Fail-safe Glyph Encoding based on Quasi-Hamming Distances: With a Case Study on Visualizing File System Events

Philip A. Legg, Eamonn Maguire, Simon Walton, Michael Goldsmith,
Sadie Creese, and Min Chen, *Member, IEEE*

Abstract—In many applications of spatial or temporal visualization, glyphs provide an effective means for encoding multivariate data objects. However, because glyphs are typically small, they are vulnerable to various perceptual errors. In information theory and communication, the concept of *Hamming distance* underpins the study of codes that support error detection and correction by the receiver without the need for corroboration from the sender. In this work, we propose a novel concept of *quasi-Hamming distance* in the context of glyph design. We examine the feasibility of estimating quasi-Hamming distance between a pair of glyphs, and the minimal Hamming distance for a glyph set. This measurement enables glyph designers to determine the differentiability between glyphs, facilitating design optimization by maximizing distances between glyphs under various constraints (e.g., the available number of visual channels and their encoding bandwidth). We demonstrate this concept through a case study on visualizing file system events in Dropbox and Git. Our evaluation shows that the concept of quasi-Hamming distance allows us to design fail-safe glyphs, significantly reducing the vulnerability of glyph-based visualization by empowering users to detect and correct communication error.

1 Introduction

Glyph-based visualization [40, 2] is a common form of visual design where some data records are depicted by pre-defined visual objects, which are called *glyphs*. Glyph-based visualizations are ubiquitous in modern life since they make excellent use of the human ability to learn abstract and metaphoric representations in order to facilitate instantaneous recognition and understanding. Glyphs can be used to encode variables of different data types, categorical (e.g., [24, 29]) as well as numerical (e.g., [22, 10]). However, glyphs are typically small, and are often designed with a high-degree of similarity in order to facilitate mapping consistency, semantic interpretation, learning and memorization. In many applications of spatial or temporal visualization, there are quite often a large number of small glyphs required. Hence we are particularly concerned about the *differentiability* of glyphs and potential perceptual errors in observation and exploration.

Fig. 1 shows example cases that may render some glyphs indistinguishable. Zooming-out actions in data exploration can reduce glyph size significant. For example, they could make some shapes (e.g., circle and hexagon) and textures appear similarly, while confusing the categorization of sizes (e.g., big, medium, small). Meanwhile, environmental lighting conditions and printing or photocopying facilities can cause color and greyscale degeneration. Not only would such changes make some glyphs indistinguishable, but would also confuse the association between different colors or grayscales. Whilst a dynamic legend may help alleviate the confusion about various mappings, it demands users to view the legend on a regular basis, incurring additional cognitive load in terms of the effort for the bothersome visual search and memorization of the unstable mapping keys. Other issues could also include color- or change-blindness, short- or long-sightedness, clustering, occlusion, distortion, and so on.

In the visualization literature, there are many useful guidelines that could be adopted for effective visualization [2]. Bertin [1] advised that size is not associative, hence unsuitable for encoding categorical attributes. Tools such as ColorBrewer [17] can be used to generate qualitative colormaps for effective separability of attributes. Many glyph designers apply their creative intuition to ensure the diversity and legibility of different glyphs. This poses some challenging research questions for effective glyph design, including, '*Is there a theoretical framework to encompass various design guidelines?*' and '*Is there a systematic approach to design a fail-safe glyph set?*'

In this work we propose a conceptual framework for glyph-design based on the Hamming distance (Section 3). Because of the perceptual nature of many design aspects, we introduce the notion of *quasi-Hamming distance* (QHD) (Section 4). Using this notion, we are able to translate qualitative assessment of perceptual distances in a design to Hamming distances. When the minimal Hamming distance for a glyph set is 1, the glyph set is vulnerable to the 'noise' during observation and exploration. When the minimal Hamming distance is 2, the glyph set facilitates some error detection, with which the viewer can use interaction (e.g., zooming-in, or looking at the legend) to investigate the error. When the minimal Hamming distance is 3 or more, the glyph set facilitates some error correction at the receiving end. This enables us to adjust the design to ensure a minimal Hamming distance among a set of glyphs. It is a systematic approach to optimize the design of a set of glyphs, providing fail-safe glyph encoding.

To support this novel concept for glyph-based visualization, this work includes the following additional contributions:

- We outline several methods for estimating QHD, and present two proof-of-concept of experiments for estimating QHD based on the grading by human participants and using image-comparison metrics respectively (Section 4).
- We present a case study on visualizing file system events, where glyphs were designed to facilitate error detection and correction, and were evaluated by human participants and image-comparison metrics (Section 5).
- We demonstrate the uses of the set of fail-safe glyphs to visualize event log data captured from two popular real-world file systems, namely, Dropbox and Git (Section 6). The former provides users around the world with file sharing facilities, while the latter is a distributed version control system that supports collaborative software development.

- *Philip A. Legg is with the Cyber Security Centre, University of Oxford. E-mail: phil.legg@cs.ox.ac.uk.*
- *Eamonn Maguire is with the Oxford e-Research Centre, University of Oxford. E-mail: eamonn.maguire@oerc.ox.ac.uk.*
- *Simon Walton is with the Oxford e-Research Centre, University of Oxford. E-mail: simon.walton@oerc.ox.ac.uk.*
- *Michael Goldsmith is with the Cyber Security Centre, University of Oxford. E-mail: michael.goldsmith@cs.ox.ac.uk.*
- *Sadie Creese is with the Cyber Security Centre, University of Oxford. E-mail: sadie.creese@cs.ox.ac.uk.*
- *Min Chen is with the Oxford e-Research Centre, University of Oxford. E-mail: min.chen@oerc.ox.ac.uk.*

Manuscript received 31 March 2014; accepted 1 August 2014; posted online 13 October 2014; mailed on 4 October 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

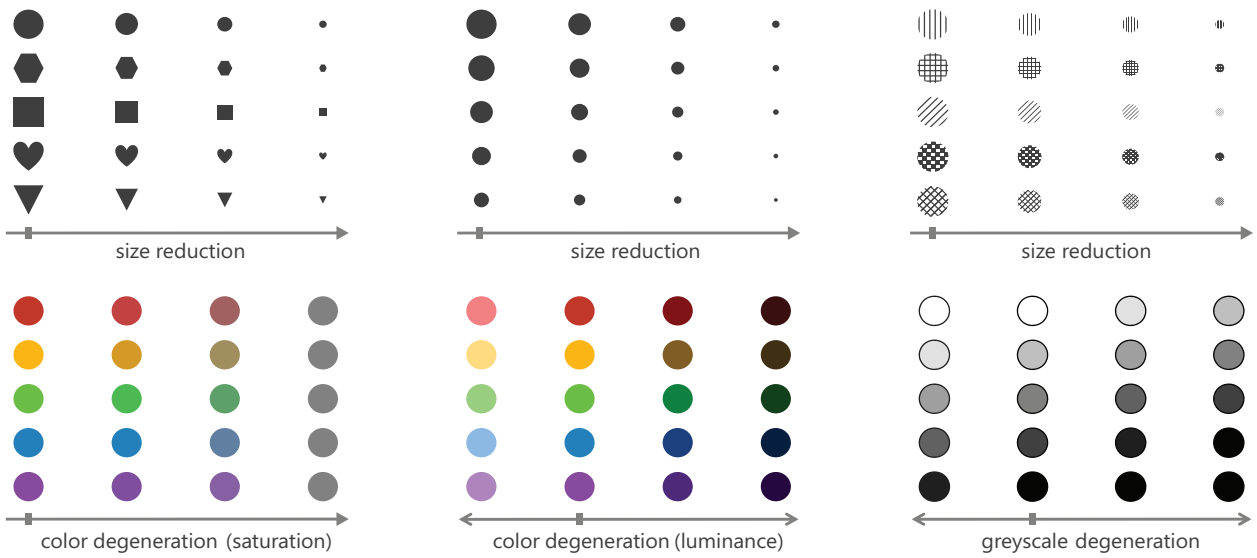

Fig. 1. Three different types of quality generation are applied to several glyphs, each of which is encoded using a single visual channel. The original quality is indicated by a marker on the x-axis. When size, saturation and luminance are changed, they become more difficult to differentiate.

