Computers & Graphics ([redacted])

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Fast and reliable incremental dimensionality reduction for streaming data

Tácito T. A. T. Neves[1,*], Rafael M. Martins[2], Danilo B. Coimbra[3], Kostiantyn Kucher[4], Andreas Kerren[5,6], Fernando V. Paulovich[7]

ARTICLE INFO

Article history:
[redacted]

Incremental Dimensionality Reduction, Streaming Dimensionality Reduction, Multidimensional Projection

ABSTRACT

Streaming data applications are becoming more common due to the ability of different information sources to continuously capture or produce data, such as sensors and social media. Although there are recent advances, most visualization approaches, particularly Dimensionality Reduction (DR) techniques, cannot be directly applied in such scenarios due to the transient nature of streaming data. A few DR methods currently address this limitation using online or incremental strategies, continuously updating the visualization as data is received. Despite their relative success, most impose the need to store and access the data multiple times to produce a complete projection, not being appropriate for streaming where data continuously grow. Others do not impose such requirements but cannot update the position of the data already projected, potentially resulting in visual artifacts. This paper presents *Xtreaming*, a novel incremental DR technique that continuously updates the visual representation to reflect new emerging structures or patterns without visiting the high-dimensional data more than once. Our tests show that in streaming scenarios where data is not fully stored in-memory, *Xtreaming* is competitive in terms of quality compared to other streaming and incremental techniques while being orders of magnitude faster.

1. Introduction

Prompted by the massive quantity of available data, our society is facing a paradigm shift towards making decisions based on more data-driven processes [1]. Although a positive trend, it presents several challenges for the existing visualization techniques, from visual and computational scalability to latency issues. One particular scenario that presents requirements difficult to fulfill involves streaming data. Compared to more conventional static applications, where data is first recorded in persistent tables to be later processed, in streaming applications, the data needs to be processed as it is received since it is typically discarded after that [2]. Examples include environmental monitoring [3], network intrusion detection [4], and real-time social media analysis [5].

Among the existing visualization techniques devoted to processing significant amounts of data, multidimensional projection or dimensionality reduction techniques [6] are emerging as fundamental tools by allowing the identification and analysis of similarity and neighborhood relationships and patterns [7]. Although many projection techniques can successfully handle large datasets using out-of-sample strategies [8, 9, 10, 11, 12, 7], most are not appropriate for the streaming scenario

*Corresponding author
[1]Tácito T. A. T. Neves is with Federal University of Alagoas, Brazil. E-mail: tacito.neves@arapiraca.ufal.br.
[2]Rafael M. Martins is with Linnaeus University, Sweden. E-mail: rafael.martins@lnu.se.
[3]Danilo B. Coimbra is with Federal University of Bahia, Brazil. E-mail: coimbra.danilo@ufba.br.
[4]Kostiantyn Kucher is with Linnaeus University, Sweden. E-mail: kostiantyn.kucher@lnu.se.
[5]Andreas Kerren is with Linnaeus University, Sweden. E-mail: andreas.kerren@lnu.se.
[6]Andreas Kerren is with Linköping University, Sweden. E-mail: andreas.kerren@liu.se.
[7]Fernando V. Paulovich is with Dalhousie University. E-mail: paulovich@dal.ca.

given its transient nature. A few methods currently address this limitation by using online or incremental strategies, continuously processing data, and updating the visualization. Despite their success, such methods present significant bottlenecks that impair their practical usage. Some methods impose the storage and multiple accesses to all (received) high-dimensional data [13, 14, 15, 16, 17, 18, 19, 20], being appropriate to incremental and progressive scenarios but not streaming where data continuously grow. Others create one projection function at the beginning of the process and use it for all data [21, 22, 23], in general, resulting in low-quality layouts since the function is not adapted to new structures or patterns that emerge over time. Finally, a few methods evolve the projection function to capture new incoming structures and patterns [24, 25, 26, 27, 28, 29, 30]. However, the already projected instances are not updated to comply with the new function, potentially resulting in visual artifacts produced by overlapping subsequent misaligned projections.

In this paper, we present *Xtreaming*, a novel incremental projection technique to address the challenging design conflict of updating over time the position of the already projected instances without storing or revisiting all high-dimensional data that has already been processed. *Xtreaming* combines a change detection approach, an out-of-sample projection technique, and a novel re-projection strategy to update the projection without revisiting the high-dimensional input data. *Xtreaming* is competitive in terms of global distance preservation compared to other streaming and incremental techniques, but it is orders of magnitude faster. In summary, the main contributions of this paper are:

- a precise and fast dimensionality reduction technique, called *Xtreaming* that can be efficiently and effectively used in streaming scenarios where data needs to be processed as received;
- a novel strategy to re-project the already processed data to comply with a new projection function without the need for revisiting the original high-dimensional data, a critical aspect of any technique designed to handle streaming data that is usually discarded after being processing.

The rest of the paper is organized as follows. In Sec. 2, we discuss related work regarding dimensionality reduction techniques for streaming scenarios. In Sec. 3, we formalize the problem and present a complete overview of the *Xtreaming* algorithm and framework. In Sec. 4, we present an extensive set of experiments that attest to the stability, sensibility, and quality of the proposed technique. A case study is described in Sec. 5, showing how *Xtreaming* can be used for sentiment analysis in social media. In Sec. 6 we discuss *Xtreaming* limitations and future work, and the paper is concluded in Sec. 7.

2. Related Work

Dimensionality reduction or multidimensional projection techniques create computational models that map data instances into graphical elements preserving pairwise distances or neighborhoods [6, 31]. One existing classification splits them into two groups, the *in-sample* and the *out-of-sample* techniques [10]. While in-sample techniques produce layouts processing the data instances altogether, the out-of-sample initially select and project a small sample of the dataset and then map the remaining instances interpolating the sample projection.

Regarding overall distance and neighborhood preservation, in-sample techniques usually produce more precise results but incur high computational costs. Some examples include t-SNE [32], classical Multidimensional Scaling (MDS) [33], Sammon's Mapping [34], ISOMAP [35] , and Local Linear Embedding (LLE) [36]. In contrast, out-of-sample techniques can handle much larger datasets in a fraction of the runtime, but with a penalty in precision. Most out-of-sample techniques are approximations of in-sample techniques where the in-sample strategies are used to project an small data sample followed by the interpolation of the remaining instances. Pekalska et al. [37] propose an approximation of Sammon's Mapping. Landmarks MDS (LMDS) [8] and Pivot MDS [38] are approximations of MDS. Landmarks ISOMAP (L-ISOMAP) [9] and Landmarks LLE (L-LLE) [10] are approximations of ISOMAP and LLE, respectively, and Espadoto et al. [11] define a general strategy based on neural networks to approximate any in-sample method.

Recently, some out-of-sample techniques have been developed not as approximations of in-sample methods, but as strategies to incorporate user knowledge in the projection process. Examples include Least Squares Projection (LSP) [39], Piecewise-Laplacian Projection (PLP) [12], and Local Affine Multidimensional Projection (LAMP) [7]. Although out-of-sample techniques can handle large datasets, they are not appropriate for scenarios where (new) data is continuously fed into the process. Since the quality of the out-of-sample techniques typically depends on the initial sample's quality, all data has to be known before the process starts to ensure that the sampling process collects instances that faithfully represent the data distribution. To tackle this limitation, online strategies have been devised, processing data as received.

Basalaj [13] presents an online version of MDS. When a new instance is received in this technique, MDS is applied considering the new instance and the already processed ones, creating a new full pairwise distance matrix. Similarly, Alsakran [14] employs a force-based approach that is updated to consider new instances, also recomputing an in-memory full pairwise distance matrix. Jenkins et al. [15] and Law et al. [16] present online versions of ISOMAP. In both cases, the techniques are focused on updating the (geodesic) distances. When a new instance is received, all instances are processed, and a full pairwise distance matrix is necessary. Law et al. [17] speed-up this process by avoiding the need for a full pairwise matrix by presenting an online version of LMDS. In this version, only the distances between the new instance to all other instances are needed. Kouropteva et al. [18] and Schuon et al. [19] present online versions of the LLE technique by defining strategies to update the neighborhood relationships when a new data instance is received. Recently, Rauber et al. [20] presented Dynamic t-SNE, a method to project windows of time-dependent datasets while keeping the spatial coherence across projections. Despite the

interesting results, to project the last received data (window), Dynamic t-SNE needs to re-project all received data to create an entire dataset projection. Consequently, none of these online approaches are appropriate for streaming scenarios where data continuously grow, either because they need to store and revisit all received data during the projection process or due to other method-specific constraints.

A few methods address this storage problem. Partial-Linear Multidimensional Projection (PLMP) [21] defines a strategy that artificially generates an initial sample without visiting the data, using this sample to create a single projection function employed to process all data. Saul et al. [22] devise a streaming version of LLE that creates a projection function considering the first incoming instances and employs this to process all data. Using a similar strategy, Mahapatra and Chandola [23] propose a streaming adaptation of the ISOMAP technique. Although such methods can handle streaming data, creating one projection function at the beginning of the process and using it to project all data is a bottleneck. The computed projections will only be satisfactory if the new incoming data is similar to the data employed to create the initial function. If new structures or patterns emerge over time, these techniques cannot represent them since the function does not evolve with the data. Thereby, in general, the produced layouts are of low quality.

