

Using Spatial Sorting and Ranking in Model-Based Object Recognition

Gísli R. Hjaltason*

Manjit Ray†

Hanan Samet*

Isaac Weiss†

Computer Science Department, Center for Automation Research, Institute for Advanced Computer Studies

University of Maryland, College Park, Maryland 20742

{grh,manjit,hjs}@cs.umd.edu, weiss@cfar.umd.edu

Abstract

A new method has been developed for overcoming some of the major obstacles of object recognition. It represents both 2D images and 3D models from a database by view-point invariant descriptors in the same invariant space. As there is a large number of possible matches between the image and model invariants, there is a crucial need to perform this match efficiently. We answer this need by using an adaptation of spatial sorting based on the octree representation of the model points and image lines in three-dimensional space. We show the advantage of the octree method over the traditional “brute-force” approach and we compare the merits of the two methods. This makes it possible for the recognition task to be performed in a reasonable time for a large number of models.

1 Introduction

In a typical recognition task, one has an image of an unknown object, and the objective is to match it with a known set of objects, given in an appropriate representation such as models or reference images. Since the viewpoint from which the image was taken and other camera parameters are unknown, the matching involves searching in a prohibitively large search space.

To avoid the search for the correct viewpoint, we can match viewpoint invariant descriptors of the models and images rather than the original representations. Such invariants exist for projective transformations, including viewpoint changes, between spaces of the same dimensionality, e.g. a projection of a planar object onto an image. However, when we project a 3D object onto a 2D image, the depth information is lost. This loss creates difficulties in two ways:

1. There are no full invariants, only invariant constraints. That means that the search space cannot be eliminated

*The support of the National Science Foundation under Grant IRI-97-12715 is gratefully acknowledged.

†The support of DARPA under Order No. E655 and Air Force Write Lab under Grant F49620-96-1-0355 is gratefully acknowledged.

completely as in the 2D case, although it is far smaller than the non-invariant description.

2. General objects cannot be identified uniquely. In principle, many different objects, differing only in their depth coordinates, can yield the same 2D image.

In this paper we deal with the two issues above. In the first issue, since a search is unavoidable, our goal is to reduce the complexity of the search as much as possible. Two key steps are involved here: a) representation of both the models and the images as entities in an invariant space, and b) using spatial sorting and ranking algorithms to perform efficient matching in this space.

The second issue is dealt with by the use of modeling assumptions. Such assumptions should make up for the missing depth information. One such assumption is the use of a predefined set of known models rather than trying to recognize a general unconstrained object. Such modeling is already partly implied in our definition of object recognition. However, we have to be careful that our set of models do not contain many models that can project the same image.

Our method is described briefly as follows. We represent each 3D model as a set of points in a 3D invariant space. An image is represented as a set of invariant *lines* in the same invariant space. The fact that we have lines rather than points reflects the fact that the image is missing the depth information. The goal is now to find the model point set which is closest to the set of lines representing the image. This involves ranking distances between lines and points in 3D. The ranking is facilitated by sorting the points and lines with respect to the space that they occupy. The sorting makes use of a hierarchical spatial data structure known as an octree which is a three-dimensional variant of the quadtree.

2 Object Recognition and Modeling

Most work in invariants has concerned transformations between spaces of equal dimensionality [4, 6]. For a single view, invariants were found for planar curves, while for 3D objects, multiple views with known correspondence were involved.

Since it has been shown that there are no invariant functions of a single projection from 3D to 2D [1], modeling assumptions are required to recover 3D shape. Such assumptions can take the form of generalized cylinders, or surfaces

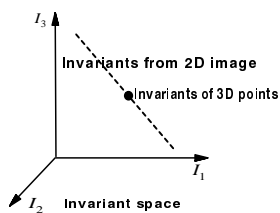


Figure 1: Invariant space.

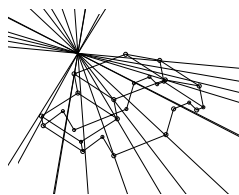


Figure 2: Intersections between lines and model points.

of revolution or orthogonal vertices. Another approach is to characterize an object by local properties like tangencies and inflection points. Our method does not make any such specific modeling assumption, but uses a set of given models.

Our task is to recognize general 3D models from a single perspective view. Using geometric invariance, the recognition task is reduced to a search for intersections between simple subspaces (lines, points) in an invariant space.

The method is based on the invariance properties of 5-point sets [7]. Each 5-point set in 3D has 3 invariants, which can be represented as a point in a 3D invariant space. The same five points, when projected on the image, give rise to a line in the same invariant space. The line parameters depend on the projection, but it always passes through the original invariant point. The invariant space is shown in Fig. 1.

The recognition algorithm proceeds as follows:

1. *Calculation of 3D model invariants:* The invariants of various quintuples are found and stored in the invariant space. An example is shown in Fig. 2.
2. *Extraction of image features:* As feature points, we used corners visible in the image.
3. *Purging of feature points:* We use only corners lying on principle directions.
4. *Determination of line-point intersections in invariant space:* Points lying close to an invariant line derived from the image are extracted using spatial sorting and ranking (Section 3). A set of possible models together with their respective poses can be hypothesized.
5. *Verification:* Candidate matching models are projected onto the image for final selection and verification.

