1998-7

Digital Image Retrieval using Shape-based Queries

James Carswell
Dublin Institute of Technology, jcarswell@dit.ie

Recommended Citation

Carswell, J. (1998) Digital image retrieval using shape-based queries. *Spatial Data Handling (SDH'98)*, pp. 613-625; Vancouver, Canada; 11-15 July.

1998-07-01

Digital image retrieval using shape-based queries

James D. Carswell
Dublin Institute of Technology, jcarswell@dit.ie

DIGITAL IMAGE RETRIEVAL USING SHAPE BASED QUERIES

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
tel: 207-5812180 fax: 207-5812206
{peggy, tony, carswell}@spatial.maine.edu

Abstract

In this paper we present the development of a spatial data management system utilizing sketch-based queries for the content-based retrieval of digital images from topographic databases. We discuss our overall strategy and associated algorithmic and implementational aspects, and present the 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 strategy is scale-independent. It is inspired by least-squares matching (lsm), and represents an extension of lsm to function with a variety of raster representations. 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.

Keywords: Image Retrieval, Spatial Information Management Systems, Shape Queries, Digital Image Matching

1. Introduction

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]. 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), and emphasize on system and database design (section 3), and digital image analysis issues related to matching sketches to images for querying (section 4). It should be mentioned that while our research originates from topographic applications, the developed methodology can be applied to any type of imagery.

2. Strategy

A description of the envisioned 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]. 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 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.

In this sense, a *match* is an image which satisfies the given query parameters. Database searching is performed progressively. 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*). The reason for this two-stage design are: metadata searches are computationally inexpensive, fast, and therefore optimal for thinning large search spaces. 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. Metadata information is further supporting shape-based queries, as it indirectly assigns some semantic connotations to the sought features. For example, a specific shape has different meaning when found in an aerial image of scale 1:5,000, than when found in a satellite image with 5-meter ground resolution.

---

[1] Metadata of interest includes various forms of information (e.g. sensor characteristics, date of capture, resolution) describing/enhancing the content and/or properties of common data files (e.g. digital images, DEMs, maps in digital format).

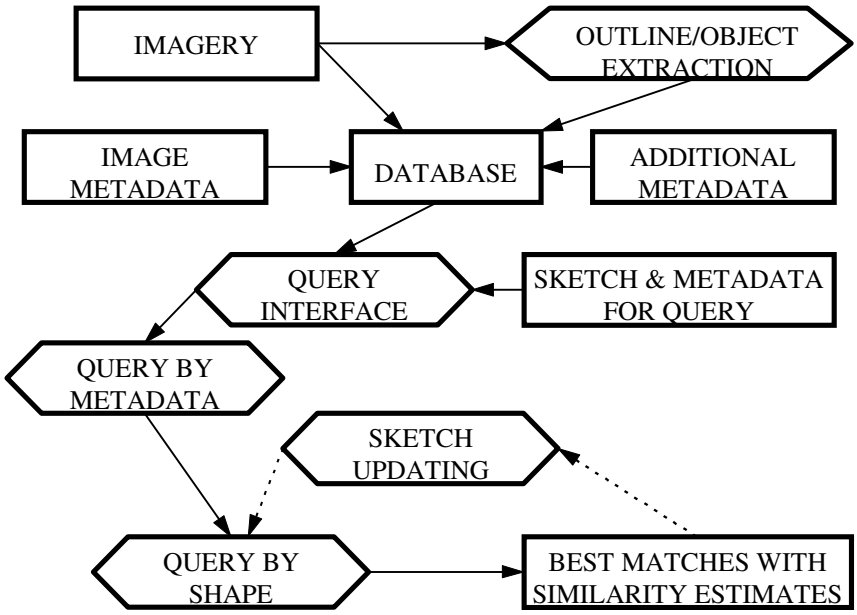

Fig. 1: Operation environment for *J.Q*.

The need for sketch-based queries within topographic databases is related to a simple fact: general color and texture properties are typically inadequate to differentiate aerial or satellite images. The variations caused by different depicted areas on histogram and other relevant global properties are only somewhat noticeable, especially when comparing images of rural to urban and coastal areas. But they are not sufficiently perceptible to be exploited in query processes within the context of topographic applications. Whereas the term "query-by-content" tends to be used for a wide variety of applications, the nature of our problem forces us to use it quite literally: perform searches in image databases according to objects depicted in them.