2 Related Works

As the related works, we consider three topics of interest: representing hierarchical and temporal information, glyph-based and event-based visualization, and visualization separability and cognition.

Originally proposed by Johnson and Shneiderman [20], Treemaps have proven a popular technique for visualizing hierarchical data, such as file systems, using a nested rectangle approach. In the literature, there is much work that aims to extend upon this by increasing the capacity of data that can be visualized, and the addition of temporal information. Tu and Shen propose using contrast Treemaps to depict change between two or more snapshots of hierarchical data [38]. Card *et al.* introduce *TimeTree*, which allows a user to interactively browse temporal change in hierarchical data [4]. Lamping and Rao propose the use of a hyperbolic browser for visualising large hierarchical data that incorporates a fish-eye lens to provide focus and context [23]. Holten extends this approach to show hierarchical edge bundles that depict relationships between data point [18]. Burch and Diehl introduce *TimeRadarTress*, that use a radial tree layout to depict hierarchy, with associated circle sectors to also show temporal changes [3]. Therón also uses a radial layout, based on a tree-ring metaphor for depicting hierarchical data whilst also incorporating the temporal element [37]. More recently, Guerra-Gómez *et al.* introduced *TreeVersity2* for visualizing temporal changes in dynamic hierarchical data [15].

Ward [40, 41] provides a technical framework for glyph-based visualization that covers aspects of visual mapping and layout methods, as well as addressing important issues such as bias in mapping and interpretation. Borgo *et al.* [2] provide a state of the art report on glyph-based visualization that address many of the design guidelines and techniques that have been utilized in the field. Lie *et al.* [25] discuss a variety of design considerations for glyph-based visualization including data mapping, glyph instantiation, and rendering, for three-dimensional data. Event visualization aims to allow the user to understand not only that a change in time has occurred, but more specifically, to understand the semantic attributes surrounding the change event. Glyph-based visualization is popular for event-based visualization due to the capability of intuitive encoding of the event semantics, and the capacity of multivariate glyph representation. Legg *et al.* propose *MatchPad* for analyzing sports event data using glyph-based visualization [24]. Parry *et al.* also use event-based visualization for mapping temporal events and context in Snooker [30]. Kapler and Wright developed a prototype system *GeoTime* that displays military events in a combined temporal and geo-spatial visualization [21]. Gatalsky *et al.* use the similar concept of the 'space-time cube' to visualize spatio-temporal information relating to earthquake events [14]. Pearlman and Rheingans use glyphs for visualizing network security events [31]. Suntinger *et al.* [35] also use glyph-based event visualization to create an *Event Tunnel* for business analysis and incident exploration. Jänicke *et al.* [19] developed *SoundRiver* that mapped movie audio/video content to glyph visualizations on a timeline. Ware and Plumlee [42] investigate the use of glyph-based visualization for encoding multi-variate weather data such as temperate, pressure, wind direction and wind speed. Duffy *et al.* [10] use glyph-based video visualization to encode 20 different variables in a single glyph design for semen analysis. Wongsuphasawat *et al.* introduce LifeFlow as a scalable interactive overview of temporal event sequences [43]. Ferreira *et al.* visualise spatio-temporal events for assessing New York taxi trips [13]. Luo *et al.* visualise events in automated text analysis from large text collections, using their proposed system *Event River* [28]. Recently, storyboard visualization has become popular for depicting key events in temporal data such as movie content [36, 26].

Chen and Jänicke [6] proposed an information-theoretic approach that compares the process of developing visualizations with the traditional communication model. Chen [5] discusses an information-threotic viewpoint for the development of visual analytics. Duke *et al.* [11] discuss the importance surrounding interpretation, examining how viewers could potentially perceive different meaning from the same visualization. Liu *et al.* [27] distributed cognition as a theoretical frame for visualization. Van Wijk [39] discusses the value of visualization, and how it is a combination of art, science, and the real world. Chung *et al.* [8] discuss eight design principles for sortable glyphs. Particular principles that are important to our work are separability, searchability, attention-balance, and learnability. As part of our interest in separability, there are numerous image comparison metrics that have been proposed in image processing literature (e.g., [33, 12]). Such metrics have also been adopted in visualization, where Zhou *et al.* [44] studied 11 metrics for comparative visualization. Daniel and Chen [9] also studied image comparison metrics for video visualization. Recently, Schmidt *et al.* [34] developed an interactive web application for supporting image comparison.

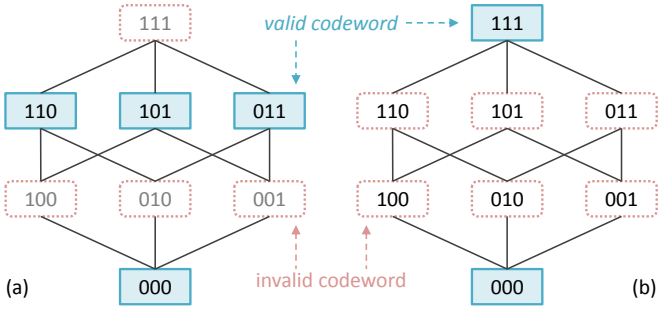

Fig. 2. Two 3-bit codes. (a) A code can detect 1-bit errors. (b) A code can detect 2-bit errors and correct 1-bit errors.

Fig. 3. Two examples that illustrate the phenomena of error detection and error correction in glyph-based visualization. (Above) A viewer may sense that the glyph on the left may not be correct in a distorted visualization, and consult the legend to correct the error. (Below) A viewer may unconsciously perceive the glyph on the left as a star shape due to gestalt effects and a priori knowledge about the glyph set.

3 Hamming Distance

In information theory and data communication, a *code* consists of a finite set of *codewords*, each of which is a digital representation of a letter in an alphabet. In the context of binary encoding, *Hamming distance*, proposed by Richard Hamming in 1950 [16], is a measure of the number of bit positions in which two codewords differ. Considering all pairs of codewords in a code, the minimal distance is referred to as the minimal Hamming distance of the code. (In the literature, the word minimal is often confusingly omitted.) In communication, there are two main strategies for handling errors that occur during transmission. *Automated error detection* allows the receiver to discover that any error has occurred and to request a retransmission accordingly. *Automated error correction* enables the receiver to detect an error and deduce what the intended transmission must have been. Hamming defined the following principle:

Theorem. A code of $d+1$ minimal Hamming distance can be used to detect d bits of errors during transmission. A code of $2d+1$ minimal Hamming distance can be used to correct d bits of errors during transmission [16].

For example, given a 3-bit code as illustrated in Fig. 2, there are 8 possible codewords. One may select a subset of these codewords to construct a code with its minimal Hamming distance equal to 2 bits or 3 bits. Fig. 2(a) shows one of such codes, which has 4 codewords and is of 2 bits Hamming distance. This code can detect 1-bit errors since any change of a valid codeword by 1 bit would result in an invalid codeword, which would lead the receiver to discover the error. Fig. 2(b) shown another code, with 2 codewords and is of 3 bits Hamming distance. It can detect 2-bit errors and correct 1-bit errors. When a valid codeword (e.g., 111) is changed by 1 bit during transmission (e.g., 110), the receiver can detect such an error and recover the intended codeword based on the nearest neighbor principle. Of course, if a 2-bit error occurred during transmission, the receiver would be able to detect the error but could not make a correct 'correction'. Nevertheless, if it is known that 2-bit errors are likely to occur then this should either be used as only an error detection code, or a code with a longer Hamming distance should be used instead.

4 Quasi-Hamming Distance for Glyph Design

A set of glyphs is a code, and each glyph in the set is a valid codeword. During visualization, there can be errors in displaying or perceiving a glyph. If a viewer can detect that a perceived glyph is not quite 'right', conscious or unconscious effort can be made to correct such an error. Conscious effort, which is an analogy of error detection and repeated transmission, may typically include zooming in to have a close look, or consulting the legend. Unconscious effort, which is an analogy of error correction, may include some gestalt effects [7], and inference from other visual information [32]. Fig. 3 shows two example glyph sets, each with 8 codewords. Given the two display errors depicted on the left, i.e., an arrow glyph is skewed in a distorted printout and a shape glyph is occluded by another shape, one can detect both errors easily. The error with the arrow glyph may need some conscious effort, and that with the shape error can usually be corrected unconsciously. This suggests that it is possible to establish a conceptual framework, similar to Hamming distance, for error detection and error correction in glyph-based visualization.

