Recognition Algorithms for the Connection Machine

Anita M. Flynn and John G. Harris

MIT Artificial Intelligence Laboratory

ABSTRACT

This paper describes an object recognition algorithm both on a sequential machine and on a SIMD parallel processor such as the MIT Connection Machine. The parallel version is shown to run three to four orders of magnitude faster than the sequential version.

I. INTRODUCTION

Many problems in early vision appear to be inherently local and exploitable on parallel computer architectures. Can algorithms for higher perceptual functions also be mapped onto the coming wave of massively parallel machines? This paper examines the process of recognising an unknown object and matching it to a data base model given only sparse sensory data points. The algorithm presently used on a sequential machine is first explained, and then various algorithms for parallel computation are explored. Tradeoffs in space-time efficiency are discussed in terms of implementation on a Connection Machine [5].

II. SEQUENTIAL ALGORITHM

The problem considered here is that of recognizing an object and determining its position and orientation [2, 3]. Sensory data can be acquired from a variety of modalities, such as tactile sensors, or laser or sonar rangefinders, but the algorithm is independent of the specific type of sensor used. It assumes that only a few data points are available, and uses local constraints intrinsic to the shape of the object to prune the search space of possible interpretations.

As an example, suppose we had a rectangularly shaped figure with opposing sides of length three and four, Figure 1. The object is lying on a table, and we also have a description of it in a data base. Imagine a three fingered robot hand equipped with tactile sensors touching the object at the points marked P_1, P_2 and P_3. From this data only, we would like to determine whether or not these three points could possibly be resting on the faces of our test object, and if so, where the object lies with respect to the robot hand.

Support for the laboratory's Artificial Intelligence rcscaich is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505. the Office of Naval Research under contract number N00014-77-C-0389, and the System Development Foundation.

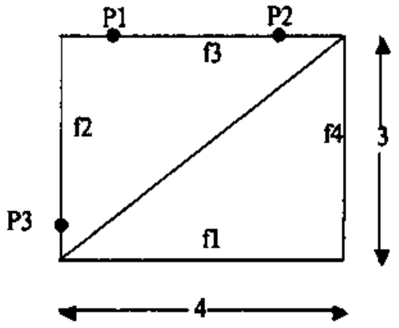

Figure 1:A four sided object with three data points.

A tree of possible matchings of points to faces is shown in Figure 2. The first level of the tree represents the fact that P_1 could lie on one of the four faces of the object. Similarly, the second level shows that P_2 could lie on any one of four faces given P_1's assignment For three data points, there are 64 possible matchings of data points to object faces. Each possibility must be checked to see if the object can be rotated in some way, so that the points actually do fall on the faces of the object

For an object with n faces and k data points, there are n^k possible rotations to check. That involves quite a lot of computation for a multi-faceted object The basic idea of the

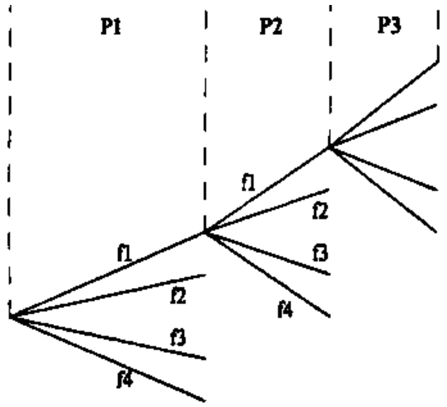

Figure 2:The Interpretation Tree

algorithm described in [2], is to prune the interpretation tree so that fewer rotations, or model tests, need be performed.

One constraint used to narrow the search space is the distance constraint between faces of the object. For the example object, we know that the minimum distance between face2 and face3 is zero and the maximum distance is five. Figure 3 shows the range of possible distances between pairs of faces on our lest object The first number is the minimum distance and the second number is the maximum distance.

	f1	f2	f3	f4
f1	04	05	35	05
f2	05	03	05	45
f3	35	05	04	05
f4	05	45	05	03

Figure 3:Distance Ranges Between Objects

As can be seen, any experiment which measures the distance between P_3 and P_2 to to be 6.2, renders inconsistent any branch of the tree which assigns P_3P_2 to f_2f_3. or P_2 $P_3$to f_2f_3. Thus, those branches of the tree can be pruned and need never be model tested.

