A Distributed Blackboard System for Vision Applications

Malcolm D. Brown* and Robert B. Fisher
Dept. of Artificial Intelligence, University of Edinburgh
5 Forrest Hill, Edinburgh EH1 2QL, Scotland, United Kingdom
email: rbf@uk.ac.edinburgh.edai
fax: 31-225-9370
telephone: 31-667-1011 x2553

Abstract

This paper describes an implementation of a distributed parallel blackboard system that runs on a Meiko multi-transputer system. The blackboard is split up amongst the transputers to allow for distributed local processing, yet the access of the blackboard is transparent to the user processes, irrespective of whether the data is local or remote. Multiple expert processes are invoked as processing resources become available and task dependencies are resolved. The implementation allows both task and data parallelism. A Canny edge detector implementation achieved a speedup of 21 times on 64 transputers.

1 Introduction

Blackboard systems provide a powerful mechanism for solving complex vision problems by using multiple domain expertise, implemented as independent knowledge sources (KS). The system described in this paper is a distributed memory blackboard system implemented on the Edinburgh Concurrent Supercomputer - a multi-transputer system. The distributed blackboard system applies both task and data parallelism to vision problem solving.

The increasing size and complexity of vision problems has resulted in the development of blackboard systems as a problem solving technique. Typically, specialist vision systems consist of large amounts of knowledge stored and used in one way. This is fine for certain classes of problems; however, it seems unlikely that systems with only a single approach will be able to handle larger vision problems involving large quantities of data, uncertainty, cooperative multiple and alternative KS application and independent exploration of alternative hypotheses.

* Supported by a SERC studentship. Currently employed at AI Ltd, Greycaine Road, Watford, England

The motivations behind building a distributed blackboard system are:

- Multiple KSs can be used to cooperatively solve a problem.
- Task and data parallelism improves the problem solving performance of the system.

Mixing the two types of parallelism is particularly important to vision applications, where substantial data parallelism is needed to interpret raw data during low-level vision, and task parallelism is needed to investigate independent or alternative hypotheses during high-level vision processing.

The distributed blackboard system[1] presented in this paper consists of a distributed blackboard data structure, a central control mechanism and distributed hierarchical KSs.

2 Blackboard Systems

Blackboard systems[2] are characterized by a collection of KSs, a shared memory blackboard and a task control mechanism. The KSs cooperate with each other only through: (1) reading and writing to the blackboard and (2) by directly requesting the task controller to execute other KSs. The blackboard holds the specification of the original problem and contains the partial solutions developed by the KSs so far. The task control mechanism's goal is to ensure that the KSs cooperate together to solve the problem.

The blackboard or shared memory is used by the KSs to record the outstanding problems that need to be solved for the system to reach a solution and the partial solutions developed so far. The system can develop several possible solution paths in parallel and proceed both forwards from the data and backwards from the goals.

A task control mechanism invokes the KSs, but only if the KS can potentially contribute towards the solution of the problem. Thus, it needs to determine the set of KSs that are applicable. This set can be determined by the KSs reacting to events on the blackboard, or by the task controller matching the known capabilities of the KSs with the outstanding tasks or by requests from previously executed KSs.

Two parallel blackboard systems are the CAGE[3] and POLIGON[4] systems. CAGE is a shared memory multiprocessor implementation of the AGE[5] system that allows parallelism to be used at varying levels (e.g. at the level of individual rules, or as collections of rules). The POLIGON system is implemented on a message passing multi-processor, and has a distributed control mechanism. POLIGON most closely resembles the system described in this paper, except our system uses physically distributed memory, rather than a large shared memory. This seems to be more suited for parallel MIMD (multiple-instruction, multiple-data) machines, where each processor has a co-located memory. The SCHEMAS system[6], the VISIONS system[7], and the SBS system[8] are examples of blackboard systems used in computer vision applications.

3 The Distributed Parallel Blackboard System

The distributed parallel blackboard system consists of a task control subsystem and a database management subsystem. Both are implemented by using a message passing kernel[9] to forward processing and data requests to the appropriate processes.

