Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 907-912, 1996.

Connectionist Networks for Feature Indexing and Object Recognition

Clark F. Olson
Department of Computer Science
Cornell University
Ithaca, NY 14853
clarko@cs.cornell.edu

Abstract

Feature indexing techniques are promising for object recognition since they can quickly reduce the set of possible matches for a set of image features. This work exploits another property of such techniques. They have inherently parallel structure and connectionist network formulations are easy to develop. Once indexing has been performed, a voting scheme such as geometric hashing [10] can be used to generate object hypotheses in parallel. We describe a framework for the connectionist implementation of such indexing and recognition techniques. With sufficient processing elements, recognition can be performed in a small number of time steps. The number of processing elements necessary to achieve peak performance and the fan-in/fan-out required for the processing elements is examined. These techniques have been simulated on a conventional architecture with good results.

1 Introduction

Techniques that use sets of image features to index sets of model features that may match them (e.g. [3, 4, 6, 9, 10, 11, 12, 16]) are promising for object recognition because of their ability to quickly reduce the set of possible matches for a set of image features. In this work, we exploit another beneficial property of such indexing systems. They have inherently parallel structure and can be implemented using very fine grain parallelism. To this end, we give a framework for the connectionist implementation of feature set indexing and object recognition techniques. This framework uses a very large number of simple processing elements communicating in a fixed pattern. In such a connectionist framework, eliminating groups from consideration is not necessary for fast recognition, since the work on separate feature sets can be performed in parallel. Matches can simply be given varying levels of likelihood, as in Bayesian formulations of indexing [5, 15], which are then used to determine which objects are likely to be present in the image.

In this work, we assume that features have already been extracted from the image, and we use these features as the primary tool for recognition. Most feature detection techniques can be implemented without difficulty in a connectionist manner, but the examination of such techniques is outside the scope of this work.

Once features have been extracted from the image, recognition can be performed in a small number of time steps in this framework. When we have sufficient processing elements (with fixed fan-in and fan-out) to achieve maximum parallelization, the time required is $O(\log n)$. If the processing elements can broadcast a value on $O(n)$ dedicated connections in $O(1)$ time and maximize and sum $O(n)$ inputs on dedicated connections in $O(1)$ time, then the running time is $O(1)$.

If we use randomization concepts to limit the number of image feature sets that must be considered, maximum parallelization can be achieved with $O(mn^k)$ processing elements, where m is the number of features in the object model, n is the maximum number of image features at which high accuracy is maintained, and k is the number of matches between model and image features necessary to perform indexing. Note that this number of processing elements is not required, but processing time and/or accuracy will suffer as the number of processing elements is decreased. Alternatively, we can use $O(n^k + I)$ processing elements and achieve maximum parallelization, where I is the number of processing elements necessary to cover the indexing space.

Simulation of this system on sequential hardware has yielded good results.

It should be noted that parallel implementations

of indexing systems (particularly geometric hashing) have been previously studied (e.g., [2, 14, 15]), but these works have primarily considered general-purpose parallel-processing systems and have not achieved full parallelization since they retain sequential examination of image feature sets.

2 Object recognition using an election

Feature set indexing systems (e.g. [4, 10, 12, 16]) attempt to quickly determine which sets of object features could have projected to specific sets of image features under certain noise assumptions. Such systems consist of two phases. First, a preprocessing phase is performed in which an index table (or some other indexing data structure) is created. In this phase, the feature sets of some predetermined cardinality are stored in one or more locations in the index table according to parameters describing their geometry (and possibly other attributes). Ideally, these parameters are invariant to the transformations and projections that may be applied to the object in forming its image. Recognition is performed during the second phase. In this phase, the parameters of the sets of image features are used to index the sets of object features in the index table that may match them.

One of the uses of indexing has been in voting schemes like the geometric hashing system of Lamdan *et al.* [10]. This system uses indexing to determine the number of additional feature matches that are brought into alignment by the same transformation that aligns various basis sets of matches and is thus conceptually similar to the alignment method [8]. The geometric hashing method allows much of the work to be moved off-line and uses randomization to achieve improved efficiency. This shift of work off-line and the natural parallelism of the technique make such methods excellent for connectionist implementation, since we wish to keep our parallel processing elements as simple as possible.

