNE would not be applicable. A further distinction is that A-tSNE incorporates the principles of the Progressive Visual Analytics by means of providing a visualization of the level of approximation, the ability to refine the approximation based on user input, and allowing the manipulation of the high-dimensional data without waiting for the recomputation of the exact similarities.

Density-based visualization of the tSNE embedding has been used in several works [5], [6], [9], however, they employ slow-to-compute offline techniques. In our work, we integrate real-time Kernel Density Estimation (KDE) as described by Lampe and Hauser [30]. The interaction with the embedding is important to allow the analyst to explore the high-dimensional data. Selection operations in the embedding and the visualization of the data in a coordinated multiple-view system are necessary to enable this exploration. The iVisClassifier system [31] is an example of such a solution. In our work, we take a similar approach, providing a coordinated multiple-view framework for the visualization of a selection in the embedding.

3 tSNE

In this section, we provide a short introduction to tSNE [4], which is necessary to explain our contribution. tSNE interprets the overall distances between data-points in the high-dimensional space as a symmetric joint-probability distribution P. Likewise a joint-probability distribution Q is computed, that describes the similarity in the low-dimensional space. The goal is to achieve a representation, referred to as *embedding*, in the low dimensional space where Q faithfully represents P. This is achieved by optimizing the positions in the low-dimensional space to minimize the cost function C given by the Kullback-Leibler (KL) divergence between the joint-probability distributions P and Q:

$$C(P,Q) = KL(P||Q) = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} p_{ij} \ln\left(\frac{p_{ij}}{q_{ij}}\right) \quad (1)$$

Given two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the dataset $X = \{\mathbf{x}_1 ... \mathbf{x}_N\}$, the probability p_{ij} models the similarity of these points in the high-dimensional space. To this extent, for each point a Gaussian kernel, P_i, is chosen whose variance σ_i is defined according to the local density in the high-dimensional space and then p_{ij} is described as follows:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}, \quad (2)$$

$$\text{where} \quad p_{j|i} = \frac{\exp(-(||\mathbf{x}_i - \mathbf{x}_j||^2)/(2\sigma_i^2))}{\sum_{k\neq i}^{N} \exp(-(||\mathbf{x}_i - \mathbf{x}_k||^2)/(2\sigma_i^2))} \quad (3)$$

$p_{j|i}$ can be seen as a relative measure of similarity based on the local neighborhood of a data-point $\mathbf{x}_i$. The perplexity value μ is a user-defined parameter that describes the effective number of neighbors considered for each data-point. The value of σ_i is chosen such that for fixed μ and each i:

$$\mu = 2^{-\sum_j^N p_{j|i} \log_2 p_{j|i}} \quad (4)$$

A *Student's t-Distribution* with one degree of freedom is used to compute the joint-probability distribution in the low-dimensional space Q, where the positions of the data-points should be optimized. Given two low-dimensional points $\mathbf{y}_i$ and $\mathbf{y}_j$, the probability q_{ij} that describes their similarity is given by:

$$q_{ij} = \left((1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)Z\right)^{-1} \quad (5)$$

$$\text{with} \quad Z = \sum_{k=1}^{N} \sum_{l\neq k}^{N} (1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1} \quad (6)$$

The gradient of the Kullback-Leibler divergence between P and Q is used to minimize C (see Eq. 1). It indicates the change in position of the low-dimensional points for each step of the gradient descent and is given by:

$$\frac{\delta C}{\delta \mathbf{y}_i} = 4 \sum_{i=1}^{N} (F_i^{\text{attr}} - F_i^{\text{rep}}) \quad (7)$$

$$= 4 \sum_{i=1}^{N} \left(\sum_{j\neq i}^{N} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j\neq i}^{N} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right) \quad (8)$$

The gradient descent can be seen as a *N-body simulation* [32], where each data-point exerts an attractive and a repulsive force on all the other points (F_i^{attr} and F_i^{rep}).

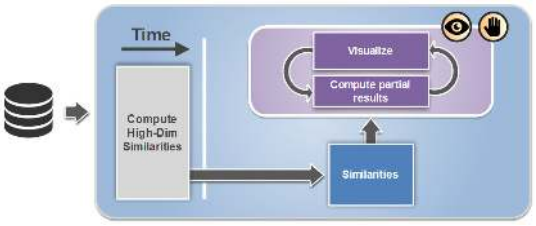
3.1 Barnes-Hut-SNE

In the original tSNE, the force is computed using a brute-force approach, resulting in computational and memory complexity of $O(N^2)$. Barnes-Hut-SNE (BH-SNE) [5] is an evolution of the tSNE algorithm that introduces two different approximations to reduce the computational complexity to $O(N \log(N))$ and the memory complexity to $O(N)$.

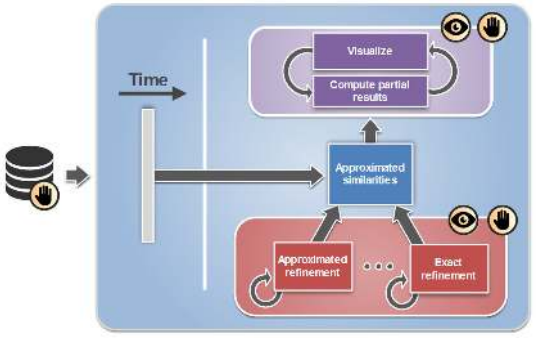
The first approximation is based on the observation that the probability p_{ij} is infinitesimal if $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar. Therefore, the similarities of a data-point $\mathbf{x}_i$ can be computed taking into account only the points that belong to the set of nearest neighbors $\mathcal{N}_i$. The cardinality of $\mathcal{N}_i$ can be set to $K = \lfloor 3\mu \rfloor$, where μ is the user-selected perplexity and $\lfloor \cdot \rfloor$ describes a rounding to the next-lower integer. Without compromising the quality of the embedding [5], we can adopt a sparse approximation of the high-dimensional similarities. Eq. 3 can now be written as follows:

$$p_{j|i} = \begin{cases} \frac{\exp(-(||\mathbf{x}_i - \mathbf{x}_j||^2)/(2\sigma_i^2))}{\sum_{k\in\mathcal{N}_i} \exp(-(||\mathbf{x}_i - \mathbf{x}_k||^2)/(2\sigma_i^2))} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

The computation of the K-Nearest Neighbors is performed using a Vantage-Point Tree (VP-Tree) [23]. A VP-Tree is data structure that computes KNN queries in a high-dimensional metric space, in $O(\log(N))$ time. It is a binary tree that stores for each non leaf-node a hyper-sphere centered on a data-point. The left children of each node will contain the points that reside inside the hyper-sphere, whereas the right one will contain the points outside it.



(a) Progressive Visual Analytics workflow for tSNE.



(b) Progressive Visual Analytics workflow for A-tSNE.

Fig. 1. **Comparison between the traditional and our tSNE workflow.** The eye icon marks modules which produce output for visualization, whereas the hand icon marks modules that allow manipulation by the user. The increased performance of the similarity computation allows the user to seamlessly manipulate the input data. The level of approximation can be visualized and the user can steer the refinement process to interesting regions.

The second approximation makes use of the formulation of the gradient presented in Eq. 7. As described above tSNE can be seen as an N-body simulation and thus the Barnes-Hut algorithm [33] can be used to reduce the computational complexity to $O(N \log(N))$. For further details, we refer to van der Maaten [5].

4 A-tSNE in Progressive Visual Analytics

In this work, we introduce Approximated-tSNE (A-tSNE), an evolution of the BH-SNE algorithm, using approximated computations of high-dimensional similarities to generate meaningful intermediate results. The level of approximation can be defined by the user to allow control on the trade off between speed and quality. The level of approximation can be refined by the analyst in interesting regions of the embedding, making A-tSNE a computational steerable algorithm [21]. tSNE is well suited for the application in Progressive Visual Analytics: after the initialization of the algorithm, the intermediate results generated during the iterative optimization process can be interpreted by the analyst while they change over time, as shown in previous work [11], [12]. Fig. 1a shows a typical Progressive Visual Analytics workflow for tSNE.

Algorithms that can be used in a Progressive Visual Analytics system often have a computational module, e.g. the initialization of the technique, that cannot be implemented in an iterative way, creating a *speed bump* [10] in the user analysis. tSNE is a good example for such an algorithm. It consists of two computational modules that are serialized. In the first part of the algorithm, similarities between high-dimensional points are calculated. In the second module, a minimization of the cost function (Eq. 1) is computed by means of a gradient descent. The first module, depicted in light grey in Fig. 1a, is slow to compute and does not create any meaningful intermediate results.