3. System Design

Outline and object extraction from digital images is a computationally expensive operation, and it is therefore performed not during the queries, but rather off-line when new images are introduced in the database. This results in a system architecture shown in Fig. 2, where dashed lines indicate off-line operations. *J.Q*. is reached by Java servers through the Web, and accesses a library of features and metadata to retrieve the proper images.

The database design is shown in Fig. 3, and comprises the following elements:

• 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).

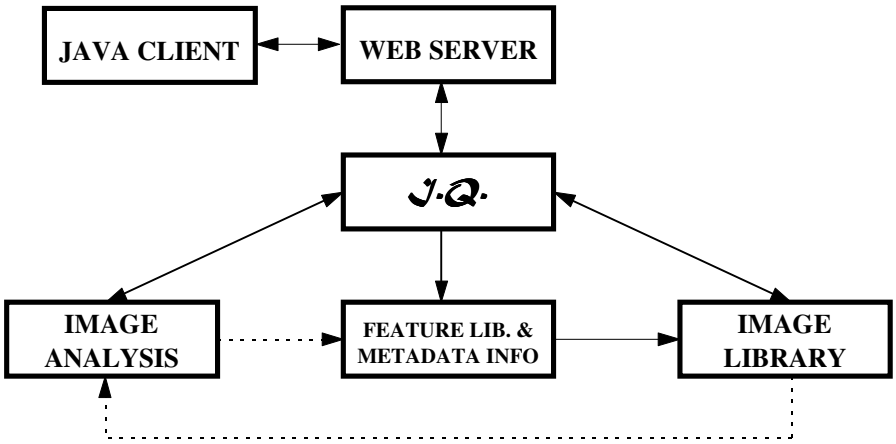

Fig. 2: System architecture

•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 order for the query to be efficient, this library needs to be optimal. The optimality criteria are two: the members of the library should be *exhaustive* (thus being able to describe all possible input features), and the members of the library should be *independent* (avoiding unnecessary duplications). The two properties, when satisfied, are equivalent to an ideal library which is approaching a base spanning the space of shapes.

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. Under this design, the *on-line* part of the query is performed in this manner:

- the user provides as input a set of metadata values and a sketch; the sketch depicts the object that we wish to retrieve, and the metadata values describe the acceptable characteristics of the images in which the object may appear,
- using the input values we identify an acceptable subset of the metadata library; this provides links to features within the feature library and thus defines a feature subset,
- the input sketch is matched to the feature subset (instead of the complete feature library) using our on-line matching tool,
- acceptable matches give links to specific images (and locations within them) where objects similar to our input sketch do appear, and
- this information is returned to the user who then has the option to edit his query[2].

[2] If a semantic library is used, the above mentioned results will pass through another check (whereby the semantic properties of the detected object will be examined), and the query results would be provided after this added step.

