

# Intelligent Icons: Integrating Lite-Weight Data Mining and Visualization into GUI Operating Systems

Eamonn Keogh    Li Wei    Xiaopeng Xi    Stefano Lonardi    Jin Shieh    Scott Sirowy

University of California – Riverside

{eamonn, wli, xxi, stelo, shieh, ssirowy}@cs.ucr.edu

## ABSTRACT

*The vast majority of visualization 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. In this work we introduce a novel framework for allowing visualization to take place in the background of normal day to day operation of any GUI based operation system such as MS Windows, OS X or Linux. By allowing visualization to occur in the background of quotidian computer activity (i.e. finding, moving, deleting, copying files etc) we allow a greater possibility of unexpected and serendipitous discoveries.*

*Our system works by replacing the standard file icons with automatically created 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 allows us to take advantage of small multiples paradigm advocated by Tufte. We can further 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. We demonstrate the utility of our approach on data as diverse 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.*

## 1 INTRODUCTION

At the heart of many information visualization and data mining techniques is a single question “*compared to what?*” [20]. 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 for allowing lite-weight visualization and data mining to take place in the background of normal day-to-day operation of any GUI based operation system such as MS Windows, OS X or Linux. By allowing visualization to occur in the background of quotidian computer activity we allow a greater possibility of unexpected, serendipitous and useful discoveries.

Our system works by replacing the standard file icons with automatically created 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 allows us to take advantage of *small multiples* paradigm elucidated by Tufte, allowing us to answer the question “*compared to what?*” 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 files, 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 spirit to the recent VisualIDs work of Lewis et. al. [15]. Here the authors note that “*search and memory for images is known to be generally faster and more robust than search and memory for words*”, and they leverage off this fact by automatically creating distinctive icons for desktop interfaces. The icons are created by hashing the filenames 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, thus allowing a user to see at glance when two files are related.

The most important distinction between VisualIDs and INTELLIGENT ICONS is that the former only looks at a file *name*, whereas the latter looks at file *content*. The authors of VisualIDs make a convincing case that producing the icons appearance exclusively from the file name does lend a “*permanence*” (in spite of editing changes) that may aid the user in navigating through a large file system. However our goal extends beyond creating a simple aide-memoire for navigation. The objective is to create a system that also supports information visualization and query by content. As a simple example of the difference of the two approaches consider Figure 1.

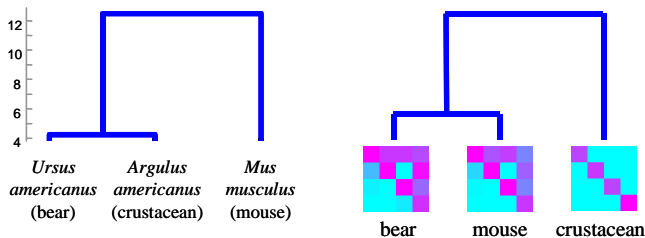


Figure 1: The similarity of 3 DNA files based on file name (left) and file contents (right).

The three files in the example are ASCII text files, each of which contains approximately 16,000 base pairs of mitochondrial DNA. Here we used string edit distance as suggested in [15] to measure the distance between file names, and Euclidean distance to measure the distance between the file icons (as explained in more detail below). 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<sup>1</sup>, whereas the INTELLIGENT ICON approach captures the correct relationship between the three species.

An additional limitation of VisualIDs is that most people do not explicitly name the vast majority of files on their hard drive. Rather the file names are inherited when the files are downloaded, or they are automatically named by automatic links to an external database (i.e. music files named by CDDDB.com), or they are automatically generated by an application, for example Canon-001.jpg, Canon-002.jpg etc. In such cases VisualIDs may have limited utility. Given the different goals of our approach and VisualIDs, we will not discuss this work any further.

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 [6]. Beddow and others exploited the availability of color display and printers to extend this mapping to colors [5]. Keim et. al. introduced *Recursive Patterns* in [8]. Recursive patterns can be considered as a general technique to map data to bitmaps, although icons were not explicitly considered.

The arrangement of icons on the screen is an important component of our work. Ward [22] contains an excellent overview and some important original contributions.

## 2 AN EXAMPLE OF AN ICON GENERATION ALGORITHM

For concreteness we begin with a particular example of an icon generation algorithm before considering the more general framework below. We have chosen DNA data for our first example. We recognize that DNA is a rather specialized file type. However there are two reasons for using it as the introductory example. First, its special structure lends itself to simple elucidation. Second, DNA

<sup>1</sup> It might be argued that the discovered similarity of specific name of “americanus” is somehow geographically meaningful. However, the specific name part of most organisms such as “orientalis”, “japonicum”, “asiatica” are used in fairly arbitrary and inconsistent ways that have little utility for taxonomy.

is unique in that it is the only dataset for which there exists a unique taxonomy, and this taxonomy is near universally agreed on for most “major” animals. This fact will allow us to objectively test the similarity of icons.

### 2.1 DNA to INTELLIGENT ICON

Consider a DNA string, which is a sequence of symbols drawn from the alphabet {A, C, G, T}. DNA strings can be very long. For example the human mitochondrial DNA has 16,571 such symbols, beginning with GATCACAGGTCTATCACCTATTAACCACT... and ending with ...ACATCACGATG. This long sequence (approximately five pages of text in this papers format) is only a tiny subset of the three billion letters that actually make up the entire human genome. We want to note here that size of the icon (32 by 32 pixels) is the limiting factor in summarizing the information content of large files.

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. The first attempt to map a sequence in an icon would be to divide the bitmap into four quadrants and count the frequency of each of the four possible base pairs. We can then map the observed frequencies to a linear colormap to produce a icon using the indexed colors to fill in the corresponding sections of the bitmaps as shown in Figure 2.