Towards the challenging idea of evolving the projection function over time, Len et al. [24] present a streaming version of LLE. This technique stores the necessary parameters to construct the projection function and adapts it to consider the incoming data. Similarly, Ross et al. [25] store the parameters of a PCA function, updating the PCA eigenbasis using the Sequential Karhunen-Loeve algorithm [26] as instances are received. Fujiwara et al. [27] use the same strategy to evolve PCA projections and use Procrustes analysis [40] to align consecutive projections to preserve the user's mental map. Supervised strategies have also been developed to evolve with the data, such as the online versions of Maximum Margin Criterion [28] and Linear Discriminant Analysis [29]. The so-called Autoencoders [30] could also be used in these scenarios by continually re-training an autoencoder as new incoming data is received, adjusting the network over time as a fine-tuning process. Although the projection function changes to capture new incoming structures and patterns in all these cases, the already projected instances are not adapted to the new function. Consequently, a misalignment between the placement of the already projected instances and the new instances' positions could exist. This misalignment is a significant bottleneck for visualization purposes since meaningless visual patterns can be obtained by overlapping subsequent (misaligned) projections.

Xtreaming addresses the problems of capturing new arising data streaming structures and adjusting the already projected data through a framework capable of evolving the projection layout over time to represent the new emerging structures, while it adapts the previously-projected data without revisiting the multidimensional data that was already processed. *Xtreaming* is detailed in the next section.

3. Streaming Multidimensional Projection

3.1. Problem Formalization

To set notation, let $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^m$ be a multidimensional dataset with $\delta(x_i, x_j)$ a dissimilarity function between two data instances, and $Y = \{y_1, y_2, \ldots, y_n\} \in \mathbb{R}^2$ its mapping to the visual space with $d(y_i, y_j)$ a distance function between two graphical elements. A batch (in-sample or out-of-sample) multidimensional projection technique is a bijective function $f : X \to Y$ that maps X into Y preserving the dissimilarity or neighborhood structure of multidimensional datasets that are fully stored in persistent tables.

In streaming scenarios, the dataset X is, however, not entirely available at the beginning of the process, and k data partitions $X = X_1 \cup X_2 \cup \ldots \cup X_k$, with $X_i \cap X_j = \emptyset, \forall\, X_i, X_j$ with $i \neq j$, are successively received. Considering a progressive scenario where data is projected as obtained and then discarded, it is not possible to have one function to project all the data, instead, it is necessary a set of r functions to project the first r partitions ($r < k$) so that the union of the partitions' projections approximates the projection of all received data, that is, $Y_{[1\ldots r]} = f^1(X_1) \cup f^2(X_2) \cup \ldots \cup f^r(X_r) \approx f(X_1 \cup X_2 \cup \ldots \cup X_r)$, where f^i represents the function employed to project the partition X_i.

Although the state-of-the-art incremental/online projection techniques are based on such approximation, e.g., the streaming version of LLE [24] or the online versions of PCA [25, 27], Maximum Margin Criterion [28], and Linear Discriminant Analysis [29], if new structures emerge over time (such as groups of similar instances) this can fail to differentiate them from the existing structures. Therefore, for an effective streaming projection strategy, not only the projection function needs to be updated to capture new structures, but also the projection $Y_{[1\ldots r-1]} = Y_1 \cup Y_2 \cup \ldots \cup Y_{[r-1]}$ of the previous partitions needs to be adapted to comply with the new projection function f^r, that is, $Y_{[1\ldots r]} = f^r(X_1) \cup f^r(X_2) \cup \ldots \cup f^r(X_{r-1}) \cup f^r(X_r)$.

However, in data streaming scenarios, where storing all data is typically unfeasible, most of the partitions $X_1, X_2, \ldots, X_{r-1}$ are no longer available when projecting X_r, so $f^r(X_1) \cup f^r(X_2) \cup \ldots \cup f^r(X_{r-1})$ cannot be computed.

3.2. Overview

Our approach, *Xtreaming*, addresses both design challenges, namely, (1) to update the projection function to represent new emerging dissimilarity structures, and (2) to adapt the current projection to comply with the updated function without revisiting the multidimensional data already processed. *Xtreaming* handles the former by using a change detection strategy to trigger the reconstruction of a new function every time a change in data distribution is detected, and the latter by using the information contained in the current projection to evolve the visual representation over time. Notice that, instead of directly processing the data as received in partitions, we use buffers of fixed sizes so that our approach is independent of partitions' sizes, and buffering strategies can be used to process the data in constant rates without depending on the speed the data is being received.

Fig. 1 outlines our technique. The first buffer is used to initialize the process, resulting in the first projection function (f^1).

After that, if a new incoming buffer does not represent a change in the distribution, it is projected using the current projection function. If a change is detected, the projection function is recreated, the data already projected is re-projected to comply with the new function, and finally, the buffer is projected (Δ indicates that our projection function operates using only distances, later discussed in Sec. 3.4). This process is repeated while there is incoming data. The next sections detail each step involved in our approach.

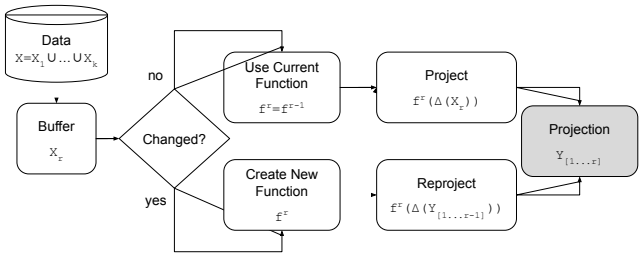


Fig. 1. *Xtreaming* overview. Every time a new data buffer is received, we check if it represents a change in the data distribution. If no changes are detected, the buffer is projected using the current projection function. Otherwise, the projection function is updated, the already projected data is re-projected, and the buffer is finally projected.

3.3. Change Detection and Sampling

In our approach, to check if a new buffer X_r represents a change in data distribution, we combine a clustering technique with a change detection strategy composing a two-step approach.

In the first step, some representative samples are recovered from X_r to represent its distribution. To do this, a distance-based clustering technique, called bisecting k-means [41], is applied splitting X_r into q disjoint clusters $X_r = C_1 \cup \ldots \cup C_q$ containing similar instances. Then, for each cluster C_i, its medoid x_i' (the instance closest to its centroid) is selected as a representative, composing a pre-sample $X' = x_1', \ldots, x_q'$. Based on a common heuristic, we set the number of clusters to $q = \sqrt{|X_r|}$ since it defines a good upper-bound for the number of clusters [12, 21, 7]. Other sampling methods can also be used in this process, from simple random selections to strategies based on spectral decomposition [42]. Here we use a cluster-based strategy given the good tradeoff in terms of runtime and the recovered samples' quality.

In the second step, each medoid x_i' in the pre-sample X' is tested to verify if it should be added to the partition sample X_r or not. The reason is to avoid including instances to the buffer sample which the statistical local distribution is already represented by the current sample $X_{[1\ldots r-1]}$. In this process, we use the iLOF [43] technique to measure how different a medoid is from the current sample using the concept of local densities. In this process, for each medoid x_i' we first calculate its nearest neighbors considering the current sample $X_{[1\ldots r-1]}$. If the medoid lies in a multidimensional space region with substantially lower density than its nearest neighbors, it is added to the sample X_r. Although a precise and well-established method, iLOF is computationally expensive. That is why we pre-sample the data before applying it.

The described process serves two purposes. It discovers if new structures are present in the incoming buffer, considering the existing sample, and it also samples the data so that the projection function can be updated accordingly. The idea is if $X_r = \emptyset$ then $f^r = f^{r-1}$, otherwise a new projection function needs to be calculated considering the new sample. Notice that, since we are using an out-of-sample strategy to project the data, the first step is to project $X_{[1\ldots r]}$ when building a new function f^r. Given that distance-based projections are invariant to rotation [44], every time a new sample is projected, we aligned it with the projection of the previous sample $X_{[1\ldots r-1]}$ using Procrustes analysis [40]. In this way, we maintain the spatial coherence of the projection as it evolves, similarly in intent to what has been done with dynamic graphs [45, 46, 47], avoiding problems related to drastically changing subsequent projections which negatively affect layouts stability and users' mental map [48, 49].

3.4. Constructing the Projection Function

As previously discussed, if an incoming buffer X_r does not represent a change in distribution, the current projection function f^{r-1} is used to process it, that is, $f^r = f^{r-1}$. Consequently, such a function needs to be "stored" or represented without accessing all the data. To address this issue, we use an out-of-sample strategy. Out-of-sample strategies are two-step approaches that project a small sample X of the dataset, with $|X| \gg |X|$, and then interpolate the remaining instances using this initial projection as a base. Hence, we only need to store a fraction of the entire dataset (and a few technique parameters) to represent a projection function, obeying data storage constraints of streaming scenarios, where usually it is not desirable or possible to store all the collected data given hardware capacity limitations.

