Journal of Universal Computer Science, vol. 11, no. 11 (2005), 1820-1834
submitted: 1/9/05, accepted: 1/10/05, appeared: 28/11/05 © J.UCS

Integrating Lite-Weight but Ubiquitous Data Mining into GUI Operating Systems

Li Wei
(University of California, Riverside, USA
wli@cs.ucr.edu)

Eamonn Keogh
(University of California, Riverside, USA
eamonn@cs.ucr.edu)

Xiaopeng Xi
(University of California, Riverside, USA
xxi@cs.ucr.edu)

Stefano Lonardi
(University of California, Riverside, USA
stelo@cs.ucr.edu)

Abstract: Most visualization tools introduced in the literature are specialized for a particular task. In this work, we introduce a novel framework which allows visualization to take place in the background of normal day to day operations of any GUI based operating system such as MS Windows, OS X or Linux. Our system works by replacing the standard file icons with automatically generated icons that reflect the contents of the files in a principled way. We call such icons Intelligent Icons. While there is little utility in examining an individual icon, examining groups of them provides a greater possibility of unexpected and serendipitous discoveries. The utility of Intelligent Icons can be further enhanced by arranging them on the screen in a way that reflects their similarity/differences. We demonstrate the utility of our approach on data as diverse as DNA, text files, electrocardiograms, and Space Shuttle telemetry. In addition we show that our system is unique in also supporting fast and intuitive similarity search.

Keywords: data mining, visualization, icon
Categories: H.3.0, H.3.3, H.3.4

1 Introduction

At the heart of many information visualization and data mining techniques is a single question "*compared to what?*" [11]. In several application domains, the main objective of data exploration is to arrange the data such that meaningful similarities and differences are exposed. However the vast majority of visualization/data mining tools introduced so far are specialized pieces of software that are explicitly run on a particular dataset at a particular time for a particular purpose. The human effort involved in this process is high enough that most of these tools are used rarely, even when data keeps accumulating at very high rates.

In this work we introduce a novel framework which allows lite-weight visualization and data mining to take place in the background of quotidian computer activities of any GUI based operating system such as MS Windows, OS X or Linux. This enables a greater possibility of unexpected, serendipitous and useful discoveries.

Our system works by replacing the standard file icons with icons that reflect the contents of files in a principled way. We call such icons INTELLIGENT ICONS. While there is little utility in examining an individual icon, examining groups of them allows us to take advantage of *small multiples* paradigm elucidated by Tufte. We can enhance the utility of INTELLIGENT ICONS by arranging them on the screen in a way that reflects their similarity/differences, rather than the traditional "view by date", "view by size" etc. As we will demonstrate, our approach has utility for data as diverse as DNA, text, and time series.

The rest of the paper is organized as follows. We conclude Section 1 with a discussion of related work. Section 2 introduces our ideas on a single data type, DNA. In Section 3 we generalize these ideas to other types of data. Section 4 contains demonstrations and experiments. Finally in Section 5 we discuss future directions.

1.1 Prior and Related Work

Our work is closest in sprit to the recent VisualIDs work of Lewis et. al. [7]. The authors create distinctive icons for files by hashing the file names to seeds of a pseudorandom generator that in turn is used to create a shape grammar. In this way, similar filenames will map to similar shapes.

Figure 1 is a simple example which shows the difference between VisualIDs and INTELLIGENT ICONS. There are three ASCII text files, each of which contains approximately 16,000 base pairs of mitochondrial DNA. We used string edit distance as suggested in [7] to measure the distance between file names, and Euclidean distance to measure the distance between the file icons (as explained in more detail later). Note that two of the species share the same specific name of "*americanus*" (with a different generic name) and this makes them similar in a way that is not biologically meaningful, whereas the INTELLIGENT ICON approach captures the correct relationship between the three species.

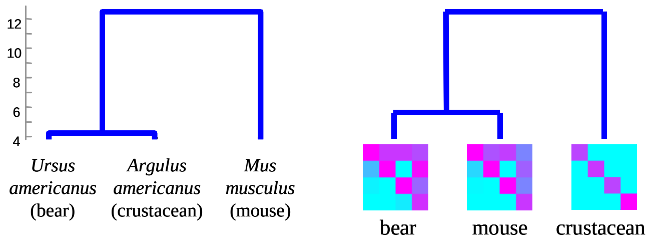

Figure 1: The similarity of three DNA files based on name (left) and content (right)

The idea of using the values of variables to change the shape of an icon (glyph) dates back at least to the classic work of Chernoff [4]. Beddow and others extended this mapping to colors [3]. Keim et. al. introduced *Recursive Patterns* in [1]. Recursive patterns can be considered as a general technique to map data to bitmaps, although icons were not explicitly considered.

