Learning Indexing Functions for 3-D Model-Based Object Recognition

Jeffrey S. Beis and David G. Lowe
Dept. of Computer Science
University of British Columbia
Vancouver, B.C., Canada
email: beis@cs.ubc.ca
lowe@cs.ubc.ca

Abstract

Indexing is an efficient method of recovering match hypotheses in model-based object recognition. Unlike other methods, which search for viewpoint-invariant shape descriptors to use as indices, we use a learning method to model the smooth variation in appearance of local feature sets (LFS). Indexing from LFS effectively deals with the problems of occlusion and missing features. The indexing functions generated by the learning method are probability distributions describing the possible interpretations of each index value. During recognition, this information can be used to select the least ambiguous features for matching. A verification stage follows so that the final reliability and accuracy of the match is greater than that from indexing alone. This approach has the potential to work with a wide range of image features and model types.

1 Introduction

Model-based object recognition consists of matching features between an image and a pre-stored object model. The hypothesize-and-test paradigm uses minimal sets of features to form correspondence hypotheses. From these, model pose is calculated, and model presence is verified by back-projection into the image and a search for further matches. The entire process can be very expensive because each verification is expensive, and because of the computational complexity of generating hypotheses through exhaustive comparison.

Indexing is one way to combat the computational burden. It is a 2-stage process. At compile-time, feature sets derived from object models are used to generate vectors corresponding to points in an index space. These are stored in some appropriate data structure, with pointers back to the appropriate models. At run-time, the same process is used to generate index vectors from test images. The data structure is then used to quickly access nearby pre-stored points. Thus, correspondence hypotheses are recovered without comparing all pairs of model/image feature sets.

Previous methods have largely ignored the potential ambiguity among indexed hypotheses. This situation arises when a single index vector from a test image retrieves several conflicting match hypotheses, and occurs in almost every case. To deal with this, we can demand that our indexing mechanism supply us with more information, i.e., not just with a set of correspondence hypotheses, but also the probability that any one of them is a correct interpretation for the data.

We introduce *indexing functions* which are designed to estimate the probabilities. A learning-from-examples approach is taken to build the functions, in an off-line learning stage. The "examples" used are groupings of features ("Local Feature Sets") extracted from sampled views of the objects in the database. Grouping is an essential step which reduces the number of indices that will be considered. The indexing functions interpolate between nearby views and model the probability distribution over the possible interpretations at each point in the index space.

At run-time, the first part of the indexing stage efficiently recovers a set of nearest neighbors to each index value. Then, the indexing functions interpolate and smooth the local interpretations. The resulting probabilities are used to rank the match hypotheses before sending them to a verification stage. These rankings will in general mean that correct interpretations can be investigated first, which leads to improved over-all efficiency in the recognition process.

2 Previous work

We are interested in indexing 3-D models from single 2-D images. Much work in the past on indexing has dealt with the easier problems of 2-D models, e.g. [KJ86] [SM92a], or 3-D models where more information is available, such as with 3-D range data [SM92b] or multiple images [MWS93].

When compiling an index, the most efficient storage possible can be achieved by using shape descriptors invariant with respect to viewpoint. For each underlying model feature set, only a single index value must be stored. It has been shown (e.g. [CJ91]) however, that no general, non-trivial invariant functions exist for 3-D point sets under the various standard projection models. This explains the two major strains of indexing approach that are present in the literature.

The first methodology involves adding constraints to create a restricted domain of 3-D feature sets that can be used to generate invariants. Forsythe, *et al.*, [FMZB90] create invariant functions from grouped planar curves, such as coplanar pairs of conics or sets of at least 4 coplanar lines. Lambdan and Wolfson [LW88] also restrict to planar faces. They generate affine bases in which the coordinates of points on the face are invariant. [RZMF92] generalize this work to deal with the more general perspective transformation.