We extend the Progressive Visual Analytics paradigm by introducing approximated computation rather than aiming at exact computations, in the modules that are not suited for a per-iteration visualization. Fig. 1b shows the analytical workflow for A-tSNE. While the generation and the inspection of the intermediate results is not changed, we introduce a refinement module, depicted in red in Fig. 1b, which can be used to refine the level of the approximation in the embedding in a concurrent way. Furthermore, the increased performance of the initialization module and the ability to update the high-dimensional similarities during the gradient descent minimization, allows the analyst to manipulate the high-dimensional data without waiting for the reinitialization of the algorithm. We follow the guideline proposed by Stolper et al. [10], focusing on providing increasingly meaningful partial results during the minimization process (purple modules in Fig. 1). Furthermore, we impose the following requirements to the modules that compute the approximated similarities (grey and red modules in Fig. 1):

1) The performance gain due to the approximation must be high enough to enable interaction.
2) The amount of degradation caused by the approximation must be controllable. A small increase of approximation must not lead to large degradation of the results.
3) The approximation quality can be measured and visualized to avoid misleading the user.
4) The approximation can be refined during the evolution. The refinement can be steered by the user.

In the following Sections 4.1 to 4.4, we describe the A-tSNE algorithm in detail using the MNIST [34] dataset for illustration. The dataset consists of 60k labeled gray scale images of handwritten digits (compare Fig. 2a). Each image is represented as a 784 dimensional vector, corresponding to the gray values of the pixels in the image.

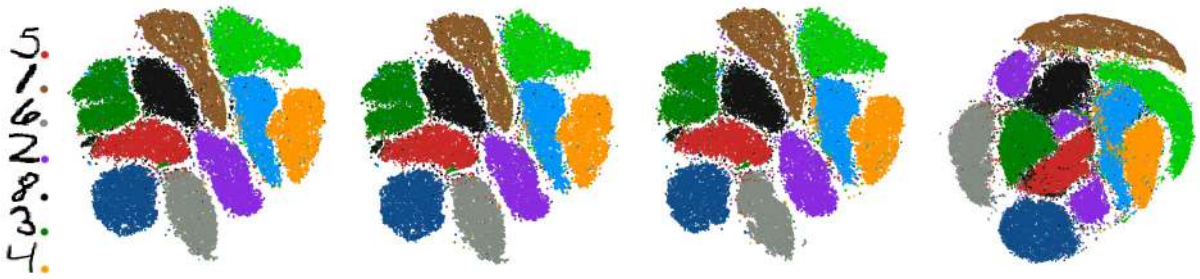
4.1 A-tSNE

A-tSNE improves the BH-SNE algorithm, by using fast and Approximated KNN computations to build the approximated high-dimensional joint-probability distribution P^A, instead of the exact distribution P. The cost function $C(P^A, Q^A)$ is then minimized in order to obtain the approximated embedding described by Q^A.

The similarity between points can be computed using the set of approximated neighbors $\mathcal{N}_i^A$, instead of the exact neighborhood $\mathcal{N}_i$ (see Eq. 9). We define the precision of the KNN algorithm as ρ. ρ describes the average percentage of points in the approximated neighborhood $\mathcal{N}_i^A$ that belongs to the exact neighborhood $\mathcal{N}_i$:

$$\rho = \sum_{i=1}^{N} \frac{\rho_i}{N} \quad \rho_k = \frac{|\mathcal{N}_k^A \cap \mathcal{N}_k|}{|\mathcal{N}_k|}, \tag{10}$$

where $|\cdot|$ indicates the cardinality of the neighborhood. The cardinality of $\mathcal{N}_k$ is indirectly specified by the user



(a) (b) BH-SNE - Time: 3191.8 s (c) $\rho = 0.34$ - Time: 30.1 s (d) $\rho = 0.23$ - Time: 20.4 s (e) $\rho = 0.07$ - Time: 13.0 s

Fig. 2. **Embeddings of the MNIST dataset** using different approximation levels. Each point represents an image of a handwritten digit, few examples are shown in (a). Points are colored according to the classification of the image. It can be seen that a reasonable approximation as in (c) and (d) produces nearly identical results, compared to the original BH-SNE (b) two orders of magnitude faster. Even very low precision (e) produces clearly distinguishable clusters, even though the embedding visually differs from (b)-(d). Extensive tests on the quality of the results are provided in Sec. 4.4.

as explained in Sec. 3.1, as three times the value of the perplexity parameter μ. ρ is an input parameter that can be defined by the user. The larger the value of ρ the more similar will P_A be to P and in turn the more similar the approximated embedding will be to the exact one.

To better understand the effect of the approximated queries, it is useful to interpret the BH-SNE algorithm as a force-directed layout algorithm [35], which acts on an undirected graph created by the KNN relationships. A data point $\mathbf{x}_i$ is repelled by all other data-points but to a subset of the data-points given by its neighborhood relationships, where attraction forces are created by a set of springs which connect $\mathbf{x}_i$ with all the points in $\mathcal{N}_i$.

When specifying a lower precision ρ, resulting in a coarser approximation, some springs that connect points, which are close in the high-dimensional space will be missing and instead distant points will be connected. This will result in a false repulsion between the points missing a connecting spring. Using P^A reduces the quality of the embedding but improves its computation time by several orders of magnitude. However, reasonable results can be achieved even with low precision, because each data point is usually connected to a large number of springs and, therefore, the overall structure can be preserved. This observation holds for local as well as global structures. Intuitively, even if two points are no longer connected, they might share a common neighbor, which indirectly connects both.

Fig. 2 shows the embeddings generated using different precision values ρ for the computation of the high-dimension similarities. We use the whole MNIST dataset as the input and we color each data-point accordingly to the digit it represents for validation purposes. Fig. 2b shows the embedding generated with the exact neighborhood, whereas Fig. 2c shows the embedding generated with a precision of $\rho = 0.34$. It can be seen that similar structures are preserved using approximated neighborhoods. Fig. 2e shows the embedding generated with $\rho = 0.07$. Even though the embedding visually differs from the exact embedding, depicted in Fig. 2b, the overall clustering of the data is preserved rather well, whilst the time needed for the computation of the similarities is greatly reduced. Where the original algorithm needs 3191 seconds for the initialization using a precision of $\rho = 0.34$ we can achieve a speedup of two orders of magnitude, resulting in a computation time of 30 seconds. By using a precision of $\rho = 0.07$, it is further reduced to 13 seconds.

4.2 Approximated KNN

We achieve different levels of precision by means of different parameterizations of an approximated KNN algorithm called *Forest of Randomized Kd-Trees*. In this section, we describe this technique and how its parameters can be mapped to the precision ρ.

When the dimensionality of the data is high, there are no exact KNN algorithms performing better than linear search [36]. Therefore, the development of approximated KNN algorithms is needed to deal with high-dimensional spaces. A survey on existing algorithms, including an extensive set of experiments, can be found in the work of Muja et al. [37]. In this work, we use a space partitioning technique called *Forest of Randomized KD-Trees* [38] to compute the approximated neighborhoods. This technique has proven to be fast and effective in querying of high-dimensional spaces [36]. A KD-Tree [22] is a binary tree used to partition a k-dimensional space. Each node in the tree is a $k-1$ dimensional hyper-plane, orthogonal to one of the initial k-dimensions, that splits the space into two half spaces. The recursive splitting creates a hierarchical partition of the k-dimensional space.

In a *Forest of Randomized KD-Trees*, a number $\mathcal{T}$ of KD-Trees are generated. The splitting hyper-planes are selected by splitting along a randomly selected dimension among the $\mathcal{V}$ dimensions characterized by the highest variance. A KNN search is computed on all $\mathcal{T}$ KD-Trees, while a maximum number of leaves $\mathcal{L}$ are visited. A priority-queue, ordered by increasing distances to the closest splitting hyper-plane, is used to decide which nodes must be visited first across the forest. The process is stopped when the necessary number of leaves have been evaluated. The parameterization of the Forest of Randomized KD-Trees can overburden the typical end user. To hide this complexity, we integrate the work by Muja et al. [36] and expose only the single precision parameter ρ to the user. The parameters $(\mathcal{T}, \mathcal{V}, \mathcal{L})$ used for the creation and querying of the *Forest of Randomized KD-Trees* are then generated automatically, as described by Muja et al. [36].

4.3 Steerability

A-tSNE is computationally steerable [21], in the sense that the user can define the level of approximation to specific, interesting areas. In this section, we present the changes we made to the BH-SNE algorithm to allow for the refining of the approximation.