2 An Example of an Icon Generation Algorithm

For concreteness we begin with a particular example before considering the general framework.

2.1 DNA to Intelligent Icon

A DNA string is a very long sequence of symbols drawn from the alphabet {**A**, **C**, **G**, **T**}. For example the human mitochondrial DNA has 16,571 such symbols, beginning with **GATCACAGGTCTATCACCCTATTAACCACT**.

Although the rich literature on the problem of classifying DNA sequences contains very sophisticated approaches, here we pursue a very simple technique based on the frequency of short substrings. First we divide a bitmap into four quadrants and count the frequency of each of the four possible base pairs. Then we map the observed frequencies to a linear colormap and produce an icon by filling each section of the bitmap with the corresponding indexed color, as shown in Figure 2.

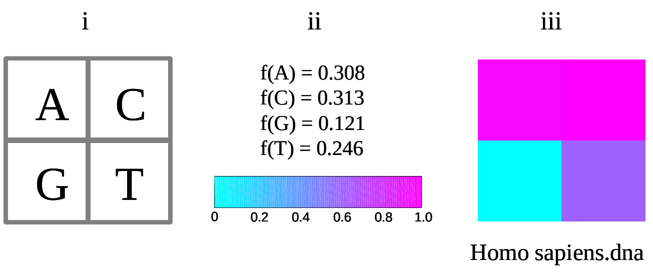

Figure 2: Illustration of the file icon generation for DNA

Note that in this case both the arrangement of the four letters and the choice of colormap are arbitrary. In order to use as much of the color spectrum as possible, we normalize the data such that the symbol with lowest frequency maps to zero and the symbol with highest frequency maps to one. More concretely, if *j* is one symbol in the alphabet, then the color index of *j* is denoted as *ci*(*j*), and calculated as:

$$ci(j) = (f(j) - \min[f(A), f(C), f(G), f(T)]) / \max[f(A), f(C), f(G), f(T)] \qquad (1)$$

We apply this simple mapping to DNA sequences of different mammals. Unsurprisingly however there is very little difference between the icons obtained. To improve the discrimination ability of the icons we use more features. Below we show a general mapping for DNA that has a potentially useful property.

We begin by assigning each letter a unique key value, *k*:

A $\rightarrow$ 0	**C** $\rightarrow$ 1	**G** $\rightarrow$ 2	**T** $\rightarrow$ 3

We use *l* to represent the length of the DNA words. Each word has an index for the location of each symbol, for clarity we show them explicitly as subscripts. For example, the first word with *l* = 4 extracted from the human mitochondrial DNA is $\mathbf{G}_0\mathbf{A}_1\mathbf{T}_2\mathbf{C}_3$. So in this example we would say k_0 = **G**, k_1 = **A**, k_2 = **T** and k_l = **C**. To map a word into a bitmap we use the following equation to find its row and column values:

$$col = \sum_{n=0}^{l-1} (k_n * 2^{l-n-1}) \bmod 2^{l-n}, \quad row = \sum_{n=0}^{l-1} (k_n div 2) * 2^{l-n-1} \tag{2}$$

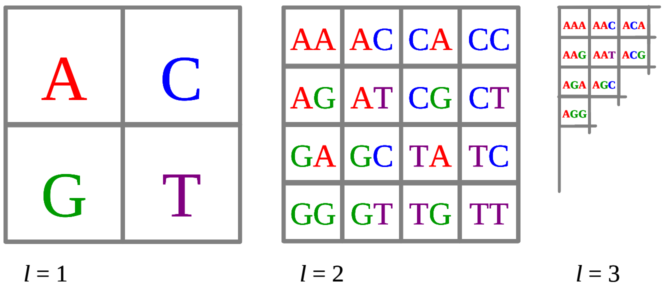

Figure 3: The mapping of DNA words of l = 1, 2 and 3

Figure 3 shows the mapping for l = 1, 2 and (part of) 3. Note that bitmaps generated this way might be self-similar across different scales, as shown in Figure 4.

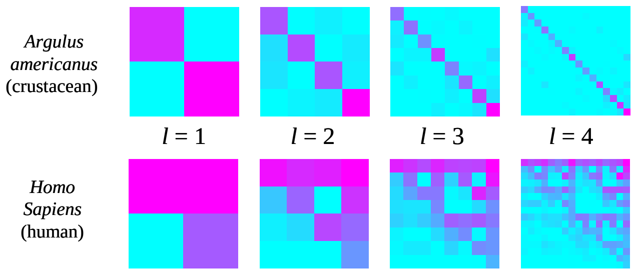

Figure 4: The icons created for two species at each level from 1 to 4

2.2 Optimizing and Arranging the Icons

We measure the similarity of icons by the Euclidean distance between their frequency counts matrices. The distance between two matrices A and B, of the same level l, is

$$dist(A, B) = \sqrt{\sum_{i=1}^{2^l} \sum_{j=1}^{2^l} (A_{ij} - B_{ij})^2} \tag{3}$$

In Figure 5 we have clustered five familiar species based on the Euclidean distance between their bitmap representations.