Such invariants are very useful when available, but many objects will not contain one of these. A second approach ignores invariants, and attempts to store an index value for each possible appearance of a feature set. These methods generate indices for multiple views of a model over some tesselation of the viewing sphere.

Thompson and Mundy [TM87] use simple groupings of pairs of vertices to retrieve hypothetical viewing transforms for their models. Because of the lack of specificity in the indices, a voting scheme must be used to collect the noisy pieces of information into meaningful hypotheses. Clemens and Jacobs [CJ91] show that the view variation of indices derived from point sets generate 2-D sheets embedded in 4- (or higher) dimensional index spaces. Jacobs [Jac92] provides a more space-efficient construction which reduces the 2-D sheets to two 1-D lines, each embedded in a 2-D space.

These methods use hash tables to store and access their indices. Attempting to cover all possible appearances of a feature set by filling in a discrete look-up table is a difficult task, especially in higher-dimensional spaces. A better approach is to use interpolating functions to approximate the regions between samples (i.e. views). [PE90] suggested using Radial Basis Functions (RBF) for this purpose. Although segmentation and correspondence were achieved manually in their system, they were able to model the full range of appearances of an object from a relatively small number of examples.

3 General framework

3.1 Indexing

As just mentioned, a common factor in previous indexing methods is that they discretize the indices and fill hash tables with the quantized values. The advantage is that the run-time indexing step can be done in constant time. The disadvantage is that the space requirements become excessive with even moderately complex feature groupings and moderate noise in the data. This arises due to the standard way these methods have dealt with noisy images. During table construction, an entry is made in each hash bin that is within an error radius ϵ of the actual model index values that are to be stored. Then all relevant hypotheses can still be accessed by looking in just one bin.

The number of entries is exponential in the index dimension, and the base of the exponential increases with noise and the required degree of specificity (more specificity equals finer bin divisions.) The only other way to deal with the noise issue is, for each image index value, to search the ϵ-volume of index space around that index at run-time, and then to form the union of the recovered hypotheses. This certainly mitigates the original advantage of the approach.

A second issue which is not adequately addressed by other methods is the saliency of the indices. They generally treat all recovered hypotheses as being equally likely to be correct interpretations. Because the verification stage is relatively expensive, a heavy burden is placed on the indexing stage to provide only a small number of hypotheses. A higher-dimensional and/or more finely partitioned index space becomes a necessity, exacerbating the storage problem. We can do better by weighting each match hypothesis according to the proximity of the index vector to the pre-stored model feature vectors, and according to the uniqueness of the possible interpretation.

Our approach addresses the above issues. During a learning stage, index vectors are generated from multiple views of the models. These are stored in a k-d tree [FBF77], which allows for efficient run-time determination of nearest neighbors. In contrast to hash-bin methods, no extra storage is needed to account for noise: a single datum is stored for each view of a model group. At the same time, a machine learning algorithm is used to tailor indexing functions, which will give us probability distributions over the set of potential interpretations stored in the index structure.

At run-time, a set of neighbors is recovered for each index vector computed from the test image. These are fed to the indexing functions, and the resulting probability estimates used to rank the hypotheses for the verification stage. An index vector with many of the nearest neighbors corresponding to the same hypothesis will generate a single, high-probability output, while one with several nearby conflicting interpretations will produce many lower-probability hypotheses.

The ranking step allows us to limit, in a principled manner, the time spent searching for objects in an image. Ambiguous shapes (those which match too many of the pre-stored indices) might be dropped from further consideration by thresholding on the computed probability estimates. Or, we could choose to look at only a certain number of the most likely candidates before stopping. Either way, ranking serves as an extra filter on hypothesis generation, leading the whole recognition process to be more efficient.

3.2 Grouping

For indexing to be effective it is important that some data-driven grouping mechanism produce sets of features likely to come from a single object [Low85]. Grouping can be based on certain non-accidental properties of images such as edge parallelism, co-termination, and symmetry. Groupings are non-accidental if there is a high probability that their components stem from the same underlying cause, i.e., from the same object.

