1999-06-01

Sketch-based Image Queries in Topographic Databases

James Carswell
Technological University Dublin, james.carswell@tudublin.ie

Recommended Citation

Carswell, J..: Sketch-based image queries in topographic databases. *Journal of Visual Communication and Image Representation*; Vol. 10, No. 2, pp. 113-129. June, 1999. doi:10.1006/jvci.1999.0414

Manuscript No. JVIS98-0004 **REVISED**

SKETCH-BASED IMAGE QUERIES IN TOPOGRAPHIC DATABASES

by

Peggy Agouris
Anthony Stefanidis
James D. Carswell

Dept. of Spatial Information Science & Engineering
and the National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469-5711
{peggy, tony, carswell}@spatial.maine.edu

Proposed Running Head:
Sketch-Based Image Queries
in Topographic Databases

Contact Author:

Peggy Agouris
Dept. of Spatial Information Science & Engineering
and the National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469-5711
Tel: (207) 581 2180
Fax: (207) 581 2206
peggy@spatial.maine.edu

Abstract

In this paper we present the development of a system prototype for sketch-based queries for the content-based retrieval of digital images from topographic databases. We discuss our overall strategy and associated algorithmic and implementation aspects, and present associated database design issues. The query tools devised in this research are employing user-provided sketches of the shape and spatial configuration of the object(s) which should appear in the images to be retrieved. Our matching tool is inspired by least-squares matching (lsm), and represents an extension of lsm to function with a variety of raster representations. Our strategy makes use of a hierarchical organization of feature shapes within a feature library. The results are ranked according to statistical scores and the user can subsequently narrow or broaden his/her search according to the previously obtained results and the purpose of the search. Our approach combines the design of an integrated database environment with the development of a feature library and the necessary matching tools. We discuss our overall strategy and individual database components, and present some implementation results.

1. INTRODUCTION

Advancements in sensor/scanner technology have resulted in the availability of constantly increasing volumes of digital imagery. Geosciences and spatial information engineering have been greatly affected by this development. In particular, digital photogrammetric applications have become more robust, moving from the experimental use of few images to large scale projects which employ numerous images. We have reached a point where digital images have practically substituted analog ones (e.g. prints, diapositives, or negatives) as the popular medium for the extraction of precise and up-to-date spatial information like digital elevation models (DEMs), or features with their precise 3-D coordinates. The increased volume of digital imagery necessitates the development of novel methods to efficiently retrieve images from large digital image databases.

Intelligent image retrieval from large databases is one of the novel applications which are receiving increased attention in the computer vision community [2,8,9,10,14,15,19]. Some prototype systems have also been reported, including Chabot [14], IBM's QBIC [5], VisualSeek [18], ImageRover [16], and PicHunter [3]. The common trend in these efforts is that they focus in general-use multimedia-type image databases. Such a database includes for example images of sunsets, cartoon characters, snow-covered mountains, and wild animals. The objective of a query might be to retrieve the images of sunsets from the database. In such a scenario, low-level image properties (e.g. color, pattern) are adequate for information retrieval, since the image members of the database display substantial differences in these properties and can be distinguished by them alone. However, topographic image databases contain very large numbers of images (typically aerial and/or satellite) which represent striking similarity in terms of general low-level image properties. Therefore, general-purpose image retrieval approaches like the ones mentioned above are not sufficient for information retrieval in topographic image databases. Instead, what distinguishes images in a topographic database is the *shape* and *configuration* of the objects they contain.

In this paper we present our approach for the retrieval of images from topographic image databases using query-by-sketch operations. Our approach is *progressive*, as we employ increasingly specific information to retrieve imagery. It is also *content-based*, with the term *content* referring to objects depicted in the images. We present the strategy and design considerations behind *I . Q*. (Image Query), our prototype system for image retrieval (section 2). We also emphasize on the development of a novel matching tool used in our system to match object outlines for querying (section 3). Furthermore, we present in detail a new approach for the organization and structure of a feature library to support our queries (section 4), as well as the maintenance and updating of this feature library. It should be mentioned that while our research originates from topographic applications, the developed concepts and methodologies can be applied to any type of imagery.

2. STRATEGY AND SYSTEM DESIGN

A description of our operation environment for our system is shown in Fig. 1. A searchable topographic database comprises images (typically aerial or satellite), outline/object information for these images, and metadata[1]. Until today, information retrieval in topographic image datasets is mostly supported by metadata (e.g. describing image location, date of capture, and scale). The support of metadata for image retrieval is compromised by two emerging trends:

- the increasing availability of multi-temporal image representations of specific regions, essential for GIS updating; in such images general metadata properties remain similar (besides, of course, the date parameter) while certain objects within the images evolve, and
- the need to make complex scene understanding decisions, based on the behavior (existence, modification, absence, specific relative positions) of objects within scenes, an increasingly important issue within modern integrated GIS environments.

[1] The term *metadata* refers to data about data: various forms of information (e.g. sensor characteristics, date of

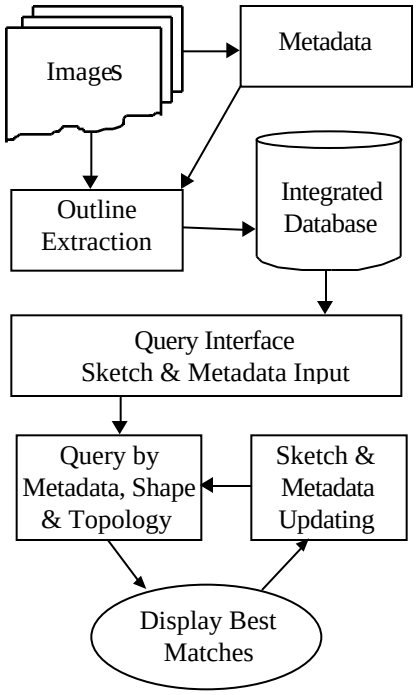

Fig. 1: Operation environment for *I.Q.*

The aim of our strategy is to take advantage of the intuitive method humans use to express spatial scenes, namely *sketching*. In accordance, the query interface of *I.Q.* allows us to access individual members of this database by using as input parameters *metadata* information and a *sketch*. Our approach is designed to proceed as follows:

- an operator sketches[2] a configuration of objects,
- he/she also provides additional metadata information, and
- the database is searched to yield the images which satisfy the given metadata information, and in which spatial configurations similar to the given sketch appear.

---

capture, resolution) describing/enhancing the content and/or properties of common data files (e.g. digital images, DEMs, maps in digital format).

[2] The term *sketch* is used here in a broad sense. It can refer to an on-screen drawing of an outline using standard software tools. It can also refer to a pre-existing outline (e.g. extracted from a digital image or a scanned map) selected by the user.

In order to support our queries, our searchable database comprises three components (Fig. 2):

· Image library: contains one entry for every image of the database, and provides a link/pointer to the corresponding filename.