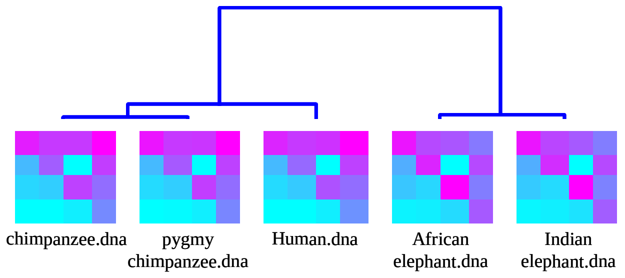

Figure 5: Five species clustered by the distance between their bitmap representations

Although the clustering is objectively correct, the differences are very subtle to the naked eye. For example the bottom right element of all five icons shown in Figure 5 appears to be minor variations of blue violet. This motivates us to enhance the subjective visual discriminatory power of the icons by normalizing the (ith, jth) element across all icons. In Figure 6 left, normalization has emphasized the differences among the bottom right element of all five icons. At this point, we finally see a hint of the potential utility of INTELLIGENT ICONS. Imagine we encountered the icon shown in the right of Figure 6. Simply by glancing at all the file icons we might guess that this animal is more similar to the chimps/human than to the elephants. In fact, this *is* the case, Macaca mulatto is commonly known as the rhesus monkey.

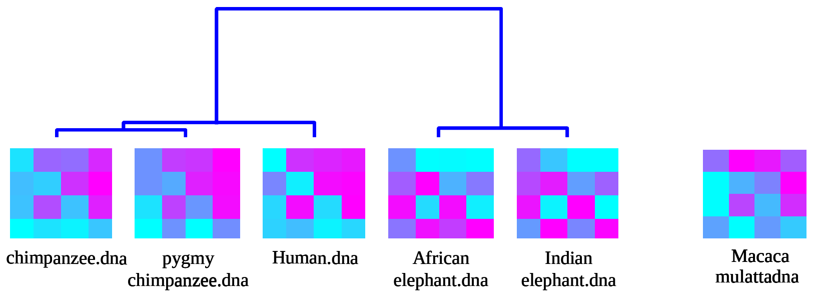

Figure 6: Left) Five species clustered by the distance between their normalized bitmap representations. Right) The icon for another African mammal

We can further leverage off the INTELLIGENT ICONS by arranging them within a file browser based on their similarity. By way of contrast consider the classic file browser interaction shown in Figure 7. Using the bounding box section tool, it is hard to extract *meaningful* subsets.

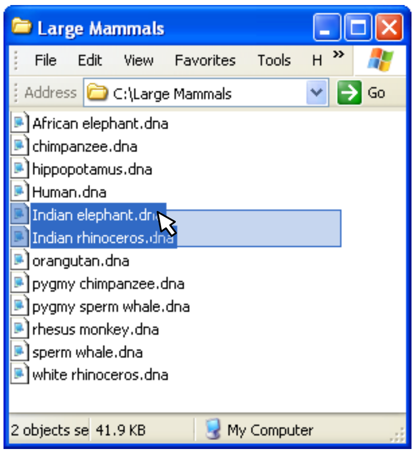

Figure 7: Twelve DNA files, sorted by name, in a typical file browser

We can use INTELLIGENT ICONS to solve this problem by arranging the icons in the file browser based on their *similarity*. Here we adopt Multi-Dimensional Scaling (MDS) and the "snap-to-grid" technique suggested by Basalaj [2] to arrange the icons. Figure 8 shows 12 mammals being arranged in this way. Using standard bounding rectangles, we can select several logical groups, such as both types of Rhinos (*Rhinocerotidae*), both types of elephants (*Elephantidae*), etc. We call the combination of INTELLIGENT ICONS and the MDS layout a *Smart Browser*.