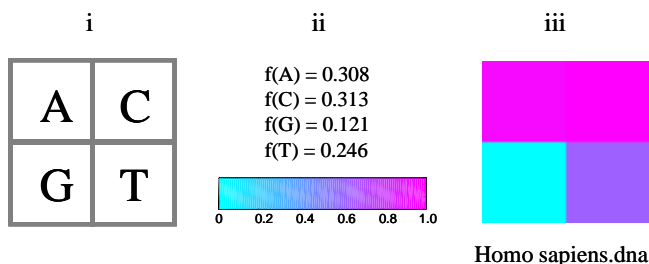


Figure 2: i) The four DNA base pairs arranged in a 2 by 2 grid. ii) The observed frequencies of each letter in human mitochondrial DNA can be indexed to a colormap to produce a file icon as shown in iii.

Note that in this case the arrangement of the four letters is arbitrary, and that the choice of colormap is also arbitrary. In order to use as much of the color spectrum as possible, we normalize the data such that the lowest frequency symbol maps to zero and the highest frequency symbol 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:

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

One could apply this simple mapping to a set of DNA sequences corresponding to different species and examine the icons in a file browser. Unsurprisingly however (and unfortunately for human vanity) there is very little difference between the icons obtained in this way for most mammals. In an attempt to improve the discrimination ability of the icons we can use more features, examining the frequencies of all possible pairs of letters. For example the substring **AT** appears 3 times in the first 30 base pairs of the human mitochondrial DNA,

which is **GATCACAGGTCTATCACCTATTAACCACT**. If we attempt this strategy, we must consider the best way to map the new features to our 32 by 32 bitmap. We could do this arbitrarily as before, for example we could sort lexicographically the words and fill in the bitmap left to right, top to bottom. However below we show an alternative method that has a potentially useful property.

Most GUI operating systems allow the user to view files icons at different sizes. For example MS Windows XP can show the icons in 4 different sizes depending on whether you chose “thumbnails”, “tiles” “icons”, “list” in the view options. It would therefore be desirable if we could arrange for a file icon to be similar to itself regardless of the number of features used to create it. Surprisingly, this is easy to arrange for DNA. Below we show a general mapping for DNA that has this property.

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

A → 0	C → 1	G → 2	T → 3
-------	-------	-------	-------

We can control the desired number of features by choosing  $l$ , the length of the DNA words. Each word has an index for the location of each symbol, for clarity we can show them explicitly as subscripts. For example, the first word with  $l = 4$  extracted from the human mitochondrial DNA is  $G_0A_1T_2C_3$ . So in this example we would say  $k_0$  is **G**,  $k_1$  = **A**,  $k_2$  = **T** and  $k_3$  = **C**.

To map a word into a bitmap we can use the following equation to find its row and column values:

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

Figure 3 shows the mapping for  $l = 1, 2$  and (part of) 3.

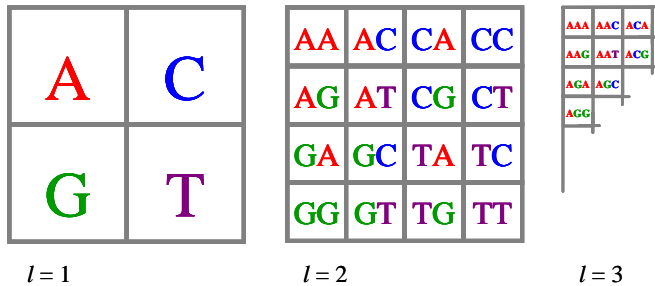


Figure 3: The mapping of DNA words of  $l = 1, 2$  and 3. (The colors of the text are just to allow visualization of the mapping algorithm).

If one examines the mapping in Figure 3, one can get a hint as to why a bitmap for a given species might be self-similar across different scales. For example note that for any value of  $l$ , the top column consists only of permutations of **A** and **C**, and that the two diagonals consist of permutations of **A** and **T**, or **G** and **C**. Similar remarks apply other rows and columns.

To demonstrate this self-similar property we have created the icons for two different species at multiple levels in Figure 4.

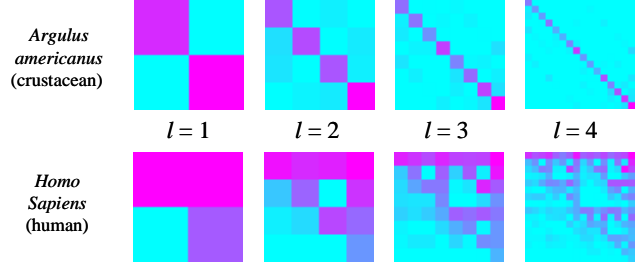


Figure 4: The icons created for two species at every level from 1 to 4. Note that the icons for a given species look similar across all levels.

Note that this converging property of has been noted before (for a more complex variation of our mapping scheme) and has been used to study genomes [1]. For example, a biologist can recognize that a particular DNA word, say in a bacterial genome, is rarely used. This would suggest the possibility that the bacteria have evolved to avoid a particular restriction enzyme site, which means that it might not be easily attacked by a specific bacterio-phage.

## 2.2 Optimizing and Arranging the Icons

Recall that our intent is to produce icons that reflect the similarity of the files. We can objectively measure the similarity of two icons by using the Euclidean distance between the matrices of original frequency counts. Given matrices  $A$  and  $B$ , of the same level  $l$ , and denoting the  $i^{\text{th}}$ ,  $j^{\text{th}}$  element as  $A_{ij}$ , we can measure the distance as:

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

This distance measure assumes that we have access to the original frequency counts matrices. However in practice we can use the actual icons, provided we keep the colormap to translate the 3-dimensional color (RGB values) back to 1-dimensional frequency counts.