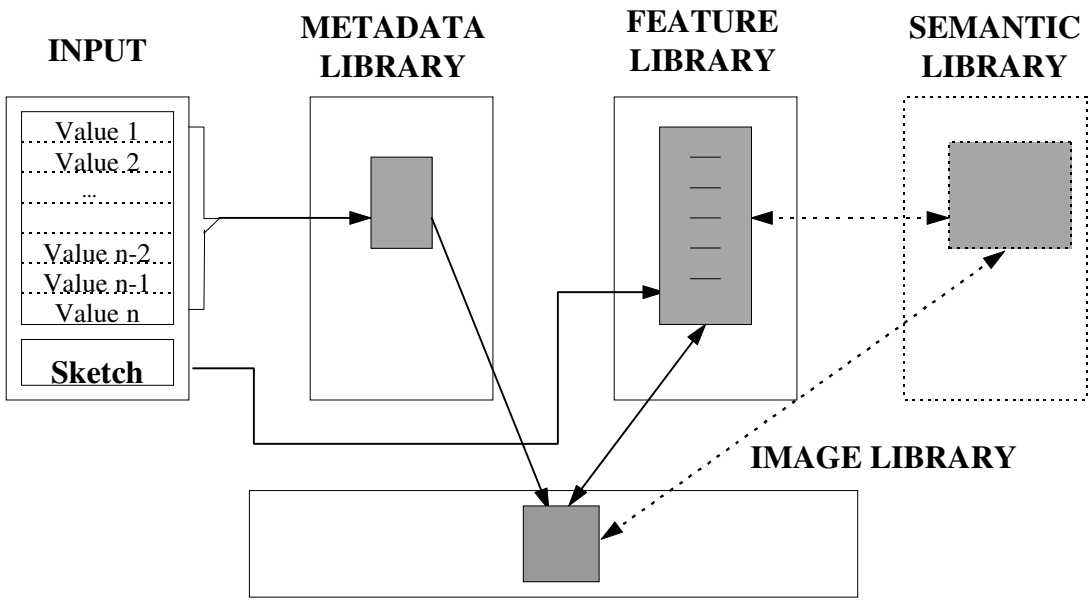

Fig. 3: Database design

In order to support the above sequence, the following sequence has to be performed *off-line* every time an image is introduced in the database:

- the user provides the appropriate metadata information for this image,
- the metadata library is updated to add the new entry,
- objects/features are extracted from the input image using digital image analysis tools,
- these new features are compared to the existing complete feature library using our off-line matching tool, and library entries are updated accordingly to include links to the new image, and
- the links between metadata and feature libraries are updated to connect the metadata values of the new image to the features detected in it.

To extend the queries in configurations of objects, we employ the well-known concept of 9-intersection, describing the major topological relations between areal, linear, and point [4]. Thus, we break the query for a spatial scene into two tasks: identifying where individual objects similar to the input exist in the database, and then identifying the combinations of these locations that satisfy the given topological relationship. For example, when attempting to retrieve raster files which depict an hexagonal building and a cross-like structure separated by an L shaped object, we identify the locations in the database where each of the three objects exists. Subsequently we analyze this information (which now comprises filenames and locations within these files) to come up with the combinations which satisfy the desired configuration. This formulation of scene queries (as combination of a space query and a topological check) permits easy and fast query edits. When attempting to edit a query by reconfiguring a given set of objects, we do so by re-examining their topological relations only, an operation simpler than searching the shape library.

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 declare what kind of features appear at a specific area of the geographic space. When querying the database, we use the metadata information to narrow the area of interest, and then we perform the shape query against the shapes that we know (from the off-line process) to exist in our region of interest. In essence, our design reverses the traditional processes by which we identify objects in the geographic space.

It should be mentioned here that for implementation purposes, one can initially use as library any readily available set of shapes, for example a library like IPE, the Interactive Picture Environment [17], which provides nearly two hundred illustrations, augmented to incorporate certain standard cartographic figures. The shape library can be augmented every time a new shape query does not find an adequate match in the library. In this case the query pattern becomes a member of the library, and is compared off-line against the database to acquire its link information (images and positions within them where a pattern similar to it appears).

4. Matching

Our matching algorithms are influenced by the concept of least squares matching (lsm), modified to function during both the on-line and off-line stage of our prototype. Least squares matching 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].

4.1 Off-line Matching

The off-line matching process begins when a new image is entered into the image database. First the user inputs the relevant metadata (Fig. 4) associated with the new image and then **J.Q**. extracts the edges/objects from the image by applying a generic edge enhancing/thresholding filter to produce a final edge file consisting of only black (gray level = 0) and white (gray level = 255) pixels. This edge file is linked to the image with an identical filename but with different extension. Second, **J.Q**. begins to match all the features from the feature library to the edge file and records in the feature library which features found a match and the matching location within the image.

It does this by first determining where in each of the library features is the center of mass. This central pixel will be used as the feature origin for translating, rotation and scaling the library

feature during the matching process. This method proves more useful (than center of minimum bounding rectangle) under circumstances where multiple groupings of features are combined to make a single feature template to be matched to the image.