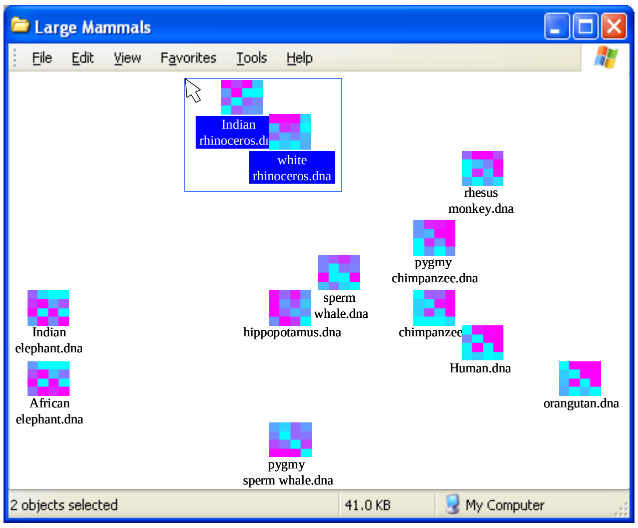

Figure 8: Twelve DNA files, arranged by Intelligent Icons, in a typical file browser

3 Generalizing from the DNA Example

We have seen a concrete example of INTELLIGENT ICONS and their utility. We want to have a software tool that is capable of changing the individual icons of selected file types and allows arranging the file icons by similarity. The user must create or download plug-ins that tell our software how to convert their file types.

Below we consider plug-ins for text, time series, and metadata and provide general guidelines for arbitrary data types. Let us begin by considering the desirable properties of INTELLIGENT ICONS.

3.1 Desirable Properties of INTELLIGENT ICONS

Below we list four desirable properties of INTELLIGENT ICONS:

- File types should retain distinctiveness. In current operating systems, most file types (e.g., PDF, PowerPoint, etc.) have a particular icon associated with them. This makes it easy to determine the file type at a glance.
- Similar files should have similar icons. This is the fundamental property which allows users to spot clusters, duplicates and outliers in their data.
- File icons should look similar at different resolution (cf. Figure 4). This is because most operating systems allow user to view icons at various sizes.
- File icon updates should be fast. It is important that files can be added, deleted or edited, and have their icons instantaneously reflect their content.

3.1.1 Distinctiveness of file type

There is little doubt that having distinctive icons for different file types aids rapid file navigation. We can retain file distinctiveness while allowing individuality by a combination of two techniques: 1) Using different colormaps for different file types; 2) Using different mappings for different file types. Figure 9 shows an example.

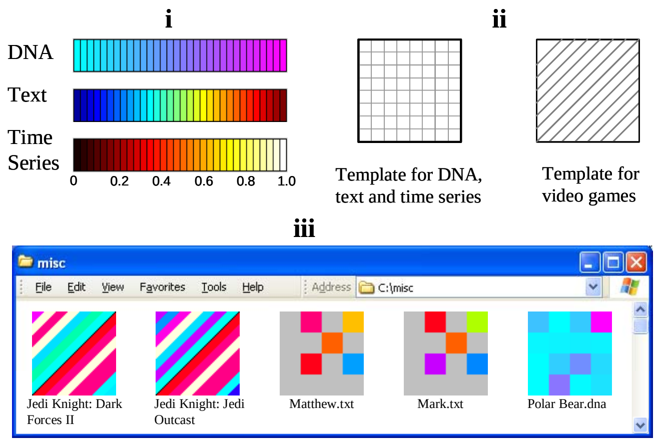
i
ii
DNA
Text
Time Series
0
0.2
0.4
0.6
0.8
1.0
Template for DNA, text and time series
Template for video games
iii
misc
File Edit View Favorites Tools Help
Address C:\misc
Jedi Knight: Dark Forces II
Jedi Knight: Jedi Outcast
Matthew.txt
Mark.txt
Polar Bear.dna

Figure 9: i) Colormaps used for different file types. ii) Examples of different mapping templates. iii) A screen capture of a folder with three different file types

3.1.2 Similar files should have similar icons

The basic idea discussed in Section 2 of extracting features from the file, measuring their frequency, and mapping these frequencies to color and spatial arrangements can be easily applied to other domains. We provide some general guidelines in this section.

Text: Files containing text, such as MS Word, PDF, TEX, TXT files etc. are perhaps the most commonly encountered file types for the majority of people. To map these files to icons, we first discard stop words, such as "*the*", "*of*", "*and*" etc. Such words tend to have equal frequency across all documents and thus have little discriminative power. Next we stem the words using Porters algorithm [9], so that variations on a word map to a single root, for example "divid*ing*", "divid*ed*" and "divid*e*" all map to "divid". Since the number of words left is still much greater than the number of pixels available, we use a classic text-processing algorithm called Latent Symantec Indexing (LSI) to reduce the dimensionality of the features.

Time Series: Time series are a ubiquitous and increasingly prevalent type of data. There is some existing work on visualizing time series that could be adapted for our needs. For example the Recursive Pattern work of Ankerest et. al. [1] allows recursive generalization of arbitrary line and column oriented arrangements. Another possibility is to discretize time series and use the approach for text, or to discretize the time series into *exactly* four symbols and use the algorithm for DNA. Here we consider the later approach in more detail.