Let k be the number of feature matches necessary to perform indexing. We'll call a set of k image features an *image group* and a set of k object features an *object group*. Similarly, a set of $k-1$ features will be called an *object basis* or an *image basis*. If we restrict ourselves to conventional indexing methods using point features, then $k = 2$ for two-dimensional translation only, $k = 3$ for two-dimensional translation, rotation, and scale, and $k = 4$ for full three dimensional transformations under weak-perspective.

The basic procedure for recognizing an object using an election is as follows:

1. Generate an index table storing the relevant information from each possible object group.
2. Detect the features in the image.
3. Choose a random image basis.
4. Form all image groups comprised of the image basis and one additional feature.
5. Index possible matches for each image group that was formed in step 4.
6. For each model group that is indexed, record a vote for the model basis that is matched to the random image basis.
7. If some model basis receives enough votes, then an object hypothesis has been found. Otherwise, Steps 3-7 are repeated until enough image bases have been examined to rule out the presence of the object in the image with high probability.

Step 1 is a preprocessing step and is performed prior to run-time. Step 2 is performed once per image. This step will not be examined in this paper. Steps 3-7 form the recognition phase of the algorithm. The criteria for stopping the procedure will be discussed further in Section 4.

3 Connectionist framework

Feature set indexing and object recognition by election translate naturally into a connectionist framework, where we can use massive parallelism to examine possible matches simultaneously. Our framework breaks the process into layers of processing elements that correspond to the following conceptual entities:

Layer 1: Image features.
Layer 2: Image feature groups.
Layer 3: Matches between image and model groups.
Layer 4: Matches between image and model bases.
Layer 5: Model hypotheses.

The first three layers perform the indexing and the final two layers perform the election. Here we assume that the preprocessing has already been performed and that the model group parameters have been loaded into the appropriate processing elements. The computations that take place in each of these layers and the communication pattern between them are as follows:

Layer 1: Image features. The parameters of the image features comprise the input layer. This layer

does not perform any computation. The parameters are simply fed to the appropriate processing elements in the following layer.

Layer 2: Image feature groups. The processing elements in the second layer correspond to the image groups (feature sets of size k). These processing elements generate the indexing parameters, which are typically easily computed functions of the feature parameters. Each of the processing elements in this layer receives input from the processing elements in the previous layer corresponding to the k features that make up the image group, and sends output to each of the group matches that contain this image group.

Layer 3: Group matches. The processing elements in the third layer represent matches between image groups and model groups. They take input from the processing element corresponding to a particular image group and they are tuned to the indexing parameters of a particular model group such that they output a high value if the indexing parameters of the input image group are close to the parameters of the model group. This value is ideally some likelihood function of the group match being correct. These processing elements send output to the processing elements corresponding to the basis matches contained in the group match.

Layer 4: Basis matches. The fourth layer performs the voting for each possible match between a model basis and an image basis. Each processing element at this layer receives output from each of the processing elements at the previous layer that matches the given image basis and model basis. The sum (or some other combination) of these likelihoods is used as the score for the basis match. A large score at this stage indicates that the basis match is likely to be correct.

Layer 5: Model hypotheses. The final layer determines which objects are present in the image by considering the information from the model bases for each object. These processing elements receive input from each of the basis matches for some object model and output the maximum of the inputs. If more than one instance of an object model may be present in the image, this layer may be omitted and the basis matches can be used to indicate where the objects are present in the image.

It should be noted that we do not select a subset of the possible matches for a particular image group to be indexed in this framework. Each possible match is assigned a score, which is then propagated through the network. This is similar to Bayesian formulations [5, 15], except that we do not eliminate matches in the connectionist framework even if they have a small likelihood of being correct in the Bayesian formulation. In addition, we note that the pattern of connections does not change for different images. We need only feed the feature parameters into the processing elements in the first layer.