However, understandably, measuring the distances and errors in visual perception is not as simple as measuring those represented by binary codewords. We thereby propose an approximated conceptual framework based on the principle of Hamming distance, and we call it *Quasi-Hamming Distance* (QHD). The term 'quasi' implies that the distance measure is approximated, and the quantitative measure of perceptual errors is also approximated. The main research questions are thereby (i) whether we can establish a measurement unit common to both measures, and (i) how we can obtain such measurements.

The answer to the first research question is that we can utilize 'bit' as the common unit for both distance and error measurement. Let us first consider an ordered visual channel, such as brightness or length, as a code C. Theoretically C can have a set of codewords $\{c_1, c_2, \ldots c_n\}$ such that the difference between two consecutive codewords is the just-noticeable difference (JND) of this visual channel. We can define the QHD between each pair of codewords c_i and c_j as $|i-j|$ bits. During display and visualization, if c_i is mistaken for c_j, we can call this a d-bit error where $d = |i-j|$. Now let us extend this concept to a less ordered visual channel (e.g., hue) or an integrated channel (e.g., color). Theoretically, we can construct a code C by uniformly sampling the space of the visual channel (e.g., the CIE L*a*b* color space) while ensuring that every pair of samples differ by at least the JND of this channel. These codewords, i.e., samples, can be organized into a network, where the distance between any two codewords can be approximated proportionally according to the JND (i.e., JND = 1 bit). Note that the possible perception error rate with a code that maximizes the number of codewords based on JND is likely to be very high. In practice, one designs a glyph set based only on a small subset of samples in a visual channel or more commonly in the multivariate space of several visual channels. Hence a QHD measure based on JND would be too fine to use in practice, though in a longer term, JND can provide an *absolute reference measure* once we have obtained such measures for most visual channels in visualization.

This leads to the second research question, i.e., given a glyph set, how can we measure the distance between glyphs? One may consider using the following methods:

1. **Estimation by expert designers.** This practice has always existed in designing exercises such as for traffic signs and icons in user interfaces. To formalize this practice, designers can explicitly estimate and label the distance between each pair of glyphs in a glyph set. While this approach may be most convenient to the designers, its effectiveness depends very much on the experience of the designers concerned and it is rather easy to overlook certain types of display and perception errors.

2. **Crush tests.** One can simulate different causes of errors, such as those illustrated in Fig. 1, and determine at which level of degeneration glyphs may become indistinguishable. The corresponding level of degeneration can be defined as QHD. While this approach would yield more consistent estimation of QHD, more research would be required to compile a list of different

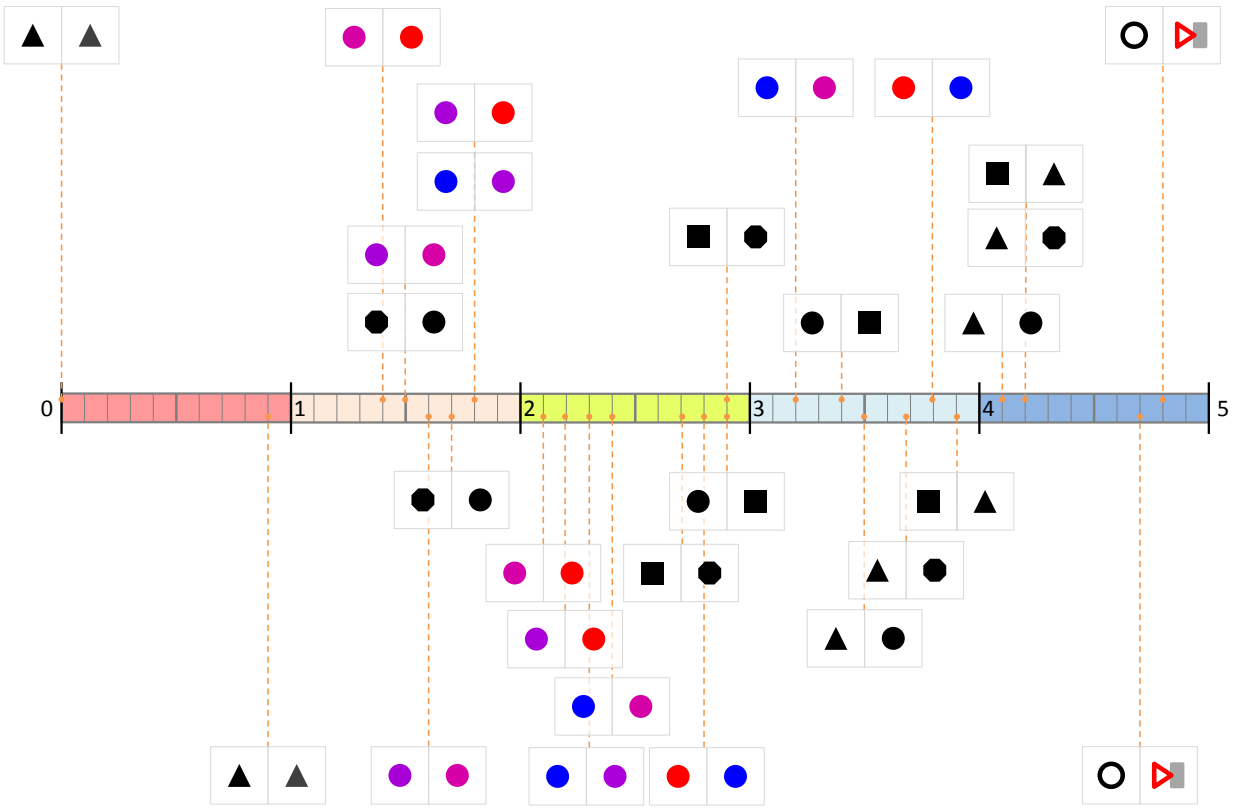

Fig. 4. Sample of results for the primitive glyph pairs used in both the user-centric estimation (top) and by computer-based similarity measure (bottom). Low values indicate difficult to differentiate and high values indicate easy to differentiate.

causes of errors and define coherent levels of degeneration across different causal relationships.

3. **Task-based evaluation.** Similar to (2), one can simulate different visualization conditions, enlist users to perform their tasks, measure users' performance, and transform performance measures to QHD. On the one hand, this approach is perhaps most semantically meaningful for a particular glyph set in a specific application context. On the other hands, the performance measures collected may feature many confounding effects, while there may only be a small number of users available for such an evaluation.
4. **User-centric estimation.** One may conduct a survey among human participants about how easy or difficult to differentiate different glyphs. By removing task-dependency in (3), more participants can be involved in such a survey, yielding more reliable estimation of QHD.
5. **Computer-based similarity measures.** There are a large collection of image similarity measures in the literature [33, 12]. In a longer term, it is likely that we will be able to find measures that are statistically close to user-centric estimation, though there is not yet a conclusive confirmation about optimal image similarity measures, and there are hardly any metrics specially designed for measuring similarity of glyphs.

To demonstrate the feasibility of estimating QHD, we conducted two proof-of-concept experiments based on methods (4) and (5). We conducted a survey among 20 participants, all of whom are either employees or students at University of Oxford. About 50% encountered glyph-based visualization in their course or work. The results of one participant were considered as an outlier and were not included in the statistics. After a brief introduction by one co-author of this paper, the participants were asked to rate how difficult or easy to differentiate 104 pairs of glyphs in an integer scale from 0 to 10. The survey was conducted during an informal lunch, where pizzas were served. No cash payment was involved.

The 104 stimuli pairs were divided into three main categories, 8 *reference pairs*, 48 *primitive pairs*, and 48 *application-specific pairs*. We will discuss the last category in Section 5 in detail. The 96 primitive and application-specific pairs were mixed together in a randomized order. The 8 reference pairs were placed at positions 1, 2, 35, 36, 69, 70, 103 and 104 for helping participants to regularize their scores and for enabling us to check temporal consistency. The details about the questionnaire, the stimuli grouping and the survey results can be found in the supplementary materials. Here we briefly describe the survey results in relation to the reference and primitive pairs.