We adopt the SAX technique of Lin et. al. [8] to convert real valued time series into discrete symbols. The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. Each section is then substituted by its mean value. This representation is then discretized in such a manner as to produce a word with approximately equi-probable symbols. Figure 10 shows a relatively short time series being converted into a pseudo DNA word of 8 symbols.

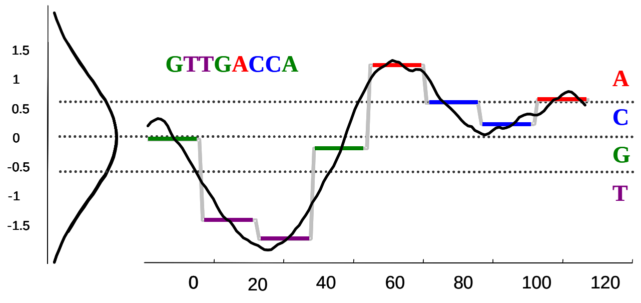

Figure 10: A real valued time series being discretized into a SAX word

Metadata: It is extremely difficult to extract useful features from many file types, including executables, music and video files. Fortunately, many such file types can be mapped to extensive repositories of metadata. For example, we create icons for MP3 music files based not on the file contents, but on metadata provided (automatically) by CDDB.com. The features available include, Track Artist, Record Label, Year, Beats Per Minute etc. For video games, there is no completely automatic metadata server, but an hour's work enabled us to write a crawler which extracted features from www.metacritic.com/games/pc/scores/.

3.1.3 File icon at different resolution should look similar to itself

File icons should look similar when viewed at different scales because most operating systems allow user to view icons at different resolutions. In some cases this "self-similar" property can be easily arranged. For example in Figure 4 our mapping for DNA has this property, and our mapping function for time series inherits it.

More generally, this property may be hard to ensure if we wish to use every pixel of say a 48*48 bitmap. When we reduce the size of this bitmap to 24*24, we must average the quartets of pixels into one. If the original pixels elements are independent (a general requirement cf. section 3.1.2), the smaller bitmaps will not resemble the larger bitmaps. The good news is that it is unlikely we would use all 2,304 pixels of the largest icon size. Decades of research in machine learning and information retrieval strongly suggests that although objects may exist in very high dimensional spaces, meaningful similarity can best be captured in some low dimensional subspace. We therefore restrict ourselves to some small number of features, typically less than one hundred, and map each feature to several contiguous pixels in the smallest bitmap. The larger sizes bitmaps can then be obtained by simple linear extrapolation. Figure 11 shows how we combine variable level mappings and simple linear extrapolation for the DNA file icons. The smallest icon is a level 2 mapping of one feature to 4 pixels; the next size up is simply an enlargement of the smallest size. The 32*32 size icon is a level 3 mapping of one feature to 4 pixels, and the largest icon is simply an enlargement of the second largest size.

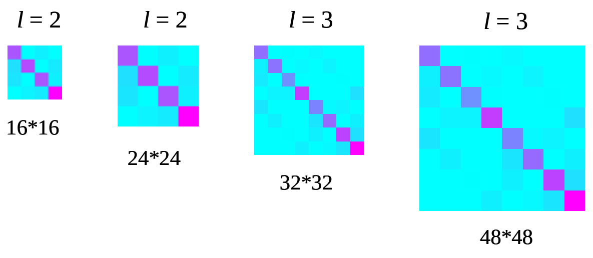

Figure 11: Four different sized DNA icons for Argulus americanus

3.1.4 File icon updates should be fast

In general, if we only need to process a few files to create their INTELLIGENT ICONS, time complexity might not be an issue. However the issue of time complexity does become important if the mapping algorithm requires access to multiple files.For example, we have shown that DNA icons look better if we normalize the frequencies across all icons. This means that every update (deletions, insertions, and editing changes) to our files should be accompanied by an update to all icons. These updates could become unacceptably slow if we have many files.

Our solution is to use a classic idea in the database community, *lazy updates* [6]. The basic idea is to learn the best mapping on all *N* files offline and use it to create icons for all *N* files. If we later add a new file to the collection, we simply use the current mapping function to immediately create the new icon, and wait for an opportunity to create the optimal icons for all *N* + 1 icons.

4 Experimental Evaluation of INTELLIGENT ICONS

The central claim of our paper is that INTELLIGENT ICONS allow unexpected and serendipitous discoveries. This is difficult to prove in anything but an anecdotal way.