To continue the example, suppose we measure the distances between all possible pairs of our three data points and determine that the distances are: $P_2P_1 = 3.7$, $P_2P_3 = 4.5$, and $P_3P_1 = 2$. By consulting our table, we find that P_1P_2 cannot both be assigned to f_2 or both to f_4. It also cannot be the case that P_1P_2 is consistent with an assignment to f_2f_4 (or vice versa).

The second measurement is inconsistent with a pairing of P_2P_3 to f_1f_1, f_2f_2, f_3f_3, or to f_4f_4. The third measurement similarly cannot assign P_1P_3 to f_3f_1, f_1f_3, f_4f_2, or to f_2f_4. The number of possible interpretations of the data is reduced from 64 to 33, as shown in Figure 5. Other constraints can be used to further prune the tree before model testing need be done.

At present, this algorithm, running on a sequential machine, is set up to do a depth first search, pruning when it can. For an object with about a dozen sides and several data points, the program takes on the order of a few seconds. Recently, an algorithm has been proposed by Grimson and Lozano-Perez which allows the search space to be explored in parallel on a Connection Machine [4]. After outlining that algorithm, we describe a new one that captures more parallelism in the problem and therefore runs faster.

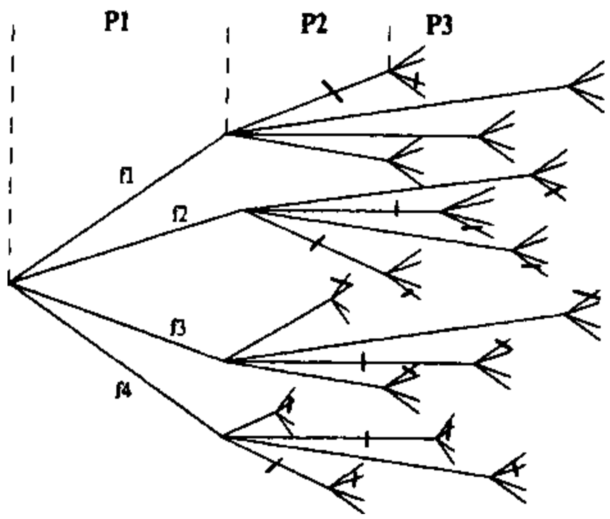

Figure 4:Possible interpretations pruned.

III. PARALLEL BIT ARRAY ALGORITHM

The MIT Connection Machine is a massively parallel machine, containing 256,000 processors connected together both in two-dimensional nearest-neighbor fashion and also through a global routing mechanism whereby any processor can talk to any other processor arbitrarily. Each processor contains 4K bits of memory.

The algorithm proposed in [4] exploits the router feature of the Connection Machine because there is no inherent locality to the recognition problem. The interpretation tree is a data abstraction, in contrast to the pixel-based locality characteristic of early vision problems.

This algorithm generates and prunes new levels of the tree in parallel by having Connection Machine processors hold bit arrays which represent consistent pairings of points to faces. Figure 6 shows three processors holding the consistency arrays for our previous example. The zeros represent pairings of data points with the data base model that were inconsistent Imagine ones in all the empty boxes.

Generation and pruning of new branches in the tree is done in parallel when adding the constraint imposed by a new data point. The algorithm for updating, say P_1P_2 with respect to P_3, is to do a data base merge on the P_1P_3 and P_2P_3 consistency arrays and then AND the resulting array back into the P_1P_2 array. A data base merge is similar to a matrix multiply except that one array is transposed and the logical operations of AND and OR are used instead of multiplications and additions. What all this accomplishes, is a propagation of constraints and a generation of a new level of the tree. For an n-sided object and k data points, there

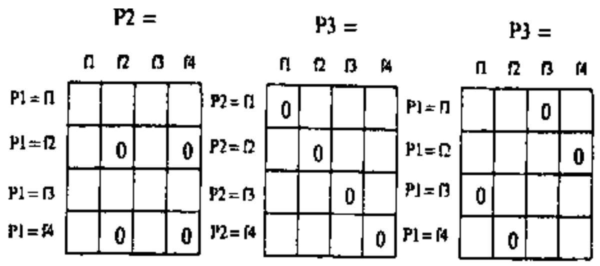

Figure 5:Consistency Arrays

will be k -k nxn arrays of possible pairing of points to faces. Each one of these arrays (in this case, six) is updated in parallel but each of these processors is multiplying two nxn matrices which is an order n^3 operation.

