UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

A visual approach for user-guided feature fusion

Gladys Marleny Hilasaca Mamani

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Gladys Marleny Hilasaca Mamani

A visual approach for user-guided feature fusion

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Fernando Vieira Paulovich

USP – São Carlos
February 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

H641v

Hilasaca Mamani, Gladys Marleny
A visual approach for user-guided feature fusion / Gladys Marleny Hilasaca Mamani; orientador Fernando Vieira Paulovich. -- São Carlos, 2019.
117 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2019.

1. Feature fusion. 2. Grid visualization. 3. Distance preserving grids. 4. Dimensionality reduction. 5. Exploratory Data Visualization. I. Vieira Paulovich, Fernando , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Gladys Marleny Hilasaca Mamani

Uma abordagem visual para fusão de características guiada pelo usuário

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Fernando Vieira Paulovich

USP – São Carlos
Fevereiro de 2019

This work is dedicated to my beloved parents Meliton and Rita.

Acknowledgements

First, I would like to thank God for the provided strength during this Ph.D. period.

I would like to express my special thanks to my supervisor and mentor, Prof. Fernando Paulovich, for everything he has done for me over the past years as a master and doctoral student. He has introduced me to the journey into the field of research, I find myself extremely lucky to have done this journey under his supervision. I will never forget his phrase, "I let the students kill themselves." Thanks for not let me die :p.

Also, I offer my highest thanks to Prof. Evangelos Milios for giving me the opportunity to come and stay longer in Canada. For being a huge example of a supervisor. I wish you could visit my country someday.

A mis queridos padres, por darme su amor y apoyo incondicional. Fueron varios años lejos de casa, ni sé como sobreviví este tiempo sin ustedes. Pero el solo hecho de recordar sus sonrisas me alegraba. ¡Los amoo!

To my little brother Jhon. For being the best brother in the world and told me that I am his inspiration to be a better version of himself. However, sometimes, I doubt that. I really love you!

To my beloved boyfriend, for his support, love, and companionship. I am very happy to have you in my life, and I hope to enjoy your companionship for many years to come. I love you!

To Marco Antonio, my best friend since undergrad, thanks for believing in me and send me positive energies. Also, thanks for inspiring me to go abroad and discover a new world of opportunities. This accomplishment is for us!

To my friends that I met in Sao Carlos city: Rayner, Jorge, Edwin, Paulo, and Mayra; I really appreciate all the awesome moments that we share all together. Specially thanks to Rayner and Mayra for logistics in my absence. Also, I would like to thank my roommates: Tahcita and Alana for the great time at home.

To Yasmini for being my best friend when I arrive in Canada. I will never forget our pizza domino's and our movie meetings. Also, thanks to Silvia for our parties in the Pacific disco. We will be forever, the three superpower girls!

To Carlos Ureta, a Peruvian boy that I met in Canada. Thanks for introducing me to Lorena and Justin. Our free lunches were really fun and relaxing. This group grows

and I meet many more people. Thanks, Misaki, Shawn, Magda, Murwan and Emad.

To Magdalena Jankowska, for inviting me to her home at Christmas. I was a little sad because it was my first Christmas far from home; however, your companionship made me happy.

To my ELAP friends: Cecilia, Nicolas, and Alexandre, for our meetings and fun times. It was really fun to listen if Brazil or Argentina has the best barbecue.

I would like to thank all students from VICG Laboratory. Specially to Evinton, Filomen, and Paulo, for our daily meetings, conversations, and our lunch in the "bandeco". Also, I would like to extend my gratitude to Danilo, Tacito, and Renato for their advice in all this process.

I also thank CNPQ for financial support during my Ph.D. studies and the ELAP scholarship for financial support to stay in Canada.

Finally, I want to thank Manolo, my dear doggy. I will never forget our nice time. Each time before going back to Perú, I always thought of you. I imagine hugging you and touching your nose. It will be really sad to go back home and not to see you. I really miss you!.

ABSTRACT

HILASACA, G. M. **A visual approach for user-guided feature fusion**. 2019. 117 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

The goal of Dimensionality Reduction is to transform the data from highdimensional space into visual space preserving the existing relationships of the data in the original space. This abstract representation of complex data enables exploration of data similarities, but brings challenges about the analysis and interpretation for users on mismatching between the visual representation and the user expectation. A possible way to model these understandings is via different features to describe an object because each feature has its own way to encode characteristic. In this thesis, we propose a visual approach to support users to combine different features that best approach their point of view regarding similarity. Our approach is a two-step strategy, starting from a small sample of the features, where users can easily test different feature combinations and check in real-time the resulting similarity relationships. Once a combination that matches the user expectation is defined, it is propagated to the whole dataset through an affine transformation.

A traditional way to visualize data similarities is via scatter plots; however, it suffers from overlap issues. Overlapping hides data distributions, and it makes the relationship among data instances difficult to observe, which hampers data exploration. In this work, we present a technique called Distance-preserving Grid (DGrid) to tackle this issue. 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 is as precise as the existing state-of-the-art techniques based on grid representations, whereas requiring only a fraction of the running time and computational resources. Despite its simplicity, the quality of the produced layouts and the running times render DGrid as a very attractive method for large datasets.

Keywords: Feature Fusion, Grid Visualization, Distance Preserving Grids, Dimensionality Reduction, Exploratory Data Visualization, Visual Analytics.

RESUMO

HILASACA, G. M. **Uma abordagem visual para fusão de características guiada pelo usuário**. 2019. 117 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

O objetivo da redução de dimensionalidade é transformar os dados de um espaço de alta dimensionalidade para um espaço visual preservando as relações existentes entre os dados no espaço original. Esta representação abstrata dos dados complexos permite a exploração das relações similaridade, mas traz desafios sobre a análise e interpretação para os usuários devido à incompatibilidade entre a representação visual e a expectativa do usuário. Uma maneira possível de modelar os entendimentos dos usuários é através de diferentes características que descrevem um mesmo objeto, porque cada característica tem sua própria forma de codficar propriedades. Nesta tese, propomos uma abordagem visual para auxiliar os usuários para combinar diferentes características que melhor se aproxime ao ponto de vista do usuario em quanto o que é similaridade. Nossa abordagem é uma estratégia de duas etapas, onde começando com uma pequena amostra das features os usuários podem testar facilmente diferentes combinações de features e verificar em tempo real as relações de similaridade resultantes. Uma vez definida a combinação que corresponda à expectativa do usuário, ela é propagada para todo o conjunto de dados por meio de uma transformação afim.

Uma maneira tradicional de visualizar as relações de similaridade entre as instâncias de dados é através de *scatterplot*, no entanto, esse sofre de problemas de sobreposição. A sobreposição oculta a distribuição dos dados e dificulta na exploração das relações de similaridade. Neste trabalho, apresentamos uma técnica chamada *Distance-preserving Grid* (DGrid) para resolver esse problema. O DGrid emprega um processo de particionamento de espaço binário em combinação com a saída de uma redução de dimensionalidade para criar *layouts* de grade regulares. O DGrid garante que as instâncias não fiquem sobrepostas, devido a que cada instância de dados é atribuída apenas a uma célula da grade. Nossos resultados mostram que o DGrid é tão preciso quanto as técnicas existentes no estado da arte que estão baseadas na criação de grades, requerendo apenas uma fração do tempo de execução e poucos recursos computacionais. Apesar da sua simplicidade, a qualidade dos layouts produzidos e os tempos de execução tornam o DGrid um método muito atraente para grandes conjuntos de dados.

Palavras-chave: Fusão de Características, Visualização em Grades, Grades de Preservação de Distâncias, Redução de Dimensionalidade, Visualização Exploratória de Dados, Analítica Visual.

List of Figures

List of algorithms

List of Tables

List of abbreviations and acronyms

CC	Cross Correlation
DGrid	Distance-preserving Grid
DR	Dimensionality Reduction
IVA	Interactive visual analysis
k-NN	k-Nearest Neighbors
KS	Kernelized Sorting
LAMP	Local Affine Multidimensional Projection
LAP	Linear Assignment Problem
LDA	Linear Discriminant Analysis
LSP	Least Square Projection
MDS	Multidimensional Scaling
NP	Neighborhood Preservation
PCA	Principal Component Analysis
SSM	Self Sorting Map
SVM	Support Vector Machine
t-SNE	t-distributed stochastic neighbor embedding
VA	Visual Analytics

List of symbols

n,N — Number of instances

D — Set of instances in the original space q-dimensional

q — Dimension of the original space

d_i — i-th instance in the set D, whose vectorial representation is given by $\mathrm{d}_i = (d_{i1}, d_{i2}, ..., d_{iq})$

P — Set of instances in the projected space m-dimensional

m — Dimension of the projected space

p_i — i-th instance of the projected space, whose vectorial representation is given by $\mathrm{p}_i = (p_{i1}, p_{i2}, ..., p_{im})$

G — Grid with w rows and u columns

$\delta(d_i, d_j)$ — Dissimilarity between instances i and j in the original space

$\partial(g_i, g_j)$ — Distance between assigned cells for the instances i and j

Δ — Dissimilarity matrix

Λ — Parameter to control the grid aspect-ratio

k — Number of nearest neighbors of an instance

h — Number of feature extractors

F_i — Set of feature vectors obtained with i-th extractor

S_i — Sample obtained from the feature space F_i

R_i — Set of instances in the projected space m-dimensional of the sample S_i

V_i — Set of instances in the projected space m-dimensional of F_i

α — Weight values

E_{al} — Alignment error

E_{st} — Stress error

λ — Parameter to control the importance of the distance preservation and the alignment

γ — Learning rate

κ — Decay power

$\tilde{f}_i$ — Coordinates of the anchor representing feature F_i

$\tilde{d}$ — Coordinates of the dial

Contents

Chapter

1

Introduction

Nowadays, data have become a crucial component for intelligent decision making. For example, business analysts take more decisions with data, and they have witnessed a remarkable increase in the adoption of visualization as an important tool to guide the decision making process. Visualization provides an effective bridge for analysts to make sense of the data and sometimes unveil interesting insights (TELEA, 2014).

A way to more closely involve users in an analysis process using visualization is known as Visual Analytics (VA). VA provides a visual interface between automated techniques and users with the aim of effectively combine their strengths (KEIM *et al.*, 2010; SACHA *et al.*, 2014). This duality results in a closely communication between users and machines, where computational results are communicated by visualizations and user's feedback is expressed by interactions.

One of the most common techniques in VA is Dimensionality Reduction (DR) (JEONG *et al.*, 2009; CHOO *et al.*, 2010; SACHA *et al.*, 2017). 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 insight on the similarities. Hence, there is a disconnect between how users and machines perceive data similarity, and it negatively impacts the communication between users and machines, since machines do not have any clues about the user semantic similarities.

A possible way to model user semantics is via different features to describe an object. For example, an image can be described with different characteristic, such as color, texture, shape, and so on. For instance, an expert in plants may be interested in differentiating leaves. In this context, the most meaningful feature is texture and the less relevant feature is color because most leaves are green. Another example is in photo album organization, some people could be interested in organizing photos by their relatives and others by places (e.g. museums, natural places etc). In this setting, the most relevant

feature is object detection which can recognize faces of your relatives or objects, such as sculptures, trees, grass, etc. These objects are relevant to photo album categorization.

As a data scientist, we would like to find the best feature for an specific task. However, each feature provides complementary information and there is no perfect one. A possible solution for that is to integrate all features in a process known as feature fusion. A simple strategy for feature fusion is concatenate all features into one single feature vector (MANGAI *et al.*, 2010). Another strategy is weighing features automatically, which is commonly done in classification and image retrieval tasks. Typically, the learned weights aim to improve an objective function. For example, Ma *et al.* (2016) 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. Users define semantic weights that enhances feature fusion applicability. For example, to differentiate older pictures from newer ones, we can focus on color features. And, to classify different leaf types, we can use a texture extractor.

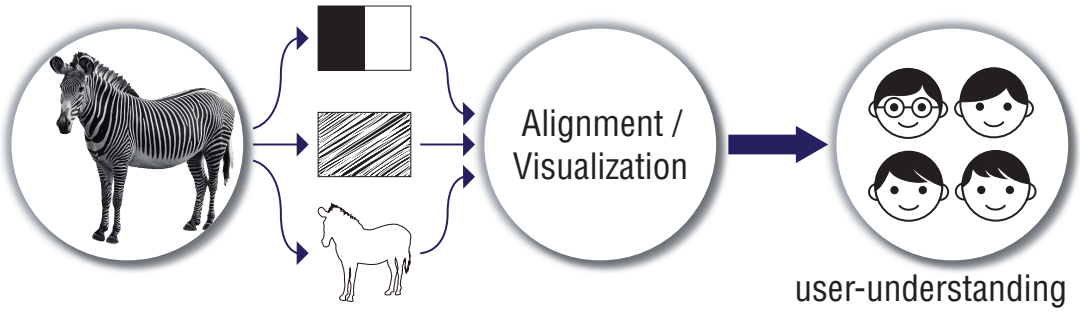


Figure 1 – Color, texture, and shape features are extracted from an image. Users find a suitable combination of these features that matches their understanding to recognize a zebra.

Since the similarity between data instances is on the eye of the viewer, it is desirable to incorporate users understanding and control the semantics 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. The idea of alignment is similar to that employed in image registration (BROWN, 1992; ZITOVA; FLUSSER, 2003). However, in contrast to image registration, where two or more pictures of the same scene are aligned using key point pixels, we align features from different extractors. This mapped sample is used to configure the weights for feature fusion process. Then the sample and the initial

weight configurations are the input for a local affine transformation. This transformation propagates the user-defined semantics of the similarity to the whole data ensuring that user-defined understanding is preserved. We illustrate this idea in Figure 1. First, features such as color, texture, and shape are extracted from animals domain. Each feature captures different properties from images. In order to combine these features properly, they are aligned. Then users define a weighted combination of color, texture and shape features using a visualization tool. For example, this process can be useful to recognize a zebra from the whole animals domain.

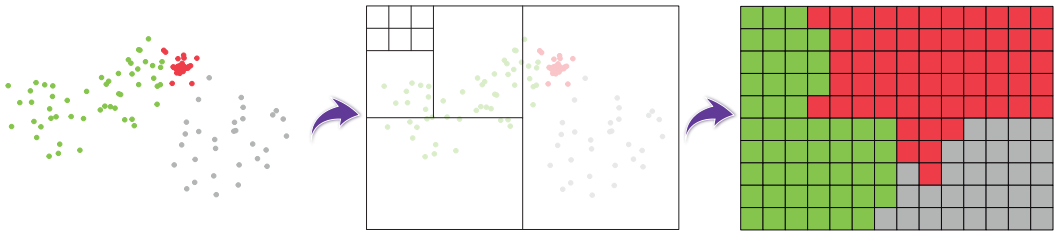


Figure 2 – DGrid visualization. DGrid receives a traditional 2D layout and convert it into a grid representation via space-partitioning.

Typically when analyzing similarity between instances, scatterplot is normally used. However, scatter plots 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 (GOMEZ-NIETO *et al.*, 2014; STROBELT *et al.*, 2012) or put constraints on the projection process (PINHO; OLIVEIRA; LOPES, 2010) to remove the overlapping. However, they make poor use of the visual space, creating layouts with void areas. Aiming at making 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 (FRIED *et al.*, 2015; QUADRIANTO *et al.*, 2010) or use permutations to optimize cost functions (STRONG; GONG, 2011; STRONG; GONG, 2014). 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 dimensionality reduction 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. This idea is illustrated in Figure 2. Despite its simplicity, the quality of the produced layouts and the running times render DGrid as a very attractive

method for large datasets.

1.1 Our contributions

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

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 contribution 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, which 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.

1.2 Organization

The remaining of this thesis is organized into five chapters. Chapter 2 covers relevant literature to contextualize the addressed problem. Chapter 3 and 4 present in detail the proposed methods. Chapter 5 present an application of our methods. Finally, chapter 6 presents the conclusion and future work.

In summary the subjects covered by each chapter are:

- **Chapter 2** reviews related work covering topics from visual analytics to user-guided visualizations. We start with visual data analysis, focusing on user interactions. Next, we discuss feature fusion strategies and visual metaphors for similarity analysis (e.g. grid-based visualizations), which are the main topics tackled in this thesis.
- **Chapter 3** focus on feature fusion. First, an initial sample of each type of feature is defined, and they are mapped to a common space. This mapped sample is used

to configure the initial feature fusion, which could be defined by users. Then this initial configuration will be propagated to the whole data set.

- **Chapter 4** presents a novel grid visualization technique, which competes favorably, or even exceeds state-of-the-art grid techniques, such as computational scalability and precision in preserving distance relationships.
- **Chapter 5** presents an application of our proposed works. We present an interactive mechanism that let users embed their semantic understanding of similarity in real-time. We also show that our visual method is scalable for large data sets allowing a level-guided navigation for exploration.
- **Chapter 6** summarizes our general conclusions, discusses limitations and describes some future works to further develop this research topic.

Chapter

2

Related Work and Background

This chapter discusses relevant background and work related to the topic of this thesis. It starts with an introduction to Visual Analytics (VA) focusing on users interaction and exploratory analysis (Section 2.1). Then, we review related topics for our project such as *feature fusion* (Section 2.2) and visual metaphors for similarity analysis (section2.3). Finally, in section 2.4, we discuss how our project differs from current approaches.

2.1 Interactive Visual Data Analysis

Interactive Visual Data Analysis, also known as Visual Analytics, aims to more-closely involve the user into automatic methods through interactive visualization techniques (KEIM *et al.*, 2010). Tam, Kothari and Chen (2017) show case studies where user interactions can produce better results than purely automatic machine learning methods. The VA process can be expressed through different pipelines, for example, by Keim *et al.* (2010) or Endert (2014). The visual analytics process proposed by Keim *et al.* (2010), which is widely adopted in the community, is depicted in Figure 3. The pipeline starts with raw data, then pre-processing steps are performed, such as data transformation. After the data pre-process procedure, both visual exploration and automatic analysis methods are available. Visual data exploration methods offer an interactive visual interface to display data, while automatic analysis methods require users to generate models by applying data mining methods. These models can be visualized for evaluation and refinement, visualization allows users to participate in the model generation and modification process by updating the model or refining parameters. In this process, knowledge can be acquired from visualization, automatic analysis, as well as the interactions between visualization, models, and users. However, interactions in this pipeline focus on allowing users to change the visual representation of model building through parameter refinement (WANG *et al.*, 2016).

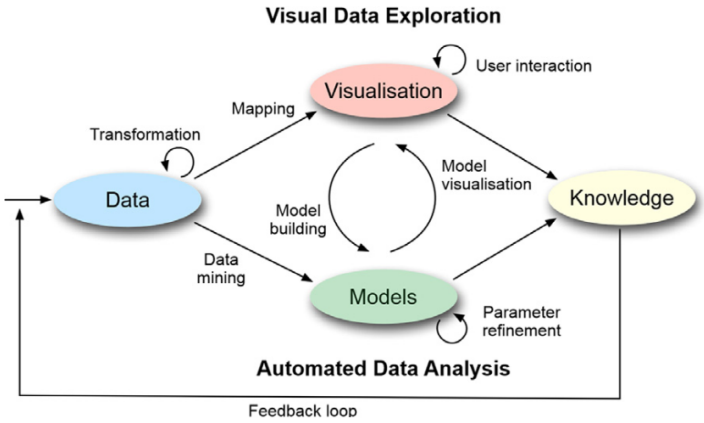


Figure 3 – Keim *et al.* (2010)'s pipeline models VA as a process that start with raw data. That is visualized and used by automatic methods. Then, users can explore and analyze the data and models visually.

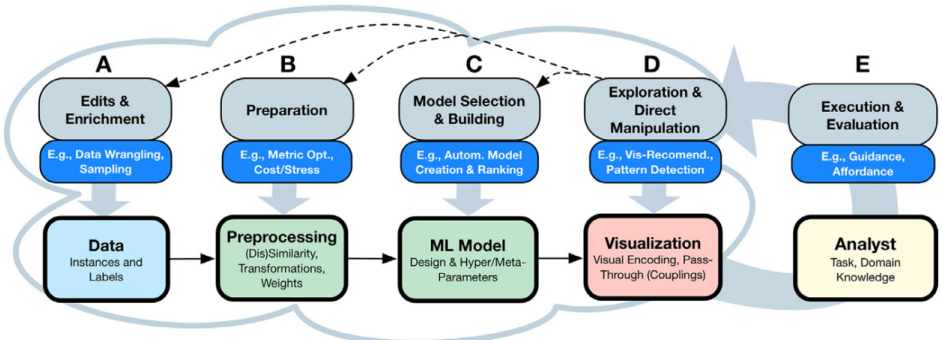


Figure 4 – Pipeline proposed by Sacha *et al.* (2017). The visual analytics process is shown on the left (A-D) and (E) shows the user evaluation. Visual interfaces (D) is a bridge between the automatic method and user interpretation. Light blue boxes denote interactions options and dark blue boxes denote automatic methods to support interaction.

Sacha *et al.* (2017) proposed a pipeline to describe other visual analytic interactions, such as direct manipulation and model building. This pipeline is illustrated in Figure 4. It combines conventional VA pipeline (Figure 4, A-D), complemented by interaction options (light blue boxes), with user's evaluation and refinement process (Figure 4 E). Using the visual interface (Figure 4 D), users can interact with each stage of the pipeline (dashed arrows), and the visualization can be updated and inspected by users (solid arrows). The dark blue boxes denote automated methods to support the user interactions.

Next, we detail the interactions involved in each stage of the analysis process,

- Edits and enrichment (A): This pipeline allows data cleaning, editing, and enrichment, but it incorporates the user in the process. For example, data split does not follow an automatic approach, such as cross-validation. Instead, users decide how to split the data. Similarly, automatic sampling techniques are useful for large data set for easy user interaction.

- Preparation (B): In this step, users can find some differences between their domain knowledge and distance measures. Hence, users transform the data with standardization, scaling, and feature selection/weighting to closely follow their interpretation.
- Model selection and building (C): Model selection interactions allow the user to choose among several ML methods or combine some of them via ensemble learning. Model building interactions let the user adjust the parameters (such as the number of nodes in a neural network) to change the quality and accuracy of the produced model.
- Exploration and direct manipulation (D): All pipeline stages can be visualized and allows navigation within views to help users to understand and interpret the visualization. It also allows manipulation of objects by users. Through this interaction, the system is able to learn about the user's reasoning and update the ML model. This behaviour is also known as semantic interaction (ENDERT; FIAUX; NORTH, 2012; DOWLING *et al.*, 2018).
- Execution and evaluation (E): From the previous stages, intuitive visualizations are acquired. Then, users perform an interactive and iterative analysis process to understand the data. Users formulate hypothesis to get insights from the data which are observed, interpreted, validated and refined by them.

Next we will present some examples of VA systems described in the literature that engage users through interactive visualization. We grouped these works in two categories: User interaction to improve machine learning methods and user interaction to improve similarity semantic.

2.1.1 User interaction to improve machine learning methods

EnsembleMatrix (TALBOT *et al.*, 2009) allows users to combine already computed classification models The system visualizes each classifier using a confusion matrix (bottom right side of the Figure 5). EnsembleMatrix provides two basic mechanisms for exploration: (1) a partitioning operation on the confusion matrix, which divides the class space into multiple partitions. It separates the data instances into subsets, allowing the user to develop specialized predictors for each subset; and (2) a linear combination , where users select weights for each classifier. Users can experiment and evaluate different linear combinations of individual classifiers by interactively adjusting their weights through a single two-dimensional interpolation widget (top right in figure 5). It also uses a confusion matrix visualization to assess the behavior of the resulting ensemble (left side of 5).

Schneider *et al.* (2017) presents an interactive tool for the exploration of the data space (classification outputs) in close integration with the model space. It allows the user

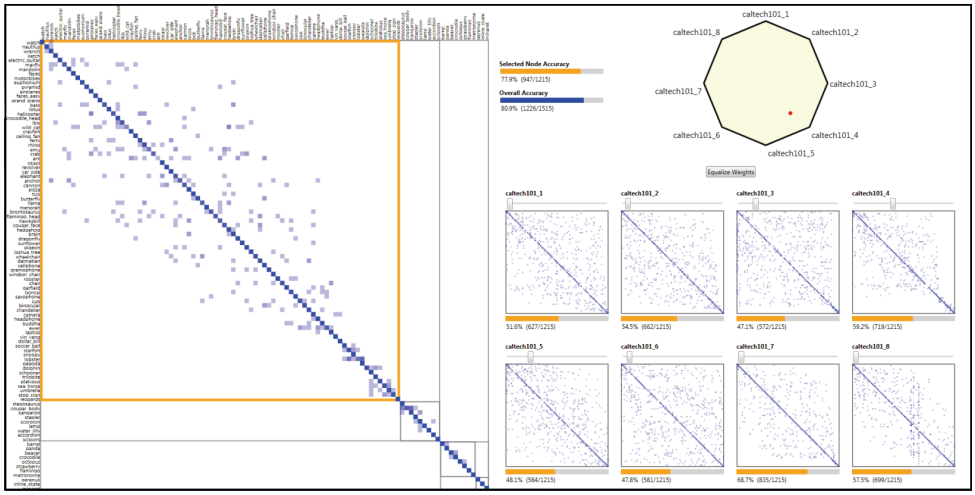


Figure 5 – **EnsembleMatrix** interface. Confusion matrices of individual classifiers are shown on bottom right. The matrix on the left shows the confusion matrix of the ensemble built by the user. Users can adjust the weights of individual models through a linear combination widget (top right). Users can also partition the confusion matrix to split and refine ensemble in a subset of categories.

to look for interesting regions in data space in order to improve the performance of the ensemble. First, several models are produced and next an automatic approach looks for the best combination of models that achieve higher performance. Then, the process for exploring ensembles of classifiers starts with the visualization of the classification outputs (the data space, see Figure 6(1)) and the initial ensemble (model space, see Figure 6(3)) that produce the final classification. In the data space, each point corresponds to one data instance of the test data set and the color indicates the predicted label. Analogously, in the model space, each point corresponds to a classifier and the color shows if the model is part of the current selected ensemble or not. In the data space, vertical axis represents attribute values while horizontal axis shows classification probabilities. It allows users to explore and select regions of interest to fix classification problems. The selection of a region of interest updates the model space to show how each individual model classifies the current data selection. The model space compares each single model by customizing the axes; axes show performance measures of the model. And, the visualization allows inclusion or removal of models from the ensemble. The interactions in the model space trigger an ensemble update with immediate impact on the data space.

