CONF-960105--22

LA-UR 96-0373

RECEIVED
APR 01 1996
OSTI

TITLE: **INTERACTIVE LAYOUT MECHANISMS FOR IMAGE DATABASE RETRIEVAL**

AUTHOR(S): John MacCuish, Allen McPherson, Julio Barros, Patrick Kelly

SUBMITTED TO:

IS&T/SPIE Symposium on Electronic Imaging: Science & Technology
San Jose, CA
Feb. 1-5, 1996

MASTER

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

[illegible]

Interactive Layout Mechanisms for Image Database Retrieval

John M^{ac}Cuish

University of New Mexico

Allen M^{c}Pherson

Los Alamos National Laboratory

Julio Barros

University of Virginia

Patrick Kelly

Los Alamos National Laboratory

January 29, 1996

ABSTRACT

In this paper we present a user interface, CANDID Camera, for image retrieval using query-by-example technology. Included in the interface are several new layout algorithms based on multidimensional scaling techniques that visually display global and local relationships between images within a large image database. We use the CANDID project algorithms to create signatures of the images, and then measure the dissimilarity between the signatures.

The layout algorithms are of two types. The first are those that project the all-pairs dissimilarities to two dimensions, presenting a many-to-many relationship for a global view of the entire database. The second are those that relate a query image to a small set of matched images for a one-to-many relationship that provides a local inspection of the image relationships. Both types are based on well-known multidimensional scaling techniques that have been modified and used together for efficiency and effectiveness. They include nonlinear projection and classical projection. The global maps are hybrid algorithms using classical projection together with nonlinear projection. We have developed several one-to-many layouts based on a radial layout, also using modified nonlinear and classical projection.

Keywords: user interface for image databases, multidimensional scaling, query-by-example, content-based image retrieval, layouts algorithms, iterative non-linear projection, classical MDS

1 INTRODUCTION

With the increase in the number of large image databases, there is a developing need for effective and efficient tools for image database searching and browsing. Though there is a small set of commercially available software packages (e.g. *Ultimedia Manager* by IBM, and *PhotoFlash* by Apple Computer) that provide a limited array of image database searching and browsing capabilities based on color, texture or shape matching, there is no software to the best of our knowledge that assists the user in visually navigating an image database. There are however several examples for visually navigating document databases and we draw on some of this work.[3–5]

We have created *CANDID Camera*, a prototype software interface tool for query-by-example, image database retrieval. Currently, only color or texture measures are used for matching, and there is a limited capability for text matching within the database. CANDID Camera provides the user with several interactive features based around global and local layouts of image signature relationships. The global layouts allow the user to inspect the relationships between all of the images in the entire database, whereas the local layouts help the user to focus on a much smaller subset of images in the database. The interface was built with the idea of having numerous ways in which to visualize the image relationships, independent of the measure used to compare the images. Such features include visualizing both the two-dimensional maps with a one-dimensional linear layout of a query point representing an image with a set of points representing a specified set of images. In the interface the points are displayed as small colored squares that we will call either points or glyphs. The image points in either layout can be selected so that the image it refers to can be displayed. Query images and the selected set of images are also displayed within in the interface as thumbnail sketches.

We developed several methods to lay out the points representing the images within a two-dimensional rectangular display, and thereby visually represent the relationships between images in the database. These methods are based on multidimensional scaling (MDS) techniques and effectively act like maps of the image relationships within the database. We use the term *layout* or *embedding* from the graph drawing nomenclature to mean the placement of vertices and edges on the 2-D plane. In our application, we do not show the edges. With edges removed, these are essentially scatter plots, which is the more common term used in statistics. In order to have some basis of comparison between images we used content-based image retrieval algorithms of the CANDID project,[12,13] but any similarity or dissimilarity measure would be suitable. The comparisons provide a dissimilarity measure between the image signatures and conform to a metric.[1] This allows us to use a standard MDS technique based on principal components, as well as iterative non-linear projection techniques to project points in what would be a large or infinite dimensional space down into 2 dimensions.

The layout algorithms are of two types. The first are those that project the all-pairs dissimilarities to two dimensions, presenting a many to many relationship for a global view of the entire database. The second are those that relate a query image to a small set of matched images for a one to many relationship that provides a local inspection of the image relationships. Both types are based on well known techniques that have been modified and used in conjunction for efficiency and effectiveness. They include the well know iterative nonlinear projection[10,6,9,1] and classical projection.[2] The global maps are hybrid algorithms using classical projection, or classical projection in conjunction with nonlinear projection. We have developed several one to many layouts based on a radial embedding, also using modified nonlinear and classical projection.

