Image retrieval using Image Context Vectors: first results

Stephen I. Gallant Michael F. Johnston

Belmont Research Inc.
84 Sherman St.
Cambridge, Massachusetts 02138
(617) 868-6878 / (617) 868-2654 Fax
{sg,mfj}@belmont.com

The key question for any image retrieval approach is how to represent the images. We are exploring a new *image context vector* representation that avoids the need for full image understanding. This representation:

- is invariant with respect to translation and scaling of (whole) images,
- is robust with respect to translation, scaling, small rotations, and partial occlusions of objects within images,
- avoids explicit segmentation into objects, and
- allows computation of image-query similarity using only about 300 multiplications and additions.

A context vector is a high (~300) dimensional vector that can represent images, sub-images, or image queries. Image context vectors are an extension of previous work in document retrieval where context vectors were used to represent documents, terms, and queries.

The image is first represented as a collection of pairs of features. Each feature pair is then transformed into a 300-dimensional context vector that encodes the feature pair and its orientation. All the vectors for pairs are added together to form the context vector for the entire image.

Retrieval order is determined by taking dot products of image context vectors with a query context vector, a fast operation.

Results from a first prototype look promising.

Keywords: image retrieval, context vector, *ImageFinder,* image database, image representation, neural network learning, image understanding.

1. MOTIVATION

1.1 The problem of image retrieval

Image databases are rapidly proliferating, and a growing segment of our economy is devoted to producing video imagery for private consumption. In the Government domain, the recent Magellan mission to Venus returned 30,000 1 Mbyte radar images of the planet, more data than that from all previous NASA interplanetary missions combined[1].

Effective and rapid search of image databases is becoming an increasingly acute problem. The most common approach is to hand-label images with text, and then to perform keyword searches on the text alone. However exponential growth in image data requires the ability to *automatically* process images into efficiently searchable representations for later querying. Moreover, manual indexing of text is notoriously inconsistent and error prone, and there is little reason to believe that indexing of pictures produces better results.

Another disadvantage of relying exclusively upon text or database labels for images is that it forces reliance on *textual or database queries*. For example, we may be interested in all images in an image database that are like a particular image or subimage ("image snippet"), such as the "front end of a car." Finding such images would be extremely difficult for current systems, because it is unlikely that most images containing car front ends would be labeled using these particular terms. It would be difficult for a human indexer to foresee that car front ends would be the object of interest for a later query. Even worse, an ultra-careful indexer would also have to include all other car terms (windshield, door, dent, tires, ...). An alternative approach, relying upon a (non-existent) general-purpose semantic network containing sufficient information to know all such part-whole and synonym relationships, seems overly ambitious, although there are efforts in this direction such as CYC at MCC.

There are other image processing problems that need to be addressed by an image retrieval system. These include invariances with respect to *object rearrangement* (translations of objects) within a picture, *scale* (size/distance) of objects and *rotational invariance* in some cases. An image retrieval system should also be relatively insensitive with respect to *occlusion* of objects and *image noise*. Moreover, a general purpose image retrieval approach must be able to handle many *qualitatively different types of images*, for example news photos, satellite imagery, fetal ultrasound images, etc. Thus we seek an approach that is *independent* of any particular set of low-level, medium-level, and high-level image features, and hence general and extensible with respect to the feature set.

1.2 Images *vs.* documents

In this *ImageFinder* project we seek to demonstrate that a new representation for images, Image Context Vectors (ICV's), will allow *fully automatic indexing* and subsequent retrieval of similar images using any combination of images, subimages (snippets), and text shorthands for snippets. (A *text shorthand* is the assigning of one or more sample images to a term such as "car-front-end" in order to permit easy references during queries.)

Several considerations motivate this objective and the approach we will be taking. First, we see similarities between the tasks of document retrieval and image retrieval (Table 1): both are easier versions of more general problems (natural language understanding and image understanding), and both are also susceptible to simpler approaches. In the document retrieval domain, notably little success has been achieved by more general natural

language approaches involving frames, semantic networks, etc.[2]. However, a variety of "surface" approaches have shown good performance by, roughly speaking, treating a document as an unordered set of words and ignoring deeper structure[2-4].

In the image retrieval domain, we hypothesize that we can also obtain better precision/recall performance by using a "surface structure" of images, rather than a deeper, hierarchical analysis of images as employed by current image understanding approaches[5-7].