Unfortunately, the norm in most recognition problems that use a robot hand and tactile sensors is for n (the number of faces of the object) to be quite large, while k (the number of data points) is usually small. For an object with n = 100, and assuming a conservative 1 μsecond clock, it has been determined that this algorithm would take roughly two seconds. Our parallel algorithm here isn't buying us much. It would be much better to be doing n operations in parallel, with each processor taking k -k steps (six, in our example here).

IV. STATIC ALGORITHM

If we use n^k hardware we can do the computation in nearly constant time. We allocate a separate processor to represent each element in a three-dimensional consistency array of possible pairings of $P_1P_2P_3$ to appropriate faces of the object This representation, shown in Figure 4, which can be extended to arbitrary dimensions for more data points, does away with the need for database merges or constraint propagation.

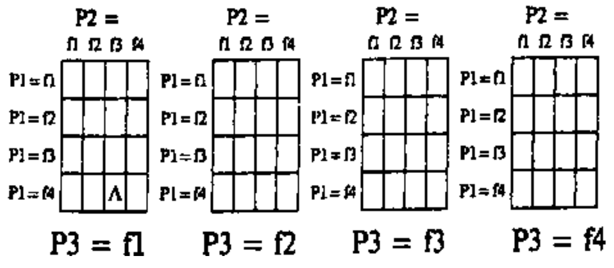

Figure 6:Slices of a Three Dimensional Consistency Array

Each processor contains in its own state variables the appropriate part of the distance constraint table for the faces it represents. The processor marked A, because of its location in the array, is to mark the truth or falsehood of whether or not the pair $P_1P_2P_3$ ↄ $f_4f_3f_1$ is consistent. Before run time, it must have in its state variables the appropriate distance constraints between these faces. At run time, as each distance measurement is broadcast to all the CM processors, each processor checks if that measurement falls within the allowed range.

All the processors' flag bits have previously been initialized to one. If any broadcast measurement does not fall in the specified range, the flag is ANDed with zero. After all data is broadcast any processors still marked one are valid interpretations. The time to do this is order k^2 since there are $[k^2-k]/2$ distance measurements to broadcast. Assuming 16 bit precision, it would take 16 cycles for a processor to read in a broadcast measurement Comparison with the upper and lower ranges would lake 2x16 cycles, plus 2 cycles to AND the results with the flag. These 50 cycles are repeated $[k^2-k]/2$ limes. The number of processors needed is n^k and memory per processor is $[k^2-k]$ registers plus 1 bit for flags. For 3 data points and a 1 /xsecond cycle time, this process could be completed in 150 /iseconds, using n^3 processors, where each processor needs six registers and one bit of memory. This is roughly four orders of magnitude faster than the sequential algorithm on a Lisp Machine.

For some objects, with n large, there may not be enough processors available. State variables for several processors can then be grouped into one. where that processor sequentially checks its constraints. If G processors are grouped into one processor, then the time to run the algorithm becomes order Gk^2, the number of processors needed will be n^k/G, and the memory per processor becomes Gk(k-l) 16-bit registers plus G bits for flags.

In general, we'd like to have many objects in the data base. If there are enough idle processors, we can put all the objects in the CM array and again do the search in order [k(k-l)]/2 time. If there aren't enough processors, different objects can be worked on sequentially, or G processors can be grouped into one so that all objects fit into the CM array. Either way, it takes the same amount of time and space per processor. For 250 objects, each with 10 sides, and 3 data measurements, we would need 250,000 processors. The recognition algorithm would still run in 150 jiseconds and each processor would again need six registers and one flag bit However, we still aren't fully utilizing our machine. Only 100 bits out of the 4K available is in use in each processor. If we put the data for 40 more objects in each processor, wc will have 40 times as many objects in the data base. The search will then be 40 times as long. This means that we can recognize an object from a data base of 10,000 objects, each with 10 sides in 6ms.

All of this assumes that the constraint table has been loaded into the processor array at compile time. In actuality, the' time to load the table is much longer than the time to recognize the object Even so, for 250 ten-sided objects, one-time loading takes only half of a second.