These types of grouping could be said to have "qualitative invariance" in that, for example, segments co-terminating on a model will always project to co-terminations in an image. We can use them to generate real-valued, multi-dimensional indices. Each index dimension will correspond to either some property of a particular feature in the grouping (e.g., a statistical value for a texture region) or to a relationship between two or more of the features (e.g., angle between two edge segments - see Figure 1).

Features should be chosen to be invariant to as many of the model's degrees-of-freedom (DOF) as possible. Then the learning algorithm only has to model the variation in the few remaining DOF of the pose. In this case, angles and edge-length ratios are invariant to translations, scaling, and image-plane rotations. These features can also be measured very precisely,

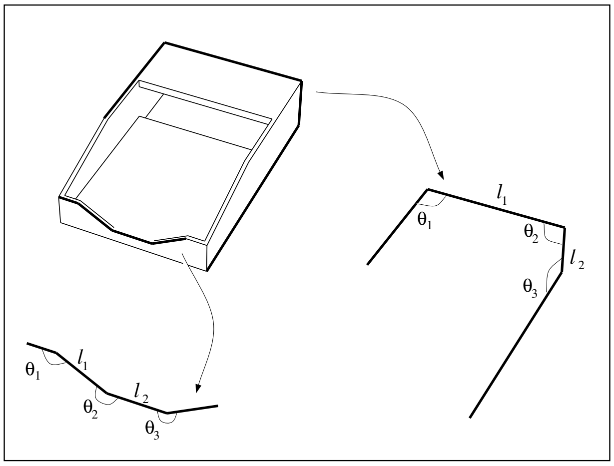

Figure 1: N-segment chains as examples of Local Feature Sets (with $N = 4$). This type of grouping generates feature vectors $\vec{\mathbf{x}} = (\theta_1, \theta_2, \theta_3, l_2/l_1)$.

leading to accurate indexing. (Note that the exterior edge lengths are not included, as they have much greater uncertainty than the interior ones due to their "free" ends.)

As a further requirement, we stipulate that our groupings must be "local" within the images, where "local" will be some function of the size and separation of features in the set. In general, several of these groupings will be derived from each view of a model. Because of this redundancy, using Local Feature Sets (LFS) for indexing increases robustness to missing features either from occlusion or due to the inadequacies of feature detectors.

3.3 Learning

As mentioned above, while the qualitative properties which are used to form feature groupings are viewpoint invariant, the feature vectors derived from these groupings are not. Since we do not have invariant descriptors, each underlying model grouping generates not a single index value but a range of values occupying some volume of the index space. Changing one's viewpoint about a model grouping corresponds to moving about within that volume.

As we do not have analytic expressions to describe these volumes, we use a learning algorithm to approximate them. Note that at any point in the space, several volumes may be overlapping. Formally, if $\vec{\mathbf{x}}$ is a local feature vector derived from an image and m is a correct match of features to a particular model, we want to learn $P(m|\vec{\mathbf{x}})$ for each possible interpretation m using a set of examples $(\vec{\mathbf{x}}, P(m|\vec{\mathbf{x}}))$.

Because the index values for a model grouping are a complex, non-linear function of viewpoint and projection from 3-D to 2-D, we require a method that can learn non-linear mappings. In the context of object recognition, Radial Basis Functions have been used to learn the continuous range of appearances of simple wire-frame models [PE90] [PG89]. While successfully demonstrating the interpolation ability of RBFs, these systems are brittle. RBF output is a binary decision on object presence/absence based on a single feature vector derived from an image, and this fails in the case of missing features.