The category of reference pairs are divided into two groups. Group A consists of 4 pairs of very similar glyphs, and Group B consists of 4 pairs of very different glyphs. We expected that participants will assign very low scores (difficult to differentiate) to those in A and high scores (easy to differentiate) to those in B respectively. As mentioned early, we place one pair from A and one from B at regular intervals. The average scores for the 4 pairs in group A are (0.0, 0.4, 1.4 and 2.8) respectively and those for group B are (9.3, 9.0, 8.9, 9.7) respectively, indicating that they have statistically served as references for the minimal and maximum QHD in this survey.

The category of primitive pairs consists of 8 groups for estimating QHD in relation to 8 visual channels, namely hue, shape, components, connection lines, luminance, size, texture, and orientation. Each group has 6 pairs of stimuli, facilitating pairwise comparison of 4 different codewords of each channel. For hue and luminance channels, after choosing the 1st and 4th codewords we used a perceptually uniform color model (Hunter's Lab) to determine the 2nd and 3rd codewords at 50% and 75% distance from the 1st. The upper part of Fig. 4 shows a small selection of survey results, where we converted the $[0, 10]$ score

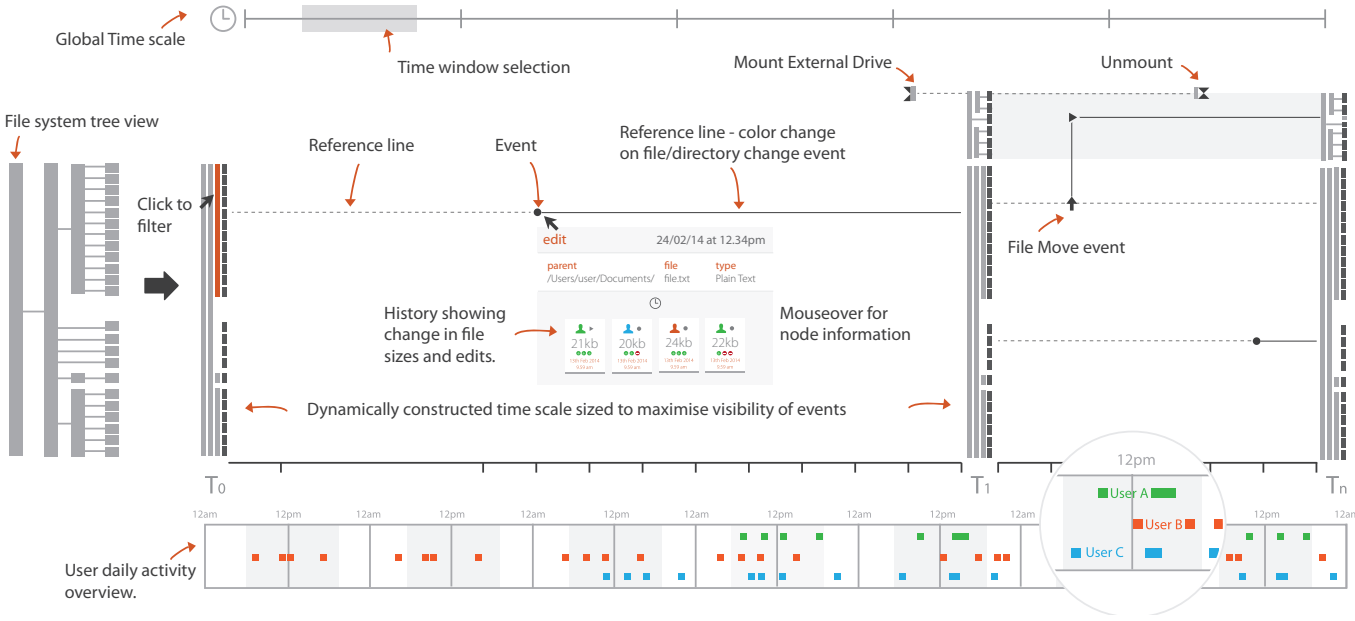

Fig. 5. Visualization overview. We use a timeline approach with a condensed file system hierarchy at the left of the visualization. File system activities are depicted by glyphs on the timeline, with connected lines to show correspondence to the file system. Keyframes are displayed to depict significant changes to the file system hierarchy, or at user-specified time intervals. Interation with mouse cursor displays context box with detailed information on the selected file, directory, or activity.

range to a $[0,5]$ QHD range, and we considered that any QHD < 2 is potentially risky for error detection, and any QHD < 3 is potentially risky for error correction.

In our second experiment, we measured the similarity between each pair of glyphs using a computer-based metric. The metric combines difference of pixel colors and that of spatial occupancy. The former captures a variety of feature differences such as colour, luminance, size, and orientation, etc. and is defined as the mean Euclidean distance between all corresponding pixels in the two images representing the pair of glyphs. The latter captures some location-invariant features such as spatial occupancy and additional components, and is defined as the difference between the numbers of pixels with $\leq 80\%$ luminance. Both difference measures are first normalized to the $[0,1]$ range, and are then scaled to the same QHD range as the survey (with the same min, mean, max), before being combined into a single metric. The lower part of Fig. 4 shows the computed similarity measures for the same selection of stimuli pairs.

5 Case Study: Visualizing File System Events

The problem of visualizing file systems plays a significant role in the short history of computer-assisted visualization. In 1991, Johnson and Shneiderman, who were motived by the need to visualizing the structure of a file system, published their seminal paper on treemaps [20]. Today, not only are file systems much larger and contain many more files, they are also shared by many more users and have many more events. One important aspect of a file system is to support collaborative activities, such as sharing files within multi-partner projects and developing software by a team of programmers. While there are text-based mechanisms for recording events in relation to a file system or a specific folder, the amount of data contained in typical log files can easily escalate to the point where it becomes too overwhelming for anyone to read on a regular basis until perhaps some disastrous events take place. To our knowledge, there is no effective visualization technique for allowing users of such collaborative environments to observe events in a cost-effective manner.

In this case study, we designed and developed a novel glyph-based visualization tool for observing events in a file system. There are several technical challenges. Firstly, the hierarchical nature of the file system needs to be depicted so that the spatial context of where a particular event has occurred can be identified. Secondly, the temporal information about events needs to be conveyed so that the activity ordering can also be observed and reasoned. Thirdly, there are a wide range of activities (e.g., copying a file, modifying a file) that are typically performed, which would need to be distinguishable in a visualization. Finally, the visualization should be able to support collaborative environments by depicting activities from different users.

Fig. 5 presents an overview of the visualization and the design process that was adopted for the layout. The visualization layout consists of a number of different components that we shall discuss. The central area of the visualization is the main activity window where the events performed on the file system are displayed on a timeline using glyph-based visualization. The glyphs that have been designed for this application are discussed in detail in the next section. To the left of the main activity window is the file system tree view, which represents the file system using a traditional tree representation where directories are shown as light gray nodes and files shown as dark gray nodes. Since most directories are likely to contain files, leaf nodes are typically file objects. This tree is shown as a condensed display to the left of the main timeline and serves as a reference to spatial context of the file location. File system events in the main window are positioned in correspondence to the file system hierarchy. Connected lines are displayed to relate the file activities back to the particular file in the hierarchy. For events that involve a change in the file system hierarchy, such as copying or moving files, these connecting lines also indicate the new position of the file. The condensed file hierarchy also serves as a 'keyframe', where the current state of the file system hierarchy is shown, either after a significant change to the hierarchy (e.g., copying files, or mounting an external drive), or at a particular time interval specified by the user. Finally, the file hierarchy can also be used to select the directory of interest that should be shown on the visualization. This provides a mechanism for 'zooming in' to a particular directory, or 'zooming out' to view the root directory. At the top of the display is a time window selection bar, that allows the user to specify the time period that should be shown in the main activity window. At the bottom of the display is a daily activity overview that provides a summary of which users have perform some event and at what time. The interface