· Metadata library: contains a listing of potential values for a set of attributes which describe general properties of the image. These attributes include date and time of acquisition, date and time of introduction in the database, scale/resolution, and location of the image (expressed in hierarchically arranged geographic entities like state, county, city). For more complex databases the attributes may be extended to incorporate sensor information and imagery type (e.g. b/w, color, pseudocolor).

· Feature library: contains a set of distinct features (i.e. object shapes) and links to image files where such features appear. The role of the feature library is to provide the crucial link which allows us to reduce the search space of a query from a database to an abridged group of features.

In addition to the above, the database may include a semantic library, which will contain semantic object information (e.g. object X is a hospital) and the corresponding links to image files.

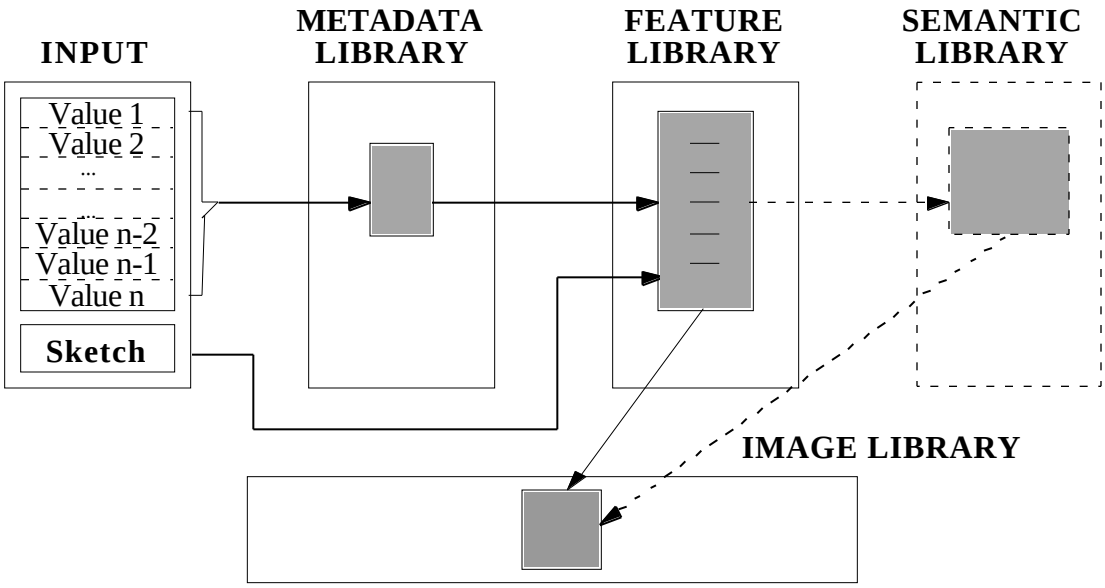

Fig. 2: Database design

Under this design, during the *on-line* part of a query, the input query outlines are matched to elements of the feature library using an on-line matching tool, and acceptable matches give links to specific images and locations within them. The metadata information is used to eliminate potential candidates within the feature library. Entries in the feature library contain links to image files and locations within these files, and an analysis of metadata values given as query input makes certain image files invalid candidates. Elements of the feature library which have links only to invalid images are temporarily inactivated. If a semantic library is also used, the matching results will have to pass through another check (whereby the semantic properties of the detected objects will be examined), and the query results would be provided after this added step. This information is returned to the user who then has the option to edit his/her query.

In this sense, a *query match* is an image which satisfies the given query parameters. Database searching is essentially performed in a progressive manner. Metadata information is used to thin the pool of potential matches (*query by metadata*). Additionally, an analysis of shapes is performed to identify the best matches of the query, and to assign estimates of confidence to them (*query by shape*).

In order to support the above, another sequence of actions has to be performed *off-line* every time an image is introduced into the database. The user inputs manually the appropriate metadata information for this image, and the metadata library is updated to add the new entry. Subsequently, objects/features are extracted from the input image using any digital image analysis tools, and these new features are compared to the existing feature library using our off-line matching tool. Library entries are updated accordingly to include links to the objects existing within the newly introduced image. Links between metadata and feature libraries are also updated, to connect the metadata values of the new image to the features detected in it.

The rationale behind our database design becomes apparent when analyzing the meaning of the metadata and feature libraries, and their connection. Assuming that we have *n* distinct metadata values, the metadata space is an n-dimensional one. A point within this space corresponds to all images of the same area, captured at the same scale, at the same date, with similar sensor. When one or more of these parameters can accept less specific values we move to blobs within the n-dimensional metadata space. For example, photos of various scales of a specific area taken on a specific date form a blob in the metadata space. This blob represents the scale space of the area at the time of data capture. When defining points (or blobs) of the metadata space we actually define a set of representations of a specific geographic area, and of the features within it. When querying the database, we use the metadata information to narrow the area of interest, and then we perform shape query against the shapes that we expect (from the off-line establishment of links) to exist in our region of interest. In essence, our design reverses the traditional processes by which we identify objects in the geographic space. Instead of identifying an object and then positioning it, we now identify a region and access a list of objects within this region. Computationally, metadata searches are inexpensive and fast. On the other hand, shape-based searches are in general computationally demanding, but allow us to move from global image properties - which are conveyed by metadata - to individual features (*content*) within images. By using metadata properties to narrow the search space for

subsequent shape matches we gain computational time without compromising the quality of the query results.

3. MATCHING CONSIDERATIONS

The matching tools developed for the environment presented in the previous section need to handle both the *off-line* (population and update of the feature library) and *on-line* (comparison of query input to the feature library entries) matching parts of this project. Conceptually, our matching algorithms are a variation of least squares matching (lsm), modified to function with edge files.

3.1 Least Squares Matching

Least squares matching (lsm) offers a robust method for establishing correspondences among image windows. Its mathematical background, based on least-squares principles, permits its successful extension for application in a multiple image matching scheme [1], or even for the establishment of correspondences in sets of 3-dimensional images [12].

Assuming $f(x,y)$ to be the reference edge template and $g(x,y)$ to be the actual image patch, a matching correspondence is established between them when

$$f(x, y) = g(x, y) \qquad (1)$$

However, considering the effects of noise in the actual image, the above equation becomes

$$f(x, y) - g(x, y) = e(x, y) \qquad (2)$$

with $e(x,y)$ being the error vector.