3 Ranking and Join

3.1 Octrees

We use two different hierarchical sorting algorithms for the different spatial features — that is, the points that make up the models and the infinite lines that make up the image. The sorting methods are based on octrees (a 3-dimensional form of quadtrees). The idea is that we recursively decompose the underlying three-dimensional space into eight cube blocks until each block contains one or a very small number

of points. This is the basis of the PR octree [5]. Fig. 3a is an example of the block decomposition resulting from a PR quadtree for points in two-dimensional space.

The octree sorting technique can also be adapted to non-point objects such as collections of line segments. To avoid excessive splitting due to objects close together (or touching/intersecting), we use a splitting strategy wherein if a block contains more than s objects, then the block is split once only (i.e., the new child blocks are not split further for that insertion). Fig. 3b is an example a PMR quadtree for two-dimensional line segments inserted in alphabetical order. Notice that non-point objects may be stored in more than one leaf block.

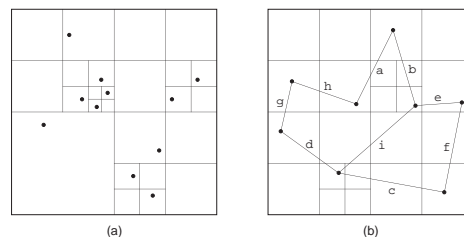


Figure 3: (a) A PR quadtree for points, and (b) a PMR quadtree for line segments with a splitting threshold of 2.

3.2 Incremental Nearest Neighbor Algorithm

The incremental nearest neighbor algorithm obtains objects from a set of spatial objects S in the order of distance from a given query object q (termed *ranking*). The objects are organized with a hierarchical spatial data structure, octrees in our case. The algorithm is incremental in that the nearest neighbors are reported one-by-one with the aid of a priority queue. Thus, it is especially useful when the exact number of neighbors needed is not known in advance. The algorithm is described in detail in [3].

3.3 Incremental Distance Join

When the ranking is with respect to a set of reference objects, the process is more complex than finding the nearest object to one reference object. The distance join operation is an extension of the ranking operation that operates on two sets of objects S_A and S_B . Informally, it is an ordering on all pairs of objects from S_A and S_B , based on distance. If one of the sets contains only one element, then the distance join operation is equivalent to the ranking operation.

The incremental distance join algorithm [2] is based on the same principles as the incremental nearest neighbor algorithm. Thus, the two sets must be organized with a hierarchical spatial data structure, A and B . The incremental join algorithm may be thought of as applying the incremental nearest neighbor algorithm simultaneously to A and B . As in the incremental nearest neighbor algorithm, a priority queue is used, where the elements are pairs of nodes or objects, one from each of A and B .



Figure 4: Images of Model 2.

4 Experiments

Experiments were performed on two images: a truck and a HMMWV vehicle (Fig. 4). From each of the two corresponding models, two invariant point sets, P_1 and P_2 , are extracted, the smaller P_1 under a more restrictive selection of feature sets. From each image, two invariant line sets are defined, the smaller from true corner features detected in the image and the larger from both true and false corners.

Our experiments were run on a Sun Ultra 1 Model 170E machine, with 64MB in main memory. Three search methods were tested: 1) The incremental join algorithm the incremental nearest neighbor algorithm, and 3) brute force. For the incremental join algorithm, we tested two methods of building an octree for lines. The first is based on PMR quadtree rules; the second builds the octree based on the same spatial decomposition as that of the octree for points. The second method led to faster octree build times, but sometimes slower search times. For the experiments using the nearest neighbor algorithm, we computed a fixed number of neighbors for each line, using the PR octree for points.

The first two methods showed a clear advantage over the brute force method. For the join method, the speedup ranged from about 2 to more than 6 (this accounts for both the build time of the octree for lines as well as search time), while for the nearest neighbor method the speedup ranged from about 4 to as high as nearly 17. In both cases, the speedup increased as the number of points and lines in the invariant space increased. The nearest neighbor approach appears to be superior to the join approach. However, we have not yet explored the full advantages of the join method. The advantage of the nearest neighbor approach is that it enables adding lines and finding their nearest n points. In this case, we have the ability to converge on a solution in an incremental manner as data is added.

The advantage of the join method over the nearest neighbor method is that it operates simultaneously over all the lines in the line set. With the nearest method, once a nearest neighbor query has been terminated for line l_1 and a new one started for the line l_2 , we cannot request more neighbors for l_1 without computing all the nearest neighbors from scratch, even the ones that have already been determined. The reason is that after each nearest neighbor query has terminated, we do not keep the contents of the priority queue.

5 Concluding Remarks

The use of hierarchical methods for finding distances in an invariant space greatly speeds up the task of object recognition. The speedup factor increases as the number of image and model features increase. These hierarchical methods would not be possible without the invariant representation of both the models and the images.

References

- [1] J. B. Burns, R. S. Weiss, and E. M. Riseman. The non-existence of general case invariants. In *Geometric Invariance in Machine Vision*, pp. 120–134. MIT Press, Cambridge, MA, 1992.
- [2] G. R. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *Proc. of the ACM SIGMOD Conf.*, Seattle, WA, June 1998.
- [3] G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *Advances in Spatial Databases — 4th Intl. Symp., SSD'95*, pp. 83–95, Portland, ME, August 1995. Springer Verlag LNCS 951.
- [4] E. Rivlin and I. Weiss. Local invariants for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:226–238, 1995.
- [5] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.
- [6] I. Weiss. Geometric invariants and object recognition. *Intl. Journal of Computer Vision*, 10:207–231, 1993.
- [7] I. Weiss and M. Ray. Model-based recognition of 3D object from one view. *Proc. of European Conf. in Computer Vision*, June 1998.