	Documents	**Images**
More Difficult Problem	natural language understanding	image understanding
...Approach/Representation	parsing / semantic networks	segmentation / object hierarchies
'Easier' Problem	document retrieval	image retrieval
...Surface Representation	unordered sets of terms and phrases	unordered sets of feature pairs and their relative orientations
Basic Element	term or phrase	feature pair and angle
... Represented By	term or phrase Context Vector	pair ICV
Composite Element	document, paragraph, query	image, snippet, object, query
... Represented By	weighted sum of term Context Vectors	weighted sum of pair ICVs

Table 1: Analogies Between Document Processing Image Processing.

Another motivating factor is the success of Context Vector approaches and Latent Semantic Indexing approaches in document retrieval[8-13], and our long-standing interest in generalizing the context vector approach to image data.

The key specific questions that we will attempt to answer in this project are:

1. Can we automatically and efficiently convert a variety of images into image context vector representations?

2. Will image queries detect reasonable sets of similar images (as measured by precision/recall)?

3. Will image snippets detect reasonable sets of full images?

2. IMAGE CONTEXT VECTORS

The key to our approach is the conversion of images to easily searchable image context vector representations, as illustrated in Figure 1.

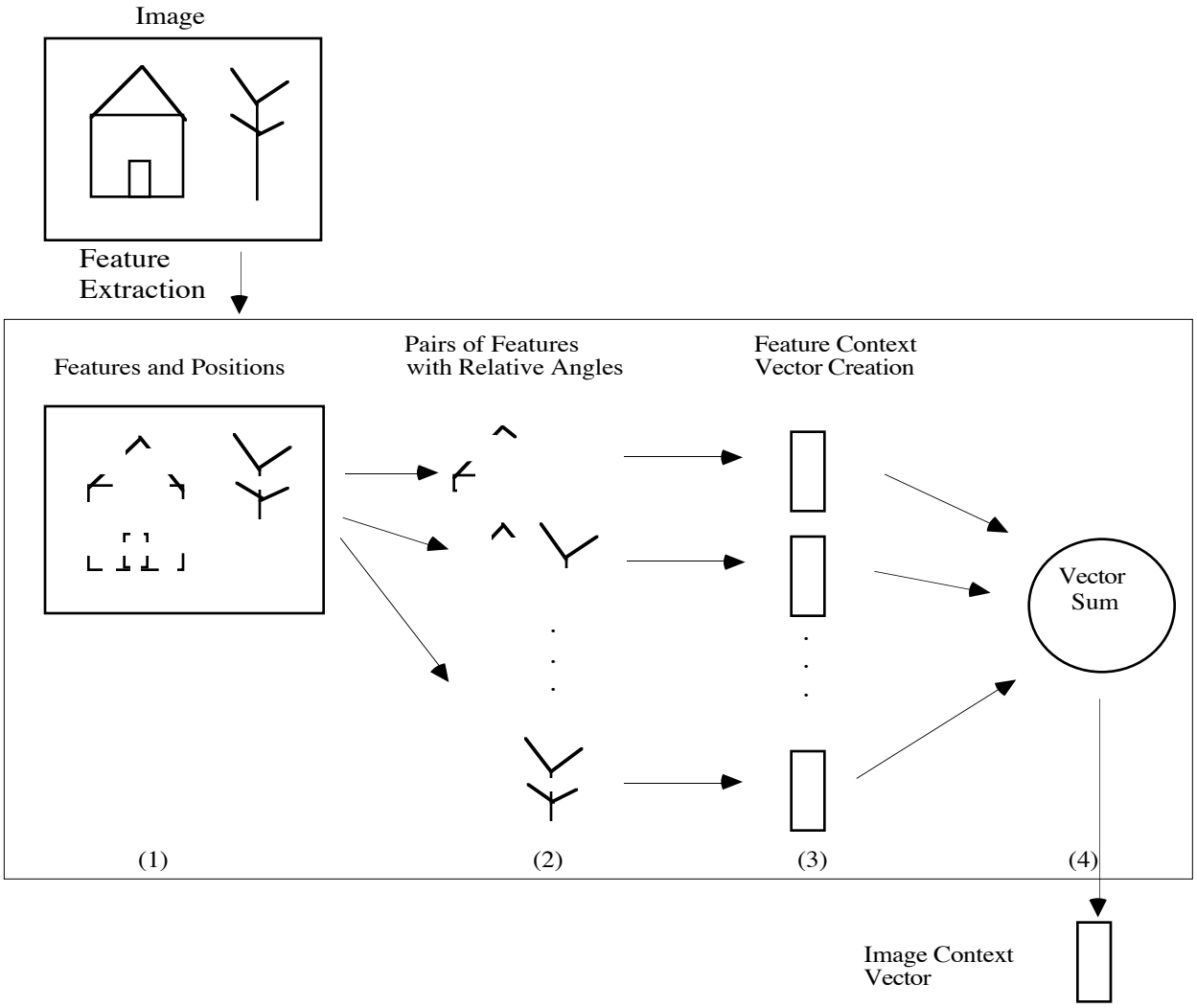


Figure 1. Conversion of Image to Image Context Vector (ICV): (1) extraction of features and positions; (2) pairs of features with relative orientations; (3) creation of context vectors for pairs; (4) (weighted) vector sum of pair context vectors to produce image context vector. For simplicity, only line-based features are illustrated in the figure; this is not a restriction in the approach.