In a typical least squares matching method, observation equations can be formed relating the gray values of corresponding pixels. They are linearized as

$$f(x, y) - e(x, y) = g^{o}(x, y) + \frac{\partial g^{o}(x, y)}{\partial x} dx + \frac{\partial g^{o}(x, y)}{\partial y} dy \qquad (3)$$

The derivatives of the image function in this equation express the rate of change of gray values along the x and y directions, evaluated at the pixels of the patch. The two patches are geometrically related through an affine transformation

$$x_i = a_{11} + a_{12}x + a_{21}y \qquad (4)$$

$$y_i = b_{11} + b_{12}x + b_{21}y \qquad (5)$$

The affine transformation parameters are the unknowns which allow the repositioning of the image window to a location which displays better radiometric resemblance to the reference template. They are introduced in the derivative terms $(\partial g/\partial x, \partial g/\partial y)$ of the linearized observations above as

$$f(x,y) - e(x,y) = g^o(x,y) + g_x da_{11} + g_x x_0 da_{12} + g_x y_0 da_{21} + \\ + g_y db_{11} + g_y x_0 db_{12} + g_y y_0 db_{21} \qquad (6)$$

The resulting observation equations are grouped in matrix form as

$$-e = Ax - l \quad ; \quad P \qquad (7)$$

In this system, l is the observation vector, containing gray value differences of conjugate pixels. The vector of unknowns x comprises the affine transformation parameters, while A is the corresponding design matrix containing the derivatives of the observation equations with respect to the transformation parameters, and P is the weight matrix. A least squares solution allows the determination of the unknown parameters as

$$\hat{x} = (A^T PA)^{-1} A^T Pl \qquad (8)$$

Through the adjusted transformation parameters we determine a new position in the image as conjugate of the template. The robust mathematical foundation of least-squares matching allows us to obtain meaningful statistical measures for the accuracy of the matching process. The a posteriori variance of unit weight

$$\hat{\sigma}^2{}_o = \frac{V^T PV}{df} \qquad (9)$$

is an excellent measure of the overall accuracy. The residuals vector V expresses the deviation

between observations and their adjusted values (and is actually an evaluation of e in Eq. 7 once we obtain estimates of the unknown parameters x). The degree of freedom (df) of the system expresses its redundancy and equals the difference between formed equations and unknown parameters. An expression of the a posteriori variance is used as a score index reflecting the confidence level of a match.

The above system can also be increased to incorporate additional parameters. By taking advantage of the diffusion equation of the Gaussian function [11], according to which

$$\frac{\partial g(x, s_x)}{\partial s_x} = \frac{1}{2} \frac{\partial^2 g(x, s_x)}{\partial x^2} \qquad (10)$$

the derivative with respect to the scale parameter is equivalent to the second derivative of gray values with respect to the spatial coordinate, allowing us thus to directly introduce it in the linearized least squares matching observation equations (with the second derivatives of gray values as corresponding coefficients in the Jacobian matrix A of equation 7). Thus, first derivatives of gray values express positional/rotational differences between two conjugate windows, while second derivatives express scale differences among them.

3.2 The I.Q. Matching Tool

Our matching tool is a modification of lsm to function using object outlines instead of gray values, and avoiding computationally expensive matrix manipulations. We obtain a solution by analyzing the dissimilarities between a template (user-provided sketch) and an outline file window. When a template is compared to an edge file, the template is divided into four quadrants. Each quadrant is matched separately to a corresponding image area, and template pixels vote to stay put (or move) according to their similarity (resp. dissimilarity) to corresponding file pixels. Moves can be performed in the +/- x and y directions, in one of 5 options: left, right, up, down or stay put. This resembles the comparison of gray values in least squares matching and the use of image gradients to identify shifts, rotations, and scalings. The sum of the votes within each quadrant is recorded, together with recommendations for the

direction and magnitude of move. An example of template repositioning situations may be found in Figure 3. An analysis of the votes of all quadrants allows us to solve sequentially for shift, rotation, and scale parameters, to reposition the template to better match the file object. The final solution is obtained after a set of iterations. Furthermore, by analyzing voting patterns within different quadrants we can turn quadrants off, to accommodate occlusions.

The analysis of the quadrant shift directions and distances proceeds first through translation, then rotation then scale. The amount of translation is taken as the average distance calculated from the 4 quadrants. For rotation, the transformation of coordinates for the feature pixels are calculated using:

$x' = x \cos \alpha + y \sin \alpha$

$y' = y \cos \alpha - x \sin \alpha$

where α is the angle of rotation between the x' axis and the positive x axis. The rotation angle is taken in 15° increments, positive in the counter clockwise direction. If upon the next iteration the quadrant shifts determine a rotation angle in the opposite direction, then it is halved to 7.5°. This halving of the rotation angle continues as long as the direction for rotation keeps alternating between positive and negative α or until the arbitrary limit of 20 iterations is met.

For scaling the feature, there are two approaches: independent axis and global. For the independent axis scaling, it is only used when the user digitizes his own feature to match, therefore in the on-line matching process. This will allow for the feature to scale different amounts in the x and y directions. For global scaling, it is used in the off-line matching process and may be used in the on-line matching process if the user chooses an existing feature from an image to search the feature library against. Global scaling assumes a constant scale change in all directions which is usually the case when images are scanned at different resolutions or taken with different focal lengths or flying heights. For example, if $I.Q.$ determines through

analyzing the quadrant shifts that scaling in the x direction only is required, the entire feature will scale the same amount in all directions.

During the *on-line matching* process, a feature sketch template is matched against the feature library elements using the same matching tool. Once the template has settled onto a match, the matching accuracy is determined by how many of the template pixels continue to vote to move compared to those that vote to stay put. The comparison of a template to various feature library entries will result into matching accuracy estimates. These matching estimates can be used to determine ranking of different matches. During the *off-line matching* process feature library templates are compared to outline files (e.g edge files, or GIS layers coreferenced to digital image files) to establish matches and links to the corresponding image files.

