A visual approach for user-guided feature fusion

Gladys M. Hilasaca
University of São Paulo (ICMC-USP)
São Carlos, SP, Brazil

Fernando V. Paulovich
University of São Paulo (ICMC-USP)
São Carlos, SP, Brazil

***Abstract*—Dimensionality Reduction transforms data from high-dimensional space into visual space preserving the existing relationships. This abstract representation of complex data enables exploration of data similarities, but brings challenges about the analysis and interpretation for users on mismatching between their expectations and the visual representation. A possible way to model these understandings is via different feature extractors, because each feature has its own way to encode characteristics. Since there is no perfect feature extractor, the combination of multiple sets of features has been explored through a process called feature fusion. Feature fusion can be readily performed when machine learning or data mining algorithms have a cost function. However, when such a function does not exist, user support needs to be provided otherwise the process is impractical. In this project, we present a novel feature fusion approach that employs data samples and visualization to allow users to not only effortlessly control the combination of different feature sets but also to understand the attained results. The effectiveness of our approach is confirmed by a comprehensive set of qualitative and quantitative experiments, opening up different possibilities for user-guided analytical scenarios. The ability of our approach to provide real-time feedback for feature fusion is exploited in the context of unsupervised clustering techniques, where users can perform an exploratory process to discover the best combination of features that reflects their individual perceptions about similarity.**

A traditional way to visualize data similarities is via scatter plots, however, they suffer from overlap issues. Overlapping hides data distributions and makes the relationship among data instances difficult to observe, which hampers data exploration. To tackle this issue, we developed a technique called Distance-preserving Grid (DGrid). DGrid employs a binary space partitioning process in combination with Dimensionality Reduction output to create orthogonal regular grid layouts. DGrid ensures non-overlapping instances because each data instance is assigned only to one grid cell. Our results show that DGrid outperforms the existing state-of-the-art techniques, whereas requiring only a fraction of the running time and computational resources rendering DGrid as a very attractive method for large datasets.

I. Introduction

Nowadays, data is a critical component for intelligent decision making. For example, business analysts use data for decision making and observe a remarkable increase in the adoption of visualization as an important tool to guide decision making. Visualization provides an effective bridge for analysts to make sense of the data, and unveil interesting insights [1]. A way to involve users in an analysis process using visualization is known as Visual Analytics (VA). VA provides a visual interface between automated techniques and users to effectively combine their strengths [2], [3]. This duality results in a closely communication between users and machines, where computational results are communicated by visualizations and user feedback is expressed by interactions.

One of the most common techniques in VA is Dimensionality Reduction (DR) [4]–[6]. DR transforms data to a lower-dimensional space, so that salient structures or patterns are perceived while exploring data similarities. However, DR might provide a non-understandable visualization for users, because they might also have other insights on similarities. Hence, there is a disconnect between how users and machines perceive data similarities. One possible way to model user understandings are via different feature extractors. An image can be described with different characteristics such as color, texture, shape, and so on. Among these features, we would like to find the best feature for a specific task. However, each feature provides complementary information, and there is no perfect one. A possible solution is to integrate all features in a process known as feature fusion. A simple strategy is to concatenate all features into one single feature vector [7]. Another strategy is weighing features automatically. Typically, the learned weights aim to improve an objective function. For example, [8] optimize an error function that decides which feature to combine through a deep learning model. Usually, this combination improves an optimization function associated with an evaluation metric (e.g. accuracy). However, when an optimization function is not available or there is some subjectivity associated, the most reliable source are users.

Since the similarity between data instances is on the eye of the viewer, it is desirable to incorporate users understanding and control the semantic of similarities. In this thesis, we introduce **User-guided feature fusion**. Our goal is to generate an interpretable feature fusion using a visual representation and an initial weighting feature combination, which is defined by users according to their point of view. In this process, first a sample of each feature is selected, and they are mapped to a common space preserving the distance relationships of the individual feature, therein an alignment of all features are performed to ensure consistency among features. This mapped sample is used to configure the weights for the feature fusion process. Then, the sample and the initial weights configuration are the input for a local affine transformation. This transformation propagates the user-defined semantic of similarity to the whole data ensuring that user-defined understanding is preserved.