4 Ideal number of processing elements

For any object that has at least some fraction, f, of its features appearing in the image, it is possible to achieve any fixed probability of examining a correct image basis while examining far less than each of the $O(n^{k-1})$ image bases. If we choose some number, x, of image bases with cardinality $k-1$ at random and examine only those bases, the probability of not examining any correct bases for any particular object is bounded approximately by:

$$p \leq \left(1 - \left(\frac{fm}{n}\right)^{k-1}\right)^{x}$$

This is true since the probability of any particular image feature being from the object is at least $\frac{fm}{n}$ in this case. If we require this probability to be less than some small constant, δ, we can solve for the minimum number of trials, x, necessary:

$$x \geq \frac{\ln \delta}{\ln\left(1 - \left(\frac{fm}{n}\right)^{k-1}\right)} \approx \left(\frac{n}{fm}\right)^{k-1} \ln \frac{1}{\delta}$$

We can thus examine $O(\frac{n^{k-1}}{m^{k-1}})$ image bases and achieve high accuracy (f and δ are constants). For each image basis, we consider each of the $\binom{m}{k-1}$ object bases as possible matches and for each possible match, we examine each of the $(n-k+1)(m-k+1)$ additional feature matches to determine if the match between the bases is correct. In total, we examine $O(\frac{n^{k-1}}{m^{k-1}}) \cdot O(m^{k-1}) \cdot O(mn) = O(mn^k)$ group matches and achieve probability $1-\delta$ that a correct image basis is examined for a particular object.

In the connectionist implementation, we can simply select the appropriate number of random image bases to examine in parallel and achieve probability $1-\delta$ of examining a correct image basis. The processing elements corresponding to the remainder of the matches are unnecessary. We thus gain maximum parallelization when we use $O(mn^k)$ processing elements in this case. Of course, in practice, the number of processing

Layer	Description	Number	Fan-In	Fan-Out
1	Image features	n	1	$\binom{n-1}{k-1}$
2	Image groups	$\binom{n}{k}$	k	$\sim \alpha m$
3	Group matches	$\sim \alpha m n^k$	1	1
4	Basis matches	$\sim \alpha n^{k-1}$	$\sim mn$	1
5	Models	1	$\sim \alpha n^{k-1}$	1

Table 1: The number of processing elements and fan-in/fan-out necessary per object model at each layer.

elements available will set some limit on the number of image features that can be handled with rate of failure δ. Performance will be good on images of this complexity or less and will degrade gracefully as the image complexity rises past this level. Alternatively, we can let processing elements perform more than one task to maintain a high performance at the cost of additional processing time.

Note that perceptual organization can be used to determine sets of features that are likely to come from the same object. In this case, we can use these sets as the image bases that we examine. This will not only result in better performance in finding objects, but it will allow the number of image bases that we need to examine to be reduced.

We can now consider the ideal number of processing elements at each level, as well as the fan-in and fan-out required for each processing element. We need to examine approximately αn^{k-1} basis matches (where $\alpha = \frac{\ln \frac{1}{\delta}}{f^{k-1}}$, and thus is constant when f, δ, and k are set), since for each image basis we must examine each of the $(k-1)!$ permutations of each of the $\binom{m}{k-1}$ model bases that may match it. This implies that the number of group matches that must be examined is approximately $\alpha(m-k+1)(n-k+1)n^{k-1}$. The required fan-in and fan-out of the processing elements in each level can be determined from the connection pattern described in the previous section. See Table 1. If we don't have individual processing elements with this fan-in/fan-out capability, we can build up the fan-in/fan-out using a tree of simple elements.

It is possible to reduce the ideal number of processing elements at the group match stage by combining processing elements for matches that share the same (or very similar) indexing parameters. These processing elements perform the same function, since they are matching image groups against the same set of parameters. This would then require $O(n^k + I)$ processing elements, where I is the number of processing elements necessary to cover the space of indexing parameters.

This analysis of the ideal number of processing elements is on a per object basis. As the number of objects in the database increases, the ideal number of processing elements increases linearly. Alternatively, we can process the objects sequentially, using the same processing elements for each object.