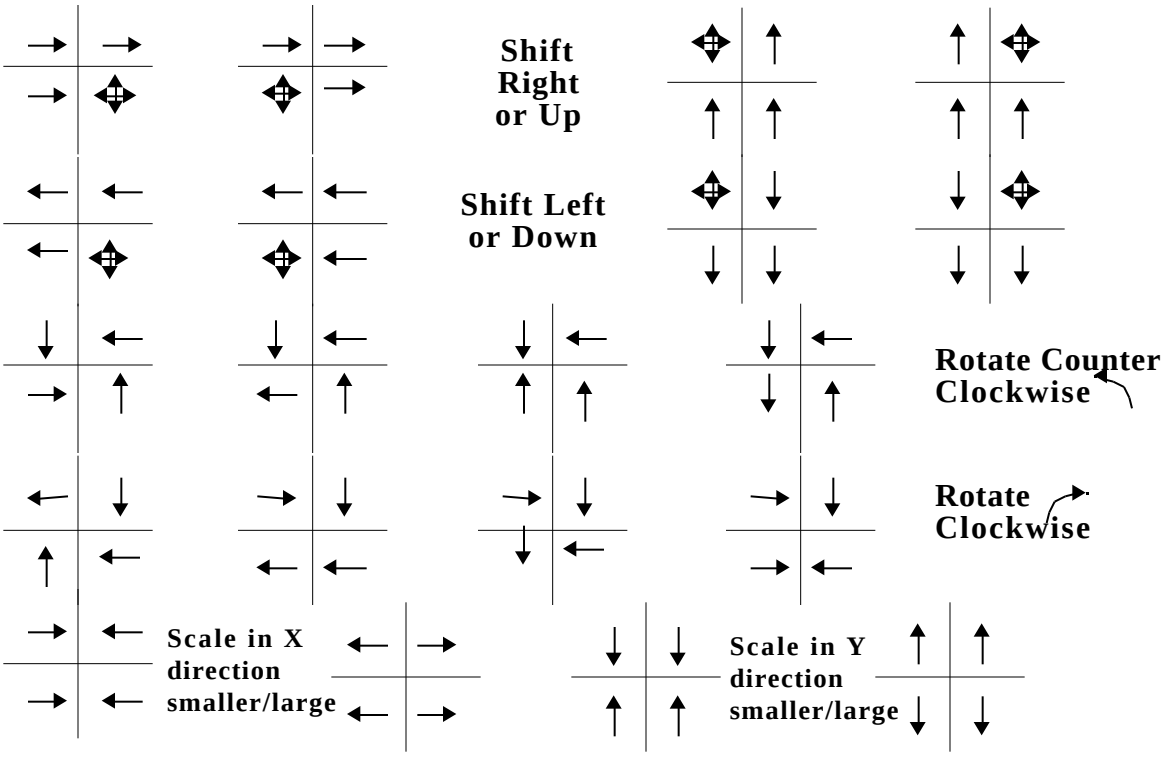

Fig. 3. Examples of template repositioning.

4. ORGANIZING THE FEATURE LIBRARY

The feature library permits us to narrow the search space from a large database of images to a limited group of feature outlines. In order for the query to be efficient, the library needs to be organized in an optimal way. The optimality criteria are two: the members of the library should be *exhaustive* (thus being able to describe all possible query input features), and the members of the library should be *independent* (minimizing unnecessary duplications). The two properties, when satisfied, are equivalent to an ideal library, which is approaching a base spanning the space of shapes.

The organization of data within the feature library is intended to enable it to act like a multi-stage screening mechanism that minimizes the risk of wasting considerable time making passes over extensive data that have no chance of selection. For example, the first screening criterion

will eliminate as potential matching candidates most of the features within the library. The secondary screening criterion will eliminate the next greatest number of alternatives and so on down through the feature library tree hierarchy.

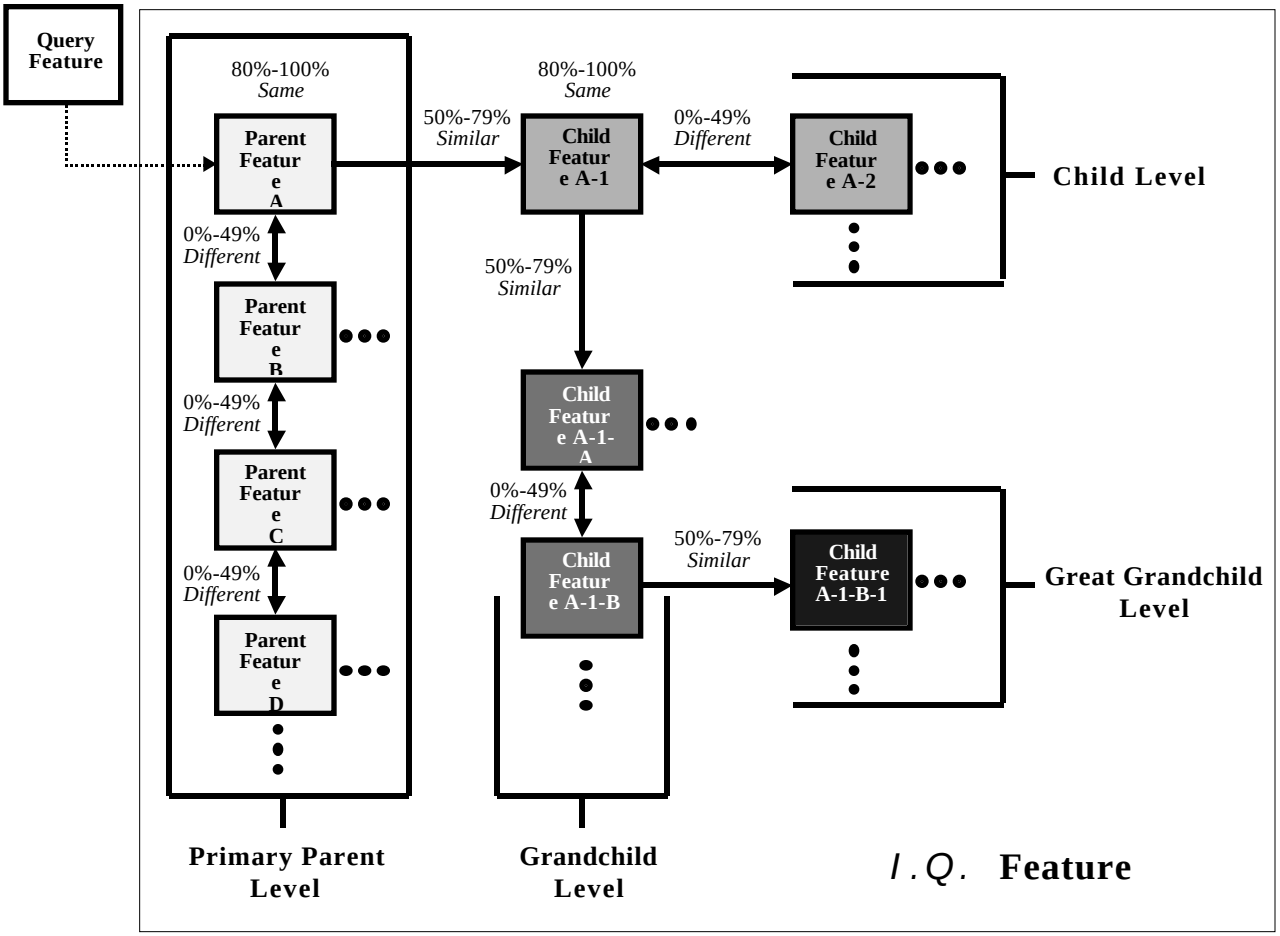

Fig. 4 - Feature Library Hierarchy

Furthermore, due to the dynamic natures of the image database and query processing, the feature library is constantly adding, subtracting and otherwise updating its features, links, and internal organization. It therefore needs to be autonomous in that it be able to automatically

maintain its contents depending on the changing states of these external but integrated components.

Our feature library is organized according to a hierarchical (tree-like) structure (Fig. 4) as follows:

The Primary Parent Level - where every new query shape is first tested against, and its respective matching percentage recorded (Fig. 5). The features at this level are the roots of all trees comprising the feature library. Different features within the parent level are considered dissimilar, as their mutual similarity percentages are lower than 50%. When a query is performed, the matching percentage between the query feature and a parent feature will fall in one of three ranges: (0%-49%), (50%-79%) and (80%-100%). In the latter case, with a matching percentage in the (80%-100%) range, the query feature is considered to be the "same" as the parent feature. The links the parent feature has to the images in the image database are then returned as the results to the query. In the event that there exist more than one parent feature that matches 80% to 100% to the query feature, the links of the highest matching parent (higher matching percentage) are returned first, with the links of the remaining matches following. In practice, this case of multiple parent matches is highly unlikely, due to the mutual dissimilarity of the parent features, but it can occur in cases where the input design is generic and non-complex (e.g. a simple square). If the matching percentage is in the (0%-49%) range, then the query feature is considered to be "different" than the parent feature. If this holds for all parent level features, the query feature is a candidate to be inserted into the feature library, at the primary parent level. Subsequent off-line processing is then required to establish its links to images in the database.