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

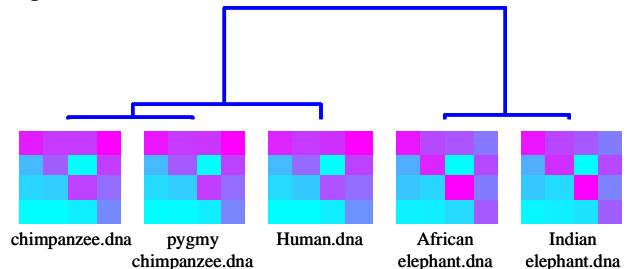


Figure 5: Five species clustered using the distance between their bitmap representations (for clarity we used their common names).

These results are something of a mixed bag for us. Although the clustering is objectively correct, the differences detected by Euclidean distance measure are very subtle to the naked eye. For example one must look

quite closely to observe that the top right element of the primates bitmap is pink, whereas the corresponding element for the elephants is blue.

We have therefore identified the need to enhance the subjective visual discriminatory power of the icons. We will devote an entire section to a discussion of general techniques for doing just that. In what follows we show an example of the *type* of modification that could enhance the subtle visual similarities and differences of icons. An obvious possible “trick” would be to normalize the  $i^{th}, j^{th}$  elements across all icons. This has the effect of enhancing subtle differences in color. For example the bottom right element of all 5 icons shown in Figure 5 appear to be minor variations of blue violet, it requires careful inspection to note that the elephants have a slightly darker shade of it. In Figure 6 however, normalization has emphasized the differences such that the element in question is magenta (fuchsia) for the elephants and pale turquoise for the human.

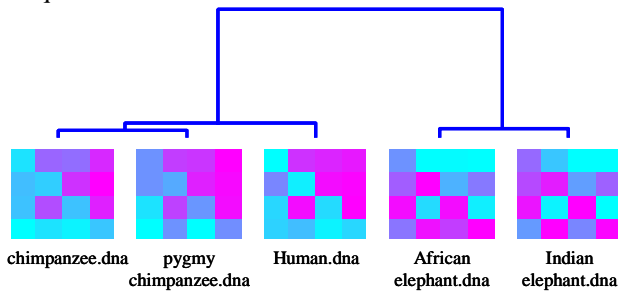


Figure 6: Five species clustered using the distance between their *normalized* bitmap representations.

Here the visual patterns are much more satisfactory, and we can finally see a hint of the potential utility of INTELLIGENT ICONS. As a simple example of this, imagine we encountered the icon shown in Figure 7.

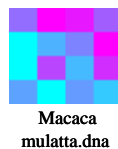


Figure 7: The icon for another African mammal. Is this animal more similar to an elephant or ape?

Based on the file name, we probably cannot say anything about this animal, but simply by glancing at the file icon and comparing it to the icons in Figure 6 we might reasonably guess that this animal is more similar to the chimps/human than to the elephants. In fact, this *is* the case, *Macaca mulatto* is more commonly known as the rhesus monkey.

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 8.

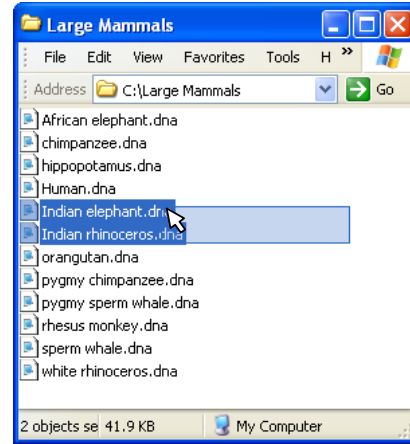


Figure 8: Twelve DNA files, sorted by *name*, in a typical file browser. Using the classic technique of bounding box selection we can select subsets of the files, in this case the Indian elephant and the Indian rhinoceros.

We can use the classic bounding box section tool to selection various subsets. However in this example it is hard to extract *meaningful* subsets, other than the dubious pairs pygmy chimpanzee/pygmy sperm whale and Indian elephant/ Indian rhinoceros (Note we are using familiar English names here for clarity, however using scientific names does not help, for example the two types of elephant, *Elephas maximus* and *Loxodonta Africana*, are not alphabetically close).

We can use INTELLIGENT ICONS to solve this problem by arranging the icons in the file browser based on their *similarity*, rather than the classic options of *name*, *size*, *date* etc. There has been much work on arranging icons (glyphs/photo thumbnails etc) on a screen (see [22] for an excellent overview). We have adopted Multi-Dimensional Scaling (MDS), which requires a distance matrix between all icons as its input (calculated using E.q. 3). In order to prevent the icons from partially or completely overlapping, we snap-to-grid the icons to the nearest unoccupied grid point as suggested by Basalaj [4].

Although the time complexity of MDS is cubic in the number of objects, we found that even on a large screen full of small icons, an efficient MDS implementation can dynamically adjust the position of the icons in real time as the user changes the aspect ratio of the file browser. Figure 9 shows the same 12 mammals as shown in Figure 8 arranged in this way<sup>2</sup>.

<sup>2</sup> To mitigate some of the problems of reproducing screen captures at a small scale, this screen capture and some those that follow, have had minor touch ups in a photo editing program. For example, the cursor was made twice its normal size. However in no case where the colors or locations of the icons changed.

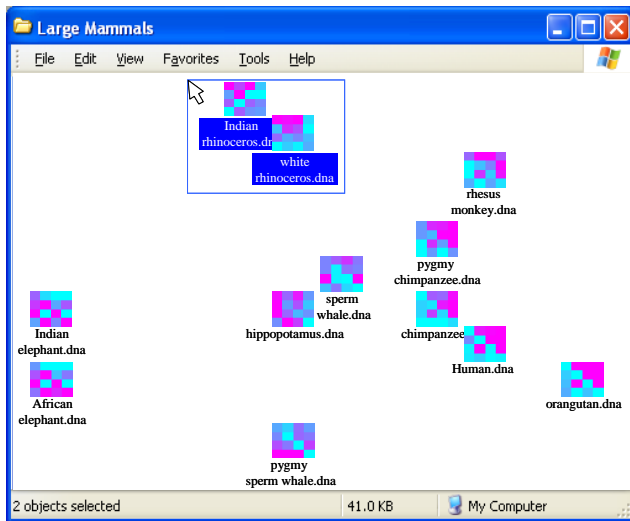


Figure 9: Twelve DNA files, arranged by *INTELLIGENT ICONS*, in a typical file browser. Using the classic technique of bounding box selection we can select subsets of the files, in this case the Indian rhinoceros and the white rhinoceros.