There is a more robust way to apply RBFs, which is as an indexing mechanism. With m from above, we define indexing functions $\vec{\mathbf{f}}$:

$$P(m|\vec{\mathbf{x}}^I) \approx \vec{\mathbf{f}}_m(\vec{\mathbf{x}}^I) = \sum_n C_{mn} G(\vec{\mathbf{x}}^I - \vec{\mathbf{x}}_n)$$

where $\vec{\mathbf{x}}^I$ is an image vector, G are the (radial) basis functions centered at the $\vec{\mathbf{x}}_n$, and the C_{mn} are coefficients determined by the learning algorithm. In this paper, a simple form of RBF network is used: G are taken to be Gaussians; the centers $\vec{\mathbf{x}}_n$ are the full set of training examples (T); and the calculation of C_{mn} simply involves the pseudo-inverse of the matrix with entries $G_{ij} = G(\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j)$ for $\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j \in T$ (see [PG89] for details). Note also the hidden parameters $\vec{\sigma}$, of dimension equal to the index space, which we set manually.

For the learning stage, images of 3-D object models are taken from various viewpoints. The training vectors are derived from the LFS groupings found in these images. For recognition, we use real images and, from each image vector $\vec{\mathbf{x}}_p^I$ ($p = 1$ to P), we recover N nearest neighbors $\vec{\mathbf{x}}_n^m$ ($n = 1$ to N, and m can be different for each n). The neighbors indicate which RBFs m are likely to have a significant response. Then $\vec{\mathbf{f}}_m(\vec{\mathbf{x}}_p^I)$ is evaluated for the up to PN unique correspondence hypotheses (i.e. for a given image vector p, several of the N nearest neighbors may indicate the same interpretation m).

The results are probability estimates for the image grouping $\leftrightarrow$ model grouping correspondences. These are used to rank the hypotheses so that the least ambiguous ones will be investigated first. A rigorous verification stage (iterations of back-projection and match extension) ensures that the final reliability of interpretation is much higher than that of the initial indexing.

One advantage of the RBF method is that the functions smoothly interpolate between the training examples (to "fill in" sections of the viewing sphere where no training views exist). This is in contrast to hash-table methods, where the influence of stored index values abruptly terminates at bin boundaries. Another plus is that often a single RBF center can effectively replace several of the training examples, and thus reduce the amount of data that needs to be stored.

The drawback is that it is very difficult to determine the optimal number and positioning of the centers. A simpler scheme, that contains many fewer free parameters, is Parzen windows. It is a Gaussian-weighted, k-nearest neighbors method:

$$P(m|\vec{\mathbf{x}}^I) \approx \vec{\mathbf{f}}_m(\vec{\mathbf{x}}^I) = \frac{\sum_{n:\vec{\mathbf{x}}_n^m \in NN(\vec{\mathbf{x}}^I)} G(\vec{\mathbf{x}}^I - \vec{\mathbf{x}}_n^m)}{\sum_m \sum_{n:\vec{\mathbf{x}}_n^m \in NN(\vec{\mathbf{x}}^I)} G(\vec{\mathbf{x}}^I - \vec{\mathbf{x}}_n^m)}$$

where $NN(\vec{\mathbf{x}}^I)$ are the nearest neighbors to $\vec{\mathbf{x}}^I$, and the denominator is a normalizing factor, so that the sum over all considered interpretations m equals 1. One difference is that the only pre-stored vectors factored into the estimate are those recovered from the NN search, whereas the RBF method looks at potentially many more (i.e. for each m, all training vectors associated with its RBF). Parzen windows are simple to implement, and do not require the extra storage for coefficient matrices that RBFs do.

4 Experiments

Experiments were run to test the absolute indexing performance, the degradation of performance with size of database, and to compare the two learning methods. The model database consisted of 4 CAD-type models. The training stage for a model consisted of projecting it from 120 viewpoints over a hemisphere, then forming the LFS from the projections and training vectors from the LFS. Primitive features were straight-line segments and LFS groupings were chains of 4 coterminating segments. For the RBF learning method, coefficient matrices were then computed.

