Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

Visual Analysis of Sets of Heterogeneous Matrices Using Projection-Based Distance Functions and Semantic Zoom

Michael Behrisch[1], James Davey[2], Fabian Fischer[1], Olivier Thonnard[3], Tobias Schreck[1], Daniel Keim[1], Jörn Kohlhammer[4]

[1]Universität Konstanz, Germany
[2]Fraunhofer IGD, Germany
[3]Symantec Research Labs, France
[4]Technische Universität Darmstadt, Germany

Abstract

Matrix visualization is an established technique in the analysis of relational data. It is applicable to large, dense networks, where node-link representations may not be effective. Recently, domains have emerged in which the comparative analysis of sets of matrices of potentially varying size is relevant. For example, to monitor computer network traffic a dynamic set of hosts and their peer-to-peer connections on different ports must be analysed. A matrix visualization focused on the display of one matrix at a time cannot cope with this task.
We address the research problem of the visual analysis of sets of matrices. We present a technique for comparing matrices of potentially varying size. Our approach considers the rows and/or columns of a matrix as the basic elements of the analysis. We project these vectors for pairs of matrices into a low-dimensional space which is used as the reference to compare matrices and identify relationships among them. Bipartite graph matching is applied on the projected elements to compute a measure of distance. A key advantage of this measure is that it can be interpreted and manipulated as a visual distance function, and serves as a comprehensible basis for ranking, clustering and comparison in sets of matrices. We present an interactive system in which users may explore the matrix distances and understand potential differences in a set of matrices. A flexible semantic zoom mechanism enables users to navigate through sets of matrices and identify patterns at different levels of detail. We demonstrate the effectiveness of our approach through a case study and provide a technical evaluation to illustrate its strengths.

Categories and Subject Descriptors (according to ACM CCS): H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Search process

1. Introduction

Relational data, such as computer or social networks, can be modelled as graphs. These graphs can be visualized as matrices by simply using their adjacency matrices. The matrix cells can be coloured to show binary, categorical or continuous attributes for each edge, e.g., the edge weight [Ber81, p. 33]. Matrix visualizations are particularly suitable in cases where the associated graph is dense [GFC05]. Interactive features may improve the usability of matrix visualization when specific aspects, such as groups or connectivity, should be explored in the data [HFM07, HF06].

Existing techniques generally support the display of a single, static graph. However, graphs such as computer or social networks may *change* over time. Many analysis tasks are focused on exploring that change; e.g. the comparison of a series of *snapshots* of a network over time. This gives rise to a matrix *comparison problem*. This problem is particularly challenging since both the edge sets and the node sets may change, yielding graphs, and thus matrices, of different sizes. Comparison tasks in large datasets can be supported effectively through ranking or clustering operations. These typically require the definition of a *distance function* for sets of matrices. Here too, the variation in size is problematic; the Euclidean distance, for example, can only be applied to matrices of the same size.

We introduce a novel technique for the comparison of sets of univariate matrices. We make matrices comparable through data reduction, i.e. by projecting matrices of different size to the same low dimensional space. For each pair of matrices we obtain a pair of point clouds, which is com-

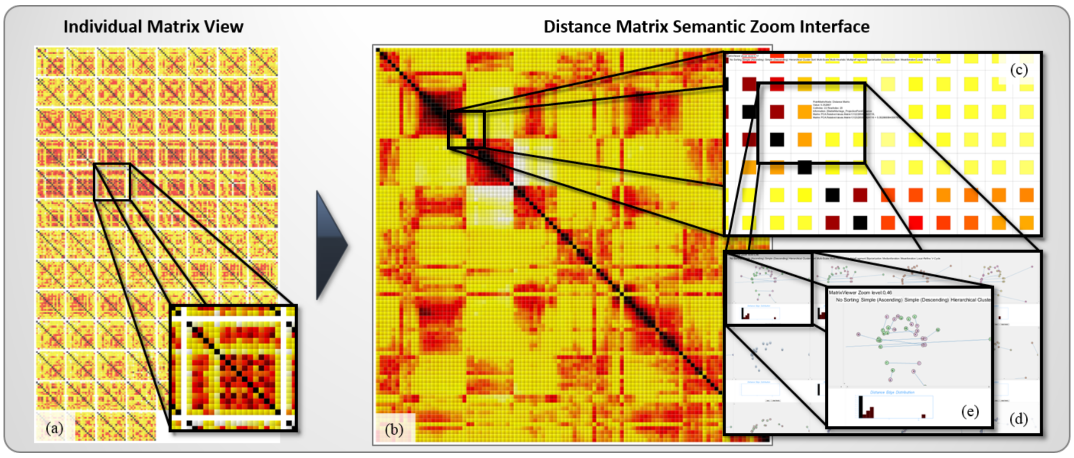


Figure 1: *Using our semantic zoom interface users can explore distances between matrices (a) (here: 100 matrices; ordered by time stamp; cf. Section 6.1). Starting from an overview distance meta-matrix (b) showing the pairwise distances between matrices, users can identify patterns (e.g. strong groups or outliers). Having found such patterns, users can investigate the impact of matrix size variations on the distance calculation (c) and steer it using a simple set of interactions (d) and (e).*

pared by solving a bipartite graph-matching problem. Our distance function therefore relies on essential information of the matrices. The function is visually interpretable, which can enhance user understanding in the analysis of change. Finally, it enables the implementation of ranking and clustering tools for sets of matrices of varying size. We demonstrate the utility of our distance function by applying it to test data and we explore the sensitivities and precision of the distance function based on benchmark data. In particular, we consider alternative projection methods, and assess the effect of outliers, scaling factors and size variations in the input data. We also apply our technique to selected, real-world network-analysis scenarios and show that insights can be found which are difficult to perceive with existing tools.

The remainder of the paper is structured as follows. In Section 2 we discuss related work. In Section 3 we introduce our distance function for matrices and describe its implementation in Section 4. Section 5 shows how our technique can be used to support real-world matrix comparison tasks. In Section 7 we present the results of sensitivity experiments for the assessment of the stability of our distance function. In Section 8 we discuss limitations and possible extensions. Finally, Section 9 concludes the paper.

2. Related Work

We briefly review related work on the visualization of static and dynamic matrices, and on matrix matching.