The following sections discuss and motivate the key steps in the figure.

2.1 Creation of image context vectors.

The first processing step is to use image feature detectors to extract between 10 and 300 key features and their corresponding locations in the image. Features might include angles, homogeneous areas (and their centroids), shapes, textures, etc. Here some care must be taken to insure that too many feature points are not generated. For example, we cannot use a line segment detector if it produces too many feature points, without modifying its threshold to be more selective. Note that the approach is general and extensible in that it is not restricted to any particular set of feature detectors, and that higher-level features are permitted.

The second step is to take all *pairs* of feature and their relative orientations. For example, suppose there are two occurrences {A1, A2} of feature A (e.g. trihedral junction) in an image, three occurrences {B1, B2, B3} of feature B (e.g. dark glob), and one appearance{C1} of feature C (e.g. square). Then for each of the 15 pairs

$$<A1, A2>, <A1, B1>, <A1, B2>, ..., <B3, C1>$$

we consider the pairs of feature types and Θ_i, the relative orientations of the two features (expressed as an angle with respect to an arbitrary fixed direction)

$$<A, A, \Theta_1>, <A, B, \Theta_2>, <A, B, \Theta_3>, ..., <B, C, \Theta_{15}>.$$

There may be repeated pairs and angles. Thus we do not track *where* a pair of features occurs in the image, just how many times it occurs in a particular orientation of its elements. Such a surface representation is analogous to document retrieval systems that track which words appear, but without noting location information. See Figure 2 for another example.

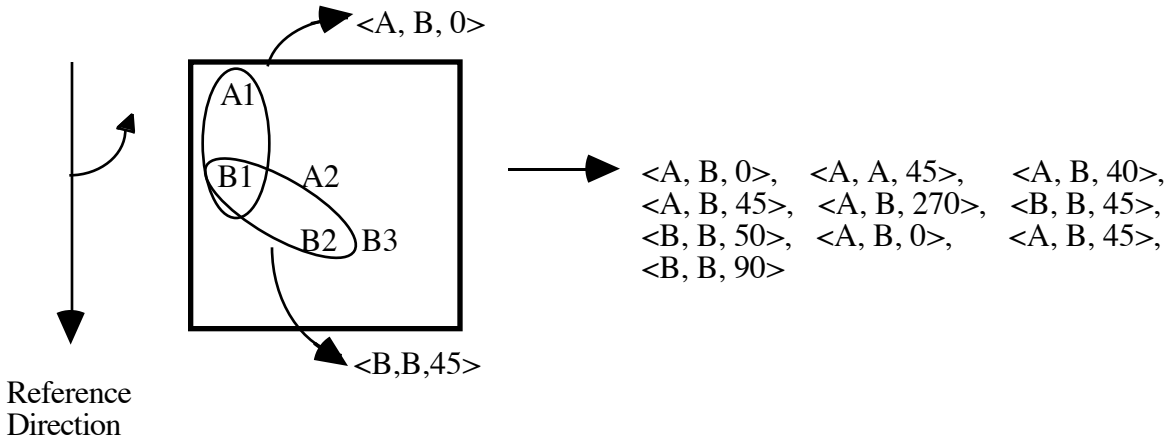
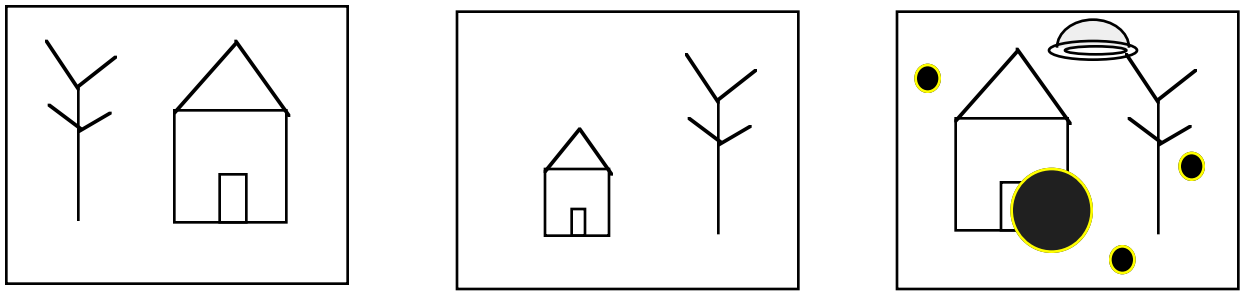


