Image Retrieval and Robot Vision Research at Stanford *

Leonidas J. Guibas and Carlo Tomasi
Computer Science Department, Stanford University
Stanford, CA 94305

Abstract

We report on the two areas of our research that are sponsored by ARPA, namely, the retrieval of images from data bases and vision for mobile robots.

In image retrieval, we have developed two demos. The first is based on texture and color information, while the second explores the use of shape information in the presence of occlusions. We plan to merge these two retrieval demos into one once the technologies involved are well understood in isolation. For our next demo, we have stored twenty-thousand images from a stock-photograph collection onto a laser disc recorder. Thumbnails and retrieval indices are stored on a computer hard disk.

For robot vision, we have developed, together with Stanford's robotics group and Nomadic (a local robot manufacturer), a robot observer that uses motion planning and visibility graphs to stalk a moving target in an environment cluttered with obstacles. We have also built a depth-from-focus vision system that allows a Nomadic robot to navigate for hours in a crowded environment. The robot avoids obstacles, both static and moving, and turns away from steps, both up and down, in a very reliable fashion.

1 Introduction

This report covers some of the new activities in computer vision that started with the arrival of one of us (Tomasi) at Stanford in 1994. Among these activities, two projects are being sponsored by ARPA. The first is on the retrieval of images from data bases based on pictorial information. The interest in this area is growing, as testified by more academic research [Jain, 1992; Picard and Minka, 1995], articles in several new publications on multimedia (Multimedia Systems, IEEE Multimedia), special issues in scientific journals (PAMI, edited by Picard, to appear), startup companies (Virage), and the first commercial retrieval system [Faloutsos *et al.*, 1994]. In section 2, we summarize the state of our project in this area.

Our second ARPA-sponsored project concerns vision for robot navigation. This is a more established area of research, but the rewards are no less important in terms of applications, particularly in the fields of manufacturing, handling of hazardous materials, and teleoperation. In section 3, we show two contributions we made to this area last year.

2 Image Retrieval

That solutions to the image retrieval problem are still in their infancy is demonstrated by Time Warner's decision to have their 20-million photograph collection scanned and catalogued by hand, with detailed textual descriptions appended to each image [Bielski, 1995]. If research by us and others in the field of automatic image retrieval is successful, this enormous effort will become unnecessary well before it is completed at the current rate of about 300,000 photographs per year. Browsing through image data bases on the World-Wide Web, analyzing satellite and space exploration images, and accessing medical images are but a few other applications of image retrieval.

We have developed two demos. The first is based on texture and color information, and is being tested on about two-hundred images on various subjects. The second demo explores

*Research on image retrieval is supported through ARPA grant DAAH04-94-G-0284 monitored by the US Army Research Office. Research on robot vision is supported through ARPA grant DAAH01-95-C-R009 monitored by the US Army Missile Command, and by grants from the NSF and from the US Air Force.

the feasibility of using shape information in the presence of occlusions by retrieving drawings from a collection of about two-hundred illustrations for a computational geometry textbook. We plan to merge these two retrieval demos into one once the technologies involved are well understood in isolation.

The next section summarizes our approach to image retrieval. Then, section 2.2 describes the interface and architecture of our retrieval system based on texture and color. Section 2.3 examines the principles and implementation of our shape-based retrieval system. Finally, section 2.4 discusses both foundational and technical work in progress towards larger-scale image retrieval demonstrations.

2.1 Design Principles

Image queries are both generic and unpredictable. Most of the time, users look for pictures they have never seen before, so they do not look for a very specific picture. Looking for a picture of firemen at work may sound specific, but in the picture there may or may not be fire, smoldering buildings, water hoses, helmets, or fire trucks. How does one go about looking for a fireman picture in a large, unknown data base of images without captions? Hoping for an "exact answer," all and only the pictures with firemen at work, is unrealistic.

