Pixel-Oriented Visualization of Change in Social Networks

Klaus Stein
Computing in the Cultural Sciences
University of Bamberg, Germany
klaus.stein@uni-bamberg.de

René Wegener
Information Systems
Kassel University, Germany
wegener@uni-kassel.de

Christoph Schlieder
Computing in the Cultural Sciences
University of Bamberg, Germany
christoph.schlieder@uni-bamberg.de

***Abstract*—We propose a new approach to visualize social networks. Most common network visualizations rely on graph drawing. While without doubt useful, graphs suffer from limitations like cluttering and important patterns may not be realized especially when networks change over time. Our approach adapts pixel-oriented visualization techniques to social networks as an addition to traditional graph visualizations. The visualization is exemplified using social networks based on corporate wikis.**

***Index Terms*—network visualization, pixel-oriented visualization, wiki**

I. Introduction

One important aspect of social network analysis consists in finding interaction patterns between social actors by appropriate visualization paradigms [1]. The visualization problem addressed in this paper first arised when we analyzed user collaboration in organizational wikis where we missed a good visualization for temporal dynamics which could be used in the context of visual data mining. The social networks studied are extended coauthor networks extracted from organizational wikis using the interlocking measure.[1] The idea of interlocking is simple: if an user B edits a page previously edited by another user A, a directed link from B to A is established. An user C editing the same page afterwards establishes links to A *and* B and so on. Each link is associated with a time stamp, so the network holds the interaction record of users in time.

The analysis of interlocking networks can be applied to many different data sets from SVN or GIT repositories over email corpuses to newsgroups, discussion boards, CMS. Interlocking gives a directed network, as each user action is considered as an “answer” to actions of other users before. The visualizations presented in this paper are nevertheless also useful on undirected graphs, as long as time stamps of interaction are available.

This paper makes the following major contributions: 1) we introduce a new kind of social network visualization based on the pixel-oriented visualization paradigm and extend it for the presentation of networks evolving in time in a way that supports visual data mining; 2) we compare different glyph layout patterns and their application to the problem domain; and 3) we show that temporal patterns indicating cooccurrence and similar behavior are perceptually salient in our visualization even on larger networks.

[1]see [2], [3] for a detailed discussion of this measure

Our paper is organized as follows. We discuss common visualization approaches for social networks and time series in section II. Section III introduces our pixel-oriented network visualization approach and describes the adequate choices of glyph layouts and color scales. Section IV presents a case study that illustrates how to apply our approach to real-world network data. We conclude in section V with a discussion of the results and an outlook on future work.[2]

II. Related work

A. Visualization

Visual data mining techniques take advantage of the efficient perceptual grouping processes of the human visual system (see e. g. [4], [5]). Even in large data sets, perceptual saliency draws the observer’s attentions to patterns. Visualization is most useful to generate hypothesis about regularities in a data set. On the other hand visualization does not provide a proof. For instance nodes positioned close together by a layout algorithm suggest some correlation which vanishes by using another layout.

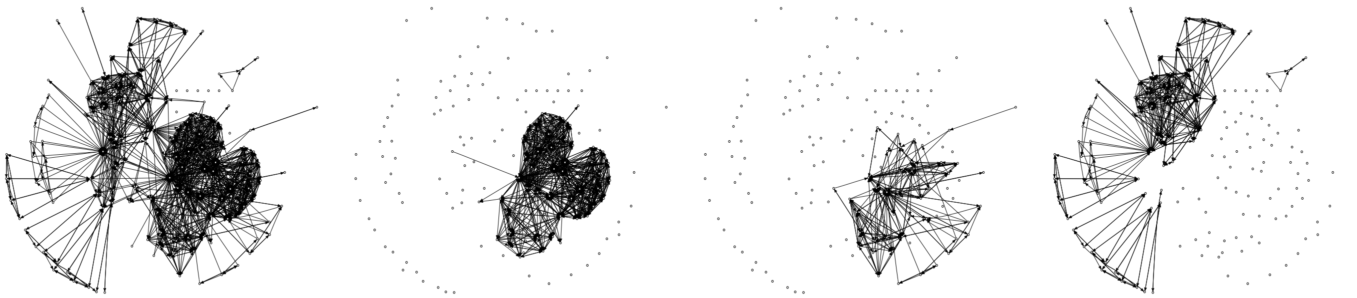
B. Social Network Graphs

Much research has been conducted in the field of social network visualization (for an overview see [1]). The most common way to present a social network is the network graph. The nodes are arranged by one of various graph layout algorithms which continue to be improved in their computational properties as well as their usability (see e. g. [6]). Most software packages for network analysis include graph drawing functionality. Furthermore, a variety of specialized graph drawing packages are available, see e. g. [7].[3]

Alternatively networks can be presented by adjacency matrices that represent the network by some kind of numbers for actors and relations. Ghoniem, Fekete and Castagliola [8] compare node-link diagrams and matrices. While probably less appealing to the user, matrix visualizations avoid common problems like cluttering. The authors conclude that node-link representations are best suited for smaller graphs while

[2]This work was supported by a grant from Volkswagenstiftung. Part of the work presented here was developed in a master’s thesis [22].

[3]Graphviz (http://www.graphviz.org/), JUNG (http://jung.sourceforge.net/), UCINET (http://www.analytictech.com/ucinet/), Pajek (http://pajek.imfm.si/), GUESS (http://graphexploration.cond.org/) and others.