[1]The CANDID project provides content-based retrieval of digital imagery from massive databases. Digital signatures are automatically extracted from images and are used to identify similarities between these images. This methodology has been extended to support retrieval of multi-spectral satellite imagery, such as *Landsat*^{TM} data, allowing a user to locate geographic areas having similar spectral reflectance characteristics. CANDID has also been used to manage a database of medical imagery. A pulmonary CT data archive is searched to identify lungs that exhibit the similar gross pathologies as a current user-provided example.

2 PREVIOUS WORK

Previous work in this area can be found in the image database retrieval software, largely based on relational database technology coupled with various types of properties for image matching. Visualizing the relationships between objects in a database, however comes from the document database retrieval community.

2.1 User Interfaces for Image Database Retrieval

Ultimedia Manager and PhotoFlash each manage image and textual databases and use query-by-example for image retrieval with varying amounts of success. The interaction between their interfaces and the underlying databases provide adequate means for searching and browsing the databases common to most relational databases. They have a number of properties by which to match images, and can, as is the case for example with Ultimedia Manager, match on combinations of these properties. The effectiveness of these packages is however somewhat limited.[8] Finding a close match to an image or finding a small set of close matches does not always prove to be an easy task, since the match criterion from the human perceptual perspective is often difficult to quantify. The relative effectiveness of these methods is therefore quite subjective. These packages do nevertheless provide the user with numerous ways in which to search, which helps to ameliorate the subjective nature of the notion of a close match.

Recent work in document database retrieval has begun exploring the use of visually representing the relationships between documents using MDS techniques. Often the similarity measures are based on *n-gram* comparisons or keyword frequency. It is difficult, however, to capture semantic content with these measures and document retrieval methods, like image retrieval methods, suffers from a considerable degree of subjectivity as well. Using a number of different methods to help the user search and browse the document database is again a worthwhile goal, given the inherent subjectivity of the activity.

2.2 MDS

2.2.1 Iterative Non-linear Projection Technique

In the graph drawing community the iterative non-linear projection techniques are often referred to as spring embeddings or force-directed placement algorithms. In our implementations we use a form of the following stress function:

$$S = \frac{1}{\sum_{i=1}^{n}\sum_{j>i}^{n} d(i,j)} \sum_{i=1}^{n}\sum_{j>i}^{n} \frac{(d(i,j) - D(i,j))^2}{d(i,j)},$$

where $d(i,j)$ represents the dissimilarity between i and j, n is the number of objects, and $D(i,j)$ is the current dissimilarity or distance between i and j in the projected dimension. S is therefore the mean-square-error between the two distance or dissimilarities spaces. We can identify several forms of stress based on the following:

$$S_k = \frac{1}{\sum_{i=1}^{n}\sum_{j>i}^{n} (d(i,j))^{2-k}} \sum_{i=1}^{n}\sum_{j>i}^{n} \frac{(d(i,j) - D(i,j))^2}{(d(i,j))^k},$$

where $k = 0, 1, 2$. S_0 is known as the absolute stress and errors in long and short distances are equally weighted. S_1 is called semi-proportional stress and difference in the longer distances are not weighted as much, allowing greater leeway representing the larger distances. S_2 can be thought of as proportional stress, where the difference between the projected distances and the real distances is measured only as a fraction of the real distance. S_0 takes less time to compute, but S_1 and S_2 are generally more effective at finding global minimums.

Proportional stress can be thought of as proportional to the energy of a system of particles joined by springs. The relaxed length of the spring is $d(i,j)$, and the spring constant is:

$$K_{ij} = \frac{1}{(d(i,j))^2 n(n-1)},$$

where n is the number of particles. The problem is then to minimize the stress on the entire system of particles within the projected space, where all-pairs[2] of particles are connected by springs with varying relaxed lengths and spring constants.

By minimizing the stress function, the distances within the projected dimension closely approximate the real distances. The spring embedding algorithm works by placing points randomly within the projected dimension (in our case 2-D) and iteratively finding a new placement that minimizes the stress function through, for example, the steepest descent technique; namely, by moving the points along the gradient of stress.

