Image Retrieval With Embedded Region Relationships

Sharat Chandran
sharat@iitb.ac.in

Naga Kiran
naga@cs.unc.edu

Indian Institute of Technology
Bombay, India, 400076

ABSTRACT

Image retrieval based on content from digital libraries, multimedia databases, the Internet, and other sources has been an important problem addressed by several researchers. In this regard, one cannot overestimate the use of appropriate features such as color, texture, and shape. It has also become increasingly evident that the decomposition of images into regions is critical for useful results.

In this paper we further study region-based image retrieval. We argue that a relationship between regions (such as a tiger amongst yellowish-green grass, or a plane against the blue sky with mountains in the background) is also important. Our local segmentation algorithm is used to detect regions a priori. Further, while searching for a match for an 'object' in the database, we allow for probabilistic 'multiple matches,' which are later pruned based on global consistent information. We provide a simple, fast algorithm implemented as an Internet thin client connecting to a web server. Experimental results indicate that our method has high precision, is robust towards translation, rotation, and scale changes, can handle partial occlusion, as well as many popular image transformations (such as shear and blur) much the way humans can.

Keywords

Multimedia, subgraph, region, similarity, database.

1. INTRODUCTION

In recent years, there has been a tremendous increase in the number of images on the Internet. As the number of people using the Internet explodes, the need to develop efficient and practical information retrieval techniques in different areas is increasing. Prevalent retrieval techniques using text annotations for images is useful, but insufficient since different people provide different annotations to describe the same image.

Image querying by computer (referred to in the literature as "query by content"or "query by example") has evolved ([8]) from the classic object recognition problem in computer vision. In robot vision, the computer autonomously identifies objects in its environment. Querying images is different and more doable, partly because the users now play an active role in seeking assistance from the computer (or the other way around!). For example, users may be allowed to supply a charcoal sketch, a color painting, or a scanned low resolution image. The problem still appears insurmountable for several non-mutually exclusive reasons as outlined below

- The query image posed by the user is *semantically* different from the set of images. This could be due to several factors. Perhaps the query image is that of person in the database, and he has aged.
- The object of interest in the query is not different from the one in the database, but the *relationship* of the object with other objects has changed. Perhaps it has been occluded by some other object.
- The *quantifiable* aspects of the object in the query image is distorted when compared to the one in the database. Examples of these aspects include color, shading information, and the like.
- The object in the query image is *spatially* varied; it might be rotated, translated, or scaled when compared to the one in the database.

As elucidated in Section 1.2 we study these fundamental issues in this work.

1.1 Previous Work

We first consider the searching capabilities on the Internet as available to the lay person such as Google Image Search. Despite (or because) these search engines use text annotation, results are disappointing.

Therefore, the need to develop a system which uses *image features* as vectors, and appropriate distance metrics has been shown to be very important by the vision research community. The first wave of research incorporated color histogram as features. Image feature vector indexing techniques have been implemented in various systems like QBIC [5], Photobook [9], and WBIIS [12] with varying degrees of success. One problem with the color histogram approach is that histograms do not contain the shape, location, or texture information. Two images having the same color histogram may differ widely in shape and image semantics.

An alternative method is to use wavelets for the image signatures as in [12], [14], and [6] system. Since wavelets capture shape and location to some extent, some problems are eliminated. However, these systems usually consider the lower frequencies in the image. Since texture is known to be in high frequencies [11], these systems show deteriorated performance in some cases.

Significantly, since a single signature is calculated for an image, the system suffers when, for instance, two images contain similar

SAC 2003 Melbourne, Florida USA

objects that are spatially varied. Most of the above systems have trouble in handling rotated, translated, or scaled versions of content.

The more general approach is to reconsider the classic segmentation problem in computer vision, and retrieve similar images based on regions within a query image. Blobworld [2] is a system which does segmentation based on high level features of the image. The user is asked to select a blob on the image and a few other parameters (such as the importance of background) to assign the weightages for each of the feature vectors. Simplicity [13] is another segmentation based system. It uses the "Integrated Region Matching" [7] approach to compute the distance between query image and target image. WindSurf [1] is another wavelet based system which does image segmentation using a k-means clustering algorithm. It uses both the low level and the high level wavelet coefficients, thus using texture in the process of segmentation.

