Pattern Recognition Letters 13 (1992) 433-441
North-Holland

June 1992

Parallelizing object recognition on the hypercube

Suchendra M. Bhandarkar*

Department of Computer Science, 415 Graduate Studies Research Center, University of Georgia, Athens, GA 30602, USA

Received 27 September 1991

Abstract

Bhandarkar, S.M., Parallelizing object recognition on the hypercube, Pattern Recognition Letters 13 (1992) 433-441.

In this paper a parallel Interpretation Tree search algorithm for object recognition on the Intel iPSC/2 hypercube multicomputer is presented. The input to the algorithm is a set of sensory measurements consisting of sensor locations and the directions of the surface normal. Such data is typical of sparse range or tactile sensors. At this time the algorithm is capable of handling two-dimensional objects or three-dimensional objects located on a plane with all sensory measurements parallel to the plane. The objects are typically those which a robot would encounter in an industrial scene during the process of automated inspection or automated assembly. The algorithm is able to handle polyhedra or objects which can be approximated as piecewise combination of polyhedra. Both single- and multiple-object scenes are considered. Three strategies for mapping the Interpretation Tree search process on the hypercube are considered. The performance of the algorithm for each of these mapping strategies is examined in terms of parameters such as speed-up, processor utilization, inter-processor communication overhead and load sharing between processors. It is shown that the requirement for uniform load sharing leads to increased inter-processor communication. Experimental results are presented and future directions are outlined.

1. Introduction

The problem of 3-D object recognition is one of great combinatorial complexity. The problem is essentially one of labeling objects in the scene as instances of prestored object models. In this sense, the problem of 3-D object recognition can be looked upon as a *consistent labeling problem* or a *constraint satisfaction problem. Recognition-via-localization* is a popular paradigm in 3-D object recognition. In this paradigm, constraints arising from matches of local geometric features are propagated until a consistent set of constraints has been determined. A set of consistent constraints is deemed to be a *consistent scene interpretation*. A consistent scene interpretation contains both, the identity and the location of objects in the scene.

Interpretation Tree (I.T.) search is a commonly used constraint propagation/constraint satisfaction technique for recognition-via-localization [1, 2, 3]. Each successful match of a scene feature with a model feature is represented by a node in the I.T. The matches are subject to constraints based on local distance and angle measurements. A consistent set of matches (or a scene interpretation) is represented by a path in the I.T. A match which is found to be inconsistent causes the corresponding path in the I.T. to be pruned. The control structure of the I.T. search for a conventional SISD (uniprocessor) architecture is that of *hypothesize-and-test* with *backtracking*. The I.T. search is found to

* This research was partially funded by a Faculty Research Grant from the University of Georgia Research Foundation Inc., Athens, GA 30602, USA.

be fairly successful in single-object scenes where it has been analyzed to have quadratic complexity in the number of scene and model features [4]. In multiple-object scenes with occlusion and clutter, however, the combinatorial explosion of the search space of possible scene interpretations causes a severe degradation in the performance of the I.T. search. The I.T. search has been analyzed to have exponential complexity in the number of scene and model features in such cases [5]. The degradation of the I.T. search is manifested as loss of accuracy in recognition and localization and increased recognition and localization time. Parallel implementation of the interpretation tree search on a suitable parallel architecture is one way of speeding up the search process.

In this paper a parallel search algorithm on the Intel iPSC/2 hypercube is presented. The Intel iPSC/2 hypercube is a MIMD architecture with distributed memory and is based on the message-passing paradigm. The iPSC/2 at the University of Georgia is a 32 node hypercube connected to a host machine. The input to the algorithm is a set of sensory measurements consisting of sensor locations in the scene coordinate frame of reference and the directions of the surface normal. Such data is typical of sparse range or tactile sensors. The algorithm is capable of handling two-dimensional objects or three-dimensional objects located on a plane with all sensory measurements parallel to the plane. The objects are typically those which a robot would encounter in an industrial scene during the process of automated inspection or automated assembly. At this time, the algorithm is constrained to handle polyhedra or objects which can be approximated as piecewise combination of polyhedra. Both single- and multiple-object scenes are considered. Three strategies for mapping the I.T. search process on the hypercube are considered. The performance of the algorithm for each of these mapping strategies is examined in terms of parameters such as speed-up, processor utilization, inter-processor communication overhead and load sharing between processors. It is shown that the requirement for uniform load sharing leads to increased inter-processor communication. Experimental results are presented and future directions are outlined.