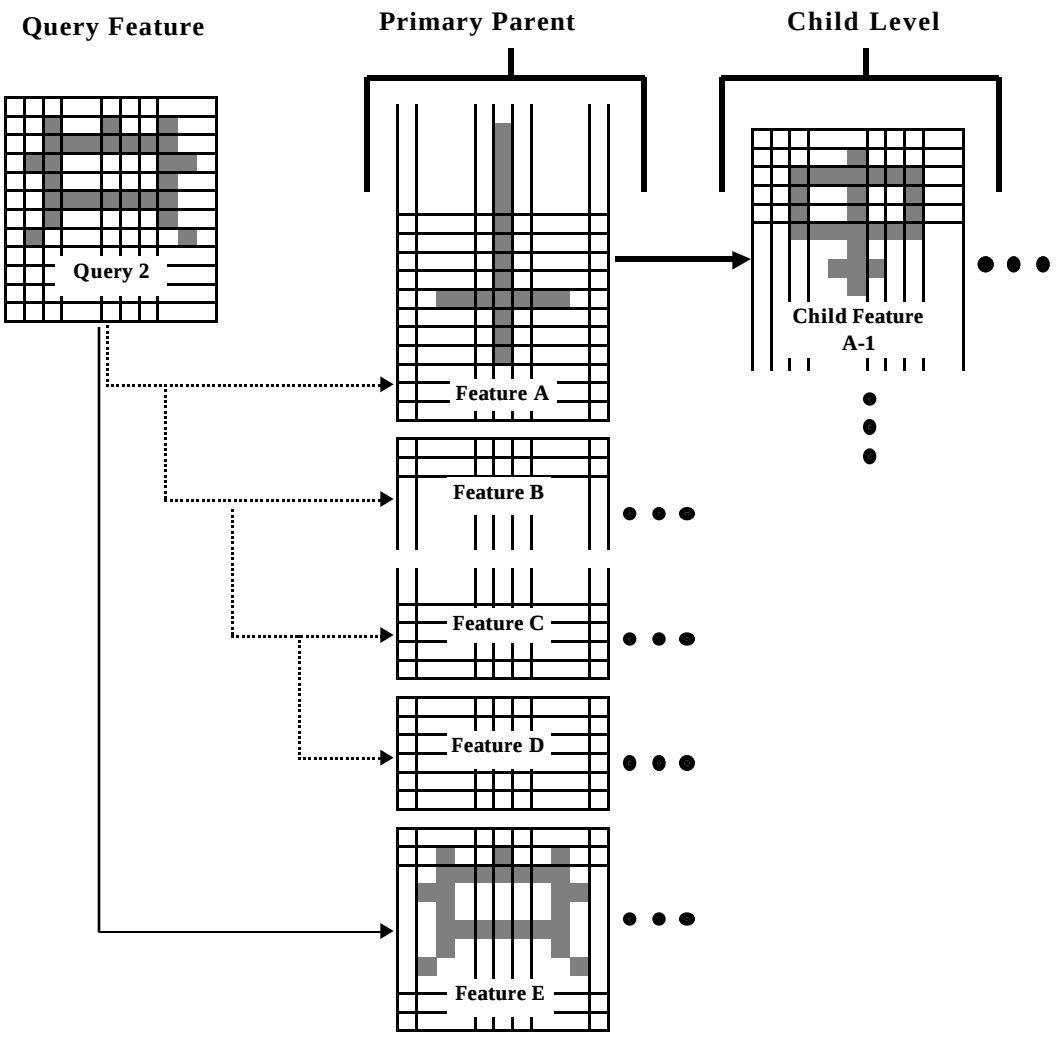

Fig. 5 – Query feature matching and insertion at the primary parent level. In query 1, Feature A-1 matched "similar" to Feature A and was inserted as Child Feature A-1. In query 2, the query feature matched "different" to all existing features at the parent level so is inserted as a new Parent Feature E.

The Child Level - If the query feature matched in the (50%-79%) range to any parent feature, it is considered "similar" to the parent and is then tested against this parent's respective child features. In case of multiple parent candidates, we select the one with the highest percentage first and continue with other candidates in order. If there are as yet no child features for the selected parent, the query feature gets inserted into the feature library as a descendent of the best matched parent (Fig. 5). Its links are then added and prioritized through off-line matching. If there are child features already residing at this level for any of the selected parents, the query feature gets matched against each of them. If the matching percentage is in the (80%-100%) range, the query feature is considered the "same" as this child feature. The links of this child feature to the image database are returned as the results to the query. If the matching percentage is in the (0%-49%) range, then the query feature gets inserted into the feature library at the child level of the best matched parent, and additional links are added and prioritized through off-line matching.

The Grandchild Level – If, after matching in the (50%-79%) range to a parent feature, the query feature matched in the (50%-79%) range to any child feature, the query feature is tested against the respective grandchild features. Obviously, the relationship between child and grandchild is similar to the one between parent and child. Similar processes to the ones already described take place at this library level to identify the "same" grandchild, to establish a new grandchild, or to move further down the tree to the level of great-grandchildren. Combined, these similarity tests that have to be satisfied within branches of the tree structure form the *insertion testing* criterion. The children levels may be alternatively referred to as *n-child levels,* with the child being 1-level, grandchild being 2-level child, etc.

The matching percentages mentioned here are intended to serve as example values. The percentages may become lower or higher, and they can be considered as tuning parameters of the approach, with their variations affecting the structure of the feature library in a predictable and organized manner. For example, by narrowing the high ("same") matching range - e.g.

using the range (90%-100%) instead of (80%-100%) – we increase the number of distinct entries at each level of the tree structure, making the tree wider and more shallow. This will increase the search space and make the processing time longer. However, it would also allow the queries to become very specific, i.e. “retrieve images containing features which look *exactly* like the query sketch”, as opposed to “retrieve images containing features which look like the query sketch”. Widening the high range – e.g. using a (70%-100%) matching range instead – will result in fewer entries in the database, and will make the tree deeper and more narrow. In turn this produces faster queries, with less accurate results. Narrowing or widening the second (“similar”) matching range will have comparable effects to the structure of the feature library.

5. Feature Library Housecleaning

The above described method, whereby new features are compared to existing ones and progress through tree levels until they are eventually matched to an entry or inserted in the library might result in some inconsistencies. For example, it is possible that a new query feature might match above 50% to a particular child/sub-child feature but less than 50% to any of the current parent features. This results in the query feature being inserted into the feature library at the primary parent level although the child/sub-child feature that matched better than 50% to this newly inserted parent feature might now be residing under the “wrong” parent, i.e. a parent feature that matches less well (to its child) than the newly inserted parent feature. In this case the child feature must shift its position within the feature library to reside under the “better” parent in the tree hierarchy.