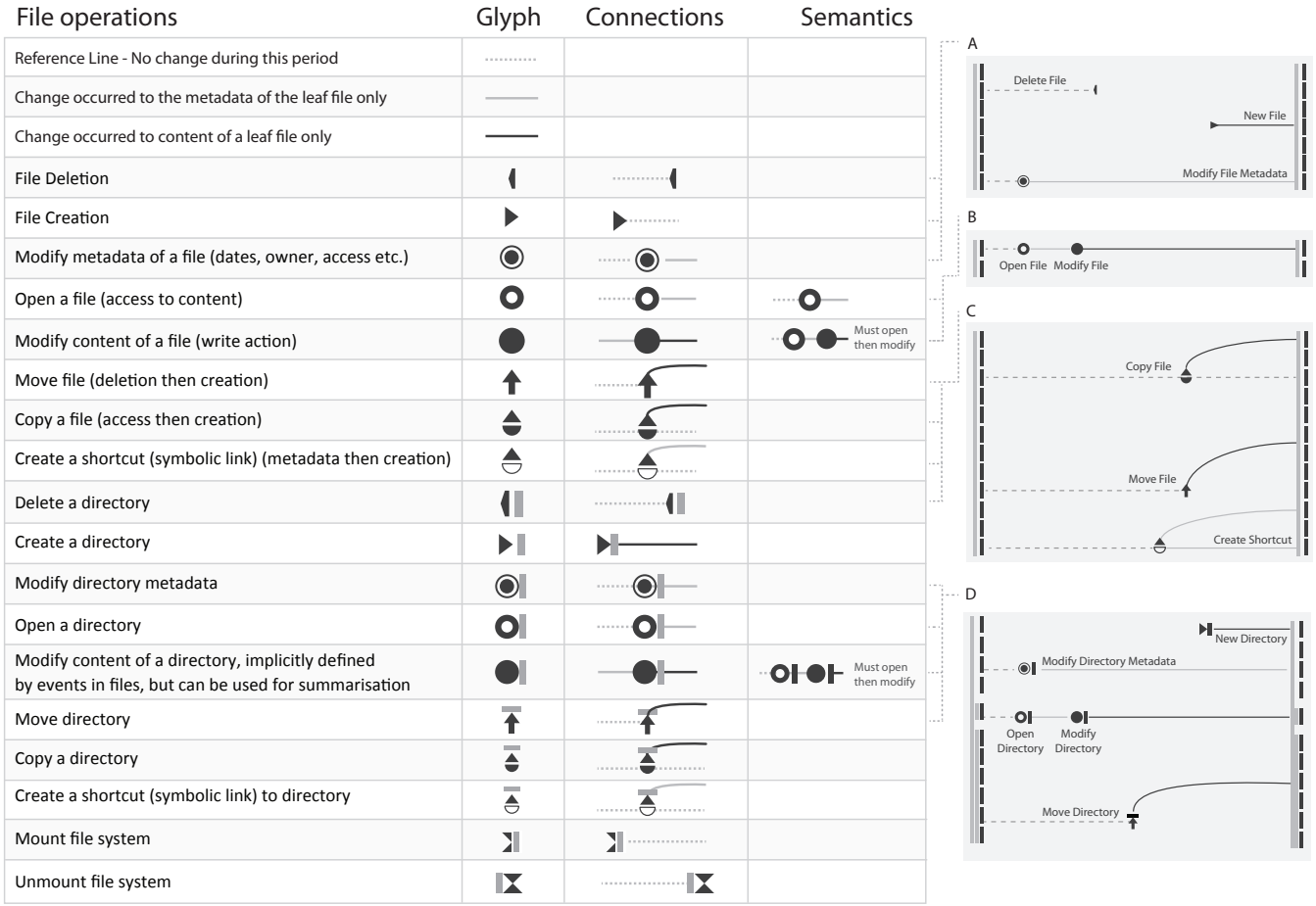

Fig. 6. The 18 glyphs designed to represent different events in file systems. Each event is associated with its primary glyph representation in the second column. In addition, an event may be associated with special signatures in terms of connection (in the third column) and semantic ordering (in the fourth column).

also supports a pop-up window that is displayed as the mouse hovers over a particular event. The window provides more information that is related to both the current event, and the file.

5.0.1 Designing Event Glyphs

The concept of QHD has been considered throughout the design, development, evaluation and application of the aforementioned glyph-based visualization tool. Our understanding and appreciation of the concept have improved along with this process. Fig. 6 shows 18 glyphs for most common events in file systems. These events include creation, modification, deletion, copying, moving and renaming. The action may be applied to a file, a directory, a device, a shortcut (symbolic link), or meta-data. The designs of these event glyphs were evolved in several stages.

Initial Design. We first designed a set of glyphs in conjunction with the overall visual design of the visualization tool as shown in Fig. 5. This allowed us to appreciate how these glyphs may be used, and what are the typical display conditions such as glyph sizes, density, and available visual channels. It was at this stage when we decided that the basic glyph designs should not feature the hue channel, and reserve this intuitive and powerful visual channel to depict user-specific or data-specific variables.

Expert Estimation. Four visualization researchers took part in this research, and all had publications in areas of glyph-based visualization. We used our knowledge about different visual channels and our experience in glyph designs to improve the original designs. This is similar to method (1) in Section 4. We noticed that although we could reach agreement as to how easy or difficult it can be to differentiate pairs of glyphs, we could not easily agree on the reasons why. When we explicitly tried to determine the QHD between a pair of glyphs, we were often influenced by many different features, component shapes, convexity, aspect ratio, curvature, and so on. This experience partly led us to appreciate more the multi-faceted nature and the complexity in estimating QHD. For most of the glyphs in Fig. 6, their designs became stabilized at this stage.

Crush Tests. We applied crush tests to all glyphs designed during the case study. In several cases, we carried out systematic testing by applying consistent zooming factors to all glyphs. More often, when we were considering individual glyphs, we carried out ad hoc crush tests by using facilities in our drawing software, such as zooming, and overlaying a translucent shape on top of glyphs. At this stage, we realized that simulating different conditions that would cause glyph quality to degenerate was not a trivial undertaking. In many ways, this also echoed the multi-faceted nature and the complexity in estimating QHD as mentioned above.

Human-centric Estimation. We conducted a survey, partly to gain a better understanding about QHD in the context of individual visual channels (see Section 4), and partly to evaluate our event glyphs in Fig. 6. We considered 20 different glyph designs, for which there would be 190 pairwise comparisons. We selected 48 pairs that were considered potentially more risky than other pairs. We found that only 1 pair scored below 2 bits in terms of QHD in the survey. The final designs of the glyphs did not include this pair. The details of this evaluation will be given in Section 5.0.2.