In a method developed by Cohen,[1] rather than placing all of the points randomly in the projected dimension, the algorithm is performed on a small subset of points, initially placed randomly in the projected dimension. The algorithm is iteratively performed on the remaining particles by including into the original subset additional small subsets from the entire set. The algorithm halts when either the stress of the system has reached a minimum threshold value or a specified number of minimization iterations has been reached. The algorithm will not necessarily find global minimums. This is especially true as the size of the problem grows. The best success at finding a global minimum can be had by using S_2. For many applications, local minimums are sufficient.

Placing the points randomly within the projected dimension can often be a poor choice of an initial embedding for the algorithm. There have been several attempts to find more suitable initial embeddings that would help the algorithm reach the minimum threshold value in a fewer number of iterations. Some methods use local techniques such as placing the points with respect to one or two nearest neighbor points, or global techniques, such as the use of classical MDS (described below), whereby the points are placed with respect to the axes of the greatest variance within the original dissimilarity space.

2.2.2 Classical MDS

Given a matrix of distances between n points in Euclidean space, coordinates for those points can be found where the dimension of that space is less than or equal to $n-1$, such that the original distances between the points are maintained. The matrix, B, formed by the inner product of the coordinate vectors, can be found from the squared distances. The spectral decomposition of the inner product matrix in turn results in the coordinates for the points.

$$B = V\Lambda V^T,$$

where Λ is the diagonal matrix of eigenvalues and V is the matrix of corresponding eigenvectors. The rank of B determines the number of zero eigenvalues and hence the dimension of the space wherein the point-to-point distances are preserved. If we let p be the rank of B, then B has $n-p$ zero eigenvalues and can thus be rewritten as

$$B = V_1\Lambda_1 V_1^T.$$

Expressing the inner product with X to denote the $n \times p$ matrix of coordinates, we have

$$B = XX^T$$

From the spectral decomposition we have

$$X = V_1\Lambda_1^{\frac{1}{2}},$$

[2]These algorithms will work without all-pairs connections, providing some of the points are connected so that the graph formed by the points and connections is still connected.

where

$$\Lambda_1^{\frac{1}{2}} = diag(\Lambda_1^{\frac{1}{2}} \ldots \Lambda_p^{\frac{1}{2}}).$$

The distances are maintained and the coordinates have been recovered from the distance matrix.

By finding X in this manner the points already refer to their principal axes. This means that we can take the first several axes—those that represent the largest variance of the points within the space—and use the coordinates of the points on these axes. Thus, from the inner product matrix we can find the best possible projection of the points onto the small number of dimensions that we choose, such that the distances between the points are approximately preserved.

The approximations vary in success depending on what percentage of the total variance the specific eigenvalues chosen represent. The larger the number of dimensions that have a significant percentage of the total variance of all dimensions, the poorer the approximations will be if the projection is onto only a few dimensions. Since this often depends on the data, this method's effectiveness varies from application to application. This is one of the reasons why there has been much research into other forms of MDS.

3 CANDID Camera

The CANDID Camera application was developed to examine the utility of graphically assisted image database exploration. A major goal of the project was to produce a usable proof-of-concept system that facilitates the exploration of new layout algorithms in conjunction with the CANDID content-based image retrieval algorithms, and their combined effectiveness in exploring existing image databases. The color application is written in C++ for Unix platforms, and the underlying layout algorithms are written in C. The X/Motif Tools[7] are used for the interface components and the Kanvas widget[14] is used for layout area graphics.

In the following sub-sections we describe the configuration of the user interface along with the functions of its component parts, and give an example scenario of the use of the system.

3.1 Screen Display and Functions

The application consists of one main screen that contains five subwindows, as can be seen in the screen configuration depicted in Figure 1. The *query image* area contains the current query image. Once a query image is selected and displayed in this window, then a command can be selected that finds a specified number of closest matches to the query image and displays this set in three different windows, each with a different display described below. The bottom left-most window area contains a selected image from the database, not necessarily the query image. There is command button that when clicked on will make the selected image into the query image and display it in the query image area. The large *2-D map area* displays the local or global layouts with small squares representing the images in the database. We will call these squares *glyphs.* The currently loaded query image is represented by a large square glyph.

Once a set of closest matches is determined, the image glyphs of this set are displayed as slightly enlarged squares. The narrow retangular window in the upper right-hand portion of the main screen contains a linear, *1-D area* loaded with glyphs representing the set of images that most closely match the queary image. The matches are in sorted order from left-to-right, based on the scaled true dissimilarity measure between images and the query image. (Note that this is the same measure that is used to construct the proximity matrix.) Finally, a subset of the matches are displayed as reduced size images in the lower-right *thumbnail* area. A user may select any of the three representations of a match—a glyph from the layout area, a glyph from the 1-D area, or the thumbnail. Upon selection that image will be loaded for browsing in the *selected image* area. A selection on any of these three representations will cause all of them to highlight. This allows a user to see that image's location in each of those areas.