2.1.2 User interaction to improve similarity semantic

Dis-Function (BROWN *et al.*, 2012) allows users to modify the distance matrix between instances, by moving points in the visual representation based on their understanding. The system first presents users with a scatter plot of the data projected via MDS with an initial distance function. It uses a weighted euclidean distance, i.e., euclidean distance with each dimension of the data weighted by a coefficient. It is defined

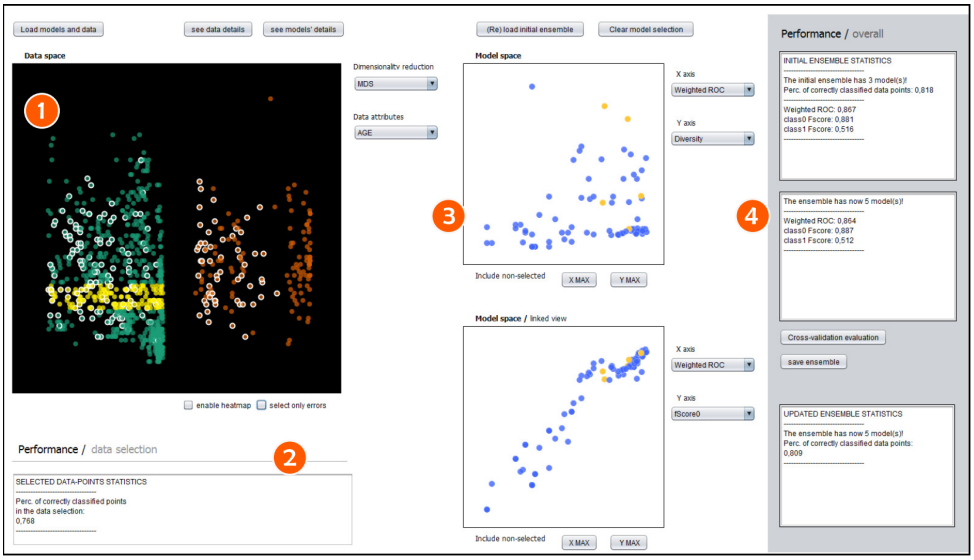


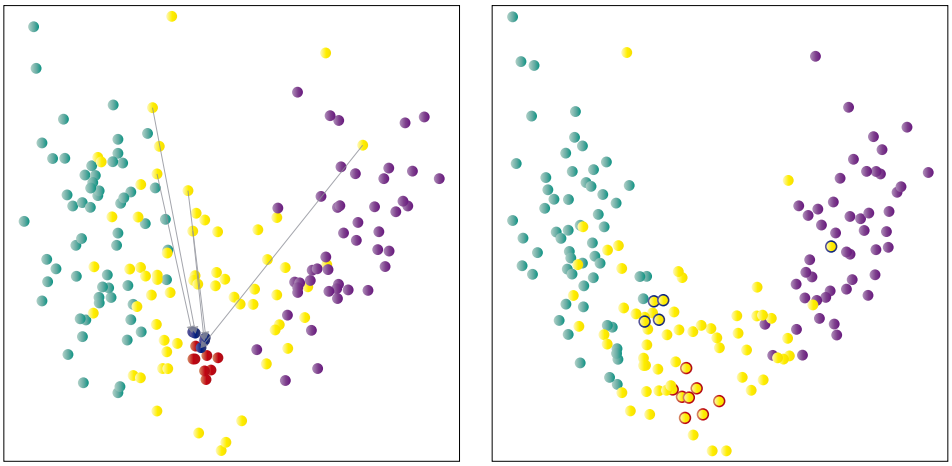
Figure 6 – Schneider *et al.* (2017) interface for exploring ensembles of classifiers. Scatter plot (1) shows the data space; where user can select regions of the classification outputs. (2) shows how the produced ensemble classify these regions. Scatter plot (3) shows the model space. It depicts a collection of models, where orange points are models of the current ensemble. It also allows updates to the ensemble by adding or removing individual models.

by

$$\delta_{\Theta}(d_i, d_j) = \sum_{K=1}^{q} \theta_k (d_{ik}, d_{jk})^2, \tag{2.1}$$

where $\Theta = (\theta_1, \theta_2, \ldots, \theta_q)$ is the vector of coefficients being learned, initialized to $\theta_q = \frac{1}{q}$. The systems expects the user to compose two sets of points that should be nearer to one another or further apart in accordance with users understanding. These two sets of points are marked by blue and red in the visualization (see Figure 7 (a)), which will have their distances updated. After selecting which points will compose both sets, users proceed to move the points. When users finished with manipulations, a distance function is learned and the system recomputes the projection with new metrics. An example of such changes are illustrated in Figure 7. Notice, it maintains the relative distances of points the user did not select while selected blue points are close in the desired direction. An important aspect is that the user is capable of seeing the changes iteratively caused by interactions. However, due to the system employs MDS, it is limited to a small data set.

iPCA (JEONG *et al.*, 2009) is an interactive tool that visualizes the results of PCA using linked views (i.e., scatter plot, parallel coordinates, or matrix views as shown in Figure 8(a)) and a rich set of user interactions. For example, users can re-position a point in the projection view, and see how eigen space values change. It also allows the user to change the weights of original data dimensions using multiple sliders (one slider per dimension). However, as the size and dimensionality of datasets increase, visualizing



(a) (b)

Figure 7 – An example of interaction in **Dis-function** (BROWN *et al.*, 2012). (a) The blue and red set are defined and blue points are dragged closer to red points. (b) The updated view with a new distance metric reflects the previous manipulation.

and refining the dimensions causes an scalability problem.

Mamani *et al.* (2013) offers the ability to manipulate a sample of the data and then view the transformation incurred on the whole dataset based on the current user manipulation. First, the system acquires and project a sample. Then, users re-positions the projected points to create groups of interest such as in Figure 8(b). As this manipulations change distance relationships between points, it is necessary to reflect these changes back on the original (q-dimensional) space. Force Scheme technique (TEJADA; MINGHIM; NONATO, 2003) was adapted to map back to the original space R^q. After samples are updated in R^q, the authors modified LAMP (JOIA *et al.*, 2011b) to map the dataset in R^q. This is helpful to update the remaining data and reflect the users manipulation.

Molchanov and Linsen (2014) allows users to adapt a projection matrix by re-positioning points on the projection view. It inverts the process of modifying the projection matrix in a star coordinates changing positions of the anchors by allowing the user to specify the desired configuration directly in the projection view (by rearranging control points). Once users perform a re-positioning action, the projection matrix is recalculated based on a least-square (LS) solution of a system of linear equations. The control points could be selected by users, however, if a classification of data is given the authors recommend using cluster medians for better cluster separation. Figure 8(c) depicts the initial and final projection after interaction. We observe a clear separation of groups in the final result.

These visual analytics systems all share a number of common characteristics in

(a) iPCA (b) Mamani et al. (c) Molchanov et al.

Figure 8 – Approaches that benefits from involving users into the visualization process. iPCA fosters interpretation of the PCA output by assisting the user with multiple coordinated views. Mamani *et al.* (2013) allow modification of neighborhood structures in a projection. Molchanov and Linsen (2014) adapt the projection matrix by allowing the user manipulation in the visual space.

their support of user interaction. Next, we compare them according to Sacha *et al.* (2017)'s interactions scheme.

2.1.3 Comparison

Table 1 show Sacha *et al.* (2017)'s interactions supported by the above-mentioned systems. Regarding the Edit & Enrichment interactions, iPCA allows the user to remove data instances from the visual representation.

The preparation interaction is performed by iPCA, Molchanov and Linsen (2014), Dis-function, and Mamani *et al.* (2013), however, these works perform it in different ways. iPCA offers slider controls directly coupled to dimension loadings. Molchanov and Linsen (2014) and Dis-function allow preparation from direct manipulation, through optimization and learned weighting. Finally, Mamani *et al.* (2013) performs an inverse transformation on features.

In regard to Model Selection & building interaction, EnsembleMatrix offers the ability to build ensembles from a limited and small number of classifiers. And Schneider *et al.* (2017) allows the user to select and build ensembles from several types of classifiers with different parameter settings.

Concerning Manipulation, iPCA, Mamani *et al.* (2013), Molchanov and Linsen (2014) and Dis-Function allow users to explicitly manipulate data values by moving points in the 2D projection. In response to user-driven repositioning, iPCA recomputes PCA; Molchanov and Linsen (2014) infers the dimension loadings; Dis-Function learns a compatible distance function; and Mamani *et al.* (2013) transform the data based on local affine mappings.

All systems offer visualizations for exploration and initial evaluation. For example, EnsembleMatrix uses confusion matrix visualizations. Schneider *et al.* (2017) use scatter plot, therein axis could be classification probabilities or classifier performance measures. The rest of the systems also use scatter plot, however, they show results of DR techniques. Each performed interaction results in an observable behavior within the visualization. Hence, users interpret, observe and validate the new updated results.

Table 1 – Comparison of studied grid-based techniques via interactions.

System	Edit & Enrichment	Preparation	Model Selection & building	Exploration & Manipulation	Execution & Evaluation
EnsembleMatrix			✓	✓	✓
Schneider *et al.* (2017)			✓	✓	✓
iPCA (JEONG *et al.*, 2009)	✓	✓		✓	✓
Mamani *et al.* (2013)		✓		✓	✓
Molchanov and Linsen (2014)		✓		✓	✓
Dis-Function		✓		✓	✓

2.2 Feature fusion

The process of integrating information from multiple sources to produce a unified enhanced data model is called data fusion (BOSTROM *et al.*, 2007). The intuition is to combine different data representations into a single model aiming at incorporating properties of the various sources.

The concept of merging features after feature extraction is called feature fusion. Feature fusion is also known as multi-view learning or data integration from multiple feature sets (ZHAO *et al.*, 2017). Many data are often extracted from different methods. For instance, color and texture features can be extracted through Lab color histogram and Gabor filter, respectively. These are two different kinds of features, which can be regarded as two-view data. Multi-view learning aims to find complementary and useful information from multi-view data, instead of single view data which can be restrictive.

We categorized three types of fusion strategies, namely, concatenation-guided fusion, distance guided fusion, and model-guided fusion. Concatenation-guided fusion concatenates all multiple views into one single view and it is used directly. Distance-guided fusion combines the distances calculated from the views. And finally, model-guided fusion combines model's output, which have been generated with multi-view data.

2.2.1 Concatenation-guided fusion

This is the traditional solution for multi-view learning. Giving the sets of features $F_1, F_2, \ldots, F_h$, the unified representation is defined by $[F_1, F_2, \ldots, F_h]$ (MANGAI *et al.*, 2010;

SUDHA; RAMAKRISHNA, 2017)., and then machine learning algorithms are applied directly on the concatenated vector. Although straightforward, it has been shown that feature concatenation is more effective than a single feature because additional features can provide complementary information (DOLLÁR *et al.*, 2012; ANNE; KUCHIBHOTLA; VANKAYALAPATI, 2015). Next, we will present some examples from the literature.

Wang, Han and Yan (2009) concatenated Histograms of Oriented Gradients (HOG) and Local Binary Pattern (LBP), showing a visible improvement over standard HOG in pedestrian detection. The approach can be seen in Figure 9 (a). Here, the classical sliding window technique is used to detect objects. A window of predefined size is slided over the input image. At each slide step, HOG and LBP features are concatenated. HOG features has two sub-procedures.: Once the features are concatenated, the SVM classifier is trained for detection.

Similarly, in (MANSHOR *et al.*, 2012), boundary-based shape features and local features are concatenated for improving performance of object class recognition. The feature fusion pipeline is illustrated in Figure 9(c). Specifically, the boundary-based shape features such as Fourier Descriptors (FD), Elliptical FD (EFD) and Moment Invariants (MI) (GONZALEZ; WOODS; EDDINS, 2004) are extracted from a segmented dataset. For local features, Scale Invariant Feature Transform (SIFT) (LOWE, 1999) was adapted to cooperate with shape features. Then, the new feature vector is trained by Support Vector Machine (SVM) to predict an unknown object class. The authors show that the performance of feature fusion improved the classification accuracy as compared to using single features.

Chu, Guo and Leng (2018) proposed an object detection algorithm for small and occluded objects that uses a multi-layer convolution feature fusion approach. Feature maps from low-level convolution contain more pixel information, which are helpful to detect small objects. In contrast, feature maps from high-level convolution contain more semantic information, which are useful to detect large objects. The authors follow these intuitions combining low and high level layers in a network. They uniformize the dimensions of low and high level layers via max-pooling and deconvolution, which are concatenated. This concatenation is feed to a region proposal network (RPN) to generate object bounding boxes for detection. This whole process is illustrated in Figure 9(b).

Feature concatenation was also used in the text domain. In (LONI; KHOSHNEVIS; WIGGERS, 2011), the authors extract seven types of lexical, syntactical and semantic features and combine subsets of them to improve text classification. The authors performed experiments in two different scenarios: either applying the Latent Semantic Analysis (LSA) reduction technique or not. In the first scenario, features are concatenated and reduced through LSA. Then, the reduced space is used to train and test the classifier. The second scenario is similar to the first, but lacks the feature reduction step, the classifier is only

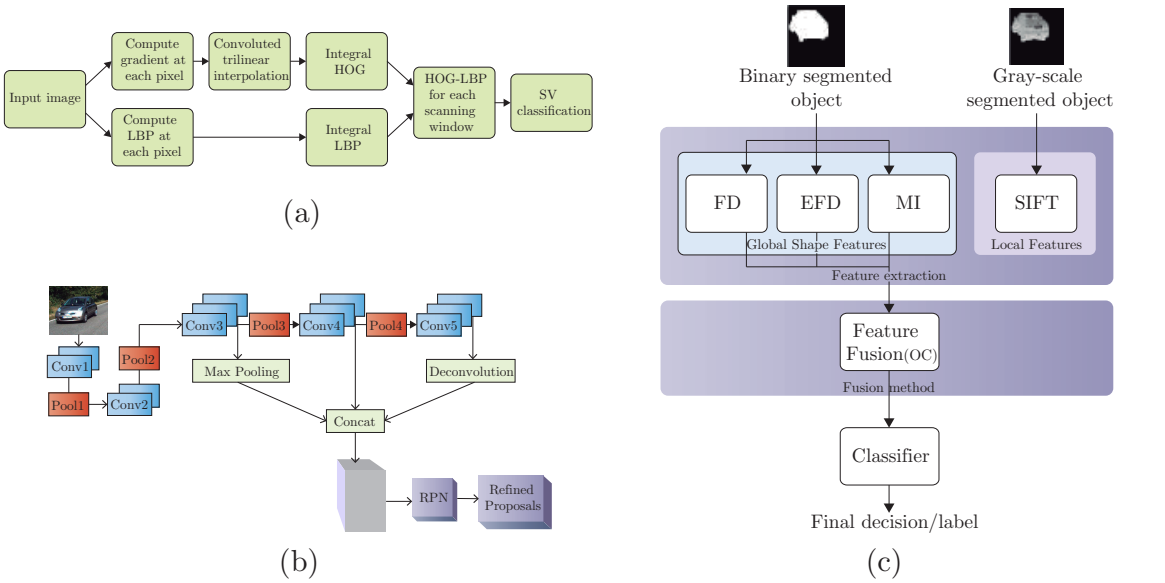


Figure 9 – Feature fusion Pipelines from Wang, Han and Yan (2009) (a), Chu, Guo and Leng (2018) (b), (MANSHOR *et al.*, 2012) (c)

trained and tested with the concatenated features. They compared the classification accuracy of different feature sets concatenation and found that the reduced space performs better than the original concatenated feature.

Weights can be used in the concatenation process to control the influence of the different features. In this process, the unified representations is given by $[\alpha_1 F_1, \alpha_2 F_2, \ldots, \alpha_p F_h]$ (MANGAI *et al.*, 2010), where $\alpha_1, \alpha_2, \ldots, \alpha_h$ are the weights. For example, in (LONI *et al.*, 2011), the weighted concatenation was used to improve text classification by combining lexical, syntactic, and semantic features. The weight assignments are found using a greedy approach.

In (MA *et al.*, 2016), a neural network learns weights to concatenate and combine different image features, such as color, shape, and texture as shown in Figure 10. The main purpose of this work is to preserve interpretability and retain meaningful information. This is accomplished using hand-crafted image descriptors for interpretability (See layer H_1), and an auto-encoder sub-networks to preserve the most relevant information among different layers in the network (See remaining layers H_2 - H_m). The author employed a robust auto-encoder that it is tolerant to noise, and can reconstruct the original input.

In (YOU; TANG, 2017), the authors use a saliency detection model to fuse color and texture features through a weighting strategy. They first transform the color and texture features in saliency features and then linearly combine the saliency features with adaptive weight according to the texture complexity of each image. Different from the previous weighted techniques, in this case, they linearly combine the features instead

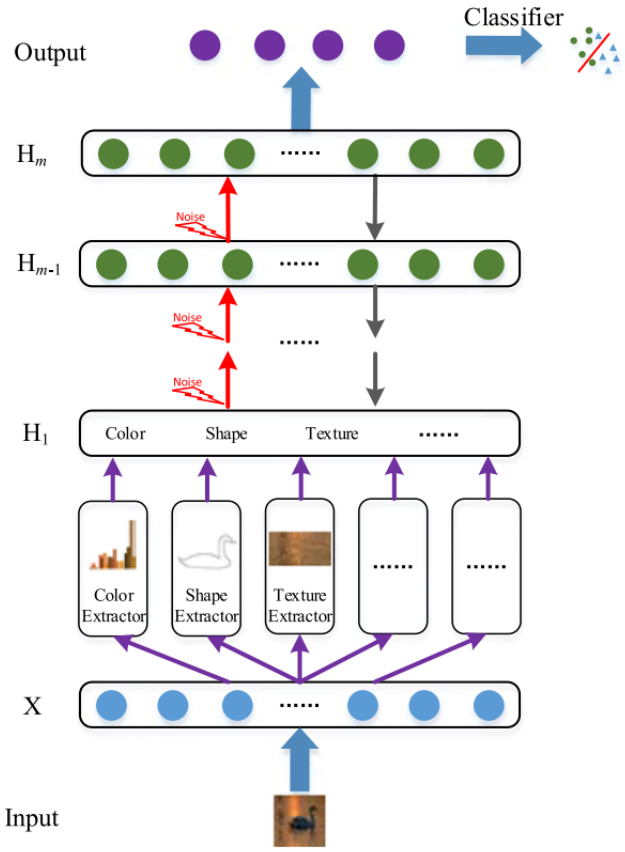


Figure 10 – Deep neural network that combines traditional feature extractors. It preserves interpretability via hand-crafted image extractors (See H_1) and preserve useful features via auto-encoders (See H_2 - H_m)

of concatenating them, that is, the unified representation is given by $[(1-\alpha)F_1 + \alpha F_2]$. This is possible since the saliency representations have the same dimensionality. In this approach α is adaptive weight given by Equation 2.2.

$$\alpha = \frac{1}{T} exp(\frac{V_{max}}{K} - 1) \tag{2.2}$$

where $V_{max} = V_1, V_2, ..., V_{NR}$, V_{max} represents the complexity of the image texture. The larger the value of V_{max}, the texture features have a greater contribution to the measure of saliency. Thus, a high value of the fusion coefficient is expected. T and K are constants.

In practice, the feature concatenation is not recommended since it may result in a huge feature vectors leading to the curse of dimensionality problem (MANGAI *et al.*, 2010). One solution is to apply a dimensionality reduction after the concatenation (YU; ZHU, 2017), or to perform a distance fusion.

2.2.2 Distance-guided fusion

In the distance fusion, instead of combining the vectorial representations, the distances calculated from the representations are combined. Distance fusion are typically used in image retrieval tasks. During retrieval, a similarity function is calculated between the query f_q and the features. If $\delta(f_q, F)$ represents the distance calculated from the query to feature F, a global similarity function is computed as a weighted sum of the similarities $\alpha_1\delta(f_q, F_1) + \alpha_2\delta(f_q, F_2) + \ldots + \alpha_p\delta(f_q, F_h)$.

In (VADIVEL; MAJUMDAR; SURAL, 2004), Manhattan distances calculated from color histogram and Wavelet based texture features are combined to support content-based image retrieval applications. The authors empirically define the weights and showed that the weighted fusion of color histogram and texture with weights has higher precision compared to only color histograms and fusion of both without weights. In Huang *et al.* (2010) also color and texture features are combined, but using HSV color moments and Gabor texture features. It improves the results of a content-based image retrieval system in terms of complexity and accuracy. Here, users assign the weights to each feature respectively and calculate the similarity with combined features of color and texture according to normalized euclidean distance. This approach allows fast and improved accuracy image retrieval due to their lower dimension.

Finally in (LIU *et al.*, 2017), the authors retrieve images using a proposed metric that combines low-level and high-level features. Low-level features are extracted by a Dot-difuse block truncation coding (DDBTC) and high-level features are calculated using an intermediate layer form Google net. The new proposed pipeline is depicted in Figure 11. The proposed metric retrieval contains a hierarchical approach with high-dimensional (HD), mid-dimensional (MD) and low-dimensional (LD) embedings. HD embedings are calculated from a histogram of DDBTC low-level features concatenated with previous high-level features. Then, MD features work similarly, but it is employed a compressed histogram to reduce dimensions. Finally, MD features are fed to an VLAD dimension reduction to generate LD features. Hence, for retrieve a query image, first database images are ranked using the LD embedings. Then, top nearest images are re-rank with HD embeding to refine the results. HD features rank images via a weighted distance of the mentioned low-level and high-level features. Notice, LD features are useful for fast ranking, and HD features are helpful for refine the retrieval results.

2.2.3 Model-guided fusion

Different from previous fusion strategies, model fusion combines computational models instead of data. Such combination can be performed in two different ways: by combining different models (parametrizations) processing a single feature set (data set), or by combining different models processing different feature sets (KIM *et al.*, 2016).

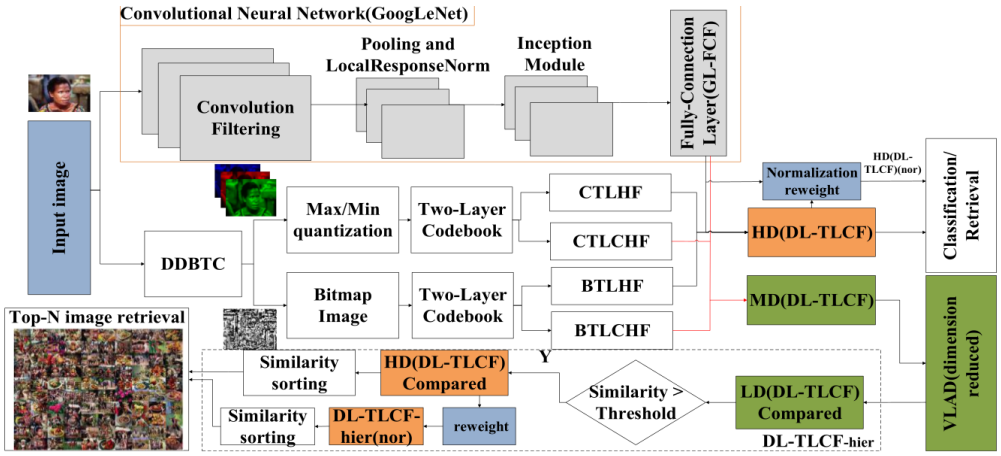


Figure 11 – Image retrieval setup that combines traditional low-level features (DDBTC) and high-level features (Google net) via high-dimension, mid-dimension an low-dimension features. LD features are useful for a fast ranking, and HD are helpful to refine the results.

The former is called ensemble learning and has been extensively used for classification tasks. The idea is to combine the prediction of different models using some voting strategy to improve model diversity and classification accuracy (MENDES-MOREIRA *et al.*, 2012; DIETTERICH, 2000). Ensembles of classifiers typically outperform single classifiers (SCHNEIDER *et al.*, 2017) and have been used in different domains, including remote sensing, computer security, financial risk assessment, fraud detection, recommender systems, medical computer-aided diagnosis, and others (WONIAK; nA; CORCHADO, 2014; KIM *et al.*, 2016). Similarly, the later also employs a (weighted) voting strategy to combine different models, but in this case, the models use as input different sets of features. Next, we present some approaches, which receive multiple features as inputs.

Kuang *et al.* (2016) employs a meta-learner to learn classifiers from individual features, and then combine their scores via a new classifier. First, the system learns a multi-class classifier for each feature using a SVM. Then, on top of the previous classifier scores, a new SVM is trained for each class.

In (KIM *et al.*, 2016), the authors used a hierarchical approach to join multiple feature extractors and multiple classifiers in local and global blocks. For local blocks, they combine two features extractors and two classifiers (See Figure 12, left). The weighted combination of classifiers from the local block is learned via reinforcement learning (See Figure 12, center). The state is represented by 4 weights, one weight for each combination of features and classifiers (2*2). Then, eight actions are considered: an increment and decrement on each weight. The amount of update is determined by a gradient-descent method. Finally, the reward function is defined if it is an improvement in accuracy. Once the local blocks are calculated, the final decision is achieved in the global block combining the local blocks decisions using a Bayesian network (See Figure 12, right). This Bayesian

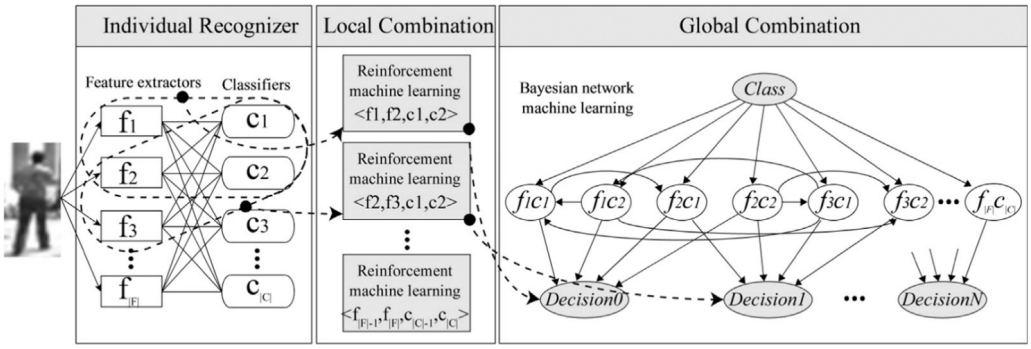


Figure 12 – Reinforcement learning and Bayesian network for model-guided fusion. First, each two feature extractors are coupled with two classifiers generating local blocks. Then, weights to combine classifiers are learnt via reinforcement learning. Finally, optimized local blocks are linked in a Bayesian network to exploit their relationships.

network is useful because it helps to model the dependencies among local blocks.

Common to all these fusion strategies is that the combination is defined manually without any support offered to the user. In applications where a loss function exists, like classification, such function can be used to aid in the weight definition. However, when such function is not available, or there is a degree of subjectivity in the process, the definition of weights without proper user support hampers its applicability in practice or real scenarios.

2.3 Visual Metaphors for Similarity Analysis

Different approaches have been proposed to create visual metaphors for conveying distance information. Among the existing strategies, the multidimensional projections have emerged as one of the fundamental tools for data analysis (JOIA *et al.*, 2011c).

Dimensionality reduction (DR) techniques, also called multidimensional projections, transform the data from q-dimensional space to a m-dimensional space (where $m < q$), preserving structures of the data (NONATO; AUPETIT, 2018). Results of DR are typically presented in a two-dimensional scatterplot where proximity between points indicates how similar they are (SEDLMAIR; MUNZNER; TORY, 2013). Thus, users can employ it to reason about the high-dimensional data patterns, see whether the input data consists of coherent groups of highly similar instances by looking for groups of densely-packed point. Given their nature of approximating distances, projection techniques tend to produce layouts with overlapped graphical elements, so suffering from occlusion problems. Figure 13 shows a projection on Iris dataset. The dataset contains 150 instances divided into 3 classes: Iris Setosa, Iris Versicolour and Iris Virginica, with 4 measures of sepal's width and length, and petal's width and length. (LICHMAN, 2013). In the projection color points indicate flower class. Notice, the red points are separated from the other

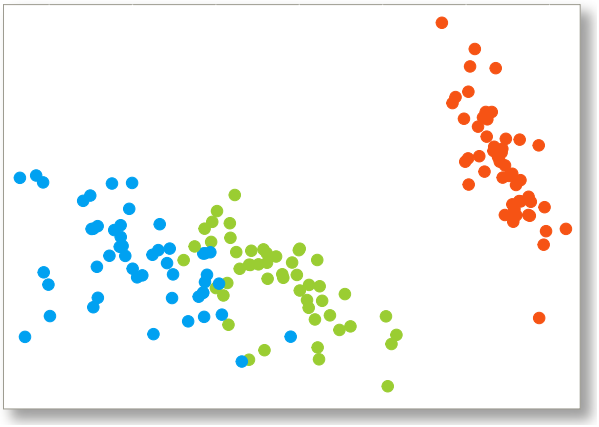


Figure 13 – 2D scatterplot of Iris dataset reduced with LAMP.

two group points, and there is some amount of overlap among them. In this case, the overlapping could be avoided by making the points smaller. However, when the graphical elements are used to convey information, such as images, data exploration and navigation tasks are still affected by overlapping. A way to address the overlapping issue is to arrange graphical elements into grids preserving distance relationship as much as possible. Also, there are other techniques that directly create grids, that is, these are not going through dimensionality reduction step. However, these techniques employed distance similaritires.

In the following sub-sections, we present interactive multidimensional projection and grid-based visualization techniques.