2. Problem description

The problem discussed in this paper is that of recognizing and localizing 2-D objects or 3-D objects located on a plane with all sensory measurements parallel to the plane. The input sensory data is assumed to be sparse range data as is typical of tactile or spot range sensors. The object(s) in the scene are recognized and localized by matching the sensory data with the prestored models of the objects in the memory of the computer. An issue of particular interest in this paper is how a coarse-grained, distributed memory MIMD architecture such as the Intel iPSC/2 hypercube could be used to parallelize the I.T. search process.

The two-dimensional objects considered in this paper are modeled as N-sided polygons. Each sensory measurement is in the form of a 4-tuple (x, y, n_x, n_y) where the coordinates (x, y) denote the sensor location and the coordinates (n_x, n_y) denote the sensed surface or side normal at point (x, y). The sensory measurements are in the scene coordinate frame of refeıence. The I.T. search process determines if a given sensory measurement is consistent with a given side (face) of the object model, i.e., it pairs a sensory measurement with a given side (face) of the object model. Angle and distance measurements are used to discard inconsistent matches. Given M sensory measurements numbered from 0 to $M-1$, the I.T. for an object model with N sides numbered from 0 to $N-1$ is generated by matching a sensor measurement $s_i, 0 \leqslant i \leqslant M-1$, with each of the model sides $m_j, 0 \leqslant j \leqslant N-1$. Each match (s_i, m_j) is represented by a node in the Interpretation Tree as shown in Figure 1.

The branching factor of the tree is N where the children of node (s_i, m_j) are the nodes

$$(s_{i+1}, m_0), (s_{i+1}, m_1), \dots (s_{i+1}, m_{N-1}).$$

If the root node is at level 0, then the resultant Interpretation Tree has M levels and N^M leaf nodes. A consistent set of pairings of the sensory measurements with the object model sides is called a *valid scene interpretation* and is represented by a path in the Interpretation Tree from the root node to the leaf node. A valid scene interpretation is also used to compute a global pose of the object which

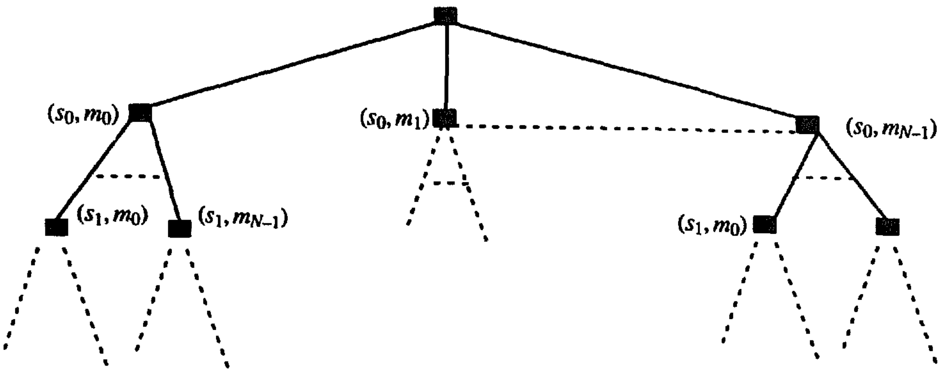

Figure 1. Interpretation tree.

would place the model in registration with the object in the scene.

2.1. Model data

The data structures for the representation of the object models are as follows:

(i) A two-dimensional array *dist*[ ][ ] of size $N*N$ is used to store the maximum and the minimum distances between each side in the object model. For $i<j$, *dist*[i][j] contains the minimum distance between sides i and j in the object model. For all $i>j$, *dist*[i][j] contains the maximum distance between sides i and j in the object model. *dist*[i][i] contains the length of side i.