These systems underscore the importance of working with regions in retrieving images similar to a query image. As a double-edged sword, improper segmentation leads the implementation astray in many cases as can be seen by trying out the systems.

1.2 Our Contributions

Our segmentation based image retrieval system gives encouraging results and is based on the following points:

1. **Segmentation:** Unsatisfactory image segmentation distorts expected results. We have found the graph based local segmentation algorithm in [4] useful. Unfortunately the number of segments it outputs (despite using various options in the algorithm) is often too high, and is unsuitable for image retrieval. We have modified the algorithm even while improving the overall running time to represent macro objects (similar to blobs in [2]) by regions.

2. **Graph Abstraction:** By representing regions within images using graphs, we look for more than similar objects in the query image amongst the database of images. While graph based approaches have been used in the past (e.g., [10]), region relationships between one object in the query, and other objects in the query have not been pursued. We also allow one object in the query to be matched to multiple objects within other images in the database. This many-to-one mapping is resolved by a global consistent system using the notion of cliques in a graph. The resulting problem is shown to be NP-complete. Unlike other combinatorial problems, we do *not* need optimal cliques and instead use a fast approximation algorithm, again tailored for the first time (to our knowledge) for image retrieval.

3. **Invariance:** Desirable properties like shift, rotation, and translation invariance is a challenging problem in systems using single signatures for images. Also, retrieval of partially occluded objects becomes a difficult task for all prior systems. Using the notion of maximal common subgraph enables us to obtain encouraging (see Section 4) results. Others systems we have used are unable to the same quality of results.

The rest of this paper is organized as follows. In the next section an overview of our algorithm is given, following which some details are given in Section 3. In Section 4, we provide empirical results based on our implementation to support our approach. Some concluding remarks are made in Section 5

2. THE ALGORITHM

Conceptually, there are four steps in the algorithm.

The first step in our approach is based on the local segmentation in [4]. Each image in the database (henceforth termed as a target image) is segmented (offline) and represented as a graph based on the connectivity of the regions. The initial graph is connected, but as we "find" objects in the query image matching parts of the target image, the graphs may temporarily (in main memory) be decomposed into connected components when a query is made.

Let $G^1(V^1, E^1)$ and $G^2(V^2, E^2)$ be the graphs corresponding to the query image and target image respectively. The second step computes a function ϕ defined by $\phi : V^1 \rightarrow V^2$. The function ϕ maps the labels of the target image to that of the query image. In contrast with many earlier techniques, the function ϕ is a many-to-one function. This approach allows for multiple regions in target image to be considered as a match for a region in query image. The best alternative will be later chosen based on the global connectivities of the neighboring regions in both query and target images. ϕ is computed in our approach based on feature vectors such as color, and shape. (Superior feature vectors will result in better matches.) An example of the mapping is shown in Figure 1.

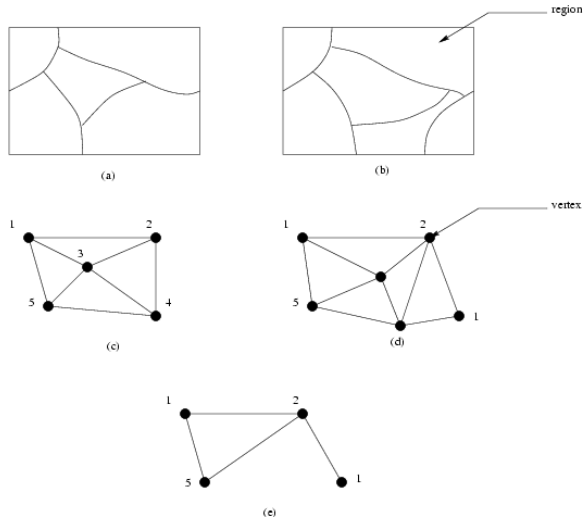

Figure 1: (a) Segmented Image I_1. (b) Segmented image I_2. In (c) and (d), we show the graphs based on connectivity of regions in (a) and (b) respectively. Notice the labeling of nodes in the graphs (i.e., the function ϕ). Observe that some nodes in G_2 are not labeled. (e) Correspondence graph.

In the third step, we compute the Maximal Common Subgraph (MCS) of the two graphs and thereby obtain a measure of similarity between the two images. One novelty of our method is the use of a particular correspondence graph (also shown in Figure 1) discussed in Section 3.1.