3.2 Sample Session

The following sequence of items describe a typical usage of the system.

Load Initial Layout Before a query can be performed a user must:

1. First select an image database. We created our own image databases of moderate sizes (anywhere from 120 to 310); a collection of Landsat images, a collection of ordinary photographs, and a collection of pulmonary computed tomography images.
2. Select a signature set for that database. The signature sets are pre-computed and, depending on how the set was computed, provide different different levels of accuracy and effectiveness of the pair-wise dissimilarity. This is often application dependent and can be very subjective in terms of providing reasonable measures.
3. Select a pre-computed, static global layout for the selected signature set.

Each image database may have multiple signature sets and each signature set may have multiple global layouts created for it. Any number of these all-pairs layouts can be browsed before queries begin. Each layout has one glyph per image and is initially scaled so that the entire layout is visible (see Figure *2a*).

Select Query Image An image must be loaded into the query image area before a query can be performed. The image can be selected from a textual list of all images in the database, or by clicking on any glyph in the 2-D area (see Figure *2b*).

Perform Query for Closest Matches Under the *Options* menu, the number of closest matchs can be selected. Once an image is loaded the user presses the query button to start the search for the closest matches. As described above, images matching the query (to a selectable maximun number of matches) are denoted by enlarging their glyphs slightly in the 2-D area. Additionally, the matching images are depicted in sorted order in the 1-D area by their true dissimilarity. Finally, the top matches (to a selectable maximum number) are loaded at reduced size in the thumbnail area. There are now three representations of the query results depicted in the interface (see Figure *2c*).

Browse Results Once the query results are shown a user may select an image for display by clicking on any glyph (in the 1-D or 2-D area) or a thumbnail. When a glyph or thumbnail is selected that image is loaded into the selected image area and all three of its representations (1-D, 2-D, thumbnail) are highlighted.

After a selected image is loaded the user can scan subsequent or previous matches (ordered by dissimilarity) by clicking on the left or right arrow buttons. Each click loads a new image and highlights its representations.

The user may also zoom in on areas of interest in the layout (see Figure *2d*) or color image glyphs based on textual keywords. For example, in the Landsat images that were in our database, Moscow, Cairo, Albuquerque and Los Alamos would be representative keywords stored in the image database (see Figure *2e*). The ability to color glyphs is especially useful in detecting distinct or overlaping clusters of images in the layouts.

At any time, the currently selected image can be loaded as a new query image and closest matches queries performed on it.

Load Dynamic Layouts Finally, the user may create dynamic, local layouts using only the closest match set and the query image. These layouts are created dynamically in real time (see Figure *2f*). Zooms can be performed and glyphs colored as in the static layouts. The user can switch back to the static layout at any time, select a new query image and closest match set, and continue browsing the image relationships.(see Figure 1(f)).