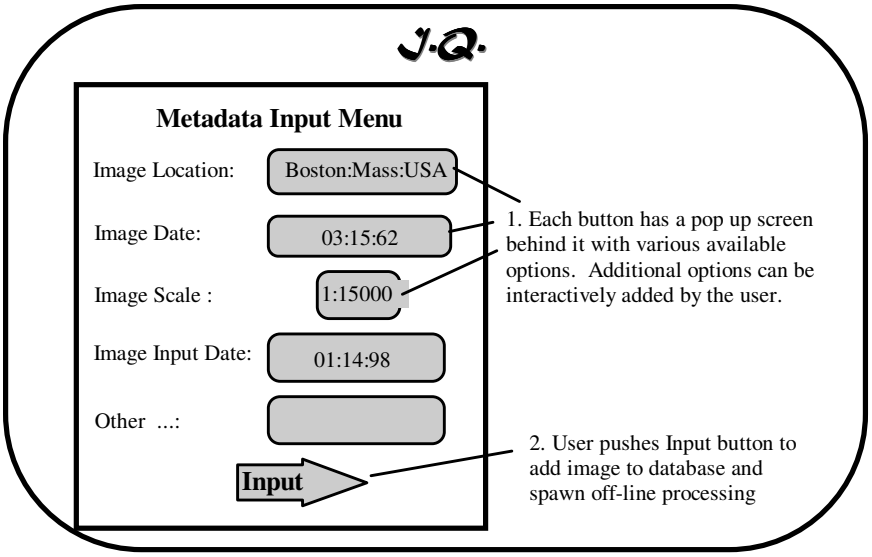

Fig. 4: Metadata Input Menu

Second, the image edge file is subdivided into thirds both in the x (horizontal) direction and the y (vertical) direction thereby creating a 9x9 matrix of equal sized subregions. (Fig. 5) Each of these 9x9 subregions become the boundary for the library feature to translate, rotate and scale inside of while trying to match. These boundaries are set up so the feature doesn't waste time searching around in an area of the image which may have no relevant data. Also, when the feature is rotating and scaling, it will need more space in order to avoid breaking the original library template boundary. By subdividing the image into these regions, the feature is restricted from translating uncontrolled over the image. Its movements are confined to the subregion and each subregion is checked individually for matches.

When the library feature is searching for a direction and distance to move to, it will only do so for a distance of half the dimension of the subregion. If it does not find a suitable match within this distance, then that pixel votes to move the entire feature to a new position within the subregion. The new position will be half the dimension of the subregion in the x direction, where the searching then repeats. Once the feature encounters the boundary of one of the subregions, it drops down half the subregion dimension in the y direction and repeats its movement along the x direction until it either finds some features in the image to match to or again hits the boundary of the subregion. Once the entire subregion has been searched in this way, either successfully or unsuccessfully, the feature will then jump to the beginning of the next subregion and repeat the process.

At the completion of checking all the subregions, the position of the best match from all the regions is identified as the best match for the image. To avoid missing a match that may occur at

the borders of two subregions, additional subregions are added to overlap this area thereby giving 25 separate regions in each image to be searched (Fig. 6). This number of subregions was decided upon assuming the feature being searched for is smaller than $1/9^{th}$ of the total image size. If the feature being searched for is larger than this then the image can be subdivided into quarters. Decisions on how to subdivide the image can also include the scene complexity with a more complex scene being subdivided more to speed up the searching process.