In the last step, we compute the distance between matched portions of the query and candidate target images and produce acceptable results. In the interests of scalability for large databases, automatic clustering (based on the number of components and color) is used along with hash table indices. The overall algorithm is shown in Figure 2.

3. SOME DETAILS

3.1 The Correspondence Graph

For each vertex in graph G^1, assign a unique distinct label. For each vertex $v_j^2 \in V^2$ of graph G^2 corresponding to some element

```
RetrieveSimilar(image Query)
1   Assign unique labels to vertices in G^1 (query graph)
2   Initialize each region as not matched
3   For each target image T in the clustered subset of database {
4     G^2 = Graph of T
5     Find minimum distance m for each vertex pair in G^1, G^2
6     For each vertex v_i^2 ∈ G^2, assign closest label v_j^1 ∈ G^1 if
      distance(v_i^2, v_j^1) ≤ αm
7     Compute the correspondence graph G considering only the
      vertices in G^1 and G^2 which are labeled.
8     While (maximum regions in query are matched) {
9       vc = VertexCover(G)
10      Update regions not in vc as matched
11      Update unmatched regions in G^2
12      Recompute the correspondence graph as described in Section 3.1.
      }
13    Calculate distance between query and T
    }
14  Sort and display the results.
15  return.
```

Figure 2: Pseudocode of the overall algorithm.

$v_i^1 \in V^1$, assign the label corresponding to v_i^1 based on image features. Now generate a correspondence graph $G(V, E)$ such that each vertex in G corresponds to (v_i^1, v_j^2) ($v_i^1 \in V^1$ and $v_j^2 \in V^2$, and $label(v_i^1) = label(v_j^2)$). Define an edge between two vertices $((v_i^1, v_p^2), (v_j^1, v_q^2))$ either if there is an edge between both (v_i^1, v_j^1) in G^1 and (v_p^2, v_q^2) in G^2, or if there is no edge between both (v_i^1, v_j^1) in G^1 and (v_p^2, v_q^2) in G^2. (As a degenerate case, if $v_i^1 = v_j^1$, then do not insert the edge in the correspondence graph.) A key lemma (not shown here in the interests of space) is that the maximal clique (MC) of the correspondence graph G corresponds to the MCS between the two images.

3.2 Using Vertex Cover

It is well known that the clique problem is a hard problem, but we have a simple linear time algorithm. The MC of a graph corresponds to the minimal vertex cover of the its complementary graph. An approximation algorithm for a minimum vertex cover for a graph $G = (V, E)$ and having a bound of 2 appears in [3]. We adapt the algorithm (details skipped here) for image retrieval to ensure that the approximate vertex cover has a size at most $|V| - 1$. This would in turn ensure that the approximate clique obtained is at least of size 1.

3.3 Segmentation

The approximation algorithm is fast; however, in order to ensure that the overall algorithm runs fast, we must ensure that the number of segments obtained are meaningful and small.

We modify the fast local image segmentation algorithm described in [4]. The original algorithm (using Kruskal's MST algorithm) produces a significant number of small sized regions. These are negligible for image retrieval and are removed by merging them with larger segments. The important thing is to perform this step without increasing the running time of the original algorithm (details of this step and the faster Prim-based MST algorithm are excluded here.)

Figure 4 and Figure 5 contrast the segmentation of an image (Figure 3) using the original method in [4] and the proposed method.

Figure 3: Input image. Results are best seen in color on a monitor.

Figure 4: Prior method with identical input parameters.

Figure 5: Our method. Observe wings clearly demarcated.

3.4 Distance Metric

The distance between the query and target image is calculated by using the weighted sum of the distances of matched regions. The significance (or weight) of each region matched is proportional to its area. If I_1 and I_2 are the images, the distance is given by $d(I_1, I_2) = \sum_{R^i \in I_1, R^j \in I_2} w(R^i, R^j) D(R^i, R^j)$, where $w(R^i, R^j) = area(R^i) + area(R^j)$, and D is the feature vector distance in an appropriate metric.