2.3.1 Interactive Multidimensional Projection

A multitude of DR techniques exist, ranging from classical approaches such as Principal Component Analysis (PCA) (Jolliffe, 2002), variants of Multidimensional Scaling (MDS) (FRANCE; CARROLL, 2011; BORG; GROENEN, 2005), t-distributed stochastic neighborhood embedding (t-SNE (MAATEN; HINTON, 2008a)), user-centered techniques (NONATO; SILVA; PAULOVICH, 2012) , etc. Each technique geared towards preserving a different aspect. These techniques can be divided into several categories, for example, Paulovich, Silva and Nonato (2010) categorized according to the their mathematical formulation, such as spectral decomposition, nonlinear optimization, and force based schemes. Silva and Tenenbaum (2003) and Joia *et al.* (2011b) organized the techniques into two groups, global and local. Maaten, Postma and Herik (2009) grouped the DR techniques into two criteria: convex and non convex. Nonato, Silva and Paulovich (2012) organized projections according to the capability of interactions. Recently, Nonato and Aupetit (2018) have categorized as to: Data types, linearity, flexibility for Supervision, capability for dealing with Multi-level structures, locality, steerability, stability, and capability of handling Out-of-Core (OOC) data.

Given that one topic in this thesis is user interaction, below we briefly present techniques that perform mappings according to user interaction. These technique require as first step a subsample of the data to perform an initial mapping; therein, user can incorporate their knowledge by manipulating such mapping.

Least Square Projection (LSP) (PAULOVICH *et al.*, 2008) employs an approach based on the solution of linear systems that aims to preserve neighborhood relations among q-dimensional instances in the visual space. First, instances of the dataset are selected as representative samples, called control points, next they are projected in the visual space. Positions of the control points, are used to define the geometry of the final projection, since neighboring points should be positioned in similar locations in order to preserve neighborhood structure of the data. Control point placement is done by the MDS method. Users can manipulate the points allowing users to directly interfere with the geometry of the mapping. k-neighbors for each instance of the dataset is defined. Making use of the neighborhood relationship of the instances in R^q and the cartesian coordinates of the control points in visual space a linear system is built whose solution are the coordinates of the remaining instances. LSP is very precise in preserving neighborhoods from the q-dimensional space to the visual space, however, the solution of the linear system becomes computationally expensive to large datasets.

Piecewise Laplacian Projection (PLP) (PAULOVICH *et al.*, 2011) addresses computational scalability for large datasets solving small linear systems, instead of just a large one. This technique has three steps as shown in the pipeline of Figure 14(a). First, samples are selected from a full dataset. The selection can be made using a clustering approach, or it can be provided by the user in order to drive the projection. Second, for each sample a neighborhood graph and control points are created. Each graph gives rise to a laplacian matrix that is used to project nodes from the graph to the visual space. Third, Laplacian linear system resolution is performed, here each instance of the dataset can be written as a convex combination of its nearest neighbors in the visual domain.

When users manipulate an instance in the visual space, they induce a change in the laplacian matrix associated with the neighborhood graph that includes the nearest control point to the manipulated instance. The update to the laplacian matrix then drives an update to the projection.

Part-Linear Multidimensional Projection (PLMP) (PAULOVICH; SILVA; NONATO, 2010) also can address the scalability problem by constructing a linear mapping of the control points. Then, this linear mapping is used to position the remaining instances, by a simple and fast matrix multiplication of the data instances matrix with the linear mapping matrix. PLMP steps are shown in the pipeline of Figure 14(b). First, samples are

selected and they are mapped through a non-linear Force Scheme technique (TEJADA; MINGHIM; NONATO, 2003). Then, a linear transformation $\Phi : R^q \rightarrow R^m$ is computed. That minimizes the difference between the projected distances d_{ij} and the original dissimilarities δ_{ij}. Therefore, Φ should satisfy Equation 2.3.

$$\Phi = \arg \min_{\Phi \in \ell_{p,q}} \left\{ \frac{1}{\sum_{ij} \delta(d_i,d_j)^2} \sum_{ij} \left(\delta(d_i,d_j) - \delta(\hat{\Phi}(d_i) - \hat{\Phi}(d_j)) \right)^2 \right\} \tag{2.3}$$

where $\ell_{p,q}$ is the space of linear transformations from R^q to R^m. And $\hat{\Phi}$ is the transformation applied in that space. Minimizing Equation 2.3 is prohibitive for large values of n. Consequently, an approximation of $\hat{\Phi}$ is proposed by using a priori information obtained from the control points. Assuming, that number of the control points is larger than q, Φ can be solved through m linear systems in the form

$$D'^T D' \phi_j = D'^T P' \tag{2.4}$$

for all $1 \leq j \leq m$, where D' is the matrix of control points and P' are the coordinates in the visual space. Each ϕ_j denotes an approximation for a column of Φ.

Once the approximation for Φ is obtained, the remaining instances are mapped by multiplication with the approximation of Φ. This operation become PLMP, one of the fastest techniques. However, it follows a global transformation that considers partially users interactions.

Local Affine Multidimensional Projection (LAMP) (JOIA *et al.*, 2011a) aims to allow more user control over the final mapping. LAMP also works by defining control points, which are used to build a family of orthogonal affine mappings. Figure 14(c) depicts LAMP process. LAMP maps each instance d to the visual space by finding the best affine transformation $f_x(p) = pM + t$ that minimizes Equation 2.5

$$\sum_i \alpha_i \| f_x(d_i) - p_i \|^2, \; s.t. \; M^T M = I \tag{2.5}$$

where matrix M and vector t are the unknowns. $\alpha_i = \frac{1}{\|d_i - d\|^2}$ are scalar weights, these depend on the point of evaluation. Therefore, a distinct affine transformation is obtained for each instance d. The solution to the minimization problem in Equation 2.5, can be rewritten in matrix form, giving rise to Orthogonal Procrustes Problem, whose solution is known. LAMP requires few control points to define good final mapping. Therefore, it is suitable for interactive applications.

2.3.2 Comparison

The previous multidimensional projection techniques work in a similar manner. First, control points are selected from the dataset and mapped to the visual space. Next,

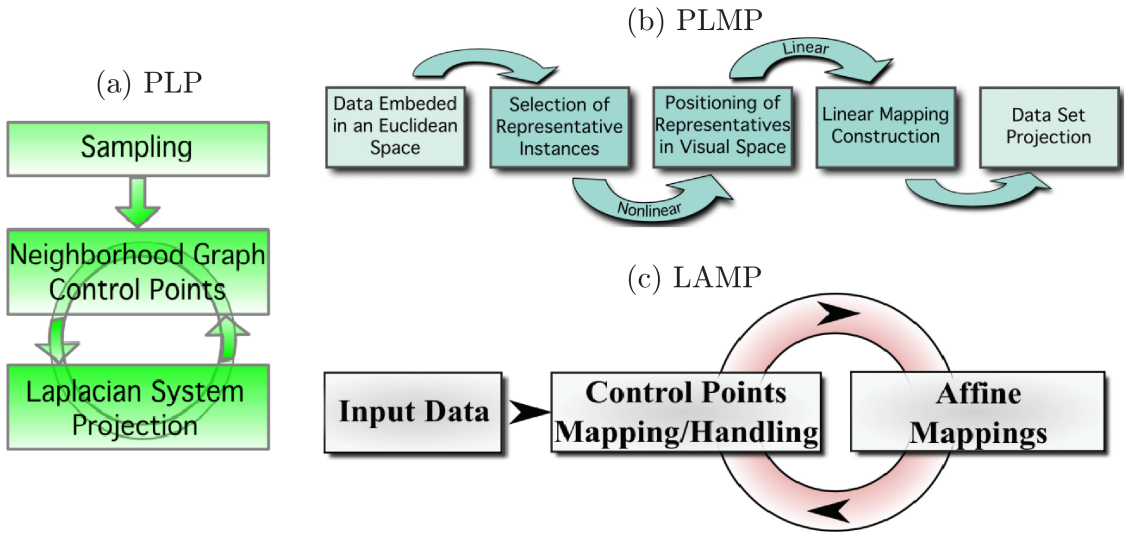


Figure 14 – Pipelines from PLP (PAULOVICH *et al.*, 2011)(a), PLMP (PAULOVICH; SILVA; NONATO, 2010)(b), LAMP (JOIA *et al.*, 2011a)(c)

the remaining instances are interpolated according to the geometry defined in the initial mapping. Thus, users can drive the final mapping by manipulating the control points in the visual space. Due to a large number of control points overwhelm the interactive process, it is desirable to keep the number of control points small. LSP, PLP, and PLMP have restrictions regarding the number of dimensions against the number of points, these demands a number that should be at least equal to the data dimension. On the other hand, LAMP needs a reduced number of control points to steer a good mapping (NONATO; SILVA; PAULOVICH, 2012).

Regarding the final mapping, while LSP and PLP rely on Laplacian linear system solvers to perform the mapping to the visual space, PLMP build a linear mapping and LAMP build a orthogonal affine mappings.

Computational performance is an issue for LSP. PLP, PLMP and LAMP address computational scalability for large datasets.

2.3.3 Grid-based visualization

As we mentioned in the introduction of this section, multidimensional projection results are tipycally shown in scatter plots. However, they produces overlapping between elements. To address such limitation, some approaches employ post-processing strategies, such as RWordle (STROBELT *et al.*, 2012), IncBoard (PINHO; OLIVEIRA; LOPES, 2010), ProjSnippet (GOMEZ-NIETO *et al.*, 2014), and the UnTangle Maps (CAO; LIN; GOTZ, 2016). It is beyond the scope of this work to survey all possible techniques for creating distance layouts. Here we focus on strategies that arrange data into orthogonal regular grids preserving similarity relationships.

One technique that can generate grids is the Self-Organizing Map (SOM) (KOHONEN, 1988). SOM is an unsupervised neural network that creates a discretized lower dimension representation of the data by arranging it into a two-dimensional regular spacing hexagonal or rectangular grid. The main drawback is that it can map several data instances to a single grid cell, opening spaces and overlapping instances on the composed grid (QUADRIANTO *et al.*, 2010; FRIED *et al.*, 2015). The same occurs to its probabilistic counterpart, the Generative Topographic Mapping (GTM) (BISHOP; SVENSéN; WILLIAMS, 1998). Spectral Hashing (SH) (WEISS; TORRALBA; FERGUS, 2009) can also be used (or adapted) to create distance preserving grids. SH creates hashing codes so that the Hamming distance between two codes approach the Euclidean distance between the instances they represent. Thereby, by splitting the code into bins corresponding to the rows and columns of a grid, SH codes can be used to assign data instances to grid cells preserving distances. Though a promising strategy, it suffers from the inherent problem of hashing techniques, collisions. Consequently, the produced grids also present opening spaces and overlapping instances.

In contrast to the overlapping problems, Self-Sorting Map (SSM) (STRONG; GONG, 2014; STRONG; GONG, 2011), Kernelized Sorting (KS) (QUADRIANTO *et al.*, 2010) and IsoMatch (FRIED *et al.*, 2015) assign data instance to grid cells, ensuring a non-overlapping so that each instance is mapped to a single cell.

Self-Sorting Map (SSM) is a multi-dimensional pseudo-sorting algorithm. It begins with a random assignment of the data instances to grid cells. Then, SSM technique uses a permutation process, swapping instances between grid cells, aiming at maximizing a cross-correlation function between the distances among data instances and distances among the cells' positions. The optimization cross-correlation function is given by,

$$CC = \sum_{i}^{N}\sum_{j}^{N} \frac{(\partial(g_i,g_j)-\overline{\partial})(\delta(d_i,d_j)-\overline{\delta})}{\sigma_\partial \sigma_\delta} \tag{2.6}$$

where $\delta(d_i,d_j)$ is the dissimilarity between pairs of instances, $\partial(g_i,g_j)$ is the distance between assigned cells, $\overline{\delta}$ is the mean distance between any two pairs of instances, $\overline{\partial}$ is the mean distance between any two cells, and σ_δ and σ_∂ are the corresponding standard deviations.

Since the total number of possible permutations is equal to the factorial of the dataset size, the SSM technique searches for a locally optimal organization. In this process, the grid is split into 4 blocks and each one is further split into four smaller blocks in the next stage, so on until the block contains only one instance. In each split, the blocks are grouped alternating between even-odd and odd-even settings. The former stands even-indexed blocks with odd-indexed blocks along both X and Y direction (see left side of Figure 15a), while in the latter odd-indexed blocks are grouped with even-indexed blocks

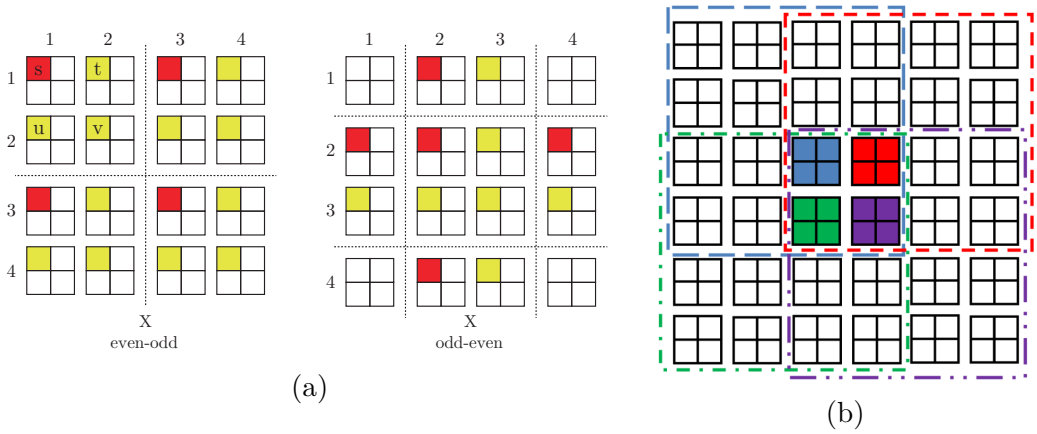


Figure 15 – Process into SSM. (a) Splitting a grid into blocks and grouping blocks using even-odd and odd-even settings. In both subfigures, the red cell is matched to the three yellow cells from the grouped blocks to perform permutations. (b) The neighborhoods defined for the four color-coded paired blocks in the center. Each neighborhood encompasses a block window that is offset away from the other blocks in the same group.

along both directions (right side of Figure Figure 15a). Once the blocks are grouped, a neighborhood $\Omega(B_i)$ for each block B_i is determined. The neighborhood is defined using windows that are centered from grouped blocks. For example in Figure 15b, we observe blue, red, green and purple neighborhoods. Once the neighborhood is determined, the target for each block is computed using Equation Equation 2.7.

$$T_i = \frac{1}{|\Omega(B_i)|} \sum_{B_j \in \Omega(B_i)} \frac{\sum_{s \in B_j} s}{|B_j)|} \tag{2.7}$$

Where $\Omega(B_i)$ is the neighborhood defined for block B_i and s is any instance from the cell B_j. Once the targets are calculated, the next step is to swap data between grouped blocks. In essence, data instances in the corresponding cells of the grouped blocks form quadruples, which are swapped. For example, the data instance in the red cell "s" shown in Figure. 15a is grouped with the ones in the three yellow cells "t, u, v" to form a quadruple under each block. To place a quadruple (s,t,u,v) into four cells, there are 24 permutations and the best one is chosen minimizing Equation 2.8.

$$\underset{(s,t,u,v)}{\operatorname{argmin}} \begin{pmatrix} \delta(s,T_{i,j}) + \delta(t,T_{i+1,j}) + \\ \delta(u,T_{i,j+1}) + \delta(v,T_{i+1,j+1}) \end{pmatrix} \tag{2.8}$$

Given the binary nature of this process, strategies need to be used to support non-square power-of-two grids (STRONG; GONG, 2014). Also, if the number of cells exceeds the number of data instances, it is not possible to control the position of the empty cells. They are (randomly) spread over the grid.

Kernelized Sorting (KS) (QUADRIANTO *et al.*, 2010) creates distance preserving grids finding correspondences between data instances of D and grid positions of G by permuting G. First, KS constructs kernel matrices K and L (on D and G, respectively). Then, KS maximizes the dependency of these matrices by representing a matching in terms of a permuting matrix $\pi \in \{0,1\}^{N \times N}$. Where $\pi_{ij} = 1$, indicates a match between d_i and g_j. The method selects the best permutation with the following equation:

$$\begin{aligned} \pi^* = \arg\max_{\pi \in \Pi_N} \ & \mathrm{tr}(\bar{K}\pi^T \bar{L}\pi) \\ s.t.\ & \pi 1_N = 1_N \ \& \ \pi^T 1_N = 1_N, \end{aligned} \tag{2.9}$$

where $\bar{K}$ and $\bar{L}$ are centered matrices of K and L respectively, Π_N is all permutation matrices with size $N \times N$ and 1_N is a column vector of ones of size N. Both constraints from Equation 2.9 indicate that an instance from D is matched to only one instance in G. Intuitively, applying π interchanges the columns of matrix $\bar{L}$ and π^T interchanges the columns from $\bar{K}$ so that g_i is matched to d_i. For example, applying $\pi_i = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ on $\bar{L}$ interchanges the first and second column (see step 1 of Figure 16). Similarly, applying π^T on $\bar{K}$ interchanges the rows. The optimization from Equation 2.9 is iteratively solved, i.e given a permutation matrix π_i, Equation 2.9 solve a permutation matrix π_{i+1}. This optimization problem is converted into a Linear Assignment Problem (LAP) and solved for optimal π_{i+1}.

Figure 16 depicts the entire process. In the beginning there are no matching between G and D instances. Then, an initial permutation π_i is picked. In each iteration, the columns of $\bar{L}$ are permuted according to the current permutation π_i. Then it finds the best current match between instances according to π_i. This is done by building a weighted bipartite graph between D and G, with weights equal to $\langle D_i, G_j \rangle$ (see step 2, Figure 16). Then, in step 3 a Linear Assignment Problem (LAP) finds an optimal assignment of instances in G to instances in D which becomes the new permutation π_{i+1} for step 4.

IsoMatch (FRIED *et al.*, 2015) is another technique that also employs an assignment strategy for constructing distance preserving grids. First, it projects the data into a 2D plane using the ISOMAP technique (TENENBAUM; SILVA; LANGFORD, 2000) then a grid of cells is generated around the center of the data (see step 1, Figure 17). Next, it builds a weighted bipartite graph between the projection and grid cell positions, with weights equal to pairwise distances between the projected instances and the grid positions (see step 2, Figure 17). Then, using the Hungarian algorithm (KUHN, 1955), it calculates a bipartite matching of this graph, assigning a projected instance to a grid cell. This assignment procedure aims to minimize the displacement of the projection positions into the grid positions (see step 3). The assignment is evaluated with an energy function,

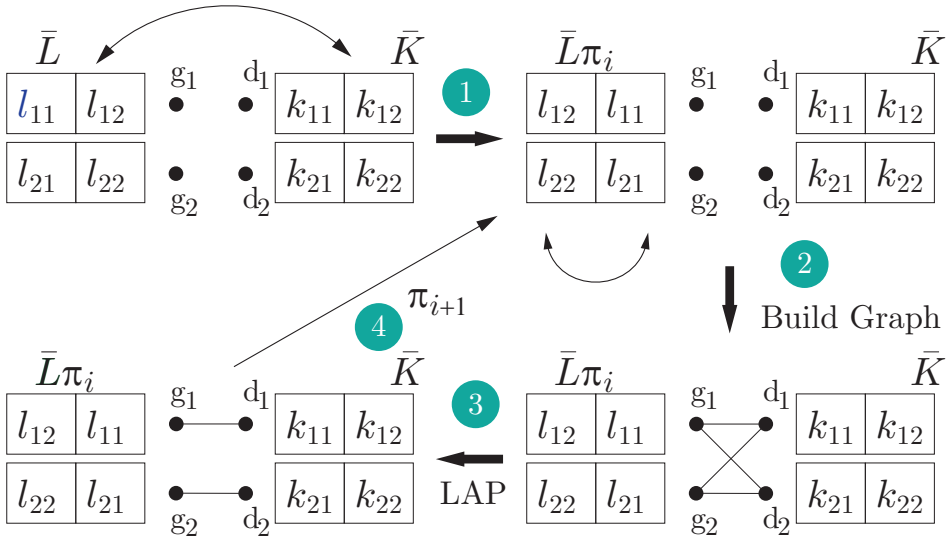


Figure 16 – The process of kernelized sorting to match observations of G and D. Kernel matrices K and L are created. A permutation matrix ϕ_i is applied on L (step 1), interchanging the columns of L. Then, a bipartite graph is built between D and G instances (step 2). A linear assignment problem (LAP) is used to find assignments (step 3) thus generating a new permutation matrix (step 4).

that measures how well the pairwise distances between the data instances are preserved by the corresponding distances in the grid, an "ideal" assignment yields energy equal to 0. This energy functional is computed as:

$$E_p = \left(\sum_i^N \sum_j^N \frac{|c \cdot \delta(d_i, d_j) - \partial(g_i, g_j)|^p}{\sum_i^N \sum_j^N \partial(g_i, g_j)^p} \right)^{\frac{1}{p}} \tag{2.10}$$

where $\delta(.,.)$, $\partial(.,.)$ are the distance measures between data instances and grid points respectively, c is a scale factor between the two metric spaces and p defines the employed norm. Once the arrangement is constructed it is possible to improve the results a small amount by randomly swapping pairs (where swaps are only kept if they improve the energy function). Notice that the more swaps it performs, the better the energy function is but the execution time will be more expensive.

2.3.4 Comparison

The three presented techniques arrange the instances while preserving their pairwise distances as much as possible in the grid locations. Table 2 summaries some characteristic of the grid techniques previously presented. While the input to IsoMatch is data projected into 2D, KS and SSM receive data in the original m-dimensional space. Regarding the solution employed for arrangement, both KS and Isomatch are based on LAP, and SSM uses a hierarchical swapping process. SSM approach only works on a regular grid. KS and IsoMatch are not limited to regular grids. They can create grids with

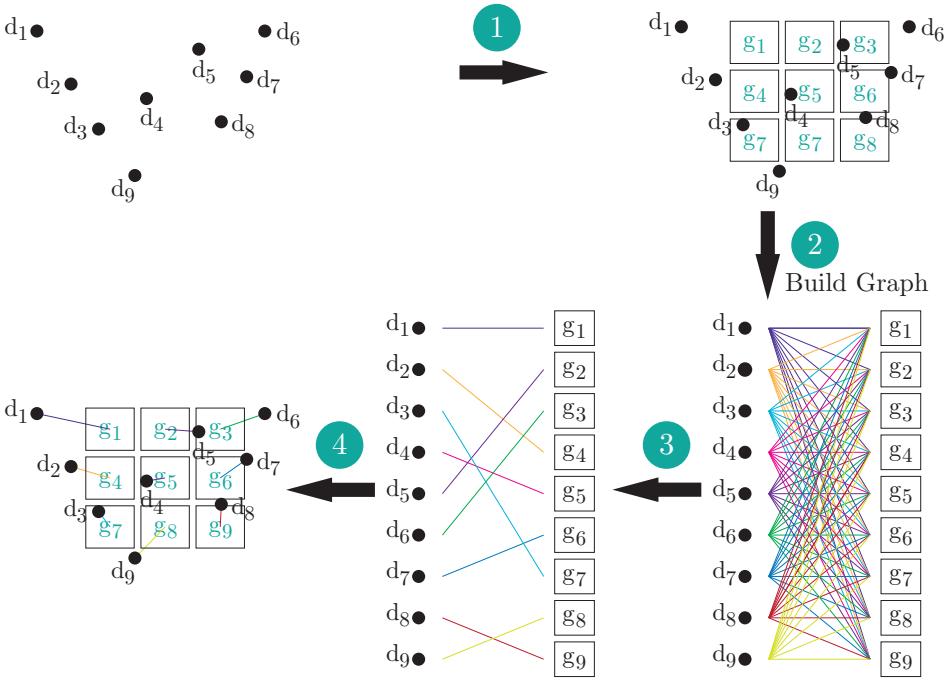


Figure 17 – The process of Isomatch to assign the data instances into the grid. First, data is projected onto 2D. Then, grid cells are created around the projected data (step 1). Next, a bipartite graph between 2D positions and grid cell positions is built (step 2). Finally, LAP is applied to find a minimal bipartite matching on the graph (step 3)

arbitrary shapes. However, since they solve assignment problems, they are computationally expensive $O(N^3)$, not being able to handle large datasets or even small dataset in real-time.

If the number of grid cells exceeds the number of data instances, KS and IsoMatch group the empty cells in one corner of the grid. SSM can not control the position of the empty cells, so they are (randomly) spread over the grid.

Table 2 – Characteristics of grid-based techniques studied.

Technique	Solution	Complexity	Input space	Output
Isomatch	LAP	$O(n^3)$	$\mathscr{P} \in \mathbb{R}^2$	grids with arbitrary shapes
Kernelized sorting	LAP	$O(n^3)$	$\mathscr{D} \in \mathbb{R}^m$	grids with arbitrary shapes
SSM	swapping	$O(n^2)$	$\mathscr{D} \in \mathbb{R}^m$	regular grid

2.4 Summary

Interaction enables users to explore data analysis. All works presented in Section 2.1 performs an interaction via direct manipulation within the visualization, or indirectly using control panels that show necessary parameters. In the work in this thesis, we also

consider user-defined interactions among multiple features. This setup differs from previous works in they only use one feature, which is restrictive. We take advantage of multiple features, their dependency and complementary knowledge.

Given multiple features, it is hard to identify which features are relevant. Automatic methods find associated weights to features to determine their relevance using a loss function. However, when such function is not available, or there is a degree of subjectivity in the process, the definition of weights without proper user support hampers its applicability in practice or real scenarios. In this thesis, we develop an approach that provides a closer interaction human-computer in order to generate easy understandable feature extractors combination. We provide a framework that allow users to explore individual features and combine them according to user-defined semantics. In this thesis, we provide user-guided feature combination on traditional scatter plots and grid-based visualizations.

Similarly to traditional scatter plots, we adapt our user-guided approach for grid-based visualizations. Our main motivation is that scatter plot can not represent properly overlapped data points. In contrast, grid-based visualizations represent each instance in a different cell position avoiding overlap issues. Also, we develop an efficient and scalable grid-based approach for big data sets. Current state-of-the-art approaches can not provide a real-time processing times as our proposed method.

Chapter

3

User-guided Feature Fusion

Multi-view data are common in real world applications. They are usually collected from diverse feature extractors, each providing complementary information. For example, in case of images, Histograms of Gradients (HOG) performs poorly when the background is cluttered with noisy edges, but Local Binary Patterns (LBP) can filter out noisy edges and avoid this issue (WANG; HAN; YAN, 2009). Hence, leveraging them properly can substantially improve the performance compared with using only a single type of feature. The integration of multiple feature sets is known as feature fusion or Multi-view learning (ZHAO *et al.*, 2017). A naive solution for feature fusion considers concatenating all multiple features into one single feature vector or combining linearly a set of similarity metrics from individual features (WANG *et al.*, 2017). Another commonly used approach is weighting features automatically in order to improve an objective function associated with an evaluation metric (e.g. accuracy in classification tasks).

Weighting features automatically have been successful. For example, Ma *et al.* (2016) optimize an error function to decide which feature to combine through a deep learning model. Liu *et al.* (2017) also learn weights using deep learning for image retrieval. However, when such function is not available, or there is a degree of subjectivity in the process, the definition of weights without proper user support hampers its applicability in practice. In that case, a feature fusion guided by the user's perception could be necessary. Some examples of subjective setting are music collections or photo albums, where user tastes differs greatly and different ways to capture users semantics and tastes are required. Teenagers and adult people have different music tastes due to age differences. Similarly, on photo albums, some people could prefer natural landscapes instead of hand-made buildings (such as museums).

In this chapter, we present the design of an efficient and effective method to generate an interpretable feature fusion defined by a user in real time. First, an initial sample of each feature is defined, and then they are mapped to a common space preserving the

distance relationships of the individual feature. In this common space, we perform an alignment of all features to ensure consistency among views through a gradient descent approach. This sample is used to configure the initial feature fusion process. Then, this configuration is the input for a local affine transformation, which propagates the user semantic understanding to the whole data. Hence, we ensure that its user-defined knowledge and interpretability is preserved.