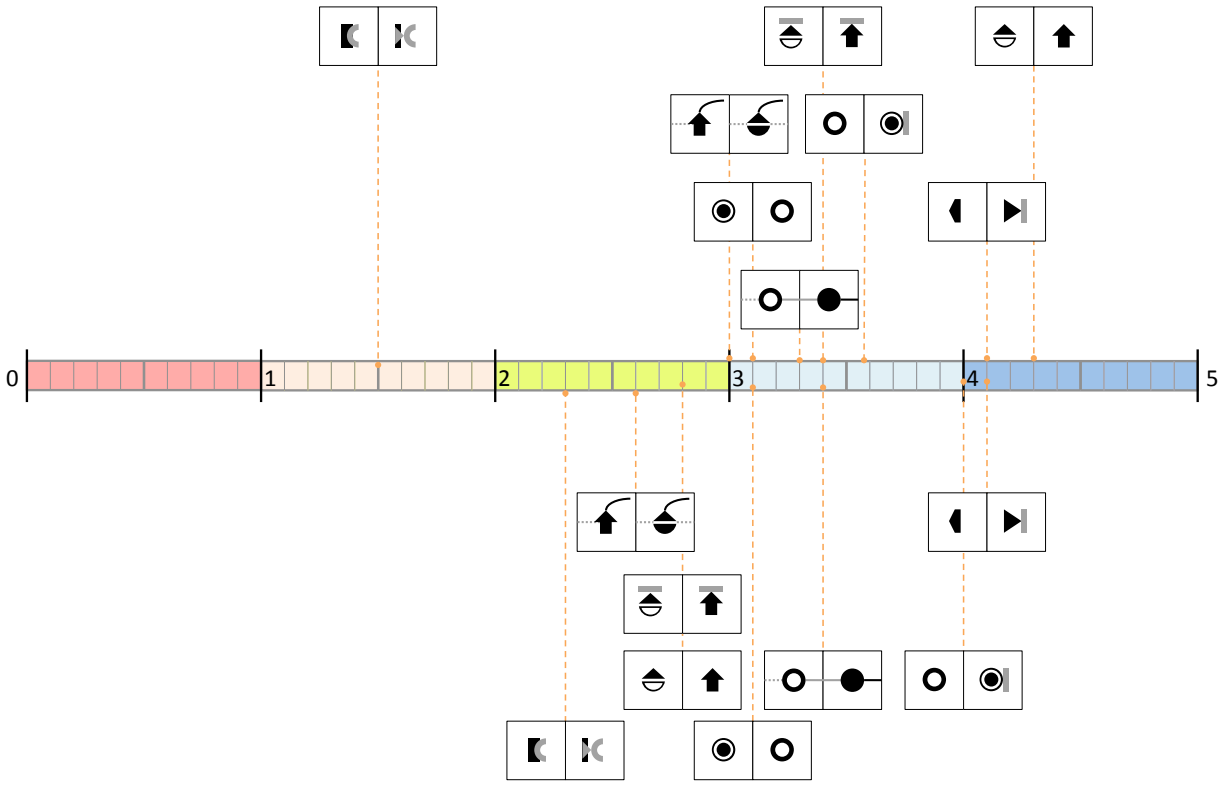

Fig. 7. Sample of results for the file visualization glyph pairs used in both the user-centric estimation (top) and by computer-based similarity measure (bottom). Low values indicate difficult to differentiate and high values indicate easy to differentiate.

Computer-based Similarity Measures. We used the same metric as mentioned in Section 4 to measure the QHD of the 48 pairs that might be potentially risky. We found that they all passed this QHD test. The details of this evaluation will be also given in Section 5.0.2.

Deployment in Software. In addition to the above design effort based on the concept of QHD, we incorporated the glyph set into the visualization tool and used the tool to visualize events in a Dropbox folder and a Git repository. This allowed us to gain direct experience about how these glyphs might be viewed and interpreted in practical applications. The details of this deployment will be discussed in Section 6.

Differentiability is only one aspect of glyph design. We have to consider other aspects such as how easy it is to learn and to remember glyphs, how glyphs may be connected, and how they may be ordered if the corresponding events happened to the same file or directory. As shown in Fig. 6, we utilized some similar designs for files and directories to assist in learning and memorization. Meanwhile, we also consider how they may be connected. The three types of connection lines as shown on the top of the figure and the different orientations as shown in the third column may potentially add additional features for differentiating glyphs. For example, all lines connecting to a *deletion* glyph will always come from left, and all connecting to a *creation* glyph will always extend towards right. All lines connecting to a *copy* or *move* glyph will suggest a spatial shift vertically. In addition, semantic ordering, such as *open* before *read*, may also add addition QHD to the glyph designs as illustrated on the right side of Fig. 6.

5.0.2 Evaluating Event Glyphs

We evaluated those glyphs in Fig. 6 based on the QHD obtained from a human-centric survey and by using computer-based similarity measures. We utilized the same survey and metrics as described in Section 4 because this allowed us to compare the QHD for our glyph set against that for the reference pairs and primitives discussed in Section 4. Note that for our glyphs only the potentially risky pairs were evaluated.

The human-centric estimation provided us with most meaningful insight about the quality of the glyphs. The 104 pairs of glyphs evaluated by participants have an average QHD of 2.9 bits. The average QHD for the reference pairs (Groups A and B) is 2.7 bits. The average for the primitive pairs (Groups C, D, E, F, G, H, I, J) is 2.6 bits. The average for the potentially risky pairs in our glyph set (Groups O, P, Q, R, S, T, U, V, W, X, Y, Z) is 3.2 bits. Almost all of our glyph pairs have their QHD above 2 bits, except one pair (QHD = 1.5 bits) which was not used in the final design. The upper part of Fig. 4 shows a selection of the survey results.

Meanwhile, the computer-based metric also measured our glyph pairs favourably. As mentioned in Section 4, the average QHD estimated by the metric is normalized to have the same min, mean and max as the human-centric estimation. The complete set of 104 glyph pairs have an average QHD of 2.9 bits The average QHD for the reference pairs is 2.6 bits. The average for the primitive pairs is 2.7 bits. The average for the potentially risky pairs in our glyph set is 3.1 bits. The lowest QHD for our potentially risks glyph pairs is 2.1 bits.

The evaluation also showed some interesting phenomena. The additional features added to the directory glyphs have reduced QHD among directory glyphs. For example, when comparing Group O and Group Q, where the glyphs for *creation*, *modify metadata* and *modify content* were compared within the context of files and directories respectively, human-centric estimation shows a noticeable difference. The average QHD for Group O (files) is 3.4 bits and that for Group Q (directories) is 2.6 bits. Similarly, for Group T and Group V where *move*, *copy* and *short cut* glyphs were compared, the average QHD for Group T (files) is 3.3 bits, and that for Group V (directories) is 2.8 bits. Meanwhile, the computer-based similarity measures suggest little difference between O and Q and between T and V. This suggests that further research is necessary to enrich the existing findings in percep-

tion about the distance functions for integrated and separable visual channels [29].

6 Example Applications: Dropbox and Git

To demonstrate the applicability of the above glyph encoding scheme, we developed an interactive visualization tool for visualizing event log data captured from two real world systems, namely Dropbox[1] and Git[2], both of which are popular for collaborative file usage.

Our glyph-based visualization software is comprised of two parts: (a) a back end for processing commit logs of Dropbox and Git, and (b) a front-end serving as a web-based user interface as well as a software library. The back end was written in Python. It generates two types of JSON (JavaScript Object Notation) files for the front-end, one describing the directory structure and the other describing the events that occurred. The front-end was created with a combination of HTML5, CSS and JavaScript (utilising Raphaël.js[3] for the visualization element and jQuery for control of popup events). The glyphs are stored in a font file created using the IcoMoon[4] service.

In addition to glyph-based visualization, the system supports a variety of interactions including:

- Filtering different types of file system events;
- Filtering different users;
- Selecting a specific directory as a subtree;
- Selecting different time period;
- Zooming, and scrolling along both the directory axis and the time line;
- Showing a detailed pop up window when the mouse hovers over an event glyph.

6.1 Visualizing Dropbox Activity Log

Dropbox is a popular cloud service that allows users to synchronise their files across multiple devices. It allows users to create shared directories that other users can be invited to access. This makes it especially useful for collaborative activities between institutions and colleagues, and for sharing personal media with family and friends. Since the Dropbox service was founded in 2007, its user base has grown to 200 million in 2013. Because of the file sharing capability, it is desirable for users to visualize events in a shared folder, for instance, to see which file has been created or modified recently and by whom. Although the service does provide a text-based activity log that users can access, it is time-consuming, and to some extent, tedious to read a long list of events.

As shown in Fig. 8, glyph-based visualization allows users to gain an overview of the events in a shared folder at ease. As discussed previously, we purposely avoided using colors in our glyph design. This provides the visualization system with the flexibility to encode context-sensitive information, such as different users, or different types of files. In Fig. 8, colors are used to depict different users. We can observe that three different users have who have accessed the system during this time (shown by the blue, orange, and green glyphs).

The three vertical bars are simplified views of the directory tree concerned. It is a fairly large directory with three further levels of subdirectories. The period displayed spans over 39 hours, and a variety of events took place. Some indicate close collaboration, when a line of activities linked different users to a single file. For example, the top line in the left half section shows that the orange user created a file, and then opened and modified it. After several hours, the blue user modified the metadata of the file (possibly renaming it or changing its access date). Several hours later, the green user opened and modified it. If a viewer wishes inspect an event in detail, or simply forget the meaning of a glyph, he/she can simply hovers the mouse over the glyph and a pop-up window will displays details about the event and the file or directory concerned. The detailed information includes the filename, the parent directory, and the file type. It also shows file-specific history, such as the access that each user performed and the file size at each access. With this, the collaborative activities become more transparent. Collaborative colleagues can become aware of ongoing actions, reducing the needs for informing each other of every file access actions using emails.