(ii) A two-dimensional array *angle*[ ][ ] stores the angle between two sides in the object model. For all $i<j$, *angle*[i][j] contains the angle between sides i and j in the object model. The other entries in the array are zero.

(iii) The one-dimensional arrays $x0$[ ] and $x1$[ ] contain the x coordinates of the endpoints of the sides of the object model. For example, $x0[i]$ and $x1[i]$ are the x coordinates of the endpoints of side i in the object model. Similarly the one-dimensional arrays $y0$[ ] and $y1$[ ] contain the y coordinates of the endpoints of the sides of the object model.

(iv) The one-dimensional arrays *nx*[ ] and *ny*[ ] contain the x and y coordinates of the normal vector to the sides (faces) of the object model. For example, *nx*[i] and *ny*[i] contain the x and y coordinates of the normal to side (face) i in the object model.

A side (face) m_i in the object model is specified by the 6-tuple

$$(x0[i],\ y0[i],\ x1[i],\ y1[i],\ nx[i],\ ny[i])$$

in the model coordinate frame of reference.

2.2. Scene feature data

The data structures used to represent the sensory data are as described below.

(i) The two-dimensional array *initdist*[ ][ ] contains the distance between two sensory measurements. For $i<j$, *initdist*[i][j] represents the distance between sensory measurements s_i and s_j.

(ii) The two-dimensional array *initangle*[ ][ ] contains the angle between two sensory measurements. For $i<j$, *initangle*[i][j] represents the angle between sensory measurements s_i and s_j.

(iii) The one-dimensional arrays *sx*[ ] and *sy*[ ] denote the x and y coordinates of the sensory measurement (sensor position). For example *sx*[i] and *sy*[i] denote the x and y coordinates of the ith sensory measurement s_i.

(iv) The one-dimensional arrays *nsx*[ ] and *nsy*[ ] denote the x and y coordinates of the side (face) normal as sensed by the sensor. For example, *nsx*[i] and *nsy*[i] denote the x and y coordinates of the normal sensed by the sensory measurement s_i.

The sensory measurement s_i is specified by the 4-tuple

$$(sx[i],\ sy[i],\ nsx[i],\ nsy[i])$$

in the scene coordinate frame of reference.

2.3. Pruning constraints

At each stage in the I.T. search process a sensory measurement is matched with a side (face) in the object model. An *interpretation* is a set of sensory measurement-object model side (face) pairs $\{(s_i, m_j)\}$ where $0 \leqslant i \leqslant M-1$ and $0 \leqslant j \leqslant N-1$. An interpretation is said to be *valid* if any two pairs (s_i, m_k) and (s_j, m_l) within the interpretation satisfy the following three constraints.

(i) *Unique Interpretation Constraint.* The two pairs (s_i, m_k) and (s_j, m_l) are said to satisfy the unique interpretation constraint if $i \neq j$. That is to say, a sensory measurement cannot be assigned to more than one side in a single interpretation although more than one sensory measurement can be assigned to a single side.

(ii) *Distance Constraint.* Given two pairs (s_i, m_k) and (s_j, m_l) within an interpretation, if $d(s_i, s_j)$ denotes the distance between the sensors, then

$$d_{\min}(m_k, m_l) \leqslant d(s_i, s_j) \leqslant d_{\max}(m_k, m_l)$$

where $d_{\min}(m_k, m_l)$ and $d_{\max}(m_k, m_l)$ denote the maximum and minimum distances respectively between the sides m_k and m_l on the object model.

(iii) *Angle Constraint.* Given two pairs (s_i, m_k) and (s_j, m_l) within an interpretation, if $\theta(s_i, s_j)$ is the angle between the surface normals sensed by the sensors s_i and s_j and $\theta(m_k, m_l)$ the angle between the normals to the sides m_k and m_l on the object model, then $\theta(s_i, s_j) = \theta(m_k, m_l)$ within the bounds of sensing error.