To the best of our knowledge, as stated in the introduction, this is the first time that visual representations are exploited as a mechanism for feature fusion. This procedure correlates instances from feature spaces according to user's knowledge, characterizing another innovative aspect of this work.

This chapter is structured as follow. Section 3.1 describes our proposed feature fusion, including our approach details. In Section 3.2, we evaluate our approach using features extracted from 5 datasets. We also show that our method improves upon the standard method for feature fusion. Section 3.3 concludes the chapter.

3.1 Proposed Methodology

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. On the first phase, samples $S_1, S_2, \ldots, S_h$ are extract 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 feature. 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 visualized.

The user can then change the features 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 sets 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 users expectations defined by the sample visualization. Figure 18 outlines our approach showing the involved steps. Next we detail these steps, starting with the sampling and the dimensionality reduction.

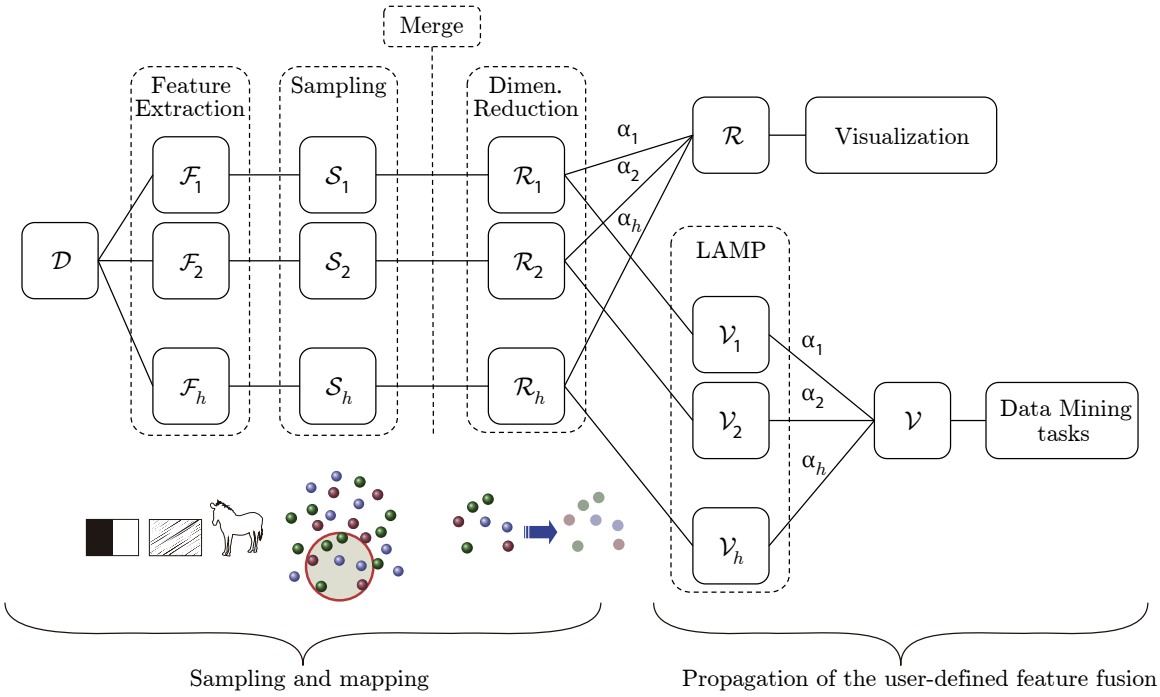


Figure 18 – Overview of our process for feature fusion. Initially a sample is extracted, combined and visualized. Based on that, the user can test different weights to fuse the features and observe the outcome. Once sample combination reflects the user expectation, the same weights are used to combine the complete sets of features that can them be used on subsequent tasks, such as clustering.

3.1.1 Sampling and Mapping

The first step of our process is sampling. Since users employ the sample visualization to guide the feature fusion process, it is important to have all possible data structures of 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 or random strategy. For the first strategy, we employ the k-means algorithm to create $\sqrt{n}$ clusters, getting the medoid of each cluster as a sample, where n is the number of instances in the raw dataset D. We set the number of cluster to $\sqrt{n}$ since this is considered a good heuristic for the upper-bound number of clusters in a data set (PAL; BEZDEK, 1995). After extracting the sample sets $S_1, S_2, \ldots, S_h$, we merge their indexes defining 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, 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. Notice that, the combined sample features $\overline{R}$ will have at most $\sqrt{n} \times h$ instances, enhancing the probability of having samples that represent the distribution of each individual set of features while

not hampering the computational complexity of the overall process since $h \ll \sqrt{n}$.

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 F_i is mapped to $\mathbb{R}^m$ preserving as much as possible the distance relationships in F_i. We do this by minimizing

$$E_{st}(F_i) = \frac{1}{|F_i|^2} \sum_{i}^{|F_i|} \sum_{j}^{|F_i|} \left(\delta(f_i^i, f_j^i) - ||r_i^i - r_j^i||\right)^2 \tag{3.1}$$

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

Besides preserving distance relationships, our mapping process aim 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 $\overline{\Delta} = \frac{1}{h} \sum_i^h \Delta_{F_i}$ combining the distance matrices of all sets of features, where Δ_{F_i} is the distance matrix calculated from F_i. Then we map $\overline{\Delta}$ to the m-dimensional space using the Equation (3.1). The idea is to use this average representation as a guide to align the vectorial representations $R_1, R_2, \ldots, R_h$ minimizing

$$E_{al}(F_i) = \frac{1}{|F_i|^2} \sum_{i}^{|F_i|} \sum_{j}^{|F_i|} \left(d(\overline{r}_i, \overline{r}_j) - ||\overline{r}_i - r_j^i||\right)^2 \tag{3.2}$$

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

Joining Equation (3.1) and (3.2) we define the function we optimize in our mapping process seeking to preserve, as much as possible, the distance relationships of the original features $F_1, F_2, \ldots, F_h$ in the vectorial representations $R_1, R_2, \ldots, R_h \in \mathbb{R}^m$ while aligning them. This function is given by

$$E(F_i) = \lambda \cdot E_{st}(F_i) + (1 - \lambda) \cdot E_{al}(F_i) \tag{3.3}$$

where λ is a used to control the importance of the distance preservation and the alignment to the produced vectorial representations. λ is a a hyperparameter and can be changed to defined a good tradeoff between distance preservation and alignment.

To minimize Equation (3.3) we use a stochastic gradient descent approach with a polynomial decay learning rate. We set the initial learning rate to $\gamma_0 = 0.1$ and the

decay power to $\kappa = 0.95$ following common choices found in the literature. Algorithm 1 outlines our mapping process. Function RANDOM(F_i, n) select n samples from F_i randomly and function INIT() initialize the mapping also randomly. We have tested a deterministic initialization using Fastmap (FALOUTSOS; LIN, 1995) but the gain in quality does not justify the computational overhead. Notice that we normalize all features $f_i \in F_j$, $\forall\ i, j$ before this process, so that the Euclidean norm $||f_i|| = 1$. Given the triangular inequality property ($||f_i - f_j|| \leq ||f_i|| + ||f_j||$), this guarantees a upper limit for the maximum pairwise distance between features. Therefore the distances are in the same range despite the type of feature or its dimensionality, avoiding biasing the process towards the type of feature with the largest maximum distance. In addition, we define the desire dimensionality m of the resulting mappings $R_1, R_2, \ldots, R_h \in \mathbb{R}^m$ as the largest intrinsic dimensionality of $F_1, F_2, \ldots, F_h$, calculated using the maximum likelihood estimation (LEVINA; BICKEL, 2004). Such dimensionality can also be defined by the user if the target dimensionality is known, such as, $m = \{1, 2, 3\}$ for visualization purposes.

Algorithm 1 – Algorithm for mapping different feature sets to a common vectorial space.

$\overline{\Delta} \leftarrow \frac{1}{h}\sum_i^h \Delta_{F_i}$ ▷ calculate the average distance matrix
$\overline{R} \leftarrow$ MAPPING$(\overline{\Delta}, 1.0)$ ▷ compute the dimensionality reduction of $\overline{\Delta}$
for $F_i \in F_1, F_2, \ldots, F_h$ **do**
  $R_i \leftarrow$ MAPPING$(F_i, \overline{R}, \lambda)$
end for

function MAPPING(F, $\overline{R}$, λ)
  $R \leftarrow$ INIT() ▷ initialize the dimensionality reduction
  for $it = 0$ *to* Ω **do**
    $\gamma \leftarrow \gamma_0 \times \left(1 - \frac{it}{\Omega}\right)^{\kappa}$ ▷ polynomial decay of the learning rate
    $F_{rand} \leftarrow$ RANDOM$(F, \sqrt{|F|})$ ▷ get $\sqrt{|F|}$ random samples from F
    for $f_i \in F_{rand}$ **do**
      for $f_j \in F_{rand}$ **do**
        $\nabla E_{st} \leftarrow \left(\delta(f_i, f_j) - ||r_i - r_j||\right) \frac{(r_i - r_j)}{||r_i - r_j||}$
        $\nabla E_{al} \leftarrow \left(d(\overline{r}_i, \overline{r}_j) - ||\overline{r}_i - r_j||\right) \frac{(\overline{r}_i - r_j)}{||\overline{r}_i - r_j||}$
        $r_j \leftarrow r_j - \gamma(\lambda \cdot \nabla E_{st} + (1 - \lambda) \cdot \nabla E_{al})$
      end for
    end for
  end for
  return R
end function

3.1.2 Weighted Feature Combination

Given the samples vectorial representations $R_1, R_2, \ldots, R_h$ we build a set of functions using the process defined in (JOIA *et al.*, 2011b) 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 define in R_i. In this process, each instance $f_j^i \in F_i$ is mapped to the m-dimensional space trough a 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:

$$\sum_k \beta_k \| T_j^i(s_k^i) - r_k^i \|^2 \tag{3.4}$$

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.

Equation (3.4) can be re-written in the matrix form $\| D(AM - B) \|_F$, where $\| \cdot \|_F$ denotes the Frobenius norm, D is a diagonal matrix with entries $D_{ii} = \sqrt{\beta_i}$, and A and B are matrices with the j-th row given by the vectors

$$s_j^i - \frac{\sum_k \beta_k s_k^i}{\sum_k \beta_k} \quad \text{and} \quad r_j^i - \frac{\sum_k \beta_k r_k^i}{\sum_k \beta_k}, \text{ respectively.}$$

Based on that, M is computed as $M = UV$ where U and V are obtained from the singular value decomposition of $A^\top DDB = USV^\top$. Then the vectorial representation v_j^i of f_j^i is given by

$$v_j^i = \left(f_j^i - \frac{\sum_k \alpha_k s_k^i}{\sum_k \alpha_k} \right) M + \frac{\sum_k \alpha_k r_k^i}{\sum_k \alpha_k} \tag{3.5}$$

Equation (3.4) is subject to $MM^\top = I$, which avoids scale and shearing effects, therefore preserving the distance relationships of the input features. Also, notice that 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 (JOIA *et al.*, 2011b).

3.1.3 Feature Combination Widget

To visually support the feature sample combination, we create a widget inspired by the strategy presented in (PAGLIOSA *et al.*, 2015). The idea is to position anchors (circles) representing each different set of features over a circumference, computing the weights $\alpha_1, \alpha_2, \ldots, \alpha_h$ according to their distances to a "dial" contained in the circumference. If

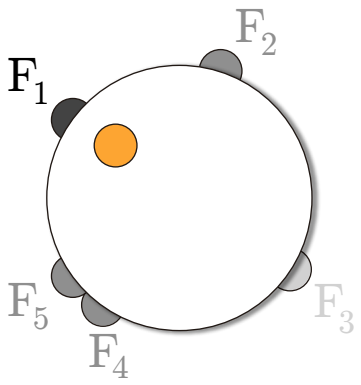


Figure 19 – Feature Combination Widget. Using the orange "dial" users can control the contributions of the different types of features to the final feature combination.

$\tilde{f}_i$ are the coordinates of the anchor representing the feature F_i on the plane and $\tilde{d}$ the coordinates of the "dial", the weight α_i related to F_i is calculated as

$$\alpha_i = 1 \Bigg/ \left(\sum_{j}^{h} \frac{(1 + \|\tilde{f}_i - \tilde{d}\|)^2}{(1 + \|\tilde{f}_j - \tilde{d}\|)^2} \right) \tag{3.6}$$

To help the perception of the weights, we change the transparency level of the anchors and fonts according to $\alpha_1, \alpha_2, \ldots, \alpha_h$. Furthermore, anchors can be moved together in case the users want to assign similar weights to a subset of features. Figure 19 shows the combination widget. In this Figure, the "dial", in orange, is closer to the anchor representing the feature F_1, so the corresponding anchor is darker than the other anchors. In the widget, anchors F_4 and F_5 are positioned close to each other, because users consider both features have the same importance.

3.2 Experimental Validation

In this section, we evaluate our mapping and feature combination processes using different datasets aiming at showing 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.

3.2.1 Datasets

We use five datasets in our tests, named **STL-10** (COATES; NG; LEE, 2011), **Animals** (LAMPERT; NICKISCH; HARMELING, 2009), **Zappos** (YU; GRAUMAN, 2014), **CIFAR-10** (KRIZHEVSKY, 2009) and **Photographers** (THOMAS; KOVASHKA, 2016). These datasets come from a variety of different domains. For example, **STL-10** consists of 13,000 images split into 10 classes of different objects. Similarly, **CIFAR-10** contains 60,000 images of 10 commonly seen object categories (e.g., animals, vehicles, and such) in

lower resolution. The **Animals** dataset is more specific and has 50 animals with 30,475 images in total. **Zappos** is another specialized dataset for shoes with 50,025 images from <Zappos.com> composed of over 10 shoe categories. Finally, the **Photographers** consists of 181,948 photos taken by 41 well-known photographers. Table 3 summarizes the datasets, showing the number of instances and classes.

Table 3 – Datasets employed in the evaluations. We report its size and number of classes.

Name	Size	Classes
STL-10	13,000	10
Animals	30,475	50
Zappos	50,025	4
CIFAR-10	60,000	10
Photographer	181,948	41

3.2.2 Features

We use 4 distinct methods to extract features, representing low-level and high-level image components. Low-level means that each dimension of the feature vector has no inherent meaning, but represent a basic understanding of the image such as edges or color. High-level features have semantic meaning. For example, they denote the presence of an object in the image.

For the low-level features, we represent (1) color with LAB color histogram; (2) texture with Gabor filters (CHEN; LU; ZHANG, 2004) with 8 orientations and 4 scales; and (3) shape with HoG technique (DALAL; TRIGGS, 2005) with a window size of 8. For the high-level, we extract deep-features from the *pool5* layer using a pre-trained CNN CaffeNet (JIA *et al.*, 2014). This network was trained on approximately $1.3M$ images to classify images into 1,000 object categories.

We believe that these features are discriminative for our datasets. For example, we can differentiate a leopard from a panda using a texture extractor. Texture can identify spots in leopard, and differentiate them from other animals. Similarly, color features can be helpful to recognize pandas, where the more common colors are black and white. Also, HOG is helpful to differentiate the type of animals by their shape, e.g., quadrupeds from birds. Finally, object recognition can complement the HOG descriptor. These examples can be generalized to other datasets as well.

3.2.3 Qualitative Results

To confirm the quality of our approach, we quantitatively evaluate our mapping and feature combination processes. For the mapping process evaluation, the five datasets of Table 3 are sampled 10 times randomly reducing them to 5% of their original sizes. We

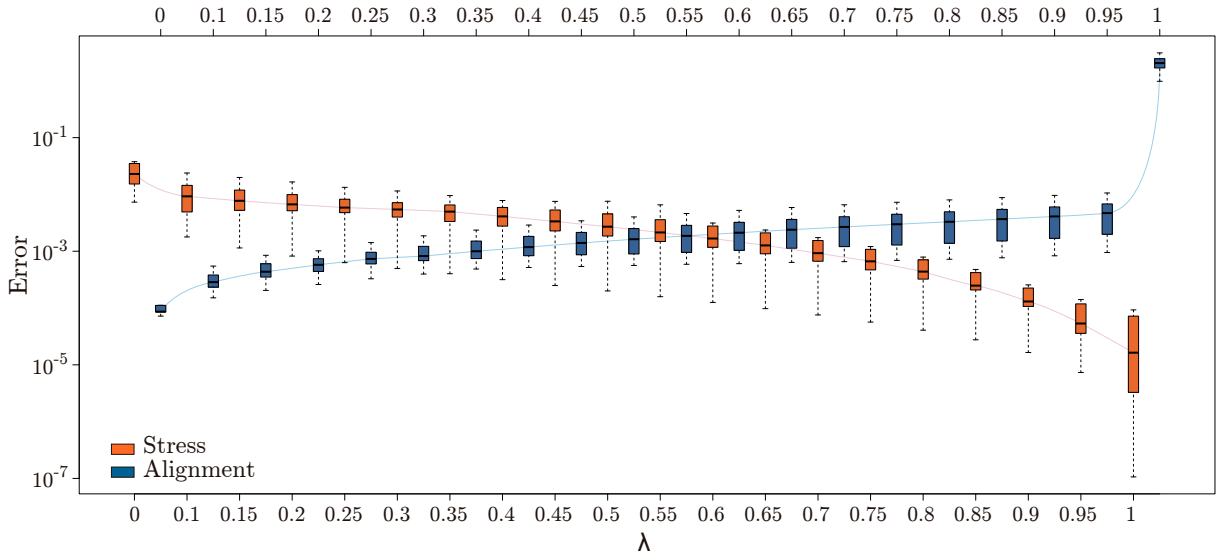


Figure 20 – Comparison for distance preservation and alignment error varying λ. The best trade-off is achieved in the range $[0.45-0.65]$. Line connects the mean values of all box plots.

sample the data since we cannot execute the mapping process with large datasets since its memory footprint is $O(n^2)$. Due to the random initialization (see Algorithm 1), we repeat the mapping process test 15 times. 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 3.1 and Equation 3.2, respectively). We summarize our results in Figure 20 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 small values align the data.

Setting $\lambda = 1$ preserves as much as possible the original distance relationships. This is reflected on a average stress of $E_{st} = 0.0009$, but it does not ensure a good alignment (average alignment of $E_{al} = 2.0343$). On the other hand, $\lambda = 0$ delivered almost a perfect alignment (average alignment of $E_{al} = 0.0001$), but it does not enforce the distance preservation (average stress of $E_{st} = 0.0345$). 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 20).

For a qualitative evaluation, we generate two-dimensional representations of the samples using our mapping process setting the target dimensionality to two. We show the results for the **STL-10**, **Zappos** and **CIFAR-10** datasets in Figures 21, 22, and 23, respectively. In these figures, the points are colored according to image classes. The stress and alignment error values are shown on 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 of the stress error, but the alignment error decreases considerably compared to the first column (see the second measure on the top-left corner). The same behavior is verified in the remaining columns. The last column aligns almost completely all features. As expected, as lambda decreases, the distance preservation also decreases (stress increases), and the alignment improves (alignment decreases). However, the stress changes are minimal. Hence, our approach is capable of making a good alignment between features whereas preserving distance relationships. Similar behavior can be observed in Figure 22 and Figure 23.

3.2.4 Quantitative Results

For the feature combination, we assess the degree the distance relationships of the sample are preserved into the feature fusion of the whole dataset, intending to demonstrate the effectiveness of the user sample manipulation to the produced dataset. In this evaluation, we first generate 30 different weight combinations randomly summing up to 1 and apply it to sample data. Then, we reuse these weights for the whole data fusion and measure if the distance relationships induced by the weights on the sample are presented in the whole dataset. We use the Nearest Neighbor Measure (NNM) (CUI *et al.*, 2006) in this analysis.

NNM quantifies the similarity of each instance in the whole data with its nearest neighbor in the sampled data. NNM is given by Equation 3.7, where D_i is the smallest distance among the i-th instance in the complete dataset and the instances in the sample, and N denotes the number of instances. The authors normalized each dimension of the data to the range $[0, 1]$. However, this results in the loss of the magnitude of the dimensions. So, we change the normalization per dimension by a unit vector normalization per instance to avoid such an effect. The output of NNM is in the interval $[0, 1]$ with larger values indicating better results.

$$NNM = 1.0 - \frac{\sum_i^N D_i}{N} \tag{3.7}$$

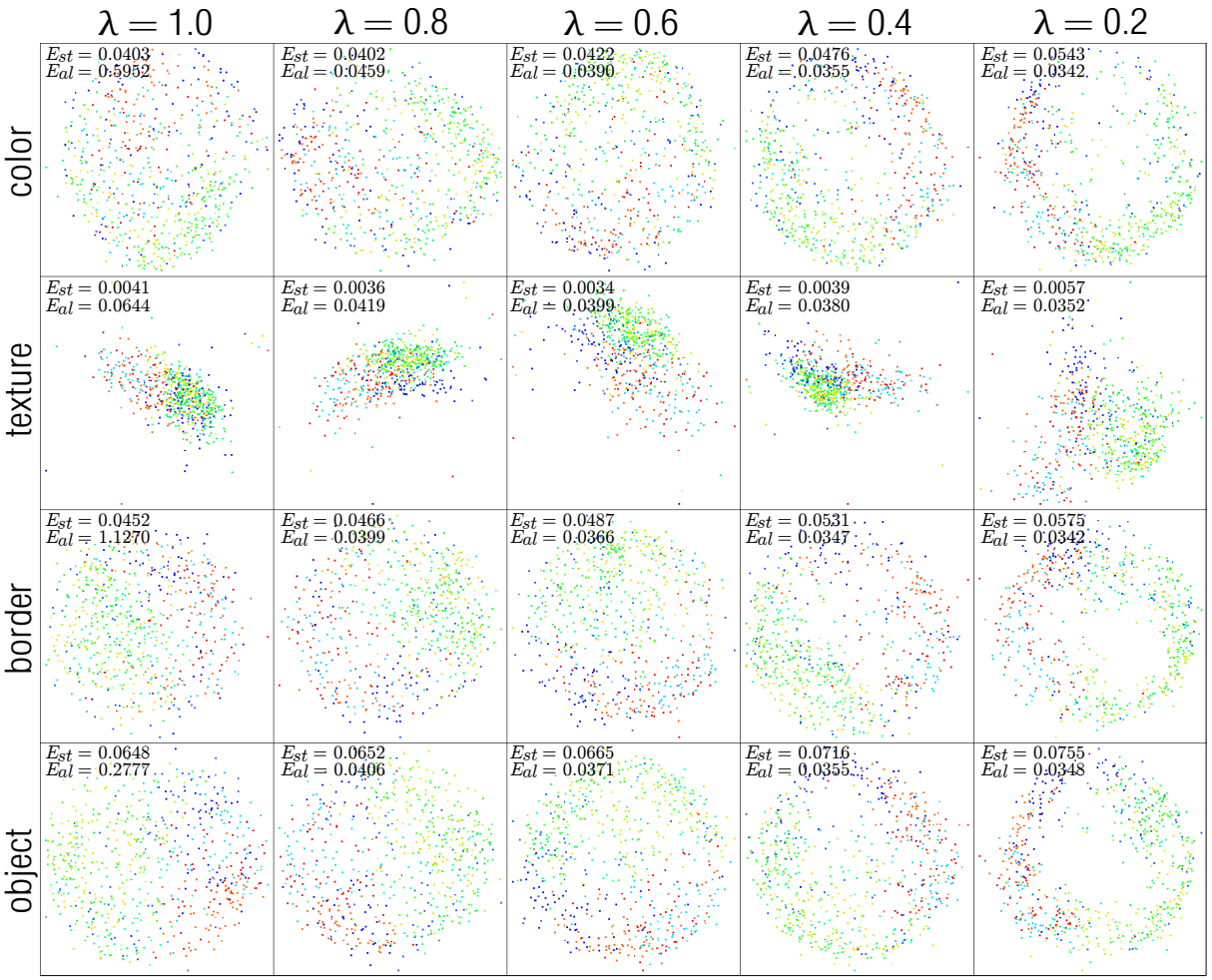


Figure 21 – Resulting mapping process for the **STL-10** dataset. As λ decreases, the features get more aligned (See column 5). Top-left numbers correspond to stress and alignment error.

We compare the NNM values of our feature fusion with two baselines: feature concatenation and distance fusion (see Section 2.2). Boxplots in Figure 24 show that our approach outperforms the other two baselines by at least 5%. The mean value for our method is 0.9365, and the baselines achieve 0.8877 and 0.8958, respectively. Hence, our method preserves more accurately the data distribution of the sample in the whole dataset fusion.

3.3 Summary

In this chapter we presented an approach for feature fusion considering users' semantic understanding. Our approach finds a trade-off between similarity preservation and feature alignment. We test and compare our method on 5 datasets and we demonstrated its efficiency when comparing with traditional feature fusion methods. We also show qualitative results with three different similarity semantics defined by users, the mapping to the complete dataset preserves the initial user-defined semantic.

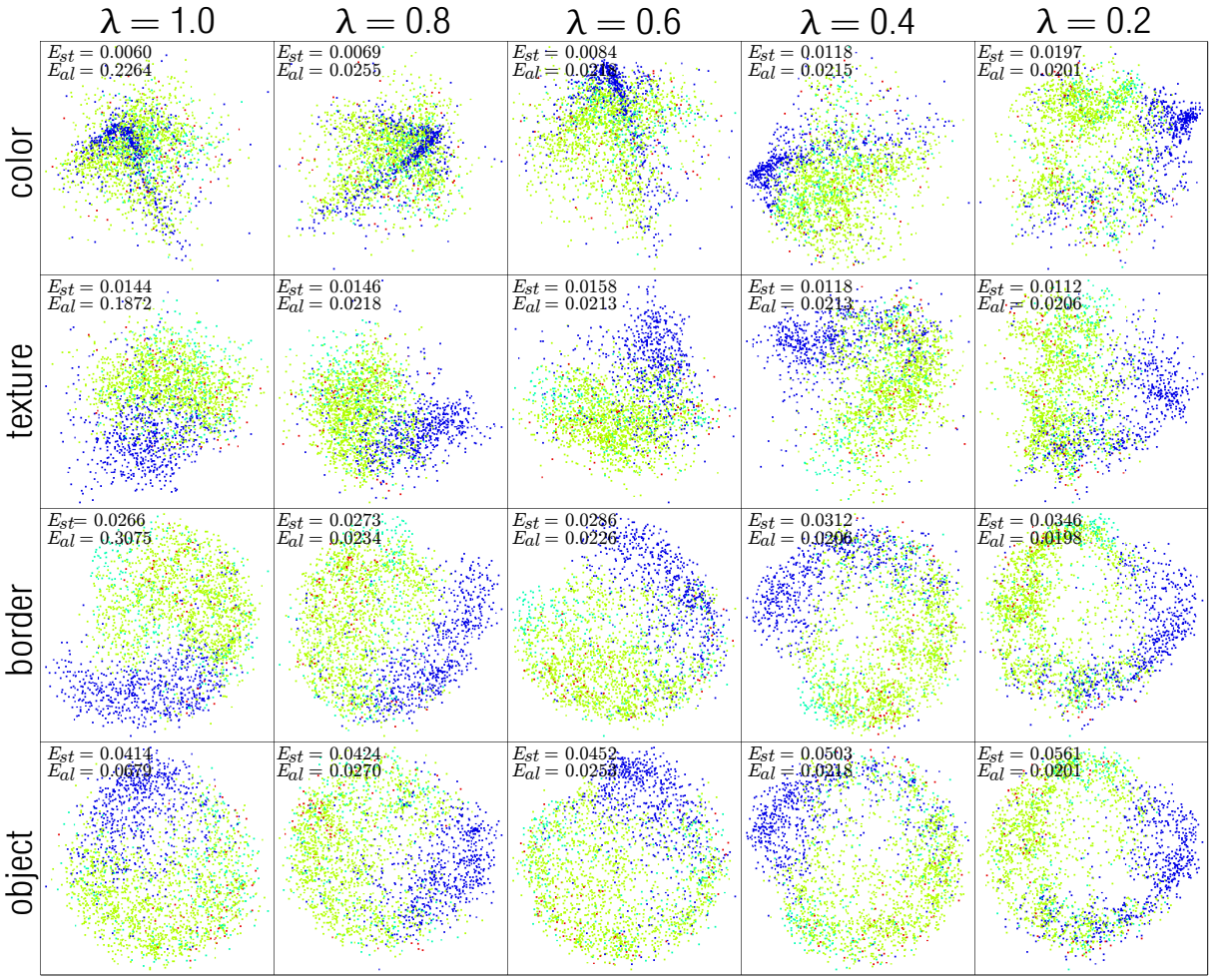


Figure 22 – Resulting mapping process for the **Zappos** dataset. As λ decreases, the features get more aligned (See column 5). Top-left numbers correspond to stress and alignment error.