[1] http://www.dropbox.com
[2] http://git-scm.com
[3] http://raphaeljs.com
[4] http://icomoon.io

6.2 Visualizing Git Repository History

Git is a distributed version control system, allowing programmers to develop software in a collaborative manner. It was first released in 2005, and became popular among programmers after the launch of web-based hosting services such as GitHub. The main file storage is known as the repository, and users can pull from, or push to, the repository from their local version of the Git file storage. Although Git maintains a comprehensive history of file access and revision for a repository, it is time-consuming for Git users to read the history data of a Git repository. Glyph-based visualization can offer an efficient means to gain a quick overview of such history data.

Fig. 9 shows an example visualization of a shared repository between two of the co-authors of this work. As both users in this collaboration had a good understanding of the joint software development project, they only need a quick glance at the visualization to sense of the ongoing programming effort by each other. In comparison with reading the repository history, the visualization provides a much more efficient and effective means for their 'silent communication' and 'autonomous coordination'. It is also easy for any one of the co-developers to know major actions, such as deleting a file. From Fig. 9, one can quickly identify two file deletion actions, one around 17:00 on Monday and one around 17:00 on Tuesday.

7 Conclusions

In this paper, we have presented a novel conceptual framework, called *quasi-Hamming distance* (QHD), which facilitates a systematic approach to designing fail-safe glyph encoding schemes. As the conceptual framework is built on the well-proven theories and practice in communication, it offers the potential to stimulate further research on this topic with a depth and a breadth comparable to the topic of error detection and error correction in communication. To demonstrate the feasibility of this conceptual framework, we presented two proof-of-concept experiments, where we obtained QHD measures from a human-centric survey and from computer-based similarity measures. To demonstrate its practical applicability, we conducted a case study where event logs from Dropbox and Git were visualized using a set of fail-safe glyphs. From the very beginning, the glyph set was designed to ensure a high-level of differentiability, while accommodating the necessary requirements such as minimal uses of colors and metaphoric consistency. The glyphs were evaluated using QHD measures obtained from a human-centric survey and computer-based similarity measures.

We very much consider this as the first step towards the establishment of a collection of mathematical and cognitive theories, experimental findings and statistics, design techniques and computational metrics for guiding and aiding glyph designs. This work highlights a number of gaps, where further research is needed. For example, it is highly desirable for us to understand the relationship between the JND measures of various visual channels and differentiability of glyphs encoded using such visual channels. It is also highly desirable to research into computer-based similarity measures that are statistically closer to human-centric estimation, or even better to collected task performance measures.

References

[1] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[2] R. Borgo, J. Kehrer, D. H. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen. Glyph-based visualization: Foundations, design guidelines, techniques and applications. In *Eurographics State of the Art Reports*, EG STARs, pages 39–63, May 2013.

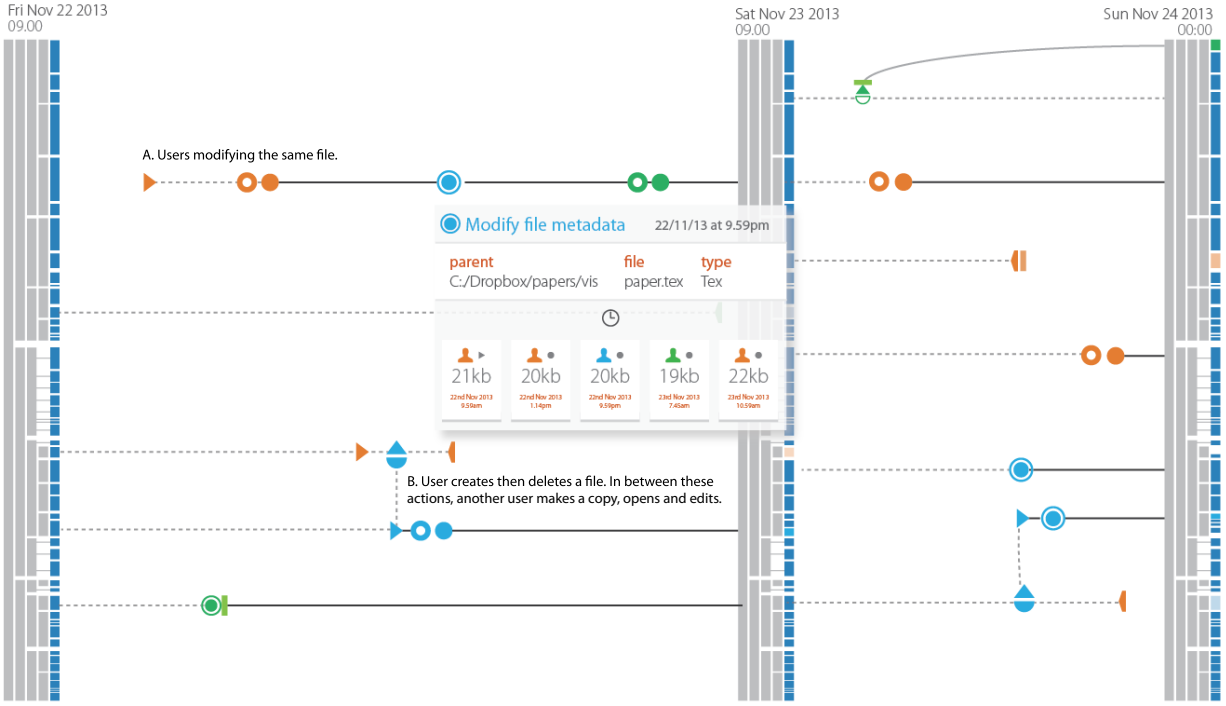

Fig. 8. Glyph-based visualization is used to display events in a Dropbox activity log. The vertical bars are abstract representation a directory tree and the timeline flows from left to right. At (A) we can see the case where a number of users have been modifying the same file. The popup gives more details about the modifications, who did them and when to allow for provenance tracking. At (B) we have another interesting case where *user X* (orange) creates a file, then *user Y* creates a copy of this file, opens it and modifies its contents. *User X* then deletes the original file, however a modified copy exists elsewhere.