Static and One-Dimensional Matrix Data. Graph data can be depicted with the help of adjacency matrix visualizations [WTC08]. The cells of the matrix can encode the relationships between nodes (i.e the edge attributes) of a graph. Matrix visualizations provide a highly scalable visual representation of graphs [vLKS*11, GFC05]. They can reveal important aspects of graph structure if they are appropriately sorted and rendered. However, matrix representations are less intuitive than node-link diagrams, thus they need to be supplemented by additional visualizations and interaction techniques to improve understanding. In [HFM07], matrix visualizations and node-link visualizations are combined in an interactive system. The matrix is used to provide an overview representing very dense areas of the graph, and a node-link view shows details for selected parts that are globally sparse. Semantic zoom interaction can help navigate matrices which do not fit into the available screen space. In [EDG*08, AvH04] zooming and dynamic aggregation techniques support the navigation process in large matrices. Matrices, and thus matrix visualizations, occur in many real-world analysis tasks. Matrix visualizations have been used in the analysis of social networks [HF06], and gene regulatory networks [DWvW12]. Time series can efficiently be summarized in a matrix visualization [SKU*12]. Visualizations of similarity matrices are frequently used in the analysis of non-numeric attributes, such as in the pairwise comparison of text documents [BKSK12].

Time-Dependent and Multivariate Matrix Data. Many analysis tasks involve multiple, heterogeneous matrices. For example, a social network is a time-dependent graph whose nodes correspond to entities and edges and/or their attributes corresponds to relations such as friendship or a message (count). Both, nodes and edges may change over time. Each attribute dimension yields a matrix with a single value/dimension encoded in each cell. Each time step may give rise to a different attribute matrix. Many matrix vi-

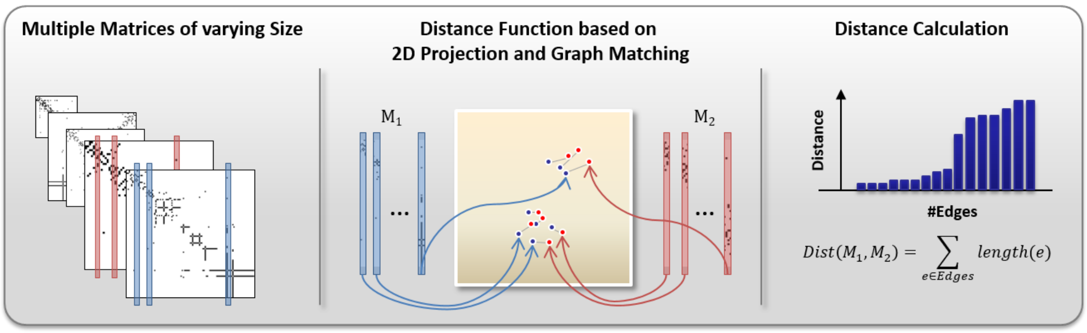

Multiple Matrices of varying Size
Distance Function based on 2D Projection and Graph Matching
Distance Calculation
M_1
M_2
Distance
#Edges
$Dist(M_1, M_2) = \sum_{e \in Edges} length(e)$

Figure 2: *Processing pipeline for our projection-based matrix comparison technique: A set of matrices of potentially different size is input (left). Their columns and/or rows are interpreted as high-dimensional vectors and projected to the plane (middle). Solving a bipartite graph-matching problem on the resultant point clouds leads to a set of allocation edges. Aggregating the euclidean lengths of the edges results in a similarity score for each pair of matrices (right).*

sualizations were developed for one-dimensional and static matrix entries and do not support dynamic and complex matrix data well. One approach to handling time-dependency in graphs is [BVB*11], where graph states are represented as consecutive narrow stripes, in which vertices are arranged vertically on each side. Directed edges connect vertices from left to right to show the graph evolution. In [BSB*10], the interactive visualization of pairs of matrices was addressed. Specifically, one matrix contains weight values and the other contains target values in a correspondence-matrix representation of molecular data, and interaction allows cross-filtering in both matrices. In [BDS*12] time-series data is presented in a triangular matrix, where the matrix cells are statistical aggregates over all possible subintervals.

Dimension Reduction and Fuzzy Graph Matching. Many techniques exist to reduce the dimensionality of data [POM07] and support the exploration process in analysis tasks [YXRW07]. We use projection to compare matrices based on their row/column elements, and define a distance function based on bipartite graph matching. Matching approaches can compare data by finding correspondences of local data properties. An example is the matching of regions in images based on the correspondence of local SIFT features [Low04]. Comparison of graphs by edit distances has been proposed in [ZWS96], however this is an expensive process. Inexact or fuzzy graph matching approaches try to cope with the computational effort by applying tree search/indexing algorithms [SF83, CFSV96, Pel98], transforming the graph matching problem into a continuous, nonlinear optimization problem [FE73, WH97, WW02] or exploring spectral characteristics of the graph [Ume88, CK04, KC02]. Similar to our approach, in [KC02] a vector space is defined using the Eigenvectors of the adjacency matrices. The graph nodes are projected onto points in this space. Then, a clustering algorithm is used to retrieve common local relational structures between different graphs [KC02]. The authors state that this method is robust with respect to graph distortions; corresponding nodes are always close to each other in their graph Eigenspace. The comparative analysis of sets of graphs has been considered in [vLGS09] by means of clustering of statistical graph features.

Our approach goes beyond the state-of-the-art by incorporating a visual-interactive analysis of differences between matrices. We are able to show these differences on a row/column basis, even if the compared matrices vary in size. Implementing a projection-based matrix representation enables us to compute the similarities between row/column vectors and allows these to be visualized for interactive exploration. To the best of our knowledge, the combination of row/column projection and the subsequent solving of a bipartite graph matching to compute a measure of distance is new. It gives rise to a steering mechanism to control the fuzziness in inexact graph matching problems. We introduce a set of interaction operations to steer the computation and to perceive its outcome visually.

3. Basic Idea for Matrix Comparison

We regard a matrix as a set $\mathbb{M}$ of high-dimensional row or column vectors. Two sets $\mathbb{M}_1, \mathbb{M}_2$ of vectors can be compared by computing an aggregate vector-based similarity score. If the matrices have the same size then we can easily compare them, e.g. using the Euclidean distance. However, for matrices of different size this is not possible. In the following, we introduce an alternative approach.