One drawback of this project is that we present our result in a two-dimensional scatter plot (results of DR are typically presented in a two dimensions). However, it produces overlapping when graphical elements are used to convey information. Therefore, data navigation is difficult for users. A solution for that is to arrange the graphical elements into a grid. We could use current state-of-the-art techniques designed for that task but they are computationally expensive, and they are limited to small datasets. In the next chapter, we solve this issue by designing an efficient technique.

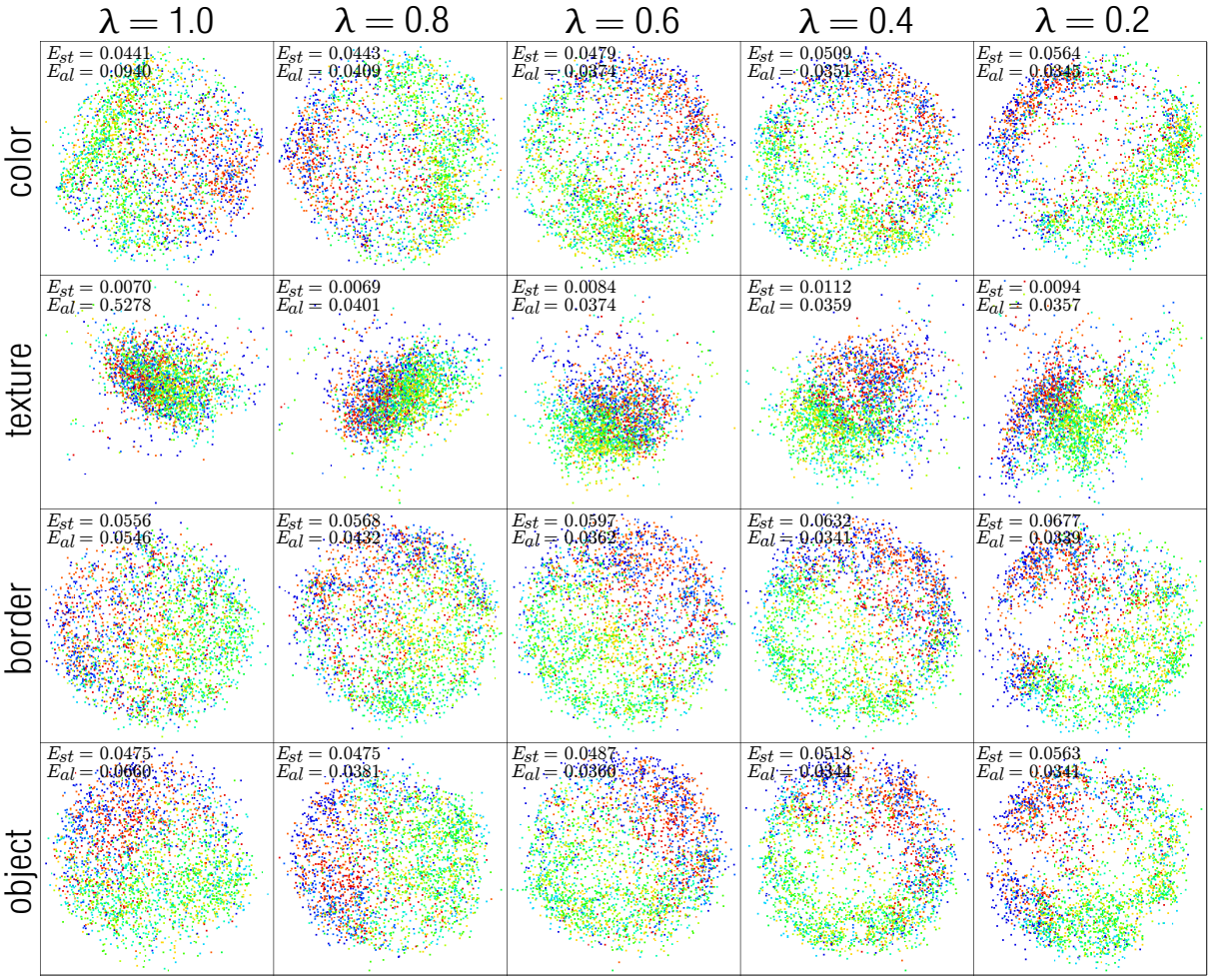


Figure 23 – Resulting mapping process for the **CIFAR** dataset. As λ decreases, the features get more aligned (See column 5). Top-left numbers correspond to stress and alignment error.

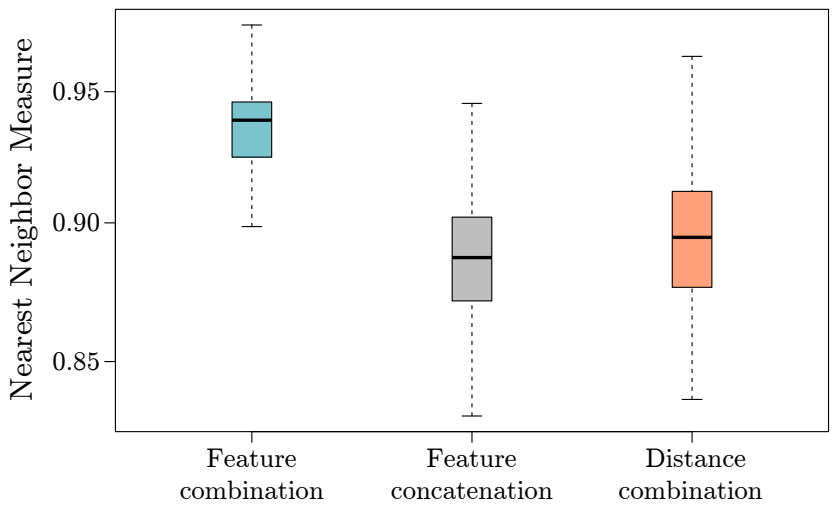


Figure 24 – NNM evaluation. We compare our approach of user-guided feature fusion, with two baselines: feature concatenation, and feature combination trough distance measures.

Chapter 4

Distance Preserving Grid Layouts

Distance preserving visualization techniques compose a family of strategies that seek to map data instances into graphical elements so that the pairwise distances calculated between instances are preserved as much as possible between graphical elements. Among the existing strategies, the Dimensionality Reduction (DR) techniques have emerged as one of the fundamental tools for data analysis (JOIA *et al.*, 2011a). Through the definition of a proper function to compute the distance between instances, DR techniques produce layouts where the dissimilarity patterns are "visible," allowing the execution of tasks that involve the identification and analysis of distance relationships. However, DR techniques present limitations when the graphical elements are used to convey information. These tend to produce layouts with overlapped graphical elements, so suffering from occlusion problems. This occlusion problem make data exploration difficult to users. To address such limitation, some approaches employ post-processing strategies (GOMEZ-NIETO *et al.*, 2014; STROBELT *et al.*, 2012) or put constraints on the projection process (PINHO; OLIVEIRA; LOPES, 2010) to remove the overlapping. However, they make poor use of the visual space, creating layouts with void areas. Aiming at making 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 grids are computationally expensive and limited to small datasets.

In this chapter, we propose Distance-preserving Grid (DGrid), a technique to efficiently arrange layouts generated by DR techniques. DGrid is a two-step approach that combines a projection technique with an assignment strategy to create orthogonal regular 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. Its computational efficiency is demonstrated by a set of comparisons against other techniques that creates

distance preserving grids.

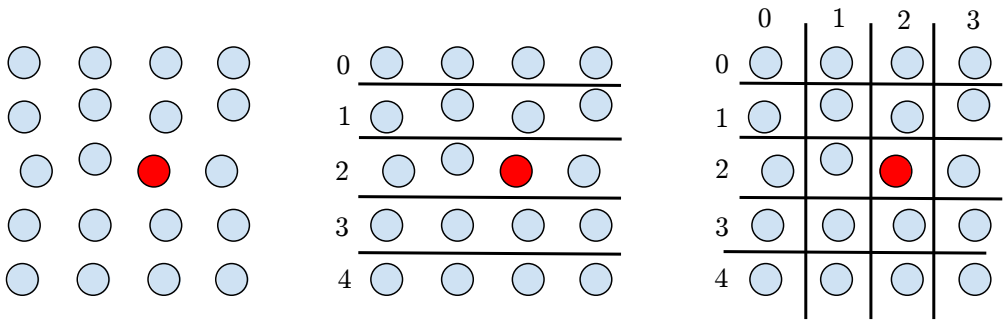
The remainder of this chapter is organized as follows. In section 4.1, we describe our approach for creating grid layouts, including our formulation, and implementation details. In Section 4.2, we show that our method improves upon standard approaches via quantitative experiments using performance and time comparison measures. Finally, we summarize this chapter in Section 4.4.

4.1 Proposed Methodology

The *Distance-preserving Grid (DGrid)* employs a two step approach to generate uniform grids that preserve, as much as possible, the distances relationships of a given dataset. On 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 two-dimensional 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 $g_{p,q}$.

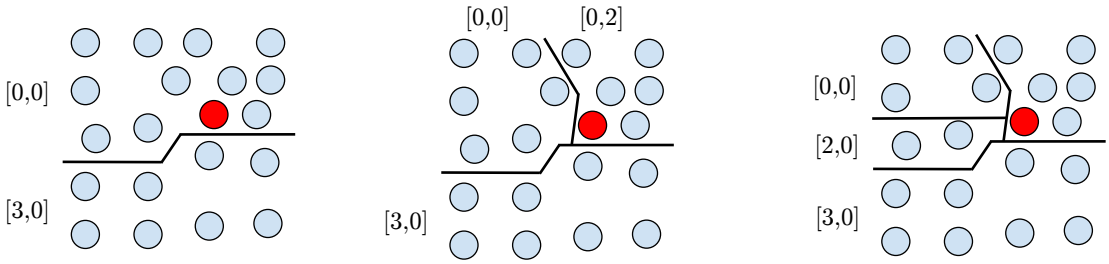
The reasoning behind our approach is that if the projection $\mathscr{P}$ precisely preserves the distances relationships in $\mathscr{D}$, and if $\mathscr{G}$ preserves the geometry of $\mathscr{P}$, $\mathscr{G}$ will preserve the distances relationships in $\mathscr{D}$. Consider that $\mathscr{P}$ has been obtained from $\mathscr{D}$ (this is later discussed in Section 4.1.2). 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, like in Figure 25(a) for a grid with 5 rows and 4 columns, the process to assign $\mathscr{P}$ to $\mathscr{G}$ is trivial. First, we horizontally split the space into 5 partitions, with 4 instances in each, defining the row index of the instances in each partition (the vertical numbers in Figure 25(b)). Then, we vertically split the partitions so that each instance is placed in its partition, defining the column index of each instance (the horizontal numbers in Figure 25(c)). In this example, the instance colored in red is mapped to the grid cell $g_{2,2}$, that is, the cell occupying the third row and third column.

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, $|\mathscr{P}_1| = w \times \lceil u/2 \rceil$. Figure 26(a) presents the result if this



(a) Initial projection (b) Horizontal split (c) Vertical split

Figure 25 – Process of assigning a projection to a grid when the projection is uniformly distributed over the plane and follows the grid pattern.



(a) Horizontal bisection (b) Vertical bisection (c) Horizontal bisection

Figure 26 – Process of bisecting the projection and calculating the top-left corner indexes of the resulting grids. This process is applied until each data instance is assigned to a grid cell.

process is applied to a slightly modified version of Figure 25(a) to show an example where the bisectors cannot be straight lines in $\mathbb{R}^2$ but arbitrary curves and the simple process of Figure 25 cannot be directly used.

Figure 26(b) and Figure 26(c) show the bisecting process recursively applied. 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. For instance, on Figure 26(a), the index of the top-left corner cell of the upper partition is $[0,0]$, indicating that the grid generated from it starts at row 0 and column 0. For the lower partition, the index of the top-left corner cell is $[3,0]$, indicating that the grid generated from it starts at row 3 and column 0. Consider that the input projection $\mathscr{P}$ is split into $\mathscr{P}_1$ and $\mathscr{P}_2$, where $\mathscr{P}_1$ is the upper partition for a horizontal cut or the left partition for a vertical cut. Also, let (i, j) be the index of the top-left corner cell of $\mathscr{P}$. By construction, the index of the top-left corner cell of $\mathscr{P}_1$ is (i, j), and the index of $\mathscr{P}_2$ is $(i + \lceil w/2 \rceil, j)$ for a horizontal cut, and $(i, j + \lceil u/2 \rceil)$ for a vertical cut.

As mentioned before, this 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. Notice that this returns the same grid as the process presented in Figure 25 if the input projection obeys the uniform distribution and grid pattern constraints. Therefore, it is not necessary to test if such constraints hold in any step of the algorithm, rendering a much faster and simpler process to implement.

Algorithm 2 puts all these pieces together, showing the overall process adopted by our approach to assigning a projection to a grid. The function $\textsc{Split}_y(\mathscr{P},k)$ performs the horizontal bisection. In this process, $\mathscr{P}$ is sorted according to the y-coordinates ($\mathscr{P}$ is viewed as a list), and the first k instances are assigned to $\mathscr{P}_1$ and the remaining to $\mathscr{P}_2$. The function $\textsc{Split}_x(\mathscr{P},k)$ performs the vertical bisection using the same process, but sorting $\mathscr{P}$ according to the x-coordinates. Notice that since the bisecting process always assigns to the upper and left partitions enough elements to result in a filled grid, spaces are not opened in the interior of the final grid. All empty cells are grouped on the bottom-right corner.

Algorithm 2 – Process of assigning a projection to a grid.

```
function DGrid(𝒢, 𝒫, (w,u), (i,j))
  if 𝒫 ≠ ∅ then
    if |𝒫| = 1 then                          ▷ 𝒫 has one instance
      g_{i,j} = x_0                          ▷ cell g_{i,j} ∈ 𝒢 receives the only instance in 𝒫
    else
      if w > u then
        𝒫_1, 𝒫_2 ← Split_y(𝒫, ⌈w/2⌉ × u)
        DGrid(𝒢, 𝒫_1, (⌈w/2⌉, u), (i, j))
        DGrid(𝒢, 𝒫_2, (w − ⌈w/2⌉, u), (i + ⌈w/2⌉, j))
      else
        𝒫_1, 𝒫_2 ← Split_x(𝒫, w × ⌈u/2⌉)
        DGrid(𝒢, 𝒫_1, (w, ⌈u/2⌉), (i, j))
        DGrid(𝒢, 𝒫_2, (w, u − ⌈u/2⌉), (i, j + ⌈u/2⌉))
      end if
    end if
  end if
end function
```

A different interpretation of this binary partition method is to consider it a translation process that removes void spaces so that the distribution of the projected points on the vertical and horizontal directions is as uniform as possible and follows the grid dimensions, that is, the desired number of rows and columns. Since translation is a rigid transformation that preserves relative distances, in the worst case scenario half of the distance relationships are fully kept when a projection is bisected. Typically, more than that is preserved, so that the geometry of $\mathscr{P}$ is preserved, up to an extent, by $\mathscr{G}$, and,

consequently, the produced grid preserves the distances relationships in $\mathscr{D}$.

4.1.1 Grid Dimension

The process of assigning a projection to a grid defined in the previous section is very flexible. Since it focuses on splitting the partitions considering the number of rows and columns, instead of the number of data instances, it is not limited to any particular grid shape. The only constraint is that the number of grid cells should be larger or equal to the number of data instances, that is, $w \times u \geq N$.

In this work, we allow the control of the grid shape by defining its aspect-ratio Λ. Here, the aspect-ratio can be the ratio between the number of rows and the number of columns of the final grid, or the ratio between the height and width of the visual space. Given Λ, the target grid dimension is calculated as

$$\begin{aligned} w &= \lfloor \sqrt{N * \Lambda} \rfloor \\ u &= \lceil N/w \rceil \end{aligned} \tag{4.1}$$

If $\Lambda = 1$, the resulting grid will as square as possible. If $0 < \Lambda < 1$, the resulting grid will present more columns than rows. The opposite if $\Lambda > 1$.

4.1.2 Projecting the Dataset

We aim to preserve on the produced grid the distance relationships of a given dataset $\mathscr{D}$ by assigning similar data instances to close grid cells, and dissimilar ones to far apart cells. In this context, the projection $\mathscr{P}$ plays a central role since it guides the grid geometry. Therefore, $\mathscr{P}$ should preserves, as much as possible, the distance relationships in $\mathscr{D}$. In the current literature, there are several multidimensional projection techniques to derived $\mathscr{P}$ from $\mathscr{D}$. Typically, the most precise techniques, such as the classical multidimensional scaling (TORGERSON, 1965) or the t-SNE (MAATEN; HINTON, 2008b), are computationally expensive, whereas the less precise out-of-sample methods, such as LAMP (JOIA *et al.*, 2011a) or the PLMP (PAULOVICH; SILVA; NONATO, 2010), can handle very large datasets in a reasonable amount of time. Thereby, the choice of what technique to use rely mostly on the trade-off between the size of the dataset and the desired precision.

4.2 Experimental Validation

We first describe our datasets (Section 4.2.1). In sections 4.2.2 and 4.2.3, we compared Distancepre-serving Grid (DGrid) and baselines such as Self-Sorting Map (SSM) (STRONG; GONG, 2014), Kernelized Sorting (KS) (QUADRIANTO *et al.*, 2010) and IsoMatch (FRIED

et al., 2015). All the experiments were generated in an Intel Core $i7$ CPU@2.8GHz, with 16GB of RAM. For the SSM, KS, and IsoMatch techniques; we used the original codes, made available by the authors.

4.2.1 Datasets

For evaluation, we selected some datasets from the *UCI Machine Learning Repository* (LICHMAN, 2013) with real-valued attributes. We only get real-valued datasets to calculate (Euclidean) distances properly. Also, we discarded the datasets with missing values. We divided them in two groups according to sizes. The former group varies between 100 to 2,500 instances, resulting in 38 datasets and we limited their sizes due to the high computational complexities and running times of KS and IsoMatch. The latter group is used for time comparison (Second part of Table 4) of our technique and SSM. It contains 10 much bigger datasets, varying the sizes from 17,000 to 130,000 instances. Table 4 details these datasets, presenting their names, sizes, and number of dimensions. For all tests, we normalized the datasets so that the columns (attributes) have zero mean and standard deviation equal to one.

4.2.2 Quantitative results

In this section, we present a quantitative evaluation of the DGrid technique, comparing it against the state-of-the-art distance-preservation grid techniques, viz., KS, SSM, and IsoMatch. In this comparison, we use three different quality metrics, k-neighborhood preservation index (FADEL *et al.*, 2015), cross-correlation (STRONG; GONG, 2014), and energy function (FRIED *et al.*, 2015).

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 $\mathscr{G}$. It is calculated as

$$NP_k = \frac{1}{N}\sum_{i}^{N}\frac{N_{k_i}^{\mathscr{D}} \cap N_{k_i}^{\mathscr{G}}}{k} \tag{4.2}$$

where $N_{k_i}^{\mathscr{X}}$ is the set containing the indexes of the k-nearest neighbors of d_i in $\mathscr{D}$, and $N_{k_i}^{\mathscr{G}}$ is the set containing the indexes of the k-nearest neighbors of g_i in $\mathscr{G}$. 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. It measure is presented in Section 2.3.3 and it is reproduced below,

Table 4 – Datasets employed in the evaluations. We have selected all datasets from the *UCI Machine Learning Repository* with real-valued attributes and sizes up to a limit, allowing the comparison of the techniques in different scenarios.

Name	**Size**	**Dimensions**
Concrete Slump Test	103	10
Breast Tissue	106	10
LSVT Voice Rehabilitation	126	309
Iris	150	4
Urban Land Cover	168	148
Planning Relax	182	13
Parkinsons	197	23
Connectionist Bench	208	60
Seeds	210	7
Glass Identification	214	10
Yacht Hydrodynamics	308	7
Vertebral Column	310	6
Ecoli	336	8
Leaf	340	16
Libras Movement	360	91
PEMS	440	138,672
Forest Fires	517	11
Vowel Recognition	528	10
Istanbul Stock Exchange	536	8
Climate	540	18
WDBC	569	32
DrivFace	606	6,400
Hill-Valley	606	100
Blood Transfusion	748	5
Gene Expression	801	20,531
Arcene	900	10,000
MicroMass	931	1,300
Cloud	1,024	10
Concrete Compressive Strength	1,030	9
Geographical Original of Music	1,059	68
Banknote Authentication	1,372	5
Yeast	1,484	8
Airfoil Self-Noise	1,503	5
Plant species leaves	1,600	64
Drug Consumption	1,885	32
Cardiotocography	2,126	23
Image Segmentation	2,100	19
Statlog (Image Segmentation)	2,310	19
HTRU2	17,898	9
Default of credit card	30,000	23
Online News Popularity	39,644	61
Facebook Comments	40,949	54
Tamilnadu Electricity Board	45,781	4
Sensorless Drive Diagnosis	58,509	49
Corel Image Features	68,040	89
Blog Feedback	56,497	281
FMA: A Dataset For Music Analysis	106,574	518
MiniBooNE Particle Identification	130,065	50

$$CC = \sum_{i}^{N}\sum_{j}^{N} \frac{(\partial(g_i, g_j) - \overline{\partial})(\delta(d_i, d_j) - \overline{\delta})}{\sigma_\partial \sigma_\delta}$$

where $\delta(d_i, d_j)$ is the dissimilarity between pairs of instances, $\partial(g_i, g_j)$ is the distance between the cells the instances are assigned to, $\overline{\partial}$ is the mean distance between any two cells, $\overline{\delta}$ is the mean distance between any two pairs of instances, and σ_∂ and σ_δ are the corresponding standard deviation. 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. This function is presented in Section 2.3.3 and it is recall:

$$E_p = \left(\sum_i^N \sum_j^N \frac{|c \cdot \delta(d_i, d_j) - \partial(g_i, g_j)|^p}{\sum_r^N \sum_s^N \partial(g_r, g_s)^p} \right)^{\frac{1}{p}}$$

where p defines the employed norm, and c is a scaling constant (see (FRIED *et al.*, 2015) for more details). As suggested in (FRIED *et al.*, 2015), we set $p = 1$ since it favors solutions which preserve the smaller distances more than the larger ones. Also, we invert the original equation using $E'_p = 1 - E_p$ so that it ranges in $[0, 1]$ with larger values rendering better results.

Figure 27 presents boxplots summarizing the results of each technique considering all the 38 datasets. For the DGrid, we report results using the t-SNE and LAMP techniques to generate the input projections. Although the IsoMatch originally employs the ISOMAP as input, we also report results using the t-SNE and LAMP, so it is possible to compare the DGrid and the IsoMatch isolating the projection contribution to the quality of the produced results. The 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 27, 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 between the projections and the produced grids is expected since the techniques we use do not create uniformly distributed projections (see Section 4.1). Also note that direct comparisons only make sense among grid layouts, not among grids and projections.

Regarding the k-neighborhood preservation index, Figure 27a, 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 to 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 $k = (\lfloor\sqrt{0.05 * N}\rfloor)^2$. We use this approximation instead of 5% to match the grid topology when calculating the neighborhoods.

Figure 27b shows the cross-correlation results. On average, DGrid with LAMP ($\overline{CC'} = 0.80$) presents better results than IsoMatch ($\overline{CC'} = 0.78$) and KS ($\overline{CC'} = 0.78$), with a smaller spread regarding the best and worst results. Different from the k-neighborhood preservation index, which is a local measure, the cross-correlation is global, explaining why the results considering LAMP as input are better than the ones considering the t-SNE, which is also confirmed by the projection boxplots in red. This renders exceptional flexibility to our technique since it allows selecting a projection technique that fulfills specific needs, considering different global or local geometry properties, generating grids that satisfactorily preserve them.

The quality of our approach is also confirmed by the energy function metric (Figure 27c). The DGrid with the LAMP ($\overline{E'_p} = 0.65$) again outperforms the other techniques, IsoMatch ($\overline{E'_p} = 0.63$) and KS ($\overline{E'_p} = 0.63$). Since the energy function is a global measure, the LAMP technique presents better results than the t-SNE (see the red boxplots), which reflects in the produced grids. In (FRIED *et al.*, 2015), the authors show that the energy function strong correlates with human performance in search tasks, pointing that this is a good measure of grid organization. The same holds for the cross-correlation measure. Therefore, the attained results provide evidence to place the combination of DGrid with the LAMP as one of the best choices for tasks that involve the analysis of similarity relationships based on grids.

To complement the statistical analysis conveyed by the boxplots, providing more detailed information, we show in Figures 28, 29, and 30 the resulting grids for some selected datasets. Aiming at showing different aspects of each technique, we choose datasets with varied distance distributions, from a dataset with most instances similar among themselves (Forest) to a dataset with most instances dissimilar among themselves (MicroMass). 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 the DGrid, KS, and IsoMatch place all empty cells on the grid borders whereas the 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 processed in a fraction of the time but also it can address larger problems that neither KS nor IsoMatch are capable of.

Regarding the k-neighborhood preservation index grids (Figure 28), the KS and the DGrid with t-SNE, which attained the best results on average (see Figure 27), do not present spots concentrating bad quality cells, the error is uniformly spread over the grid. Conversely, SSM tends to group the bad quality cells close to the empty cells, showing their negative impact on the produced layouts. The cross-correlation grids (Figure 29) report an intriguing pattern produced by all techniques. For the Forest dataset, there is a clear spot with bad quality cells, concentrated in the border of two different regions. A close examination explains the reason. Since the Forest dataset is composed of two very distinct groups of instances, approaching them in the produced layout increases the error in the border cells, an inevitable aspect of grid layouts. The energy function grids (Figure 30) also present an unusual pattern regarding the Forest dataset. The two different groups of instances can also be identified, but in these grids, the larger group presents significant worse results if compared to the smaller one (compare with Figure 29). In this case, the problem is related to the size of the groups. Since one group is much larger than the other, considering the groups individually defines two different scenarios regarding distance distribution. For the smaller group, most instances are dissimilar between themselves, but for the larger, most instances are similar between themselves. Since the energy function is global and measures the distance preservation, the differences in distribution affect the quality of the grid cells.

Given the process we develop to derive grids from projections, our approach can obtain better results if the number of rows and columns, controlled by $\mathbf{\Lambda}$, is defined considering the distribution of the input projections. Figure 31 shows the impact of varying $\mathbf{\Lambda}$ into the quality of the produced grids. To have better control of this test, we artificially generate a projection with three times more points in the vertical direction than in the horizontal direction. As expected, the best results are attained when $\mathbf{\Lambda} = 3$ (the dashed line). In all examples in this section we have used squared grids, setting $\mathbf{\Lambda} = 1$. Since most techniques we are comparing to do not depend on projections, we prefer to set a fixed value instead of using the best possible value of $\mathbf{\Lambda}$ to not bias the evaluation favoring our approach.

Finally, we have compared DGrid with SSM regarding the running times. We have removed the other techniques from this comparison since they are computationally expensive, not capable of processing large datasets. In this test we have selected 10 datasets from the *UCI Machine Learning Repository*, varying the sizes up to $130,000$ instances. The employed datasets are detailed in the second part of Table 4. Figure 32 summarizes the results. To allow a fair comparison, DGrid and SSM are both implemented in Java.

Besides the boxplots for the DGrid and the SSM techniques, the figure shows individual boxplots for the projection and the grid assignment steps. In this example, we are using the LAMP to project the data. On average, the DGrid is almost two orders of magnitude faster than the 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 from large datasets.