When analyzing the similarity between instances, scatter plot is usually used. However, scatter plot might produce

layouts with overlapped graphical elements and suffers from occlusion problems, which can hamper exploratory data analysis. To address such limitation, some approaches employ post-processing strategies [9], [10] or put constraints on the projection process [11] to remove the overlapping. However, they make poor use of the visual space, creating layouts with void areas. In order to make better use of the available space, distance preserving grid techniques have been devised to arrange the graphical elements into grids, using as much as possible the visual space. Currently, the state-of-the-art approaches to produce distance-preserving grids solve assignment problems [12], [13] or use permutations to optimize cost functions [14], [15]. Although precise, such strategies are computationally expensive, limited to small datasets. However, in the era of Big Data, there is a clear need of simple and efficient techniques for creating grid layouts. In this thesis, we introduce a novel approach, called **Distance-preserving Grid (DGrid)** that combines DR techniques with a space-partitioning strategy to create orthogonal regular distance-preserving grids. Our process maps points from R^2 to a grid. The grid is split in half on the axis with more elements. Next, we assign the first half of the points to the top or left grid. The remaining half is assigned to the bottom or right half of the grid. This process is repeated recursively until there is only one point in each grid cell. Despite its simplicity, the quality of the produced layouts and the running times render DGrid as a very attractive method for large datasets.

II. Our contributions

The main goal of this thesis is to allow users to control similarity relationships according to properties in different features via visual representations. Specifically, we investigate the following hypothesis:

> *"Visual representations will support users to control similarity relationships for feature fusion matching users' point of view, in comparison to methods without using these representations."*

The contributions of this thesis are threefold:

- A novel visual approach for feature fusion, users visually correlate features to match their point of view regarding similarity. To the best of our knowledge, this is the first time that visual representations are exploited as a mechanism for feature fusion;
- A novel distance-preserving grid layout technique, that preserves distance and neighborhood, while running in a fraction of time in relation to current state-of-the-art techniques;
- A framework to explore multimedia data that allows real-time tuning of the semantics of the similarity between instances to match user's expectations and the navigation of large collections into different levels of detail.

In Section III and IV, we present our two approaches with their respective evaluations. Finally, in section V, our conclusion is presented.

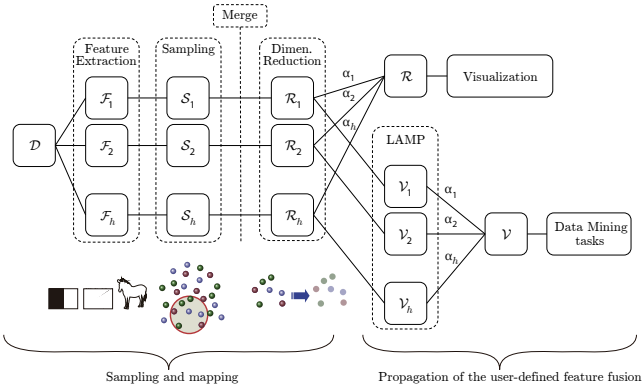

Fig. 1. Overview of our process for feature fusion. Initially a sample is extracted, combined and visualized. Based on this, the user can test different weights to fuse the features and observe the outcome. Once sample combination reflects the user expectation, the same weights are used to combine the complete sets of features that can them be used on subsequent tasks, such as clustering.

III. User-guided Feature Fusion

Our approach for feature fusion employs a two phase strategy to support users on defining combinations that reflect a particular point-of-view regarding similarity relationships. In the first phase, samples $S_1, S_2, \ldots, S_h$ are extracted from each different set of features $F_1, F_2, \ldots, F_h$ and merged so that each set S_i presents the same objects but represented using the different types of features. Each sample S_i is then mapped to a vectorial representation $R_i \in \mathbb{R}^m$ preserving as much as possible the distance relationships between the instances. These vectorial representations are then combined to generate a single representation $\overline{R} = \alpha_1 R_1 + \alpha_2 R2 + \ldots + \alpha_h R_h$, which is then visualized.

The user can then change the feature weights and observe the outcome. Once the sample visualization reflects the user expectations, that is, once the proper weights $\alpha_1, \alpha_2, \ldots, \alpha_h$ are found, the second step takes place and the defined weights are used to combine the complete set of features. In this process, the vectorial sample representations $R_1, R_2, \ldots, R_h$ and the samples $S_1, S_2, \ldots, S_h$ are used to construct models to map each set of feature F_i to a vectorial representation $V_i \in \mathbb{R}^m$. Since these vectorial representations are embedded in the same space, they can be combined using the weights $\alpha_1, \alpha_2, \ldots, \alpha_h$, obtaining the final vectorial representation $\overline{V}$ that matches the user's expectations defined by the sample visualization. Figure 1 outlines our approach showing the involved steps. Next, we detail these steps, starting with the sampling and the dimensionality reduction.

A. Sampling and Mapping

Since users employ the sample visualization to guide the feature fusion process, it is important to have all possible data structures from the different features represented. Therefore, we recover samples from each different set of features so as to faithfully represent the distribution of each individual set.