Mar. 2007–Sep. 2008 (a) full timespan | Mar.–Sep. 2007 (b) first term | Sep. 2007–Mar. 2008 (c) second term | Mar.–Sep. 2008 (d) third term

Fig. 1. network evolution in time (*students wiki*)

a higher number of nodes and higher degree of density are better visualized as matrix representations. Shneiderman and Aris [9] state that network graphs can be understood best if they contain between 10-50 nodes and 20-100 links. A higher number of nodes and links makes it more difficult to follow the links, count or identify nodes etc. For this reason Henry, Fekete and McGuffin [10] combine node-link layouts and matrix representations in order to give an easy to understand overview of a network while also revealing details that couldn't be recognized in a pure node-link visualization.

C. Change in Time

For visualizing time-oriented data a variety of methods is available, for an overview see [11], [12]. Broadly speaking, the methods for visualizing time oriented data in social networks fall into two classes: sequences of snapshots and (interactive) animations. Moody [13] considers a third class: network summary statistics plotted as a line graph over time. Since the network topology cannot be recovered from this visualization, we will not consider it.

Two major problems arise with using a temporal sequence of snapshots as visualization: 1) the temporal resolution, i. e., the number of elements of the sequence, is severely restricted, 2) the optimization of the layout algorithm conflicts with changes. We illustrate the problems of visualizations using sequences of snapshots with a network from our own data.

The network presented in figure 1 shows students working on a larger project across three terms using a wiki as documentation platform. So in each term a bunch of new students join, others leave. The visualization uses a spring embedding layout algorithm that optimizes the length of the edges between the nodes (see [14]) in order to achieve a grouping of highly interconnected sets of nodes. (b) to (d) show snapshots of the network at different times that illustrate the temporal change. While for each subgraph both spatial dimensions are used to lay out the graph on the big scale, the x-axis represents time from left (past) to right (future). The spatial extension of each "data point" (network graph) restricts the temporal resolution to few (three) time points.

We could not layout each of the graphs with spring embedding but had to keep each node at a fixed position to be able to compare the graphs. So while the change shows up very well in (b) to (d) the grouping in the single graphs not optimal.

Moody states that the "poor job" of representing change in the network is a problem "fundamental to the media" and suggests to use animations. Network animation software is available either stand alone (e. g. SoNIA[4], see [15]) or part of dedicated network analysis software (e. g. SONIVIS[5]) as most of the network visualization libraries support dynamic node and edge change. By interpolating between the graphs that represent the different intervals of the network, nodes are moved to smoothly re-layout the graph responding to changing edges. One drawback here is that this representation cannot be published in print, some kind of digital medium is needed.[6] More important, animations are nice and striking in presentations but less useful for analysis as they do not give an overview at a glance. Animations inherit the issues of change blindness which means that some changes (maybe just because the glimpse of an eye) won't be realized by the user [4].

The static graphical presentation of data allows to externalize concepts as everything is available on paper. The visual system provides the data analyst with scanning routines that permit a very efficient processing of externalized data. Presenting all the information on one single figure instead of using an animation takes advantage of this ability. In fact, switching our attention from one area of a single figure to another will usually not only be much faster than finding the right interval of an animation with the help of a slide bar [16], it will also allow to for deeper inspection. Human's visual working memory only stores a handful of objects at a time [16] and watching animations forces us to *remember* what we have seen while a static representation keeps everything accessible in parallel.

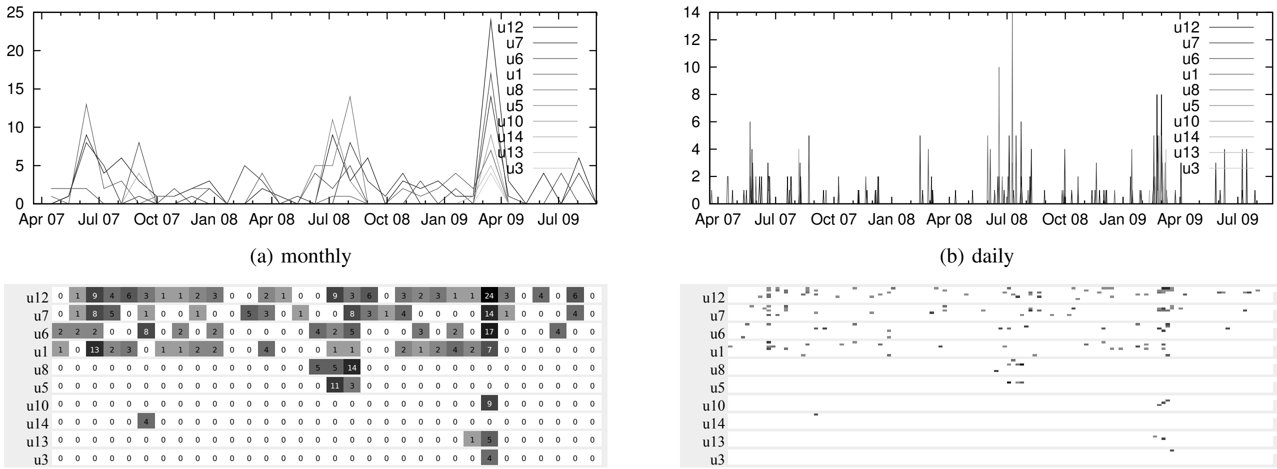
D. Pixel-Oriented Visualization

To avoid overlapping and cluttering and still be able to visualize great amounts of data we make use of every single pixel of our computer screen. Each pixel can represent a

[4]http://www.stanford.edu/group/sonia/

[5]http://sonivis.org

[6]For an animated representation of the network shown in Fig. 1 see http://www.kinf.wiai.uni-bamberg.de/mwstat/examples/wiki_1_weekly_network.swf



(a) monthly

(b) daily

(c) monthly

(d) daily

Fig. 2. collaboration activity per user (*workgroup wiki*)

data value by its color. This idea is called pixel-oriented visualization. It was introduced by [17] and further developed in [18], [19], [20] and other publications. By mapping each data value onto one single pixel in theory the amount of data displayed is equal to the number of pixels. Pixel-oriented visualization has been applied by Guo et al. [21] for the visualization of very large scale network matrices (BOSAM[7]), but to the best of our knowledge it has not been tried on temporal and weighted networks yet.