Besides being out-of-sample, the projection technique needs to fulfill other requirements: (1) it has to be fast and precise so that the runtime and the quality of the produced mappings are not impaired; and (2) it needs to be a distance-based strategy (receive distances as input) so that the re-projection of the projected data can be performed without accessing all data. So that, our projection function takes the form of $f : \Delta(I) \rightarrow Y$, where $\Delta(I)$ represents the distances between instances in the input space I, either the high-dimensional data in $\mathbb{R}^m$ or the projection in $\mathbb{R}^2$. Currently, only a few out-of-sample techniques can comply with such requirements, such as the Pekalska approximation [37], LMDS [8], and UPDis [50]. This paper uses the UPDis technique since it renders better results regarding distance preservation (see [50] for comparisons) while presents competitive runtimes.

3.5. Re-Projection

Every time a change is detected in a buffer X_r and a new (out-of-sample) projection function f^r is calculated, the projection $Y_{[1\ldots r-1]}$ of the previous buffers $X_{[1\ldots r-1]} = X_1 \cup X_2 \cup \ldots \cup X_{r-1}$ needs to be updated to comply with this function. Given that

we are using a distance-based out-of-sample technique, for re-projecting $X_{[1...r-1]}$ it is only necessary to compute the distances between $x_i \in X_{[1...r-1]}$ and the samples $x_j \in X_{[1...r]}$, where $X_{[1...r]}$ represents the current sample, including the new instances detected by the change detection method.

The problem is, since $X_{[1...r-1]}$ is not stored, the distances among them and the samples cannot be calculated. We address such a limitation using a simple but effective strategy. Considering that the projection function is precise in terms of distance preservation (first requirement of the previous section), the distances between the instances $X_{[1...r-1]}$ and the samples $X_{[1...r-1]}$ can be approximated, with some degree of accuracy, by replacing the original distances by the projected distances, that is, by taking $\delta(x_i, x_j) \approx d(y_i, y_j), \forall\ x_i \in X_{[1...r-1]}, x_j \in X_{[1...r-1]}$ and this information can be fully recovered from $Y_{[1...r-1]}$.

For a complete projection, the only information that is missing is the distance between the samples X_r discovered by the change detection method and the instances already projected. We handle this problem by projecting X_r using the previous projection function f^{r-1}, using this projection to recover the distances, that is, by taking $\delta(x_i, x_j) \approx d(y_i, f^{r-1}(x_j)), \forall\ x_i \in X_{[1...r-1]}, x_j \in X_r$.

Since $|X_r| \gg |X_r|$, this process does not affect our approach's runtime. Notice that, since a high-precision out-of-sample strategy is employed, this is a good approximation for global distance preservation, and only uses the current projection and the sample, not requiring access to the instances' coordinates in $\mathbb{R}^m$ of the projected data.

4. Technique Analysis and Comparisons

This section presents a series of tests and comparisons to attest the proposed technique's stability and sensibility and confirm its quality compared to other in-sample, out-of-sample, and incremental/online techniques. In these tests, we use datasets with different sizes and dimensionalities, enabling the analysis of different scenarios. The first dataset, **shuttle** [51], is composed of 43,500 instances with 9 dimensions representing log information. The **mammals** [51] is an artificially generated dataset representing 72 different features of 50,000 mammals of four distinct classes (dogs, cats, horses, and giraffes). **MNIST** [52] is a dataset of 60,000 images of handwritten digits each one represented by 784 intensity values. The **corel** [51] dataset is composed of 68,040 images represented by color histograms in 32 dimensions. The **viscontest** [53] dataset corresponds to a 10-dimensional sample with 100,000 instances of one time step of a ionization front instability simulation. The **FMA** [51] is a music dataset which contains 518 audio features of 106,574 tracks. The **quantum** [54] dataset is related to particle physics and was obtained from the *KDD-Cup 2004* containing 150,000 instances with 78 dimensions. Finally, the **fibers** [12] dataset is composed of 250,000 30-dimensional instances representing fiber tracks. Notice that, since we compare our approach against non-incremental techniques, we opt to use datasets without timestamp to not bias the process.

4.1. Technique Analysis

Our first analysis aims to verify the influence of the buffer size on the processing time and quality of the produced layouts considering the global preservation of distances using Kruskal's stress [55]. In this test, we use all the mentioned datasets and split them into buffers of varying sizes and present these buffers subsequently to the *Xtreaming* technique as if it is streaming data. The buffer sizes tested were: 1,000, 5,000, 10,000, 15,000 and 20,000, covering a range of sizes proportional to the employed datasets. Fig. 2 shows the results. Irrespective of the buffer size, the stress presents similar values on average, with marginally better results for the largest buffer. In terms of runtimes, the larger the buffer, the longer the runtime, which is expected due to the sampling and change detection steps cost. Based on that, we set the buffer size to 1,000 instances for the remainder of this paper.

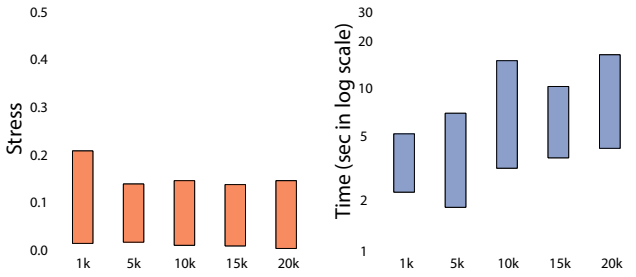


Fig. 2. Stress and time boxplots for different buffer sizes. Different buffer sizes produces similar layouts in terms of distance preservation but the larger the buffer the longer the processing time.

The second test aims to verify if the change detection mechanism produces samples of reasonable sizes as the buffers are received. Fig. 3 presents the results. The vertical axis represents the percentage of the upper-bound limit ($\sqrt{n}$) for the sample size typically employed by state-of-art out-of-sample projection techniques [7, 21]. The horizontal axis represents the percentage of dataset received by the technique (buffers with 1,000 instances). Notice that the sample sizes never hit the sample upper-bound for the tested datasets. Actually, for most, it stays well below and stabilizes after receiving few buffers, showing that our change detection mechanism, although very simple, result in samples with sizes compatible with the literature. However, notice that this may not be true for datasets with distribution constantly changing over time or that do not present well-defined or too many groups of similar instances. Given the nature of the iLOF technique, such datasets will trigger the change detection more than well-behaved datasets and may more frequently activate the insertion of instances into the set of samples, potentially leading to an upper-bound extrapolation.

Another analysis was conducted to verify if the technique is sensitive to data ordering. In this test, *Xtreaming* is executed 30 times for each dataset, randomly shuffling the order the data instances are presented to the technique (the data instances are shuffled before the data is processed). Again, we split each dataset into buffers and present them subsequently to *Xtreaming* as streaming data. Fig. 4 presents boxplots summarizing

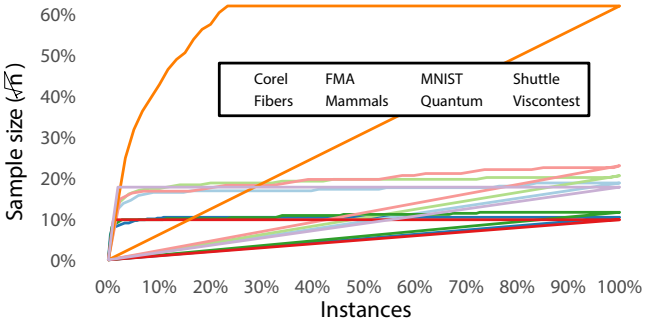


Fig. 3. Sample size vs. received buffers. For the tested datasets, our cluster-based change detection mechanism constructs a sample never larger than a typical upper-limit adopted by most out-of-sample techniques ($\sqrt{n}$), and normally stabilises it after receiving the first partitions.

the attained results in terms of distance preservation. Observe that each dataset's stress variation is small, and for most, it is virtually zero. Only the **MNIST**, **corel**, and **quantum** datasets present some apparent variation. In these cases, something associated with the structures present on the datasets that depend on the order they are accessed causes the differences in stress quality. Since there is no correlation between this observed phenomenon and data characteristics (size, dimensionality, and classes), to shed some light on the phenomenon, we verified if clustering tendency [56], that is, the presence or not of dense groups of instances, has some influence. In our experiments (not reported due to space constraints), we observed that these three datasets present lower clustering tendency than the others, indicating that the number of instances necessary to represent their overall distribution is larger. So depending on the order the data is accessed, distribution can be better or worse represented. However, notice that the standard deviation, even for the **quantum** dataset, is indeed quite small (0.035), indicating that the produced layouts' overall quality is not heavily affected by the order the data is processed. This is an essential feature for a data streaming technique since the results are consistent in different execution scenarios, assuring a reasonable degree of stability and reproducibility.