4.2.3 Qualitative results

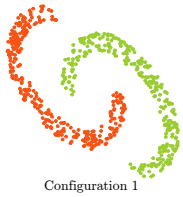
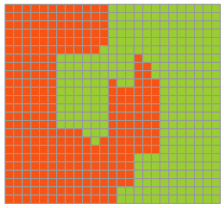
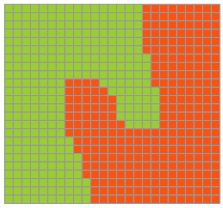
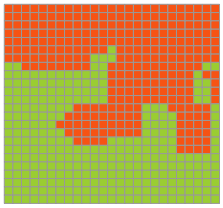
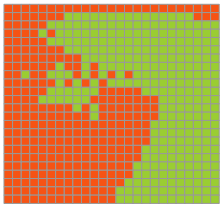
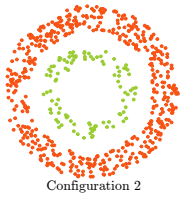
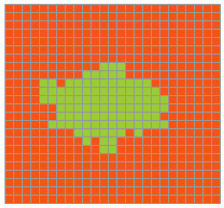
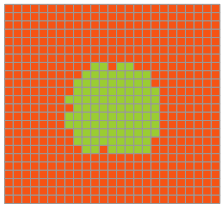
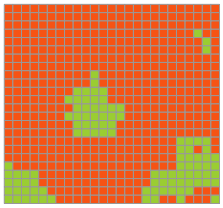
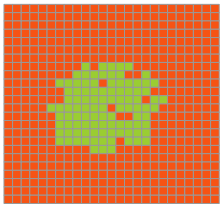
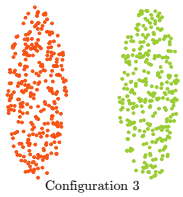
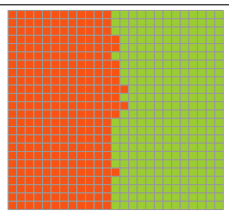
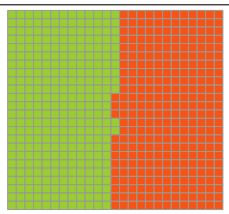
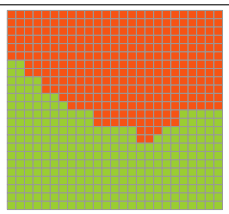
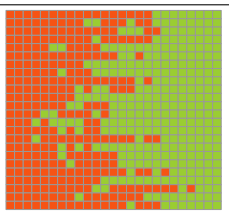
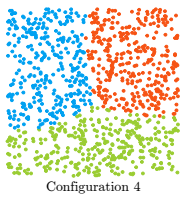
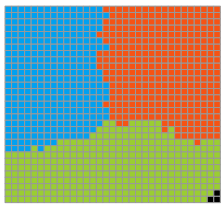
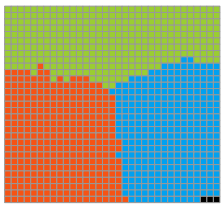
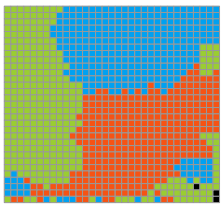
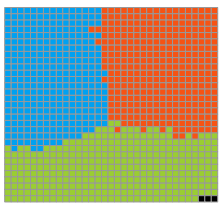
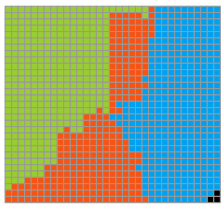
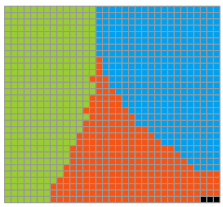
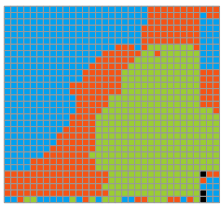
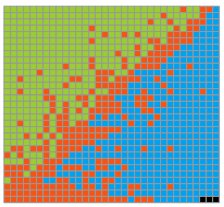
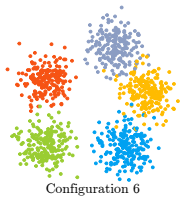
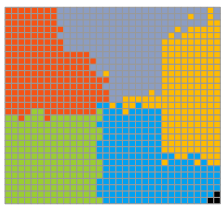
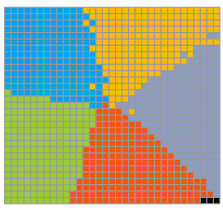
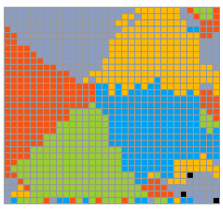
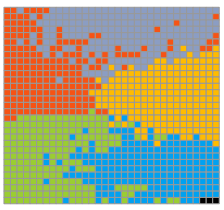
In order to analyze and understand better the behavior of DGrid and the baselines in scenarios with different properties, we have employed six different initial configurations in our validation. Figure 33 shows these initial configurations. Figure 33(a) and Figure 33 (b) measures the capacity of the techniques to preserve the shape of initial point locations considering nonlinear relationships. In addition, configurations (c) and (d) contains points to check the capacity of the techniques to preserve the groups and their separation on the final layout. Finally, Figures 33 (e) and 33 (f) contain more groups and different density distribution. . The first and second row of Figure 33 contain 600 and 1120 points, respectively.

Table 5 presents the comparison of each of the previous initial configurations with the baseline techniques. The points' colors of the 2D configuration are used to color the grid cells. For configuration (d), IsoMatch presents good results, but it fails for the remaining configurations. We suspect that the assignment problem is not generating good correspondences. SSM presents good results for configuration (c) (although final shapes that do not correctly match the original positions), but it fails with the remaining datasets. Visually, SSM has the worst results in most configurations. This is due to the permutation and swapping process of instances between grid cells that are non-square power-of two grids. In general, DGrid and KS visually present the best results. However, KS do not match the original positions because instances are transformed to a kernel space. DGrid correctly matches with the original positions, showing its superiority with the baselines.

4.3 Discussion and Limitations

The space partition strategy presented in Algorithm 2 shares similarities with the kd-tree technique (BENTLEY, 1975). In both cases, the recursive process of bisecting the space into partitions and sub-partitions constructs complete binary trees. The difference is that the kd-tree only considers the spatial position of the points in this process whereas our approach incorporates the number of rows and columns to it. As a result, our technique

Table 5 – Results of applying DGrid, KS, SSM, and IsoMatch techniques on different initial configurations, respectively. The color of the initial points are used to color the grid cells. In general, the DGrid produced layouts that best preserves the shape and neighborhoods of the initial configurations, achieving a more reliable representation of the distance preservation.

Configurations	DGrid	KS	SSM	IsoMatch
 Configuration 1				
 Configuration 2				
 Configuration 3				
 Configuration 4				
 Configuration 5				
 Configuration 6				

can create grids with an arbitrary number of rows and columns while the kd-tree can only create squared grids that are power-of-two (STRONG; GONG, 2014).

Regarding the computational complexity, the DGrid has two distinct steps. The projection and the space partition. Different techniques can be used in the first step, with different complexities. Here we use the t-SNE which is $O(N^2)$ and the LAMP which is $O(\min\{m^2N^{\frac{3}{2}}, mN^2\})$, where m is the number of dimensions of the dataset. Recalling

that every time a partition is bisected, its instances are sorted according to the x or y coordinates. If an $O(N \log N)$ sorting algorithm is used, the computational complexity of the space partition step is $O(N \log^2 N)$. Therefore, the overall computational complexity is dominated by the projection process, which is confirmed by the running times (see Figure 32). If a faster approach is required, changing the projection technique is an option, for instance, PLMP (PAULOVICH; SILVA; NONATO, 2010) is $O(N)$, but probably the quality of the obtained grids will be penalized.

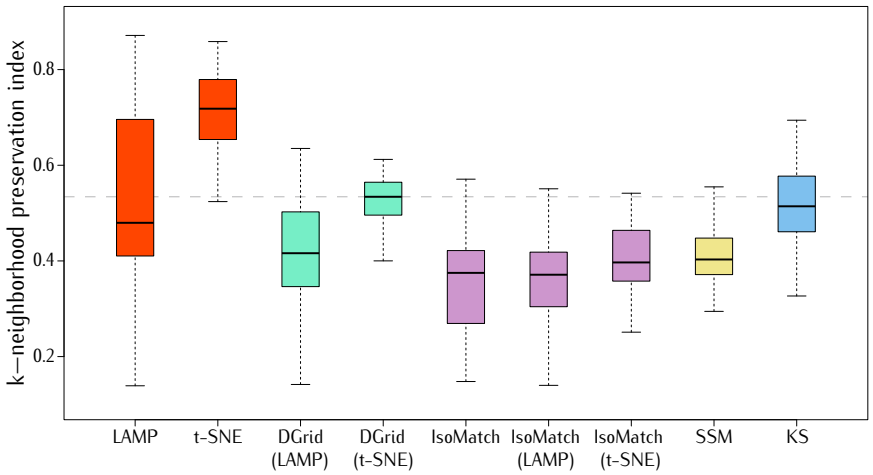
The way DGrid was conceived only allows it to generate orthogonal regular grids. KS and IsoMatch are more flexible. Since they are based on assignment processes, they can map data instances into non-orthogonal domains. However, they are computationally expensive, $O(N^3)$. For orthogonal grids the DGrid is much faster, attaining similar or even better results considering different quality metrics, rendering DGrid a very attractive technique for processing large datasets.

Finally, in our space partition process, we consider a simple splitting method, dividing the space so that the resulting partitions contain approximately half of the instances. Defining a better way to partition the space is an aspect that deserves to be investigated more deeply, for instance, guiding the partition according to the distribution of the points on the plane. However, finding the best partitioning considering both the data distribution and the grid dimension constraint is not a trivial task.

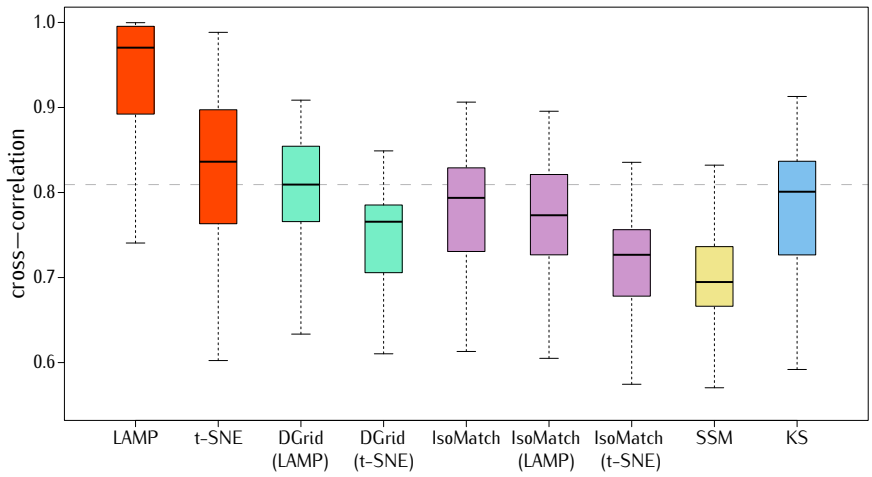
4.4 Summary

We showed an approach for generating grid layouts that preserve distance information, called *Distance-preserving Grid (DGrid)*. DGrid is a two-step approach that combines the output generated by a DR technique with a space-partitioning strategy to create orthogonal regular grids. The set of comparisons we have provided shows that DGrid outperforms the existing state-of-the-art techniques considering three quality metrics. We also outperform current techniques being almost two orders of magnitude faster than the fastest existing technique.

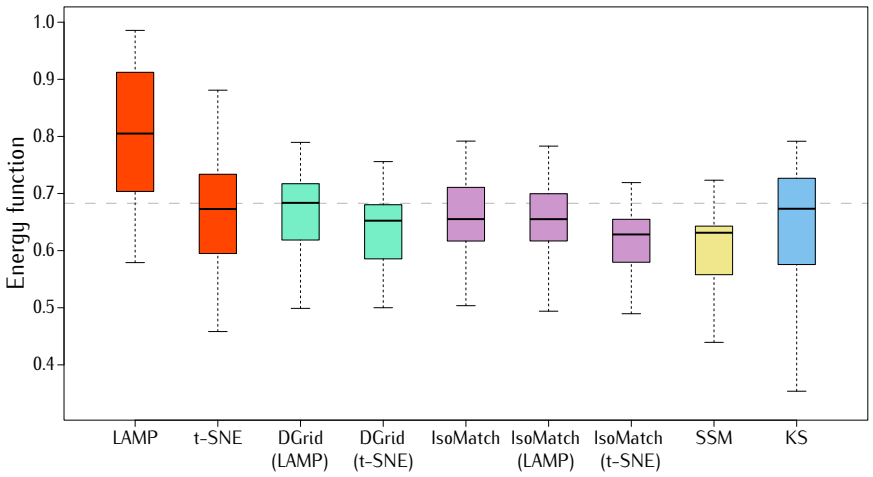
As mentioned before the input for DGrid is a layout generated by any DR technique. In our experiments, we use LAMP and t-SNE. When we evaluate these techniques in term of quality metrics, we find that one is better than the other and vice versa. This is due to the nature of the algorithms. T-SNE favors the preservation of small neighborhoods instead of global distance preservation by LAMP. It could be interesting to find a technique that has a trade-off between measures or even better find a technique that matches user requirements. This let us as future work.



(a) *k*-neighborhood preservation index



(b) cross-correlation



(c) energy function

Figure 27 – Boxplots of *k*-neighborhood preservation index, cross-correlation, and energy function. In all these aspects, 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.

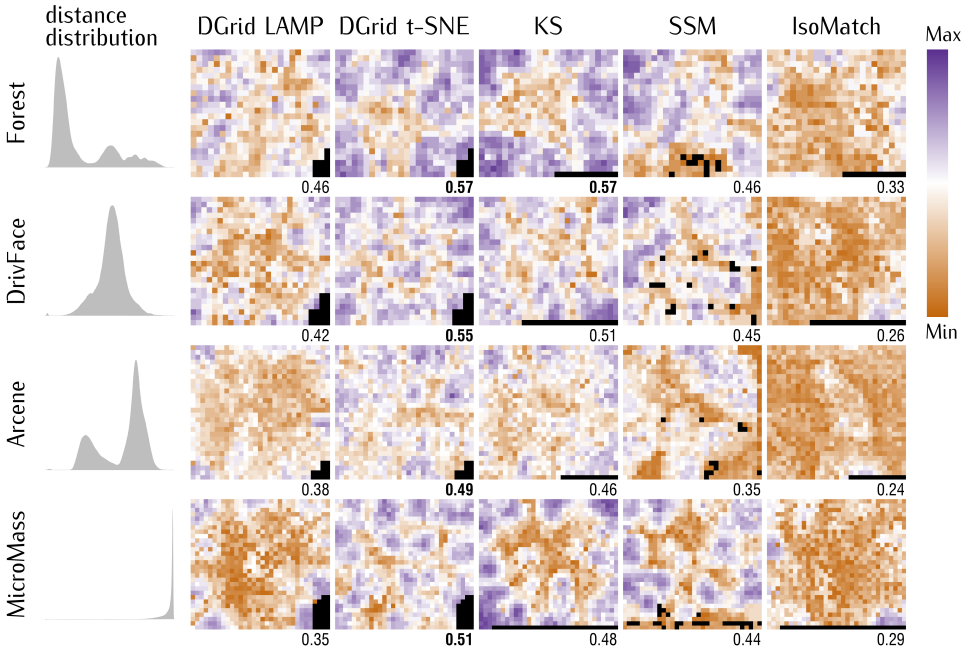


Figure 28 – Resulting grids colored according to the **k-neighborhood preservation index**. SSM technique groups bad quality cells close to the empty cells, showing the negative impact of empty spots on the produced layouts.

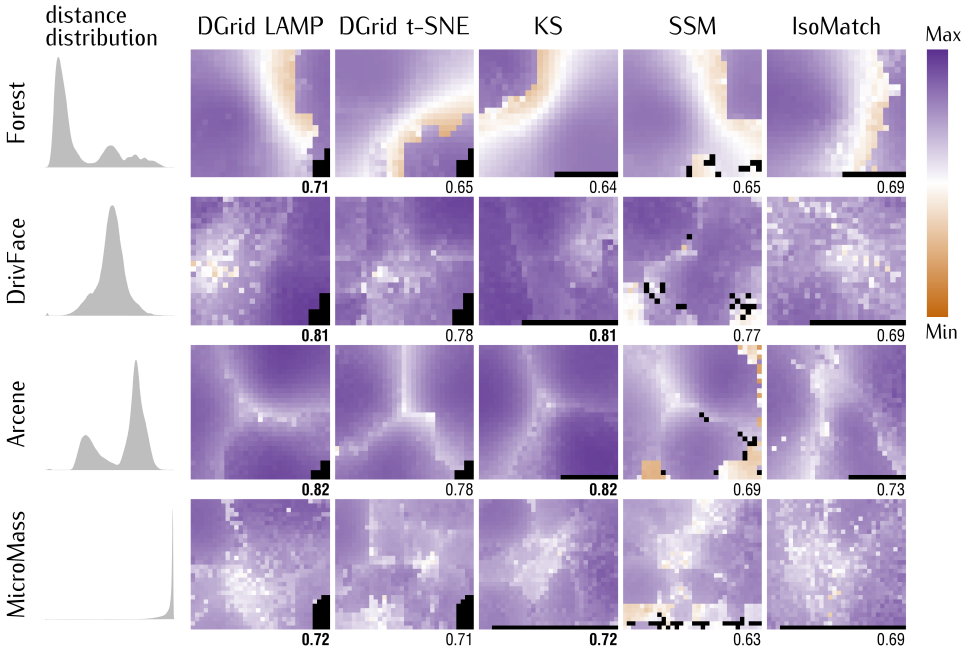


Figure 29 – Resulting grids colored according to the **cross-correlation**. Cross-correlation is global measure, so the use of a global projection technique (LAMP) as input resulted in better grids in that aspect. This renders exceptional flexibility to our approach since it allows selecting a projection technique that fulfills specific geometry properties, generating grids that satisfactorily preserve them.

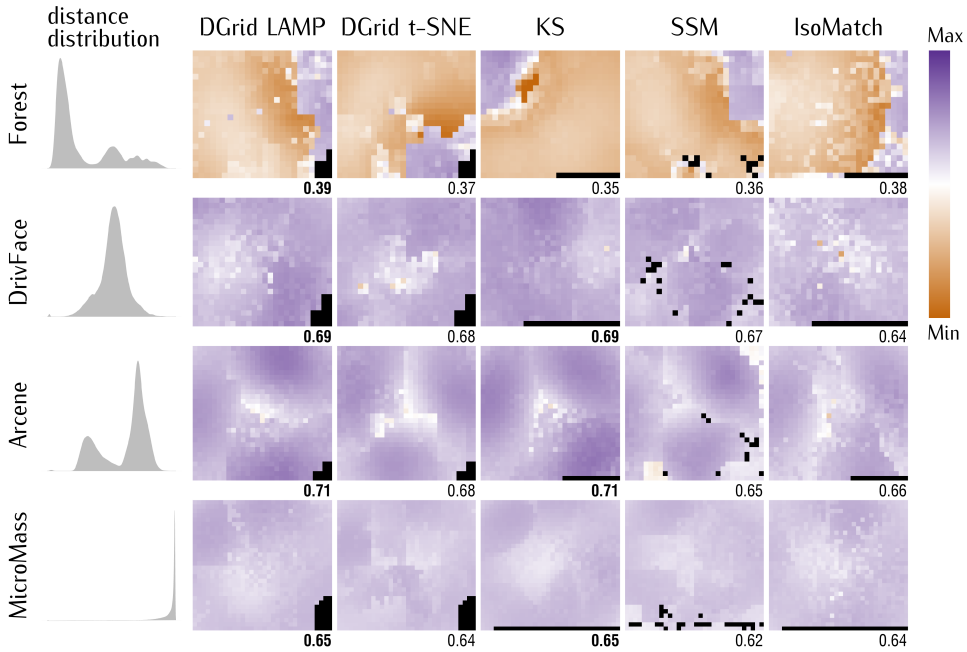


Figure 30 – Resulting grids colored according to the **energy function**. The energy function strong correlates with human performance in search tasks, placing DGrid among the best choices for tasks that involve the analysis of similarity relationships based on grids.

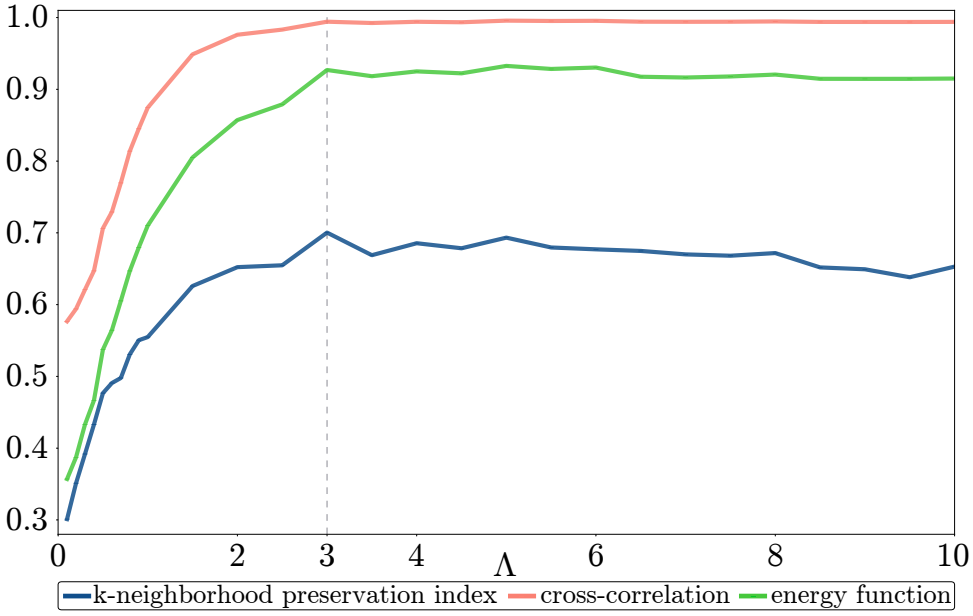


Figure 31 – Impact of varying the grid dimensions to the quality of the produced layouts. The best results are attained when the grid dimensions are related to the distribution of the input projection.

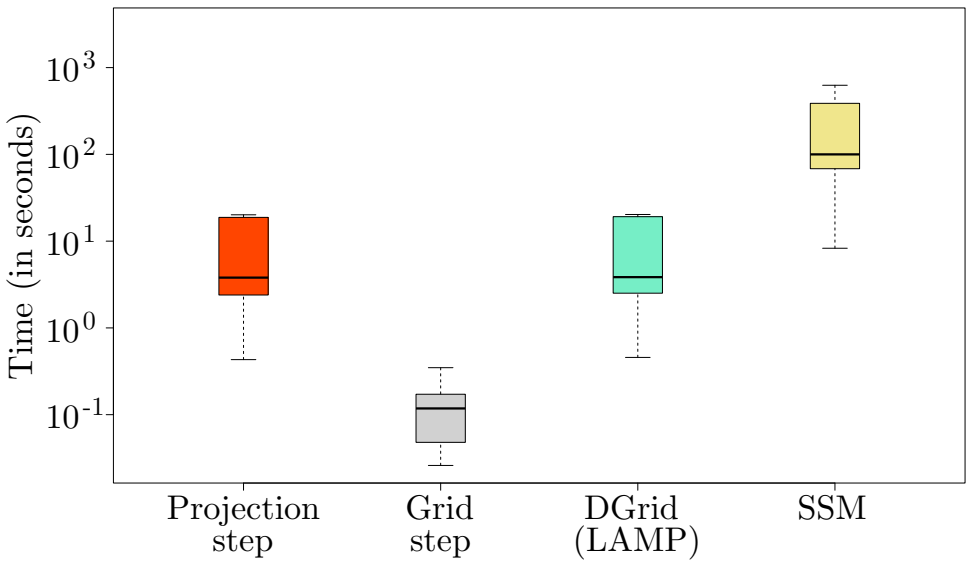
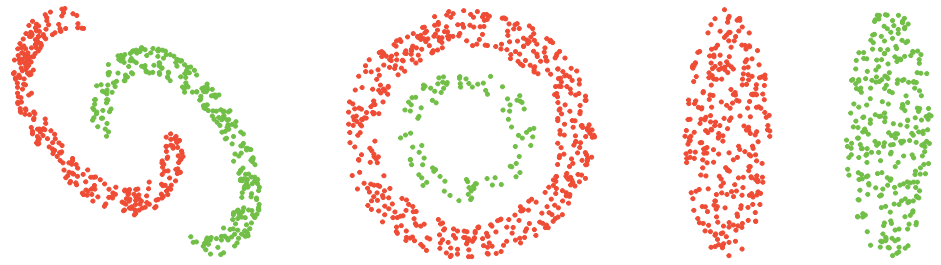


Figure 32 – Running times boxplots. DGrid is almost two orders of magnitude faster than the SSM technique, and the projection phase dominates its running times. We have removed the other technique from this comparison since they are not capable of processing large datasets.



(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

(d) Configuration 4 (e) Configuration 5 (f) Configuration 6

Figure 33 – Initial configurations used to qualitatively evaluate the techniques. These placements were designed to verify different properties of each technique, such as the ability to handle scenarios presenting non-linear relationships, and with different number of groups

Chapter

5

Exploring photo collections using visual feature fusion

Different feature extractors provide different ways to understand and analyze images. Zoologists would like to analyze animals in different settings. For example, a color descriptor could help to group together pandas and zebras because of their black and white color. However, other zoologists can be interested in texture and like to differentiate leopards (round spots) from zebras (black and white stripes). Another animal specialists could be interested in organize animals by their habitat. In that context, object recognition could help to identify objects, such as rivers, caves, and trees. These objects are clues for animal habitats. These three settings can be combined to capture more complex relations such as to identify similar textures with similar habitats, and even include color.

In this chapter, we present our feature fusion strategy using traditional scatterplot and DGrid. We explore photo collections that allow users to control the semantics of the similarity between images among a set of feature extractors and to navigate collections into different levels of detail. Section 5.1 presents a description of the employed dataset. In Section 5.2, we present our visual feature fusion approach. Finally, we summarize this chapter in Section 5.3.

5.1 Dataset

For our application, we use the Photographer dataset (THOMAS; KOVASHKA, 2016), composed of 180,193 photos taken by 40 well-known photographers. The timescale of the photos spans from the early days of photography to the present day. Figure 34 shows example photographs taken by 6 photographers.

We extract 4 different sets of features: LAB color histogram, Gabor filters, HOG and deep-features, which are described in Section 3.2. We named these features as color,

Adams Brumfield Delano Hine Kandell Lange Van Vechten

Figure 34 – Six sample photographs from the **Photographer** dataset taken by Adams, Brumfield, Delano, Hine, Kandell, Lange, and Van Vechten, respectively.

texture, border, and object. From this data set, we can categorize photographers by their most common objects. For example, we can differentiate photographers that like nature than city pictures. Also, we can differentiate black and white than colored pictures via a color descriptor.

Besides that features, we create a set of features to represent the similarity between photographers using an external source. We download Wikipedia articles from each photographer and construct a bag-of-words vector representation. We use bag-of-words in conjunction with TF-IDF. TF-IDF stands for term frequency - inverse document frequency, which reflects how important a word is to document in a collection of corpus and it is widely used in recommender systems (BEEL *et al.*, 2016). We name this feature as photographer. All photos of the same photographer share the same feature vector. Consequently, the similarity among photos is defined as the similarity between texts describing the photographers. Once the bag-of-words are generated, we project these features by LAMP technique using cosine distance. The similarity among photographers is shown in Figure 35 via agglomerative clustering. Eight photographers who are members of the Magnum Photos cooperative are grouped together (see purple cluster), with two non-Magnum photographers in their group. Magnum is a non-profit, photographic foundation located with a mission to expand diversity and creativity in documentary photography. We observe that the two non-magnum photographers (Kourab and Osullivan) take picture of civil war similar to magnum photographers: Cappa and Seymour. This commonality explain why the non-magnum photographers are confused in the magnum group. Furthermore, three photographers that worked for the Farm Security Administration (FSA) are grouped together (see the green cluster), and the two portrait photographers (Van Vechten and Curtis) appear in their own cluster (see the sky blue cluster).