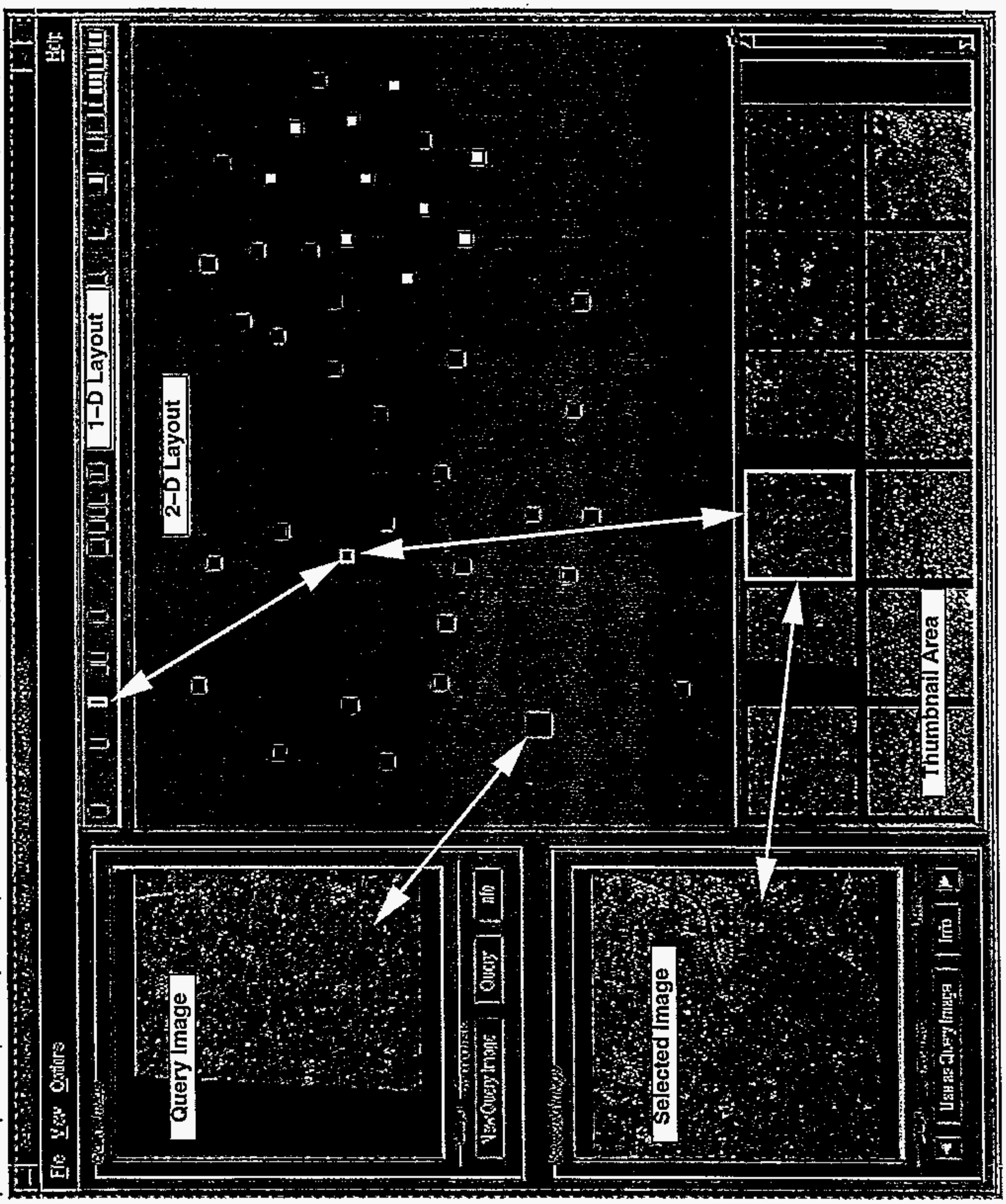
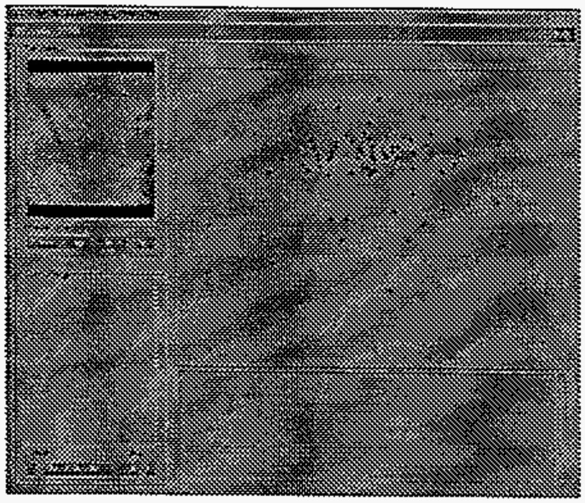


Figure 1: CANDID Camera screen configuration.