III. Pixel-oriented Visualization of Network Data

A. Collaboration in Time

Figure 2a shows typical time series data: the x-axis represents time as the independent variable, while the dependent variable (interaction activity) is displayed on the y-axis. Different colors (levels of gray) are used to distinguish the different users. Figure 2c gives exactly the same data in a different representation. The x-axis again represents time, the y-axis now is used to separate the users lying line by line and color (grayscale) represents the intensity. Both figures reveal intensity spikes around June 07, August 08 and April 09. In (a) we easily see the maximum spike to be around 24 at April 09, but it is hard to distinguish the different users. In the given grayscale representation it is nearly impossible, but also a color representation using the whole color spectrum does only help little. In (c) it is easy to distinguish the users and at which time they collaborated most, but it is harder to see the absolute intensity value as the color encoding is less obvious. We only know that darker is more so comparison between users (as well as in time) works well, but to get the absolute values we have to zoom in an read the numbers. In addition data is still highly aggregated (collaboration values are aggregated in a monthly manner).

Figures 2 (b) and (d) present the same data in higher resolution, i. e. day by day instead of month by month following the idea that "to map each data value to a colored pixel [. . .] allow[s] us to visualize the largest amount of data which is possible on current displays" [20]. The pixel-oriented visualization delivers additional implications: As you can see high collaboration values in one month are usually generated by the high collaboration rates of only a few days and not by continuing collaboration during this month.

(d) uses the calendar metaphor which means to arrange the timeline in a weekly zigzag:

user	1	8	15	22	29	36		*Monday*
	2	9	16	23	30	37		*Tuesday*
	3	10	17	24	31	38	. . .	*Wednesday*
	4	11	18	25	32	39		*Thursday*
	5	12	19	26	33	40		*Friday*
	6	13	20	27	34	41		*Saturday*
	7	14	21	28	35			*Sunday*

Although the single dots are too fine to be able to tell which day exactly it represents we nevertheless get the interesting patterns. We not only get an impression which users collaborate a lot at which time during the year (as every pixel column represents a week), we additionally see[8] that only users U6 and U1 are working on Sunday while no one works on Saturday.

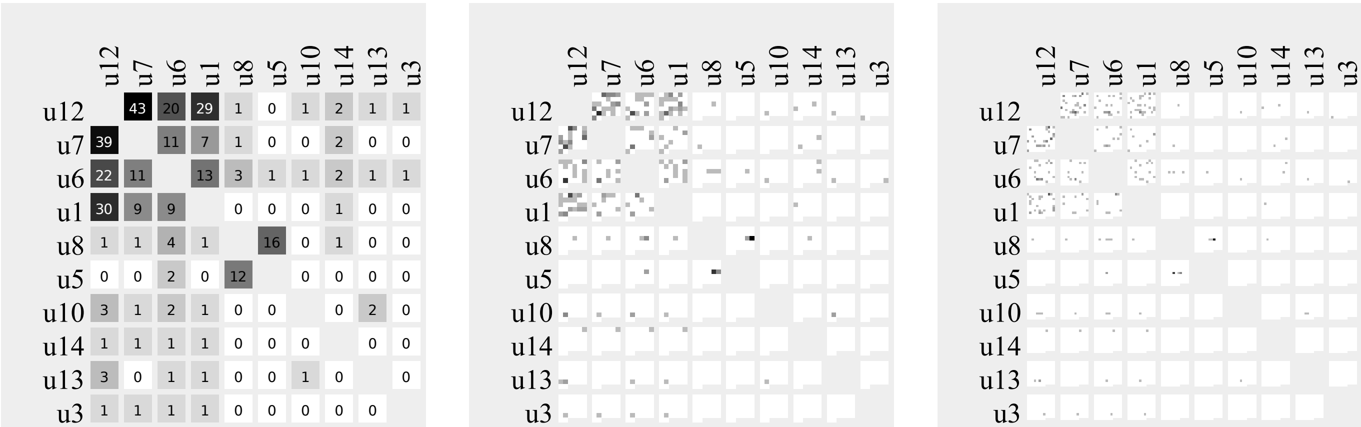
B. The Pixel Matrix

We adapt the idea of pixel-oriented visualization applying it to social network graphs. We present the network as an adjacency matrix with each row and each column corresponding to one node and the matrix elements giving the weight of the edge between the corresponding nodes, as shown in figure 3a. In this figure each matrix element is colored according to its value, and the nodes are sorted by (weighted) degree.

Following the pixel-oriented visualization paradigm to present each value by one colored pixel we can inject the whole timeline of user interaction in one matrix element by

[7] bitmap of sorted adjacency matrix

[8] at least I hope this is visible in the printout



(a) matrix

(b) 4 week steps

(c) weekly steps

Fig. 3. *workgroup wiki* network matrix

folding it. Figure 3b gives the network matrix with each matrix element showing one glyph which holds the timeline folded row by row, each pixel representing the user-user-interaction within four weeks. This is somehow the inverse representation to figure 1 where time is given on the outer x-axis while now time is folded at the inside, i. e. within each glyph.