In addition to being able to select both types of Rhinos (*Rhinocerotidae*) as shown above, we can now also use standard bounding rectangles to select other logical groups, such as:

- Both types of elephants (*Elephantidae*).
- All the primates (*Catarrhini*).
- Just the greater apes (*Hominidae*).
- Just the two types of chimps (*Panines*).
- Just the chimps and humans (*Hominids*).

At first glance the fact that we must select the hippo when we select the two types of whales seems like an error, surely the hippo belongs with either the elephants or the rhinos. Interestingly, this is not the case; the hippos are more closely related to whales than to any other mammals! Whales and hippos diverged a mere 54 million years ago, whereas the whale/hippo group parted from the rhinos about 76 million years ago, and from the elephants about 105 million years ago. The group that includes hippo and whales/dolphins, but excludes all other mammals above is called *Cetartiodactyla* [23]. We call the combination of *INTELLIGENT ICONS* and the MDS layout a *Smart Browser*.

### 3 GENERALIZING FROM THE DNA EXAMPLE

We have now seen a concrete example of *INTELLIGENT ICONS*, and shown some examples of their utility. We want to have a software tool that (1) is capable of changing the individual icons of selected file types and (2) allows the option of arranging the file icons by similarity. When the tool is first installed, the user must to create or download plug-ins that tells our software how to convert their filetypes. Below we consider plug-ins for text and time series and provide general guidelines for arbitrary data types. Note that this type of software design has recently become very popular. For example, the Google Desktop Search Tool is able to index a handful of common file types as shipped, however volunteers have

written plug-ins that allow the program to index more exotic file types such as DjVu, 3dsmax and C++ source code [14].

In order to be able to handle more file types, next we generalize the ideas presented in the previous section. 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 have a particular icon associated with them. This makes it easy to determine at a glance the file type (e.g., PDF, PowerPoint, etc.) It is desirable that *INTELLIGENT ICONS* inherit this property. As we shall see, this property is only apparently in conflict with the property 2 below.
- Similar files should have similar icons. This is the fundamental property that allows smart browsing, that is allow users to spot clusters, duplicates and outliers in their data. Furthermore, as we shall see later, this property can support query by content (e.g., *find me the file most similar to this one*), whereas current systems only support query by *name, data, size* etc.
- File icons should look similar at different resolution (cf. Figure 4). This is because most operating systems allow use to view icons at various sizes.
- File icon updates should be fast. It is important files can be added, deleted or edited, and have their icons instantaneously reflect their content.

Below we will consider how to address these properties in more detail.

##### 3.1.1 Distinctiveness of File Type

There can be little doubt that having distinctive file icons for different file types aids rapid file navigation. At first blush it may appear that the idea of basing the icons on the file contents would remove this benefit. However, this is not the case. We can retain file distinctiveness while allowing individuality with a combination of two techniques:

- Using different colormaps for different file types.
- Using different mappings for different file types.

To illustrate this we have chosen three distinctive colormaps for the three main datatypes that we have encountered (personally, given our research interests). In addition we have chosen a distinctive mapping for video games as shown in Figure 10.

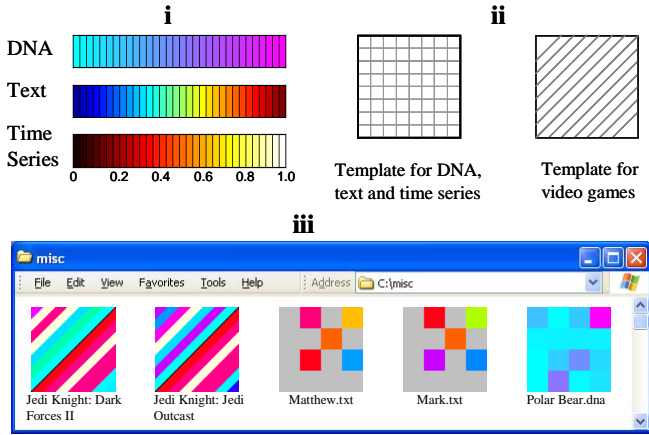


Figure 10: i) The three different colormaps used for the 3 principle file types considered in this work. ii) Two examples of mapping templates for INTELLIGENT ICONS. iii) A screen capture of a folder with 3 different file types.

In the figure above the choice of colormaps was completely arbitrary, however this need not be the case. For example gene expressions visualizations almost always use a red/green colormaps [19] and we could leverage of this fact to create intuitive icons for that file type.

### 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. These general principles are familiar to those in the machine learning and visualization communities.

We want to extract features that have high discriminatory power. For example, for text documents the feature,  $\text{frequency\_of\_word}(\textit{the})$  is not useful.

We want features that are as independent as possible. For example, for text documents if we included the feature  $\text{frequency\_of\_word}(\textit{bicycle})$ , there would probably be little utility of including the  $\text{frequency\_of\_word}(\textit{bike})$ .

Below we consider these requirements on two concrete examples, namely text and time series and a generic “type”, metadata.