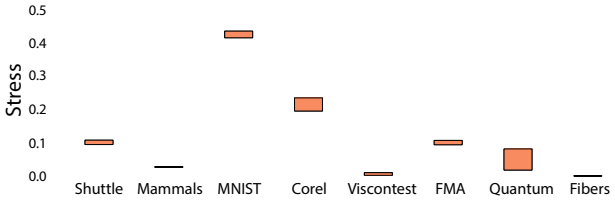
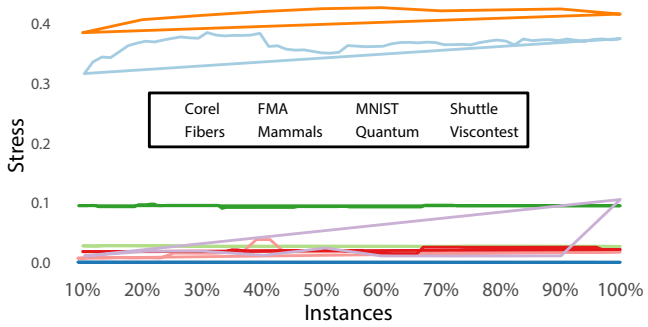


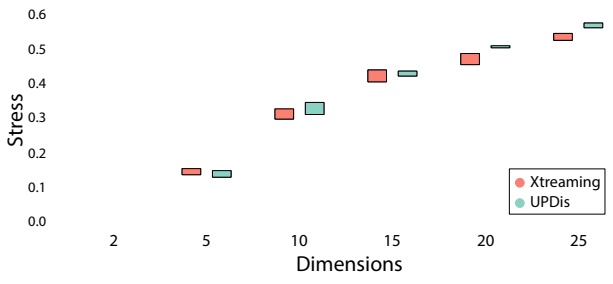
Fig. 4. Stress boxplots using *Xtreaming* randomly shuffling the datasets multiple times to change the order the data instances are processed. The small stress variation for each dataset indicates that *Xtreaming* produces consistent results on different execution scenarios, assuring a good degree of stability and reproducibility.

Next, we seek to verify if the re-projection mechanism impairs the quality of the produced layout as the partitions are processed. As detailed before, every time a new buffer is received, the previous buffers projections are adjusted to the new information. However, since no data is stored, the high-dimensional distances are approximated, replacing them using two-dimensional distances to re-project the already-projected instances. In this test, we measure the projection quality as each buffer is received. Fig. 5 shows the results. For most datasets, the stress presents similar values from the beginning to the end of the process or stabilizes early as the buffers are processed. Only the **corel** dataset presents a more significant variation since it is a difficult dataset to handle and all techniques we compare with have problems processing it.

We have also tested if the intrinsic dimensionality of a dataset affects the quality of the re-projection strategy. We employ artificially-generated datasets with different intrinsic dimensionalities (from 2 to 25). This way, we can have more control over the process since the previously-used datasets do not present a consistent variety of intrinsic dimensions. To create a dataset with k intrinsic dimensions, we generate $30,000$ vectors with k random values following a uniform distribution in $[0, 1]$, so that the variance is equal in the k directions. Given the random nature of this test, for each desired dimensionality, we generate 10 datasets. Each dataset is projected with both UPDis (i.e., the non-streaming technique internally used by our approach) and *Xtreaming*. As we can see from Fig. 5(b), the quality does not vary substantially between UPDis and *Xtreaming* as the dimension increases. This indicates that the re-projection strategy is a good approximation and surprisingly does not negatively affect the final produced projection's quality even when datasets with different intrinsic dimensionalities are considered.



(a) Stress vs. received partitions.



(b) Stress vs. intrinsic dimensionality.

Fig. 5. For most datasets, the stress does not increase over time and does not depend on the intrinsic dimensionality. It indicates that the re-projection strategy is a good approximation and does not negatively impact the quality of the produced projection.

In summary, with these tests, we show that *Xtreaming* technique is not affected by the buffer size presenting similar results in terms of global distance preservation as the buffer varies. Also, the projection quality is not affected by the re-projection phase or by the change detection and sampling strategy. The same level of distance preservation is kept from the beginning to the end of the process.

4.2. Comparisons

This section presents a set of comparisons involving batch and incremental projection techniques, checking the global and local distance preservation. For the analysis of global preservation, we use *stress*. For local preservation, we report on the results of *trustworthiness* and *continuity* [57], which are measures of the local quality of neighborhoods. We start by comparing *Xtreaming* against batch techniques. The batch techniques employed in these comparisons were chosen based on two criteria: they have to be designed to handle large datasets and present a good tradeoff in quality and runtime. We compare *Xtreaming* against LAMP [7], UPDis [50], PCA [58], PLMP (batch version) [21], LMDS [35], and Pekalska [37]. All results reported in this section were produced on an Intel® Core™ i7 CPU 2.40GHz with 16GB of RAM. All techniques were implemented in Java, including *Xtreaming*.

Fig. 6 displays boxplots that summarize the results. On average, for stress and trustworthiness, *Xtreaming* produces better results if compared to Pekalska and PLMP techniques, with PLMP presenting a significant deviation on quality. Compared to the most precise techniques, LAMP, UPDis, and LMDS, *Xtreaming* produces very competitive results. This is an unforeseen outcome since *Xtreaming* has an additional step, the re-projection, which approximates the multidimensional distances replacing them by the bi-dimensional distances. Compared to UPDis, we expected to produce worse results since this is the technique we use to construct the projection functions. In practice, however, UPDis is only slightly better. Besides the re-projection, another difference between *Xtreaming* and the out-of-sample techniques is the mechanism employed to select the sample instances. For all out-of-sample techniques, we use a clustering technique to produce $\sqrt{n}$ clusters, getting their medoids as samples. For *Xtreaming*, the sample is constructed as the data is received, without fixing the number of samples, albeit the sample never gets larger than $\sqrt{n}$ in this test.

We conduct two different analyses in terms of comparing *Xtreaming* against other incremental (streaming) projection techniques: distance preservation (stress, trustworthiness, and continuity) and runtimes. Fig. 7 presents boxplots summarizing the results. *Xtreaming* produces considerably better results when compared to the streaming version of PLMP (here called sPLMP). This is expected since sPLMP builds a projection model without considering the structures present in the data and uses it to project the full streaming without changing or adapting it over time. Compared to incremental PCA [25] (iPCA), *Xtreaming* presents close but better results on average. It is worth noticing that, when comparing iPCA against PCA (Fig. 6), it is possible to observe that the iPCA strategy of incrementally updating the projection function introduces errors to the projection process while the same drop in quality is not perceived when comparing the *Xtreaming* against UPDis (Fig. 6).

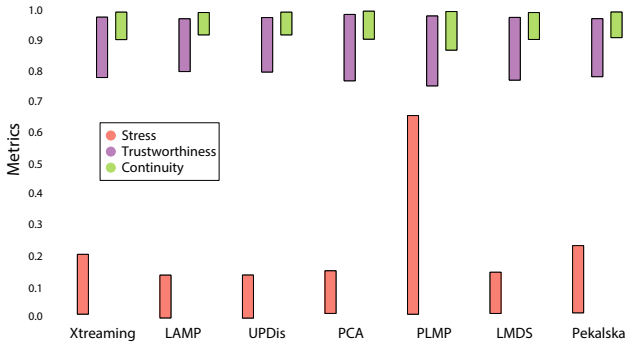


Fig. 6. Stress boxplots. Comparing to batch (non-streaming) techniques, *Xtreaming* presents very competitive results, rendering a reliable process to project streaming data.

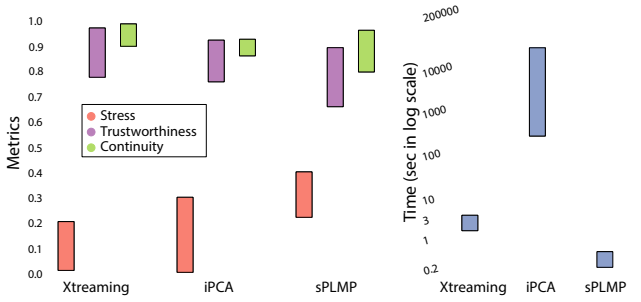


Fig. 7. Stress and runtimes boxplots of the incremental (streaming) projection techniques. *Xtreaming* presents considerably better global distance preservation than sPLMP, and similar results, on average, to iPCA. However, it is two orders of magnitude faster than iPCA, effectively being a good candidate to process streaming data in real-time.

In terms of runtime, *Xtreaming* and iPCA are slower than sPLMP. This is expected since sPLMP does not update the projection function over time, and the projection process is reduced to a matrix-vector multiplication. However, the (local and global) distance preservation is penalized, presenting unreliable and low-quality values. As mentioned before, *Xtreaming* and iPCA present similar quality, but *Xtreaming* is orders of magnitude faster than iPCA. This happens because the process employed by iPCA to update the eigenvectors is expensive, and it is executed every time a new data instance is received (the nature of iPCA is to be incremental, not streaming). To better illustrate the gap in runtime between *Xtreaming* and iPCA, we select two larger dataset from the UCI Machine Learning Repository [51] to test: **susy** containing $5,000,000$ instances with 18 dimensions and **gas** composed of $3,843,160$ with 20 dimensions. *Xtreaming* took 689s to process the **gas** and 654s to process the **susy** while the iPCA took $9,125$s and $7,241$s, respectively. *Xtreaming* is at least 11 times faster. Thereby, not only *Xtreaming* presents a better balance between quality and runtimes, effectively allowing real-time progressive analysis to take place but also it can process large datasets in a reasonable time, an essential feature for the current big data scenarios.

Worthy noting that we did not include these two datasets in the previous tests because the existing implementations to calculate the quality metrics are highly computationally expensive. We do not have access to computers that allow their computation in a reasonable (finite) time.