Figure 2 illustrates our comparison approach. A set of matrices $\mathbb{M}_1, \ldots, \mathbb{M}_n$ which vary in size is shown on the left. It is possible to interpret either the rows or the columns as high-dimensional vectors of the length of the respective matrix. Assuming a topology-preserving projection, we project the matrix vectors to the plane (see the center of Figure 2).

It is necessary to consider the effects of the projection

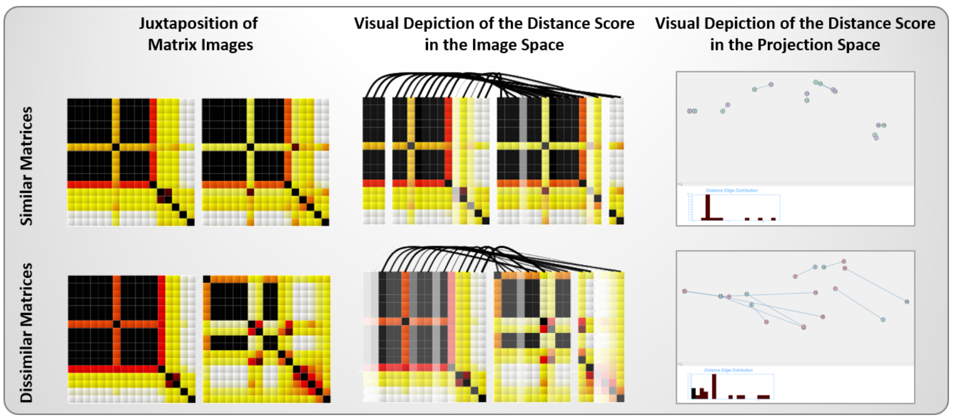

Juxtaposition of Matrix Images
Visual Depiction of the Distance Score in the Image Space
Visual Depiction of the Distance Score in the Projection Space
Similar Matrices
Dissimilar Matrices

Figure 3: *A visual interpretation of the distance calculation is possible from two perspectives. Clicking on a cell of the distance meta-matrix (see Figure 1 (b)) shows the compared matrices (left). Additionally, a transparency factor for columns indicates their impact on the overall distance score. The matrices' columns are visually connected by edges to represent the bipartite graph matching decisions (middle). These connections are also shown in the projection view (right), which lets the user explore patterns in the projection of columns, i.e. projection points which are close together represent columns which are similar.*

for our targeted distance calculation. Projection methods are conventionally used to reduce the dimensionality of data by reducing the number of dimensions while minimizing information loss (in this case, metric properties) [CK04]. They can be separated into linear and non-linear techniques. Linear projections attempt to separate important from unimportant dimensions to form the projection space. Non-Linear projections try to preserve the high-dimensional *neighborhood* properties, such as pairwise distances, in the low-dimensional space and seek to separate projection points whenever their high-dimensional counterparts are far apart.

As Kosinov and Caelli show in [KC02], with an appropriate choice of projection, the projected vectors will reflect the similarity relationships present in the original data. Consequently, a matrix $\mathbb{M}_i$ can be represented by a number of $|\mathbb{M}_i|$ (only rows or columns) or $2 \times |\mathbb{M}_i|$ (rows and columns) projection points in the plane. Since we only consider adjacency matrices they are always square. The use of only rows or columns applies to symmetric matrices (undirected graphs), while non-symmetric matrices (directed graphs) benefit from simultaneous row and column projections.

In the case of projection techniques based on eigendecomposition, we can take advantage of a well-known property: The eigenvalue spectrum of a matrix is invariant with respect to similarity transformations. For any non-singular matrix $\mathbb{M}$ and compatible, invertible matrix $\mathbb{P}$, the product $\mathbb{P}\mathbb{M}\mathbb{P}^{-1}$ has the same eigenvalues as $\mathbb{M}$. This means that the spectrum of a graph represented by its adjacency matrix is not affected by row/column permutations [KC02]. This result is important, since it allows us to compute of the distance between two matrices irrespective of the ordering of rows/columns.

To compare two matrices, we match the projected points of one matrix $\mathbb{M}_1$ to the projected points of the another matrix $\mathbb{M}_2$. Having obtained a matching of the projected point sets, we compute an aggregate distance score over the matching, as shown in Figure 2 (c). This distance score can be used for a range of applications, such as similarity search: Given a matrix, return the most similar matrices from a larger data set to be explored; or to generate a clustering of a set of matrices. It is also possible to interpret the projected vectors (point clouds) as a complementary form of matrix visualization. Thus, our approach can easily be embedded in visual analytics tasks. In this way, it is possible to visually identify outliers in the two-dimensional space that correspond to outliers in the high-dimensional space. Furthermore, we can use linked interactions between the projected representation and the traditional matrix visualizations to help users interpret and interact with the matrix.

4. Implementation

We next detail our implementation, describing the projection and matching schemes we used.

4.1. Projection of Matrix Data

As explained in Section 3, we can take advantage of the eigenspectrum's property of being invariant to row and column permutations. In our case, we can project vertex connectivity data from a graph adjacency matrix to a smaller set of its most important eigenvectors. The projection coordinates obtained this way then represent the relational properties of individual vertices relative to the others in the lower dimensional eigenspace of the graph. In this eigenvector space, structurally similar vertices or vertex groups will be located close to each other, which can be used for approximate comparison and matching of graphs [KC02]. This is shown in Figure 3: The upper right projection view shows the projection of two similar adjacency matrices, while the lower projection view depicts a higher distance score.

As we use projection to compare matrices, we require *determinism* of the projection calculation, that is, the projection calculation needs to produce the same output on the same data set. This statement does not hold for projection techniques, in which an initial seed population is projected, followed by the projection of all other points based on their high-dimensional (dis-)similarity. t-Distributed Stochastic Neighbor Embedding (t-SNE) [vdMH08] or Kohonen's Self-Organising Maps (SOM), as described in [CP97], are representatives of possibly non-deterministic projection approaches, and we therefore exclude them from our approach.

In our implementation we apply one exemplary linear and one non-linear projection technique to our data sets. Both implementations are based on eigendecomposition approaches. The linear projection we chose was *Principle Component Analysis (PCA)* [Jol05]. The non-linear projection technique was *Classical Scaling (Metric Multidimensional Scaling)* [CC00]. The implementation of PCA and Classical Scaling was taken from the Projection Explorer Framework of Paulovich [POM07].