A better way of getting the constraint information into the processors is not to compute the table offline and then load it in, but to have each processor compute the appropriate information. This way the table (which has to be computed anyway) is now computed in parallel. In addition, the loadup cost is significantly reduced. We simply have to broadcast the coordinates of the vertices of the faces of the object Broadcasting the vertices takes order n time, whereas our earlier scheme of broadcasting an already computed distance constraint table, would take order n^2 time because we'd have to broadcast to each processor in the array.

After having received the broadcast vertices, each processor will calculate the distances between the faces appropriate for that processor and the corresponding maximum and minimum. If there are enough processors available, this computation of the distance constraint table is done in a time independent of n. The time is a small constant, the number of cycles it lakes to calculate the maximum and minimum of the four possible distances between any two sides.

We've taken an algorithm that would be unthinkable on a sequential machine and brute force implemented it on a SIMD array. Instead of generating and pruning the interpretation tree level by level, we have simply assigned processors to all the leaves of the expanded tree and actually completed one pruning step in constant time. The Connection Machine is used here as if it were a content addressable processor, and no use is made of the router.

We call this method the static algorithm, because all possible processors in the Connection Machine have been allocated before we start the search. This means that the machine must be big enough for the problem. Specifically, the number of processors needed, $[n^k]/G$, must be less than 256,000, and the amount of memory per processor required, $16Gk(k-l) + G$, must be less than 4K bits. When using tactile sensors, the number of data points, k, is small and this static algorithm is adequate.

V. DYNAMIC ALGORITHM

However, when using laser rangefinders, more data points are generated, and k often becomes as large as 100. Furthermore, when the objects are more complicated, they may have as many as 50 faces. The problem quickly gets too big even for our 256,000 processor machine. The solution is to use a dynamic algorithm which generates enough levels of the tree to fill the machine, does a pruning step, generates new levels of the tree to again fill the machine, and repeats. These ideas are the subject of future research.

This dynamic algorithm requires that the machine be able to find unused processors to represent the new branches of the tree. The machine must also be able to deallocate processors that no longer represent consistent pairings. The deallocated processors are then able to be reused for future pruning steps. The Connection Machine, with its global communication ability is able to support these mechanisms [6,1]. A strictly content addressable processor could not support this dynamic pruning algorithm.

VI SUMMARY

A well-established sequential algorithm for object recognition was explained and fast Connection Machine algorithms were devised. First, a static parallel algorithm was discussed, in which the entire tree of possible interpretations is loaded into the Connection Machine before run time. The search is then done in one step and therefore in constant time.

For problems which are too large to fit in the array, a dynamic algorithm was devised. In this algorithm, the Connection Machine is loaded with as many levels of the tree as can fit Then a pruning is done in parallel as in the static algorithm. However, processors that still represent consistent pairings must find unused processors which are assigned the new branches of the tree. This is similar to the way the sequential algorithm, first described, prunes the search space. Processors that are pruned are deallocated and reused on the next iteration. The router mechanism of the Connection Machine, which is not used in the static algorithm, is necessary here to support dynamically allocating and deallocating processors.

ACKNOWLEDGMENTS

This work was aided by contributions from Tom Knight, Eric Grimson and Phil Agre.

References

[1]
Christman D. P.
Programming the Connection Machine.
Master's thesis, MIT, January, 1984.

[2]
Gaston, P.C.. Lozano-Perez, T.
Tactile Recognition and Localization Using Object Models: The Case of Polyhedra on a Plane.
Technical Report A.I. Memo 705, MIT, March, 1983.

[3]
Grimson, W.E.L, Lozano-Perez, T.
Model-Based Recognition and Localization From Sparse Range or Tactile Data.
Technical Report A.I. Memo 738, MIT, August, 1983.

[4]
Grimson, W.EL., and Lozano-Perez, T.
Parallel Algorithms for Computer Vision.
In Winston, P. H.. editor, *A Research Proposal to DARPA,* chapter Parallel Recognition Algorithms. MIT, 1984.

[5]
W. Daniel Hillis.
The Connection Machine.
Technical Report A.I. Memo 646, Massachusetts Institute of Technology, September, 1981.

[6]
Kramer, G. A.
Brute Force and Complexity Management: Two Approaches to Digital Test Generation.
Master's thesis. MIT, May, 1984.