To make retrieval realistic, we require the user to describe pictorially what s/he wants, so that the search can proceed completely free of any semantic interpretation. To look for firemen, one can sketch fire, a shape detail of a helmet, blackened building structures, and ask that any one of them, or perhaps all of them, appear in the retrieved images. The user can specify rough image position and size, or instead let the system decide. Because there is no semantics, one cannot expect to get only semantically correct answers. For instance, when one looks for a picture of Manhattan, it may well be that photographs of VLSI chips are returned as well (see figure 1). In our approach, satisfying a query is not a process of interpretation, but one of restricting the set of pictures to be visually scanned by the user. In this process, there can be a large fraction of false positives, and a small fraction of false negatives. If we can reduce a 200,000-image data base to a 300-image data base in a few minutes of work, retrieval will be much easier and cheaper than the $450 average cost of a typical search in a data base like the Kodak Picture Exchange or PressLink [Larish, 1995]. And the result will be useful even if 280 of the resulting 300 images are not of firemen at work.

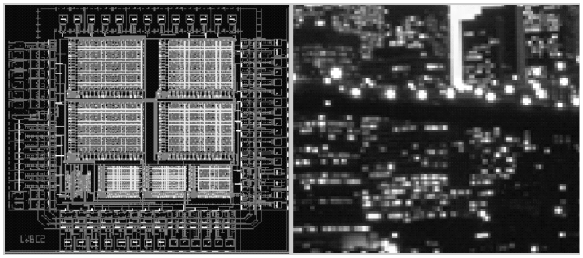


Figure 1: Manhattan may be pictorially similar to a VLSI chip.

Another guiding principle in our approach is that search time should be sublinear in the number of pictures. At query time, there is just no time to go through even a cursory description of each image, so most of the work must be done at image entry time. Images must be described in terms of basic primitives that can somehow be ordered, so the system can find images by looking up tables in sublinear time. Assuming that a given data base will be used extensively, it pays to spend a certain amount of computation time and resources when the data base is being built. This cost is then amortized over many uses of the data base.

2.2 Interface and Architecture

The interface to our texture-based image retriever is composed of three X windows: a texture easel (figure 2), a query window (figure 3) and a result window (figure 4). The user selects textures from the easel[1], a scrollable window with texture samples, and clicks them into any of the 25 regions of the query window. Not all regions need be filled. In the future, the query window will be replaced by a paint-like interface that lets the user dip a brush in texture and paint with it. The user then clicks the "Run Query" button (bottom of figure 3) and waits a few seconds for the answer. The system returns the most similar images, in a sense to be discussed below, to the query image. The four most similar ones are shown in figure 4.

In brief, here is the architecture of the query system that achieves this result. Details are given in [Rubner and Tomasi, 1996] in these proceedings. When images are entered into the system,

[1] Some of the textures and images for this demo are from R. Picard's texture database, available at `http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html`.

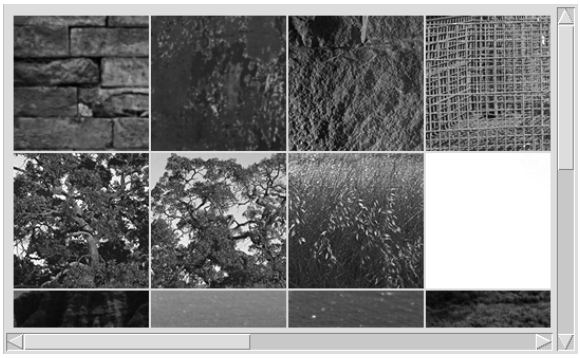


Figure 2: The texture easel.

Figure 3: The query window.

they are analyzed for texture and color content. This analysis returns one descriptor, a vector of 16 real numbers, for each of 4096 overlapping image regions for a typical 512×512 image. Thus, an image of $256K$ pixels (bytes) is described by $16 \times 4096 = 64K$ real numbers ($256K$ bytes). In bytes there is no compression, but only few of the digits in the texture descriptors are significant. We are exploring vector quantization techniques to reduce the size of the texture descriptors without losing significant information.

Figure 4: The four best matches to the query in figure 3.