**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. We can leverage off the large body of work in the text IR community to map these files to icons. For example we begin by discarding stop words, such as “*the*”, “*of*”, “*and*” etc. Such words tend to have equal frequency across all documents and thus have little discriminative power. We next stem the words using Porters algorithm [17], so that variations on a word map to a single root, for example “*dividing*”, “*divided*” and “*divide*” all map to “*divid*”. After completing these steps we are typically left with much fewer words, although for large documents collections many tens of thousands of words is still possible. Since the number of possible words is much

greater than the number of pixels available, we need to reduce the dimensionality of the features. We achieve this by using a classic text-processing algorithm called Latent Symantec Indexing (LSI). LSI finds a lower dimensionality representation of the data by projecting it onto a space that reflects the latent structure. This takes care of the problem of synonymy and also prioritizes the features by arranging them by relative importance.

**Time Series:** Time series are a ubiquitous and increasingly prevalent type of data. They occur in virtually every field of human endeavor, including finance, medicine, meteorology and entertainment. 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. allows recursive generalization of arbitrary line and column oriented arrangements, including time series. Another possibility is to discretize the time series and use the approach above for text, or to discretize the time series into *exactly* 4 symbols and use the algorithm above for DNA. Let us consider the later approach in more detail.

While there are at least 200 techniques in the literature for converting real valued time series into discrete symbols [12], the SAX technique of Lin et. al. is unique and ideally suited for our purposes [16]. The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. The mean value of each section is then calculated. This produces a reduced dimensionality piecewise constant approximation of the data. This representation is then discretized in such a manner as to produce a word with approximately equiprobable symbols. Figure 11 shows a short time series being converted to a discrete string.

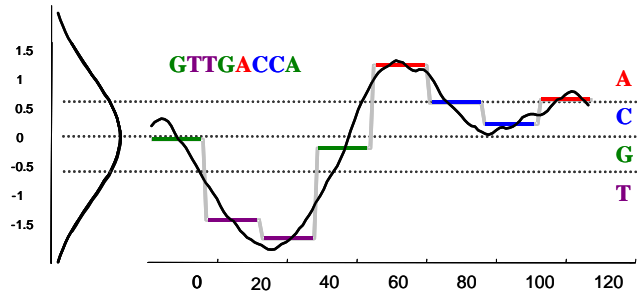


Figure 11: A real valued time series being discretized into the SAX word GTTGACCA.

The figure shows a relatively short time series converted into a pseudo DNA word of 8 symbols, hardly long enough to robustly extract frequency information. Fortunately most time series in the real world are typically much longer, for example ECG samples in a medical log often contain at least 10,000 datapoints.

**Metadata:** The diligent reader may already be wondering how we extracted meaningful features from the video game executables show in Figure 10. The answer is, we did not. 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 the file contents, but on metadata provided

(automatically) by CDDDB.com. The features available include, Track Artist, Record Label, Year, Beats Per Minute, Metagenre (rock, classical, new age, jazz, etc.), Subgenre (punk, ska, baroque, choral, ambient, bebop, ragtime) etc.

For video games, there is no completely automatic metadata server, but an hours work enabled use to write a crawler which extracted features from [www.metacritic.com/games/pc/scores/](http://www.metacritic.com/games/pc/scores/).

The idea of using external metadata to create the icons opens several exciting possibilities for future research; however for brevity we will not further discuss this here.

### 3.1.3 File icons should look similar to different resolution versions of themselves

File icons should look similar when viewed at different scales because most operating systems allow user to view icons at different resolutions. For example Windows XP supports icons sized  $48 \times 48$ ,  $32 \times 32$ ,  $24 \times 24$ , and  $16 \times 16$  pixels. Microsoft invites application developers to produce optimized versions of icons in each size; otherwise it takes the single icon provided and (linearly) interpolates it to the other sizes.

In some cases this “self-similar” property can be easily arranged, we have already seen in Figure 4 that our mapping for DNA has this property, and our mapping function for time series inherits this property. So we can use an  $l = 2$  mapping for  $16 \times 16$  pixels, and an  $l = 3$  mapping for  $32 \times 32$  pixels, and expect the two icons to resemble each other.

More generally, this property may be hard to ensure if we wish to use every pixel of say a  $48 \times 48$  bitmap. When we reduce the size of this bitmap to  $24 \times 24$ , 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 from which they were derived. The good news is that it is unlikely we would ever want to 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. Even the 256 dimensions allowed by the smallest icon size would be hard to meaningfully populate for most domains. 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.

The two techniques, variable level mappings, and simple linear extrapolation are not mutually exclusive; Figure 12 shows how we combine both techniques for the DNA file icons.

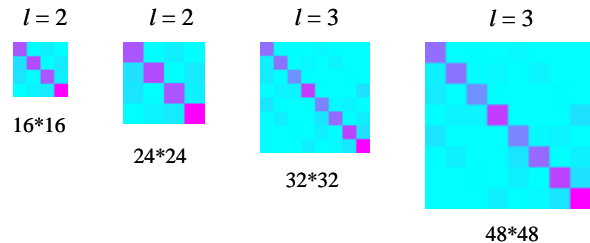


Figure 12: Four different sized DNA icons for *Argulus americanus*. The smallest icon is a level 2 mapping of one feature to 4 pixels; the next size up is simply an enlargement of the smallest size. The  $32 \times 32$  size icon is a level 3 mapping of one feature to 4 pixels, and the largest icon is simply an enlargement of the second largest size.

### 3.1.4 File icon updates should be fast

In general, if we only need to process a few files in order to create their INTELLIGENT ICON, time complexity might not be an issue. For example, using the mapping algorithm in Eq. 2 for DNA, we can create an icon in a few milliseconds for a file containing hundreds of thousands of base pairs. However the issue of time complexity does become important if the mapping algorithm requires access to multiple files. We have already seen examples of this situation. In Section 2.2 we have shown that DNA icons look better if we normalize the frequencies across all icons. Clearly, if we add a new file to our collection, these frequencies can be expected to change somewhat. 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* [13]. The basic idea is to learn the best mapping on all  $N$  files offline, use it to create icons for all  $N$  files, and save the mapping function. If we later need to 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. We do this in one of two ways, either assign a very low priority thread to the process (this is Google’s solution for its desktop search indexer) or perform all updates at a scheduled time when we are unlikely to compete with human users for CPU time, say in the middle of the night.