We begin by using *Smart Browser* to browse the hundreds of datasets in the UCR archive [5]. One such dataset, known as Kalpakis_ECG, contains 18 normal ECGS. Figure 12 shows the dataset as most people have viewed it.

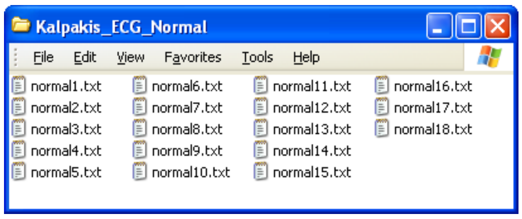

Figure 12: Kalpakis_ECG dataset shown in a typical Window XP file browser

When we glanced at this dataset with our *Smart Browser*, as shown in Figure 13, we immediately noticed that five of the 18 thumbnails had radically different icons.

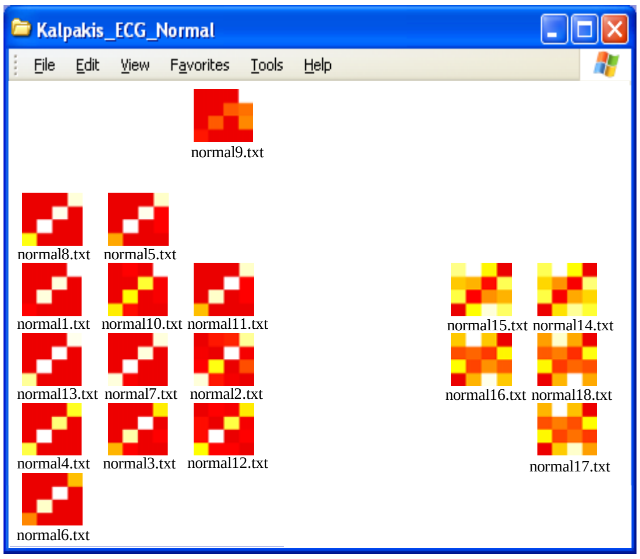

Figure 13: Kalpakis_ECG dataset shown in a Smart browser

This structure was so unexpected that we asked UCLA cardiologist, Dr. Helga Van Herle to explain these findings. She informed us that the 5 recordings in question are not ECGs! They are in fact examples of the action potential of a normal pacemaker cell (not to be confused with the man-made devices which mimic them, and are named after them). Figure 14 illustrates the difference.

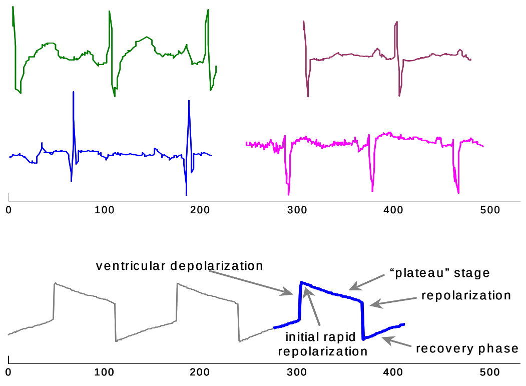

Figure 14: Top) Four snippets randomly chosen from Kalpakis ECGs. Bottom) A snippet from the "normal18.txt" ECG

Another dataset we examined was a NASA dataset containing examples of telemetry from a Space Shuttle valve. Figure 15 shows five such time series.

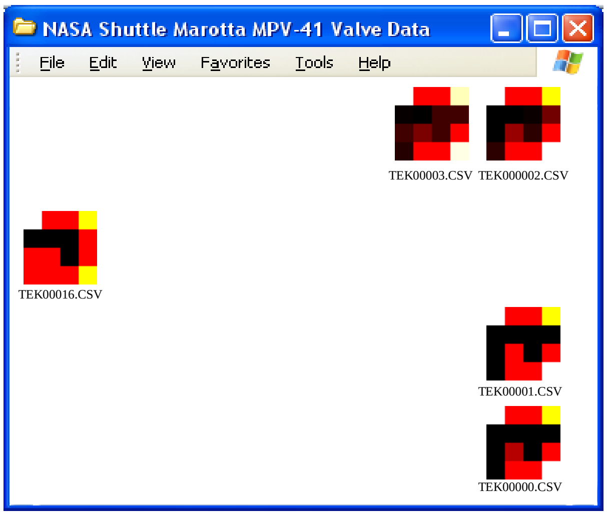
NASA Shuttle Marotta MPV-41 Valve Data
File Edit View Favorites Tools Help
TEK00003.CSV
TEK000002.CSV
TEK00016.CSV
TEK00001.CSV
TEK00000.CSV

Figure 15: Five NASA Marotta MPV-41 valve trace files shown in a Smart Browser

It is immediately apparent that one file has a quite different structure to the rest. NASA engineers explained the difference: while the other four files are normal sequences, file TEK00016.CSV is an abnormal trace, as shown in Figure 16.