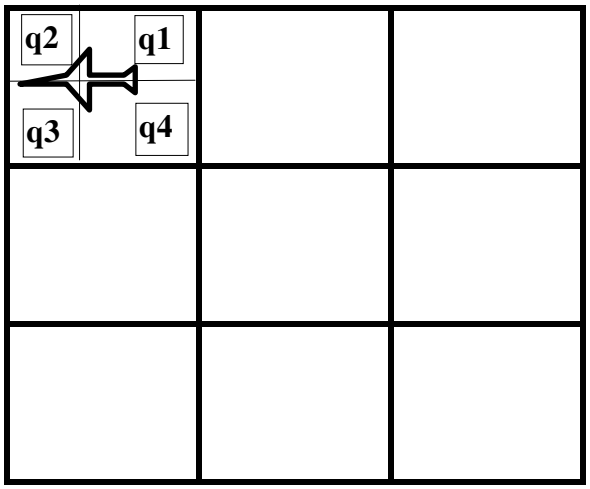

Fig. 5: The 9 Subregions of an Image

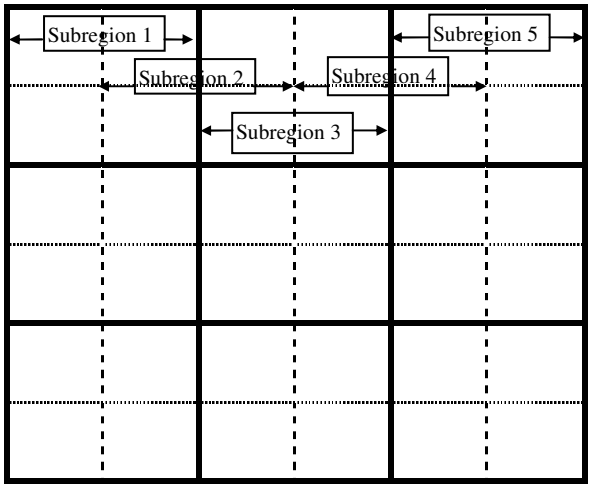

Fig. 6: 25 Overlapping Subregions

To determine which direction and how far to move the feature while matching within a subregion, the subregion is divided into quadrants centered on the central pixel of the template feature. (Fig. 5) Each quadrant is matched separately to the image and the sum of the "votes" for each pixel of the feature in the quadrant is recorded as to whether it votes to stay or move. And if it votes to move, in which direction and how far. Directions are checked in the +/- x and y directions. This means each quadrant votes to move in one of 5 directions; left, right, up, down or stay where it is, and also how far it wants to move. This method also allows for occlusions of up to half of the feature to be detected as the feature can shift its origin pixel right up to the edge of the image.

Each quadrant vote is then compared with the other quadrants and the average direction and distance is taken as the template features next position within the subregion. A sample of *some* of the possible quadrant votes and template feature shifts can be found in Table 1. (note: a vote of ✣ means this quadrant's vote could be in any of the four cardinal directions; currently J.Q. handles 40 cases of different combinations of quadrant votes).

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:

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

q1	q2	q3	q4	Shift
→	→	→	✣	→
←	←	✣	←	←
↑	✣	↑	↑	↑
✣	↓	↓	↓	↓
→	→	↑	↓	→
←	↑	↑	→	↑

q1	q2	q3	q4	Shift
→	←	←	→	**scale in x larger**
↑	↑	↓	↓	**scale in y larger**
←	→	→	←	**scale in x smaller**
↓	↓	↑	↑	**scale in y smaller**
←	↓	→	↑	**rotate** ↺
↓	→	↑	←	**rotate** ↻

Table 1: An Abridged Selection of Quadrant Votes and Feature Shifts

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. Independent axis scaling 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 by 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 J.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.

In both cases of scaling, each quadrant remains intact. That is to say, when the feature pixels within the quadrant scale, all the pixels move in the same direction the same amount. So, it can be seen that in the case of scaling the feature larger, there will now be spaces between the features pixels from one quadrant to the next after expansion. This is logical as data cannot be inserted into the empty spaces as this detail did not exist in the original scan. One method to circumvent this lack of data, and an area of further research, may be to use fractal image compression techniques where no matter how large an image is scaled (or zoomed in), there is always new detail shown.

Once the library feature has settled onto a match its accuracy is determined by how many of its pixels continue to vote to move compared to those that vote to stay where they are. Each of the subregions in an image will have its best match position and match percentage recorded and sorted for each image in the database. The best match percentage for each of the images is used to determine which of the images get linked to a particular feature within the feature template library.

4.2 On-line Matching

The on-line matching process begins by sketching a feature to be matched against the feature library (which is in turn linked to the image database). This can be done in two ways. The user can select a feature off of one of the images in the database using the J.Q. interface and edit it, or, digitize from scratch his own feature outline. The J.Q. interface allows the user to turn on and off feature pixels inside the sketch-edit area. (Fig. 7) When the user clicks on a white (255) pixel it will turn black (0) and vice versa.

If the user selects a feature from an existing image as his template feature, this object will have a nominal scale associated with it based on its scan resolution for satellite imagery and scan resolution combined with focal length and flying height for aerial images. The user can also provide his own scale to the template feature if he wishes. In this way, matches will be prioritized such that the less the template feature has to scale in order to fit the images, the closer the feature is to its true match.