A *partial* interpretation is a set of valid sensor-side pairs

$$\{(s_i, m_j) \mid 0 \leqslant i \leqslant L,\ L < M-1,\ 0 \leqslant j \leqslant N-1\}.$$

A *complete* interpretation is a set of valid sensor-side pairs

$$\{(s_i, m_j) \mid 0 \leqslant i \leqslant M-1,\ 0 \leqslant j \leqslant N-1\}$$

and wherein there exists a pair (s_{M-1}, m_j) for some value of j. At level L where $L < M$ in the I.T., a partial interpretation $\{(s_i, m_j) \mid 0 \leqslant i \leqslant L-1, 0 \leqslant j \leqslant N-1\}$ is extended by the pair (s_L, m_k) for some value of k such that $0 < k \leqslant N-1$. If the result of extending the partial interpretation at level L is a partial interpretation at level $L+1$ $\{(s_i, m_j) \mid 0 \leqslant i \leqslant L, 0 \leqslant j \leqslant N-1\}$ which satisfies all the aforementioned constraints then the corresponding path in the I.T. is explored further else it is pruned. If the partial interpretation at level $L+1$ $\{(s_i, m_j) \mid 0 \leqslant i \leqslant L, 0 \leqslant j \leqslant N-1\}$ is such that it satisfies all the aforementioned constraints and if furthermore, $L+1 = M$ then a leaf node in the I.T. has been encountered which makes the partial interpretation at level $L+1$ a complete interpretation. All the complete interpretations in a I.T. can be determined using a *hypothesize-and-test* with backtracking search procedure on a conventional uniprocessor architecture.

2.4. Localization

Given a complete scene interpretation at the end of the I.T. search, the next step is to determine the transformation from the model coordinate frame of reference to the scene coordinate of reference which will place the object model in registration with the object in the scene. This process of determining the exact location and orientation of the object in the scene is referred to as localization. Given two sensory measurement-object model side pairs (s_i, m_k) and (s_j, m_l), the exact location of each of the sensory measurements on the corresponding model sides (in the model coordinate frame of reference) is determined using the distance constraint. Let (m_x, m_y, n_x, n_y) be the position of the sensory measurement s_i on the model side m_k in the model coordinate frame of reference. Let (s_x, s_y, n'_x, n'_y) be the position of s_i in the scene coordinate frame of reference. The angle of rotation θ is determined by the equation:

$$\cos\theta = n_x \cdot n'_x + n_y \cdot n'_y. \tag{1}$$

The translation parameters t_x and t_y are determined by the equations

$$s_x = R_\theta m_x + t_x, \tag{2}$$

$$s_y = R_\theta m_y + t_y \tag{3}$$

where R_θ is the two-dimensional rotation matrix corresponding to the angle θ. Thus (t_x, t_y, θ) are the localization parameters corresponding to the pair (s_i, m_k). A complete scene interpretation is deemed to be *globally* consistent if each pair (s_i, m_j) where $0 \leqslant i \leqslant M-1$, $0 \leqslant j \leqslant N-1$ in the complete interpretation yields identical localization parameters.

3. Mapping the I.T. on the hypercube

In order to parallelize the I.T. search on the hypercube it is necessary to map the I.T. structure on the underlying hypercube architecture. The mapping strategy used has a great impact on how the original task is subdivided into subtasks and how concurrency is exploited amongst the subtasks. The process of mapping is thus essentially a task-decomposition process. In this section three strategies for mapping the I.T. on the hypercube architecture are described.

3.1. Breadth-first mapping of the I.T.

Each side of the object model is mapped onto a single node processor in the iPSC/2 hypercube. Thus if the object model has N sides, there must be at least N processors available in the hypercube. The current interpretation is stored in a structure called *interptype* which contains an array *it*[ ]. A pointer type *interpptr* points to a structure of the type *interptype*. If the variable *interp* is declared to be of the type *interpptr* then the current interpretation is represented by the array *interp*→*it*[ ] of length M (total number of sensors). The model and the scene data is made available to each node processor. At level L in the I.T., the kth node in the hypercube would try to extend each of the current partial interpretations $\{(s_i, m_j) \mid 0 \leqslant i \leqslant L-1,\ L < M-1,\ 0 \leqslant j \leqslant N-1\}$ by the pair (s_L, m_k). This is done by the assignment

interp→*it*[*L*] = *k*.