Figure 2. Feature pairs and relative angles (in degrees) for a simple case where only two feature types appear a total of five times in the image.

Note that an object in an image can be *translated* without changing the subset of pairs and relative orientations that correspond to the internals of that object. For example, in Figure 1 if we move the house to the right of the tree, all pairs of features *corresponding just to the house* remain unchanged (Figure 3a). Other pairs, e.g. a house feature and a tree feature, may change. Thus a query consisting of just a house image would have the same feature pairs in common with shifted and unshifted image.



(a) (b) (c)

Figure 3. Invariances/Robustness: (a) object translation; (b) object scaling; (c) occlusion/noise.

Also note that *scaling an object* has no effect on those feature pairs *internal to that object*. For example a smaller house (Figure 3b) would retain the same set of features and angles corresponding just to that house. (Once again, a house feature and tree feature pair may change angle).

Adding noise or occluding objects to an image (Figure 3c) is equivalent to adding or deleting some feature points. If most feature points for an image remain intact, then retrieval should still be possible for that image.

By basing our representation upon feature pairs, we finesse the entire issue of segmentation and avoid having to predict which parts of an object will be relevant to future queries.

So far, our collection of pairs and relative orientations gives a preliminary representation that satisfies translation and scaling invariance, and arguably contains enough information for retrieving images containing particular objects (more on this later). However it still lacks robustness with respect to rotation of an object. Also, the pairs and angles representation is not very amenable to comparisons with a query image that is similarly represented. The next two transformations involving context vectors are designed to solve these problems.

Context vectors were developed by Gallant[11,14,15], and were previously used in ARPA's Tipster Text Processing Initiative to represent words, documents, queries, and word senses[8,9]. Here we will use context vectors to represent feature pairs, images, subimages, and query images.

A context vector is a vector consisting of roughly 300 positive and negative fractional values. For simplicity, we henceforth assume all context vectors are of dimension 300, so that $V = \{V_i \mid i = 1, ..., 300\}$. Typically, context vectors are normalized to have length one. We make special use of *random context vectors* constructed by generating 300 random (fractional) numbers with mean 0 and then normalizing the resulting vector

$$\|V\| = V / (V \bullet V)^{0.5}.$$

It is easy to see that if V and W are two different random vectors, then

$$V \bullet V = 1; \quad V \bullet W \approx 0.$$

Returning to Figure 1, step (3), we can now convert *every pair of features* into a context vector as follows:

1. Prior to encoding any images, for every pair of feature types, <A, B>, choose two random context vectors, V_{AB1} and V_{AB2}. These will remain fixed in the system and will be used to represent all occurrences of pairs for A and B as in the next step.

2. If <Ai, Bj, Θ> is a pair and its angle, compute its pair image context vector by:

$$V_{<Ai, Bj, \Theta>} = \| (\text{Sin } \Theta) \ V_{AB1} + (\text{Cos } \Theta) \ V_{AB2} \|.$$

This transformation reduces (but does not completely eliminate) sensitivity to angles for feature pairs. In other words,

$$V_{<A, B, \Theta>} \approx V_{<A, B, \Theta+\varepsilon>} \text{ for small } \varepsilon.$$

Such robustness is important because, in the world of pixels and feature detectors, we cannot expect relative orientations to match precisely.