4.2. Comparing Matrices By Graph Matching

As a result of the projection of a pair of matrices $\mathbb{M}_1$ and $\mathbb{M}_2$ into the plane, we obtain two point sets representing the matrices. We compare the matrices by solving a bipartite graph-matching problem on the point sets in the projected space. One possible solution is to calculate the minimum sum of pairwise Euclidean distances between the points' projection coordinates. We now describe our distance calculation for the cases of equal and unequal size of the input matrices in detail.

Matrices of Equal Size. For $|\mathbb{M}_1| = |\mathbb{M}_2|$ we have to find $|\mathbb{M}_1|$ edges connecting the two vertex sets. This computational problem is referred to as the *stable marriage problem*. Gale and Shapley showed that stable marriages exist for any choice of rankings [GS62].

For the matrix comparison task, we can use the Euclidean distances between the projection point coordinates from $\mathbb{M}_1$ and $\mathbb{M}_2$ to compute the (men-)optimal allocation for the two matrices. The optimality proof is given in [GS62]. After having found a pair matching, we can define the sum over all pairwise Euclidean distances as the distance between $\mathbb{M}_1$ and $\mathbb{M}_2$. This is shown in Equation 1:

$$Dist(\mathbb{M}_1, \mathbb{M}_2) = \sum dist_{proj}(m,n) \tag{1}$$

where $m \in \mathbb{M}_1$, $n \in \mathbb{M}_2$ and m and n are matching partners. In our implementation, we use the *Extended Gale-Shapley algorithm* [GI89], which is a performance improvement over the classical Gale-Shapley algorithm [GS62].

Matrices of Unequal Size. When $|\mathbb{M}_1| \neq |\mathbb{M}_2|$, the matching defined previously needs to be modified. It is still possible to calculate the men-optimal allocation between the projection points, but only $\min(|\mathbb{M}_1|, |\mathbb{M}_2|)$ can be allocated. For the remaining points a penalty can be added to the overall distance score. Accordingly, the distance function from Equation 1 can be adapted in the following way:

$$\begin{aligned} Dist(\mathbb{M}_1, \mathbb{M}_2) = &\sum dist_{proj}(m,n) \\ &+ (\max(|\mathbb{M}_1|, |\mathbb{M}_2|) - \min(|\mathbb{M}_1|, |\mathbb{M}_2|)) \\ &\times Penalty(\mathbb{M}_1, \mathbb{M}_2) \end{aligned} \tag{2}$$

where $m \in \mathbb{M}_1$, $n \in \mathbb{M}_2$ and m and n are matching partners. The penalty function $Penalty(\mathbb{M}_1, \mathbb{M}_2)$ can be chosen in accordance with the task at hand. In the current implementation the penalty calculation schemes, depicted in Equation 3, can be interactively chosen:

$$Penalty(\mathbb{M}_1, \mathbb{M}_2) = \begin{cases} 0 \\ \max(dist_{proj}(m,n)) \\ \max(dist_{proj}(m,n))^2 \end{cases} \tag{3}$$

where $m \in \mathbb{M}_1$, $n \in \mathbb{M}_2$, and m and n are matching partners. The penalty can be set to zero (`ZeroPenalty`), or multiples of the maximum distance (`MaxDist`) or its square (`MaxDistSquare`), among all matches. Whenever the matching results in a small maximum matching distance score, only a small penalty will be added. Whenever the maximum matching distance is large, a large penalty will be added.

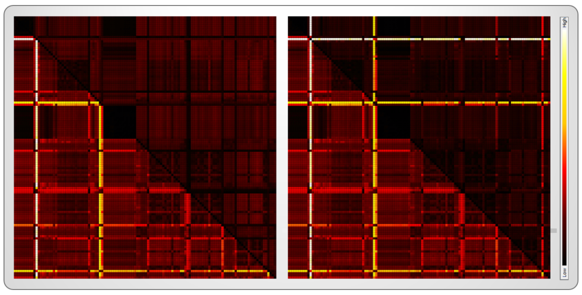
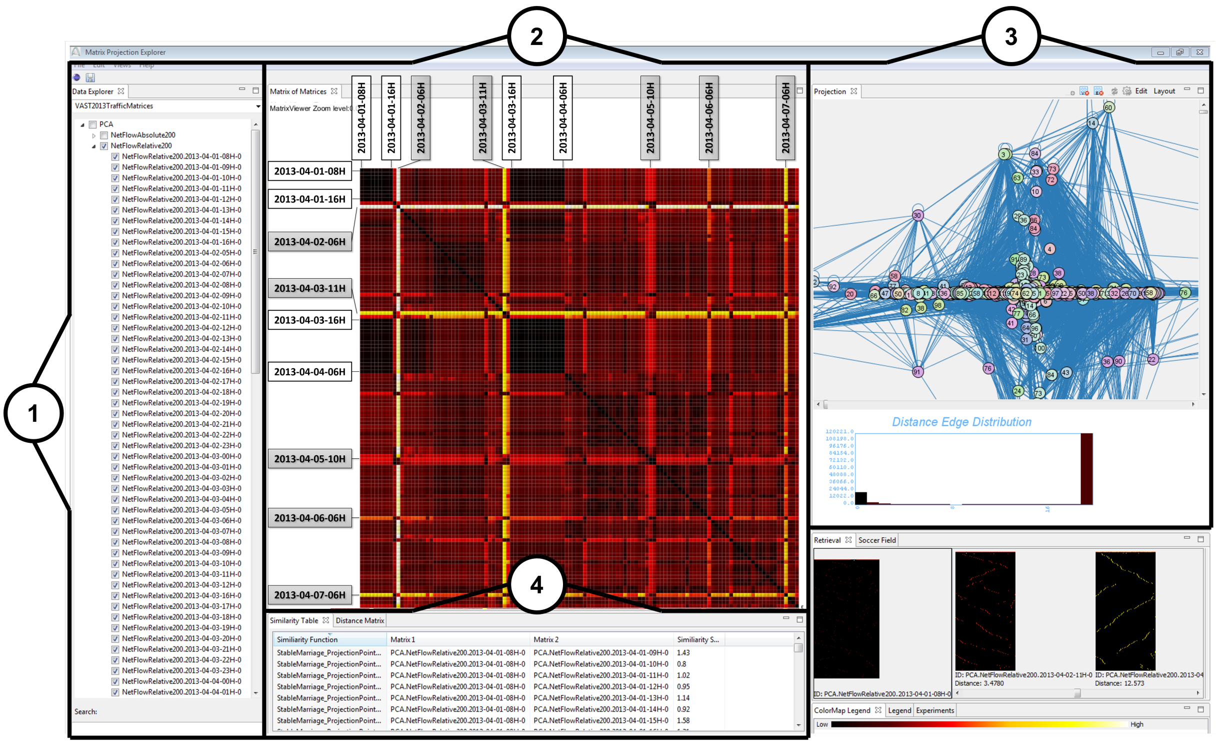