The refinement that we propose is done by computing the exact neighborhood for one point at a time. This process leads to a mix of exact and approximated neighborhoods. For each updated neighborhood, a Gaussian distribution P_i is computed and the sparse joint-probability distribution P^A must be updated accordingly. This update, however, is not straightforward. First, the symmetrization of P^A in Eq. 2 requires to combine Gaussian distributions enforced by different data-points and, second, the sparse nature of the distribution P^A renders fast updates challenging.

We solve these issues by observing that a direct computation of P^A can be avoided and the distribution can be indirectly obtained using the Gaussian distributions enforced by the K-Nearest Neighbors. Eq. 2 can be split into two components which correspond only to the Gaussian distributions P_i and P_j:

$$p_{ij} = \frac{p_{j|i}}{2N} + \frac{p_{i|j}}{2N}. \tag{11}$$

Using this formulation, we only need to store one Gaussian distribution per point. Therefore, points can be handled individually without any performance loss. This allows us to execute the refinement of the high-dimensional similarities in parallel to the gradient descent, and serves as the base for the manipulation of the high-dimensional data. Furthermore, we are not constrained to updating the neighborhood of a data-point just once. The analyst can request different levels of approximation for a given area before starting the computation of the exact high-dimensional similarities. For each data-point we store ρ_i as the requested precision for the neighborhood $\mathcal{N}_i$.

A change in a neighborhood, however, yields a change in the cost function C, see Eq. 1, which we are minimizing. To avoid the risk of getting stuck in a local minimum during the gradient descent, we introduce an optimization strategy called *Selective Exaggeration with Exponential Decay*.

Our strategy is inspired by the optimization strategy called *Early Exaggeration* presented by van der Maaten et al. [4]. The idea of *Early Exaggeration* is that, by exaggerating the attractive forces, see Eq. 7, by a factor τ during the first I_τ iterations of the gradient descent, local minima can be avoided. Using the *Selective Exaggeration with Exponential Decay*, we apply an exaggeration τ to the attractive forces acting on a data-point x_i when it is refined. The exaggeration is then smoothly removed on a per-point basis using an exponential decay of the exaggeration factor. This can be interpreted as a localized reinitialization of the gradient descent triggered by user interaction with the embedding.

4.4 Performance and Accuracy Benchmarking

In this section, we present a detailed performance analysis of A-tSNE compared to BH-SNE using several standard benchmark datsets. All performance measurements were obtained using a DELL Precision T3600 workstation with a 6-core Intel Xeon E5 1650 CPU @ 3.2GHz, 32GB RAM and a NVIDIA GTX 680. We apply the same preprocessing steps as presented by van der Maaten [5], without applying a preliminary dimensionality-reduction by means of a Principal Component Analysis. We use the MNIST dataset [34] (60k data-points, 784 dimensions), the NORB dataset [39] (24300 data-points, 9216 dimensions), the CIFAR-10 dataset [40] (50k points, 1024 dimensions) and the TIMIT dataset [41] (1M data-points, 39 dimensions). Throughout the experiments we used a parameter setup similar to the one used to benchmark the BH-SNE [5] and a fixed perplexity value of $\mu = 30$. First, we evaluate the performance of A-tSNE in relation to the parameters $(\mathcal{T}, \mathcal{V}, \mathcal{L})$ used in the *Forest of Randomized KD-Trees*, as described in Section 4.2, using three different configurations: $\mathcal{T} = 4\ \mathcal{L} = 1024$, $\mathcal{T} = 2\ \mathcal{L} = 512$ and $\mathcal{T} = 1\ \mathcal{L} = 1$. For all configurations we set $\mathcal{V}$ to 5 as suggested by Muja et al. [36].

The left chart in Fig. 3 shows the comparison of computation times (in logarithmic scale) of the high-dimensional similarities on the MNIST dataset obtained by our technique and by the BH-SNE algorithm. The right chart in Fig. 3 depicts the precision ρ of the neighborhoods. The precision is given by Eq. 10 and it is computed using the exact and the approximated neighborhoods. Generally, our approach generates a good embedding very efficiently for any given dataset we tested. Fig. 2(b-e) show the embeddings generated using the described parameter settings for the MNIST dataset after 1000 iterations. It can be seen that we achieve visually comparable results more than two orders of magnitude faster compared to the BH-SNE implementation.

Fig. 3 shows how the precision decreases when increasing the data size for a fixed parameter setting. The number of leaves (corresponding to data points) to visit, included in the parameter setting, is fixed independently of the data size. When the data size increases the same number of leaves, corresponding to a smaller fraction of the overall data, is visited, causing the lower precision. In general, we can see that with a small reduction in precision, the computation time can be greatly reduced.

Finally, we analyze the error introduced by the approximation of the similarities in the high-dimensional space using the NORB, MNIST and TIMIT datasets. For the results of the CIFAR-10 dataset we refer to the supplemental mate-

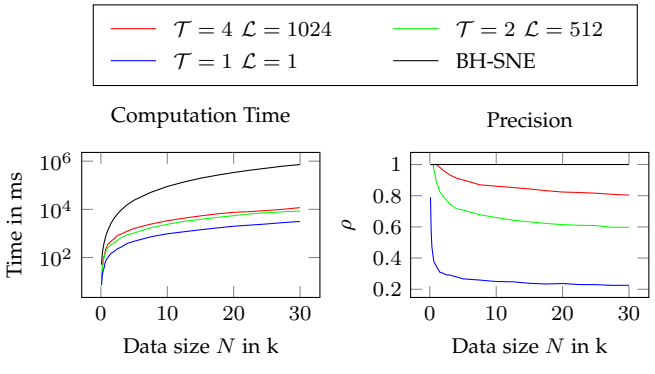


Fig. 3. **Computation time for the high-dimensional similarities** using the MNIST dataset, with BH-SNE and A-tSNE with different parameters (left) and precision with different parameter settings (right).

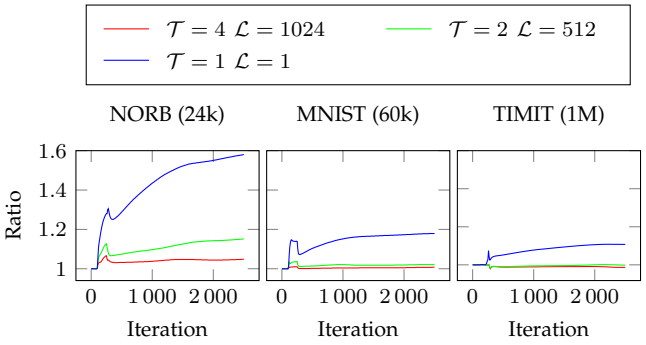


Fig. 4. **Approximated to exact cost ratio** on different datasets of increasing size. When the size of the data increases, the ratio of the approximated cost divided by the exact cost is reduced given the same set of parameters.

rial, as they are very similar to the results obtained on the MNIST dataset. The cost function $C(P, Q)$ is the most direct indication of the quality of the embedding and we compare minimizing of the cost function $C(P, Q^A)$ to $C(P, Q)$. Q^A is the joint-probability distribution that describes similarities in the approximated embedding obtained by the minimization of $C(P^A, Q^A)$. Fig. 4 shows the $C(P, Q^A)/C(P, Q)$ ratio. Smaller values indicate less error, with a value of 1 meaning that no approximation error is present. The *Early Exaggeration* of the attractive forces (see Sec. 4.3) is responsible for the peak in the ratio that is visible during the first 250 iterations. By exaggerating the attractive forces the approximation error is increased. The absolute value of the cost (not depicted in Fig. 4) decreases with every iteration.

The usage of a Forest of Randomized KD-Trees with $\mathcal{T} = 1$ $\mathcal{L} = 1$ generates an embedding with a large error. This configuration is an upper bound of the error and a lower bound in computation time; by visiting only one leaf during the traversal of the forest composed by just one tree, the approximated KNN algorithm becomes a *greedy algorithm*. We can also note that with increased data sizes the approximation error decreases. For the TIMIT dataset we observe that the approximation errors generated using $\mathcal{T} = 2$ $\mathcal{L} = 512$ and $\mathcal{T} = 4$ $\mathcal{L} = 1024$, are similar or better, than the exact one. By increasing the number of points, the effect of the false repulsive forces (Sec. 4.1) is compensated by the increasing number of attractive forces among data-points. The results clearly show that we can rapidly provide very accurate embeddings allowing immediate interaction, without misleading the user. With a large number of data points we effectively generate tSNE embeddings as demonstrated by the reduced approximation error.