The blackboard was implemented on a large Meiko Computing Surface, which is a multi-transputer processor. Each transputer has substantial local memory and four fast local neighbor communication channels. In the implementation, many independent processes run on each transputer, which are connected into a torus configuration (Figure 1). The advantage of the torus is that the central portion of the processor array is connected analogously with regular image topology. Hence, it provides a firm foundation for data-directed parallelism, wherein some operation is applied everywhere in the image. A torus connection topology also means that individual processors are not particularly far from each other for communication and database access in task-parallel regimes (although 3D connection topologies can further reduce the average interprocessor communication distance).

3.1 Knowledge Source Control

In a sequential blackboard system only one KS can be active at any time. A typical serial blackboard execution cycle consists of repeatedly selecting and invoking a suitable KS until a solution is found. Thus, the KSs passively interact with each other by running in turn.

The parallel blackboard also activates KSs as processing resources become available, except that many KSs may be active simultaneously. By implementing the parallel blackboard on a multi-processor system, true parallelism can be achieved by distributing the KSs amongst the transputers.

The KS invocation requests are enqueued by:

- the storage of new data items,
- the controller recognizing that a KS can potentially satisfy an outstanding goal,
- requests from other KSs and
- entry by the user.

If the input blackboard objects required for a task have not been created, then a data dependency is unresolved. In this case, the KS will enqueue subgoals which, when satisfied, will result in the creation of the desired blackboard objects. The KS then requeues its goal together with the names of the blackboard objects upon which it is dependent, which causes the goal to suspend until known data dependencies are resolved.

A KS is able to run when all data dependencies are satisfied. This is useful for backward chaining, where the solution to a problem is required (e.g. find the 3D edges in a scene) and the KS is dependent on its data being available, or subproblems being previously solved. This causes other goals to be created and a rescheduling of the main goal dependent on the satisfaction of the other goals (e.g. find the 2D edges in a stereo pair of images). The actual invocation of a KS occurs when it is able to run and a transputer containing the KS is available.

The control of the KSs is centralized - that is, one transputer has a control blackboard and receives all goal requests. The current implementation uses a first-come first-served queue, though facilities are provided for control KSs to manipulate the queue. Distributed control is also possible, but rejected here, because: (1) distributed control is much more difficult, and it is hard to focus the processing to achieve a solution[4] and (2) many vision applications seem to have simple goal requests that require substantial processing. Hence, the ratio of processing to control communication is high.

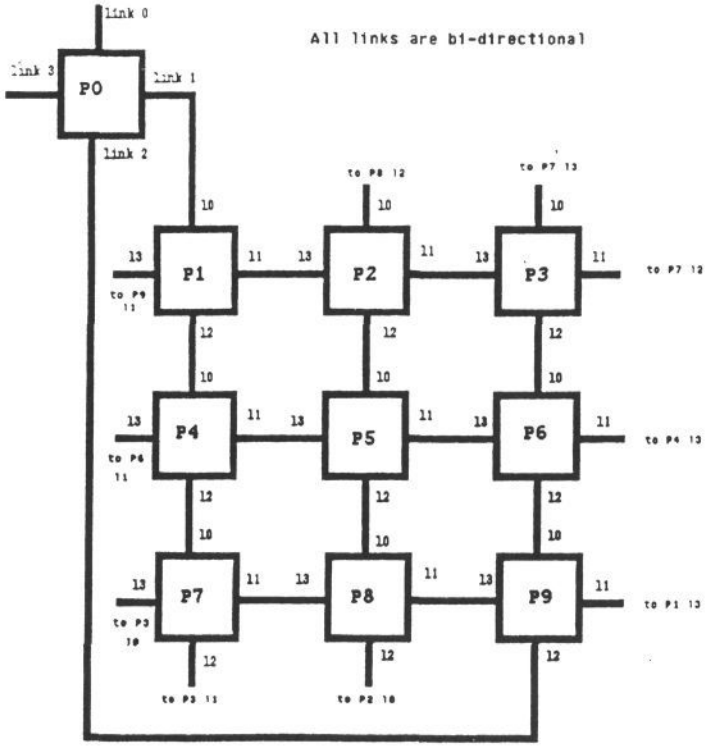

Figure 1: Transputer Connection Topology

Blackboard control for most low level vision is simple since the sequence of actions is predetermined. If alternatives exist, there are usually few paths to follow and the best one is generally easy to spot. For high level vision, this is less often the case, as the action to apply may depend on the data. For example, in a 3D scene analysis program, the reasoning may require: prediction and search for missing image features, inference of visibility, search for subcomponents - each of which might require several alternative approaches before a satisfactory result is obtained.