The scale is changed from monthly to four weeks to get timespans of equal length. While monthly data is easier to present and describe it has the disadvantage that each month covers a different *number* of days and a different number of certain weekdays: June 2010 has four Fridays, July has five. This is important when work is organized weekly, e. g. the workgroup has a weekly meeting on Friday generating extra social interaction. On a monthly raster this generates 20 % more activity for July while everything is even. The other way around if there is some monthly event the four-weeks raster would show irritating data.

We see users U12, U7, U6 and U1 interacting a lot with each other over the whole timespan. Users U8 and U5 come in after around one year for three months, working closely together with small interaction with others (U8 has few connections to the most prominent four users while U5 only meets U6). And users U10, U13 and U3 join the network even later, interacting mainly with the prominent four and not with each other.

This small example gives first insight on the potential of pixel-oriented network visualization (PONV). We do not know exactly at which time users U8 and U5 came in and we do not get a detailed analysis on the interaction between users U12 to U1, but we see gray dots sprinkled over the whole glyph telling us that there was interaction between these users all the time. And we see that U10, U13 and U3 came in at the same time even though their timelines are not layed out side by side. Our visual perception is perfect in detecting that these gray dots are on the same position within their glyph. While we are not able to read absolute data from the graph (neither exact times nor exact intensity values of collaboration) we get a good overview over the temporal behavior of each user as well as temporal cooccurrence of certain events.

And this is exactly what visual data mining is meant for: exploration, detecting something. Now we can ask our system: when exactly did U5 his first edit, when did U8 the last and so on and proof or deny our findings with hard data.

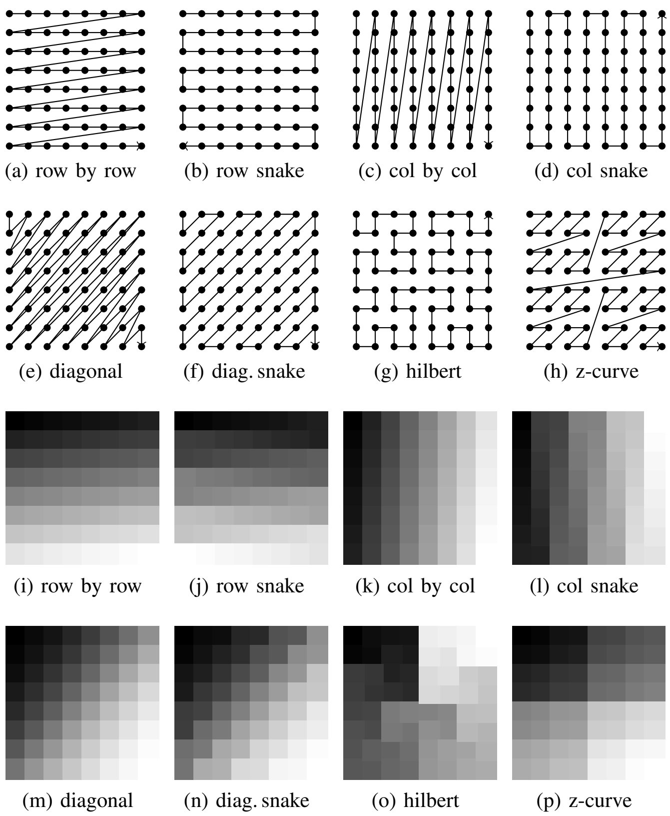
An additional remark about U14. As we see in the U14 *row* he comes in rather early, interacts once and is gone, but in the U14 *column* there are several interactions visible even at later times. This is a result of the network creation method we used. Interlocking response graphs are directed and an edge from user *B* to user *A* is set at the time *B* edits a page *A* edited before. And in the network matrix as presented in figure 3 the row gives the tail node (*R*) and the column gives the head node (*C*), so the edge points from *R* to *C*. So what we see here is that U14 edited a page users U12 to U1 edited before, and months later users U12 to U8 "responded" by editing this page. And we can also see some closer interaction between U6 and U14 as the (U6,U14)-glyph shows us U6 must have edited this page shortly before U14.

C. Inner Glyphs

Our visual perception is good in pattern recognition and edge detection. Looking at figure 3b we detect some interesting vertical bars at (U6,U12), (U6,U1) and (U7,U12). Unfortunately these findings are arbitrary. The timeline is packed into the glyph row by row (see figure 4a) and each glyph has 6×6 pixel. This means that two pixel aligned vertically contain data that is 24 weeks apart. Looking at the same data with glyphs of width 7 would reveal other arbitrary vertical bars. On the other hand we may miss timespans of high activity which start at the end of one column and end at the beginning of the next one. The latter problem can be reduced by laying out the rows in a snake-like way (see figure 4b) as here continuous timespans are not ripped apart.

Things are different in the example before (figure 2d).[9] Each column represents one week with Monday in the top and Sunday in the bottom row which allowed us to see

[9] where the timeline was presented column by column



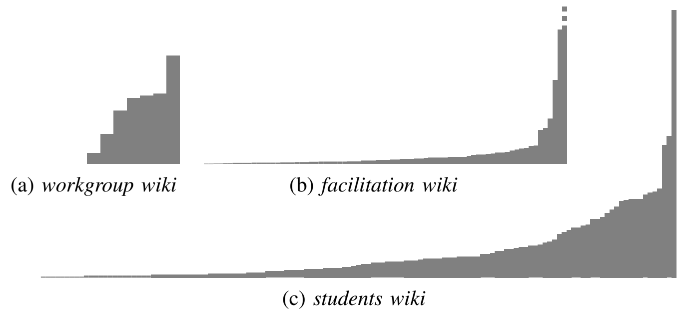
(a) row by row (b) row snake (c) col by col (d) col snake

(e) diagonal (f) diag. snake (g) hilbert (h) z-curve

(i) row by row (j) row snake (k) col by col (l) col snake

(m) diagonal (n) diag. snake (o) hilbert (p) z-curve

Fig. 4. Layout patterns for inner glyphs. (a) to (h) give the layout path while (i) to (p) show example glyphs for a color gradient from black to white in the corresponding layout. (See also [18], [20])



(a) *workgroup wiki* (b) *facilitation wiki*

(c) *students wiki*

Fig. 5. Interaction value distribution for three groups. In (b) and (c) passive users (response value 0) are omitted.In (b) the highest rule is cut of to save space. The y-scales for all graphs differ.

who is working on weekends, and obviously a snake pattern (figure 4d) would disturb.