In this process, we can extract samples from each set $F_1, F_2, \ldots, F_h$ separately using a cluster-based strategy. After

extracting the sample sets $S_1, S_2, \ldots, S_h$, we merge their indexes into a unified set of indexes. Then we recreate the sets $S_1, S_2, \ldots, S_h$ to have the instances with the indexes contained in the unified set of indexes. Therefore, all sample sets have the same instances q, which is mandatory for the sample visualization given that we visualize the combination of all features $\overline{R}$. Also we guarantee that the structures defined by the different types of features are represented by the samples.

After recovering the samples, we map them to a common m-dimensional space, obtaining their vectorial representation $R_1, R_2, \ldots, R_h \in \mathbb{R}^m$ so that we can combine them to obtain $\overline{R} \in \mathbb{R}^m$ (for the sample visualization). In this process, each set of samples S_i is mapped to $\mathbb{R}^m$ preserving as much as possible the distance relationships. We do this by minimizing

$$E_{st}(S_i) = \frac{1}{q^2} \sum_{i}^{q} \sum_{j}^{q} \left(\delta(s_i^i, s_j^i) - ||r_i^i - r_j^i|| \right)^2 \quad (1)$$

where s_i^i and s_j^i are instances in S_i, $\delta(s_i^i, s_j^i)$ is the distance between them, and r_i^i and r_j^i are the vectorial representations in the m-dimensional space of s_i^i and s_j^i, respectively.

Besides preserving distance relationships, our mapping process aims to align the vectorial representations so that r_i^i is placed as close as possible to $r_i^j \ \forall \ j \in [1,h]$ without affecting the distance preservation of the individual mappings. This is necessary since the unified sample representation is calculated as a convex combination of these representations, that is, $\overline{R} = \alpha_1 R_1, \alpha_2 R_2, \ldots, \alpha_h R_h$, with $\sum \alpha_i = 1$, and misalignments could result in meaningless unified representations. We first calculate the normalized average distance matrix $\overline{\Delta} = \frac{1}{h} \sum_i^h \Delta_{S_i}$ by combining the distance matrices of all sets of features, where Δ_{S_i} is the distance matrix calculated from S_i. Then we map $\overline{\Delta}$ to the m-dimensional space using the Equation (1). The idea is to use this average representation as a guide to align the vectorial representations $R_1, R_2, \ldots, R_h$ minimizing

$$E_{al}(S_i) = \frac{1}{q^2} \sum_{i}^{q} \sum_{j}^{q} \left(d(\overline{r}_i, \overline{r}_j) - ||\overline{r}_i - r_j^i|| \right)^2 \quad (2)$$

where $d(\overline{r}_i, \overline{r}_j)$ is the distance between two instances of the average vectorial representation.

Joining Equation (1) and (2) we define the function we optimize in our mapping process to preserve, as much as possible, the distance relationships of the original features $S_1, S_2, \ldots, S_h$ in the vectorial representations $R_1, R_2, \ldots, R_h \in \mathbb{R}^m$ while aligning them. This function is given by

$$E(F_i) = \lambda \cdot E_{st}(S_i) + (1 - \lambda) \cdot E_{al}(S_i) \quad (3)$$

where λ is used to control the importance of the distance preservation and the alignment to the produced vectorial representations. λ is a hyperparameter and can be changed to define a good tradeoff between distance preservation and alignment. To minimize Equation (3) we use stochastic gradient descent with a polynomial decay learning rate.

B. Weighted Feature Combination

Given the sample vectorial representations $R_1, R_2, \ldots, R_h$ we build a set of functions using the process defined in [16] to map each feature set F_i into its vectorial representation $V_i \in \mathbb{R}^m$ preserving as much as possible the distance relationships while obeying the geometry defined in R_i. In this process, each instance $f_j^i \in F_i$ is mapped to the m-dimensional space through an orthogonal local affine transformation $T_j^i : \mathbb{R}^{q^i} \rightarrow \mathbb{R}^m$, where q^i is the dimensionality of F_i.

The affine transformation $T_j^i(f) = fM + t$ associated to f_j^i is defined so as to minimize:

$$\sum_k \beta_k \| T_j^i(s_k^i) - r_k^i \|^2 \quad (4)$$

where $\beta_k = \| s_k^i - f_k^i \|^{-2}$, with s_k^i the original feature representation of the k-th sample in S_i.

The sample vectorial representations $R_1, R_2, \ldots, R_h$ dictates the geometry of the embeddings $V_1, V_2, \ldots, V_h$. Since they are aligned by the mapping process defined in the previous section, the linear combination $V = \alpha_1 V1, \alpha_2 V_2, \ldots, \alpha_h V_h$ can be performed to obtain the final embedding V that incorporates the structures defined by each set of features, weighted according to the user's point-of-view. For more information about this affine transformation and how the sample vectorial representation controls the final results, please refer to [16].