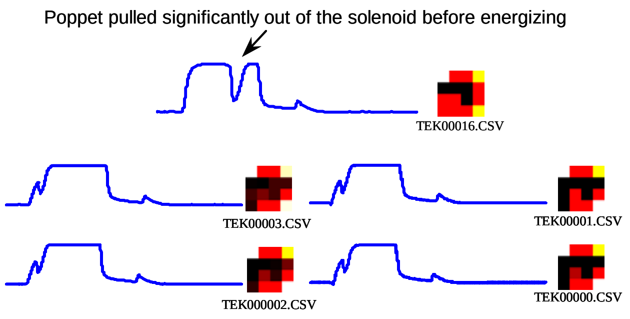

Figure 16: The five time series whose Intelligent Icons are shown in Figure 15

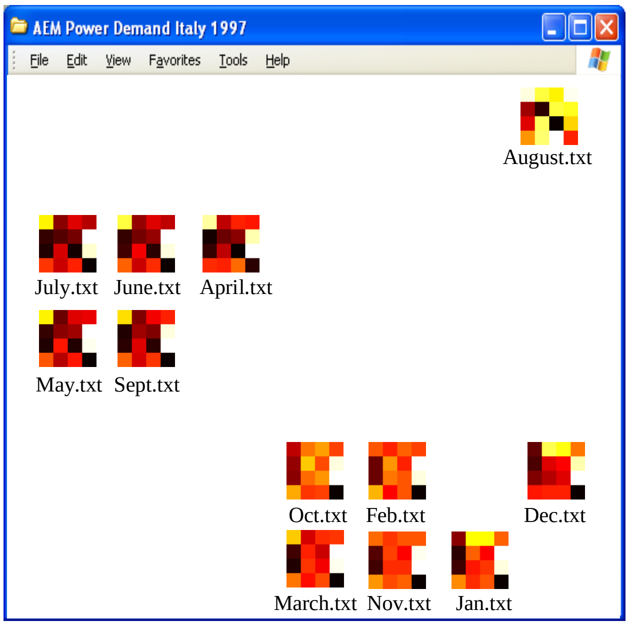

Figure 17: Twelve monthly power demand time series shown in a Smart Browser

As a final example we consider twelve monthly electrical power demand time series from Italy. Figure 17 shows the data viewed in a *Smart Browser*. It is immediately apparent that there are two major clusters that correspond to winter months and summer months. Such a division makes sense. Given that the demand for heating dominates the winter power demand (Air conditioning is still fairly rare in Italy). The other obvious observation is that the month of August is an outlier. To get some insight into this phenomenon we visualize the entire year as a single time series as in Figure 18. Clearly the month of August is a true outlier, but what is going on?

The answer lies in an Italian cultural phenomenon. According to travel writer Nella Nencini, "*By the middle of July, normal activity begins to wane and by the beginning of August, shops no longer close between 1 and 4 p.m., they close for two or three weeks. Dry cleaners close, mechanics close, factories close, wineries close, restaurants close, even some museums close. Cities like Florence and Venice would be abandoned if not for the tourists braving the heat to visit artistic treasures.*" The dramatic change in power demand reflects the fact that most major employers (like Fiat and many government offices) simply shut down for the month.

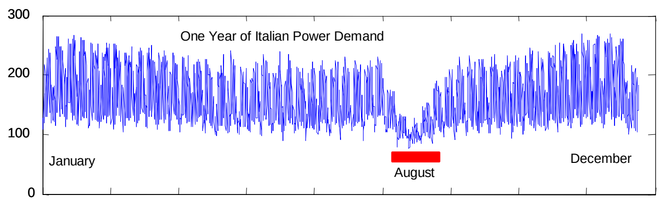

Figure 18: One year of Italian Power Demand in 1997

4.1 Intelligent Icon Search

Although the primary use of INTELLIGENT ICONS is visualization and data mining, their utility for query by content is related and potentially so useful that we briefly consider it here.

Most operating systems support search by '*name*', '*date*', '*size*' etc, and further enhance the search by '*name*' by allowing wildcards. However, no current operating systems support query by content. The utility of such search is becoming increasing obvious as commercial hard drives now exceed 400 gigabytes in size. For example, suppose we know that we have a preliminary version of a paper buried among our files, but we don't remember its name. It would be useful to be able to simply right click on the icon, and choose an option "*find most similar file*". We have built such a utility into our *Smart Browser* tool. When searching for the most similar icon we exclude from consideration files in the same folder as the query file (for files in the same folder, user can easily locate the most similar icon with a *Smart Browser*).