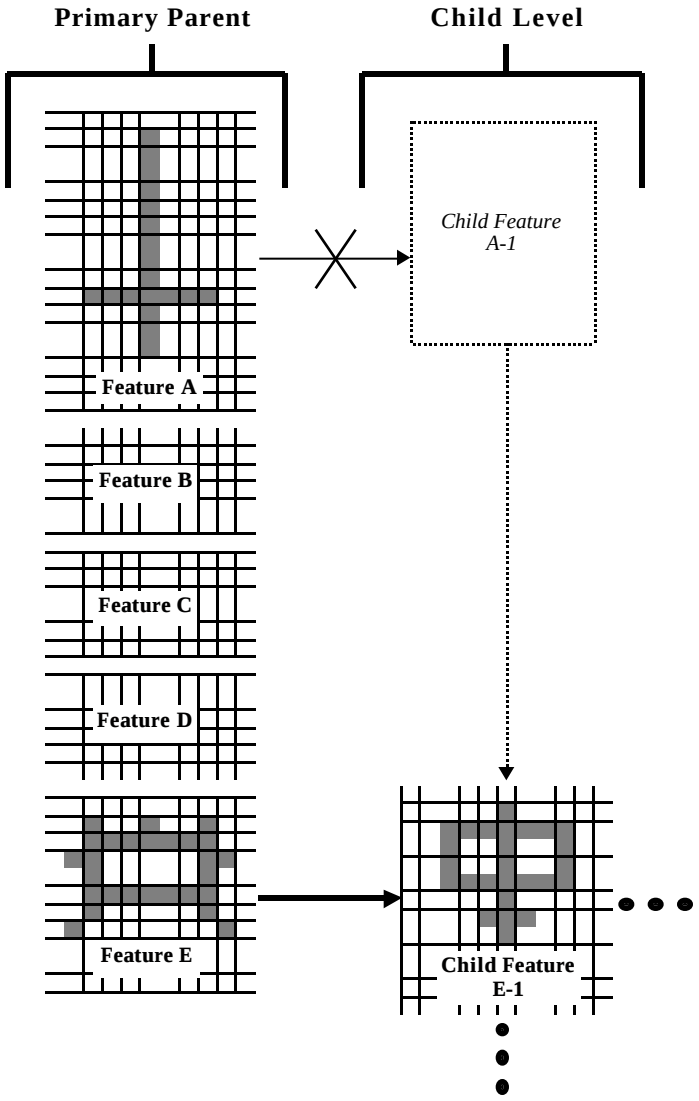

Fig. 6 - Shifting Feature Positions. Child Feature A-1 is tested against newly inserted Parent Feature E and finds a better match, i.e. 19/26=73.1%. It shifts position within the feature library to reside as a descendent of Feature E with all of its image database links intact.

Another example of a potential inconsistency in the feature library could occur when two sub-children from different primary parents match closer together than to any other feature currently within their respective tree hierarchies. Depending on the temporal sequence of feature insertion, this phenomenon could produce unnecessary duplicates in some cases (Fig. 7a) and necessary duplicates in others (Fig. 7b). This is due to the allowable range of percentages considered for

"same" (i.e. 80% or above) and "similar" (i.e. 50% to 79%) matching. If the feature library did not allow for this degree of uncertainty in its matching by allowing for exact matching only, this phenomenon would not exist. These, and other potential inconsistencies are rectified by applying general feature "housecleaning" (which utilizes the temporal aspect of feature insertion) to the feature library. This process is performed off-line, to ensure the continuous proper organization.

Feature library housecleaning is a process that corrects inconsistencies of the feature library tree hierarchy and is run by default at regular intervals (e.g. at the end of a session, or every time a preset number of new features ahs been inserted in the library). It ensures that all child/n-child features are residing under their proper parent and minimizes the theoretical possibility that unnecessary duplicate features could exist somewhere in the tree hierarchy (i.e. child/sub-child features that are between 80% and 100% similar). It also ensures that each of the parents are themselves unique.

Feature housecleaning utilizes a self-organizing table called the "Temporal Feature Index" (TFI). The TFI (Fig. 8) is a two dimensional cross referencing table of feature filenames together with their respective matching percentages. Features inserted into the feature library are simply appended to this table in the order that they are introduced in the library, and all their matching percentages recorded as they occur. Thus, ordering within TFI reflects relative insertion time. The time of insertion is important so as to reduce both the number of features to test and the number of features to test against. For example, it is not necessary to re-test Child Feature A-2-A against Parent Feature A as this matching percentage is already known and recorded previously at time of insertion.

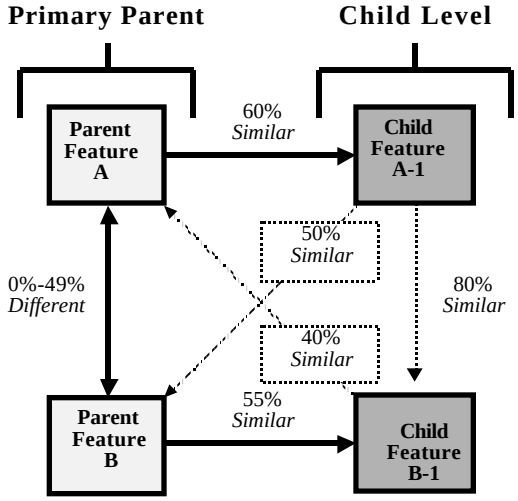

Fig. 7a - Unnecessary Duplications. Parent Features A and B already exist in the library. Child Feature A-1 matches 60% to Feature A and 50% to Feature B. It therefore gets correctly inserted as Child Feature A-1. Child Feature B-1 matches 40% to Feature A and 55% to Feature B so gets correctly inserted as Child Feature B-1 without testing against Child Feature A-1. Child Feature A-1 is 80% similar to Child Feature B-1 and 50% similar to Feature B so should pass its links to Feature B-1 and be removed from the feature library.

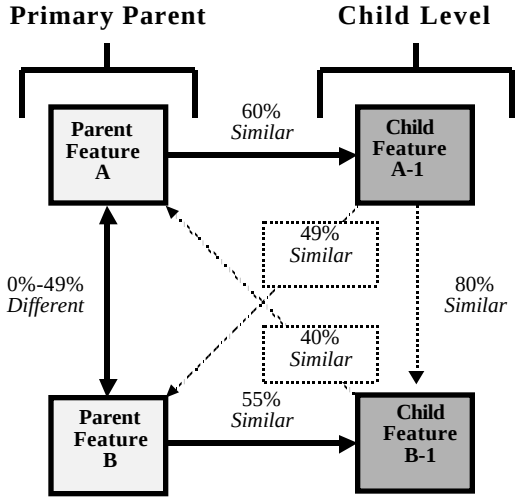

Fig. 7b - Necessary Duplications. Parent Features A and B already exist in the library. Child Feature A-1 matches 60% to Feature A and 49% to Feature B. It therefore gets correctly inserted as Child Feature A-1. Child Feature B-1 matches 40% to Feature A and 55% to Feature B so gets correctly inserted as Child Feature B-1 without testing against Child Feature A-1. Child Feature A-1 is 80% similar to Child Feature B-1 but 49% similar to Feature B so is not allowed to shift its position under Parent Feature B in the tree hierarchy - thus maintaining library consistency.