5 Interactive Analysis System

Using A-tSNE, the data analysis is started without waiting for the exact computation of the similarities in the high-dimensional space. This operation is the main bottle neck for interactivity, e.g., when data is modified or tSNE parameters are changed by the user. However, the embedding is created based on approximated information. Our system supports three different strategies for the refinement of the approximation, leading to the generation of different and more precise, embeddings.

To steer the refinement, the user must be aware of the error in the embedding. Therefore, we present a visualization that shows the level of approximation.

We also take advantage of the steerability of A-tSNE (Sec. 4.3) to allow for direct manipulation of the high-dimensional data, for example, by adding and removing data-points or by changing the dimensions used to represent the data. Finally, we implemented these techniques in a coordinated multiple-views framework that allows for the direct inspection of the data in the embedding.

5.1 User Steerable Refinement

The refinement process used to steer the computation of an A-tSNE embedding works on a per-point basis, see Sec. 4.3. A naive strategy to refine the embedding, is to progressively update the neighborhoods of all the points in X, while the gradient descent optimization is computed. However, when computational resources are scarce, it makes sense to steer the refinement process to increase precision ρ in areas of the embedding that the analyst finds interesting, e.g., based on initial visual clusters appearing in the embedding. We propose three different strategies that are used to select the data points to be refined: *user selection*, *breadth-first search* and *density-based refinement*. These strategies are presented in the following sections.

5.1.1 User Selection

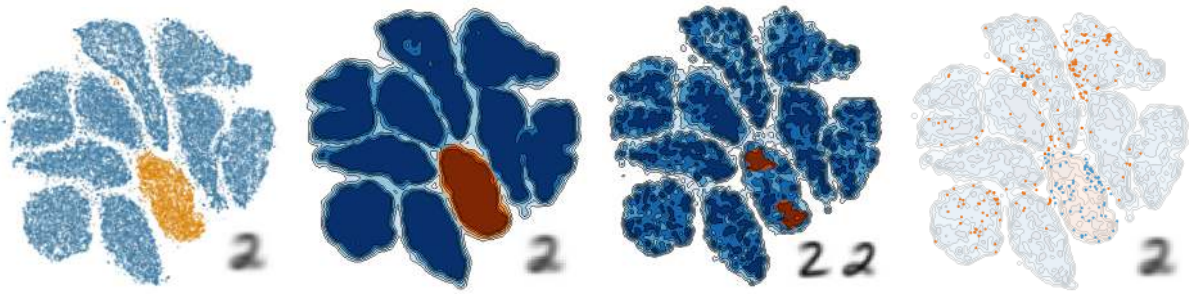
The user selects a subset of points for immediate refinement, by brushing in the embedding. This strategy is less effective when just a few points are selected for refinement, as the forces exerted on its neighbors are still approximated, which can lead to an unfaithful description of the high-dimensional data.

5.1.2 Breadth-First Search

If only a few points are selected for refinement, we extend the process to include their neighborhoods. We use a breadth-first visit on the graph created by the KNN relationships to extend the refinement. When a point is refined, its neighbors are queued for refinement. We also implemented this strategy using a priority queue, where, e.g., points can be prioritized by their euclidean distance to already refined points. This allows better control on the expansion of the refined area at the cost of slower computations introduced by the priority queue.

5.1.3 Density-Based Refinement

When the user is more interested in gaining a global overview of the exact embedding, a density-based refinement strategy is used instead of a local refinement. This strategy is based on the observation that points in the less dense areas of the high-dimensional space, are responsible for the creation of the global relationship in a tSNE embedding [4]. The data-points are refined with an order given by the density in the high-dimensional space, where low-density points are refined first. An indication of this density is the variance σ_i of the Gaussian distribution, as explained in Sec. 3. This strategy works within a user-defined selection or on the whole dataset.



(a) (b) (c) (d)

Fig. 5. **A-tSNE embedding of the MNIST dataset**. (a) uses a point-based visualization with an alpha value of 0.25, the points colored in orange correspond to the digit '2'. (b,c) uses the real-time density-based visualization as described in Sec. 5.2.1. By changing the bandwidth of the kernel density esitmation, clusters at different scales are visible. (d) shows the outliers in the data-points representing the digit '2' by means of a combination of the density-based and the point-based visualization. All figures show the average image of the selected clusters.

5.2 Visualization and Interaction

The visualization of the tSNE embedding provides an overview on the high-dimensional data and should be combined with the ability to inspect the data on demand. In our system, the user selects data points by brushing in a point- or density-based representation of the embedding, the *overview*. We provide specific visualizations of the high-dimensional space using linked views, adaptive to the data at hand. Additionally, we use a magic lens or a full-view overlay to indicate the approximation level. A detailed description of such solutions is given in the following sections.

5.2.1 Density-Based Visualization

The visualization of the embedding, using simple points, is affected by visual clutter when the number of points increases. Density-based [42] visualizations are commonly used to show a tSNE embedding [5], [6], [7], [9] because of their ability to visualize features at different scales. We apply real-time kernel density estimation (KDE) [30] for the creation of an interactive density-based visualization of the embedding. We use changes in the color hue to visualize selections, for example to highlight data points that are selected to be analyzed in other views of the coordinated multiple-view framework. The KDE is computed by assigning a value for each pixel $\mathbf{p}$ using the *kernel density estimator* $f(\mathbf{p}, h)$ as follows:

$$f(\mathbf{p}, h) = \frac{1}{N} \sum_{i=1}^{N} G(||\mathbf{p} - \mathbf{y}_i||, h). \tag{12}$$

$G(d, h)$ is a zero mean Gaussian distribution with standard deviation h, which can be interactively chosen by the user in order to reveal clusters at different scales. Additionally, we introduce a transfer function, mapping $f(\mathbf{p}, h)$ to a color, in order to highlight user-defined selections. Areas with a large percentage of selected points are visualized with a different transfer function, and selection outliers are shown as points. To achieve this goal, we introduce a new kernel density estimator $s(\mathbf{p}, h)$, which illustrates the density of the user selection in a pixel $\mathbf{p}$. Given a set of selected data-points S we use:

$$s(\mathbf{p}, h) = \frac{1}{f(\mathbf{p}, h)} \frac{1}{|S|} \sum_{\mathbf{y}_i \in S} G(||\mathbf{p} - \mathbf{y}_i||, h) \tag{13}$$

If $s(\mathbf{p}, h)$ is higher than a threshold S_{thresh}, a transfer function based on a different hue and with a higher luminance is used. We found empirically that a value $S_{thresh} = 0.5$ performs satisfactorily without compromising the quality of the visualization. We also use a point-based visualization of isolated selected data-points and, unselected data-points in selected regions. Finally, the user can adjust the opacity of the points and the density-based visualization to the needs of the analysis.

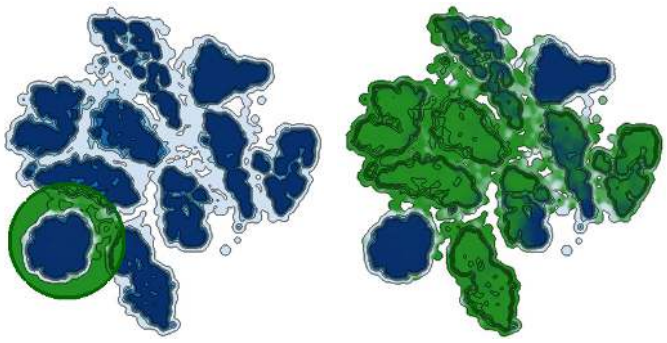
An example of different visualizations of the embedding is presented in Fig. 5, using the MNIST dataset. The analyst can change the bandwidth h, the transfer function, and the opacity interactively in order to show clusters at different scales and outliers in the selection. Fig. 5b shows the selection of a high-level cluster. If a different bandwith is chosen, as in Fig. 5c, clusters at a different level appear. Finally, if the labels are used to make a selection in the embedding, as in Fig. 5d, it is possible to see the distribution of the outliers in the density-based visualization.

5.2.2 Visualization of the Approximation

The complexity of high-dimensional structures, also known as *intrinsic dimensionality*, usually does not allow for an exact representation of the data in 2D. For this reason, it is of crucial importance to integrate the visualization of the embedding with tools that allow to assess its quality. Such an assessment is challenging and several interactive techniques have been developed in recent years [43]. In this work, we are not concerned with the quality of the embedding itself, but rather with the level of approximation introduced by A-tSNE. This information is provided to the user to focus the attention on specific areas of the embedding for a quality analysis, performed with a separate tool.