The extended partial interpretation is checked for consistency by using the previously mentioned constraints. If the extended partial interpretation is found to be consistent then it is broadcasted to all the node processors in the hypercube using the *gsendx*( ) facility. If a leaf node in the I.T. is encountered (i.e., $L = M-1$) then the extended partial interpretation (if consistent) is deemed complete and conveyed to the host. If an extended partial interpretation is found to be inconsistent then it is not broadcasted. Not broadcasting a partial interpretation is tantamount to effectively pruning it during the I.T. search process.

The traversal of the I.T. is terminated when every path in the I.T. from the root node to a leaf node has been either traversed or pruned. The host keeps track of the number of leaf nodes visited or pruned during the I.T. search process using a counter. If a node processor visits a leaf node then it instructs the host to increment the counter by 1. If the node processor prunes the I.T. at level k then it instructs the host to increment the counter by N^{M-k}. The host terminates the I.T. search process when the contents of the counter equals N^M.

Each node in the I.T. performs the same task—extending an existing partial interpretation, constraint checking, pruning and message processing. The breadth-first mapping achieves perfect load-balancing since each node processor visits the same number of I.T. nodes. The cost in terms of interprocessor communication, however, is excessive. Also, there is a limitation that the number of sides (faces) in the object model cannot be greater than the number of node processors in the hypercube.

3.2. Depth-first mapping of the I.T.

The entire I.T. at the first level is decomposed into groups of subtrees as shown in Figure 2.

A group may contain a single subtree or a block of adjacent subtrees. Each processor explores its group of subtrees using the *hypothesize-and-test* with backtracking search procedure. In the present implementation, the search procedure has been implemented as a recursion. The unique interpretation constraint, the distance constraint and the angle constraint are used to prune inconsistent paths in the I.T. When a complete interpretation is found the node processor conveys it to the host via a message. When a node processor has finished exploring its group of subtrees, it sends an appropriate message to the host. The host terminates the I.T. search when all the node processors are finished and proceeds with the pose determination.

The depth-first mapping of the I.T. involves no inter-processor communication. The only communication is between the host and the individual node processors. However, because of the unpredictable nature of the search process, some subtrees could be pruned earlier than others. This causes load imbalance amongst the node processors.

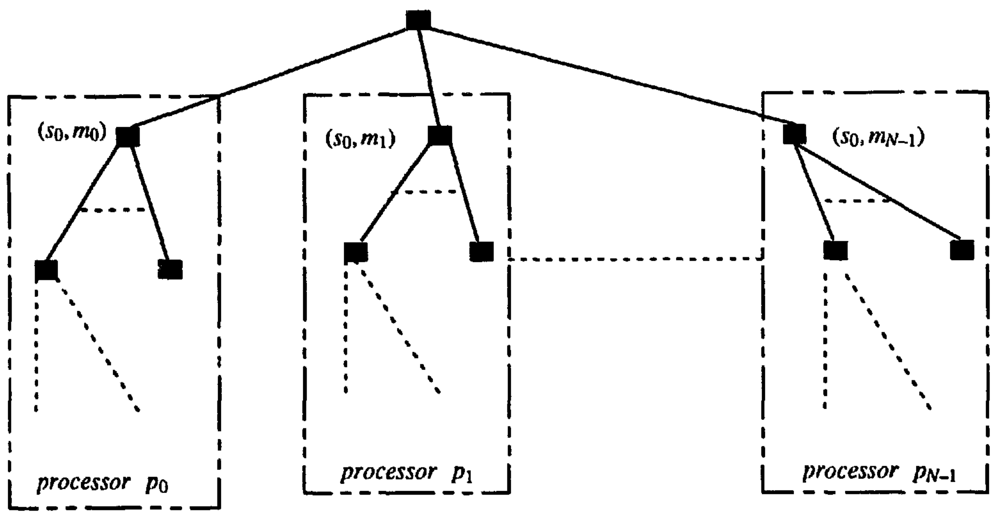

Figure 2. Depth-first mapping of the I.T.