At this point we have a collection of pair context vectors, and the last remaining job is to create a conveniently searchable representation. We do this by simply adding up the vectors and then normalizing the vector sum! It is a surprising fact that the sum of normalized, high-dimensional random vectors preserves the ability to recognize individual vector summands. This *Vector Superposition Principle* [16] is easy to demonstrate mathematically:

$$V \bullet \Sigma V_i = \Sigma (V \bullet V_i) \approx \begin{cases} 1 \text{ if } V \in \{V_i\} \\ 0 \text{ otherwise} \end{cases}$$

As with document retrieval, we expect it to be advantageous to use a *weighted* sum of pair ICVs, so that very frequently occurring pairs and angles receive less weight. (We plan to experiment with several popular weighting schemes used in document retrieval[17].) Also, the resulting image context vector will be normalized to avoid favoring, during retrieval, images with many features over images with few features.

2.2 Creation of Query ICV's and Image Retrieval

A query image or snippet is processed just like a full image to form a Query ICV.

Retrieval is now simple and computationally efficient. We merely take the dot product of the Query ICV with each image context vector, and retrieve images in order by largest dot product. (Because image context vectors are normalized, it is easy to show that taking dot products with a query vector gives the same *ordering* of ICV's as computing inverse Euclidean distances between the vectors[8].) To help with intuition, here are some examples of expected behavior:

1. Query image duplicates image in database: Dot product of corresponding ICV's will be 1, and that image will be retrieved first, followed by images with partial matches.

2. Query object or snipped contained in some of the database images: These images should have highest dot products and be ordered by fewest features, i.e. fewest objects extraneous to the query object.

3. Same as #2, but objects are translated and rescaled: Same qualitative performance as #2 expected.

4. Same as #3, but objects rotated: Graceful degradation in recognition ability according to angle of rotation expected. Note that we can try several queries with different orientations to further reduce sensitivity to object orientation. For example, to search aerial photos for those that contain airplane shapes, we could use 8 query images, each containing a plane in a different orientation.

It is possible to use term queries, for example the word "car", by assigning snippets to terms. Thus terms can serve as convenient shorthand references to images. Note here that it is easy to combine several queries into a composite query by taking the (optionally weighted) sum of their corresponding query vectors.

2.3 Complexity Analysis / Scale Up

Time for computing an image context vector scales by the number of feature pairs, which in turn scales by the square of the number of features in an image. Keeping the number of features below 300 should permit very fast image context vector calculations, probably real-time for an optimized system with parallelized feature detection.

Time for retrieval scales linearly for dot products of all image context vectors with the Query vector. (Finding the *largest* of these dot products by sorting is an $n \log n$ operation; somewhat faster with a heap organization.) From our experience with document retrieval, a workstation should be able to perform searches on a million-image library of ICV's in several minutes, depending upon disk access times.

In terms of capacity for storing image information into a 300-dimensional image context vector, experience with document retrieval has shown discrimination ability is still good for sums of over 15,000 vectors (i.e. a 15,000 word document represented by a single 300-dimensional vector). Because document term context vectors are less independent as vectors than pair ICVs, we expect to be able to efficiently search a $1/2 \times 300 = 45{,}000$ pair image vector using Vector Superposition. (This is the source for the limitation of 300 features per image.) Experiments are needed to determine the actual capacity; we may have to increase vector dimensions to 400 or 500 and take a small hit on throughput.

2.4 Dealing With Too Many Features

An important engineering task will be to keep the number of features for an image reasonable. There are several options for dealing with feature extraction software that produces too many features. For example, we can split images into (overlapping) subimages, omit features that show up too many times in an image, or use individual features in an image rather than feature pairs to get a "first order" model rather than the "second order" model we get from the feature pairs.

3. IMPLEMENTATION AND EXPERIMENTS

3.1 Design Choices

In this first prototype implementation, we made several important design choices. We wanted to focus on the basic algorithm and on quickly getting a working prototype to an entire system that permitted query by image, by 'roped' subimage, or by sketch. Therefore we decided to minimize efforts on image processing. We also did all of our programming using an application programming environment developed by Belmont Research called BTL (Belmont Toolkit Language). The Belmont Toolkit contains an extensive class library, a simple-yet-powerful GUI builder, debugger, class browser, and an automatic garbage collector; all of these capabilities proved useful. The capability to develop and debug in interpretive mode and then to extrude C++ code for a 3-5 times speedup was especially helpful.