We enhance our density-based visualization to show the precision ρ_i. Note that ρ_i is different for every data-point and changes during the refinement process, as described in Sec. 4.3. For each pixel $\mathbf{p}$ we assign a value given by the function $a(\mathbf{p}, h)$ that represents the approximation value given the bandwidth h:

$$a(\mathbf{p}, h) = \frac{1}{f(\mathbf{p}, h)} \frac{1}{\sum_{i=1}^{N} \rho_i} \sum_{i=1}^{N} \rho_i G(||\mathbf{p} - \mathbf{y}_i||, h)$$



(a) Magic Lens (b) Full View Mode

Fig. 6. **Visualization of the approximation** in the embedding by means of a magic lens (a) and the full view mode (b).

$a(\mathbf{p}, h)$ is the precision ρ_i weighted kernel-density divided by the kernel-density estimator $f(\mathbf{p}, h)$. The value $a(\mathbf{p}, h)$ is between zero and one and is used directly for encoding the approximation in the visualization.

The value of the function $a(\mathbf{p}, h)$ is visualized in two different ways. First, we introduce a Magic Lens [44] that shows the approximation with a minimal conceal of the data. We use a circular lens that can be overlayed on the density-based visualization and $a(\mathbf{p}, h)$ is used to define the transparency α of every pixel in the lens. To better highlight the refined areas, we use $\alpha = 1 - a(\mathbf{p}, h)^k$, where k is a user selected parameter, to compute α. We provide a default value of $k = 2$.

Fig. 6a shows the lens over a cluster that is already refined and, therefore, is visible through the lens. The green tone indicates the area where similarities are still approximated. Contours in approximated areas are preserved to indicate the structure of the embedding. We color the areas without points in green to put more emphasis on refined areas. In addition to the Magic Lens, we provide the possibility to map approximation to the complete view. This view is especially useful when one of the global refinement strategies is selected as it shows an overview on the refinement process. However it also diminishes the ability to distinguish high-density areas.

Fig. 6b shows the approximation in the embedding using this approach. It is possible to see that two clusters are already refined, relying on exact neighborhood relationships. The user selected a *Breadth-first search* refinement strategy, therefore, the refinement is spreading through the embedding, leading to some areas in the top-right corner having the original color. However the perception of clusters is reduced by removing the color information inside the contours.

5.3 Data Manipulation

In Sec. 4.3, we show that we are able to update high-dimensional similarities between data-points during the gradient-descent minimization. In this section, we take advantage of this possibility, introducing different operations that are used to manipulate the original data-points in their high-dimensional feature space. The embedding does not need to be recomputed but evolves dynamically as the data changes. At the center of an interactive exploration of data is the ability to add or remove data on demand, use different representations of the same dataset or adapt to any changes in the data [19]. For example, the addition and the removal of data points are two fundamental operations that enable us to monitor a high-dimensional stream in real-time.

5.3.1 Inserting Points

For a point $\mathbf{x}_a$, which we want to add to the embedding, its neighborhood $\mathcal{N}_a$ needs to be computed. We compute the neighborhood with the approximated KNN algorithm, as described in Sec 4.2. Finally, we check whether $\mathbf{x}_a$ belongs to the KNN of each point in X. We define d_i^{Max} as the maximum distance between a point $\mathbf{x}_i$ and the points in its neighborhood $\mathcal{N}_i$. The update of the neighborhoods is written as follows:

$$\begin{aligned} &\forall \mathbf{x}_i \in X \text{ if } ||\mathbf{x}_a - \mathbf{x}_i|| < d_i^{\text{Max}} \\ &\quad \text{then } \mathbf{x}_a \in \mathcal{N}_i \text{ and } \mathbf{x}_j \notin \mathcal{N}_i : ||\mathbf{x}_i - \mathbf{x}_j|| = d_i^{\text{Max}} \end{aligned} \tag{14}$$

We cache d_i^{Max}, leading to a complexity for this update of $O(N)$. A priority queue is used to efficiently update d_i^{Max} after the insertion of $\mathbf{x}_a$ in a given neighborhood $\mathcal{N}_i$. It is important to observe that the insertion of $\mathbf{x}_a$ in $\mathcal{N}_i$ will not reduce the estimated precision ρ_i. The initial position in the embedding $\mathbf{y}_a$ is given by the average position of its neighbors $\mathcal{N}_a$ weighted by their similarity $p_{j|i} : \mathbf{x}_j \in \mathcal{N}_i$. The new point $\mathbf{x}_a$ is then added in the *Forest of Randomized KD-Trees*. This operation is performed in $O(\log(N))$.

5.3.2 Deleting Points

Removing a point $\mathbf{x}_r \in X$ is performed by deleting $\mathbf{x}_r$ from the KNN of every point $\mathbf{x}_i \in X$. This operation has a computational complexity of $O(N)$. By removing $\mathbf{x}_r$ from a neighborhood $\mathcal{N}_i$ we reduce the number of $\mathbf{x}_i$ neighbors to $K - 1$ and a new neighbor must be found to maintain the precision level. However, the new point in the neighborhood is the most dissimilar of the points in $\mathcal{N}_i$ thus its attractive force is rather small and we propose to ignore the contribution of the missing point, decreasing the estimated precision ρ_i by $1/K$. To avoid degeneracies, when the size of the neighborhood $\mathcal{N}_i$ goes below a given threshold, e.g., $K/2$, the neighborhood is updated using approximated computations. The *Forest of Randomized KD-Trees* is updated in $O(\log(N))$.

5.3.3 Data Modification

The insertion and deletion of data points enables a new way of analyzing data changes, for example, changes in time. New data points are added to the embedding when ready and old ones are removed in real-time. However, data that are already present in the embedding can change over time and must be updated accordingly. We handle changes in the value of a single high-dimensional data-point by a combination of removal and addition operations. A different modification of the data is performed not by changing the values of single data points, but by changing the dimensions of the data itself. Examples of this operation are the addition or the removal of dimensions to inspect the influence of a given dimension in the generation of visual clusters. With

such a modification, all the data points in X change their position in the high-dimensional space. Therefore, all the neighborhoods must be reconsidered and it is more convenient to compute a new approximated joint-probability distribution P^A. When the distribution P^A is changed, the function that is to be minimized by the gradient descent also change, see Eq. 1. To avoid local minima, we apply the *Selective Exaggeration with Exponential Decay*, see Sec. 4.3, to all the data points. After such an operation, the user expects to see major changes in the embedding, where the extent of such modifications gives information about the differences of the new representation to the old one.

5.4 Visual Analysis Tool

We implemented A-tSNE as a module in an integrated, interactive, multi-view system for the analysis of high-dimensional data. Fig. 7 shows a screenshot of the system and its different views. The interface is divided into two main areas. At the top, three different views are used to show the intermediate embeddings (7a), the data (7b) and the state of refinement processes (7c), respectively. Controls are at the bottom of the interface: (7d) for the generation of intermediate embeddings, (7e) visualization of the embedding, (7f) data manipulation and (7g) refinement.

The data subject to the analysis are visualized in the *Data View* (7b). Selections in the embeddings are reflected in the *Data View* with strategies that depend on the data type. We implemented multiple widgets that are used to support the analysis process of different data types. These widgets include a heatmap view, a 3D volume view (7b bottom) and an image view (7b top row). If necessary multiple and different views are combined for the analysis.

The *Refinement-Status View* (7c) is used to give an overview of the progress of the refinements started by the user. The user can steer the evolution of the embedding by refining areas with strategies as described in Sec. 5.1. A refinement process is identified by the snapshot of the embedding when the user started the refinement, a user-defined description, and a progress bar that shows the percentage of the refined data-points over the selected ones.

5.5 Implementation

We implemented the system using a combination of C++ and Qt, as well as OpenGL with custom shaders in GLSL for the visualization of the embedding. Where possible, we used parallel computations with OpenMP. The approximated neighborhoods are computed using the FLANN library [36], which implements KNN algorithms. The density-based visualization is computed on the GPU using OpenGL and GLSL shaders. A precomputed floating-point texture is generated using a Gaussian kernel. A geometry shader is used to generate a quad for each point that is colored using the precomputed texture, the KDE is obtained by drawing into a Frame Buffer Object using an additive blending [30].