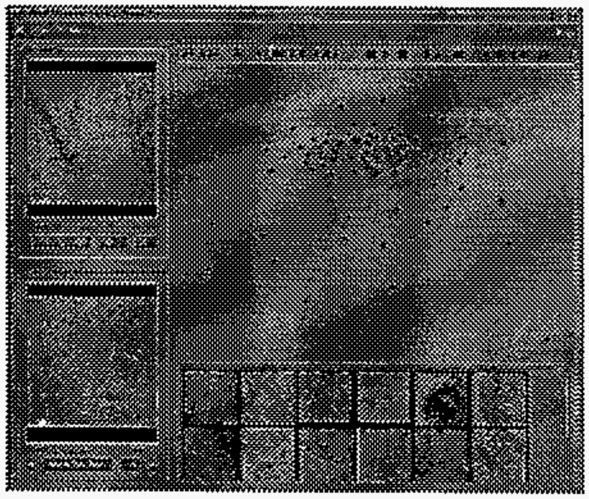
MacCuish

7/

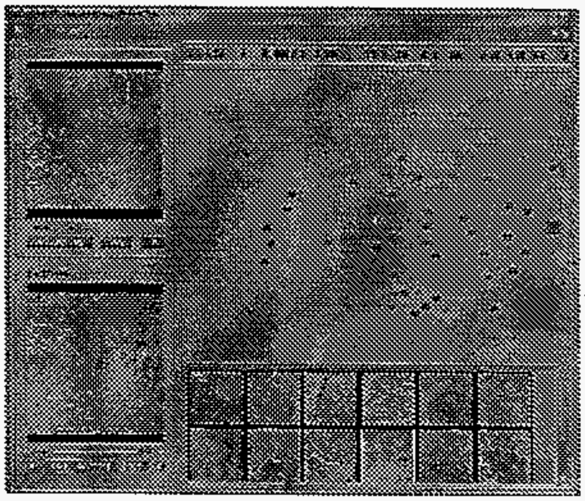
(a) Initial layout



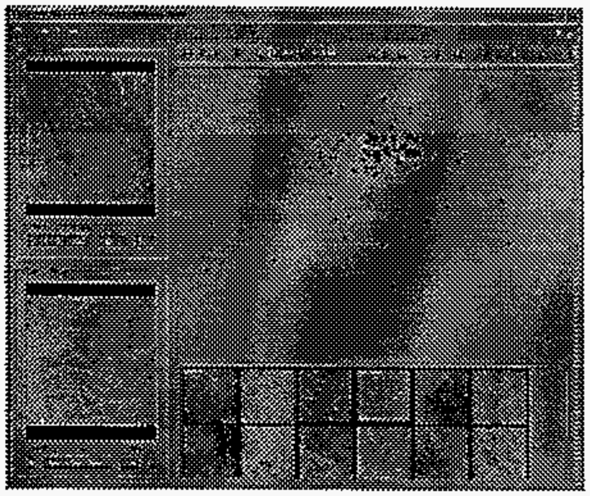
(b) Source selection



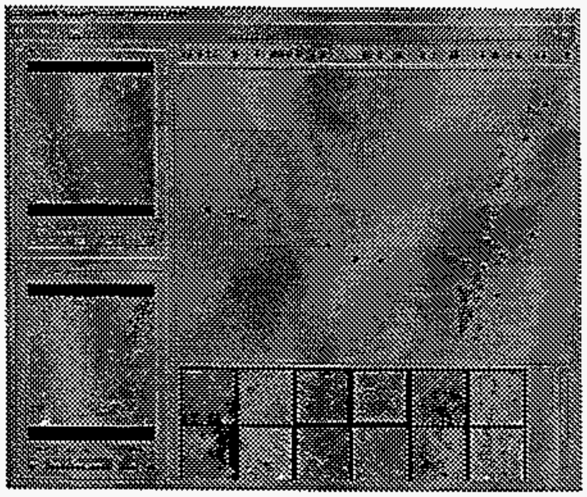
(c) Matching images



(d) Zoom layout



(e) Color glyphs



(f) Dynamic layout

Figure 2: CANDID Camera sample user session.

4 LAYOUT ALGORITHMS

The layouts that represented entire databases were computed off-line and stored, so that they could be called up immediately by the interface. Several different global layouts were generated for each database to give the user a number of different views. The local layouts typically represent a small subset of the database, and therefore can be computed in real-time. Most of the various types of local layouts take at most a few seconds. This time is within the necessary user-friendly response time for effective browsing. In most cases, where the number in the subset ranged between 20 and 50, we were able to obtain this performance. Occasionally, some of the larger subsets would take up to a few minutes to compute the radial spring layouts.

4.1 Global Layouts

For some of the global layouts we simply used the classical MDS technique described above, which was quite fast for all but the largest of our databases. The performance met or almost met the requirements of the interface, but nevertheless we stored these assuming that with larger databases we would have to pre-compute them anyway. Most of the variance of the dissimilarity data for each database was contained in the top two or three principal components, and often in the principal component. The two dimensional relationships therefore closely matched the $n-1$ dimensional relations. In many instances with our data numerous points would be grouped so close together that they would occlude one another. The non-linear projection tended to spread the points out, while maintaining the natural clustering observed with the MDS layout. As a result we felt that the global layouts produced by nonlinear projection provided more visually effective maps of the data.