Finally, Fig. 8 presents projections of some selected datasets produced by *Xtreaming*, iPCA, and sPLMP techniques at three different stages of the projection process, beginning, middle, and end. Beginning represents 33% of the dataset, middle 66%, and end 100%. In this test, we select one small (**mammals**), one medium (**viscontest**), and one large dataset (**fibers**), and intentionally order them based on the class (when available) so that instances belonging to the same class are added to the projection process subsequently. The idea is to show the impact of not updating the data instances already projected as new structures or patterns, in this case, classes of instances, emerge over time. As discussed in the related work, sPLMP does not suffer from this issue. It is order-independent. However, the employed strategy does not result in a projection separating the four distinct groups of points for the **mammals** dataset, and the overall distance preservation is considerably lower than the other techniques (values for stress, trustworthiness, and continuity are displayed with each projection in Fig. 8). The iPCA technique produces the most revealing result. Although, on average, iPCA projections are of good quality, when the projection process starts with instances that do not represent the overall distribution of the dataset, the technique cannot distinguish the emerging patterns from the existing ones, even if they are very different. The result are visual artifacts where such patterns overlap, in this case, the four different classes of the **mammals** dataset, heavily affecting the overall distance and neighborhood preservation. This problem is not observed in *Xtreaming* projections, attesting that the re-projection process is effective even when the dataset distribution changes, producing more coherent layouts that represent well the global structure of multidimensional datasets. Worthy noticing that, although the general structures are coherently spatially preserved over time, in some cases, the introduction of new groups/classes can give the impression that *Xtreaming* violates this preservation, affecting the user mental map, as observed in the **mammals** projections. Indeed, this is a normalization problem for displaying purposes which can be solved if all projections were globally normalized. However, it is necessary to know the final projection for this normalization, which is impossible in streaming scenarios.

5. Case Studies

In this section, we describe two case studies that showcase the suitability of using *Xtreaming* to support the visual analysis of time-varying multidimensional datasets, the first with an artificially-designed dataset, and the second with sentiment and stance analysis on data from Twitter.

5.1. Case Study 1: Time-varying Clusters

In order to support the visual analysis of multidimensional data streams, three requirements are desired from a projection technique that focuses on distance preservation:

- **R1:** It must always maintain a consistent view of the dataset as a whole, even if different patterns happen at different time steps. This means that not only the patterns themselves must be consistent but the inter-pattern relationships must stay consistent (e.g., distances between clusters).
- **R2:** It must fulfill **R1** without becoming too "rigid" as time passes, so that new patterns may still correctly find their way into the projection.
- **R3:** It must "learn" patterns that happen seasonally, so that different occurrences of the same pattern, even if separated by many time steps, will still be correctly identified as being the same.

This first case study (Fig. 9) is designed to demonstrate that *Xtreaming* fulfills these three requirements, and to provide a solid foundation for trusting the visual results of the technique when dealing with streams that contain multiple patterns that happen at different time steps, with little to no overlap between each other. It is based on an artificially-generated dataset with $50,000$ points spread evenly among three 3D clusters (Fig. 9(a)), arriving in the stream according to a fourth (time) dimension (Fig. 9(b)). Both the spatial and the time dimensions of the dataset were designed carefully to test the three requirements stated previously. The centroids of the clusters are positioned in three different vertices of an invisible cube, such that Clusters 2 and 3 (*blue* and *green* respectively in Fig. 9) are in neighboring vertices, so they should be positioned closer to each other, while Cluster 1 (*red* points in Fig. 9) is in a diagonally-opposite vertex. Regarding time, points in Cluster 1 are evenly distributed in three different non-overlapping occurrences (simulating a seasonal pattern), while the points in Clusters 2 and 3 are very well-separated in time, also with no overlaps.

The dataset is analyzed by *Xtreaming* as a stream of 50 consecutive steps (S) of $1,000$ points each. Figs. 9(c-g) show partial projections of the stream in key time steps. In order to denote the "age" of a point p, i.e. in which step S_p it arrived, we map its opacity O_p between 0 (fully transparent) and 1 (fully visible) according to the transfer function

$$O_p = \frac{1}{S - S_p + 1}, \tag{1}$$

so that, points from the current step (S) are always fully visible.

At $S = 5$ (Fig. 9(c)), we still see only Cluster 1, with no sign of any other pattern. Cluster 2 can be clearly seen at $S = 10$, and while no new points of Cluster 1 have arrived in the stream for a while, they can still be seen with low opacity O_p. At this point, it is interesting to notice that Cluster 2 has been positioned correctly regarding its separation to Cluster 1, even though more than 5,000 data points had already been analyzed before the points from Cluster 2 started to arrive. This is related to both **R1** (global consistency *between* clusters) and **R2** (learning new patterns).

At $S = 25$ (Fig. 9(e)), Cluster 1 has returned to the stream after a long period of inactivity, and its points were correctly

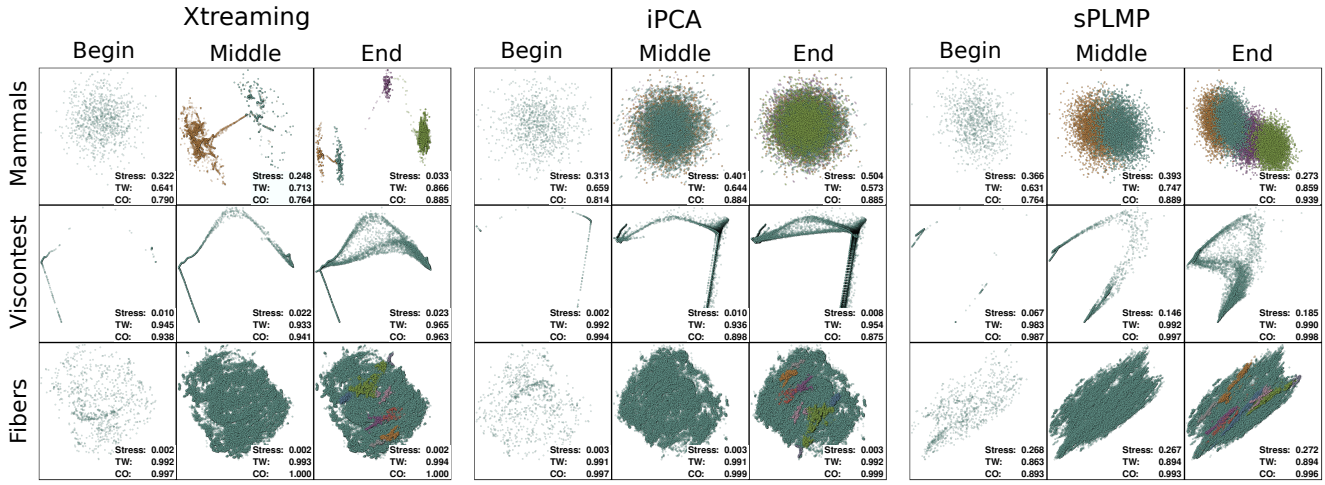


Fig. 8. Projections generated by *Xtreaming*, iPCA, and sPLMP techniques at three different stages of the streaming projection process, beginning, middle, and end. The datasets are ordered based on the class so that the first instances belong to the same class, and instances of other classes are added to the projection process subsequently. In general, the sPLMP projections are of low quality and are not reliable. Although iPCA projections are of better quality, when the projection process starts with instances that do not represent the overall distribution of the dataset, the technique cannot distinguish the emerging patterns from the existing ones (mammals dataset). This issue is not observed in *Xtreaming* projections, and it produces more coherent layouts, successfully capturing the global changes that occur over time.

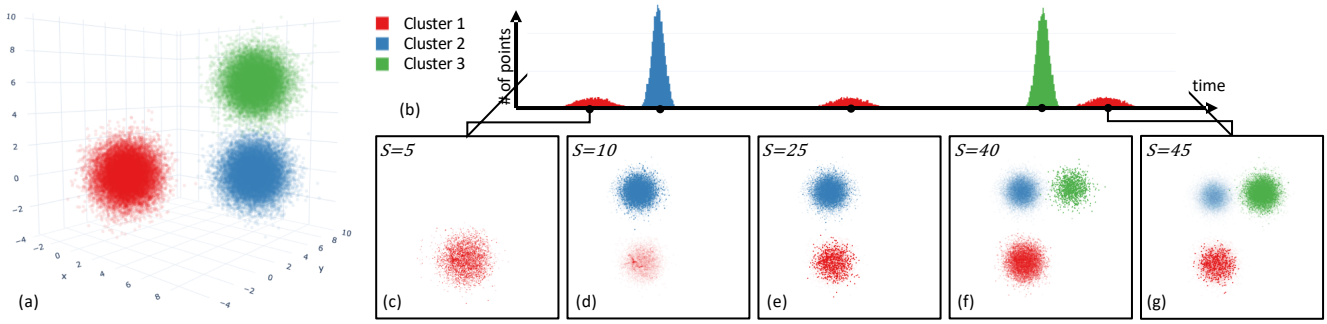


Fig. 9. A demonstration of the stability of time-varying patterns in *Xtreaming*'s projection results. The artificially-generated dataset consists of $50,000$ points located in three 3D clusters (a) arriving in the stream at different times (b). As it can be seen both in partial projections (c-f) and in the final layout (g), *Xtreaming*-maintained the overall structure of the dataset, both locally (within clusters) and globally (between clusters), even if the seasonal patterns of Cluster 1, and the two neighbor Clusters 2 and 3, do not overlap in time.

positioned on top of the previous points from the same cluster, showing that *Xtreaming* has been able to maintain its *memory* about previous patterns in a lightweight way. Some other techniques that might use *Xtreaming* as a pre-processing step for learning from the stream, for example, would be able at this point to correctly identify that the new points belong to the same cluster, even though all clusters had stopped providing points a long time ago. This is mostly related to **R3** (learning seasonal patterns), but also to **R1** (global consistency *within* clusters).