## 4 EXPERIMENTAL EVALUATION OF INTELLIGENT ICONS

The central claim of our paper is that INTELLIGENT ICONS allow unexpected and serendipitous discoveries. This is a difficult claim to prove in anything but an anecdotal way. Fortunately, UCR is the home of a very large archive of time series test datasets. We can begin by examining this archive in a smart browser.

We used the tool to browse the hundreds of datasets in the UCR archive. One such dataset, known as Kalpakis\_ECG, contains 18 normal ECGs used to test time series clustering techniques. Figure 13 shows the dataset as most people have viewed it.

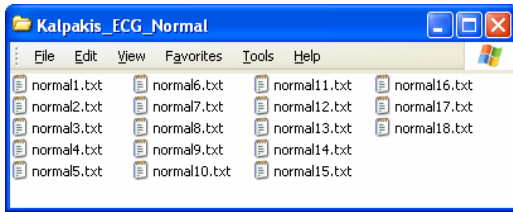


Figure 13: The 18 normal ECGs from the Kalpakis dataset shown in a typical MS Window XP file browser.

When we glanced at this dataset with our *Smart Browser*, we immediately noticed something interesting. While ECGs (and therefore the icons derived from them) can have great variability, five of the 18 thumbnails had radically different icons. Figure 14 illustrates this.

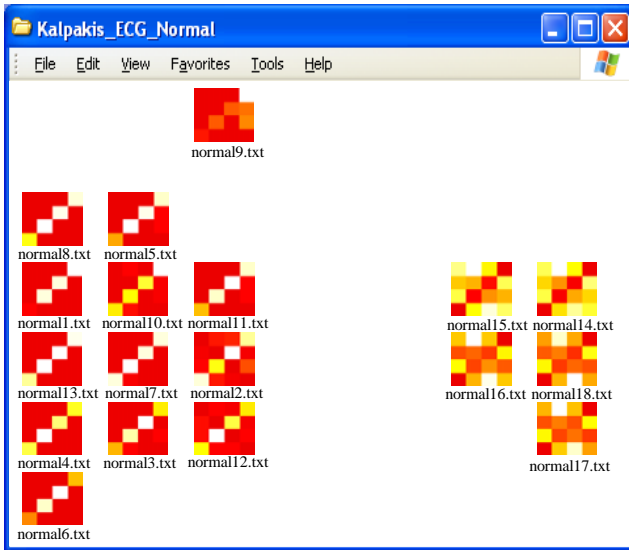


Figure 14: The 18 normal ECGs from the Kalpakis dataset shown in a Smart browser. Five of the INTELLIGENT ICONS are radically different from the rest.

This structure was so unexpected 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 15 illustrates the difference.

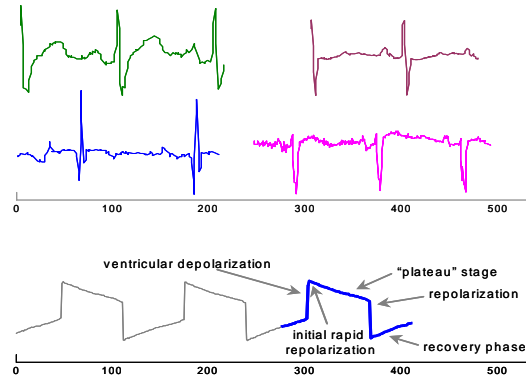


Figure 15: Top) Four snippets from randomly chosen ECGs from the Kalpakis\_ECG dataset. Note that ECGs can have great variability. Bottom) A snippet from the normal18.txt "ECG" from the Kalpakis\_ECG dataset.

In retrospect, after visualizing the data it is apparent even to the untrained eye that the five time series in question are radically different to the rest. Nevertheless many people have used this dataset to test algorithms, apparently without noticing this [7].

Another dataset we examined with *Smart Browser* was a NASA dataset containing examples of telemetry from a Space Shuttle valve. Figure 16 shows five such time series.

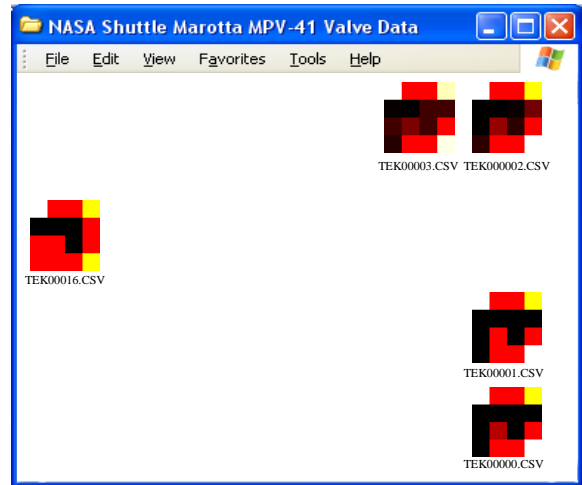


Figure 16: Five NASA Marotta MPV-41 valve trace files shown in a Smart Browser.

It is immediately apparent that one file has quite a different structure to the rest. NASA engineers were able to explain the difference by noting that while the other four files correspond to normal sequences, file TEK00016.CSV corresponds to an abnormal trace, as shown in Figure 17.

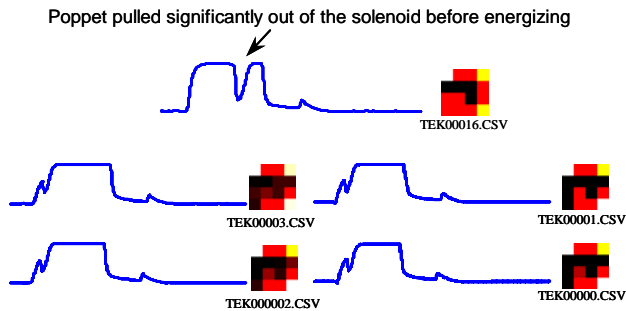


Figure 17: The five time series whose INTELLIGENT ICONS are shown in Figure 16. Note that the bottom four are normal, but TEK00016.CSV has a fault.