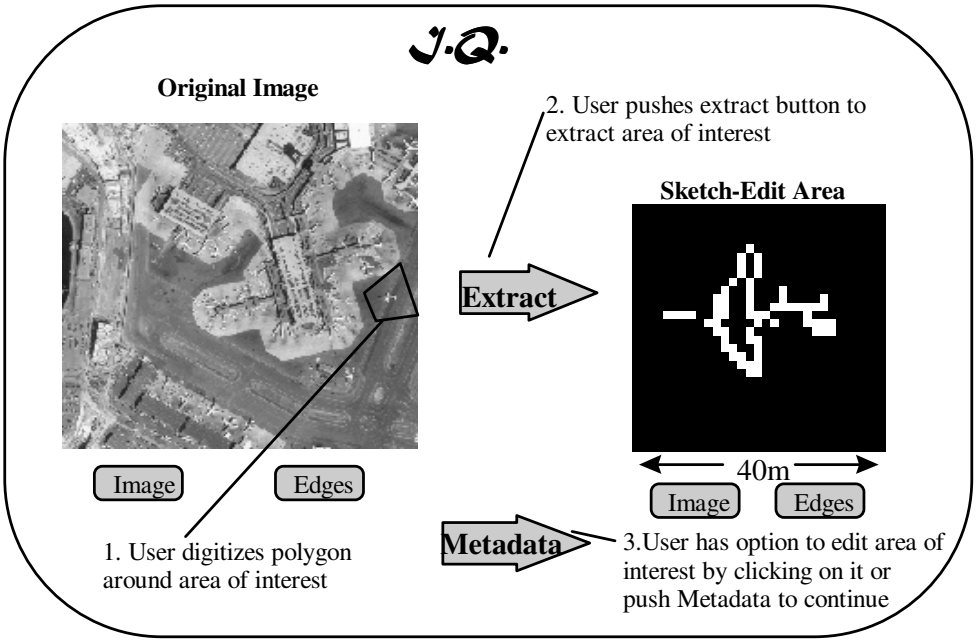

Fig. 7: Extracting Area of Interest

Once the template feature has been extracted and/or digitized, the matching can begin. The procedure begins by determining where in the template feature is its center of mass. This is determined similar to that of the off-line matching process.

Because the matching now is being done against the feature library and not the image database, no subdividing of the individual feature templates within the library is needed thus saving immense amounts of search processing time. Instead, the matching assumes only one

subregion within the image and performs all its translating, rotating and scaling operations similar to the off-line matching process. The only difference being in the scaling where the user can use global scaling if he has only selected an existing feature from an image or independent scaling if he has sketched or edited his own template feature from scratch.

Once the template feature has settled onto a match its accuracy is determined by how many of its pixels continue to vote to move compared to those that vote to stay where they are. Each library feature template will have its best match position and match percentage recorded and sorted. The best match percentage is used to determine its ranking for sorting the returned images based on the original query (Fig. 8).

Currently, J.Q. does not distinguish between how much a feature may have been scaled or rotated when determining its matching percentage. A test on how much a feature was manipulated in order to fit the image could and perhaps should be made and reflected in its ranking. Also, another addition could be to further subdivide each of the quadrants into quadrants themselves. This will facilitate the template feature in warping itself to fit all the objects in the image. This further subdividing of quadrants into quadrants could conceivably go on to many levels of feature manipulation.