Some nodes may finish exploring their group of subtrees and stay idle while other nodes are still exploring theirs. The depth-first mapping has the advantage that more than one subtree can be handled by a single node processor. This is a considerable improvement over the breadth-first mapping which requires that the number of sides (faces) in the object model does not exceed the number of available node processors in the hypercube.

3.3. *Depth-first mapping of the I.T. with load sharing*

The I.T. is decomposed into groups of subtrees just as in the case of the simple depth-first mapping. A single subtree or a block of adjacent subtrees is assigned to each node processor. Each node processor proceeds to explore its group of subtrees using the hypothesize-and-test with backtracking search procedure as in the case of the simple depth-first mapping. As mentioned previously, due to the unpredictable nature of the search process, some node processors may explore their assigned group of subtrees earlier than others. Thus it is virtually impossible to distribute the subtrees a priori between the processors so as to ensure perfect load sharing. In the depth-first mapping with load sharing, the host keeps a record of the status of each node processor. The host associates a *flag* with each node processor. The flag indicates whether the processor is *busy* or *idle*. When the I.T. search process is initiated, the status flag of all the processors is set to *busy*. When a particular node processor completes exploring its group of subtrees, it conveys the result to the host and also signals the host to change its status flag from *busy* to *idle*. The host checks to determine the closest *busy* processor to the now *idle* processor in terms of the Hamming distance in the hypercube. The host interrupts the busy processor and signals it to transfer part of its workload to the idle processor. The busy processor transfers half of its present workload to the idle processor via a message. The idle processor on receipt of the message from the busy processor sends a message to the busy processor acknowledging the receipt of the workload and asking it to resume its task. Simultaneously, the idle processor sends a message to the host requesting the host to change its status flag from *idle* to *busy*. The communication sequence is depicted in Figure 3.

The depth-first mapping with load sharing manages to achieve load balancing without excessive inter-processor communication. The communication between the host and the node processors is limited to transfer of the result of the search process from the node processor to the host. The inter-processor communication is limited to the extend of loadsharing. Other than that, the individual node processors explore their respective subtrees independently of each other.

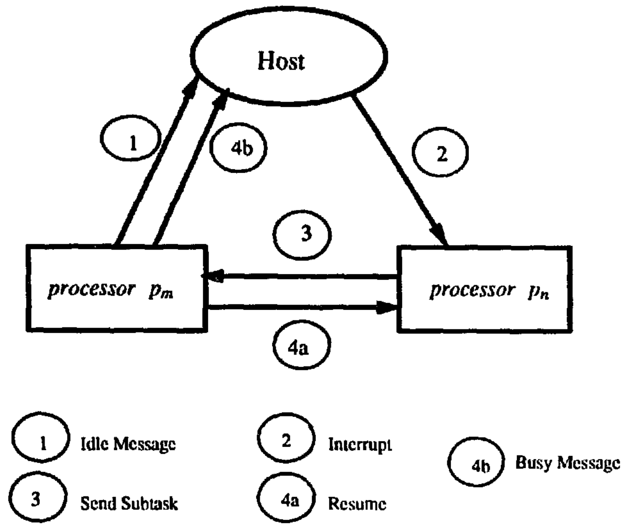

Figure 3. Communication sequence for load sharing.

4. Experimental results

The parallel I.T. search algorithm was implemented and tested on synthetic tactile data from two-dimensional scenes. Both single-object and multiple-object scenes were used. The I.T. search algorithm for a single object scene was modified slightly in order for it to be able to handle multiple-object scenes. This was done by addition of a *null* side (face) to each object model. The assignment of a sensory measurement to the null side (face) of an object model is equivalent to saying that the particular scene feature is not matched to any side (face) of the object model. The assignment of a sensory measurement to the null side (face) of an object model is not checked for constraint satisfaction.