Points from Cluster 3 have arrived and settled in the projection by $S = 40$ (Fig. 9(f)). Notice that, by the time its points started to arrive, more than $35,000$ points had already been analyzed, with two occurrences of Cluster 1 and one occurrence of Cluster 2. Nevertheless, the projection did not have problems to learn the new pattern and position the cluster correctly, not only well-separated from the other clusters but positioned closer to its neighbor (Cluster 2) and far from Cluster 1. This is the expected outcome based on how the cluster centroids were initially generated in different vertices of the cube. These observations are related to **R1** (global consistency *between* clusters) and **R2** (learning new patterns). Finally, at $S = 45$ (Fig. 9(g)), more points from Cluster 1 arrive at the stream, again being correctly positioned on top of their previous intra-cluster neighbors, providing more evidence for **R3** (learning seasonal patterns) and **R1** (global consistency *within* clusters).

One final general observation about *Xtreaming* that can be recovered from this case study as a whole is the consistency of its output throughout time steps. Consecutive projections of different time steps of the dataset maintained the overall shape of the data very closely, even after long periods of inactivity and little overlap between different clusters' time distributions. This is also a critical desirable characteristic of a technique that intends to support visual analysis of time-varying data.

5.2. Case Study 2: Sentiment/Stance Analysis on Twitter

This case study showcases the suitability of using *Xtreaming* to support the visual analysis of a streaming Twitter data collection. The dataset consists of social media posts in English collected from the Twitter streaming API during 12:00:00–23:59:59 UTC on July 9, 2018. The keywords used to filter the stream of tweets include several phrases and names related to Brexit, such as "brexit", "british referendum", "bnp", "ukip", and "nigel farage". This dataset's particular date corresponds to the day of resignation of the UK Foreign Secretary Boris Johnson, which was widely discussed on both traditional and social media.

To compose the dataset, the texts of the tweets are split into individual sentences (utterances), which are classified according to sentiment [59] and stance [60]. Sentiment classification is carried out with the VADER classifier [61] that reports normalized output weights for the categories *positivity* and *negativity*. Weights below a certain threshold (in our case, 0.3) are ignored and such utterances are considered neutral (they assume the value 0.0). Stance classification is carried out with a custom stance classifier [62] that detects multiple categories and reports the corresponding output weights (the values of the weights range within the interval $[0.5, 1.0]$, anything less than 0.5 is reduced to 0.0). The exact set of 12 stance categories includes the following: *agreement* (A), *certainty* (Ce), *concession and contrariness* (CC), *contrast* (Co), *disagreement* (D), *hypotheticals* (H), *need and requirement* (NR), *prediction* (P), *rudeness* (R), *source of knowledge* (SK), *tact* (T), and *uncertainty* (U). In its final form, the dataset includes $503,511$ rows (sentences extracted from tweets) and 14 columns with the classification results (two columns for sentiment and twelve for stance) represented as numbers in the range $[0.0, 1.0]$.

The final projection of the full dataset, generated by *Xtreaming*, is shown in Fig. 10 (a). The colors indicate the strongest detected sentiment/stance in each data point; in other words, the sentiment/stance label with the corresponding row's maximum value. They are provided as support to the visual analysis, but should not be considered the final classification of the data points. The opacity indicates how "old" each data point according to the transfer function defined in Eq. (1), so that points from the current step (S) are always fully visible. Fig. 10 (b) presents a heatmap generated from features' values into a continuous colormap ranging from 0.0 to 1.0. Each column of the heatmap (the x axis) represents one of the 14 features, and the y axis represents the selected points. The number of selected points is shown on top of each heatmap, but if a selection includes more than $5,000$ points (marked with an asterisk), they are downsampled to this number to maintain efficiency. Additionally, points are arranged in the y axis according to a PCA projection down to 1 dimension, so that similar points are grouped.

At first, some patterns/groups can be readily detected due to *Xtreaming* having isolated them clearly from others, such as tweets with predominantly *need and requirement* (Group A), *certainty* (Group B), or *source of knowledge* (Group C), as shown by the homogeneous feature distributions in the corresponding heatmaps. This initial step of detecting and understanding the easily-identifiable groups can be associated with an *overview* task of the projection. After the overview, the analyst might be interested in exploring whether the less-identifiable groups—those that are less clearly separable or salient in the layout—are also meaningful, or are simply the results of noise in the data or lack of accuracy in *Xtreaming*. However, a more detailed investigation of specific areas of the layout shows that even these less-identifiable groups can be traced to consistent patterns in feature interactions, as described next. For the following discussion, only the two most significant attributes were considered.

Group D: Contrast vs. Concession and Contrariness. Group D initially looks like a homogeneous group of points detected as mostly *contrast*, but the heatmap shows an almost-equal distribution of *concession and contrariness*. These are related categories, but the former does not include instances of *antithesis* as studied in rhetorical structure theory [63] (for instance, negations or antonyms). Thus, it is expected that these categories would co-occur in the classification results, and the fact that the projection technique has positioned the corresponding utterances in the same neighborhood confirms our expectations.

Group E: Prediction vs. Uncertainty. The co-occurrence of categories such as *prediction* and *uncertainty* can be explained considering the uncertain nature of most predictive statements, mostly based on terms like "guess","suppose", followed by simple future forms (e.g., "I guess the political crisis will continue…").

Group F: Source of Knowledge vs. Uncertainty. The presence of a cluster with a prominent number of co-occurrences of such stance categories as *uncertainty* and *source of knowledge* can be explained by the attempts to attribute uncertain information or rumors to third-party sources (e.g., "According to World Press, the event may have occurred…") or to employ hedging while presenting one's opinion (e.g., "I think I remember reading about their political failures…").

6. Discussions and Limitations

Rate of streaming. One potential problem is related to the rate in which the data is captured or produced in time-dependent streaming applications. In these scenarios, it is vital to produce the projection layout on-the-fly as the data is received. However, if the data production rate is larger than the processing time imposed by *Xtreaming*, some data may be discarded. One possible solution is to use a double-buffer strategy, where parts of the data can be stored while others are processed. This difficulty is more related to technology than to the technique itself. In practice, we expect that with appropriate hardware and implementation, *Xtreaming* runs much faster than the results reported in Sec. 4, which is already orders of magnitude faster than other comparable techniques.

Small datasets. Another limitation, which is a direct consequence of *Xtreaming*'s goals, is that it is not intended for small-sized problems. In our case, the definition of small depends on the employed buffer size (see Sec. 3.2) and hardware constraints. If a dataset is smaller than the buffer, the final projec-

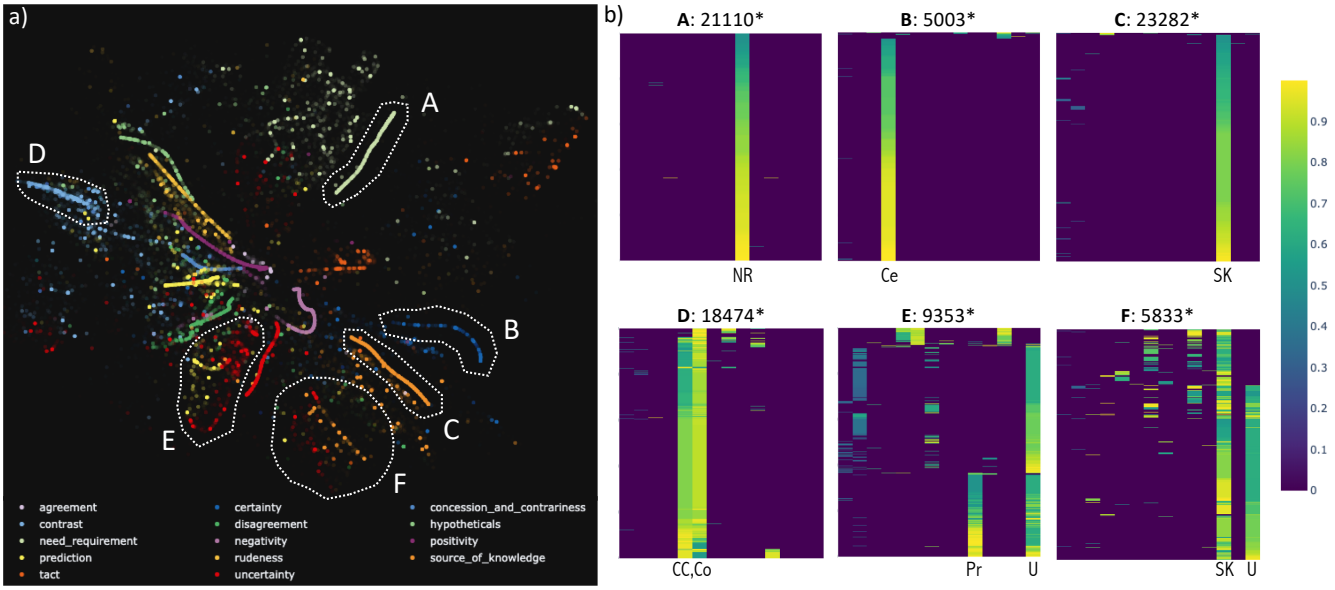
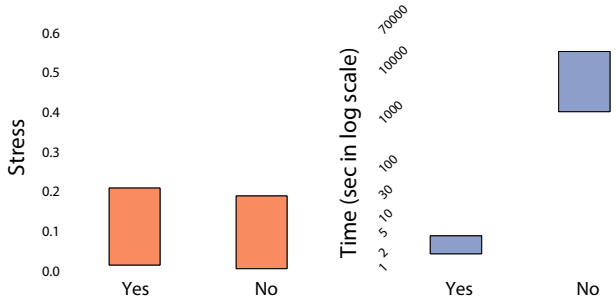


Fig. 10. A case study on using *Xtreaming* for the visual analysis of a social media stream from Twitter characterized by sentiment and stance categories. Each point is a sentence from a tweet, the colors indicate the strongest detected sentiment/stance in that tweet, and the opacity indicates how "old" it is. At first, some patterns can be readily detected, such as groups of tweets with predominantly (A) *need and requirement*, (B) *certainty*, or (C) *source of knowledge*. More investigation shows that even the less-identifiable groups can be traced to consistent patterns in feature interactions, such as (D) *contrast* vs *concession and contrariness*, (E) *prediction* vs *uncertainty*, and (F) *source of knowledge* and *uncertainty* (see the text for more details on these groups).