While figure 3b kind of allows to distinguish single pixels and – with some effort – even to see which row a pixel belongs to we are lost in figure 3c with a weekly resolution. There are darker and lighter regions and you can hardly see how many rows are involved, in other words: you see darker and lighter region patterns which may be incidental as our vision groups events which are layed out next to each other in successive rows while in fact being numbers of weeks apart.

Space-filling curves like the well-known Hilbert curve (figure 4g) are one answer to this problem as these curves lay out one-dimensional data in two-dimensional space in a locality-preserving way, i. e. data points being close together in the one-dimensional representation are kept close in the two-dimensional layout. Unfortunately it also brings data points close which were *very* far away from each other as visible in figure 4o. A second disadvantage of this layout is that the main data order is non-linear. While the layouts (a) to (f) show a main linear direction (top→down, left→right, topleft→downright)[10] the main timeline for the Hilbert curve is U-shaped which is kind of irritating not only for the uninformed reader.

The recursive z-curve (figure 4h) roughly maintains an overall topleft→downright direction with partly more locality than the pure diagonal layouts (e) and (f) but on the other hand shows leaps.

We found the row and column layouts least irritating to the uninformed as well as expert user, they are easy to explain and the more complicated layouts are not capable of solving all the issues described above. And finally its up to the decision of the user which layout he or she prefers.

If the data timeline is known to be structured by periods relevant to the problem domain, e. g. quarters or terms, the pixel oriented visualization can easily adapt to that fact. For example, to present 10 years of data month by month use a row by row or column by column layout with 12×10, and you may be able to see some pattern for Christmas time, for one year (52 weeks) day by day use 14×26 or 21×18 and so on. If no such rhythm applies choose quadratic glyphs with any layout you may like but keep in mind how to interpret it and which patterns may show up spuriously.

D. Color scales

Within this paper the values in the graphs are given in grayscale with 0 encoded as white and the maximum of all values encoded as black. In our social networks, interaction activity is not equally distributed but follows a power law (see examples in figure 5). We want to be able to distinguish 1 from 0 even on busy networks as it can be important to know if two people met at least once. Therefore we apply gamma correction to the data. The latter (distinguishing 1 from 0) certainly could also be done by starting with 1 at some grey tone and after that proceeding linearly to black.

By using color we can further enhance the resolution and expressiveness of the graphs. While it is not useful to cycle through the whole color wheel because it is not intuitively clear whether a blue or a green pixel should represent the higher value, we got nice results using the temperature scale from black to red, orange, yellow and finally white resembling the color change of a black body being heated. Changing both hue and brightness gives higher resolution while staying on the red side of the spectrum allows us to color the background in some blue or green hue giving a clear border to the glyphs. Please note that in the temperature scale 0 is black and the maximum value is white, contrary to grayscale.

[10] which could be flipped e. g. for Arabic readers who may prefer right→left

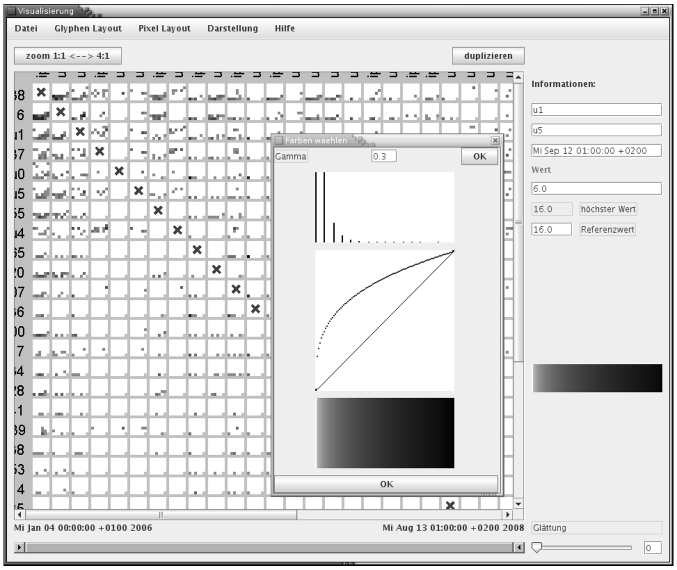


Fig. 6. Graphical user interface of PONVA. The main window shows a pixel-oriented visualization of the network matrix while the dialog in front allows to change γ-correction. To the right detailed information about one selected pixel is displayed.

While using colored graphs on monitor and presentations is nice we restrict ourself to grayscale in publications as everything else needs a color capable output device, which is seldom true for printed proceedings and other scientific papers.

IV. Case Study

A. Interactive Visual Data Mining

The visualization paradigm described in the previous section has been implemented as part of a network analysis package. Figure 6 gives a screenshot of PONVA, our pixel-oriented network visualization application, which allows us to experiment with different layout algorithms and settings on various networks.[11]

Zooming into the full plot as well as "zooming" the glyphs by changing the resolution (figure 3 (b) to (c)) allows to switch between overview and detailed inspection. Selecting the active timespan within one glyph or one column or row and highlighting the corresponding pixels in the other glyphs helps comparing user activity in time. Same holds for selecting one user or a pair of users and highlighting the corresponding row(s), column(s) (and intersections). And clicking on a pixel reveals detailed data from the point in time it represents to its numeric value.