6 Case Study I: Exploratory Analysis of Gene Expression in the Mouse Brain

In this section, we demonstrate the advantages of using A-tSNE in our visual analysis tool for the visual analysis of high-dimensional data. To this extent, we present a case study, based on the work by Mahfouz et al. [8], who use tSNE to explore the Allen Mouse Brain dataset [45]. The dataset is composed by 61164 voxels obtained by slicing the mouse brain in 68 slices. Each voxel is a 4345-dimensional vector, containing the genetic expression at the corresponding spatial position. tSNE is computed using the voxels as data-points and the expression of the genes as high-dimensional space. Mahfouz et al. discuss the hypothesis that genetic information can be used to differentiate anatomical structures in the brain. Some regions in the brain, e.g. the Cerebellum, are known to have a highly different genetic footprint compared to the rest of the brain. They demonstrate that tSNE is effective in separating different anatomical structures, e.g. white and grey matter, only based on the genetic footprint.

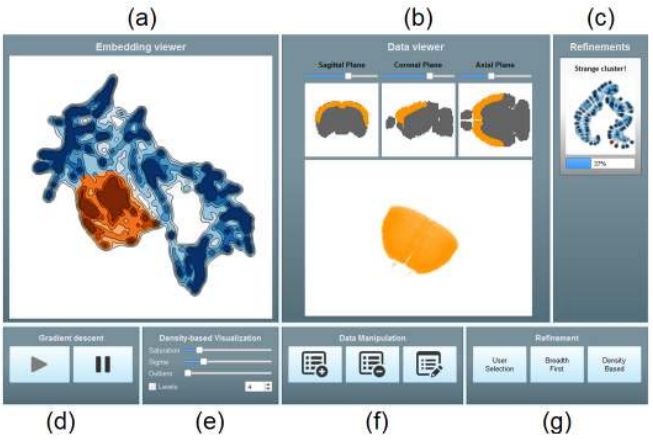


Fig. 7. **Screenshot of our integrated system** using multiple linked views for interaction. The system comprises an embedding viewer (a), a data viewer (b) and a refinement viewer (c). Controls on the gradient descent (d), the density-based visualization (e), the data-manipulation (f) and the refinements (g) are at the bottom of the interface.

Fig. 8 depicts the typical analytic workflow using our visual analysis tool. The first goal during the analysis is to validate the input data. The acquisition process may not be perfect, data can be incomplete or noisy, therefore, it must be re-acquired or preprocessed before interesting results can be generated. Driven by the need to validate the data as soon as possible, the user selects a reasonably low value for the desired precision, e.g. $\rho = 0.2$, that will be used to estimate the parameters of the KNN algorithm. With such a parameterization, A-tSNE computes the high-dimensional similarities in ≈ 51 seconds while 3 hours and 50 minutes are required by BH-SNE.

The user then analyzes the intermediate embeddings, produced by A-tSNE, in order to validate the input data. After ≈ 170 seconds several clusters become visible in the embedding as depicted in Fig. 8a. The clusters are stable for several iterations indicating that they are not an artifact of the minimization process. The user can validate this by selecting the clusters in the embedding and can inspect them in more detail, for example, by highlighting their spatial positions in the feature view, see Fig 8a. Points or clusters are selected by brushing in the embedding. During a brushing operation the generation of intermediate embeddings is

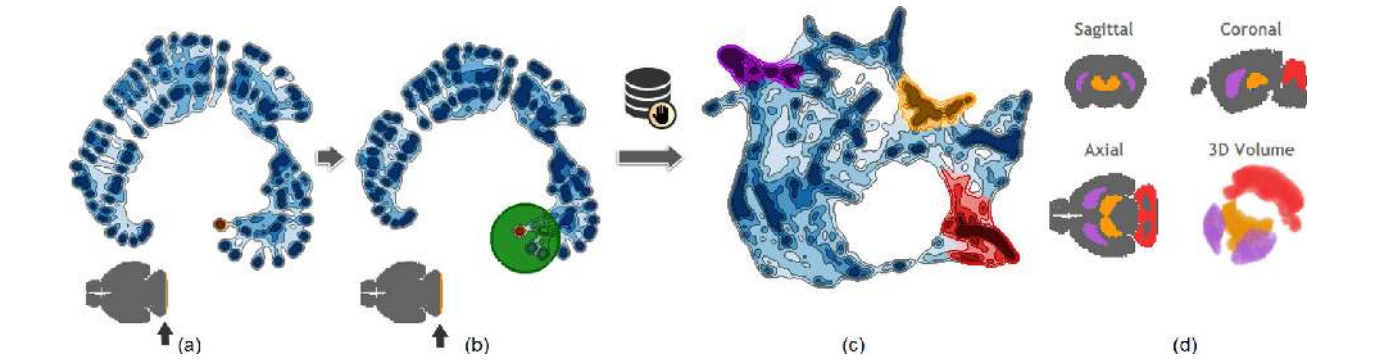


Fig. 8. **Analysis of the gene expression in the mouse brain using A-tSNE**. The first embedding (a) is generated in ≈ 51 seconds while 3 hours and 50 minutes are required by BH-SNE. The analyst inspects a cluster and finds that it corresponds to a slice in the data. The cluster does not disappear after the neighborhoods are refined, as shown by the lens in (b). A change in the high-dimensional data reveals that genetic information can be used to differentiate anatomical regions. (c) shows the final embedding based on a small number of Principal Components where three clusters are highlighted and (d) shows the corresponding regions in the brain.

stopped to make sure the user does not accidentally brush areas as they change. Selected points are then highlighted by a change of hue, in this case from blue to orange. Further inspection using the *Data View* in our interactive system, shows that each cluster corresponds to a slice in the dataset. Fig. 8a shows a cluster, highlighted in orange, and the corresponding slice in the volume.

To make sure the clusters are not an artifact introduced by the approximated similarities, the user refines the selected data-points while the embedding evolves. Fig. 8b shows the embedding after the refinement is complete. Note that the global structure of the embedding does not change during the refinement. Changes are constrained to the selected cluster, giving to the user a sense of stability in the information provided as requested by the Progressive Visual Analytics paradigm. The user can inspect the degree of approximation in the embedding using the interactive lens. The lens is less transparent over approximated areas of the embedding and transparent on the areas that contain no approximation. After the refinement of the high-dimensional similarities of the selected data points, the clusters do not disappear, which indicates that clustering is indeed driven by the data, rather than by the approximation.

Therefore, the user stops the computation of the fully refined embedding. Further analysis performed by domain experts on the raw data reveals that missing values in the input data cause the formation of small clusters in the embedding. Mahfouz et al. removed this effect by using the first 10 components, extracted by a Principal Component Analysis of the raw data, as the high-dimensional space. In the traditional analytical workflow, after the high-dimensional data are changed, a new tSNE embedding is computed from scratch. However, in our system the user directly changes the high-dimensional space and the current embedding evolves accordingly. Given that the gradient descent is minimizing a different function, the user expects structural changes that can be considerably large, see Sec. 5.3.3. The extent of these changes provides information about the modification in the high-dimensional space. If the embedding is stable, the new high-dimensional representation preserves relationships between data points, while an abrupt change means that new relationships are encoded in the data. In the traditional workflow without A-tSNE, any continuity and the encoded information are lost. Approximately 200 seconds after the change in the high-dimensional data, a stable embedding is obtained. Fig. 8c shows the final embedding, where three different clusters are highlighted. Fig. 8d depicts the selected voxels in the brain, note how the anatomical structures are now revealed. It is possible to see how the clusters that were present in the first intermediate results disappear, showing that the cluster fragmentation is removed.

Voxels that belong to the same anatomical structure are close together in the embedding. A-tSNE is able to separate anatomical structures based on the gene expression of the 4345 genes. In their work, Mahfouz et al. [8] present embeddings created using 2, 3, 5, 10, and 20 principal components as the high-dimensional space. Identifying the right number of components is a time consuming task and the adoption of our analytic workflow helps the user in finding a good compromise by interactively analyzing the resulting embedding generated changing the number of components.

7 Case Study II: Real-time monitoring of high-dimensional streams

Improved computation time and the ability to modify data are the key for applying tSNE in new application scenarios, such as the real-time monitoring of high-dimensional data streams. The original tSNE algorithm fails in providing a solution for such applications. The computation of a tSNE map imposes a time constraint that cannot be ignored, when the rate in which new data is generated is higher than the time required for the computation of a tSNE map.