4. SAMPLE RESULTS

Using a standard browser, one can query our system by uploading an image from the local filesystem, by selecting randomly generated images from the server, or by directing the server to any image on the Internet. The database consists of about 1000 standard [13] "scenery" images. Sample results (each retrieved in about 2 seconds on a low end workstation) obtained by querying with images already present in the database are shown in Figure 8. All query images appear to the left. The results have been ranked. In the case of the elephant, the last ranked output is "incorrect." In the case of the planes, birds have also been retrieved. In each case (co-

incidentally) seven resulting images have been obtained. We see [1] that the queried image is always retrieved as the top ranked image, and the results are valid based on the shape, color, and size of the image. The graph in Figure 6, based on 50 random attempts by a human observer, vindicates this claim. It shows an average "false positive" error rate of 18%. This number is comparable to a recent system [13], and better than earlier systems (which report about 60% false positives). Objective results are available in Figure 7. Also to note is the graph in Figure 9 based on another 50 random attempts. It shows an average false negative error rate of 23%. Figure 10 is more interesting. Images in the database are corrupted in a variety of ways. Once again, near matched images are ranked and displayed. All query images appear to the left. The results have been ranked. In this situation, the number of output images are *not* identical; the program adaptively decides the number of relevant images to output. In the first row, an image in the database has been filtered through a high pass filter ("sharpened" by 40%). Four images are retrieved and ranked with Roman numbers. In the second row, an image in the database has been "swirled;" again, four images are retrieved, and ranked with alphabets. In the third row, the input has been turned upside down. Nine images have been retrieved and ranked in a zig-zag fashion. In the fifth row, an image in the database has been sheared. Six images have been retrieved. In the seventh row, an image has been corrupted by Gaussian noise. Five images have been retrieved and ranked. In the last row, an image has been annotated with text. Six images have been retrieved and ranked.

In every case, the top ranked retrieved image is the source image which has been corrupted. These results are only representative. Based on ground truth, we have found our system to be quite useful.

[1] Best in color on a computer monitor.

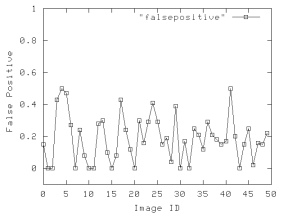

Figure 6: The fraction of inappropriate results based on 50 (out of 1000) random query images. Correctness is judged by a human observer introduced to the system for the first time. (The fraction is obtained only from the set of *retrieved* results; incorrect rejected images from the database do not figure in the computation.) A value of zero indicates that all retrieved results matched the query. The average value is 0.18.

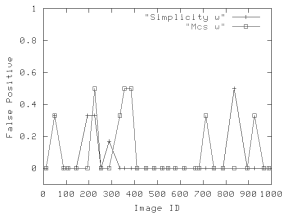

Figure 7: Comparing our system which guarantees, for example, rotation invariance with that in [13] which doesn't. The program and data were obtained (with thanks) from the authors, and false positive values are computed for 29 random images. The average false positive values are 0.057 for [13], and 0.097. For both systems, we rely upon the database being preclassified in chunks of 100 ('correct' if answer(i) $\in [100\lfloor i/100 \rfloor, 100\lceil i/100 \rceil)$. The results can be duplicated by a program. (Humans may disagree on the classification.)