Figure 5: *Changing the penalty function has a large impact on the appearance of the distance meta-matrix showing all pairwise matrix comparisons. From left to right, the* `ZeroPenalty` *and* `MaxDistSquare` *penalty functions are rendered in the upper diagonal part of the matrix. The lower part shows the* `MaxDist`*, for reference purposes.*

Figure 5 depicts the impact of changing the penalty function for the VAST 2013 Challenge Dataset discussed in Section 6.1. From left to right, the `ZeroPenalty` and `MaxDistSquare` penalty functions are plotted in the upper



Similarity Table | Distance Matrix

Similiarity Function	Matrix 1	Matrix 2	Similiarity S...
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-09H-0	1.43
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-10H-0	0.8
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-11H-0	1.02
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-12H-0	0.95
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-13H-0	1.14
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-14H-0	0.92
StableMarriage_ProjectionPoint...	PCA.NetFlowRelative200.2013-04-01-08H-0	PCA.NetFlowRelative200.2013-04-01-15H-0	1.58

Figure 4: *Matrix Projection Explorer is used to visualize matrices and their projections. The overview (2) shows a distance meta-matrix of all pairwise matrix distances for the VAST Challenge 2013 dataset with 120 matrices. Patterns, like closely related (dark groups) and outlying (light rows) matrices, stand out. The projection view (3) lets the user explore the selected matrices' structural similarities expressed in the projection space.*

diagonal part of the matrix. The lower diagonal part shows the `MaxDist`, for reference purposes.

The choice of penalty function is closely related to the graph matching tasks at hand. If the size differences are important a "strong" penalty must be applied. The lower the penalty, the more *fuzzy* the distance calculation becomes. In other words, a "weaker" penalty function, e.g. `ZeroPenalty`, shifts the focus to matrix comparisons that ignore size differences and try to derive a statement from the available information. Other penalty functions are possible, ranging from a static penalty score to a penalty that reflects the situation in high-dimensional space.

5. Exploration and Analysis of Sets of Matrices Using Projection-Based Distance

Our distance calculation technique is proposed as a basis for a range of applications. Specifically, we see two important domains: (1) Applications using the projection-based approach for the distance calculation, and (2) Applications that enable domain experts to draw conclusions from the projection view as a complementary view for visual matrix analysis. We developed a prototype that implements the proposed distance computation and uses it to support visual exploration in sets of matrices (Matrix Projection Explorer). We now describe the system and proposed analytic work flows.

5.1. Matrix Projection Explorer

Our system consists of two parts: (1) A NoSQL database to store matrices and cache projections. (2) A visual front end, consisting of four components (see Figure 4):

1. *Data Explorer View*: Allows the selection of a set of matrices to be examined and visualized.
2. *Matrix Visualization View*: Shows matrix visualizations in a heat-map-style display (for individual matrices) or an overview of a set of matrices (sorted by distance). A semantic zoom function allows the transition between projection view ports, the matrix view and the meta-matrix overview (see Figure 1 and the supplementary video).
3. *Projection View*: The main interactive component, which can be found at the lowest semantic zoom level for every matrix pair, depicts the chosen matrices' two-dimensional projections. The distance calculation is represented by means of the graph matching. The user can

interact visually with the algorithm by excluding or including vertices and edges from the matching or adapt the penalty calculation to the task at hand.
4. *Similarity Table*: Shows the distance scores for a set of two or more matrices selected in the Data Explorer View.

In addition, a legend allows the highlighting of matrices, a colour map shows colours applied in the matrices and a retrieval view ranks the most similar matrices to the current selection in ascending order. Finally, Views for experiments allow the user to start and keep track of experiments.

5.2. Workflow and Interaction

When the user selects a set of matrices, their high- and low-dimensional representations are rendered on the screen as matrix views and projection views. The two-dimensional points are rendered at the positions determined by the selected projection technique in the projection view. The matrix representation is rendered following the best practices presented in [Fek04]. All pairwise distances between the matrices are also calculated and the user can switch to a *meta-matrix* representation of these. We experimented with a MDS projection of the distance values. While it proved beneficial in perceiving similarities (i.e., groupings) we rejected the idea, because it was hard to reflect interactive changes to the distance calculation in a visually traceable manner.

For inspection purposes, the user can select a calculation to show the best possible bipartite graph matching allocation, as described in Section 4.2. The matching is visualized by adding edges to the projection view (see Figure 4 (3)), connecting the matched vertex pairs.

5.3. User-Guided Distance Calculation

Our approach explicitly supports the interactive guidance of the distance calculation by the user. As shown in Figure 1 (d) and Figure 6 (c), by zooming from the overview meta-matrix into an area of interest, the user can investigate each pairwise matrix comparison in the projection view. In this view three different interactions are possible: (1) Selecting and deactivating projection points allows the exclusion outliers or otherwise irrelevant rows/columns; (2) Selecting edges with a lasso or by clicking on the respective distance histogram bin, allows the exclusion ranges of edge lengths from the distance; (3) Adapting the penalty function (clicking on the distance histogram penalty bin or via a specific penalty dialogue) allows modification of the effect of the matrices' size differences on the distance. We demonstrate the usefulness of these interaction mechanisms in the use cases in Sections 6.1 and 6.2 and in the supplementary video.

6. Use Case Demonstration of Our Approach

We now show the use of the Matrix Projection Explorer to analyse various data sets, demonstrating its applicability to potential analysis cases.

6.1. Application to Dynamic Computer Network Analysis (VAST 2013 Challenge Dataset)

Monitoring large computer networks is a challenging, but a highly important task for network operators. While there are many tools to plot and explore time-series for single hosts and network ports, most tools lack an overview of the whole computer network independent of the number of underlying hosts. We use the VAST Challenge 2013 dataset [CGW13] to demonstrate the usefulness of our tool on a realistic dataset in a network security scenario. We preprocessed the data to fit our matrix-based data model, imported the first week of NetFlow traffic of Mini Challenge 3 into our system and aggregated the dataset hourly. Each one-hour interval is represented as a matrix, which conveys the number of bytes transferred from any source IP address to any destination ports. Obviously, the matrix sizes are quite diverse, because in each time interval there might be a different number of computers and even a different number of active destination ports. These common properties make the dataset a challenge to analyse and visualize in an overview for a long period.