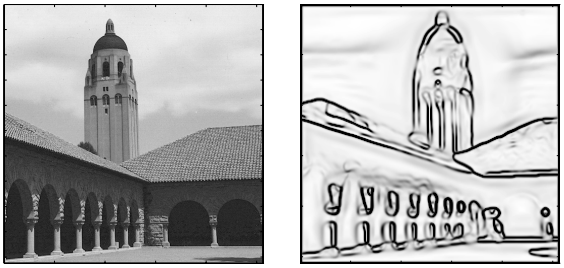
The indexing structure is then conceptually a large set of points in $\mathcal{R}^{16}$, one cloud of 4096 points for each image. This large set is split into 25 subsets, one for each rectangle in the query window (figure 3). In other words, we have essentially 25 separate data bases in our implementation. This incorporates rough positional information into the query. The mean texture vector in each query region is then used as an index into the corresponding data base.

The problem now is to take a query texture vector for one of the 25 rectangles, and look for some of its nearest neighbors in the data base. Nearest neighbor searches or range searches in a large Euclidean space are combinatorially explosive problems, but can be solved well by relaxing the guarantees on the returned results. We use the algorithm described in [Arya and Mount, 1994] for approximate nearest neighbor search.[2] This algorithm returns nearest neighbors with a relative error margin of ϵ, in the sense that a point p is an approximate nearest neighbor of q if for all points p' in the data base the distance between p and q is less than $1 + \epsilon$ times the distance between p' and q. Preprocessing takes $O(n \log n)$ time and $O(n)$ space, where n is the total number of points in the data base, and k nearest neighbors can be computed in $O(k \log n)$ time. The parameter ϵ can be specified to be as small as desired, but of course the constant factors in the asymptotic performance bounds above depend on ϵ, as well as on the dimension of the space. In practice, we have found that the algorithm takes only a few seconds on an SGI Indigo2 workstation to return a dozen neighbors in a sixteen-dimensional space of about one-hundred thousand points.

Images returned by the 25 separate queries, one for each rectangle in the query window, must be combined into the answer to the user. To this end, we devised a scoring scheme that accounts for both the size of a region with a given texture and the similarity between the query texture and the image texture. Scores computed according to this scheme are added over all 25 data bases for each of the returned images, and a list of images sorted by decreasing scores is returned to the user.

The representation of textures is described in [Rubner and Tomasi, 1996] in these proceedings. That paper makes two main contributions. The first is a computationally sound definition

[2] We thank the authors of that paper for making their code available to us.



(a) (b)

Figure 5: (a) A picture of the Stanford campus and (b) the texture edges found by our texture analysis method. Regions with no strong texture edges yield clean image descriptors for retrieval.

of *significant texture regions*, which allows finding image regions with "clean" texture, that is, regions that would usually be judged to contain a single texture. This definition is based on a novel notion of texture contrast. The second contribution is a "clustering-by-smoothing" procedure that coalesces texture vectors from a given region into small clusters of similar vectors. This procedure is based on a new form of edge-preserving smoothing, for which we have also developed an efficient approximate implementation. Figure 5 shows the texture boundaries found by our method in a picture of the Stanford campus.

With these techniques, a single image can be processed for entry into our data base in less than a minute on a workstation and yields clear, distinct clusters of texture vectors.

2.3 Shape-Based Illustration Indexing and Retrieval

We have developed a general set of ideas for indexing computer-generated technical illustrations based on the shapes present in them, so that they can be efficiently retrieved later using as the key other 'similar-looking' illustrations (either pre-existing, or interactively drawn by the user). We have restricted out attention to the domain of computer-generated technical illustrations for now, where shape information is both precisely available and the main way in which pictorial meaning is conveyed. After we have techniques that can operate successfully in this domain, we plan to port them to other kinds of pictorial or image data as well by applying shape extraction techniques from computer vision.

We proceed as follows: given an illustration P, we compute a compact index $\iota(P)$ which records the principal shapes present in P and their location/orientation/size. Then given a collection of illustrations, we compute a data-structure for recording their indices so that later queries can be answered efficiently. At retrieval time we are given another illustration Q; we compute $\iota(Q)$ and then search the data base for illustrations P_i whose index $\iota(P_i)$ is 'similar' to $\iota(Q)$.