Feature housecleaning searches the TFI top-down beginning with the first feature found (i.e. Feature A in Fig. 8). Once found, it takes this feature (Feature A) and matches it against all other features inserted after it, provided there does not already exist a matching percentage for it in the table. It should be noted that matching percentages do not hold a reflective property. The percentage of match between feature X and Y is not necessarily the same as between Y and X. An obvious example is the case where one of the two is a more complex structure, containing completely the other one.

Temporal Feature Index

Feature/ %	A	B	A-1	C	C-1	A-2	A-2-A	C-2	C-3	C-3-A	C-3-B
A	100										
B	44	100									
A-1	60	23	100								
C	33	12		100							
C-1	28	56		61	100						
A-2	66	23	39	55	33	100					
A-2-A	76	55	29	51	49	70	100				
C-2	45	11		77	38			100			
C-3	65	34	19	67	21	32	61	20	100		
C-3-A	55	54	17	50	45	27		45	56	100	
C-3-B	53	37	5	53	26	7		40	57	42	100

Fig. 8 – Example of a Temporal Feature Index.

If the matching percentage in the above mentioned comparison of feature A is in the (80%-100%) range to any of the subsequently added features, feature A will want to remove itself from the feature library. If the matched feature is a parent level feature, the links for feature A get passed to this “same” parent feature and then deletes itself from the library. If the matched feature is not a parent feature, then the matching percentages for all the nodal features above this match beginning at the primary parent level are tested, moving up the tree. If all pass the 50% to 79% similarity test, the feature is an unnecessary duplicate and therefore passes its links to this “same” feature and removes itself from the feature library. If the parent/child feature had sub-children of its own, they would momentarily be left “parentless” but are tested in turn similarly to their shifting parent as they resided after it in the TFI.

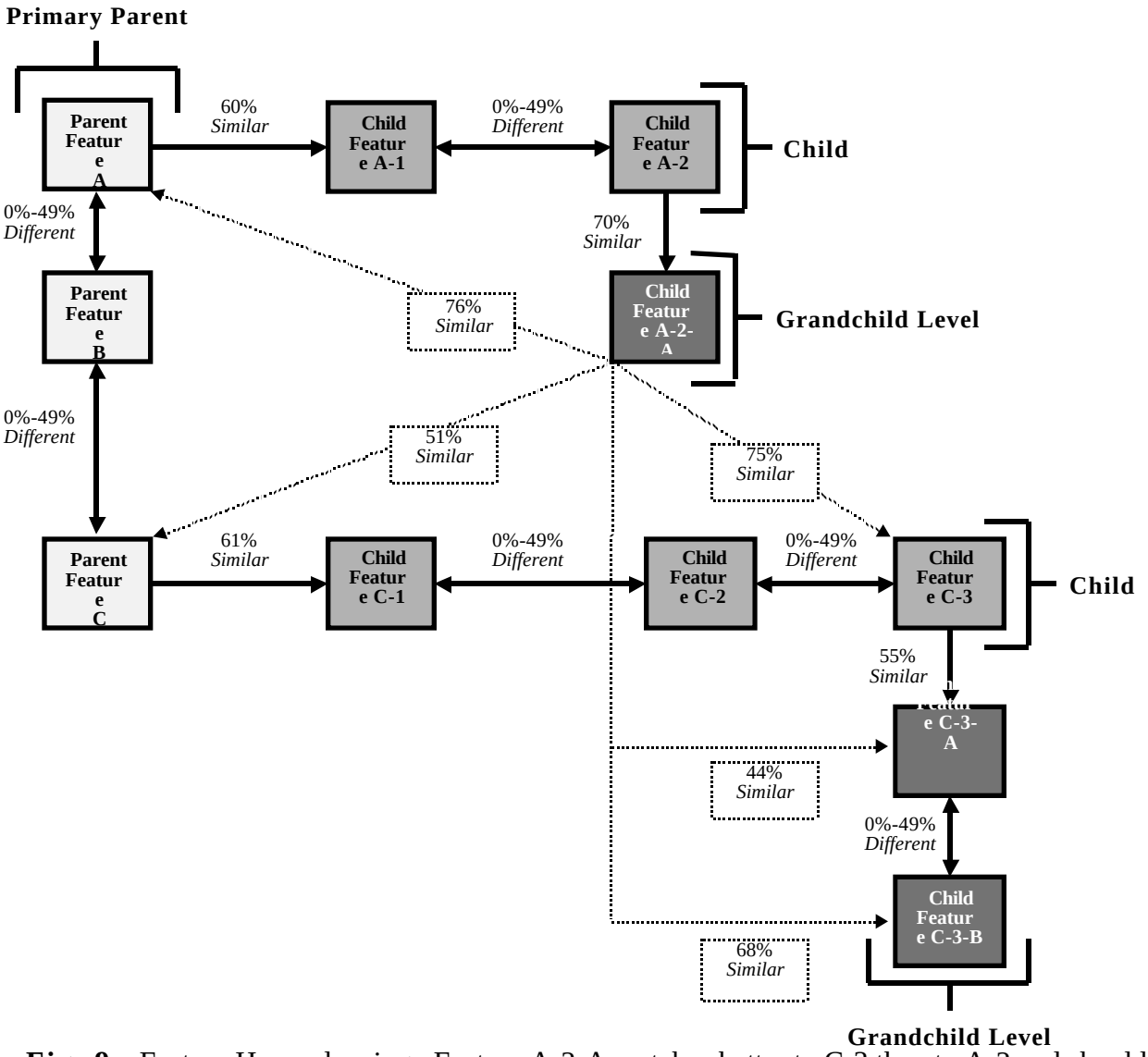

Fig. 9 - Feature Housecleaning. Feature A-2-A matches better to C-3 than to A-2 and should perhaps be a descendent of this "parent". However, because C-3 has a child (C-3-B) that matches above 50% to A-2-A but less than it's current parental match of 70%, it doesn't shift its position within the tree as this would make the feature library inconsistent. Had A-2-A matched above 70% to C-3-B, it would have shifted position with the tree hierarchy.