5.2 Visual feature fusion

To visually support the feature fusion, we develop a widget, shown in Figure 36 where anchors represent each feature. Once the features are defined, we extract samples from each set of features as established in Section 3.2.1. We show our overall approach

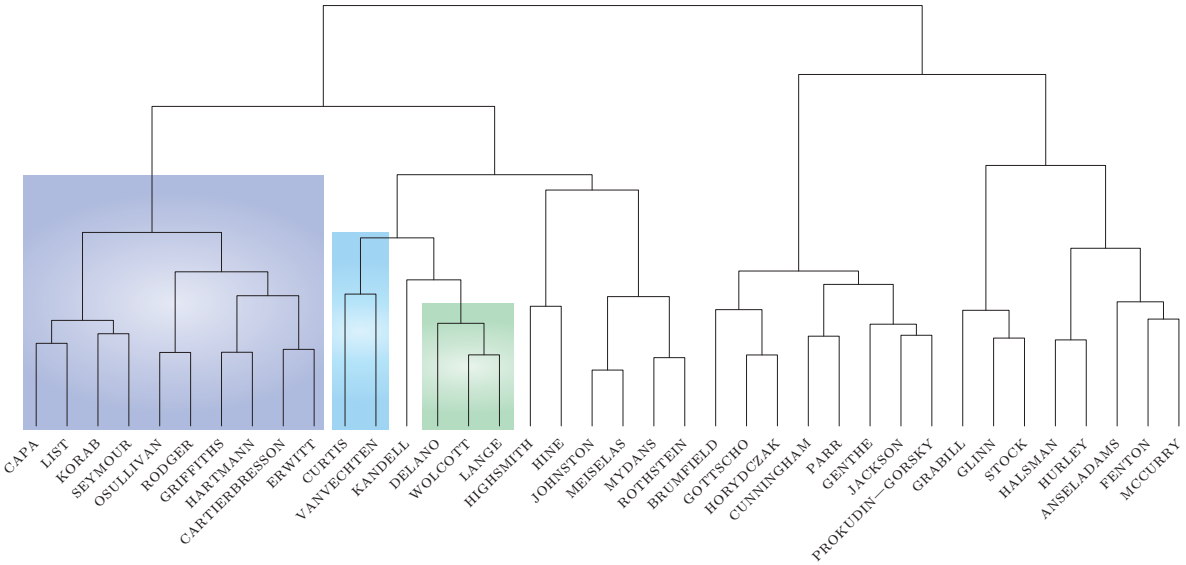


Figure 35 – Similarity among photographers created using Wikipedia articles. Purple cluster grouped members of the Magnum Photos cooperative. Sky blue cluster grouped portrait photographers, and FSA workers are grouped in the green cluster.

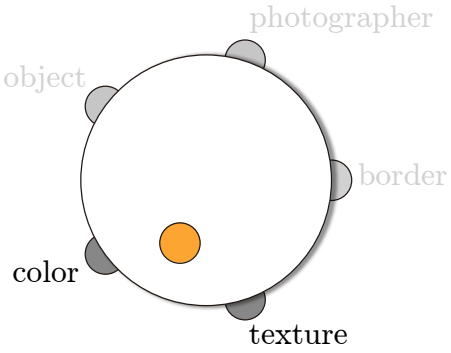


Figure 36 – Feature fusion widget. Features weights are defined based on the closeness of the orange dial to the anchors. So, color and texture feature have higher weight values than the other features.

in Figure 37. First, we get samples using the different set of features. We mainly seek to guarantee that we have images that contain the different traits captured by the different features. After that, projections $p'_{border}, \ldots, p'_{photographer}$ are generated considering the images in the sample, but using the five types of features $f_{border}, \ldots, f_{photographer}$. In the projection process, the features are projected to 2 dimensions for visual purposes as well as aligned to ensure consistency among features using our Algorithm 1 from Section 3.1.1. These projections can be depicted clicking the anchors of the widget. Since the similarity between images is on the eye of the viewer, we allow users to control the semantics of the employed similarity based on weighted combinations of features through manipulating the "orange" dial of the widget. Weights $\alpha_1, \ldots, \alpha_5$ are defined based on the closeness of the dial and the anchors. The feature combination is created as a convex combination of projections, that is, $P' = \alpha_1 p'_{border} + \alpha_2 p'_{texture} + \alpha_3 p'_{color} + \alpha_4 p'_{object} + \alpha_5 p'_{photographer}$. P'

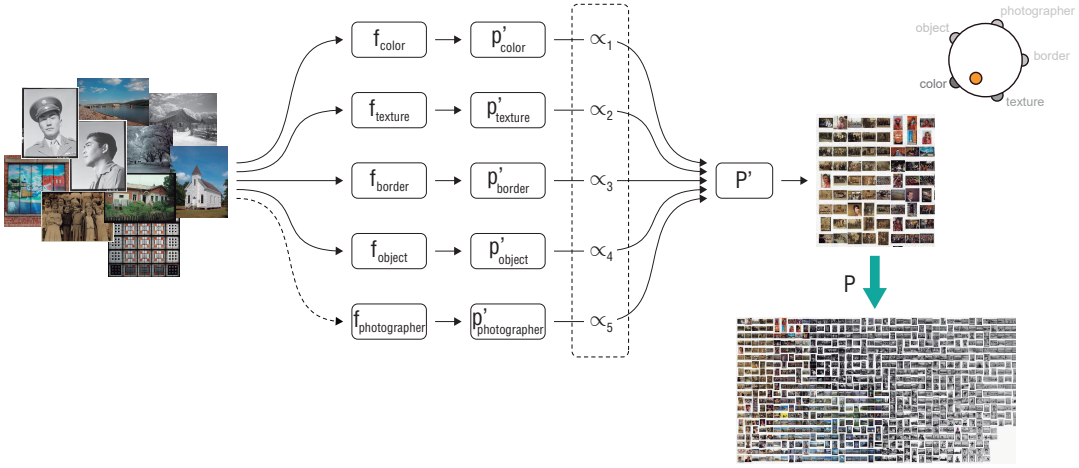


Figure 37 – Overview of our strategy to combine features that allows users to control the semantics of the similarity between images. Based on a small sample P', users can interactively combine different features seeking for the combination that best approaches their point of view regarding similarity. This combination is then propagated to the entire dataset P.

is associated with the sampled dataset combination. By changing $\alpha_1, \ldots, \alpha_5$ users can control the contribution of each different projection to the complete projection, implicitly controlling the importance of each type of feature to the semantics of the employed similarity.

Users can determine a configuration that reflect their semantic understanding updating the weights. Figure 38 shows three different users weight combinations. The first provides more importance to color and objects contained in photos and a little amount of information about photographers. The second combination is defined taking the idea of photographic style from (THOMAS; KOVASHKA, 2016) who find groups of photographers based on shared objects. So, the similarity is mainly defined as objects and photographers. Finally, the third combination shows pattern similarity using texture, borders and a little amount of color. These three combinations reflect different possible semantics of the similarity defined by users.

Once the proper combination has been defined from the users' point of view, a mapping representing the complete photo collection is constructed. It is defined as a combination of projection $p_{border}, \ldots, p_{photographer}$ of the complete dataset, that is, $P = \alpha_1 p_{border} + \alpha_2 p_{texture} + \alpha_3 p_{color} + \alpha_4 p_{object} + \alpha_5 p_{photographer}$. P denotes the whole data set combination and $p_{border}, \ldots, p_{photographer}$ are obtained with our approach from Section 3.1.2.

Figure 39 shows the complete projection using the configuration (a) from Figure 38. In this figure, since color is an important feature, we observe a clear separation between gray and colored images. Also, objects are depicted, we can see a separation between photos of people, landscapes, houses in certain regions of the figure. We zoom a small

portion of the mapping to show that the similarity semantics imposed by the features' combination is preserved. Landscapes in gray are near to color landscapes. On the color side, we observe images with sky and forest. And on the gray side, we observe houses, sky and forest.

Figure 40 shows the mapping using the configuration (b) from Figure 38. We mainly find portrait images together in the zoomed region. This behaviour reflects the photographic style of Van Vechten and Curtis. These two photographers only take pictures of people portraits (THOMAS; KOVASHKA, 2016). These findings are similar to the ones from wikipedia articles, see Figure 35.

Figures 39 and 40 produce scatterplots to display result of our feature combination. We observe that they produce overlapping between images, so suffering from occlusion problems. Thus, a grid-based visualization representing the complete photo collection is constructed using DGrid technique (described in Chapter 4). Considering the dataset size and setting $\Delta = 11/8.5$ to match the aspect ratio of the visual area (paper size), the resulting grid has 482 rows and 374 columns. Since this is too much information to present at once, we allow grid compression. In this process, we convolute the grid with a $R \times S$ mask merging the covered cells into one single cell, thus dividing the number of rows by R and the number of columns by S. In the compact layout, the cells are represented by the closest photo to the center of the $R \times S$ mask. Figures 41 and 42 present the resulting compressed photo grid for configuration (a) and configuration (b) from Figure 38, respectively. In these examples, we use a 5×5 mask, resulting in a grid with 96 rows and 75 columns.

In Figure 41, since color is one of the most important feature in configuration (a), a clear separation between gray and colored images can also be observed. In Figure 42, a close examination reveals that photos of faces, are well separated

The mask $R \times S$ defines the level of detail of the compact representation. Changing its size allows the navigation of the photo collection into different levels of abstraction, from a coarse representation to a more detailed view. Another possibility of navigation is to allow users to select a particular photo, expanding the compressed grid to show all the photos it represents. In this case, by expanding all the cells belonging to the row and column associated with the selected photo, we define a multilevel process that preserves the context. Figure 43 presents another example of grid compression using a 15×15 mask for Figure 38 (a), resulting in a grid with 33 rows and 25 columns. Notice, this compressed grid is also obeying the semantic defined in configuration (a), that is, gray and colored photos are separated. Therein, the photo highlighted in purple is selected to expand the cells of the same row and column. The expanded cells are shown in Figure 44, which are highlighted in blue. Due to the mask size, the expanded grid is increased by 14 rows and 14 columns. We observe that these expanded cells preserve the context. It means the neighboring photos around the highlighted ones have similar meanings (e.g. in relation to

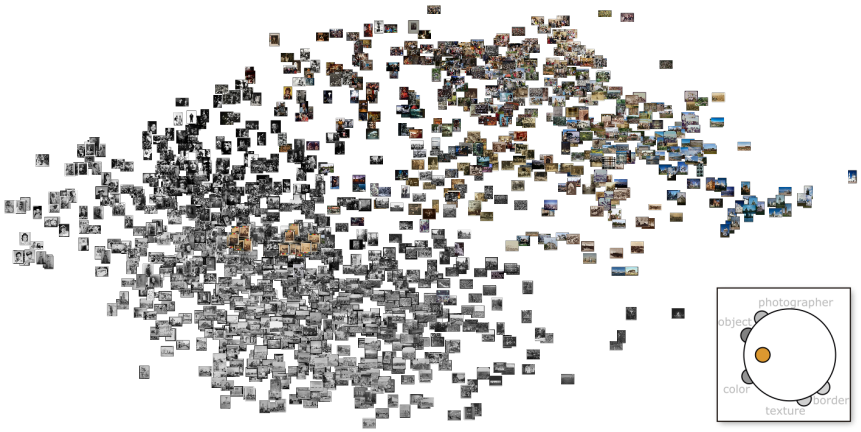
color or object presence)

5.3 Summary

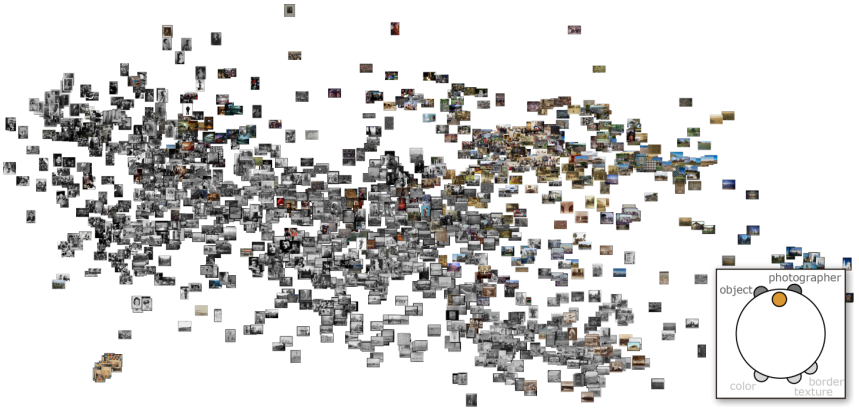
In this chapter, we have presented our visual approach for feature fusion. Since the similarity between images is on the eye of the viewer, allowing users to control the semantics of the employed similarity is a powerful mechanism. This similarity is based on combinations of different features.

We combined projections instead of features to allow a real-time exploration of the different combinations. In this strategy, projections for each set of features are calculated once. Only the visual representation (scatterplot or the grid) is derived when the combination is changed. We prefer grids over scatter plots, because they do not suffer from oclussion problems. Since grids are built from projections almost instantly for reasonable amounts of data, our approach allows the exploration in interactive rates. If we opt to combine features, a projection should be produced after changing a combination, and, currently, there are no good quality projection techniques that are fast enough to attain interactive rates. Notice that, this application is only possible given the high efficiency of DGrid to derive grids in real-time, a trait not found in the current state-of-the-art techniques in distance preserving grid layouts.

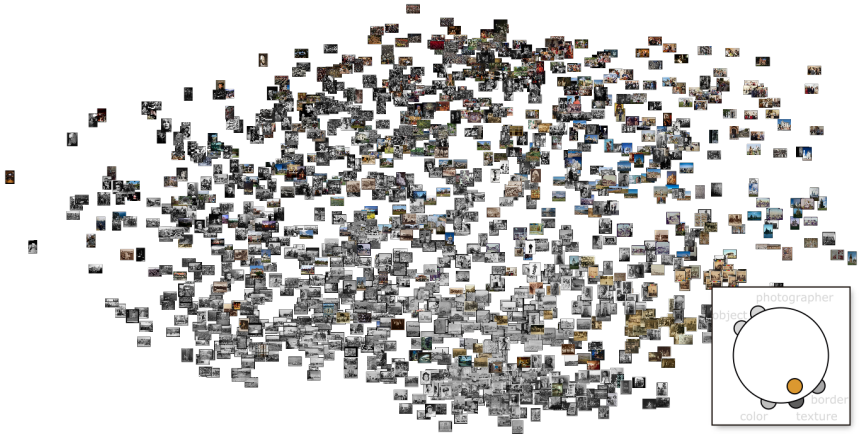
DGrid allows to compress the grid using a mask and allowing users navigate by expanding cells. This process preserves the context. In (FRIED *et al.*, 2015) a similar application was presented. However, they use a hierarchical clustering approach to group the instances, showing representatives of the groups. The user can then navigate by clicking on the representatives, displaying new grids containing the elements (or representatives) inside the groups. Therefore, losing the context whenever a zoom in operation is executed.



(a)



(b)



(c)

Figure 38 – User-defined similarity configurations. Based on a small sample P', users can interactively combine different features for a combination that best approaches their point of view. This combination is then propagated to the entire data set.

Figure 39 – Mapping of the **Photographers** data set for the sample configuration Figure 38 (a) to the whole data set. Since larger weight is assigned to the color feature, a clear global separation between gray and colored photos is observed. This configuration also considers presence of objects and photographer information.

Figure 40 – Mapping of the **Photographers** data set using the sample configuration from Figure 38 (b) to the whole data set. Larger weight is assigned to the object features and photographers. Visual style in photographs are identified. Authors focused on people portraits are highlighted on the zoomed region.

Figure 41 – Photo grid visualization of Figure 39. Still, a clear global separation between gray and colored photos can be observed.

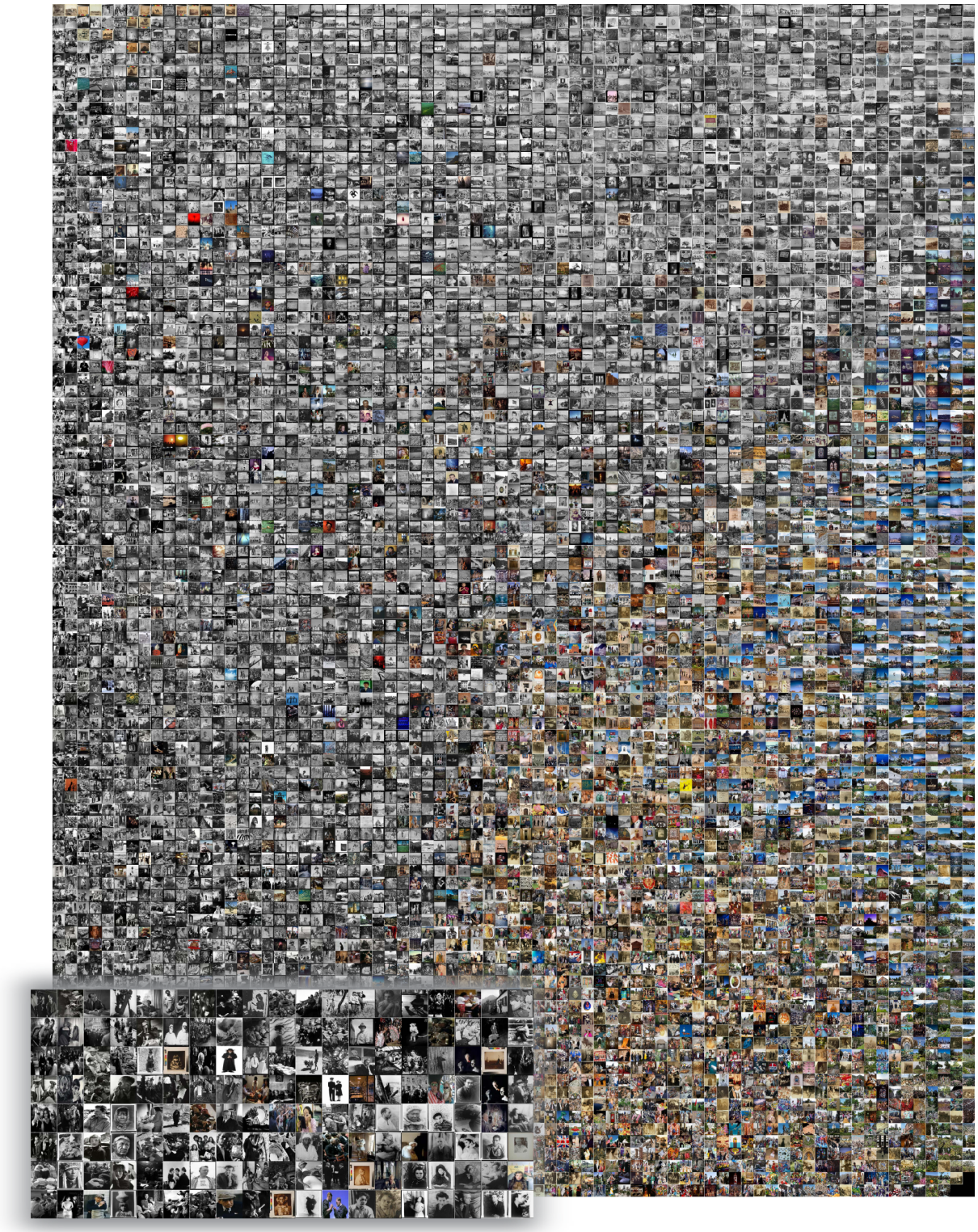


Figure 42 – Photo grid visualization of Figure 40. Still Photographers focused on people portraits are close as noted in the zoom in part of the bottom-left corner.

Figure 43 – Photo grid of the **Photographers** dataset for configuration (a). The grid is compressed using a mask of 15×15 and the purple highlighted photo will be used to expand the grid.

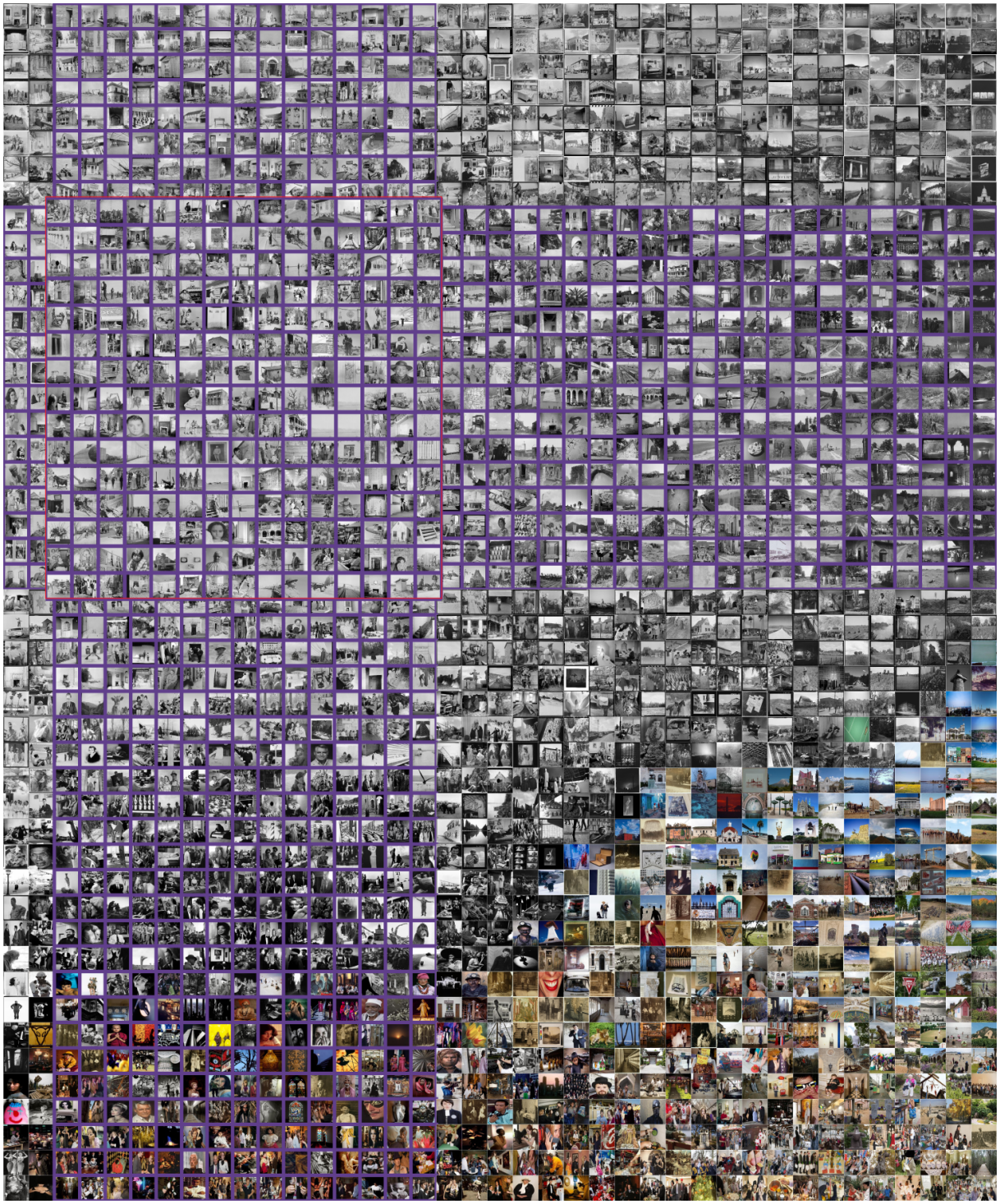


Figure 44 – Expanded version of the compressed grid for the selected photo from Figure 43. Extended cells are highlighted in purple.

Chapter

6

Conclusions and Future Works

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 the features, from which, users define an initial weighted feature combination. Such weights are propagated to combine all the features in the whole data set. Experiments show that the fusion of the complete data set preserves the initial user-defined semantic. To the best of our knowledge this is the first time that visual representations are exploited as a mechanism for feature fusion, characterizing another innovative aspect of this work.

For visualization purposes, we proposed a novel approach for generating grid layouts that preserves distance information, called Distance-preserving Grid (DGrid). DGrid is a two-step approach, which combines a projection technique with an assignment strategy to create orthogonal regular grids. We provide a set of comparisons, which show that DGrid outperforms the existing state-of-the-art techniques considering three quality metrics on 38 data sets from UCI repository. DGrid is two orders faster than existing techniques, and it is scalable for large data sets. The quality of the produced layouts combined with the low computational cost render DGrid as one of the most attractive methods for generating distance preserving grid layouts.

We integrate our approaches into a framework to explore image collections that allows real-time tuning of the semantics of the similarity between images to match users expectations and the navigation of large collections into different levels of detail. Rendering DGrid in conjunction with our feature fusion as a powerful mechanism for browsing and organizing image collections.

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 as shown in Figure 45. We also

Figure 45 – Feature fusion interaction, users can create combinations of different images' features to control the semantics of the employed similarity. The widget in the bottom-right helps on controlling such combination and indicates the importance of each feature. In this case, the similarity mostly reflects the color and a little amount of information about the photographers and less information of objects contained in the photos.

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 (MAMANI *et al.*, 2013).

On the visual metaphors side, scatterplots are a simple and intuitive way of visualizing 2D point data. Scatterplots can display data trends, can allow outlier identification easy because regions with higher density of points will be grouped perceptually, which produce overlapped data instances. DGrid and another grid-based technique 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 do not preserve some data properties, such as data structure. Figure 46 shows a comparison between DGrid and scatterplot metaphors. These layouts are produced using the same dataset. On the scatterplot, we observe a clear separation into two groups (colorful and black-and-white images) with overlapped images. However, that separation is not observed in the grid visualization form 46(b), but it does not have overlapped images. Therefore, a technique that maintains a trade-off between these two metaphors is desirable. A possible solution idea is to deform the grid using distance information to allow separation between groups.

6.1 Limitations

- Regarding the user-guided feature fusion, the first notable drawback is the lack of a study with live users. We believe a this live users study could enhance the evidence

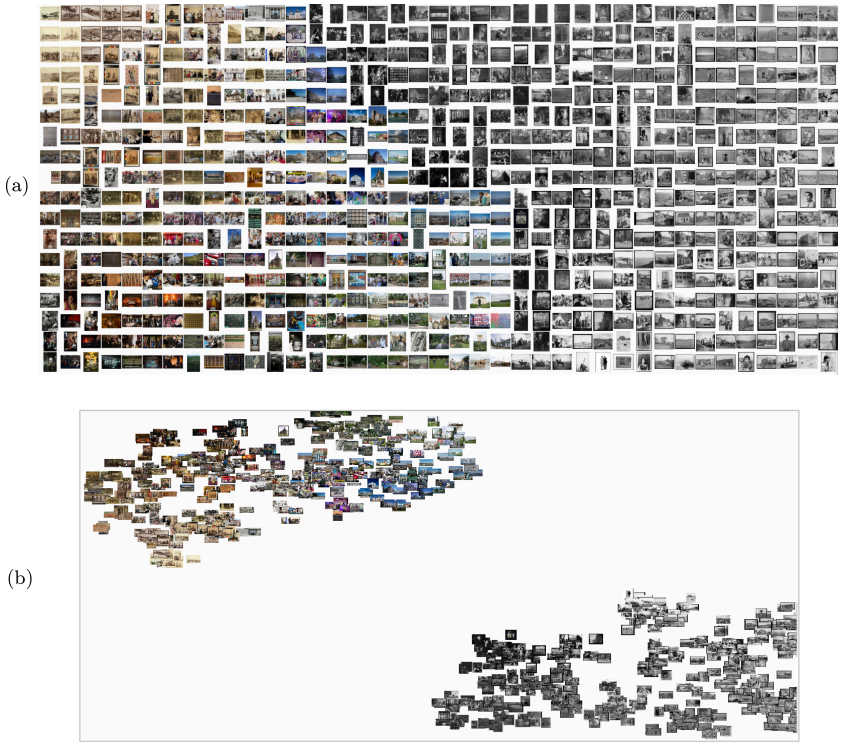


Figure 46 – Comparison of DGrid and scatterplot layouts using color feature from a sample of the Photographer dataset. (a) DGrid layout uses all the available visual space avoiding overlapped images. (b) Traditional scatter plot depicts two separated groups, but with overlapped images.

that visualization is useful in this setting.