An illustration P for us is a collection of instanced graphics primitives (lines or polylines, circular arcs, Bézier cubic or B-spline arcs, marks, etc.), as is almost universally the case with the illustrators in common use today (e.g., Adobe Illustrator, Aldus Freehand, Xfig, etc.). We start with a a collection of *basic shapes* which may be built-in, or user-definable. In the index $\iota(P)$ of an illustration P we record 'which basic shapes appear where.' In other words, for each basic shape, we record in the index the translation, rotation, and scale transformations which cause this basic shape to match well some of the shapes present in P, according to the Hausdorff distance [Chew *et al.*, 1993]. Thus we can think of the index as a list of 'colored' points in $\mathcal{R}^4$, where the four coordinates are the four parameters defining the transformation, and the color is the label of the basic shape involved. (We actually store the logarithm of the scale parameter, so as to make variations in scale correspond to point translations in $\mathcal{R}^4$, just like for translations and rotations).

When a query illustration comes in, we compute its index $\iota(Q)$ in the same way. At the moment we match $\iota(Q)$ with the index of every illustration in the data base, by computing in $\mathcal{R}^4$ the *colored one-way Hausdorff distance under translation* between the two point sets representing the indices; a fuller explanation of the matching mechanism is given in [Cohen and Guibas, 1996] in these proceedings. We are optimistic that in the future we will be able to attain sublinear query-time algorithms (algorithms which do not need to compare $\iota(Q)$ with every other illustration index) by using computational geometric techniques on the set of indices — essentially by clustering illustrations whose indices have a small 'distance' from each other.

A library of approximately two-hundred illustrations from a geometry textbook was indexed using this scheme and then used for retrieval experiments. An interactive interface was provided for specifying the data base to be searched and the query illustration, for setting various parameters regarding the match, and for displaying the best matches found in the data base. Figure 6 shows the illustrations returned by a

query that asked for figures containing squares. More details and examples are provided in [Cohen and Guibas, 1996].

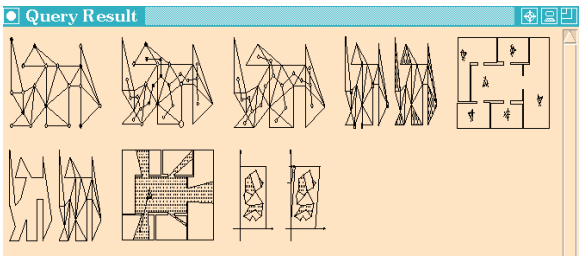


Figure 6: Eight illustrations that contain square-like parts, as returned by our shape-based query processor.

2.4 Current and Future Work

Having completed proof-of-concept demonstrations in this first year of research in the areas of texture, color, and shape, we plan to continue our project in the following directions:

- refining the basic representations;
- scaling the demos from a few hundred to several thousand images;
- merging queries based on texture, color, and shape.

2.4.1 Basic Representations

Image segmentation is our primary goal in our work towards better image representations. In fact, partitioning images into regions with approximately uniform color and texture content is important for the following reasons:

- it leads to fewer texture vectors, thereby improving the performance of range and nearest-neighbor searches;
- it provides shape primitives to be used in our shape-based searches;
- it leads to semi-symbolic descriptions of images as adjacency graphs of regions with lists of attributes.

This last advantage will enable us to provide a richer vocabulary for the formulation of queries, since we will be able to describe position with better granularity than our current implementation, as well as adjacency relations, size and shape of texture and color blobs, and so forth. We have promising preliminary results in this area, based on diffusion-like processes for color segmentation and on the combination of texture and color boundaries for a crisper definition of region contours. More on these results will appear in our future reports.

2.4.2 Merging Queries

A key goal of our image retrieval project is to provide an indexing structure for the images in our data base so that those images 'close' to a particular query image can be determined efficiently — in particular without requiring an explicit comparison with all, or nearly all, of the data base images. We are willing to pay for possibly expensive preprocessing of the data base, if this will allow the queries to be answered in such a sublinear manner.