In general, query-by-content search using icons provides very intuitive results. For example, we have arranged DNA icons for approximately 245 mammals, reptiles and birds in folders that reflect their geographical location rather than their taxonomic relationship. If we search for the most similar file to `chimpanzee.dna` in the African folder, we are told that the closest match is `orangutan.dna` in the Asian folder. Likewise, as shown in Figure 19, a search for the most similar file to `american black bear.dna`, returns `Polar Bear.dna`[1].

We omit a detailed study of the efficiency of this search feature for brevity, except to note that we can search 50,000 icons in an average of 31.9 milliseconds.

[1] The Polar Bear is found in the Alaska and Canada, in addition to Iceland, Greenland and Russia, so the choice of placing it in the Europe folder was somewhat arbitrary. Note that the Asiatic Black Bear (*Ursus thibetanus*), which may be more similar to the American Black Bear, has not yet been sequenced.

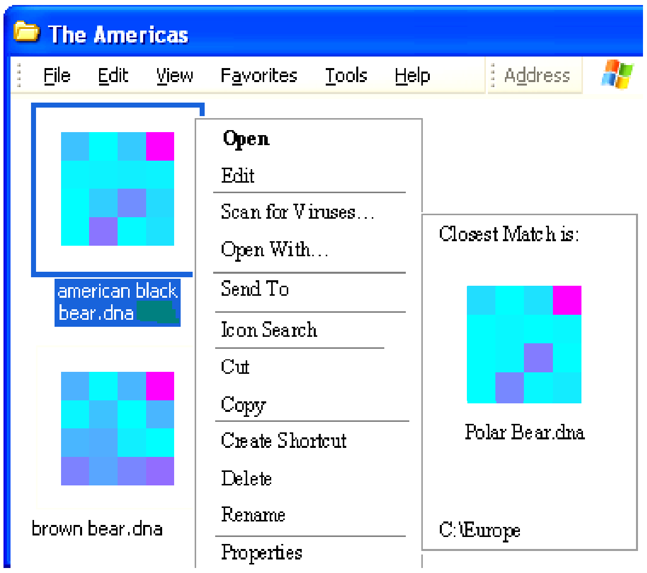

Figure 19: A screen capture of a search interaction with Smart Browser

5 Conclusions and Future Work

We have introduced INTELLIGENT ICONS, a novel technique for allowing visualization to take place in the background of day-to-day computer use. Future research directions include an extensive user study and providing support for other file types.

Acknowledgements

We would like to thank Ben Shneiderman and Marti Hearst for encouraging comments on an early draft of this work. We would also like to thank Dr. Helga Van Herle of the David Geffen School of Medicine at UCLA and all the donors of datasets.

References

[1] Daniel A. Keim, Hans-Peter Kriegel, and Mihacl Ankerst. Recursive pattern: A technique for visualizing very large amounts of data. In *Proc of IEEE Conference Visualization '95*, pages 279-286, 1995.

[2] Wojciech Basalaj. Proximity visualization of abstract data. PhD thesis, University of Cambridge Computer Laboratory, 2000.

[3] Jeff Beddow. Shape coding for multidimensional data on a microcomputer display. In *Proceedings of IEEE Conference Visualization '90*, pages 238-246, 1990.

[4] Herman Chernoff. The use of faces to represent points in k-dimensional space graphically. In *Journal of the American Statistical Association*, volume 68, pages 361-368, 1973.

[5] Eamonn Keogh. The UCR time series data mining archive. [http://www.cs.ucr.edu/~eamonn/TSDMA/index.html]. University of California, Riverside.

[6] Fabrizio Ferrandina, Thorsten Meyer, and Roberto Zicari. Implementing lazy database updates for an object database system. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 261-272, 1994.

[7] John P. Lewis, Ruth Rosenholtz, Nickson Fong, and Ulrich Neumann. VisualIDs: automatic distinctive icons for desktop interfaces. In *Proceedings of the 2004 SIGGRAPH Conference*, ACM Transactions on Graphics (TOG), volume 23, issue 3, pages 416-423, 2004.

[8] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *proceedings of the eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2-11, 2003.

[9] Martin F. Porter. An algorithm for suffix stripping. *Program*, volume 14, no. 3, pages 130-137, 1980.

[10] Jinwook Seo and Ben Shneiderman. A rank-by-feature framework for unsupervised multidimensional data exploration using low dimensional projections. In *Proceedings of the IEEE Symposium on Information Visualization 2004 (INFOVIS 2004)*, pages 65-72, 2004.

[11] Edward R Tufte. *Envisioning Information.* Graphics Press, 1990.