To simulate a larger database of 25 models, we randomly generated 2-D, 4-segment chains which were valid LFS. The average number of training vectors from each of the 4 real models was about 7000, so $147,000$ extra synthetic vectors were stored along with the original $28,000$ in a kd-tree.

The test set consisted of 25 images, each one containing one of the objects (a notebox) from varying points of view, distances, and with varying amounts of occlusion (up to about one third hidden). A representative example of the recognition process is shown in Figure [2]. The average number of segments (groupings) per image was 276 (50). 10 NN were recovered

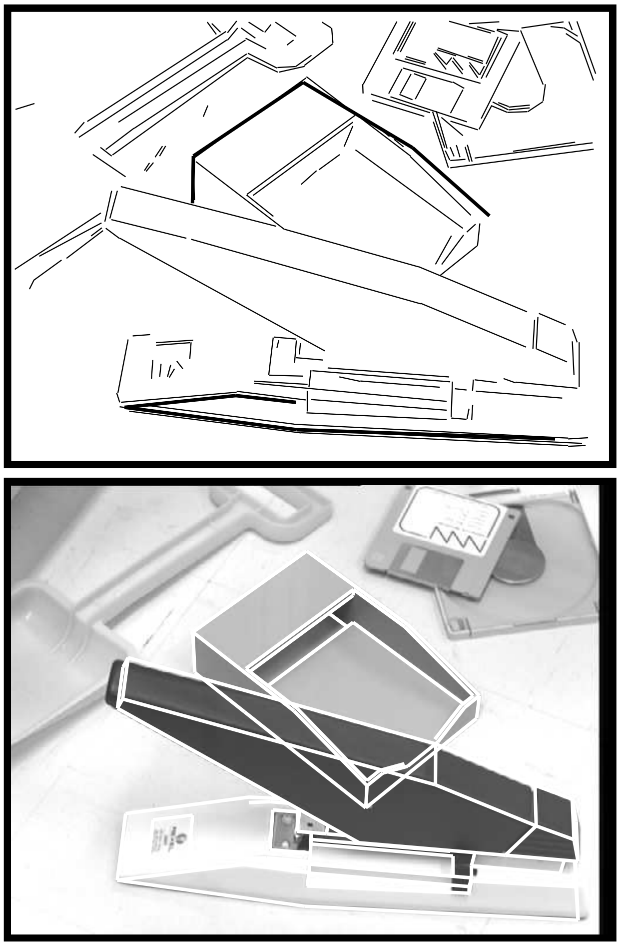

Figure 2: The upper box shows the processed image with the groupings that correctly indexed the model database. The lower box contains the final determination of object poses, overlayed on the original image.

for each vector corresponding to an image grouping, and a ranked list of match hypotheses formed by application of the indexing functions. Total time for grouping, indexing, and ranking was < 10 seconds (worst case) on a SPARC 2 computer.

Table 2 gives results for average performance on those images where indexing succeeded. Indexing can fail for two reasons. It is possible that no correct groupings are formed from detected object segments. This was the case for 3 of the 25 images. It was observed for these images that the use of other grouping types, such as sets of parallel lines, would likely have allowed for successful indexing.

The second possibility is that even if correct image grouping(s) are formed, none of the recovered neighbors is the correct model group match. In the 25

# Models	1	4	25
RBF	5.4	16.3	11
Parzen	7.3	20.4	12.2

Figure 3: Average ranking of first correct index, over all images where indexing was successful. This was 22 images for the 1 and 4 model cases, and 21 images for the 25 model case.

model experiments (both methods) this occurred for 1 image. It suggests that a good idea may be to increase the number of NN recovered with increasing database size.

The absolute performance with both methods is clearly good: all of the numbers are low. Furthermore, for those images where indexing succeeded, there was generally a large degree of redundancy, of perhaps 5 or 10 separate (often overlapping) LFS detected. In these cases, larger amounts of occlusion could have been tolerated.