Since our queries have to do with distance or closeness to given images, an appropriate organization of the data base is to cluster the stored images according to one or more notions of distance. When dealing with a data base of objects with a regular or homogeneous mathematical structure such as, for example, geometric points, there are well developed techniques for doing efficient proximity and more generally range searching [Mulmuley, 1993]. These typically involve partitioning the objects into a hierarchical collection of canonical subsets, so that the answer to any query can be formed by combining a small number of these subsets. Images, however, do not have such a clean mathematical structure. Image changes such as object motion or viewpoint or lighting changes can lead to images which are close perceptually but far mathematically, if we view them as arrays of 2D pixel values. That is why, as we already saw, we plan to base our notion of image similarity on color, texture, and shape indices. It is hard to see how to incorporate all these different modalities into a single homogeneous mathematical structure. Furthermore, as our notions of what are the best color, texture, and shape indices are evolving during this project, we do not want to closely intertwine our data base searching strategies and the detailed structure of these indices.

One way out of this dilemma is to base the data base query algorithms on high level abstract operations satisfying certain mathematical properties. In this way we can develop efficient searching strategies, while still tuning and modifying the individual image comparison operations. An example will help to illustrate this point. Suppose that all we have is a primitive which, given two images P and Q, returns a non-negative real number $d(P, Q)$ as their 'dis-

tance'. Perhaps we compute $d(P,Q)$ by using information from all our indices (color, texture, shape) to provide a measure of the minimum work needed to 'edit' image P into image Q. The theory of such 'string-edit' distance functions is well developed in mathematical biology for sequences of nucleotides or amino acids [Sankoff and Kruskal, 1983]. It should be clear that such functions naturally satisfy the triangle inequality $d(P,R) \leq d(P,Q)+d(Q,R)$. This makes the set of images into a metric space and so it makes sense to cluster images based on their mutual distances and use that structure in answering queries. In fact, if the query is far from some image in a cluster, it cannot be that close to any other image in that cluster. Yianilos [Yianilos, 1993] has developed certain tree-like structures that allow for efficient nearest neighbor searching in such general metric spaces. Alternatively, we can try to use some of the distance geometry [Havel, 1995] techniques developed in mathematical chemistry in order to embed our images as points in a fairly low-dimensional Euclidean space in a way which preserves their distances. Then much more powerful geometric range searching techniques can be used.

2.4.3 Larger Demos

In our transition to a demo with several thousand images and different query modalities, we face three main challenges:

- sharpening the basic image descriptors so that queries still return small image sets with a small ratio of false negatives;
- tuning the performance of range and nearest-neighbor search algorithms so that the larger number of images can still be accommodated in short response times;
- making images and descriptors available on line at a reasonable cost.

Improvements in the area of image descriptors have been discussed under "Basic Representations" above. As to the efficiency of search algorithms, our main tool will be the use of clustering and vector quantization techniques to "summarize" groups of similar descriptors. Given our initial experiments and the performance of typical vector quantization algorithms, we expect to see a reduction of at least one order of magnitude in the number of descriptors per image.

Making several thousand images available on line is a technical difficulty, not a conceptual one. We have placed an entire data base of 20,000 images onto a laser disc. The recorder and player is connected to a workstation through a serial port for commands and through a frame grabber for the images, since the recorder stores data in analog format. A small thumbnail sketch is stored on computer disk for every image on the laser disc, so the latter needs to be accessed only when the data base is built and when the user requests a specific image after examining the result of a query. We have developed a complete software package for accessing the pictures so that the physical storage medium is transparent to the user.

3 Vision for Robots

This section highlights two projects on vision for robots which are sponsored by ARPA. The first, described in section 3.1, involves a close cooperation between the computer vision (Tomasi) and motion planning (Latombe) groups at Stanford. It led to the construction of a prototype robot observer, which follows a moving target in an obstacle-cluttered environment without losing sight of the target.