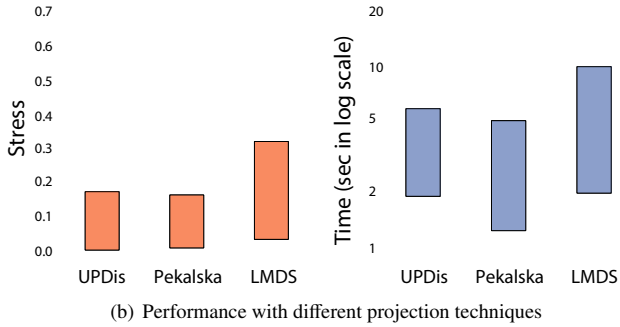
tion is the UPDis projection (the out-of-sample technique internally used). For datasets larger than the buffer, *Xtreaming* will be slower than most online or incremental techniques and not as precise given the approximations and the strategies employed to allow a single data traversal. Consequently, if the dataset is not much larger than the buffer and can be stored entirely in memory (hardware), there is no advantage to using *Xtreaming*. However, for problems where data cannot be stored and are discarded as received, this drop in precision is a small price to pay for handling streaming data, and approximated solutions are acceptable to enable streaming applications [2].

Benefits of novelty detection. In our framework, every time a projection function is updated, there is the update and the re-projection costs. The cost of re-projection is relatively small, but creating a new projection function can be prohibitive. Given that we are using an out-of-sample approach internally, it is necessary to project a growing sample to update a projection function. This is computationally demanding since the employed technique is precise and, therefore, expensive. Therefore, we included a novelty detection mechanism in *Xtreaming*, where the projection function is only updated when new data structures emerge. In order to gather evidence on the actual benefits of this mechanism, we have tested the stress and time of *Xtreaming* with vs. without novelty detection (see Fig. 11(a)). On average, there is a small drop in precision as expected; the runtime, however, increases drastically.

Different projection techniques. As mentioned in Sec. 3, our framework needs a distance-based out-of-sample technique. Although we internally use UPDis as the distance based-technique, we have also tested *Xtreaming* using two other techniques: LMDS and the Pekalska strategy for Sammon's Mapping. The results shown in Fig. 11 attest that UPDis has competitive performance given the datasets we have tested.



(a) Novelty detection: enabled (Yes) vs. disabled (No)



(b) Performance with different projection techniques

Fig. 11. Further experimental results. (a) Novelty detection incurs a small drop in precision as expected, but the runtime increases drastically without it. (b) Using UPDis results in competitive performance compared to other techniques, with lower stress (on average) and similar runtime.

Impact of projection quality. Although we have shown with comprehensive quantitative tests that *Xtreaming* can perform as well as its base (non-streaming) projection, the matter of the im-

pact of the quality of each intermediate projection in the final quality could still be debated, due to our re-projection scheme and the reuse of 2D distances as surrogates for the original distances. However, this seems to depend more on the base technique itself than in the re-projection. In case a new, improved, and compatible out-of-sample technique is developed, it can be attached to *Xtreaming* so that the streaming version also takes advantage of the new and improved base projection.

Different distance metrics. All our initial tests were performed using the Euclidean distance since it is the most common for dimensionality reduction in general. Different distance measures may affect the results of the re-projection step, but our intuition is that any distance function (i.e., a metric following the triangular inequality) should work well, as long as the same function is used in both 2D and nD. However, it is difficult to prove that since this is a general problem of dimensionality reduction methods. It remains a direction of future work to test the impact of this choice on our technique's results.

Layout Stability. One of the essential requirements for projection techniques that consider datasets that change over time is the ability to produce layouts that allow users to maintain a cohesive mental map through time [49]. This concept, called stability [49] or projection alignment [48], reflects the ability of a technique to evolve a projection as data instances change or new data is received over time without distorting it too much. For dynamic projections, this has been recently discussed [49, 64], but the employed definitions and suggested metrics do not apply to streaming scenarios. Unlike dynamic projections, which assume data instances changing over time, streaming projections, consider the dataset increasing over time, with instances remaining the same. We addressed this issue in streaming scenarios using Procrustes analysis when projecting the sample data similar to what has been proposed in the literature [27] (Sec. 3.4). In general, the results were positive (Figs. 8 and 9), but there is room for improvement, and the definition of new metrics for evaluating the stability of streaming projections is a challenging and promising future work we plan to approach.

7. Conclusions

This paper proposes a novel multidimensional projection technique called *Xtreaming*, which is shown to be one of the first reliable approaches for projecting streaming data applications. *Xtreaming* presents different novel strategies, enabling the processing of data as it is received and to adjust the projection to new emerging structures without the need for visiting the high-dimensional data more than once. The set of comparisons we provide shows that *Xtreaming* is comparable to existing out-of-sample techniques while presents a much better trade-off between quality and runtime compared to other online or incremental approaches. Moreover, the potential use of *Xtreaming* in streaming data scenarios opens possibilities for novel applications to properly process ever-growing data collections, such as the real-time analysis of social media networks. Notice that our goal is not to create a faster or more precise technique. It is about defining a strategy to fill a gap in the existing literature, appropriately handling streaming scenarios. We are currently investigating new visual metaphors for streaming data projections towards giving visual insights about structural changes and trends that emerge over time, a challenge for the next years.

Acknowledgement

Research carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP (grant 2013/07375-0). We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Butler, D. Everything, everywhere. Nature 2006;440:402–405.
[2] Gama, J. A survey on learning from data streams: current and future trends. Progress in Artificial Intelligence 2012;1(1):45–55.
[3] Morreale, P, Qi, F, Croft, P, Suleski, R, Sinnicke, B, Kendall, F. Real-time environmental monitoring and notification for public safety. IEEE MultiMedia 2010;17(2):4–11.
[4] Wang, W, Guan, X, Zhang, X. Processing of massive audit data streams for real-time anomaly intrusion detection. Computer communications 2008;31(1):58–72.
[5] Sakaki, T, Okazaki, M, Matsuo, Y. Tweet analysis for real-time event detection and earthquake reporting system development. IEEE Transactions on Knowledge and Data Engineering 2013;25(4):919–931.
[6] Nonato, LG, Aupetit, M. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. IEEE Transactions on Visualization and Computer Graphics 2019;25(8):2650–2673.
[7] Joia, P, Coimbra, D, Cuminato, JA, Paulovich, FV, Nonato, LG. Local affine multidimensional projection. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):2563–2571.
[8] De Silva, V, Tenenbaum, JB. Sparse multidimensional scaling using landmark points. Tech. Rep.; Stanford University; 2004.
[9] Silva, Vd, Tenenbaum, JB. Global versus local methods in nonlinear dimensionality reduction. In: 15th International Conference on Neural Information Processing Systems. NIPS'02; Cambridge, MA, USA: MIT Press; 2002, p. 721–728.
[10] Bengio, Y, Paiement, JF, Vincent, P, Delalleau, O, Roux, NL, Ouimet, M. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In: 16th International Conference on Neural Information Processing Systems. NIPS'03; Cambridge, MA, USA: MIT Press; 2003, p. 177–184.
[11] Espadoto, M, Hirata, NST, Telea, AC. Deep learning multidimensional projections. Information Visualization 2020;19(3):247–269.
[12] Paulovich, FV, Eler, DM, Poco, J, Botha, CP, Minghim, R, Nonato, LG. Piece wise laplacian-based projection for interactive data exploration and organization. Computer Graphics Forum 2011;30(3):1091–1100.
[13] Basalaj, W. Incremental multidimensional scaling method for database visualization. In: Erbacher, RF, Chen, PC, Wittenbrink, CM, editors. Visual Data Exploration and Analysis VI; vol. 3643. International Society for Optics and Photonics; SPIE; 1999, p. 149 – 158.
[14] Alsakran, J, Chen, Y, Zhao, Y, Yang, J, Luo, D. Streamit: Dynamic visualization and interactive exploration of text streams. In: 2011 IEEE Pacific Visualization Symposium. 2011, p. 131–138.
[15] Jenkins, OC, Matarić, MJ. A spatio-temporal extension to isomap nonlinear dimension reduction. In: Twenty-First International Conference on Machine Learning. ICML '04; New York, NY, USA: ACM. ISBN 1581138385; 2004, p. 56.
[16] Law, MH, Zhang, N, Jain, AK. Nonlinear manifold learning for data stream. In: 2004 SIAM International Conference on Data Mining. SIAM; 2004, p. 33–44.
[17] Law, MHC, Jain, AK. Incremental nonlinear dimensionality reduction by manifold learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 2006;28(3):377–391.
[18] Kouropteva, O, Okun, O, Pietikäinen, M. Incremental locally linear embedding. Pattern recognition 2005;38(10):1764–1767.