3.2 Data

Data consisted of sets of line segments for 86 full images grouped into 5 classes: AERIAL_IMAGE, HOUSE_IMAGE, ROAD_IMAGE, BARN_IMAGE, and BUILDING_CORNER_IMAGE. These sets of line segments were derived from 26 original images by using several variations of the Boldt[18] and Burns[19] algorithms developed at the University of Massachusetts. The use of different line extraction algorithms on the same images gave us a way of introducing noise into the image experiments. Prof. Allen Hanson and associates provided the images, and preprocessed them to extract the sets of line segments. This was a great help, permitting us to develop software and run tests with no need for in-house image preprocessing.

Each of the 86 images was divided into 25 overlapping subimages, yielding $25 \times 86 = 2150$ subimages for 2236 total images. These images and corresponding sets of line segments comprised our main corpus.

3.3 System Generation

We used only four features to represent images, namely the four groups of lines:

Shortest lines were eliminated when more than 200 lines were present in full images; otherwise length information was ignored. Thus our representation for images was very minimal, even impoverished.

We formed image context vectors for all 2236 images (consuming several hours on a workstation), and these context vectors were used for retrieval. Context vectors were of length 200; therefore the searchable representation of each image or subimage consisted of exactly 200 numbers, and similarly for queries.

3.4 Tuning Metric

For algorithm development, we constructed software to measure the probability that an image query in class *C*, would be closer to a randomly selected image in class *C* than a randomly selected image in some other class. Only full images were used for this computation. We can exhaustively (and efficiently) compute this probability using every image as a query (details omitted). Total processing time is about 10 seconds for the 86 queries. Figure 4 summarizes the resulting statistics. Note that these classes are not well separated for humans. For example it is easy to confuse the categories of HOUSE_IMAGE, BARN_IMAGE, and BUILDING_CORNER_IMAGE. ROAD_IMAGE queries were especially difficult, because the forest areas generated a multitude of short random lines. We expect performance for such images to improve after texture features are added.

```
Prob 0.669 of correct pair for image class AERIAL_IMAGE.
Prob 0.899 of correct pair for image class HOUSE_IMAGE.
Prob 0.522 of correct pair for image class ROAD_IMAGE.
Prob 0.920 of correct pair for image class BARN_IMAGE.
Prob 0.562 of correct pair for image class
     BUILDING_CORNER_IMAGE.

For all 86 images, prob of correct pair ordering is
     0.678.
```

Figure 4. Initial performance statistics.

3.5 User Interface

The Belmont Toolkit's graphical user interface (GUI) builder allowed us to quickly construct an integrated prototype complete with user interfaces. In this prototype, original images and representations by line segments can be separately displayed. Queries can include:

- entire images
- arbitrary rectangular subimages
- arbitrary hand-entered sketches (straight lines only in this version), using a MacDraw-like sketch pad.

The figure on the following page is a screen shot that illustrates the workings of the system. Here the query is a hand-drawn sketch with 24 lines. The closest image to the query is the house image displayed at bottom center. The second closest is the same image, but preprocessed differently. Most of the closest images are house images; the remainder are building corners. Closeness of images is reflected in the dot products of the query ICV with the image ICV, as given in the lower left of the illustration.

screen shot goes here

Constructing image context vector for the query scales by the square of the number of line segments used. For 200 line segments this takes 10-15 seconds, while for simple sketches it takes only a second or two.

For retrieval, all 2200 images (or, optionally, the 86 full images) are ranked by closeness to the query image context vector. Searching the entire database of images takes only about 3 seconds.

We have run our prototype on UNIX systems for the HP and Sun (Solaris & SunOS). Because the Belmont Toolkit is multi-platform, it should be fairly easy to get versions for the PC and the Macintosh, but we have not yet tried these ports.

4. CURRENT ASSESSMENT AND FUTURE RESEARCH

Overall the image context vector approach looks promising for finding embedded subimages. Our prototype seems to bear out the theoretical invariance and/or robustness with respect to scaling, translation, and noise.

The main shortcoming of the current prototype is its extremely limited feature set. The basic image context vector approach is designed to accommodate additional features, including texture features, and adding such features will be a key task for continued research. We also plan on greatly expanding our text database in the near future. Retrieval speed is not expected to be a problem.

Other key areas for future development are rotation-invariant versions of the basic system. There are at least two promising ways to make the system rotation invariant, but each requires a trade-off (e.g. assuming image sizes for objects of interest are known). Nevertheless, these versions would permit the algorithm to be used for such tasks as analyzing aerial imagery and retrieval of catalog images where rotation invariance is necessary.