The second project (section 3.2) adds a safety margin to the robot observer by allowing detection of unexpected, possibly moving obstacles. Using a multi-camera version of depth from focus to compute a coarse depth map in real time, this system intentionally compromises accuracy for reliability. The result is a robot that has navigated for hours in crowded rooms without ever hitting anyone or anything, using only a personal computer for its vision processes.

3.1 The Intelligent Observer

The *intelligent observer* (IO) is a system that provides a human user with intuitive, high-level control over a mobile robot which autonomously plans and executes motions to visually track a moving target (see Figure 7). The user sends commands, such as "follow the next moving object which enters the view", and receives real-time feedback, such as a graphical display of the positions of the observer and target overlaid on a map of the environment.

The IO responds to high-level commands which are specified at the task level. There is no need for a "virtual joystick" or any other such control. Furthermore, the robot uses its internal representation to provide a more flexible feedback mechanism than would be possible by simply displaying the image seen by the observer's cameras. The IO can fuse information from var-

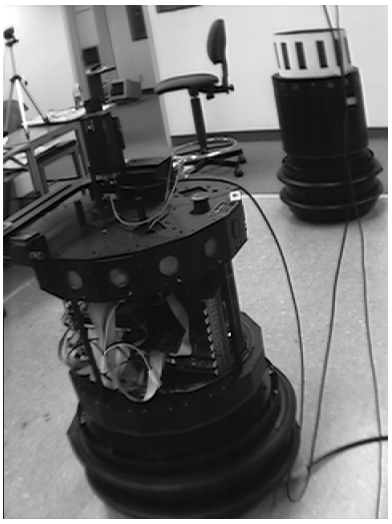


Figure 7: The intelligent observer pursuing another robot.

ious sensors and, using geometric information about the environment, reconstruct a view of the observed scene. The IO project brings together concepts and algorithms from computer vision, motion planning, and computer graphics in order to create a robust, useful, and integrated system. Possible applications are remote monitoring, surveillance, and teleoperation.

The complete IO consists of five major modules:

Landmark Detection. As the observer moves around it keeps track of its own current position by visually detecting artificial landmarks placed throughout the environment. The positions of the landmarks are included in a map which is provided to the IO.

Target Tracking. The central task of the IO is to observe moving targets, so the capability is provided to both recognize when a new target enters the IO's field of view and to track the target as it moves. Since the robot must respond to the movement of objects, all tracking occurs in real time.

Motion Planning. The IO remains in view of a moving target at all times by employing a probabilistic on-line algorithm that avoids collisions and minimizes the probability of obstructions to visibility.

User Interface. The user is presented with either a two-dimensional overhead view or a three-dimensional rendering from an arbitrary vantage point. This module uses a known geometric model of the environment and information about the positions of the observer and the target. Only a small amount of new information must be transmitted to update the display.

Motion Control. The motion controller coordinates communication with the other components and produces low-level commands to control the robot. It also updates the current estimate of the robot's position based on feedback from the landmark detector and odometric information. Finally, it requests goal points of the motion planner, and sends information updates to the user interface.

More details on the intelligent observer can be found in [C.Becker *et al.*, 1995].

3.2 Obstacle Avoidance by Depth-From-Focus

The point of this research is to show that depth from focus provides an inexpensive and reliable sensing method for obstacle avoidance. A $5,000 prototype of our system has so far accumulated more than twenty hours of navigation with no failure in demanding indoor and outdoor environments under varying lighting conditions, crowds of people walking past and towards the robot, and treacherous steps and obstacles all around. Our robot gracefully coasts around tables and chairs, mingles with people who pay little or no attention to it, and happily spins around when children hold hands around it singing "ring around the roses."

Our system performs no convolutions except those computed for free by defocused lenses. It has no explicit mathematical model of how defocusing alters an image, and the processing is simple enough to be carried out at frame rate on a personal computer. Three images taken from the same viewpoint but with lenses focused at different distances are compared for sharpness. The setting of the corresponding camera's focal length is taken to be the desired depth at any one image position. Although depth resolution could be increased by using more cameras, the simple strategy used to control our robot only needs to tell close from medium-distance from far at a relatively coarse grid of image regions. Any control strategy must eventually compress depth information into a few bits of information, since the choices for the robot's action are limited. Consequently, determining how much depth information is required before building the depth sensing module guarantees that all and only the necessary information is computed.