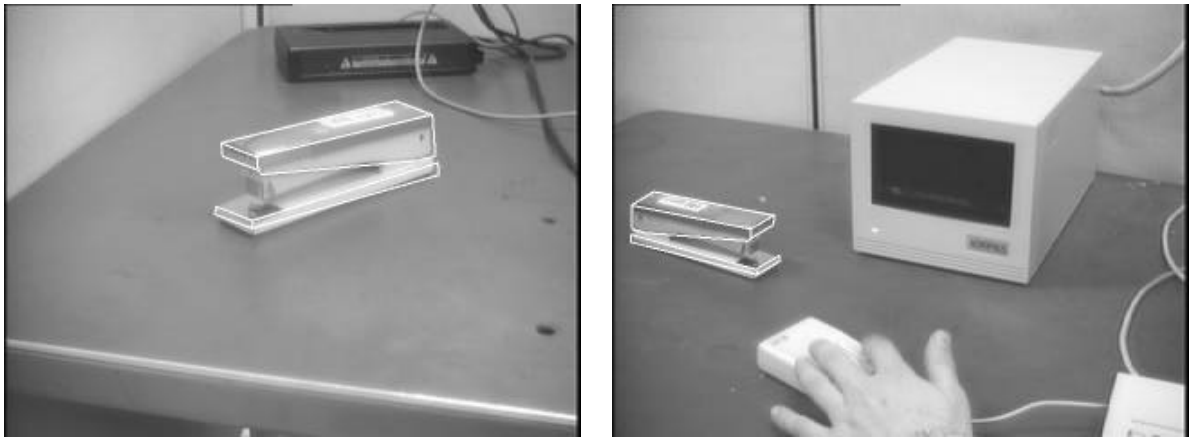
5 Results

We describe techniques for implementing several indexing schemes in this framework in [13]. These techniques have been simulated using a conventional computer architecture. In these experiments, we use a variant of the transformation metric described by Weinshall and Basri [17] to perform an indexing-like function. These techniques determine how well a set of point correspondences can be brought into alignment by a rigid transformation under weak-perspective and, in this case, $k = 4$. The transformation metric is:

$$N_{tr} = \frac{1}{2}(x^T Bx + y^T By - 2\sqrt{x^T Bx \cdot y^t By - (x^T By)^2})$$

where x and y are vectors of the x- and y-coordinates of the image points after translating them such that their center of mass is at the origin and B is what Basri and Weinshall call the *characteristic matrix* of the model points. The transformation metric, N_{tr}, will yield a distance of zero if the points can be brought into alignment exactly by a rigid transformation and otherwise measures how far the optimal affine transformation is from rigid. This value is normalized and inverted to yield an appropriate score for each group match. See [13] for additional details.

In experiments on random object groups that were projected using the perspective projection with bounded noise added (error radius $\epsilon = 1.0$), correct matches received an average score of .701, while incorrect matches received an average score of .0036. This procedure thus performs well in discriminating between correct and incorrect matches.



(a) (b)

Figure 1: Two examples recognizing a three-dimensional object in a real image.

The entire system has been simulated on recognition problems consisting of 20 models points projected onto the image using the perspective projection and 30 noise points. Once again bounded noise was added ($\epsilon = 1.0$) to the locations of the projected object points. In these experiments, an average of 6.16 correct basis matches were found for each correct object, thus objects were detected when they were present in the image. In addition, out of the 1.3 million incorrect basis matches examined per object, an average of 3.13 false positives were found that required verification to discard (a false alarm rate of $2.41 \cdot 10^{-6}$).

Figure 1 shows the results of using these techniques to recognize a stapler in two real images. In this case, feature points were determined using an interest operator [7] and their locations were used in the recognition process.

6 Discussion

Let's first consider the running time required for recognition in this framework. Note that we must have $n \geq fm$ to recognize a model in an image, so we can assume $m = O(n)$. If we have sufficient processing elements with $O(n)$ fan-in and fan-out capabilities[1], then the running time is $O(1)$, since the computation required at each layer can be computed in constant time and there are a constant number of layers. (The $O(n^{k-1})$ fan-in and fan-out capabilities can be simulated using a tree of processing elements of a constant height proportional to k.) For processing elements with limited fan-in and fan-out capabilities, we can chain processing elements in a tree with $O(\log n)$ height to provide the necessary capability. Thus, in this case, the running time of the system is $O(\log n)$. Such a system would be fast in practice, in either case, since the computations performed by each processing element are simple.

In the framework described above, we have assumed that the necessary indexing parameters are loaded into the correct processing elements prior to recognition time. An interesting possibility to consider is whether these parameters could be learned in a supervised or unsupervised manner through the examination of examples.