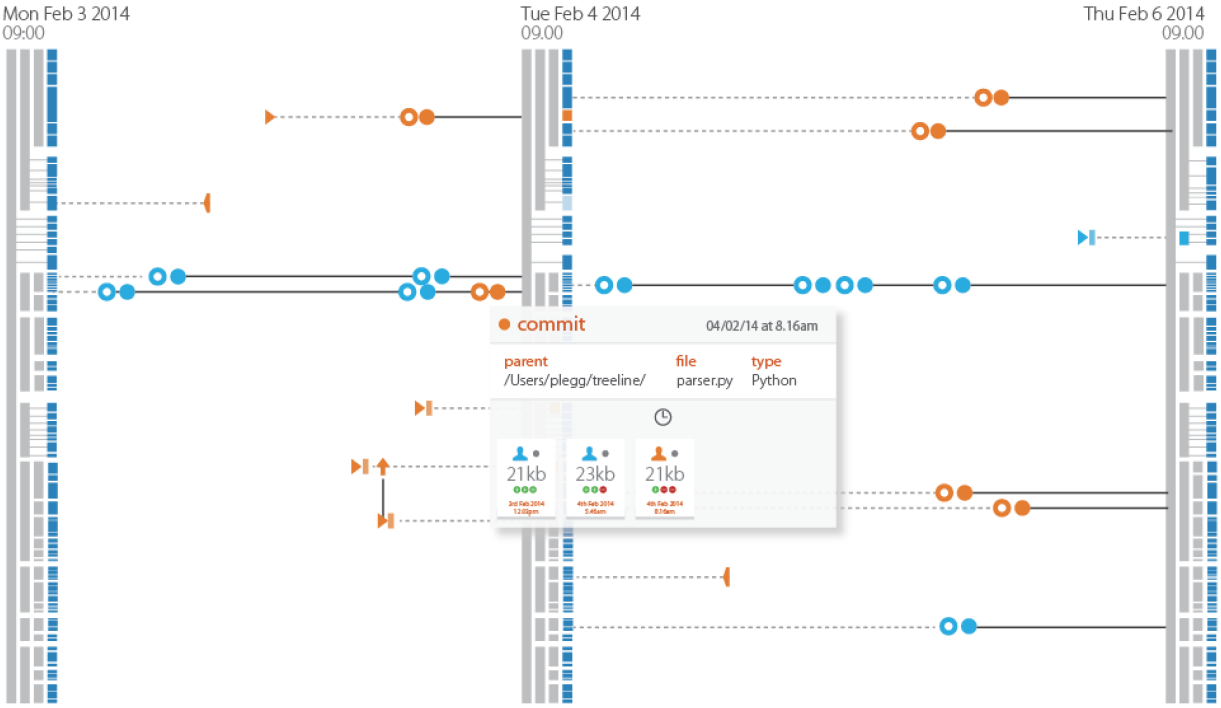

Fig. 9. Glyph-based visualization is used to display events in the history data of a Git repository, where two co-developers were actively working on different parts of the software. They distributed their effort in a coordinated manner, so as to avoid conflicts and problems when merging code. Only on one occasion, *user X* (orange) has modified a file in an area of the repository normally occupied by *user Y* (blue). We can also see that *user X* has been most active when it comes to creating and deleting directories/files compared to *user Y* who has been more active in terms of editing and committing files.

[3] M. Burch and S. Diehl. TimeRadarTrees: Visualizing dynamic compound digraphs. *Computer Graphics Forum*, 27(3):823–830, 2008.

[4] S. Card, B. Suh, B. Pendleton, J. Heer, and J. Bodnar. Time Tree: Exploring time changing hierarchies. In *Proc. IEEE VAST*, pages 3–10, 2006.

[5] C. Chen. An information-theoretic view of visual analytics. *IEEE Computer Graphics and Applications*, 28(1):18–23, 2008.

[6] M. Chen and H. Jaenicke. An information-theoretic framework for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1206–1215, 2010.

[7] M. Chen, S. Walton, K. Berger, J. Thiyagalingam, B. Duffy, H. Fang, C. Holloway, and A. E. Trefethen. Visual multiplexing. *Computer Graphics Forum*, 33(3), to appear, 2014.

[8] D. H. Chung, P. A. Legg, M. L. Parry, R. Bown, I. W. Griffiths, R. S. Laramee, and M. Chen. Glyph sorting: Interactive visualization for multi-dimensional data. *Information Visualization*, to appear, 2014.

[9] G. Daniel and M. Chen. Video visualization. In *Proc. IEEE Visualization*, page 54, 2003.

[10] B. Duffy, J. Thiyagalingam, S. Walton, D. J. Smith, A. Trefethen, J. C. Kirkman-Brown, E. A. Gaffney, and M. Chen. Glyph-based video visualization for semen analysis. *IEEE Transactions on Visualization and Computer Graphics*, to appear, 2014.

[11] D. J. Duke, K. W. Brodlie, D. A. Duce, and I. Herman. Do you see what I mean? *IEEE Computer Graphics and Applications*, 25(3):6–9, May 2005.

[12] D. Eler, M. Nakazaki, F. Paulovich, D. Santos, M. Oliveira, J. Neto, and R. Minghim. Multidimensional visualization to support analysis of image collections. In *Proc. SIBGRAPI*, pages 289–296, 2008.

[13] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual exploration of big spatio-temporal urban data: A study of New York city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013.

[14] P. Gatalsky, N. Andrienko, and G. Andrienko. Interative analysis of event data using space-time cube. In *Proc. IEEE Information Visualisation*, pages 145–152, 2004.

[15] J. Guerra-Gomez, M. L. Pack, C. Plaisant, and B. Shneiderman. Visualizing change over time using dynamic hierarchies: TreeVersity2 and the StemView. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2566–2575, 2013.

[16] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[17] M. Harrower and C. A. Brewer. *ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps*, pages 261–268. John Wiley and Sons, Ltd, 2011.

[18] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[19] H. Jänicke, R. Borgo, J. S. D. Mason, and M. Chen. SoundRiver: Semantically-rich sound illustration. *Computer Graphics Forum*, 29(2):357–366, 2010.

[20] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proc. IEEE Visualization*, pages 284–291, 1991.

[21] T. Kapler and W. Wright. Geotime information visualization. In *Proc. IEEE Information Visualization*, pages 25–32, 2004.

[22] G. Kindlmann and C.-F. Westin. Diffusion tensor visualization with glyph packing. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1329–1336, 2006.

[23] J. Lamping and R. Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7(1):33–55, 1996.

[24] P. A. Legg, D. H. S. Chung, M. L. Parry, M. W. Jones, R. Long, I. W. Griffiths, and M. Chen. MatchPad: Interactive glyph-based visualization for real-time sports performance analysis. *Computer Graphics Forum*, 31(3):1255–1264, 2012.

[25] A. E. Lie, J. Kehrer, and H. Hauser. Critical design and realization aspects of glyph-based 3D data visualization. In *Proc. SCCG*, pages 27–34, 2009.

[26] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. Storyflow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2566–2575, 2013.

[27] Z. Liu, N. Nersessian, and J. Stasko. Distributed cognition as a theoretical framework for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1173–1180, 2008.

[28] D. Luo, J. Yang, M. Krstajic, W. Ribarsky, and D. Keim. EventRiver: Visually exploring text collections with temporal references. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):93–105, 2012.

[29] E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and M. Chen. Taxonomy-based glyph design with a case study on visualizing workflows of biological experiments. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2603–2612, 2012.

[30] M. L. Parry, P. A. Legg, D. H. S. Chung, I. W. Griffiths, and M. Chen. Hierarchical event selection for video storyboards with a case study on snooker video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1747–1756, 2011.

[31] J. Pearlman and P. Rheingans. Visualizing network security events using compound glyphs from a service-oriented perspective. In *Proc. VizSEC 2007*, Springer Mathematics and Visualization, pages 131–146. 2008.

[32] P. Rheingans and C. Landreth. Perceptual principles for effective visualizations. In *Perceptual Issues in Visualization*, pages 59–74. Springer-Verlag, 1995.

[33] N. Sahasrabudhe, J. West, R. Machiraju, and M. Janus. Structured spatial domain image and data comparison metrics. In *Proc. IEEE Visualization*, pages 97–515, 1999.

[34] J. Schmidt, M. E. Gröller, and S. Bruckner. Vaico: Visual analysis for image comparison. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2090–2099, Dec. 2013.

[35] M. Suntinger, J. Schiefer, H. Obweger, and M. E. Groller. The event tunnel: Interactive visualization of complex event streams for business process pattern analysis. In *Proc. IEEE Pacific Visualization*, pages 111–118, 2008.

[36] Y. Tanahashi and K.-L. Ma. Design considerations for optimizing storyline visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2679–2688, 2012.

[37] R. Therón. Hierarchical-temporal data visualization using a tree-ring metaphor. In *Smart Graphics*, Springer LNCS, volume 4073, pages 70–81, 2006.

[38] Y. Tu and H.-W. Shen. Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1286–1293, 2007.

[39] J. van Wijk. The value of visualization. In *Proc. IEEE Visualization*, pages 79–86, 2005.

[40] M. O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3/4):194–210, 2002.

[41] M. O. Ward. *Multivariate Data Glyphs: Principles and Practice*. Springer Handbooks Comp. Statistics. 2008.

[42] C. Ware and M. D. Plumlee. Designing a better weather display. *Information Visualization*, 12(3-4):221–239, 2013.

[43] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: visualizing an overview of event sequences. In *Proc. ACM/SIGCHI CHI*, pages 1747–1756, 2011.

[44] H. Zhou, M. Chen, and M. Webster. Comparative evaluation of visualization and experimental results using image comparison metrics. In *Proc. IEEE Visualization*, pages 315–322, 2002.