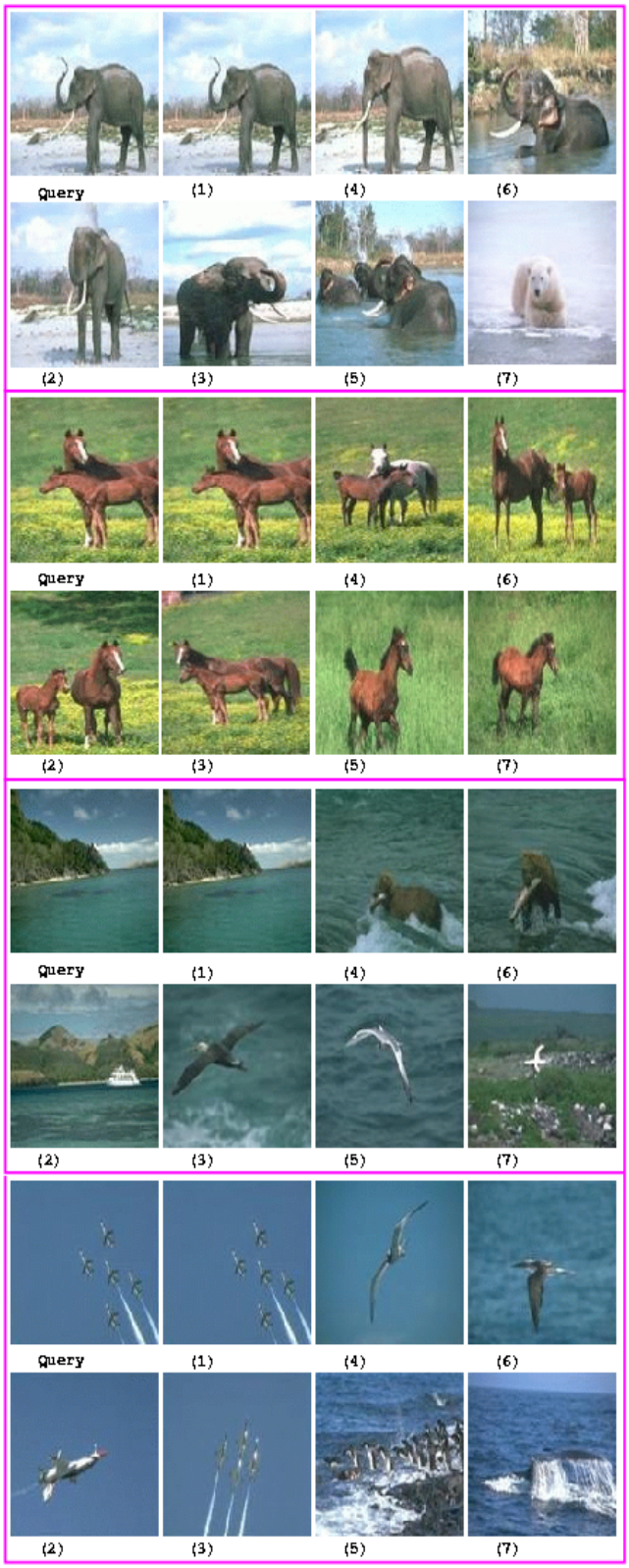

Figure 8: Images retrieved when the input image is already present in the database. In this illustration four queries with ranked (alphabetically or in Roman numerals) results have been displayed.

5. FINAL REMARKS

Recent image retrieval algorithms work with portions of images in order to find similar objects in the database in multiple poses. It is not uncommon, however, to retrieve results that are incorrect (due to improper segmentation, admittedly a hard task). It is also not uncommon to miss results even with correct segmentation because the relationship of the objects to other objects in the image is not given due importance.

In this paper we have addressed these issues using the notion of the maximal common subgraph. We have improved a popular local segmentation to obtain meaningful segmentation for image retrieval, and designed a simple globally consistent algorithm for retrieving similar images. While graph based approaches have been used, our work is novel in exploring region relationships for intelligent retrieval. Experiments show meaningful results are obtained, even when input images are corrupted by noise, image transformations, or partial occlusion.

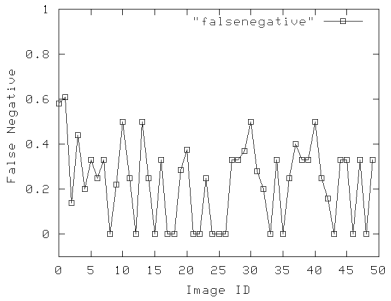

Figure 9: The fraction of correct results missed from the set of retrieved results based on 50 random attempts. Correctness is judged by an human observer who also browsed the database looking for similar images. A value of zero indicates that all images in the database that are similar to the query have been retrieved. The average value is 0.23.

6. REFERENCES

[1] S. Ardizzoni, I. Bartolini, and M. Patella. Windsurf: Region-based image retrieval using wavelets. In *DEXA Workshop*, pages 167–173, 1999.

[2] C. Carson, M. Thomas, S. Belongie, J. Hellerstein, and J. Malik. Blobworld: a system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*, pages 509–516, 1999.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. M.I.T. Press, Cambridge, Massachusetts, U.S.A., 1990.

[4] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *CVPR*, pages 98–104, 1998.

[5] M. Flickner, H. Sawhney, and W. Niblack. Query by image and video content: The qbic system. *IEEE Computer Magazine*, 28:23–32, 1995.

[6] C. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. *Computer Graphics*, 29:277–286, 1995.

[7] J. Li, J. Wang, and G. Wiederhold. IRM: Integrated Region Matching for Image Retrieval. In *Proceedings of the 2000 ACM Multimedia Conference*, pages 147–156, 2000.