Note that this system does not use weights associated with the connections. Given the number of object and image features that we wish to handle, the communication pattern is completely fixed, regardless of the specific object and image. The parameters that vary in this system are the indexing parameters stored at the processing elements corresponding to group matches. These parameters could be trained by a learning algorithm.

While it is conceivable that the indexing parameters for an entire object could be learned concurrently, a superior learning strategy is to train each of the processing elements corresponding to the group matches separately. Beis and Lowe [1] describe such a system using radial basis functions or Parzen windows to learn indexing functions for three-dimensional objects. At this time, we have made no attempt to train a system

[1] This assumes the ability to find the sum and maximum of the inputs on these dedicated connections in $O(1)$ time.

to learn indexing parameters in this framework.

7 Summary

This paper has considered connectionist methods for performing feature set indexing and object recognition. We first summarized the basic algorithm for performing object recognition using an election. Such systems use indexing to determine which sets of model features could have projected to the sets of image features that contain a basis set of features. Votes are recorded for each of the basis sets of model features that are indexed and such basis sets that accumulate many votes are considered recognition hypotheses.

We then described a connectionist framework for the implementation of indexing and voting techniques to perform object recognition. This framework uses a large number of simple processing elements performing in parallel and communicating on a fixed connection network. The ideal number of processing elements was determined to be $O(mn^k)$, where m is the number of model features, n is the number of image features, and k is the number of feature matches necessary to perform indexing. When the ideal number of processing elements are present, the system can perform object recognition in $O(\log n)$ time, even when the fan-in/fan-out of the processing elements is limited. If the processing elements can broadcast on $O(n)$ connections and select the maximum and sum of $O(n)$ inputs on dedicated connections in $O(1)$ time, then object recognition can be performed in $O(1)$ time.

Finally, we have described the results of simulating the performance of these techniques on a conventional architecture using real and synthetic images. These simulations used a transformation metric to determine which sets of features were good matches and the propagation of these values through the network yielded good recognition performance.

References

[1] J. S. Beis and D. G. Lowe. Learning indexing functions for 3-d model-based object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 275–280, 1994.

[2] O. Bourdon and G. Medioni. Object recognition using geometric hashing on the connection machine. In *Proceedings of the IAPR International Conference on Pattern Recognition*, volume 2, pages 596–600, 1990.

[3] A. Califano and R. Mohan. Multidimensional indexing for recognizing visual shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):373–392, April 1994.

[4] D. T. Clemens and D. W. Jacobs. Space and time bounds on indexing 3-d models from 2-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1007–1017, October 1991.

[5] M. Costa, R. Haralick, T. Phillips, and L. Shapiro. Optimal affine-invariant point matching. In *Applications of Artificial Intelligence VII, Proc. SPIE 1095*, pages 515–530, 1989.

[6] P. J. Flynn. 3d object recognition using invariant feature indexing of interpretation tables. *CVGIP: Image Understanding*, 55(2):119–129, March 1992.

[7] W. Förstner. Image matching. Chapter 16 of *Computer and Robot Vision*, Vol. II, by R. Haralick and L. Shapiro, Addison-Wesley, 1993.

[8] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[9] T. F. Knoll and R. C. Jain. Recognizing partially visible objects using feature indexed hypotheses. *IEEE Journal of Robotics and Automation*, 2(1):3–13, 1986.

[10] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6(5):578–589, October 1990.

[11] R. Mehrotra and W. I. Grosky. Shape matching utilizing indexed hypotheses generation and testing. *IEEE Transactions on Robotics and Automation*, 5(1):70–77, 1989.

[12] C. F. Olson. Probabilistic indexing for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):518–522, May 1995.

[13] C. F. Olson. Connectionist networks for feature indexing and object recognition. Technical Report 96-1568, Department of Computer Science, Cornell University, 1996.

[14] I. Rigoutsos and R. Hummel. Massively parallel model matching: Geometric hashing on the connection machine. *Computer*, pages 33–41, February 1992.

[15] I. Rigoutsos and R. Hummel. Distributed Bayesian object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 180–186, 1993.

[16] D. Weinshall. Model-based invariants for 3-d vision. *International Journal of Computer Vision*, 10(1):27–42, 1993.

[17] D. Weinshall and R. Basri. Distance metric between 3d models and 2d images for recognition and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 220–225, 1993.