As a final example we consider twelve monthly electrical power demand time series from Italy, Figure 18 shows the data viewed in a smart browser. It is immediately apparent that there are two major clusters that correspond to winter months (October to March) and summer months (April to September). 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 immediately obvious feature of Figure 18 is that the month of August is an outlier. This is apparent from both its icon's location on the screen and by its color. To get some insight into this phenomenon we can visualize the entire year as a single time series as in Figure 19. 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".

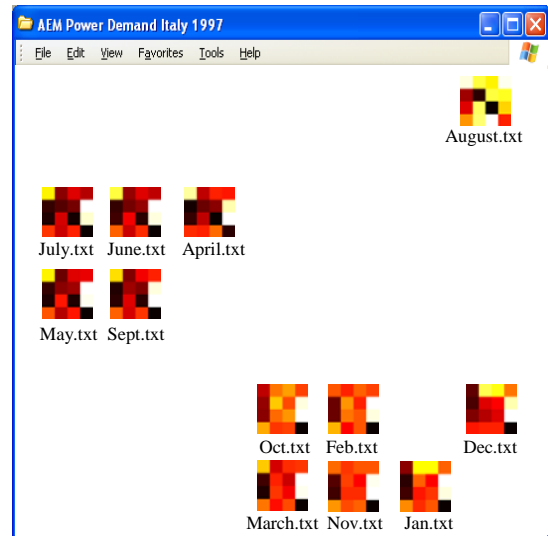


Figure 18: Twelve monthly power demand time series from Italy shown in a Smart Browser.

The dramatic change in power demand reflects the fact that most major employers (like Fiat and many government offices) simple shut down for the month.

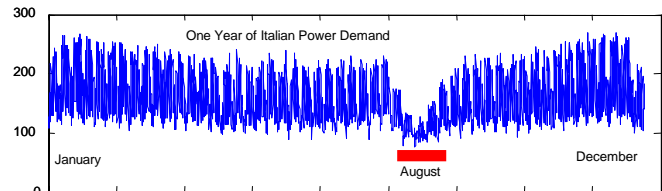


Figure 19: One Year of Italian Power Demand (1997). Note that the month of August is radically different to the rest of the year.

As before, once the data is viewed by plotting it in Matlab or MS Excel, it is fairly easy to see the differences. However, without the Smart Browser to invoke the users curiosity, this simply may never happen.

#### 4.1 INTELLIGENT ICONS FOR TEXT

A central claim of this work is that once the basic framework for INTELLIGENT ICONS has been established, it is easy for people to write "plug-ins" for their favourite data types. To test this hypotheses, the first author spent 15 minutes explaining the basics of INTELLIGENT ICONS to graduate students taking a data mining course (UCR CS235, Spring 06), and invited students to write a plug-in for any data type they were interested in. Two students (Jin and Scott, credited here as co-authors) produced a text plug-in. Their approach is somewhat different to the approach suggested above, and does not (currently) obey the "File icons should look similar at different resolution" property. However their project demonstration elicited such a positive reaction we decided to include an example of their work as an example of the kind of thing which is possible with a days work. Full details of how their approach works can be found in [18].



Figure 20: Thirty-eight PDF files represented by Shieh-Sirowy INTELLIGENT ICONS. See Figure 21 for an explanation of the results

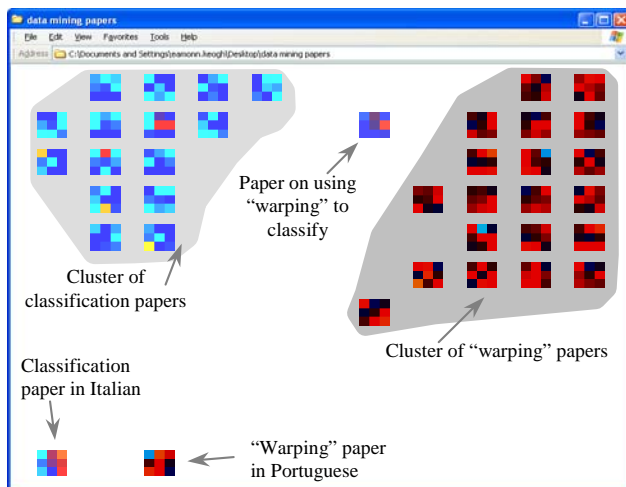


Figure 21: A visual key explaining the results of Figure 20

The dataset in question is a collection of database/data mining papers by diverse authors, which reference one of two papers by the first author. Those two papers are “*Exact indexing of dynamic time warping*” [10] and “*Learning the Structure of*

*Augmented Bayesian Classifiers*” [11]. In order to make the task more challenging, we indexed all the text *except* the references.

In Figure 20 we can see two major distinct clusters. One cluster is a collection of papers on (mostly Bayesian) classification, the other cluster is a collection of papers on Dynamic Time Warping (DTW). One icon is centered almost exactly in-between the two major clusters. This makes perfect sense, since it is a paper on *classification* of time series that using *DTW* (Decision-tree Induction from Time-series Data... by Yamada et al. [24]) and thus belongs equally to both clusters.

The two remaining icons also have intuitive placement and coloring. Both are written in languages other than English, which explains why they appear as outliers. However, their coloring still gives us a clue as to their content. The icon that has mostly blue pixels (in Italian) is about classification, and the icon that has mostly red pixels (In Portuguese) is about warping. This coloring is reflected in the major clusters. It is perhaps surprising that the icons are intuitive even in the face of been in different languages, however an examination of the texts reveals the occasional passage that lapses into English, such as: “...*verificar a superioridade da Warp Metric Distance como medida...*”, and this is enough structure for the algorithm to produce intuitive icon coloring.