[8] M.Das, E. Riseman, and B. Draper. Focus: Searching for multi-colored objects in a diverse image database. In *IEEE CVPR*, pages 756–761, 1997.

[9] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18:233–254, 1996.

[10] E. Petrakis and C. Faloutsos. Similarity searching in medical databases. *IEEE Transactions on Knowledge and Ata Engineering*, pages 435–447, 1997.

[11] M. Ramos, S. Hemami, and M. Tamburro. Psychovisually-based multiresolution image segmentation. In *Proceedings of the IEEE International Conference on Image Proceedings*, pages 66–69, 1997.

[12] J. Wang, G. Wiederhold, O. Firschein, and S. Wei. Wavelet-based image indexing techniques with partial sketch retrieval capability. In *Advances in Digital Libraries*, pages 13–24, 1997.

[13] J. Z. Wang and G. Wiederhold. Simplicity: Semantics-sensitive Integrated Matching for Picture Libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[14] J. Z. Wang, G. Wiederhold, and O. Firschien. System for screening objectionable images using daubechies' wavelets and color histograms. *Computer Communications*, pages 1355–1360, 1998.

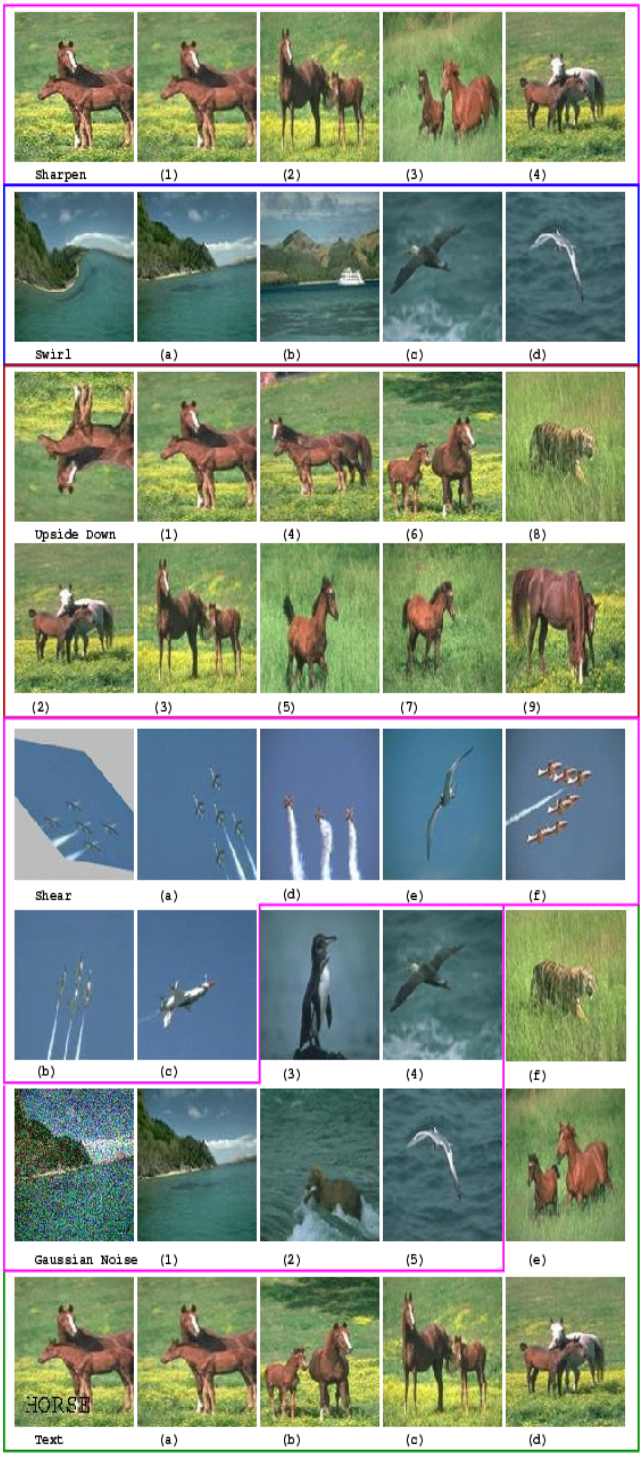

Figure 10: Images retrieved when the input image is *not* present in the database. In this illustration six queries and results (best seen on a color monitor) have been ranked alphabetically or in Roman numerals.