Selecting one of the different glyph layout mechanisms, color scales and some γ-correction appropriate to the network to be analyzed is done interactively. And finally rows and columns can be sorted according to various similarity measures, which are not discussed within this paper due to limited space.

[11] POVNA is written in Java. Not all features described are included in the current stable release. A second implementation in Ruby is part of the Wiki Explorator, available as Open Source at http://wiki-explorator.rubyforge.org/ (used to create the SVG images presented in this paper).

B. Exploring Larger Networks

In the following we analyze visually salient features of pixel-oriented network visualization. In section II-B we introduced the *students wiki* (figure 1). Figure 7 shows the pixel-oriented visualization of this network: (a) gives the interaction timeline for each user (day by day grouped weekly) and (b) the corresponding network matrix. Both plots only give users with at least 50 interactions as the full matrix with 142 users would not be readable on this paper size.[12]

The first thing catching our eye in figure 7a is this dotted vertical bar showing up at about one third of the timeline (users marked with •). This is the start of the second term and obviously the new and some of the older students did a lot of work in this week. Next we see heavy traffic at the beginning of the timeline for a two week timespan (□). Some users continue to participate, others leave, but most of the active users within the first two terms do not show up in the third one, and the users active in the third term (○) were not present before. Exceptions are only users U8 and U105 who show up again at the end of the third term and U68 who came in in the middle of the second term but without being very active. While the users of the first and second term overlap there is low connection to the third term, and this confirms what we see in figure 1 (b) to (d).

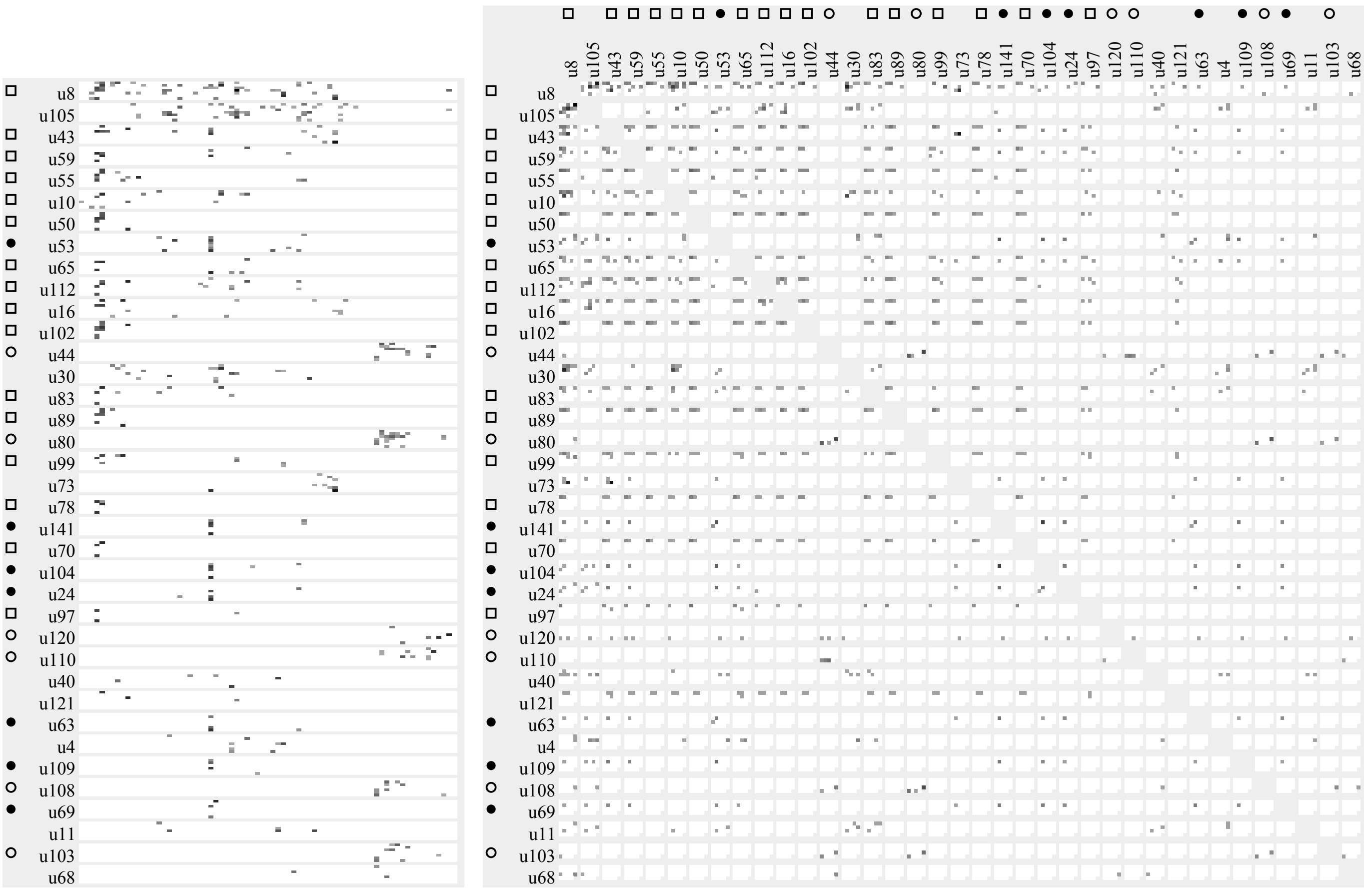
The network matrix (figure 7b) shows a nearly regular pattern of similar glyphs. What we see here is that most of the students being present in the first term (□) collaborate with most of their fellows, for the wiki this means they all worked on the same wiki pages, they did not split up in subgroups working on different pages. And user U120 stands out. While joining the wiki in the third term contrary to his fellows he connects to most of the users of the first and second term, i. e. works on the pages they edited before.