## 4.2 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 increasingly 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.

In general, query-by-content search using icons provides very intuitive results. For example, we have arranged DNA icons for approximately 380 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 22, a search for the most similar file to `american_black_bear.dna`, returns `Polar Bear.dna`<sup>3</sup>.

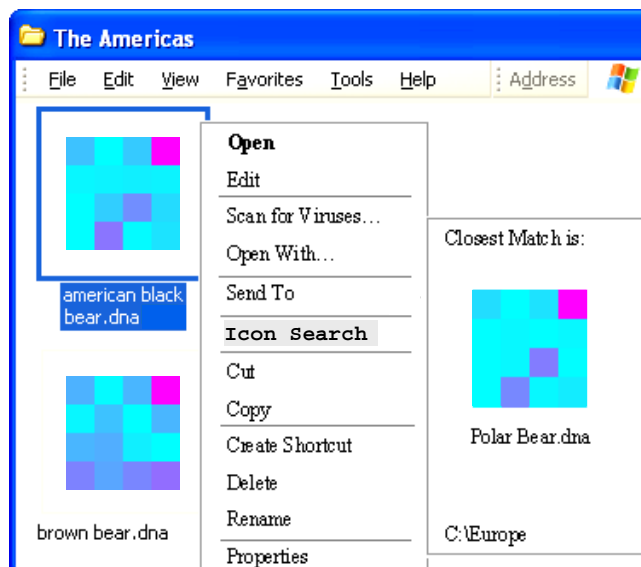


Figure 22: A screen capture of a search interaction with Smart Browser. The user right clicked on the icon for the American Black bear, and chose “Icon Search”, the closest match was the polar bear.

<sup>3</sup> 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.

Shortly before this paper was submitted, we became aware of an interesting proof of the similarity of the Polar Bear and the American Black Bear. The first example of a hybrid in the wild was confirmed by DNA tests [1].

## 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 Edward Tufte, Ben Shneiderman, Christos Faloutsos, Marti Hearst and Margaret H. Dunham for encouraging comments on an early draft of this work. We would also like Dr. Helga Van Herle of the David Geffen School of Medicine at UCLA and all the donors of datasets.

**REPRODUCIBLE RESEARCH STATEMENT:** All datasets use in this work are available by emailing the first author.

## REFERENCE

- [1] Associated Press: “Hunter Shoots Hybrid Bear”, 2006-05-12. Retrieved on 2006-06-18.
- [2] Jonas S. Almeida, Joao A. Carrico, Antonio Maretzek, Peter A. Noble, and Madilyn Fletcher. Analysis of genomic sequences by Chaos Game Representation. In *Bioinformatics*, volume 17, no. 5, pages 429-37, 2001.
- [3] Daniel A. Keim, Hans-Peter Kriegel, and Mihael Ankerst. Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of IEEE Conference Visualization '95*, pages 279–286, 1995.
- [4] Wojciech Basalaj. Proximity visualization of abstract data. PhD thesis, University of Cambridge Computer Laboratory, 2000.
- [5] Jeff Beddow. Shape coding for multidimensional data on a microcomputer display. In *Proceedings of IEEE Conference Visualization '90*, pages 238-246, 1990.
- [6] 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.
- [7] Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 273-280, 2001.
- [8] Daniel A. Keim, Mihael Ankerst, and Hans-Peter Kriegel. Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data. In *Proceedings of IEEE Visualization 1995*, pages 279-286, 1995.
- [9] Eamonn Keogh. The UCR time series data mining archive. [<http://www.cs.ucr.edu/~eamonn/TSDMA/index.html>]. University of California, Riverside.
- [10] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28<sup>th</sup> International Conference on Very Large Data Bases*, Hong Kong, pages 406-417, 2002.
- [11] Eamonn Keogh, and Michael Pazzani. Learning the Structure of Augmented Bayesian Classifiers. *International Journal on Artificial Intelligence Tools*, Vol. 11, No. 4, pages 587-601, 2002.

- [12] C. Stuart Daw, Charles Edward Andrew Finney, and Eugene R. Tracy. A review of symbolic analysis of experimental data. In *Review of Scientific Instruments*, volume 74, no. 2, pages 915-930, 2003.
- [13] 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.
- [14] Google desktop search plug-in download page. <http://desktop.google.com/plugins.html>
- [15] 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.
- [16] 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.
- [17] Martin F. Porter. An algorithm for suffix stripping. *Program*, volume 14, no. 3, pages 130-137, 1980.
- [18] Jin Shieh and Scott Sirowy: Organizing Internet Bookmarks using Latent Semantic Analysis and Intelligent Icons. [www.cs.ucr.edu/~eamonn/cs235\\_final\\_report.pdf](http://www.cs.ucr.edu/~eamonn/cs235_final_report.pdf)
- [19] 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.
- [20] Edward R Tufte. *Envisioning Information*. Graphics Press, 1990.
- [21] Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Does organisation by similarity assist image browsing? In *Proceedings of the ACM Conference on Human Factors in Computing Systems (ACM CHI 2001)*, pages 190-197, 2001.
- [22] Matthew O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. In *Information Visualization*, volume 1, no. 3-4, pages 194-210, 2002.
- [23] Bjorn M. Ursing and Ulfur Arnason. Analyses of mitochondrial genomes strongly support a hippopotamus-whale clade. In *Proceedings of the Royal Society of London, Series B*, volume 265, pages 2251-2255, 1998.
- [24] Yuu Yamada, Einoshin Suzuki, Hideto Yokoi, and Katsuhiko Takabayashi. Decision-tree Induction from Time-series Data Based on a Standard-example Split Test. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 840-847, 2003.