As proof of concept, we selected a dataset for physical activity monitoring [46] that comprises readings of three Inertial Measurement Units (IMU) and a heart rate monitor applied to 9 different subjects. Every IMU generates 17 readings every 10 ms, while the heart rate monitor generates one reading every 100 ms. Taking all sensors into account, we have a stream of data consisting of 52 readings, where a new data point is generated every 100 ms for each subject. Every subject also has a device to label the physical activity. We use the labeling of every reading to validate the insights obtained by the analysis of the embeddings.

We analyze the stream of a subject by keeping the readings of the previous $\mathcal{M}$ minutes in the embedding

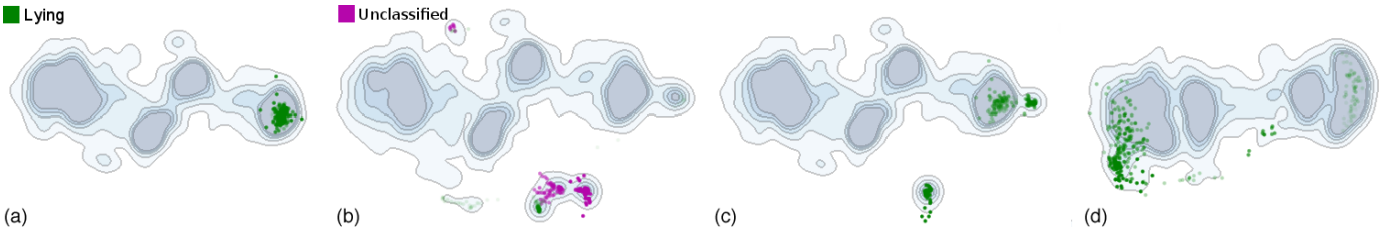


Fig. 9. **A-tSNE used for the real-time analysis of high-dimensional streams**. The embeddings are generated using the readings of the last 10 minutes. As new readings arrive they are inserted in the embedding and they are highlighted using a point-based visualization. (a) shows the initial embedding, the color of the data-points indicates that the subject is lying down. The embedding evolves as in (b), a new cluster indicates readings of a different activity. This insight is confirmed by a change in the color of the data-points that indicates a new type of label activity. (c) shows an evolution of the embedding presented in (a) where new readings are generated from a miscalibrated sensor and, therefore, are clustered together. By removing the features corresponding to the miscalibrated sensor the embedding evolves as in (d). The cluster that identifies miscalibrated readings is removed.

with a fixed approximation level. When a new reading is generated, we add it to the embedding using the technique described in Sec. 5.3. Similarly, when a reading is older then $\mathcal{M}$ minutes, we remove it from the embedding. In the test presented in this section, $\mathcal{M} = 10$ is set leading to an embedding composed, in average, by 6000 data-points that is updated every 100 ms. We add a point-based visualization to our density-based visualization, which shows the last points inserted in the embedding. The new points are colored according to the classification of the activity made by the subject and they will fade out in $\mathcal{F}$ seconds. By showing the new data-points the analyst can identify where new points are added, providing at the same time an overview of the embedding in the last $\mathcal{M}$ minutes and the trend of the last $\mathcal{F}$ seconds.

Fig. 9a shows an embedding obtained from *subject 105*, where the color of the data-points, green in this specific case, indicates that the subject is lying down. The embedding is composed of a single big cluster that represent the *lying down activity*. The cluster is divided in four different sub-clusters that identify different readings of the sensors. The readings of the last 30 seconds belong to a single sub-cluster and can be seen as points on the right side of the embedding. The embedding evolves based on new readings from the sensors, after few seconds the new data-points start to be placed further away from the original cluster, leading to the creation of a new cluster, as depicted in Fig. 9b. After a few seconds the subject changes the classification of his activity from lying down to an *unclassified activity*, whose corresponding data-points are colored in purple. It is interesting to note that, simply by looking at the embedding, it is possible to predict a change in the labeled activity before the subject is able to record the change on his labeling device. It can be seen by the fact that few data-points labeled as a *lying down activity*, hence colored in green, are in the same cluster as the ones identified as *unclassified activity*. In this particular case, we can guess that the subject sat up before changing the labeled activity.

Finally, we simulated a miscalibration in an inertial measurement unit. Differently from a faulty sensor (not generating any readings), a miscalibrated one generates readings affected by a constant offset that is different for every dimension. We simulate the miscalibration by enforcing a random offset to the readings generated by one of the IMUs. A miscalibrated sensor generates readings that are different from the normal one and, therefore, they should be clustered together as faulty readings. Fig. 9c shows the evolution of the embedding presented in Fig. 9a where the miscalibrated readings are grouped by A-tSNE. After the inspection of the readings generated from the IMUs, the analyst can identify that something is wrong with one of the sensors. At this point the sensor may be replaced or, in case this is not possible, the readings from the miscalibrated sensor can be excluded by removing the corresponding dimensions from the high-dimensional space, as presented in Sec. 5.3.3. Such an update requires a few seconds in which the embedding is updated in order to encode the new relationship in the high-dimensional space. Fig. 9d shows how the previous embedding evolves when the readings generated by the miscalibrated sensor are removed from the high-dimensional space. It is possible to see that the readings affected by the miscalibration are now close to the cluster that represents the *lying down activity*. However, differently from the test case presented in Sec. 6, the global structure of the embedding is preserved, still showing four different clusters. Liu et al. [47] demonstrate that, when dealing with real-time data, the response time of the algorithm is of great importance to the user. In the presented case study, we reach real-time performance for a limited data size for the sliding window of 6000 points. However, it should be noted that when the sampling rate or the window size of the stream is much larger, A-tSNE also will not be able to handle the data in real-time in all cases.

8 Discussion and Conclusions

Motivated by the need of interactivity in Visual Analytics, we present Approximated-tSNE. A-tSNE enables the rapid generation of approximate tSNE embeddings. We use fast approximated KNN queries for the computation of the high-dimensional similarities. Our algorithm is designed to be used within the Progressive Visual Analytics context, allowing the user to have a quick preview of the data. Insight obtained using approximated embeddings can be validated by refining the approximation in interesting areas with different strategies. Therefore, we present different visualization techniques for the level of approximation, which are used to guide the user during the refinement process in Sec. 4.3. It should be noted, that the level of approximation is only an indicator for how well the approximated embedding represents the exact embedding. It cannot, however, be used to judge the quality of the embedding itself, as even an exact

embedding might not represent the original data perfectly. The quality of the embedding itself can be analyzed, e.g. by inspecting the preservation of k-nearest-neighborhoods [43]. The full precision of BH-SNE can always be reached by setting the precision parameter accordingly, or refining the data. Therefore, A-tSNE can effectively replace BH-SNE for the analysis of dense high-dimensional data. However, A-tSNE cannot outperform algorithms such as Q-SNE in the analysis of sparse high-dimensional data.

The refinement of the approximation itself is a stable process. As demonstrated in Sec. 4.4 and Fig. 2, P^A is close to P if a reasonable parameterization is chosen. As a result gradually refining P^A will lead to small changes in the embedding, only. In addition, we present three different operations for the direct manipulation of the high-dimensional data. *Addition* and *removal* of data-points are mainly aimed at the inspection of high-dimensional streams. *Data modification* is used to visualize different models of the same data. Different from the refinement process, changing the model might lead to drastic changes in P^A (as it would in P) and as such might also create a very different embedding. We chose to start the optimization with the embedding created before changing the model. As a result points in the embedding might move drastically during the optimization process. While this might be confusing and less adequate for Progressive Visual Analytics, the amount of movement is related directly to the strength of the changes and as such is a very good indicator of the influence of the parts of the data that were modified on the whole embedding.

We presented two case studies to show the effectiveness of A-tSNE. *Case Study I* shows a typical analysis of a static dataset. In such a setting it is crucial to allow an interactive feedback loop, between modeling the data (i.e., finding the right number of dimensions for the PCA before embedding) and visualizing the data. Even though, we do not achieve real-time performance, we are able to drastically cut computation times, i.e., from four hours to less than a minute, allowing such interactive exploration of the data. *Case Study II* shows an example for the monitoring and analysis of streaming data. Here it is crucial to achieve real-time performance. We use efficient addition and removal of data points (see Sec. 5.3) to visualize a temporal sliding window of the data. As discussed in Sec. 7 even the large increase in performance provided by A-tSNE does not allow real-time analysis of large data. This work has inspired our contribution on the hierarchical exploration of large high-dimensional data [48]. We believe that this example illustrates as well, that real-time feedback can be important for data analysis. In the future we want to explore the application of A-tSNE in other research scenarios. In particular, we are interested in investigating the application of A-tSNE in the analysis of heterogeneous data and different high-dimensional streams, such as climate readings.