So figure 7 is a good example for the perceptual salience of regular temporal collaboration patterns. It not only shows cooccurrence of activity of single users but also whether these users are connected at these times or not.

Figure 8 introduces a new network. In this organization the wiki was launched as dedicated project with user U35 (•) being the project manager. U35 obviously did a lot of collaboration work until she left the company after three quarters of the timeline, where U48 (○) had to take over this position. While U35 and later U48 collaborated with nearly every other user in the wiki only few users (e. g. the pair U80 and U46 □) started to use the wiki as a collaboration tool by themselves. Contrary to the last example here no uniform patterns are visible, but temporal cooccurrence as well as temporal connection shows up in the fuzzy gray-spotted glyphs as our visual perception smoothens single dots to larger areas.

Our last example (figure 9), the *startup wiki*, was installed during the founding of the company and since then keeps its role as primary collaboration and main content management system. Growing from 3 to 26 users the primary users never stopped to use the wiki and every new employee joining

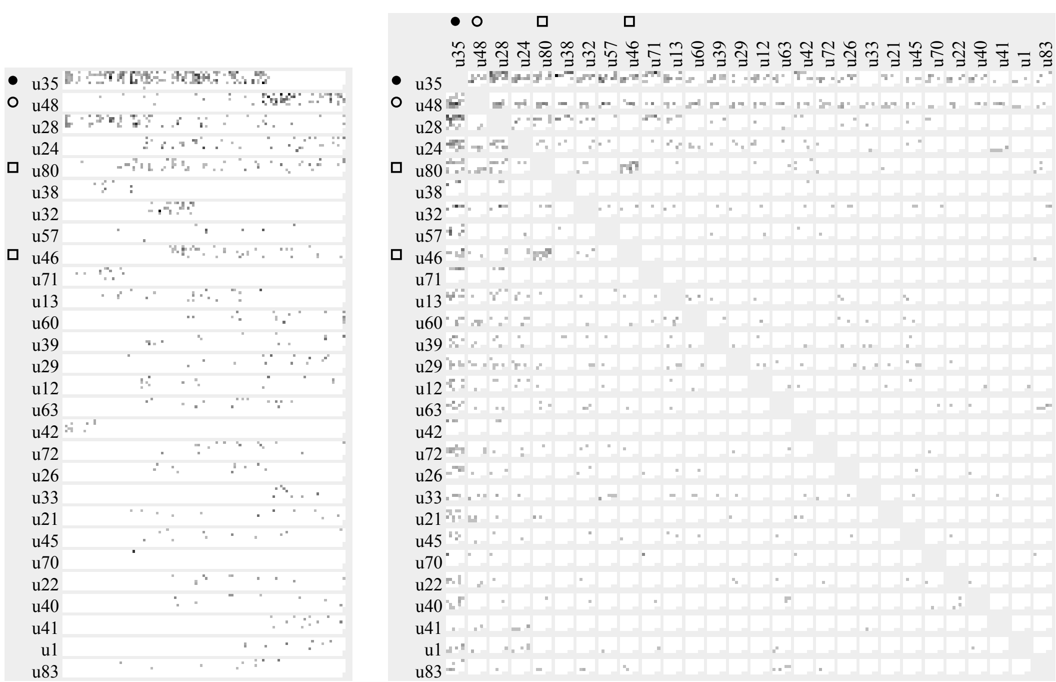
[12] this is less a problem in interactive usage where we can scroll.



(a) collaboration intensity

(b) network matrix

Fig. 7. *students wiki* (users with at least 50 links)



(a) collaboration intensity

(b) network matrix

Fig. 8. *facilitation wiki* (users with at least 50 links)

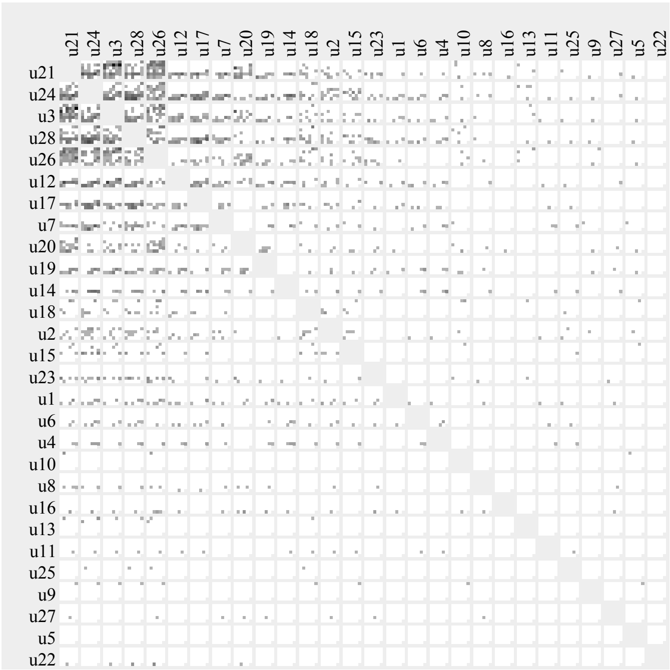


Fig. 9. *startup wiki* network matrix

immediately gets connected to the others. Wiki growth at its best, and nicely visible in the user interlocking network matrix.