This research is about an extremely simple idea. Simplicity itself is the point, as it yields at the

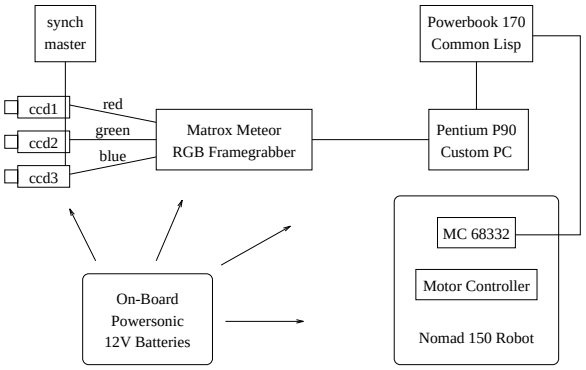


Figure 8: Schematic diagram of the depth-from-motion robot architecture.

same time efficiency and reliability. Our experiments show our depth-from-focus system to be an alternative to sonars for its greater accuracy, longer distance range, high reliability, and low computational cost. This is one of the first examples of a vision algorithm that is hard to defeat. A schematic diagram of the mobile robot system is shown in figure 8. More details are given in [Nourbakhsh *et al.*, 1996] in these proceedings.

Acknowledgements

We acknowledge the contributions of the following people who either did some of the work described in this report or contributed with useful discussions: David Andre, Hector González-Baños, Craig Becker, Scott Cohen, Michael Genesereth, David Hoffman, Jean-Claude Latombe, Illah Nourbakhsh, Brian Rogoff, Yossi Rubner, Mark Ruzon, and Guillermo Sapiro.

References

[Arya and Mount, 1994]
S. Arya and D. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the 5th SIAM Symposium on Discrete Algorithms (SODA)*, pages 271–280, 1994.

[Bielski, 1995] L. Bielski. 20 million photos will be digitized at Time Warner: the image database of the future begins. *Advanced Imaging*, 10(10):26–28,91, October 1995.

[C.Becker *et al.*, 1995] C. Becker, H. González-Baños, J. C. Latombe, and C. Tomasi. An intelligent observer. In *Proceedings of the Fourth International Symposium on Experimental Robotics, ISER '95*, June 1995.

[Chew *et al.*, 1993] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. In *Proceedings of the Fifth Canadian Conference on Computational Geometry*, pages 151–156, 1993.

[Cohen and Guibas, 1996] S. D. Cohen and L. J. Guibas. Shape-based illustration indexing and retrieval: some first steps. In *Proceedings of the ARPA Image Understanding Workshop*, 1996.

[Faloutsos *et al.*, 1994] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[Havel, 1995] T. F. Havel. Distance geometry, In *Encyclopedia of NMR*, pages 1–10. John Wiley & Sons, 1995.

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

[Larish, 1995] J. Larish. Commercial images on-line: Kodak's still picture exchange for print and film use. *Advanced Imaging*, 10(4):38–39, April 1995.

[Mulmuley, 1993] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Nourbakhsh *et al.*, 1996] I. R. Nourbakhsh, D. Andre, C. Tomasi, and M. R. Genesereth. Obstacle avoidance via depth fom focus. In *Proceedings of the ARPA Image Understanding Workshop*, 1996.

[Picard and Minka, 1995] R. W. Picard and T. P. Minka. Vision texture for annotation. *Multimedia Systems*, 3(1):3–14, 1995.

[Rubner and Tomasi, 1996]
Y. Rubner and C. Tomasi. Spectral texture descriptors. In *Proceedings of the ARPA Image Understanding Workshop*, 1996.

[Sankoff and Kruskal, 1983] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

[Yianilos, 1993] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 311–321, 1993.