REFERENCES

[1] A. Inselberg and B. Dimsdale, "Parallel coordinates," in *Human-Machine Interactive Systems*. Springer, 1991, pp. 199–233.

[2] J. A. Hartigan, "Printer graphics for clustering." in *Journal of Statistical Computing and Simulation*, 1975, pp. 187–213.

[3] L. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," pp. 66–71, 2008.

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

[5] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, pp. 3221–3245, 2014.

[6] E.-a. D. Amir, K. L. Davis, M. D. Tadmor, E. F. Simonds, J. H. Levine, S. C. Bendall, D. K. Shenfeld, S. Krishnaswamy, G. P. Nolan, and D. Pe'er, "viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia," *Nature biotechnology*, vol. 31, no. 6, pp. 545–552, 2013.

[7] B. Becher, A. Schlitzer, J. Chen, F. Mair, H. R. Sumatoh, K. W. W. Teng, D. Low, C. Ruedl, P. Riccardi-Castagnoli, and M. Poidinger, "High-dimensional analysis of the murine myeloid cell system," *Nature immunology*, vol. 15, no. 12, pp. 1181–1189, 2014.

[8] A. Mahfouz, M. van de Giessen, L. van der Maaten, S. Huisman, M. Reinders, M. J. Hawrylycz, and B. P. Lelieveldt, "Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings," *Methods*, vol. 73, pp. 79–89, 2015.

[9] K. Shekhar, P. Brodin, M. M. Davis, and A. K. Chakraborty, "Automatic classification of cellular expression by nonlinear stochastic embedding (ACCENSE)," *Proceedings of the National Academy of Sciences*, vol. 111, no. 1, pp. 202–207, 2014.

[10] C. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, no. 12, pp. 1653–1662, Dec 2014.

[11] T. Mühlbacher, H. Piringer, S. Gratzl, M. Sedlmair, and M. Streit, "Opening the black box: Strategies for increased user involvement in existing algorithm implementations," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, no. 12, pp. 1643–1652, Dec 2014.

[12] J. Choo, H. Kim, C. Lee, and H. Park, "PIVE: A per-iteration visualization environment for supporting real-time interactions with computational methods," *Visual Analytics Science and Technology (VAST), 2014 IEEE Symposium on*, 2014.

[13] P. Bruneau, P. Pinheiro, B. Broeksema, and B. Otjacques, "Cluster sculptor, an interactive visual clustering system," *Neurocomputing*, vol. 150, pp. 627–644, 2015.

[14] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov *et al.*, "DeViSE: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems*, 2013, pp. 2121–2129.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 580–587.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[17] M. Brehmer, M. Sedlmair, S. Ingram, and T. Munzner, "Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences," in *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*. ACM, 2014, pp. 1–8.

[18] Z. Yang, J. Peltonen, and S. Kaski, "Scalable optimization of neighbor embedding for visualization," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 127–135.

[19] J.-D. Fekete, "Visual analytics infrastructures: From data management to exploration," *Computer*, vol. 46, no. 7, pp. 22–29, July 2013.

[20] D. Fisher, I. Popov, S. Drucker, and m. Schraefel, "Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12, 2012, pp. 1673–1682.

[21] J. D. Mulder, J. J. van Wijk, and R. van Liere, "A survey of computational steering environments," *Future generation computer systems*, vol. 15, no. 1, pp. 119–129, 1999.

[22] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[23] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *Proceedings of the fourth*

annual ACM-SIAM Symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 1993, pp. 311–321.

[24] M. Williams and T. Munzner, "Steerable, progressive multidimensional scaling," in *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*. IEEE, 2004, pp. 57–64.

[25] T. Yang, J. Liu, L. McMillan, and W. Wang, "A fast approximation to multidimensional scaling," in *Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV)*, 2006, pp. 354–359.

[26] P. Joia, F. Paulovich, D. Coimbra, J. Cuminato, and L. Nonato, "Local affine multidimensional projection," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2563–2571, 2011.

[27] F. V. Paulovich, C. T. Silva, and L. G. Nonato, "Two-phase mapping for projecting massive data sets," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 6, pp. 1281–1290, 2010.

[28] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz, "Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 3, pp. 564–575, 2008.

[29] S. Ingram and T. Munzner, "Dimensionality reduction for documents with nearest neighbor queries," *Neurocomputing*, vol. 150, pp. 557–569, 2015.

[30] O. Lampe and H. Hauser, "Interactive visualization of streaming data with kernel density estimation," in *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, 2011, pp. 171–178.

[31] J. Choo, H. Lee, J. Kihm, and H. Park, "iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction," in *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*. IEEE, 2010, pp. 27–34.

[32] S. J. Aarseth, *Gravitational N-Body Simulations*. Cambridge University Press, 2003, cambridge Books Online.

[33] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 4, pp. 446–449, 1986.

[34] Y. LeCun, C. Cortes, and C. J. Burges. (1999) The mnist database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[35] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[36] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, 2009, pp. 331–340.

[37] M. Muja and D. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 11, pp. 2227–2240, Nov 2014.

[38] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[39] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. 97–104.

[40] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

[41] F. Sha and L. K. Saul, "Large margin gaussian mixture modeling for phonetic classification and recognition," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 1. IEEE, 2006, pp. I–I.

[42] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[43] R. M. Martins, D. B. Coimbra, R. Minghim, and A. Telea, "Visual analysis of dimensionality reduction quality for parameterized projections," *Computers & Graphics*, vol. 41, pp. 26–42, 2014.

[44] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann, "A survey on interactive lenses in visualization," *EuroVis State-of-the-Art Reports*, pp. 43–62, 2014.

[45] E. S. Lein, M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, A. F. Boe, M. S. Boguski, K. S. Brockway, E. J. Byrnes *et al.*, "Genome-wide atlas of gene expression in the adult mouse brain," *Nature*, vol. 445, no. 7124, pp. 168–176, 2007.

[46] A. Reiss and D. Stricker, "Creating and benchmarking a new dataset for physical activity monitoring," in *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2012.

[47] Z. Liu and J. Heer, "The effects of interactive latency on exploratory visual analysis," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 20, no. 12, pp. 2122–2131, 2014.

[48] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova, "Hierarchical stochastic neighbor embedding," *Computer Graphics Forum (Proc. EuroVis)*, 2016.

Nicola Pezzotti received the Laurea Magistrale in Ingegneria Informatica (MSc) from the University of Brescia, Italy, in 2011. Previously he worked as a research fellow at University of Brescia and as a research and development engineer in Open Technologies Srl. He is a PhD student at Delft University of Technology in the Computer Graphics and Visualization group. His research interests include visualization, visual analytics and machine learning. He is a member of IEEE.

Boudewijn Lelieveldt received a PhD in medical image analysis from the Leiden University in 1999. He is heading the Division of Image Processing (www.lkeb.nl) at the Leiden University Medical Center, and hold a Medical Delta professor chair of Biomedical Imaging at Leiden University and Delft University of Technology. His research interest includes dimensionality reduction methods, with application in complex biomedical datasets. He is a member of IEEE.

Laurens van der Maaten is an Assistant Professor in computer vision and machine learning at Delft University of Technology, The Netherlands. Previously, he worked as a postdoctoral researcher at University of California, San Diego, as a PhD student at Tilburg University, and as a visiting PhD student at the University of Toronto. His research interests include deep learning, deformable template models, dimensionality reduction, data visualization, classifier regularization, and tracking.

Thomas Höllt received the Diplom (MSc) from the University of Koblenz-Landau, Germany, in 2008, and the PhD in computer science from the King Abdullah University of Science and Technology, Saudi Arabia, in 2013. He is a Postdoctoral fellow at Delft University of Technology. His research interests include visualization, computer graphics and GPGPU. He is a member of IEEE and Eurographics.

Elmar Eisemann is a professor at Delft University of Technology, heading the Computer Graphics and Visualization group. Before, he was an associate professor at Telecom ParisTech and senior researcher in the Cluster of Excellence at MPII/Saarland University. His interests include real-time and perceptual rendering, alternative representations, shadow algorithms, global illumination, and GPU acceleration techniques. In 2011, he was honored with the Eurographics Young Researcher Award.

Anna Vilanova is associate professor at the Delft University of Technology in the Computer Graphics and Visualization group. Before, she was assistant professor at the Eindhoven University of Technology. She is leading a research group in the subject of multivalued image analysis and visualization. Her research interests include visual analytics, medical visualization, volume visualization, multivalued visualization, and medical image analysis.