C. Experimental Validation

In this section, we evaluate our mapping and feature combination processes using different datasets in order to show that the sample manipulation effectively controls the complete feature fusion. Next, we describe the employed datasets, detail how we extract features, and present our evaluations.

Datasets

We use five datasets in our tests, named STL-10 [17], Animals [18], Zappos [19], CIFAR-10 [20] and Photographers [21]. These datasets come from different domains.

Features

We use 4 distinct methods to extract features, representing low-level and high-level image components. For the low-level features, we represent (1) color with LAB color histogram; (2) texture with Gabor filters [22]; and (3) shape with HoG technique [23]. For the high-level, we extract deep-features from the *pool5* layer using a pre-trained CNN CaffeNet [24].

Quantitative Evaluation

To confirm the quality of our approach, we quantitatively evaluate our mapping and feature combination processes. For the mapping process evaluation, the five datasets are sampled 10 times randomly reducing them to 5% of their original sizes. We sample the data since we cannot execute the mapping process with large datasets since its memory footprint is $O(n^2)$. Each different feature from the dataset has its own dimensionality. To ensure a common dimensional space, we calculate the intrinsic dimensionality for each of them and choose the smallest value. This value is used to do the mapping. The minimum values of intrinsic dimensionality are 57, 71, 91, 41, and 83 for STL-10, Animals, Zappos, CIFAR-10, and Photographer datasets, respectively.

We use stress and alignment error to evaluate the mapping process (see Equation 1 and Equation 2, respectively). We

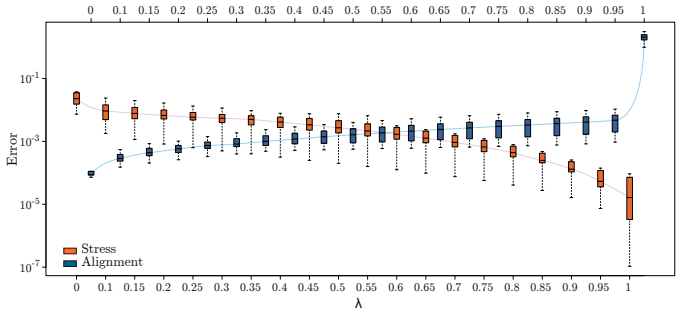

Fig. 2. Comparison for distance preservation and alignment error with varying λ. The best trade-off is achieved in the range $[0.45-0.65]$. Line connects the mean values of all box plots.

summarize our results in Figure 2 varying the value of λ. The stress boxplots (in orange) decrease as λ increases. On the other hand, alignment boxplots (in blue) have the opposite behavior. This is the expected outcome since larger values of λ preserve the distance relationships, whereas smaller values align the data.

Setting $\lambda = 1$ preserves as much as possible the original distance relationships. This is reflected by the average stress $\overline{E_{st}} = 0.0009$, but it does not ensure a good alignment ($\overline{E_{al}} = 2.03$). On the other hand, $\lambda = 0$ delivered almost a perfect alignment ($\overline{E_{al}} = 0.0001$), but it does not enforce the distance preservation ($\overline{E_{st}} = 0.03$). In this work, we are interested in the best trade-off between distance preservation and alignment so that the alignment is obtained without penalizing the overall distance preservation of the mappings. According to our experiments, we achieved this in the range $\lambda = [0.45, 0.65]$, where both stress and alignment errors are nearly 0 for our experiments (see Figure 2).

For visual analysis, we generate 2D representations of the samples using our mapping process and setting the target dimensionality to two. We show the result for the **Zappos** dataset in Figure 3. Here, the points are colored according to image classes. The stress and alignment error values are shown at the top-left corner of each scatterplot. To show the influence of different λ in the mapping process, we vary it in the range $[1.0, 0.2]$. The first column shows the result produced using $\lambda = 1.0$, best preserving the original distance relationship. Notice that the visual representations of each different feature are misaligned among themselves. The second column depicts results with $\lambda = 0.8$. Now, the 2D mappings start to align (points of representing images of the same class are placed in close positions). We observe a small increase in the stress error, but the alignment error decreases considerably compared to the first column (see the second measure at the top-left corner). The same behavior is verified in the remaining columns. The last column almost completely aligns all features. As expected, as lambda decreases, the distance preservation also decreases, and the alignment improves. However, the stress changes are minimal. Hence, our approach is capable of making a good alignment between features while preserving distance relationships. For qualitative results please refer section 5.2 from the thesis document.