The interactive overview visualization is shown in Figure 4. It depicts the overall distance of aggregated connections active for each hour. Each matrix cell represents a distance value between two different hours.

At a first glance, several horizontal and vertical lines stick out as patterns. They represent individual hours, which have highly different underlying connection matrices compared to all other points in time. The event on 2013-04-02 06:00 to 06:59 is indeed a denial-of-service (DoS) attack to one of the company's web servers, which is confirmed in the VAST Challenge scenario. Another confirmed DoS attack is clearly visible on 2013-04-03 around 11:00. Such events have a high impact and are outstanding in this overview, because they are highly different from normal network behaviour.

More subtle patterns are visible in the visualization as well. There are several rectangles (2013-04-01 08:00 until 16:00, 2013-04-03 16:00 until 2013-04-04 05:00), which are almost black, representing time intervals with quite homogeneous connection matrices. The insight can be confirmed, that the traffic in those hours seem to be quite similar. Further analysis reveals that during the mentioned time period less unique IP addresses are involved than usual. A possible reason could be a crashed server during the aforementioned DoS attack. However, such hypotheses can only be confirmed by taking further log data into account, because the NetFlow data does not provide enough detail to answer such questions.

Interestingly, there are at least three very visually outstanding hours (2013-04-05 10:00, 2013-04-06 06:00, 2013-04-07 06:00), similar to those hours in which DoS attacks occurred. These should be investigated further by the analyst. For this dataset a modification of the distance penalty calculation proves to be useful. As Figure 5 shows, changing the Penalty calculation to `ZeroPenalty` visually boosts

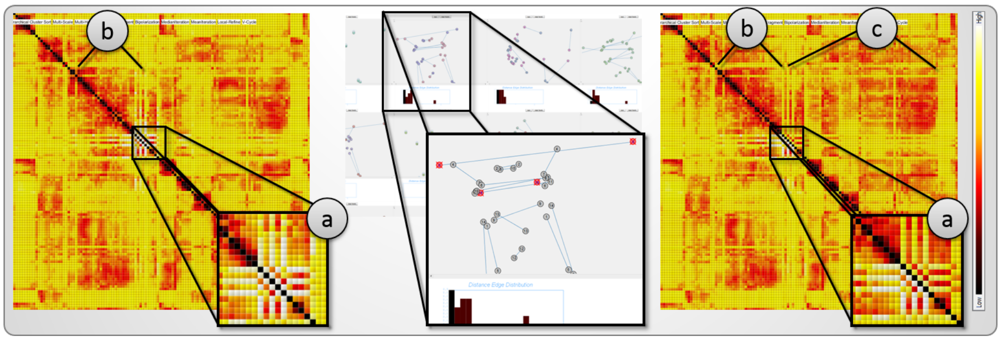


Figure 6: *Excluding vertices from the calculation helps to filter out aspects of low importance. In this soccer analysis task it makes sense to exclude goal-keepers to find semantically similar game situations, where goal-keepers have a low impact.*

areas of homogeneity, while applying the `MaxDistSquare` penalty function gives outliers a higher visual impact.

Overall, the visualization does support the identification of behaviour-related patterns. The main advantage is a compact, but dense overview, based on the notion of matrix similarity for different points in time, which proved to be quite effective for the VAST Challenge 2013 dataset.

6.2. Application to the DEBS 2013 Challenge Dataset

As a second example, we consider the ACM International Conference on Distributed Event-Based Systems (DEBS) *soccer game analysis* challenge dataset of 2013 [JZ13]. The dataset originates from a real-time movement tracking system deployed on a football field of the Nuremberg Stadium in Germany. For the duration of a game, real-time wireless sensors embedded in the player's shoes and the ball recorded the respective positions as a function of time. To transform the data into matrices, we extracted a distance matrix between all players for every second of the game; i.e. a set of 4126 matrices depict every 1-second game situation. Every row, respectively column, in the matrices corresponds to one player and every matrix cell represents this player's distance to another player at that specific time instance of the game. Figure 6 (a) shows a range of situations during the second half in the meta-matrix overview. The view is sorted by time and shows similar game situations adjacent to each other. We can see that two patterns occur: (1) The pattern [a] depicts a corner-kick (2) the diffusion-like pattern [b] refers to a goal-kick.

One possible task could be to find similar game situations in a soccer game, such as corner kicks, goal-kicks, or shots at the goal. Other tasks are exemplified in [PVF13]. Most of these tasks tend to be related to fuzzy queries, in the sense that player positions may change between semantically similar situations. Also, certain players may not even be important for a game situation (e.g., the offensive goalkeeper is normally less important for a corner kick if not attacking directly). In these cases it seems appropriate to exclude data vectors from the calculation, whenever they have no impact on the task at hand. This typical work flow is shown in Figure 6, where the user zoomed into an area of a known corner-kick (left) and selects vertices to be excluded from calculation (middle). The influence is shown in the adapted overview (right), where the area [c] changes significantly.

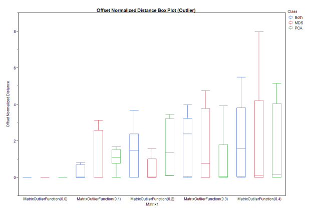
7. Sensitivity Experiments

To evaluate our technique we conducted experiments to (1) assess its dependency on the chosen projection technique, and (2) validate the tolerance of outliers, scaling factors and size variations. We used the data set from the *Petit Testsuite* [Pet03] for our evaluation. The test suite consists of 21 sparse graphs derived from random models and real-life scenarios, such as VLSI, computational fluid dynamics, or earthquake wave propagation. Out of the 21 graphs, we selected the 17 based on real data. The remaining four graphs, were replaced with five self-constructed dense graphs, each with a higher average vertex degree.