Nevertheless, this method is inefficient for even a moderate number of points (say, 100-500). For large databases it would be entirely impractical. Though maps of large sets of points are of limited value since they typically only reveal global properties of the entire set, if such maps could be made relatively efficiently they would be of some value. The hope would be that the layouts would display a natural clustering of the data. For our application we used the incremental arrangement approach of Cohen.[1] In Cohen's algorithm two initial points are randomly placed, and each subsequent point is placed in proximity to its two nearest neighbor points that have already been placed in the plane. This helps to maintain the property that points near one another remain close, but there is no real improvement in the efficiency of the algorithm. However, if there is a slight perturbation for each placement, different trials of the algorithm will produce different layouts. In this way, numerous trials can be performed in order to find the best minimum of the stress function.

In order to speed up this algorithm, we obtained better practical results by removing this method of initially placing the points and substituting the placement of the points found by classical projection. For our data, we typically obtained 2 to 3 times speed up using this method which includes the time to compute the MDS layout. The drawback with this method is that small perturbations need to be introduced to the MDS layout for there to be different results for different trials. The locality of the nearby points however is largely maintained, so that points that are nearby one another rarely get inadvertently placed far apart.

4.2 Local Layouts Based on Queries

In order to have a more local view of the data relationships we wanted to create layouts based on a query image and a small subset of the remaining images in the the database. Our two goals were to maintain the relative distance to the query point for all of the points in the subset, and at the same time maintain an approximate interrelationship between the points from the subset. Users of the application could then select a query image and a subset of images and inspect the relationships of surrounding subset as well as the very real relationship of the query point to the subset of images.

Thus, for match queries we have developed circular layouts that, given a query image, an glyph representing that image is placed at the origin and the set containing k closest image matches are placed radially about the origin at a distance that is relative to the actual computed match distance. The first of these layouts is produced by creating the MDS layout of only the set of k closest image matches and the query image. All the points are translated such that the query image coordinates are at the origin. This layout can then be compared to a further refinement of the layout, where the radial relationships are maintained, but the points around the query point are projected along their respective vector from the origin by real distance, scaled to the screen coordinates.

The second algorithm involves using the non-linear projection on the set, and again the translation of all the points are made with respect to the query image coordinates, and again the real distance projection is made. The last circular layout we created uses nonlinear projection using the angular displacement, θ, to minimize the stress function. This involves first computing an initial layout that produces the initial θ values, such that the query image point is at the origin and the matched set is arrayed about origin. In order to compute the initial layout quickly, we simply used the MDS layout of the subset and query image point. Below are the simple steps of the algorithm:

- Compute MDS for the query set and include the query point in this set.
- Translate the query point to the origin and translate the remaining points in the set with respect to the query point.
- Maintaining the same angular relationship, scale the distance to the query point for each point in the query set to the respective actual distance.
- Calculate a nonlinear projection of the set with the fixed distances, converting the stress in the system from that based on the all-pairs distances to the sum of the angular relationships.
- Display the result of this calulation as the layout.

We found that with our data sets this last algorithm typically took 3-5 times as long as the second algorithm to reduce the stress in the system to roughly the same levels. Since we are not using trigonomic functions while moving the points each iteration, the computational work is not a factor, but rather the total number of iterations needed to compute roughly the same minimum stress values. We do not yet understand why this is so.

5 FUTURE WORK

5.1 Future CANDID Camera Enhancements

The initial application is a proof-of-concept, developed for research purposes. Any production version must be made more robust and existing interface paths streamlined. Additionally, here are some interface enhancements that may prove useful:

- Modify the layout presentation to handle viewing of large databases. At some point the use of one glyph per image will become overwhelming to the user. Techniques for filtering the volume of data down will need to be developed.
- The ability to view the extent of a group of images with common properties by dynamically displaying the convex hull around this set within the layout. This would allow the user to more easily detect distinct or overlaping clusters or classes of images within the database.

- Implement what is commonly refered to as a "fisheye"[11] view of the layout. This view would scale the layout in a non-linear, hyperbolic fashion around selected areas of interest. Using this technique users can still view the entire layout and yet concentrate on a more detailed presentation around the centers of interest.
- Display the image thumbnails in the layout area. These would need to be quite small thumbnail images, and probably useful in only when the number of images in the layout is small, such as in the fisheye areas of interest or in zoomed views. Nevertheless, this would allow the user to directly compare images in the layout without refering to the thumbnail or selected image areas.
- Interactive selection of images subsets that may then be operated on as if they comprised their own database.
- A history menu, whereby a record of image selections, queries, and layouts could be recorded and retrieved.
- Add additional matching capabilities, such as color and texture in combination or shape matching.