- In relation to the DGrid technique, it receives as input the layout generated by any DR technique. However, the selection of the DR is complex because each technique focus on different optimization criterions. For example, t-SNE improves neighborhood preservation and LAMP preserves distance relationships. Therefore, an automatic or semi-automatic procedure to compare and select a meaningful DR technique is required. Also, users should understand deeply the DR techniques to select the best one according to their requirements.

- Another limitation of DGrid is that our strategy always bisects the data in two equal halves. This could hampers the visualization for imbalance datasets, where a small portion of a class is mixed with a more abundant class.

6.2 Future Work

This thesis may lead to new future work, which should be explored and studied. Here, we comprise a set of promising ideas.

- In the initial sampling for feature fusion, we do not perform an experiment of an

adequate sampling size. We believe this study is required because for very large samples, users can not extract meaningful knowledge and relations. Also, we experiment with only one sample configuration, which is restrictive. An idea to improve this limitation is considering many complementary samples and find a way to combine them (similar to ensemble learning).

- DGrid makes the bisection cutting the number of rows or columns in half. However, for imbalance datasets, this could present a problem because a small portion of a class is mixed with another class. This distorts the grid-layout structure and can generate undesirable visualization. An idea to solve this limitation is finding the best way to split the data according to their data distribution.

- The input to DGrid is any 2D layout, which is generated by DR techniques. However, each technique has their own optimization objective as stated before and varies in terms of layout and quality. Thus, it is not clear how to select a suitable DR. Also, most DR algorithms have additional parameters (such as a neighborhood size) that further affect the results, which makes the task more complex. It is interesting to develop a tool to find a balance between different optimization measures and follow the user's requirements.

- We can use our grid-based technique to analyze machine learning methods. For example, usually clustering results are visualized into scatter plots, we believe grid-based technique could present better these results because they do not suffer from occlusion problems. However, some adaptations should be made to show separation among groups. This could be achieved by calculating distances to neighbor instances, and these distances will be encoded by transparency. Another option is to separate grid cells according to the group information.

- Finally, one of the most appealing application scenarios for our feature fusion approach is to assist non-supervised strategies, such as clustering. Clustering is a subjective task that depends on how similarity is computed, and the ability to explicitly control and understand similarity is one benefit of our approach. Another application could be in text domain. A traditional text visualization technique is Word Cloud. It arranges the words randomly. Although they are useful and informative tools, the randomness of word layout does not provide a meaningful representation of the data. Furthermore, it only provides information about word frequency, without semantic relationships among words, which is critical for understanding the context. DGrid could be an option to integrate semantic context of words into the visualization layout. Both mentioned application scenarios have been explored and developed.

Bibliography

ANNE, K. R.; KUCHIBHOTLA, S.; VANKAYALAPATI, H. D. **Acoustic Modeling for Emotion Recognition**. [S.l.]: Springer Publishing Company, Incorporated, 2015. ISBN 3319155296, 9783319155296. Citation on page 43.

BEEL, J.; GIPP, B.; LANGER, S.; BREITINGER, C. Research-paper recommender systems: A literature survey. **Int. J. Digit. Libr.**, Springer-Verlag, Berlin, Heidelberg, v. 17, n. 4, p. 305–338, Nov. 2016. ISSN 1432-5012. Citation on page 92.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Commun. ACM**, ACM, New York, NY, USA, v. 18, n. 9, p. 509–517, Sep. 1975. ISSN 0001-0782. Citation on page 83.

BISHOP, C. M.; SVENSéN, M.; WILLIAMS, C. K. I. Gtm: The generative topographic mapping. **Neural Computation**, v. 10, n. 1, p. 215–234, Jan 1998. ISSN 0899-7667. Citation on page 53.

BORG, I.; GROENEN, P. **Modern Multidimensional Scaling: Theory and Applications**. [S.l.]: Springer, 2005. Citation on page 49.

BOSTROM, H.; ANDLER, S. F.; BROHEDE, M.; JOHANSSON, R.; KARLSSON, E.; LAERE, J. V.; NIKLASSON, L.; NILSSON, M.; PERSSON, A.; ZIEMKE, T. On the definition of information fusion as a field of research. 2007. Citation on page 42.

BROWN, E. T.; LIU, J.; BRODLEY, C. E.; CHANG, R. Dis-function: Learning distance functions interactively. In: **2012 IEEE Conference on Visual Analytics Science and Technology (VAST)**. [S.l.: s.n.], 2012. p. 83–92. Citations on pages 13, 38, and 40.

BROWN, L. G. A survey of image registration techniques. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 24, n. 4, p. 325–376, Dec. 1992. ISSN 0360-0300. Available: <http://doi.acm.org/10.1145/146370.146374>. Citation on page 30.

CAO, N.; LIN, Y. R.; GOTZ, D. Untangle map: Visual analysis of probabilistic multi-label data. **IEEE Transactions on Visualization and Computer Graphics**, v. 22, n. 2, p. 1149–1163, Feb 2016. ISSN 1077-2626. Citation on page 52.

CHEN, L.; LU, G.; ZHANG, D. Effects of different gabor filter parameters on image retrieval by texture. In: **Proceedings of the 10th International Multimedia Modelling Conference**. Washington, DC, USA: IEEE Computer Society, 2004. (MMM '04), p. 273–. ISBN 0-7695-2084-7. Citation on page 66.

CHOO, J.; LEE, H.; KIHM, J.; PARK, H. ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In: **2010 IEEE Symposium on Visual Analytics Science and Technology**. [S.l.: s.n.], 2010. p. 27–34. Citation on page 29.

CHU, J.; GUO, Z.; LENG, L. Object detection based on multi-layer convolution feature fusion and online hard example mining. **IEEE Access**, p. 1–1, 2018. Citations on pages 14, 43, and 44.

COATES, A.; NG, A. Y.; LEE, H. 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**. [S.l.: s.n.], 2011. p. 215–223. Citation on page 65.

CUI, Q.; WARD, M.; RUNDENSTEINER, E.; YANG, J. Measuring data abstraction quality in multiresolution visualizations. **IEEE Transactions on Visualization and Computer Graphics**, v. 12, n. 5, p. 709–716, Sept 2006. Citation on page 68.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: **In CVPR**. [S.l.: s.n.], 2005. p. 886–893. Citation on page 66.

DIETTERICH, T. G. Ensemble methods in machine learning. In: **Proceedings of the First International Workshop on Multiple Classifier Systems**. London, UK, UK: Springer-Verlag, 2000. (MCS '00), p. 1–15. ISBN 3-540-67704-6. Available: <http://dl.acm.org/citation.cfm?id=648054.743935>. Citation on page 47.

DOLLÁR, P.; WOJEK, C.; SCHIELE, B.; PERONA, P. Pedestrian detection: An evaluation of the state of the art. **IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)**, v. 34, n. 4, p. 743–761, 2012. Citation on page 43.

DOWLING, M.; WENSKOVITCH, J.; HAUCK, P.; BINFORD, A.; LONG, T.; POLYS, N.; NORTH, C. Construction and usage of the semantic interaction pipeline. 2018. Citation on page 37.

ENDERT, A. Semantic interaction for visual analytics: Toward coupling cognition and computation. **IEEE Computer Graphics and Applications**, v. 34, n. 4, p. 8–15, 2014. Available: <https://doi.org/10.1109/MCG.2014.73>. Citation on page 35.

ENDERT, A.; FIAUX, P.; NORTH, C. Semantic interaction for visual text analytics. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. New York, NY, USA: ACM, 2012. (CHI '12), p. 473–482. ISBN 978-1-4503-1015-4. Available: <http://doi.acm.org/10.1145/2207676.2207741>. Citation on page 37.

FADEL, S. G.; FATORE, F. M.; DUARTE, F. S.; PAULOVICH, F. V. Loch: A neighborhood-based multidimensional projection technique for high-dimensional sparse spaces. **Neurocomputing**, v. 150, n. Part B, p. 546 – 556, 2015. Citation on page 78.

FALOUTSOS, C.; LIN, K.-I. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. **SIGMOD Rec.**, ACM, New York, NY, USA, v. 24, n. 2, p. 163–174, May 1995. ISSN 0163-5808. Citation on page 63.

FRANCE, S. L.; CARROLL, J. D. Two-way multidimensional scaling: A review. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 41, n. 5, p. 644–661, Sept 2011. ISSN 1094-6977. Citation on page 49.

FRIED, O.; DIVERDI, S.; HALBER, M.; SIZIKOVA, E.; FINKELSTEIN, A. Isomatch: Creating informative grid layouts. **Comput. Graph. Forum**, The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, v. 34, n. 2, p. 155–166, May 2015. ISSN 0167-7055. Citations on pages 31, 53, 55, 78, 80, 81, and 96.

GOMEZ-NIETO, E.; ROMAN, F. S.; PAGLIOSA, P.; CASACA, W.; HELOU, E. S.; OLIVEIRA, M. C. F. de; NONATO, L. G. Similarity preserving snippet-based visualization of web search results. **IEEE Transactions on Visualization and Computer Graphics**, v. 20, n. 3, p. 457–470, March 2014. ISSN 1077-2626. Citations on pages 31, 52, and 73.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. Digital image processing using matlab. v. 1, 01 2004. Citation on page 43.

HUANG, Z. C.; CHAN, P. P. K.; NG, W. W. Y.; YEUNG, D. S. Content-based image retrieval using color moment and gabor texture feature. In: **2010 International Conference on Machine Learning and Cybernetics**. [S.l.: s.n.], 2010. v. 2, p. 719–724. ISSN 2160-133X. Citation on page 46.

JEONG, D. H.; ZIEMKIEWICZ, C.; FISHER, B. D.; RIBARSKY, W.; CHANG, R. ipca: An interactive system for pca-based visual analytics. **Comput. Graph. Forum**, v. 28, p. 767–774, 2009. Citations on pages 29, 39, and 42.

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. In: **Proceedings of the 22Nd ACM International Conference on Multimedia**. New York, NY, USA: ACM, 2014. (MM '14), p. 675–678. ISBN 978-1-4503-3063-3. Available: <http://doi.acm.org/10.1145/2647868.2654889>. Citation on page 66.

JOIA, P.; COIMBRA, D.; CUMINATO, J. A.; PAULOVICH, F. V.; NONATO, L. G. Local affine multidimensional projection. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 17, n. 12, p. 2563–2571, Dec. 2011. ISSN 1077-2626. Citations on pages 14, 51, 52, 73, and 77.

_____. Local affine multidimensional projection. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 17, n. 12, p. 2563–2571, Dec. 2011. ISSN 1077-2626. Available: <http://dx.doi.org/10.1109/TVCG.2011.220>. Citations on pages 40, 49, 63, and 64.

_____. Local affine multidimensional projection. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 17, p. 2563–2571, Dec. 2011. ISSN 1077-2626. Citation on page 48.

Jolliffe, I. **Principal component analysis**. New York: Springer Verlag, 2002. Citation on page 49.

KEIM, D. A.; KOHLHAMMER, J.; ELLIS, G. P.; MANSMANN, F. **Mastering the Information Age - Solving Problems with Visual Analytics.** [S.l.]: Eurographics Association, 2010. 1-168 p. ISBN 978-3-905673-77-7. Citations on pages 13, 29, 35, and 36.

KIM, K.; LIN, H.; CHOI, J. Y.; CHOI, K. A design framework for hierarchical ensemble of multiple feature extractors and multiple classifiers. **Pattern Recogn.**, Elsevier Science Inc., New York, NY, USA, v. 52, n. C, p. 1–16, Apr. 2016. ISSN 0031-3203. Available: <http://dx.doi.org/10.1016/j.patcog.2015.11.006>. Citations on pages 46 and 47.

KOHONEN, T. Neurocomputing: Foundations of research. In: ANDERSON, J. A.; ROSENFELD, E. (Ed.). Cambridge, MA, USA: MIT Press, 1988. chap. Self-organized Formation of Topologically Correct Feature Maps, p. 509–521. ISBN 0-262-01097-6. Citation on page 53.

KRIZHEVSKY, A. **Learning multiple layers of features from tiny images**. [S.l.], 2009. Citation on page 65.

KUANG, H.; CHAN, L. L.; LIU, C.; YAN, H. Fruit classification based on weighted score-level feature fusion. **J. Electronic Imaging**, v. 25, n. 1, p. 013009, 2016. Available: <https://doi.org/10.1117/1.JEI.25.1.013009>. Citation on page 47.

KUHN, H. W. The hungarian method for the assignment problem. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 2, n. 1-2, p. 83–97, 1955. ISSN 1931-9193. Available: <http://dx.doi.org/10.1002/nav.3800020109>. Citation on page 55.

LAMPERT, C.; NICKISCH, H.; HARMELING, S. Learning to detect unseen object classes by between-class attribute transfer. In: MAX-PLANCK-GESELLSCHAFT. **CVPR 2009**. Piscataway, NJ, USA: IEEE Service Center, 2009. p. 951–958. Citation on page 65.

LEVINA, E.; BICKEL, P. J. Maximum likelihood estimation of intrinsic dimension. In: **Proceedings of the 17th International Conference on Neural Information Processing Systems**. Cambridge, MA, USA: MIT Press, 2004. (NIPS'04), p. 777–784. Available: <http://dl.acm.org/citation.cfm?id=2976040.2976138>. Citation on page 63.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Available: <http://archive.ics.uci.edu/ml>. Citations on pages 48 and 78.

LIU, P.; GUO, J. M.; WU, C. Y.; CAI, D. Fusion of deep learning and compressed domain features for content-based image retrieval. **IEEE Transactions on Image Processing**, v. 26, n. 12, p. 5706–5717, Dec 2017. ISSN 1057-7149. Citations on pages 46 and 59.

LONI, B.; KHOSHNEVIS, S. H.; WIGGERS, P. Latent semantic analysis for question classification with neural networks. In: **2011 IEEE Workshop on Automatic Speech Recognition Understanding**. [S.l.: s.n.], 2011. p. 437–442. Citation on page 43.

LONI, B.; TULDER, G. V.; WIGGERS, P.; TAX, D. M. J.; LOOG, M. Question classification by weighted combination of lexical, syntactic and semantic features. In: **Proceedings of the 14th International Conference on Text, Speech and Dialogue**. Berlin, Heidelberg: Springer-Verlag, 2011. (TSD 11), p. 243–250. ISBN 978-3-642-23537-5. Available: <http://dl.acm.org/citation.cfm?id=2040037.2040070>. Citation on page 44.

LOWE, D. G. Object recognition from local scale-invariant features. In: **Proc. of the International Conference on Computer Vision ICCV, Corfu**. [S.l.: s.n.], 1999. Citation on page 43.

MA, G.; YANG, X.; ZHANG, B.; SHI, Z. Multi-feature fusion deep networks. **Neurocomput.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 218, n. C, p. 164–171, Dec. 2016. ISSN 0925-2312. Available: <https://doi.org/10.1016/j.neucom.2016.08.059>. Citations on pages 30, 44, and 59.

MAATEN, L. V. D.; POSTMA, E.; HERIK, J. Van den. Dimensionality reduction: a comparative review. **J Mach Learn Res**, v. 10, p. 66–71, 2009. Citation on page 49.

MAATEN, L. van der; HINTON, G. Visualizing High-Dimensional Data Using t-SNE. **Journal of Machine Learning Research**, v. 9, p. 2579–2605, Nov. 2008. Citation on page 49.

_____. Visualizing High-Dimensional Data Using t-SNE. **Journal of Machine Learning Research**, v. 9, p. 2579–2605, Nov. 2008. Citation on page 77.

MAMANI, G. M. H.; FATORE, F. M.; NONATO, L. G.; PAULOVICH, F. V. User-driven Feature Space Transformation. **Computer Graphics Forum**, The Eurographics Association and Blackwell Publishing Ltd., 2013. ISSN 1467-8659. Citations on pages 14, 40, 41, 42, and 106.

MANGAI, U. G.; SAMANTA, S.; DAS, S.; CHOWDHURY, P. R. A survey of decision fusion and feature fusion strategies for pattern classification. **IETE Technical Review**, Taylor Francis, v. 27, n. 4, p. 293–307, 2010. Citations on pages 30, 42, 43, 44, and 45.

MANSHOR, N.; RAHIMAN, A. R.; MANDAVA, R.; RAMACHANDRAM, D. Feature fusion in improving object class recognition. v. 8, p. 1321–1328, 01 2012. Citations on pages 14, 43, and 44.

MENDES-MOREIRA, J. a.; SOARES, C.; JORGE, A. M.; SOUSA, J. F. D. Ensemble approaches for regression: A survey. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 45, n. 1, p. 10:1–10:40, Dec. 2012. ISSN 0360-0300. Available: <http://doi.acm.org/10.1145/2379776.2379786>. Citation on page 47.

MOLCHANOV, V.; LINSEN, L. Interactive Design of Multidimensional Data Projection Layout. In: ELMQVIST, N.; HLAWITSCHKA, M.; KENNEDY, J. (Ed.). **EuroVis - Short Papers**. [S.l.]: The Eurographics Association, 2014. ISBN 978-3-905674-69-9. Citations on pages 14, 40, 41, and 42.

NONATO, L. G.; AUPETIT, M. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. **IEEE Transactions on Visualization and Computer Graphics**, p. 1–1, 2018. ISSN 1077-2626. Citations on pages 48 and 49.

NONATO, L. G.; SILVA, C. T.; PAULOVICH, F. V. User-centered multidimensional projection techniques. **Computing in Science and Engineering**, v. 14, p. 74–81, 07 2012. ISSN 1521-9615. Available: <doi.ieeecomputersociety.org/10.1109/MCSE.2012.85>. Citations on pages 49 and 52.

PAGLIOSA, P.; PAULOVICH, F. V.; MINGHIM, R.; LEVKOWITZ, H.; NONATO, L. G. Projection inspector: Assessment and synthesis of multidimensional projections. **Neurocomputing**, v. 150, Part B, p. 599–610, 2015. ISSN 0925-2312. Citation on page 64.

PAL, N. R.; BEZDEK, J. C. On cluster validity for the fuzzy c-means model. **IEEE Transactions on Fuzzy Systems**, v. 3, n. 3, p. 370–379, Aug 1995. ISSN 1063-6706. Citation on page 61.

PAULOVICH, F.; NONATO, L.; MINGHIM, R.; LEVKOWITZ, H. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. v. 14, n. 3, p. 564–575, 2008. ISSN 1077-2626. Citation on page 50.

PAULOVICH, F.; SILVA, C.; NONATO, L. Two-phase mapping for projecting massive data sets. v. 16, n. 6, p. 1281–1290, 2010. ISSN 1077-2626. Citations on pages 14, 49, 50, and 52.

PAULOVICH, F. V.; ELER, D. M.; POCO, J.; BOTHA, C. P.; MINGHIM, R.; NONATO, L. G. Piecewise Laplacian-based Projection for Interactive Data Exploration and Organization. **Computer Graphics Forum**, The Eurographics Association and Blackwell Publishing Ltd., 2011. ISSN 1467-8659. Citations on pages 14, 50, and 52.

PAULOVICH, F. V.; SILVA, C. T.; NONATO, L. G. Two-phase mapping for projecting massive data sets. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 16, n. 6, p. 1281–1290, Nov. 2010. ISSN 1077-2626. Available: <http://dx.doi.org/10.1109/TVCG.2010.207>. Citations on pages 77 and 85.

PINHO, R. D.; OLIVEIRA, M. C.; LOPES, A. A. An incremental space to visualize dynamic data sets. **Multimedia Tools Appl.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 50, n. 3, p. 533–562, Dec. 2010. ISSN 1380-7501. Citations on pages 31, 52, and 73.

QUADRIANTO, N.; SMOLA, A. J.; SONG, L.; TUYTELAARS, T. Kernelized sorting. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 10, p. 1809–1821, Oct 2010. ISSN 0162-8828. Citations on pages 31, 53, 55, and 77.

SACHA, D.; SEDLMAIR, M.; ZHANG, L.; LEE, J. A.; PELTONEN, J.; WEISKOPF, D.; NORTH, S. C.; KEIM, D. A. What you see is what you can change: Human-centered machine learning by interactive visualization. **Neurocomputing**, v. 268, p. 164–175, 2017. Available: <https://doi.org/10.1016/j.neucom.2017.01.105>. Citations on pages 13, 36, and 41.

SACHA, D.; STOFFEL, A.; STOFFEL, F.; KWON, B. C.; ELLIS, G.; KEIM, D. Knowledge generation model for visual analytics. **IEEE Transactions on Visualization and Computer Graphics**, v. 20, n. 12, p. 1604–1613, 2014. Citation on page 29.

SACHA, D.; ZHANG, L.; SEDLMAIR, M.; LEE, J. A.; PELTONEN, J.; WEISKOPF, D.; NORTH, S. C.; KEIM, D. A. Visual interaction with dimensionality reduction: A structured literature analysis. **IEEE Transactions on Visualization and Computer Graphics**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 23, n. 1, p. 241–250, Jan. 2017. ISSN 1077-2626. Available: <https://doi.org/10.1109/TVCG.2016.2598495>. Citation on page 29.

SCHNEIDER, B.; JACKLE, D.; STOFFEL, F.; DIEHL, A.; FUCHS, J.; KEIM, D. Visual Integration of Data and Model Space in Ensemble Learning. In: **Symposium on Visualization in Data Science (VDS) at IEEE VIS 2017 (BEST PAPER award)**. [S.l.: s.n.], 2017. Citations on pages 13, 37, 39, 41, 42, and 47.

SEDLMAIR, M.; MUNZNER, T.; TORY, M. Empirical guidance on scatterplot and dimension reduction technique choices. **IEEE Transactions on Visualization and Computer Graphics**, v. 19, n. 12, p. 2634–2643, Dec 2013. ISSN 1077-2626. Citation on page 48.

SILVA, V. D.; TENENBAUM, J. B. Global versus local methods in nonlinear dimensionality reduction. In: **Advances in Neural Information Processing Systems 15**. [S.l.]: MIT Press, 2003. p. 705–712. Citation on page 49.

STROBELT, H.; SPICKER, M.; STOFFEL, A.; KEIM, D.; DEUSSEN, O. Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. **Computer Graphics Forum**, Blackwell Publishing Ltd, v. 31, n. 3pt3, p. 1135–1144, 2012. ISSN 1467-8659. Citations on pages 31, 52, and 73.

STRONG, G.; GONG, M. Data organization and visualization using self-sorting map. In: **Proceedings of Graphics Interface 2011**. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2011. (GI '11), p. 199–206. ISBN 978-1-4503-0693-5. Citations on pages 31 and 53.

_____. Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts. **Trans. Multi.**, IEEE Press, Piscataway, NJ, USA, v. 16, n. 4, p. 1045–1058, Jun. 2014. ISSN 1520-9210. Citations on pages 31, 53, 54, 77, 78, and 84.

SUDHA, D.; RAMAKRISHNA, M. Comparative study of features fusion techniques. **2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)**, p. 235–239, 2017. Citations on pages 42 and 43.

TALBOT, J.; LEE, B.; KAPOOR, A.; TAN, D. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In: **ACM Human Factors in Computing Systems (CHI)**. [s.n.], 2009. Available: <http://vis.stanford.edu/papers/ensemblematrix>. Citation on page 37.

TAM, G. K. L.; KOTHARI, V.; CHEN, M. An analysis of machine- and human-analytics in classification. **IEEE Transactions on Visualization and Computer Graphics**, v. 23, n. 1, p. 71–80, Jan 2017. ISSN 1077-2626. Citation on page 35.

TEJADA, E.; MINGHIM, R.; NONATO, L. G. On improved projection techniques to support visual exploration of multidimensional data sets. **Information Visualization**, v. 2, n. 4, p. 218–231, 2003. Citations on pages 40 and 51.

TELEA, A. C. **Data Visualization: Principles and Practice, Second Edition**. 2nd. ed. Natick, MA, USA: A. K. Peters, Ltd., 2014. ISBN 1466585269, 9781466585263. Citation on page 29.

TENENBAUM, J. B.; SILVA, V. d.; LANGFORD, J. C. A global geometric framework for nonlinear dimensionality reduction. **Science**, American Association for the Advancement of Science, v. 290, n. 5500, p. 2319–2323, 2000. Citation on page 55.

THOMAS, C.; KOVASHKA, A. Seeing behind the camera: Identifying the authorship of a photograph. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. Citations on pages 65, 91, 94, and 95.

TORGERSON, W. S. Multidimensional scaling of similarity. **Psychometrika**, v. 30, n. 4, p. 379–393, Dec 1965. ISSN 1860-0980. Available: <https://doi.org/10.1007/BF02289530>. Citation on page 77.

VADIVEL, A.; MAJUMDAR, A. K.; SURAL, S. Characteristics of weighted feature vector in content-based image retrieval applications. In: **International Conference on Intelligent Sensing and Information Processing, 2004. Proceedings of**. [S.l.: s.n.], 2004. p. 127–132. Citation on page 46.

WANG, X.; HAN, T. X.; YAN, S. An hog-lbp human detector with partial occlusion handling. **2009 IEEE 12th International Conference on Computer Vision**, p. 32–39, 2009. Citations on pages 14, 43, 44, and 59.

WANG, X.-M.; ZHANG, T.-Y.; MA, Y.-X.; XIA, J.; CHEN, W. A survey of visual analytic pipelines. **Journal of Computer Science and Technology**, v. 31, n. 4, p. 787–804, Jul 2016. ISSN 1860-4749. Citation on page 35.

WANG, Y.; ZHANG, W.; WU, L.; LIN, X.; ZHAO, X. Unsupervised metric fusion over multiview data by graph random walk-based cross-view diffusion. **IEEE Transactions on Neural Networks and Learning Systems**, v. 28, n. 1, p. 57–70, Jan 2017. ISSN 2162-237X. Citation on page 59.

WEISS, Y.; TORRALBA, A.; FERGUS, R. Spectral hashing. In: KOLLER, D.; SCHURMANS, D.; BENGIO, Y.; BOTTOU, L. (Ed.). **Advances in Neural Information Processing Systems 21**. Curran Associates, Inc., 2009. p. 1753–1760. Available: <http://papers.nips.cc/paper/3383-spectral-hashing.pdf>. Citation on page 53.

WONIAK, M.; nA, M. G.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. **Inf. Fusion**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 16, p. 3–17, Mar. 2014. ISSN 1566-2535. Available: <http://dx.doi.org/10.1016/j.inffus.2013.04.006>. Citation on page 47.

YOU, T.; TANG, Y. Visual saliency detection based on adaptive fusion of color and texture features. In: **2017 3rd IEEE International Conference on Computer and Communications (ICCC)**. [S.l.: s.n.], 2017. p. 2034–2039. Citation on page 44.

YU, A.; GRAUMAN, K. Fine-grained visual comparisons with local learning. In: **Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2014. Citation on page 65.

YU, W.; ZHU, Q. Quick retrieval method of massive face images based on global feature and local feature fusion. In: **2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)**. [S.l.: s.n.], 2017. p. 1–6. Citation on page 45.

ZHAO, J.; XIE, X.; XU, X.; SUN, S. Multi-view learning overview: Recent progress and new challenges. **Information Fusion**, v. 38, p. 43 – 5, 2017. ISSN 1566-2535. Available: <http://www.sciencedirect.com/science/article/pii/S1566253516302032>. Citations on pages 42 and 59.

ZITOVA, B.; FLUSSER, J. Image registration methods: a survey. **Image and Vision Computing**, v. 21, n. 11, p. 977 – 1000, 2003. ISSN 0262-8856. Available: <http://www.sciencedirect.com/science/article/pii/S0262885603001379>. Citation on page 30.


ICMC
USP
SÃO CARLOS