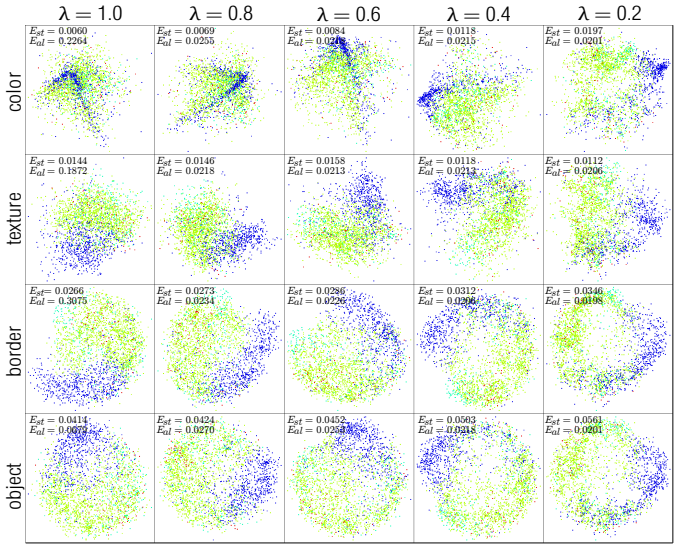

Fig. 3. Resulting mapping process for the **Zappos** dataset. As λ decreases, the features get more aligned (See column 5). Top-left numbers correspond to stress and alignment error.

IV. Distance Preserving Grid Layouts

We propose Distance-preserving Grid (DGrid), a technique to efficiently arrange layouts generated by DR techniques. It employs a two step approach to generate uniform grids that preserve, as much as possible, the distances relationships of a given dataset. In the first step, the data instances $\mathscr{D} = \{d_1, d_2, \ldots, d_N\} \in \mathbb{R}^m$ are mapped into points on the plane using a multidimensional projection technique, obtaining their 2D Cartesian Coordinates $\mathscr{P} = \{p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_N = (x_N, y_N)\} \in \mathbb{R}^2$. Then, a grid $\mathscr{G} = \{g_{1,1}, g_{1,2}, \ldots, g_{1,u}, \ldots, g_{u,1}, g_{u,2}, \ldots, g_{w,u}\}$ with w rows and u columns, where $w \times u \geq N$, is created, assigning each projected instance p_i to a grid cell $g_{p,q}$.

The reasoning behind our approach is that if the projection $\mathscr{P}$ precisely preserves the distances relationships in $\mathscr{D}$, and if $\mathscr{G}$ preserves the geometry of $\mathscr{P}$, $\mathscr{G}$ will preserve the distances relationships in $\mathscr{D}$. Consider that $\mathscr{P}$ has been obtained from $\mathscr{D}$. If the points in $\mathscr{P}$ are uniformly distributed over the plane and are arranged following the number of rows and columns of the target grid, the process to assign $\mathscr{P}$ to $\mathscr{G}$ is trivial. In practice, the assumption that the projection is uniformly distributed over the plane and follows the grid pattern seldom holds. Seeking to approximate such constraints, we recursively bisect the projection into non-overlapping partitions until the obtained partitions individually obey, as much as possible, such constraints. Then, (sub)grids are derived from each partition. For the first bisection, consider $\mathscr{P}$ as the input projection, and (w,u) the dimension of the target grid. If $w > u$, we split $\mathscr{P}$ horizontally, obtaining two partitions $\mathscr{P} = \mathscr{P}_1 \cup \mathscr{P}_2$, so that, the upper partition $\mathscr{P}_1$ contains enough instances to completely fill half of the desired grid, that is, $|\mathscr{P}_1| = \lceil w/2 \rceil \times u$. Otherwise, we split $\mathscr{P}$ vertically, so that the left partition $\mathscr{P}_1$ contains enough instances to completely fill half of the desired grid, that is, $|\mathscr{P}_1| = w \times \lceil u/2 \rceil$.

To compute the cells' indexes from the partitions, we

calculate during the bisecting process the indexes of the top-left corner cells of each (sub)grid resulted from each partition.

The process of bisecting and calculating the top-left corner indexes are successively applied until the resulting partitions obey the uniform distribution and grid pattern constraints. However, using this as a stop criterium would penalize the computational cost of the overall algorithm since it is an $O(N^2)$ procedure for a partition containing N instances. Instead, we execute the bisecting and corner computation process until each partition contains only one instance.