V. Conclusion

In this paper we introduced pixel-oriented network visualization (PONV), a new visualization for weighted social networks changing in time. Our approach focuses on using static visualization of dynamic network data as a method of data exploration by the means of visual data mining which works best on medium-size networks where the whole matrix fits on the screen or paper at once.

We discussed different glyph patterns and tended to stick to the simple row by row approach which we found to be least irritating. The interpretation of PONV is not immediately obvious but easily learned by the interested expert. Using several interlocking coauthor networks extracted from organizational wikis we could show how pixel-oriented network visualization gives perceptual salient patterns for temporal cooccurrence of user collaboration as well as user-user-connection. PONV allows to detect similar collaboration patterns across users and to reveal the collaboration between a pair of users across time. It is nevertheless meant as supplemental visualization *besides* network graphs and others and not as a substitution, as interactions between groups of more than two users only show up indirectly, it focuses on temporal patterns, not on the detection of network clusters. This also means that PONV is less useful on rather sparse networks with low traffic where no visual patterns emerge.

Topics for future work are the analysis of different sorting algorithms to group similar users as well as the development of more sophisticated glyph layouts. Additionally the understandability and usefulness of PONV shall be examined and at best improved by user studies.

References

[1] L. Freeman, “Visualizing social networks,” *Journal of Social Structure*, vol. 2000, no. Volume 1, 2000. [Online]. Available: http://www.cmu.edu/joss/content/articles/volume1/Freeman.html

[2] K. Stein and S. Blaschke, “Collaborative Intensity in Social Networks,” in *Proceedings of the International Conference on Advances in Social Network Analysis and Mining*, N. Memon and R. Alhajj, Eds. Athens: IEEE Computer Society, July 20–22 2009, pp. 60–65.

[3] ——, “Interlocking communication. measuring collaborative intensity in social networks,” in *Social Networks Analysis and Mining: Foundations and Applications*. Springer, 2010.

[4] C. Chen, *Information Visualization: Beyond the Horizon*, 2nd ed. London: Springer, 2006.

[5] L. Reeve, H. Han, and C. Chen, “Information visualization and the semeantic web,” in *Visualizing the Semantic Web: XML-Based Internet and Information Visualization*, V. Geroimenko and C. Chen, Eds. London: Springer, 2006.

[6] D. Eppstein and E. R. Gansner, Eds., *Graph Drawing. 17th International Symposium, GD 2009*, ser. LNCS, vol. 5849, 2010.

[7] M. Junger and P. Mutzel, Eds., *Graph Drawing Software*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[8] M. Ghoniem, J.-D. Fekete, and P. Castagliola, “A comparison of the readability of graphs using node-link and matrix-based representations,” in *INFOVIS ’04: Proceedings of the IEEE Symposium on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 2004.

[9] B. Shneiderman and A. Aris, “Network visualization by semantic substrates,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 733–740, 2006.

[10] N. Henry, J.-D. Fekete, and M. J. McGuffin, “Nodetrix: a hybrid visualization of social networks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1302–1309, 2007.

[11] S. Silva and T. Catarci, “Visualization of linear time-oriented data: a survey,” in *Proceedings of the First International Conference on Web Information Systems Engineering*, vol. 1, 2000, pp. 310–319.

[12] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski, “Visualizing time-oriented data. a systematic view,” *Computers & Graphics*, vol. 31, no. 3, pp. 401–409, 2007.

[13] J. Moody, D. McFarland, and S. Bender-deMoll, “Dynamic network visualization,” *American Journal of Sociology*, vol. 110, no. 4, pp. 1206–1241, 2005. [Online]. Available: http://www.journals.uchicago.edu/doi/abs/10.1086/421509

[14] T. Kamada and S. Kawai, “An Algorithm for Drawing General Undirected Graphs,” *Information Processing Letters*, vol. 31, no. 1, 1989.

[15] S. Bender-deMoll and D. A. McFarland, “The art and science of dynamic network visualization,” *Journal of Social Structure*, vol. 7, no. 2, 2006. [Online]. Available: http://www.cmu.edu/joss/content/articles/volume7/deMollMcFarland

[16] C. Ware, *Visual Queries: The Foundation of Visual Thinking*, S.-O. Tergan and T. Keller, Eds. Berlin, Heidelberg: Springer, 2005.

[17] D. A. Keim, D. A. Keim, H.-P. Kriegel, and H.-P. Kriegel, “Visdb: Database exploration using multidimensional visualization,” *IEEE Computer Graphics and Applications*, vol. 14, pp. 40–49, 1994.

[18] D. A. Keim, M. Ankerst, and H.-P. Kriegel, “Recursive pattern: A technique for visualizing very large amounts of data,” in *VIS ’95: Proceedings of the 6th conference on Visualization ’95*. Washington, DC, USA: IEEE Computer Society, 1995.

[19] M. Ankerst, S. Berchtold, and D. A. Keim, “Similarity clustering of dimensions for an enhanced visualization of multidimensional data,” in *INFOVIS ’98: Proceedings of the 1998 IEEE Symposium on Information Visualization*. Washington, DC, USA: IEEE Computer Society, 1998.

[20] D. A. Keim, “Designing pixel-oriented visualization techniques: Theory and applications,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 59–78, 2000.

[21] Y. Guo, C. Chen, and S. Zhou, “Fingerprint for network topologies,” in *Complex Sciences*, J. Zhou, Ed. Springer, 2009, pp. 1666–1677.

[22] R. Wegener, “Kollaborationsprozesse in Wikis: Entwurf und Umsetzung eines Analysewerkzeugs,” Master’s thesis, University of Bamberg, 2009.