[19] Schuon, S, Durkovic, M, Diepold, K, Scheuerle, J, Markward, S. Truly incremental locally linear embedding. 1st International Workshop on Cognition for Technical Systems 2008;.

[20] Rauber, PE, Falcão, AX, Telea, AC. Visualizing time-dependent data using dynamic t-SNE. In: Eurographics / IEEE VGTC Conference on Visualization: Short Papers. EuroVis '16; Goslar, DEU: Eurographics Association; 2016, p. 73–77.

[21] Paulovich, F, Silva, C, Nonato, L. Two-phase mapping for projecting massive data sets. Visualization and Computer Graphics, IEEE Transactions on 2010;16(6):1281–1290.

[22] Saul, LK, Roweis, ST. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 2003;4:119–155.

[23] Mahapatra, S, Chandola, V. S-Isomap++: Multi manifold learning from streaming data. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE; 2017, p. 716–725.

[24] Leng, Y, Zhang, L, Yang, J. Locally linear embedding algorithm based on OMP for incremental learning. In: 2014 International Joint Conference on Neural Networks (IJCNN). 2014, p. 3100–3107.

[25] Ross, DA, Lim, J, Lin, RS, Yang, MH. Incremental learning for robust visual tracking. International Journal of Computer Vision 2008;77(1):125–141.

[26] Levey, A, Lindenbaum, M. Sequential Karhunen-Loeve basis extraction and its application to images. IEEE Transactions on Image Processing 2000;9(8):1371–1374.

[27] Fujiwara, T, Chou, J, Shilpika, S, Xu, P, Ren, L, Ma, K. An incremental dimensionality reduction method for visualizing streaming multidimensional data. IEEE Transactions on Visualization and Computer Graphics 2020;26(1):418–428.

[28] Yan, J, Zhang, B, Yan, S, Liu, N, Yang, Q, Cheng, Q, et al. A scalable supervised algorithm for dimensionality reduction on streaming data. Information Sciences 2006;176(14):2042–2065.

[29] Ye, J, Li, Q, Xiong, H, Park, H, Janardan, R, Kumar, V. IDR/QR: An incremental dimension reduction algorithm via QR decomposition. IEEE Transactions on Knowledge and Data Engineering 2005;17(9):1208–1222.

[30] Hinton, GE, Salakhutdinov, RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–507.

[31] Espadoto, M, Martins, RM, Kerren, A, Hirata, NST, Telea, AC. Toward a quantitative survey of dimension reduction techniques. IEEE Transactions on Visualization and Computer Graphics 2021;27(3):2153–2173.

[32] Van der Maaten, L, Hinton, G. Visualizing data using t-SNE. Journal of Machine Learning Research 2008;9(2579-2605):85.

[33] Torgerson, WS. Multidimensional scaling of similarity. Psychometrika 1965;30(4):379–393.

[34] Sammon, JW. A nonlinear mapping for data structure analysis. IEEE Transactions on Computers 1969;18(5):401–409.

[35] Tenenbaum, JB, de Silva, V, Langford, JC. A global geometric framework for nonlinear dimensionality reduction. Science 2000;290(5500):2319–2323.

[36] Roweis, ST, Saul, LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000;290(5500):2323–2326.

[37] Pekalska, E, de Ridder, D, Duin, RPW, Kraaijveld, MA. A new method of generalizing Sammon mapping with application to algorithm speed-up. In: Boasson, M, Kaandorp, JA, Tonino, JFM, Vosselman, MG, editors. 5th Annual Conference of the Advanced School for Computing and Imaging (ASCI'99). Delft, Netherlands; 1999, p. 221–228.

[38] Brandes, U, Pich, C. Eigensolver methods for progressive multidimensional scaling of large data. In: Kaufmann, M, Wagner, D, editors. Graph Drawing. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-70904-6; 2007, p. 42–53.

[39] Paulovich, FV, Nonato, LG, Minghim, R, Levkowitz, H. Least square projection: A fast high precision multidimensional projection technique and its application to document mapping. IEEE Transactions on Visualization and Computer Graphics 2008;14(3):564–575.

[40] Gower, JC, Dijksterhuis, GB. Procrustes problems; vol. 3. Oxford University Press Oxford; 2004.

[41] Steinbach, M, Karypis, G, Kumar, V. A comparison of document clustering techniques. In: Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining (KDD'00). Boston, Massachusetts, USA: ACM; 2000, p. 109–110.

[42] Joia, P, Petronetto, F, Nonato, LG. Uncovering representative groups in multidimensional projections. Computer Graphics Forum 2015;34(3):281–290.

[43] Pokrajac, D, Lazarevic, A, Latecki, LJ. Incremental local outlier detection for data streams. In: 2007 IEEE Symposium on Computational Intelligence and Data Mining. 2007, p. 504–515.

[44] Cox, TF, Cox, MAA. Multidimensional Scaling. Second ed.; Chapman & Hall/CRC; 2000.

[45] Archambault, D, Purchase, HC. Mental map preservation helps user orientation in dynamic graphs. In: Didimo, W, Patrignani, M, editors. Graph Drawing. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013, p. 475–486.

[46] Diehl, S, Görg, C. Graphs, they are changing. In: Goodrich, MT, Kobourov, SG, editors. Graph Drawing. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-36151-0; 2002, p. 23–31.

[47] Diehl, S, Görg, C, Kerren, A. Preserving the mental map using foresighted layout. In: Ebert, DS, Favre, JM, Peikert, R, editors. Data Visualization 2001. Vienna: Springer Vienna. ISBN 978-3-7091-6215-6; 2001, p. 175–184.

[48] Cantareira, GD, Paulovich, FV. A Generic Model for Projection Alignment Applied to Neural Network Visualization. In: Turkay, C, Vrotsou, K, editors. EuroVis Workshop on Visual Analytics (EuroVA). The Eurographics Association. ISBN 978-3-03868-116-8; 2020,.

[49] Vernier, EF, Garcia, R, Silva, IPd, Comba, JLD, Telea, AC. Quantitative evaluation of time-dependent multidimensional projection techniques. Computer Graphics Forum 2020;39(3):241–252.

[50] Neves, TTAT, Fadel, SG, Hilasaca, GM, Fatore, FM, Paulovich, FV. UPDis: A user-assisted projection technique for distance information. Information Visualization 2018;17(4):269–281.

[51] Dua, D, Graff, C. UCI machine learning repository. 2017. URL: http://archive.ics.uci.edu/ml.

[52] LeCun, Y, Bottou, L, Bengio, Y, Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998;86(11):2278–2324.

[53] Whalen, D, Norman, ML. Competition data set and description. 2008 IEEE Visualization Design Contest; 2008.

[54] Caruana, R, Joachims, T, Backstrom, L. KDD-Cup 2004: results and analysis. ACM SIGKDD Explorations Newsletter 2004;6(2):95–108.

[55] Kruskal, JB. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 1964;1(29):115–129.

[56] Theodoridis, S, Koutroumbas, K. Chapter 16 - cluster validity. In: Theodoridis, S, Koutroumbas, K, editors. Pattern Recognition (Fourth Edition); fourth edition ed. Boston: Academic Press. ISBN 978-1-59749-272-0; 2009, p. 863–913.

[57] Venna, J, Kaski, S. Visualizing gene interaction graphs with local multidimensional scaling. In: ESANN'2006 - European Symposium on Artificial Neural Networks; vol. 6. Citeseer; 2006, p. 557–562.

[58] Jolliffe, IT. Principal Component Analysis. Springer-Verlag; 1986.

[59] Mohammad, SM. Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In: Meiselman, HL, editor. Emotion Measurement. Woodhead Publishing. ISBN 978-0-08-100508-8; 2016, p. 201–237.

[60] Mohammad, SM, Sobhani, P, Kiritchenko, S. Stance and sentiment in tweets. ACM Transactions on Internet Technology 2017;17(3):26:1–26:23.

[61] Hutto, C, Gilbert, E. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In: Eighth International AAAI Conference on Weblogs and Social Media. ICWSM '14; 2014,.

[62] Skeppstedt, M, Simaki, V, Paradis, C, Kerren, A. Detection of stance and sentiment modifiers in political blogs. In: International Conference on Speech and Computer. Springer; 2017, p. 302–311.

[63] Azar, M. Argumentative text as rhetorical structure: An application of rhetorical structure theory. Argumentation 1999;13(1):97–114.

[64] Vernier, EF, Comba, JLD, Telea, AC. Guided stable dynamic projections. Computer Graphics Forum 2021;40(3):87–98.