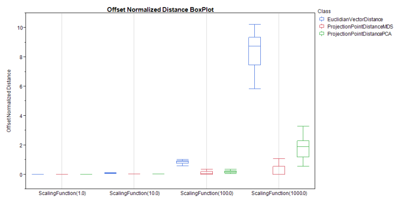
Dependency on the Projection Technique. We conducted an experiment with PCA and Classical Scaling projections. Our approach is not dependent on the chosen projection technique, as illustrated in Figures 7(a) and 7(b). Both projections allow the calculation of a distance as long as the matrices compared were projected with the same method.

For the evaluation of sensitivity to outliers, scaling factors and size variation we first defined a basis for comparison. We chose the minimum aggregate over all Euclidean comparisons of the two matrices' high-dimensional vectors. We refer to the Euclidean test results as the *EuclideanVectorDistance* and to the similarity calculation after projection into the plane as the *ProjectionPointDistance*.

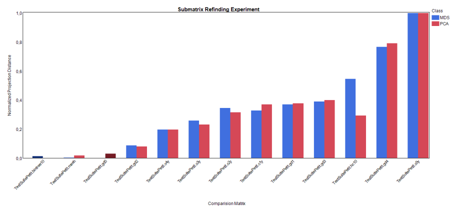
Outlier Sensitivity. We define outlier tolerance as follows: The comparison of a matrix with a certain number of outlier rows/columns to the same matrix without outliers should



(a) Outlier Sensitivity Experiments



(b) Scaling Factor Sensitivity Experiments



(c) Submatrix Retrieval Experiment

Figure 7: *The experimental evaluation shows the technique's insensitivity to noise in the input data, scaling factors and the projection technique used, as long as it is deterministic.*

result in a lower increase in the ProjectionPointDistance when compared to the EuclideanVectorDistance. Figure 7(a) shows the distribution of our experimental results. These were obtained, by varying (1) the investigated matrix, and (2) the number of outlier rows/columns between 0 and 50%. We can see that the ProjectionPointDistance is tolerant with respect to outliers. However, a large number of outliers has an impact on the projection and eventually on the distance calculation. It is also clear that MDS and PCA behave differently in the case of outliers. While MDS is more tolerant of outliers, the PCA projection distance shows a positive correlation with the maximum value of the outlier vectors.

Scaling Sensitivity. We define scaling-factor tolerance as follows: The comparison of a scaled matrix to the same matrix without the application of a scaling factor should result in a lower distance value increase of the ProjectionPointDistance when compared to the EuclideanVectorDictance. The scaling factors we considered were 1, 10, 100, 1000. As Figure 7(b) depicts, our technique is tolerant with respect to scaling factors. Similarly to the outlier experiment, the MDS projection shows less influence in comparison to PCA.

Further Experiments. We conducted a range of other experiments on the test suite. For example, we showed that the row or column ordering did not influence the matrix distance calculation. This is due to the fact that the applied projection techniques operate on the so-called eigenvalue spectrum, which is by definition invariant to similarity transformations.

In Figure 7(c) we show the results of a sub-matrix search experiment to check tolerance of size variation. We cropped a 100×100 sub-matrix from the Petit test suite's *bintree10* matrix and searched for it in the test suite. One can see that *bintree10_cropped* and its larger counterpart have the lowest distance score, meaning that their original relationship could be retrieved.

8. Discussion and Extension

While our technique has proven useful, we have identified several areas where improvements or alternatives could be explored. Firstly, there are many other projection methods. There are also dimension reduction methods which do not involve projection. A thorough analysis of their advantages and disadvantages still remains to be done. Secondly, there is the question of target dimension: We chose two dimensions, since this enables a simple graphic representation of the results. However, earlier works [KC02, CK04] incorporate considerations of the adjacency matrix's rank. While our technique can be adapted for the comparison of higher-dimensional objects appropriate and effective visualizations would be indispensable for this step. Thirdly, alternative graph-matching methods could be applied in the projection plane. Taking high-dimensional relationships for the vertex allocation into account could help to deal with matrices of dramatically varying size. Fourthly, we want to consider multivariate matrix data as a challenging future work point.

9. Conclusion

Matrices occur in many applications. While previous work centered on understanding the relationships in one matrix, the focus is shifting to a new research problem: The comparative analysis of sets of matrices. We motivated the research problem of the visual analysis of sets of matrices and introduced a novel technique for the comparison of matrices of varying size. A projection-based comparison allows analysts to understand the structure of and differences between matrices, while allowing them to comprehend and influence the distance calculation by visual means. A visual analysis framework was presented for the interpretation and adaptation of the distance score. We evaluated our method with a case study, conducted to showcase the approach on a example real-world dataset. In addition, we conducted an experimental analysis to show the insensitivity of the proposed matrix distance function to noise and scaling. Our technique is shown to be independent of the chosen projection function, as long as it is deterministic. In conclusion, the presented technique serves as a generic basis for a range of applications. While this paper focused on classical retrieval tasks and outlier analysis, the projection-based approach will enable further matrix analysis applications which we will address in future work.

References

[AvH04] Abello J., van Ham F.: Matrix zoom: A visual interface to semi-external graphs. In *Info Vis., 2004. INFOVIS 2004. IEEE Symp. on* (2004), vol. 1, IEEE, pp. 183–190. 2

[BDS*12] Behrisch M., Davey J., Schreck T., Kohlhammer J., Keim D. A.: Matrix-Based Visual Correlation Analysis on Large Timeseries Data. *Proc. IEEE Symp. on VAST (Poster Paper)* (2012). 3

[Ber81] Bertin J.: *Graphics and graphic information-processing*. de Gruyter, 1981. 1

[BKSK12] Behrisch M., Krstajic M., Schreck T., Keim D. A.: The News Auditor: Visual Exploration of Clusters of Stories. In *Proc. EuroVA International Workshop on Visual Analytics* (2012), Eurographics, pp. 61–65. 2

[BSB*10] Bremm S., Schreck T., Boba P., Held S., Hamacher K.: Computing and visually analyzing mutual information in molecular co-evolution. *BMC Bioinformatics 11:330* (2010). 3

[BVB*11] Burch M., Vehlow C., Beck F., Diehl S., Weiskopf D.: Parallel edge splatting for scalable dynamic graph vis. *IEEE TVCG 17*, 12 (2011), 2344–2353. 3

[CC00] Cox T. F., Cox M.: *Multidimensional Scaling, Second Edition*, 2 ed. Chapman and Hall/CRC, 2000. 5