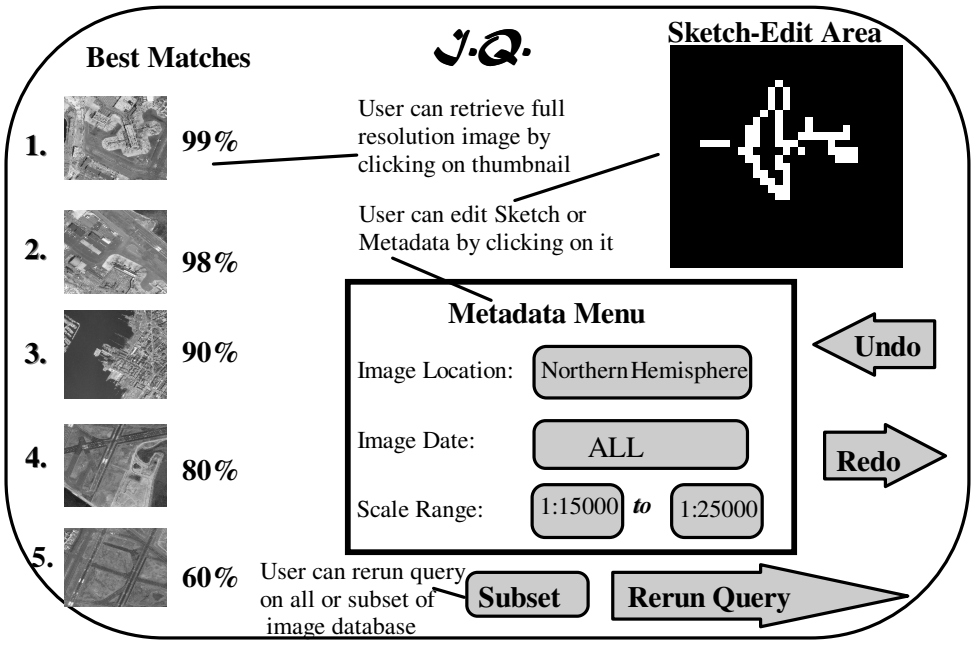

Fig. 8: Query Results

When the results of a query are returned (Fig. 8), the user can re-edit the template feature by clicking on its pixels or by changing some options concerning the metadata. The new query can then be run on a subset of the image database or on the entire database in the case where the metadata has also changed. The subset of the database is a listing of images created during the first search that match to the query. Perhaps 50 images are returned by the first query with the top 5 being displayed on the query results screen. A cutoff matching percentage can be used to determine which images are included in this query subset of images. If the second or subsequent

queries begin to return the sort of images the user is not interested in finding, he can use the undo button in (Fig. 8) to return to previous query parameters. Both the sketch and metadata query parameters are stored for each individual query and are available for recall.

5. Conclusions

A prototype of the system described in this paper has been developed and is presently being implementationally enhanced and optimized. Considering the state-of-the-art in computer vision, the advantages of our method are mainly associated with its precise yet versatile theoretical foundation, and the overall system design. The matching core module used in *J.Q.* is able to properly function for comparing sketches/outlines to digital image windows or to other sketches. This offers the potential of extending this system to function within multimedia spatial databases, allowing queries to be performed simultaneously on images, maps, and GIS vector databases. This very important aspect anticipates the move towards integrated spatial information systems, and is the focus of our future work.

Due to both the system design and the matching tool potential, our method is also scale independent. This is very important for query operations which are inherently multiresolutional, i.e. you have a sketch which depicts a feature or a combination of features at a certain resolution. By employing a multiresolutional strategy, these differences are directly accounted for, and the accuracy and overall performance potential of our technique is optimized.

Furthermore, the method is robust in terms of mathematical and statistical foundation, an advantage which can be best exploited for the objective interpretation of the query operation. Indeed, matching candidates can be arranged and ranked based on a statistical analysis of their similarity to the sought-after template. The potential for such objective interpretation is inherent in the method's set-up and is based on the least squares foundation of the employed matching module, something which is clearly extremely important for our project.

Last, but not least, this method presents a step forward in spatial database queries: we move from queries based on metadata or other global properties of images (e.g. dominant color) to queries using features within images. In doing so, we put more emphasis on the semantic information content of images, which is conceptually a more reliable, and accurate way for image retrieval. Furthermore, the use of user-provided sketches, as well as the ability of a human operator to manipulate the query process (e.g. by revising sketches, or changing metadata) brings the human operator into the process itself, taking advantage of his/her superior cognitive abilities.

Acknowledgments

This work was partially supported by the National Science Foundation through grant number SBR-8810917 for the National Center for Geographic Information and Analysis, and through CAREER grant number IRI-9702233.

References

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

[2] 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.

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

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

[5] 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*, September 1995, pp. 23-32, 1995.

[6] Gruen A. & P. Agouris, "Linear Feature Extraction by Least Squares Template Matching Constrained by Internal Shape Forces" in *International Archives of Photogrammetry & Remote Sensing,* Vol. 30, Part 3/1, pp. 316-323, 1994.

[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 Verlag, 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] Guibas L.J. & C. Tomasi, "Image Retrieval and Robot Vision Research at Stanford" in *Proceedings 1996 ARPA Image Understanding Workshop*, Vol. 1, pp. 101-108, 1996.

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

[11] Lindeberg T., *Scale-Space Theory in Computer Vision,* Kluwer Academic Publishers, Boston, 1994.

[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] Mehrotra R. & J.E. Gary, "Similar-Shape retrieval in Shape Data Management" *Computer*, pp. 57-62, September 1995.

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

[15] Pickard R.W. & T.P. Minka, "Vision Texture for Annotation" *Multimedia Systems,* Vol3, No. 1, pp. 3-14, 1995.

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

[17] Schwartzkopf O. (1995) *Manual of Ipe,* Technical Report, Utrecht University. Holland. (available from ftp://ftp.cs.ruu.nl/pub/X11/Ipe/)

[18] 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.