A. Experimental Validation

In this section, we present a quantitative evaluation of the DGrid technique, comparing it against the state-of-the-art in distance preservation grid techniques, Kernelized Sorting (KS) [13], Self-Sorting Map (SSM) [15], and IsoMatch [12]. In this comparison we use three different quality metrics, k-neighborhood preservation index (NP_k), cross-correlation (CC) [15], and energy function [12]. The k-neighborhood preservation index was originally developed to evaluate projections, but here we use it to measure how much the neighborhood in the dataset $\mathscr{D}$ is preserved in the grid $\mathscr{G}$. NP_k ranges in $[0,1]$, the larger the value the better the result. The cross-correlation measures how well the placements of the data instances in the grid correlate to the dissimilarities among them. The cross-correlation ranges in $[-1,1]$, the large the better. In this work, we normalize the cross-correlation in $[0,1]$ using $CC' = (CC+1)/2$ to easier the comparison among the different metrics. Finally, the energy function measures how well the pairwise distances between the data instances are preserved by the corresponding distances in the grid. It ranges in $[0,1]$ with larger values rendering better results.

For the tests, we have selected all datasets from the *UCI Machine Learning Repository* [25] with real-valued attributes and sizes varying between 100 and $2,500$ instances, allowing the comparison of the techniques in different scenarios. We only get real-valued datasets so that (Euclidean) distances can be properly calculated, and we limited their sizes due to the high computational complexities and running times of KS and IsoMatch. Also, we have discarded the datasets with missing values, resulting in 38 datasets.

Figure 4 presents boxplots summarizing the results of each technique considering all the 38 datasets. For DGrid, we report results using t-SNE and LAMP techniques to generate the input projections. Although IsoMatch originally employs ISOMAP as input, we also report results using t-SNE and LAMP, so it is possible to compare DGrid and IsoMatch isolating the projection contribution to the quality of the produced results. DGrid, IsoMatch, and KS techniques are deterministic, so we only run each technique once for each dataset. Given the random initialization of the SSM technique, we run it 30 times for each dataset. In Figure 4, the boxplots in red represent the results of the projections (LAMP and t-SNE) used as input by DGrid and IsoMatch techniques. They serve only as baselines to show the correlation between projection quality and grid quality. Notice that, the drop in precision

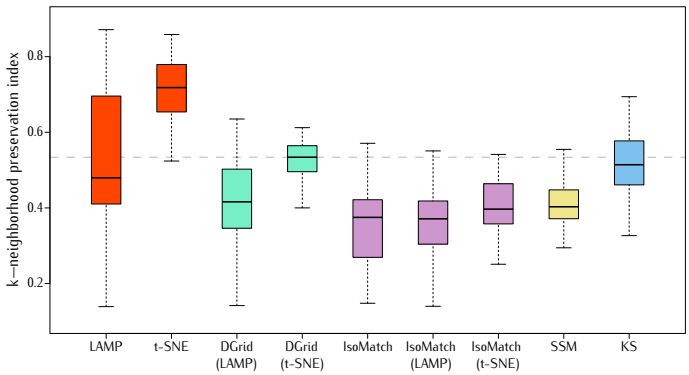

Fig. 4. Boxplots of k-neighborhood preservation index. In this aspect, the DGrid surpass (on average) current state-of-the-art techniques, indicating its quality on preserving distance relationships. The boxplots in red summarize the results of the input projection and serve as baselines to show the correlation between projections and grid properties. They are not intend for direct comparisons.

between the projections and the produced grids is expected since the techniques we use do not create uniformly distributed projections. Also note that direct comparisons only make sense among grid layouts, not among grids and projections.

Regarding the k-neighborhood preservation index, the best result was attained by the DGrid with t-SNE as input ($\overline{NP} = 0.52$), better than the other more costly counterparts, IsoMatch ($\overline{NP} = 0.36$) and KS ($\overline{NP} = 0.50$). DGrid presents not only the largest mean but also the smallest spread regarding the best and worst results. Comparing the different flavors of DGrid, the results produced using the t-SNE are also considerably superior than the results produced using the LAMP. This is an expected outcome since the formulation of t-SNE favors the preservation of small neighborhoods instead of a global distance preservation as conveyed by the LAMP, which is confirmed by the boxplots of the projections in red. This indicates the impact of the input projection on the produced grid, and also shows that our strategy for assigning the projection to grid cells satisfactorily preserves the input geometry. In this example, we approximate the neighborhood size k to 5% of the dataset size, setting $k = (\lfloor\sqrt{0.05 * N}\rfloor)^2$. The results for the remaining metrics are in the Ph.D. thesis.