4. ACKNOWLEDGMENTS

Partially supported by ARPA contract DAAH01-94-L-R284. Some of this material appeared in Proceedings of the 23rd Applied Imagery Pattern Recognition Workshop sponsored by SPIE, Oct. 12-14, 1994, Washington, DC.

REFERENCES

1. *Neural Information Processing Conference Workshop*; Smith, P., 1993.
2. Salton, G. *Automatic Text Processing*; Addison-Wesley Publishing Co.: Reading, MA, 1989.
3. *The First Text REtrieval Conference*; Harman, D., Ed.; NIST: Washington, DC, 1992.
4. *Proceedings of the Second Text REtrieval Conference*; Harman, D., Ed.; NIST: Washington, DC, 1994.
5. Draper, B.; Brolio, J.; Collins, R.; Hanson, A.; Riseman, E. *IJCV* 1989, *2*, 209-250.
6. Brooks, R. *Artificial Intelligence* 1981, *17*, 285-348.
7. McKeown, D.; Harvey Jr, W. A.; McDermott, J. *IEEE Trans. PAMI* 1985, *7*, 570-585.

8. Gallant, S. I.; Caid, W. R.; Carleton, J.; Hecht-Nielsen, R.; Pu Qing, K.; Sudbeck, d. In Harman, D., Ed., The *First Text REtrieval Conference*; NIST, Washington, DC, 1992; pp 107-11.
9. Gallant, S. I.; Caid, W. R.; et al In Harman, D., Ed., The *Second Text REtrieval Conference*; NIST: Bethesda, MD, 1994; pp 101-104.
10. Caid, W. R.; Dumais, S. T.; Gallant, S. I. *Information Processing & Management* to appear,
11. Gallant, S. I. *United States Patent 5,317,507* Nov. 7, 1990,
12. Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; Harshman, R. *Journal of the American Society for Info. Science* 1990, *41*, 391-407.
13. Dumais, S. In Harman, D., Ed., The *Second Text REtrieval Conference*; NIST: Bethesda, MD, 1993; pp 105-116.
14. Gallant, S. I. *Neural Computation* 1991, *3*, 293-309.
15. Gallant, S. I. In *AAAI-91 Natural Language Text Retrieval Workshop*; Anaheim, CA, 1991.
16. Gallant, S. I. *Neural Network Learning and Expert Systems*; MIT Press: Cambridge, MA, 1993.
17. Salton, G.; Buckley, C. *Information Processing and Management* 1988, *24*, 513-523.
18. Boldt, M., Weiss, R. Token-based extraction of straight lines. University of Massachusetts COINS Technical Report 87-104, October 1987.
19. Burns, JB, Hanson, AR, Riseman, EM. Extracting straight lines. University of Massachusetts COINS Technical Report 84-29, December 1984.

References not used:

18. Rosenfeld, A. “Image Analysis and Computer Vision: 1991,” Center for Automation Research, University of Maryland, 1992.
19. Rosenfeld, A. “Image Analysis and Computer Vision: 1992,” Center for Automation Research, University of Maryland, 1993.
20. Faloutsos, C.; Equitz, W.; Flickner, M.; Niblack, W.; Petkovic, D.; Barber, R. “Efficient and Effective Querying by Image Content,” IBM Almaden Research Division, San Jose, CA, 1993.
21. Arya, M.; Cody, W.; Faloutsos, C.; Richardson, J.; Toga, A. “QBISM: A Prototype 3-D Medical Image Database System,” IBM Almaden Research Center, San Jose, CA, 1993.
22. *NSF Workshop on Visual Information Management Systems*; Jain, R., Ed.; Redwood, CA., 1992.
23. Bach, J. R.; Paul, S.; Jain, R. *IEEE Transactions on Knowledge and Data Engineering* 1993, *5*, 619-628.
24. Pentland, A., Picard, R, et al “The MT/MIT Project on Advanced Image Tools for Telecommunications: An Overview,” MIT Media Lab Perceptual Computing Group, 1993.
25. Turk, M.; Pentland, A. *Journal of Cognitive Neuroscience* 1991, *3*, 71-86.
26. Swain, M. a. B., D. In *DARPA Image Understanding Workshop*; Pittsburgh, PA., 1990; pp 623-630.
27. Plate, T. *IEEE Transactions on Neural Networks* to appear,