If the initial matching percentage is between 50% and 79% to any of the subsequently added features and at the same time better than the current matching percentage to its immediate parent, the insertion testing criterion is applied to all nodal features beginning with the primary parent of this "better" matched feature. If all insertion tests are satisfied down to the level of the better matched feature, the child feature shifts its position in the tree hierarchy under this new parent -

providing it is less than 50% similar to any existing children already occupying this level. This eliminates the possibility of a shifting feature ending up in a position within the tree "worse off" than where it was before (Fig. 9). It also ensures that the internal consistency of the library is maintained, i.e. all features reside under their best matched parent, all things considered (i.e. satisfying the insertion testing criterion). If a feature shifts position within the tree, its position within the TFI is also affected by shifting to the bottom of the table. All features previously below it in the table move up one notch in the index. Similar to the case mentioned above where unnecessary duplicate features are removed from the library, if the shifting child feature had sub-children of its own, they would momentarily be left "parentless". However, these children are tested in turn similar to their shifting parent as they resided after it in the TFI. It can be envisioned that once tested, these temporarily "parentless" children could conceivably find themselves shifting back under their previous parent in the tree. Feature housecleaning terminates once it has completed one pass through the TFI table.

6. Experiments and Comments

A prototype system of **I.Q.**, the image query environment described in this paper has been implemented in a Windows workstation. Search and retrieval times for the feature library methodology was tested (using a single 180MHz Pentium) against a sample database of 64 features. The features provided links to an image database of around 50 files. These features comprised a grouping of both manually sketched and extracted objects from existing imagery. Typical shapes were of both generic (e.g. circular, representing cooling towers, and rectangular, representing buildings) and unique nature (e.g. shapes like outlines of airplanes and other visible image objects). As a general rule, generic shapes have links to numerous locations, often multiple locations within a single image. Unique shapes have fewer links, however these links tend to be spatially localized (e.g. numerous airplanes appear in the airport area, but none in other areas).

Testing the matching algorithm for one feature against a complete but unstructured feature library, i.e. one that has not been organized into a hierarchical tree structure like the one described in sections 4 and 5 of this paper, resulted in search times averaging around 2'20" per database. It takes approximately the same time to match a single feature against a single image block of 512x512 pixels. Substantial savings in search times therefore is realized through matching query sketches to linked features rather than to the original images, which was to be expected.

The next step was to organize the 64 features into their proper tree hierarchy according to the rules outlined previously, using the (80%-100%) range for same, (50%-79%) for similar, and (<50%) for different. The first pass through the list of features resulted in a tree that was 38 features wide and up to 3 deep (grandchild level) with 11 features being removed (deleted) as unnecessary duplicates. After applying feature housecleaning, the tree reduced to a structure of 31 features wide and 3 deep, with one additional feature being removed as an unnecessary duplicate. The search times therefore were cut in half as the minimum number of features to test against was reduced from 64 to 31. Additional testing into the tree, if required, did not take significant amounts of time as it was only three features deep, with typical search times averaging around 2" per feature.

These early results support the notion that substantial amounts of search time can be saved if features are organized into a tree hierarchy structure where features with similar shape characteristics are grouped together. Furthermore, these results support the use of our approach for on-line image database search and retrieval. It has to be mentioned that the size of the feature library increases at a much slower pace than the increase of the image library. The feature library eventually reaches a critical mass after which additions of new features are rare, since objects within new imagery are already sufficiently represented in the feature library (there already exist similar templates within it). This is particularly interesting for topographic applications where

often we have increasing numbers of image files depicting the same geographic regions (and the same objects within this region).

Our future work plans include the extension of our experimental feature library and image database to include numerous elements. Among these elements, we plan to include scanned topographic maps, which are actually highly suitable for our system, as object outlines are very prominent within them. To extend the queries in configurations of objects, we plan to employ the well-known concept of 9-intersection, describing the major topological relations between areal, linear, and point features [4]. According to this model, the topological relationships between two objects is expressed by a 3x3 matrix whose elements express whether the mutual relationships between the interior, exterior and outlines of two features are empty or non-empty sets. In order to consider topological relations between objects, we break the query for a spatial scene into two tasks, one where individual features are matched against the feature library, and another where the coordinates of these features are examined to decide whether they fulfill the required topological relations.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation through CAREER grant number IRI-9702233 and through grant number SBR-8810917.

REFERENCES

[1]Agouris P. & T. Schenk. Automated Aerotriangulation Using Multiple Image Multipoint Matching. *Photogrammetric Engineering & Remote Sensing,* Vol. 62, No. 6, pp. 703-710, 1996.

[2] Bruns H. & M. Egenhofer *"Similarity of Spatial Scenes"*, Spatial Data Handling '96, pp. 4A31-42, 1996, Delft, Netherlands.

[3]Cohen S.D. & L.J. Guibas. Shape-Based Indexing and Retrieval; Some First Steps. in *Proceedings 1996 ARPA Image Understanding Workshop*, Vol. 2, pp. 1209-1212, 1996.

[4] Cox I.J., J. Ghosn, M.L. Miller, T. Papathomas & P. Yianilos. Hidden Annotation in Content Based Image Retrieval. *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 76-81, 1997.

[5] Egenhofer M. & R. Franzosa. On the Equivalence of Topological Relations. *Int. Journal of Geographical Information Systems*, Vol. 9, No. 2, pp. 133-152, 1995.

[6] Flickner M., H. Sawney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkaani, J. Hafner, D. Lee, D. Petkovic, D. Steele, & P. Yanker. Query by Image and Video Content: The QBIC System. *Computer*, Sept. 1995, pp. 23-32, 1995.

[7] Gruen A & P. Agouris & H. Li. Linear Feature Extraction with Dynamic Programming and Globally Enforced Least Squares Matching. in *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, (A. Gruen, O. Kuebler & P. Agouris eds.), pp. 83-94, Birkhaeuser, 1995.

[8] Gudivada V. & V. Raghavan. Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity. *ACM Transactions on Information Systems*, Vol. 13, No. 2, pp. 115-144, 1995.

[9] Gupta A. Visual Information Retrieval: A Virage Perspective. TR95-01, Virage, Inc., San Mateo, CA. 1995.

[10] Gupta A., S. Santini, & R. Jain. In Search of Information in Visual Media, *Communications of the ACM*, Vol. 40, No. 12, pp. 34-42, 1998.

[11] Jain R. *NSF Workshop on Visual Information Management Systems*. Redwood, CA, 1992.

[12] Maas H.-G., A. Stefanidis & A. Gruen. Feature Tracking in 3-D Fluid Tomography Sequences. in *ICIP-94*, Vol. I, pp. 530-534, 1994.

[13] Ogle V.E. & M. Stonebraker. Chabot: Retrieval from a Relational Database of Images. *Computer*, pp. 23-32, Sept. 1995.

[14] Pickard R.W. & T.P. Minka. Vision Texture for Annotation. *Multimedia Systems*, 13(1) pp. 3-14, 1995.

[15] Sclaroff S., L. Taycher & M. LaCascia. ImageRover: A Content-Based Image Crowser for thr World Wide Web. *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries,* pp. 2-9, 1997.

[16] Smith J.R. & S.-F. Chang. Searching for Images and Video on the World Wide Web. *Multimedia Systems,* Vol3, No. 1, pp. 3-14, 1995.