To complement the statistical analysis conveyed by the boxplots, providing more detailed information, we show in Figures 5 the resulting grids for some selected datasets. Aiming at showing different aspects of each technique, we choose datasets with varied distance distributions. In these figures, the cells are colored according to different quality metrics calculated for each cell. The cells colored in black are empty. They exist because we have more cells than instances in these examples. Notice that DGrid, KS, and IsoMatch place all empty cells on the grid borders whereas SSM open spaces inside the grid. The quality metric values are shown below each grid, and the best results are highlighted using a bold font. Although KS is marginally better in one case and presents the same quality as DGrid in other cases, it is an $O(N^3)$ technique and cannot address problems involving large datasets. DGrid is much less expensive so not only small examples can be

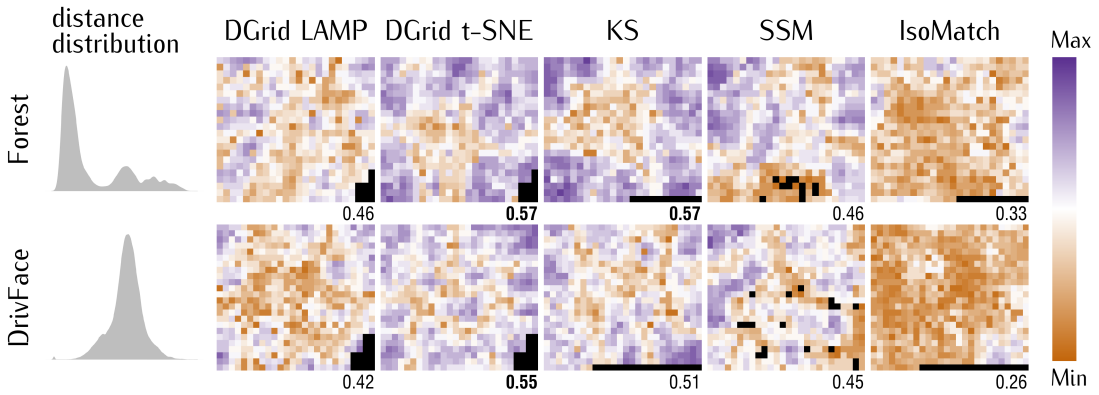

Fig. 5. Resulting grids colored according to the **k-neighborhood preservation index**. SSM technique groups bad quality cells close to the empty cells, showing the negative impact of empty spots on the produced layouts.

processed in a fraction of the time but also it can address larger problems that neither KS nor IsoMatch are capable of.

Finally, we compared DGrid with SSM regarding the running times. We removed the other techniques from this comparison since they are computationally expensive. In this test, we selected 10 datasets from UCI, varying the sizes from 17,000 to 130,000 instances. Figure 6 summarizes the results. Besides the boxplots for DGrid and SSM techniques, the figure shows individual boxplots for the projection and the grid assignment steps. In this example, we are using LAMP to project the data. In average, DGrid is almost two orders of magnitude faster than SSM, but better results can be obtained if a faster projection technique is employed (the projection step dominates the process). Considering the tested techniques, DGrid presents the best tradeoff between running times and quality of the produced grids, placing it among the state-of-the-art techniques for generating distance preserving grids of large datasets.

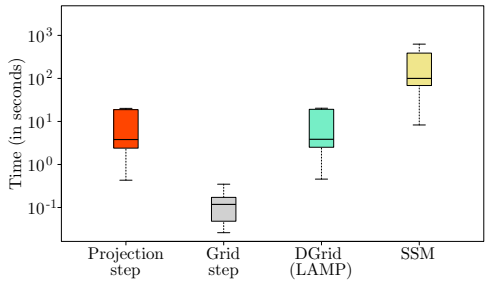

Fig. 6. Running times boxplots. DGrid is almost two orders of magnitude faster than the SSM technique, and the projection phase dominates its running times.

V. Conclusions

In this thesis, we addressed the problem of feature fusion when there is no objective function to optimize or when there is a degree of subjectivity in the process. In particular, we involve users in this process allowing them to control the feature fusion process according to their expectations. This process is performed on a sample of features from which users define an initial weighted feature combination. Such weights are propagated to combine the whole dataset. Experiments show that the fusion of the complete dataset preserves the initial user-defined semantic.

For visualization purposes, we proposed a novel approach for generating grid layouts that preserve distance information, called Distance-preserving Grid (DGrid). We provide a set of comparisons, which show that DGrid outperforms the existing state-of-the-art techniques. DGrid is two orders faster than existing techniques and it is scalable for large datasets.

We integrate our approaches into a framework to explore image collections that allows real-time tuning of similarity between images to matching user's expectations and navigation of large collections into different levels of detail.