Figure 4 shows a typical input scene containing a single object whereas Figure 5 shows a typical input scene containing two objects. Table 1 summarizes the results of the parallel I.T. search using breadth-first mapping (BFM), depth-first mapping (DFM) and depth-first mapping with load sharing (DFMWLS) respectively for the input scene shown in Figure 4. Table 2 summarizes the results of the parallel I.T. search using breadth-first mapping (BFM), depth-first mapping (DFM) and depth-first mapping with load sharing (DFMWLS) respectively for the input scene shown in Figure 5. For each mapping technique the run time of the I.T. search process for different numbers of sensory measurement (M) was noted. As can be seen from Tables 1 and 2, I.T. search with BFM was seen to be the most extensive in terms of running time for the I.T. search process. This could be attributed to the fact that the BFM entails constant broadcasting of information by each of the individual processors and also communication between the host and the node processors. I.T. search with DFM was seen to be the fastest since there was no inter-processor communication. However, the fact that DFM does not ensure a balanced load amongst the node processors, is reflected in the different running times for the I.T. search at different node processors. In Tables 1 and 2, this fact has been noted by entering the maximum and minimum running times for the node processors. I.T. search with DFMWLS is seen to be a good compromise between the BFM and DFM. The running time of I.T. search with DFMWLS is higher than that of I.T. search with DFM which is the price one has to pay for load balancing. At the same time, the running time of I.T. search with DFMWLS is lower than that of I.T. search with

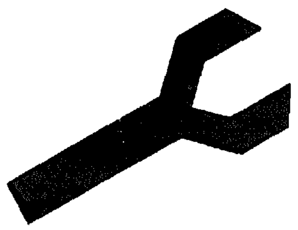

Figure 4. Scene I containing a single object.

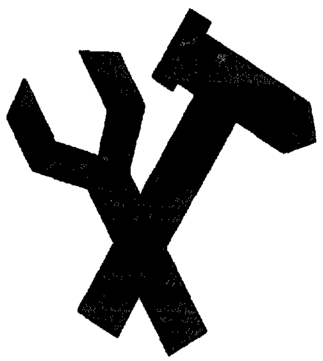

Figure 5. Scene II containing multiple objects.

Table 1
Performance of the I.T. search on Scene I

Search Technique		No. of sensory measurements		
		3	5	7
BFM		2410	3113	3140
DFM	max	3	5	5
	min	1	1	1
DFMWLS		214	216	194

All times are in milliseconds

BFM since the former avoids redundant interprocessor communication. Similar results were observed in the case of other input scenes. The parallel I.T. search technique outlined in this paper is in the process of being extended to handle three-dimensional scenes. Incorporation of heuristics in the parallel I.T. search technique in order to prune the I.T. more efficiently is also being considered.

5. Conclusions

In this paper a parallel Interpretation Tree search algorithm for object recognition using sparse range or tactile data on the Intel iPSC/2 hypercube multicomputer has been considered. The objects are typically those which can be approximated as piecewise combination of polyhedra and which a robot would encounter in an industrial scene during the process of automated inspection or automated assembly. Three strategies for mapping the Interpretation Tree search process on the hypercube have been considered. These have been termed as breadth-first mapping, depth-first mapping and depth-first mapping with load-sharing. The algorithm has been experimentally verified on synthetic tactile data from two-dimensional scenes. It has been shown that the requirement for uniform load sharing leads to increased inter-processor communication.

Table 2
Performance of the I.T. search on Scene II

Object No.	Search Technique		No. of sensory measurements		
			3	5	7
1	BFM		33128	34112	35186
	DFM	max	56	68	83
		min	14	17	26
	DFMWLS		887	1017	1218
2	BFM		21540	23115	24124
	DFM	max	47	58	74
		min	11	16	25
	DFMWLS		807	1004	1057

All times are in milliseconds

References

[1] Grimson, W.E.L. and T. Lozano-Perez (1984). Model-based recognition from sparse range or tactile data. *Int. J. Robotics Research* 3 (3), 3–35.

[2] Grimson, W.E.L. and T. Lozano-Perez (1987). Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Anal. Machine Intell.* 9 (4), 469–482.

[3] Faugeras, O.D. and M. Herbert (1986). The representation, recognition and localization of 3-D objects. *Int. J. Robotics Research* 5 (3), 27–52.

[4] Grimson, W.E.L. (1986). The combinatorics of local constraints in model-based recognition and localization from sparse data. *J. ACM* 33 (4), 658–686.

[5] Grimson, W.E.L. (1990). The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence* 44, 121–165.