[CFSV96] Cordella L., Foggia P., Sansone C., Vento M.: An efficient algorithm for the inexact matching of arg graphs using a contextual transformational model. In *Pattern Recognition, 1996., Proc. of the 13th Int. Conference on* (1996), vol. 3, IEEE, pp. 180–184 vol.3. 3

[CGW13] Cook K., Grinstein G., Whiting M.: VAST Challenge 2013. http://vacommunity.org/VAST+Challenge+2013, 2013. 7

[CK04] Caelli T., Kosinov S.: An eigenspace projection clustering method for inexact graph matching. *IEEE TPAMI 26*, 4 (2004), 515–519. 3, 4, 9

[CP97] Carreira-Perpinan M. A.: A Review of Dimension Reduction Techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09 09* (1997), 1–69. 5

[DWvW12] Dinkla K., Westenberg M., van Wijk J.: Compressed adjacency matrices: Untangling gene regulatory networks. *Vis. and Computer Graphics, IEEE Trans. on 18*, 12 (2012), 2457–2466. 2

[EDG*08] Elmqvist N., Do T.-N., Goodell H., Henry N., Fekete J.-D.: ZAME: Interactive Large-Scale Graph Vis. *2008 IEEE Pacific Vis. Symp.* (Mar. 2008), 215–222. 2

[FE73] Fischler M. A., Elschlager R.: The representation and matching of pictorial structures. *IEEE TC 22*, 1 (1973), 67–92. 3

[Fek04] Fekete J.-D.: The infovis toolkit. In *Proc. of the IEEE Symp. on Info Vis.* (Washington, DC, USA, 2004), INFOVIS '04, IEEE Computer Society, pp. 167–174. 7

[GFC05] Ghoniem M., Fekete J.-D., Castagliola P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Info Vis. 4*, 2 (July 2005), 114–135. 1, 2

[GI89] Gusfield D., Irving R. W.: *The stable marriage problem: structure and algorithms*. MIT Press, 1989. 5

[GS62] Gale D., Shapley L. S.: College admissions and the stability of marriage. *The American Mathematical Monthly 69*, 1 (1962), pp. 9–15. 5

[HF06] Henry N., Fekete J.-D.: Matrixexplorer: a dual-representation system to explore social networks. *IEEE TVCG 12* (2006), 677–684. 1, 2

[HFM07] Henry N., Fekete J.-D., McGuffin M. J.: NodeTrix: a Hybrid Visualization of Social Networks. *IEEE TVCG 13*, 6 (2007), 1302–9. 1, 2

[Jol05] Jolliffe I.: *Principal component analysis*. Wiley Online Library, 2005. 5

[JZ13] Jerzak Z., Ziekow H.: DEBS Challenge 2013. http://www.orgs.ttu.edu/debs2013/index.php, 2013. 8

[KC02] Kosinov S., Caelli T.: Inexact multisubgraph matching using graph eigenspace and clustering models. In *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2002, pp. 133–142. 3, 4, 5, 9

[Low04] Lowe D. G.: Distinctive image features from scale-invariant keypoints. *IJCV 60*, 2 (2004), 91–110. 3

[Pel98] Pelillo M.: A unifying framework for relational structure matching. In *Pattern Recognition, 1998. Proc.. Fourteenth Int. Conference on* (1998), vol. 2, pp. 1316–1319 vol.2. 3

[Pet03] Petit J.: Experiments on the minimum linear arrangement problem. *J. Exp. Algorithmics 8* (Dec. 2003). 8

[POM07] Paulovich F. V., Oliveira M. C. F., Minghim R.: The projection explorer: A flexible tool for projection-based multidimensional vis. In *Proc. of the 2008 Brazilian SIBGRAPI* (Belo Horizonte, Brazil, 2007), IEEE CS Press, pp. 27–36. 3, 5

[PVF13] Perin C., Vuillemot R., Fekete J.-D.: Soccerstories: A kick-off for visual soccer analysis. *Vis. and Computer Graphics, IEEE Trans. on 19*, 12 (2013), 2506–2515. 8

[SF83] Sanfeliu A., Fu K.-S.: A distance measure between attributed relational graphs for pattern recognition. *Systems, Man and Cybernetics, IEEE Trans. on*, 3 (1983), 353–362. 3

[SKU*12] Sips M., Köthur P., Unger A., Hege H.-C., Dransch D.: A visual analytics approach to multiscale exploration of environmental time series. *IEEE Trans. Vis. Comput. Graph. 18*, 12 (2012), 2899–2907. 2

[Ume88] Umeyama S.: An eigendecomposition approach to weighted graph matching problems. *Pattern Analysis and Machine Intelligence, IEEE Trans. on 10*, 5 (1988), 695–703. 3

[vdMH08] van der Maaten L., Hinton G.: Visualizing Data using t-SNE. *JMLR 9* (Nov. 2008), 2579–2605. 5

[vLGS09] von Landesberger T., Görner M., Schreck T.: Visual analysis of graphs with multiple connected components. In *Proc. IEEE Symp. on VAST* (2009), IEEE Computer Society, pp. 155–162. 3

[vLKS*11] von Landesberger T., Kuijper A., Schreck T., Kohlhammer J., van Wijk J., Fekete J.-D., Fellner D.: Visual analysis of large graphs: State-of-the-art and future research challenges. *Wiley-Blackwell Computer Graphics Forum* (2011). 2

[WH97] Wilson R., Hancock E.: Structural matching by discrete relaxation. *Pattern Analysis and Machine Intelligence, IEEE Trans. on 19*, 6 (1997), 634–648. 3

[WTC08] Wu H.-M., Tzeng S., Chen C.-H.: *Handbook of Data Vis.* Springer, 2008, ch. Matrix Visualization, pp. 681–708. 2

[WW02] Wyk B., Wyk M.: Non-bayesian graph matching without explicit compatibility calculations. In *Structural, Syntactic, and Statistical Pattern Recognition*, vol. 2396 of *Lecture Notes in Computer Science*. Springer Heidelberg, 2002, pp. 74–83. 3

[YXRW07] Yang D., Xie Z., Rundensteiner E. A., Ward M. O.: Managing discoveries in the visual analytics process. *SIGKDD Explor. Newsl. 9*, 2 (Dec. 2007), 22–29. 3

[ZWS96] Zhang K., Wang J. T.-L., Shasha D.: On the editing distance between undirected acyclic graphs. *Int. J. Found. Comput. Sci. 7*, 1 (1996), 43–58. 3