This thesis also contributes to broader areas such as visual analytics and visualization. In visual analytics, user's interactions are important in the analysis process. In our approach, users perform weights combination in a very simple way and they can explore different combinations interactively in real-time. We also advance the state-of-the-art user-based visualization by defining a new way to capture user understanding in feature fusion. This complements traditional similarity-based user interactions, where users individually manipulate points from the visual representation [26]. On visual metaphors side, scatterplots are a simple and intuitive way of visualizing 2D point data. Scatterplots can display data trends, and can allow outlier identification because regions with a higher density of points will be grouped perceptually, which produce overlapped data instances. DGrid and other grid-based techniques solve overlapping problems assigning each point to a single cell. Although DGrid exceeds state-of-the art grid techniques in relation to complexity and precision in preserving distance relationship, it can not replace traditional scatterplots, because DGrid does not preserve data structure. Therefore, a technique that maintains a trade-off between these two metaphors is desirable. A possible solution is to deform the grid using distance information to allow separation between groups. We are developing this idea and tackling the limitations of this thesis.

REFERENCES

[1] A. C. Telea, *Data Visualization: Principles and Practice, Second Edition*, 2nd ed. Natick, MA, USA: A. K. Peters, Ltd., 2014.

[2] D. A. Keim, J. Kohlhammer, G. P. Ellis, and F. Mansmann, *Mastering the Information Age - Solving Problems with Visual Analytics.* Eurographics Association, 2010.

[3] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. Keim, "Knowledge generation model for visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1604–1613, 2014.

[4] D. H. Jeong, C. Ziemkiewicz, B. D. Fisher, W. Ribarsky, and R. Chang, "ipca: An interactive system for pca-based visual analytics," *Comput. Graph. Forum*, vol. 28, pp. 767–774, 2009.

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

[6] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim, "Visual interaction with dimensionality reduction: A structured literature analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 241–250, Jan. 2017.

[7] U. G. Mangai, S. Samanta, S. Das, and P. R. Chowdhury, "A survey of decision fusion and feature fusion strategies for pattern classification," *IETE Technical Review*, vol. 27, no. 4, pp. 293–307, 2010.

[8] G. Ma, X. Yang, B. Zhang, and Z. Shi, "Multi-feature fusion deep networks," *Neurocomput.*, vol. 218, no. C, pp. 164–171, Dec. 2016.

[9] E. Gomez-Nieto, F. S. Roman, P. Pagliosa, W. Casaca, E. S. Helou, M. C. F. de Oliveira, and L. G. Nonato, "Similarity preserving snippet-based visualization of web search results," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 3, pp. 457–470, March 2014.

[10] H. Strobelt, M. Spicker, A. Stoffel, D. Keim, and O. Deussen, "Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives," *Computer Graphics Forum*, vol. 31, no. 3pt3, pp. 1135–1144, 2012.

[11] R. D. Pinho, M. C. Oliveira, and A. Andrade Lopes, "An incremental space to visualize dynamic data sets," *Multimedia Tools Appl.*, vol. 50, no. 3, pp. 533–562, Dec. 2010.

[12] O. Fried, S. DiVerdi, M. Halber, E. Sizikova, and A. Finkelstein, "Isomatch: Creating informative grid layouts," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 155–166, May 2015.

[13] N. Quadrianto, A. J. Smola, L. Song, and T. Tuytelaars, "Kernelized sorting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1809–1821, Oct 2010.

[14] G. Strong and M. Gong, "Data organization and visualization using self-sorting map," in *Proceedings of Graphics Interface 2011*, ser. GI '11. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2011, pp. 199–206.

[15] ——, "Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts," *Trans. Multi.*, vol. 16, no. 4, pp. 1045–1058, Jun. 2014.

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

[17] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, 2011, pp. 215–223.

[18] C. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *CVPR 2009*, Max-Planck-Gesellschaft. Piscataway, NJ, USA: IEEE Service Center, Jun. 2009, pp. 951–958.

[19] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *Computer Vision and Pattern Recognition (CVPR)*, Jun 2014.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[21] C. Thomas and A. Kovashka, "Seeing behind the camera: Identifying the authorship of a photograph," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[22] L. Chen, G. Lu, and D. Zhang, "Effects of different gabor filter parameters on image retrieval by texture," in *Proceedings of the 10th International Multimedia Modelling Conference*, ser. MMM '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 273–.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *In CVPR*, 2005, pp. 886–893.

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 675–678. [Online]. Available: http://doi.acm.org/10.1145/2647868.2654889

[25] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: {http://archive.ics.uci.edu/ml}

[26] G. M. H. Mamani, F. M. Fatore, L. G. Nonato, and F. V. Paulovich, "User-driven Feature Space Transformation," *Computer Graphics Forum*, 2013.