5.2 Layout Algorithms

There are numerous MDS techniques that could be tried in connection with image database retrieval. To be effective in assisting the user in searching and browsing the database, they must be fast. We would like to continue to explore improving the performance of the spring embedding algorithms, either through better initial embeddings or coding enhancements. This would allow larger subsets of images from the database to be laid out in real time. The global layouts can take a great deal of time, especially if the database is large. These, even for our relatively small database, often took as long as an 20 minutes to an hour on a *SPARCstation 20*. They necessarily have to be pre-computed. Another approach we have been exploring is triangulation techniques in conjunction with classical projection to obtain maps quickly, but also more visually effective, much like the non-linear projection layouts.

More extensive empirical testing is clearly needed, both in exploring the performance issues and in evaluating the effectiveness of the layouts on different data sets. Part of this process could also involve the inspection of the variance percentage of the first two principal components representing the two dimensions when using either the entire set or the matching subsets. Though the percentage of variance of the two principal eigenvalues was inspected for the each database, this was not done for the small query subsets. Providing this information for both the entire sets and the subsets would allow the user to understand the relative effectiveness of the classical MDS.

6 CONCLUSION

We found, when using CANDID Camera with the few databases that we explored, it was an effective browser of the database, both in terms of the interface features (such as the thumbnail matches and the linear match plot) and the global and local layouts. The global layouts showed that groups of similar images were clustered together, providing convincing evidence that the CANDID similarity measures are robust for our data sets. This was again true of the local layouts. The radial layouts were a quick and effective means of determining relationships between the query image and a small set of images, which enhanced the browsing capabilities of the interface.

These tools could well prove useful for other types of databases, not just image and document databases. With this also in mind, we feel from our experience with CANDID Camera and the layout algorithms thus far that their further development is warranted.

7 REFERENCES

[1] COHEN, J. Drawing graphs to convey proximity: An incremental arrangement method. Tech. Rep. R53-07-94 S-241,681, Department of Defense, 1994.

[2] COX, T. F., AND COX, M. A. A. *Multidimensional Scaling.* Chapman and Hall, 1994.

[3] CROW, V., POTTER, M., THOMAS, J., LANTRIP, D., STRUBLE, C., PENNOCK, K., SCHUR, A., WISE, J., FIEGEL, T., AND YORK, J. Multidimensional visualization and browsing for intelligence analysis. *In then Proceedings of GVIS'94*, PNL-SA-24870 (1994).

[4] CUTTING, D. R., KARGER, D. R., PEDERSEN, J. O., AND TUKEY, J. W. Scatter/gather: A cluster-based approach to browsing large document collections. *In the Proceedings of SIGIR'92* (1992).

[5] DAMENSHEK, M. Gauging similarity via n-grams: Language-independent categorization of text. *Science*, 267 (1995).

[6] EADES, P., AND SUGIYAMA, K. How to draw a graph. *J. of Information Processing 13*, 4 (1990), 424–437.

[7] FLANAGAN, D. *Motif Tools—Streamlined GUI Design and Programming with the Xmt Library.* O'Reilley and Associates, Sebastopol, California, 1994.

[8] FORSTON, R. L. Analysis of some current commercial and research methods for image retrieval using query-by-example technology. Tech. Rep. LA-UR-95-3880, Los Alamos National Laboratory, 1995.

[9] FRUCHTERMAN, T., AND REINGOLD, E. Graph drawing by force-directed placement. Tech. Rep. UIUCDCS-R-90-1609, Department of Computer Science, University of Illinois at Urb–Morana-Champaign, June 1990.

[10] JAIN, A., AND DUBES, R. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[11] KAUGARS, K., REINFELDS, J., AND BRAZMA, A. A simple algorithm for drawing large graphs on small screens. *In Lecture Notes in Computer Science: DIMACS International Workshop, Graph Drawing '94 894* (1994), 278–281.

[12] KELLY, P., AND CANNON, T. Experience with candid: Comparison algorithm for navigating digital image databases. *In SPIE Proceedings of the 23rd AIPR Workshop on Image and Information Systems: Applications and Opportunities 2368* (1994), 64–75.

[13] KELLY, P., AND CANNON, T. Query by image example: the candid approach. *In SPIE Storage and Retrieval for Image and Video Databases III 2420* (1995), 238–248.

[14] LEON, J.-M. Kanvas: A Motif canvas widget. `http://www.inria.fr/koala/jml/leon.html`, 1995.