There is a surprising improvement between the 4 and the 25 model cases. This is due to the addition of distractors, which in general prevented some incorrect groupings from being detected as NN, while leaving most of the correct ones unaffected. The implementation of the distractors is such that each one is effectively the only instantiation of some underlying grouping, so none of them is likely to be weighted very highly. Real 3-D models will add in some more highly-weighted indices, the magnitude of the effect being in proportion to how similar are the underlying shapes between models.

The contrasts in the two learning methods point out the main trade-off that needs to be made in order to quantitatively assess an index for ranking. That is, between *similarity* (how close are the neighbors?) and *non-ambiguity* (of the neighbors, are there conflicting interpretations?).

For example, in the Parzen windows scheme, a shapewise dissimilar index can be ranked highly if it is in a sparse region of index space (it is "unambiguous"). Conversely, the RBF method as implemented uses no negative examples, so incorrect matches, with high similarity but in a dense region of the space, will still be ranked highly (despite being ambiguous). In fact, these behaviors are simply the down side of what are generally the desirable properties of the system. The optimal balance of when and how much to weight each factor is an interesting topic for investigation.

5 Conclusion

We have presented a novel approach to indexing into a database of object models. By introducing *indexing functions* to associate probabilities of correctness with indexed match hypotheses, we determine which hypotheses are ambiguous (and can be ignored), and which are the most salient (and should be verified first). The learning methods chosen to generate these indexing functions interpolate between the index vectors sampled at various viewpoints about the stored objects. Thus, probability estimates exist for the full field of view of each model in our database.

We have demonstrated a recognition system that can achieve good performance on a difficult indexing problem. Importantly, the method remains efficient as problem complexity increases. We are currently investigating extending the method to deal with curved edge features, and non-geometric features such as texture and color.

References

[CJ91] D.T. Clemens and D.W. Jacobs. Space and time bounds on indexing 3-d models from 2-d images. *IEEE Trans. PAMI*, 13(10):1007–1017, 1991.

[FBF77] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–226, 1977.

[FMZB90] D. Forsyth, J.L. Mundy, A. Zisserman, and C.M. Brown. Invariance - a new framework for vision. In *Proceedings ICCV '90*, pages 598–605, 1990.

[Jac92] D.W. Jacobs. Space efficient 3d model indexing. In *Proceedings CVPR '92*, pages 439–444, 1992.

[KJ86] T.F. Knoll and R.C. Jain. Recognizing partially visible objects using feature indexed hypothesis. *IEEE J. Rob. Aut.*, RA-2(1):3–13, 1986.

[Low85] D.G. Lowe. *Perceptual organization and visual recognition.* Kluwer Academic, Hingham, MA, 1985.

[LW88] Y. Lamdan and H.J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *Proceedings ICCV '88*, pages 238–249, 1988.

[MWS93] R. Mohan, D. Weinshall, and R.R. Sarukkai. 3d object recognition by indexing structural invariants from multiple views. In *Proceedings ICCV '93*, pages 264–268, 1993.

[PE90] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, 1990.

[PG89] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report 1140, MIT AI Lab, July 1989.

[RZMF92] C.A. Rothwell, A. Zisserman, J.L. Mundy, and D.A. Forsyth. Efficient model library access by projectively invariant indexing functions. In *Proceedings CVPR '92*, pages 109–114, 1992.

[SM92a] F. Stein and G. Medioni. Structural indexing: efficient 2-d object recognition. *IEEE Trans. PAMI*, 14(12):1198–1204, 1992.

[SM92b] F. Stein and G. Medioni. Structural indexing: efficient 3-d object recognition. *IEEE Trans. PAMI*, 14(2):125–145, 1992.

[TM87] D.W. Thompson and J.L. Mundy. Three-dimensional model matching from an unconstrained viewpoint. In *Proceedings Int. Conf. Rob. Aut.*, pages 208–220, 1987.