The control system needs to determine which path to follow, or indeed which paths to follow since multiple hypotheses about different data and model objects can be evaluated in parallel. The control processing can be performed by control KSs which are invoked by the blackboard scheduler when it gets into a situation which requires complicated control decisions to be made. These control KSs will take the complicated control situation and use their control knowledge to determine the best next step to determine the solution. The blackboard control system can then use the results to invoke problem solving KSs. For example, the control KS could reason: as the estimated position of the geometric model predicts that a feature should be forward-facing and self-occlusion analysis indicates no occlusion, and direct search cannot find the missing feature, then invoke the "external-occlusion" verification process.

The control KSs resolve the decision problem by determining which of the possible next steps are both desirable and satisfy the current problem solving constraints. Seven[10] behavioral goals for intelligent control have been identified.

Knowledge sources are often classified as coarse or fine-grained, depending on whether they execute large amounts of computation (e.g. edge-detect an image) or execute only one or a few logical inferences (e.g. as in a production rule system).

For low level vision, coarse grained KSs seem to be more appropriate, because generally a large amount of data-parallel image-oriented calculation and only a little logic is required. Further, as the image is spatially distributed, it makes sense to also spatially distribute the KSs, in correspondence with the image. Otherwise, considerable performance degradation will result from having to move the data around to available KSs. As requests for processing usually require the application of a given process to a single image, this means that subrequests need to be generated and distributed. To account for these factors, the design model for the low-level KSs was based on a single master KS with multiple slave KSs, wherein the master KSs maintain data dependencies, but distributes the actual KS workload amongst the slave KSs.

Once the master KS has successfully established the presence of its dependent input objects, it can start the task of satisfying its assigned goal. If the task involves operations on distributed arrays the task can be split into subtasks which use the distributed subarrays, and these subtasks are assigned to slave KSs to achieve parallel task execution. The master KS attempts to match the location of the subarrays with slave KSs to minimise inter-processor communication. Once all the slave KSs have completed their allocated subtasks the master KS can make any resulting blackboard objects public which will allow any data dependencies on these objects to be cleared.

The slave KSs are distributed amongst the processors to execute tasks on distributed arrays in parallel and currently all transputers have copies of all slave KSs, though this need not be the case. They suspend, waiting fora start message from a master KS, which includes information about which subarray the slave KS should process. When the slave KS finishes its allocated subtask, it sends a completion message to its master KS.

For high level vision, the tasks relate more to hypotheses, but may require both image data and model based access. There will usually be multiple hypotheses, and there might be multiple approaches to verifying the hypotheses. Hence, task parallelism is likely to be the dominant paradigm.

The hypotheses need not be spatially located, but most are, because, by the nature of the computer vision prob-

lem, the data that substantiate the hypotheses are spatially located. In processing the hypotheses, more logic is needed than in low level vision, although substantial data processing might also occur (e.g. when relating model and image features to extract shape or position parameters). While fine-grained parallelism is possible, it has seemed to us to be more suitable to use coarser-grained parallelism, because most processing can be associated with individual hypotheses, and the hypotheses themselves can be spatially distributed, to be near their supporting image data.

To account for these factors, the design model for the high level KSs was:

- one KS for each reasoning process, for each processor (i.e. distributed throughout the image)
- use of the agenda to record new processing to be applied to hypotheses
- distribution of tasks to processors near to the required data

3.2 The Blackboard Database

The blackboard is designed for four types of objects, because they seem to be the most useful types of data structures for computer vision purposes:

1. atomic items (e.g. a record structure), such as might be used to record the properties of an image feature,
2. lists of atomic items, e.g. for recording a boundary, or a set of hypotheses,
3. arrays, e.g. for small matrices, or sub-arrays of an image and
4. distributed arrays (where the array is partitioned amongst the transputers), generally used for splitting up an image amongst a set of transputers.

The contents of the database is managed by use of two tables, the *BB descriptor table* and the *BB object table.* The BB descriptor table describes the types of objects that can be created in the blackboard, and records:

- descriptor name (e.g. "raw intensity image")
- data type (which of the above four types)
- array dimensions if array or partitioned array
- a daemon to be scheduled when instances of the data structure are created

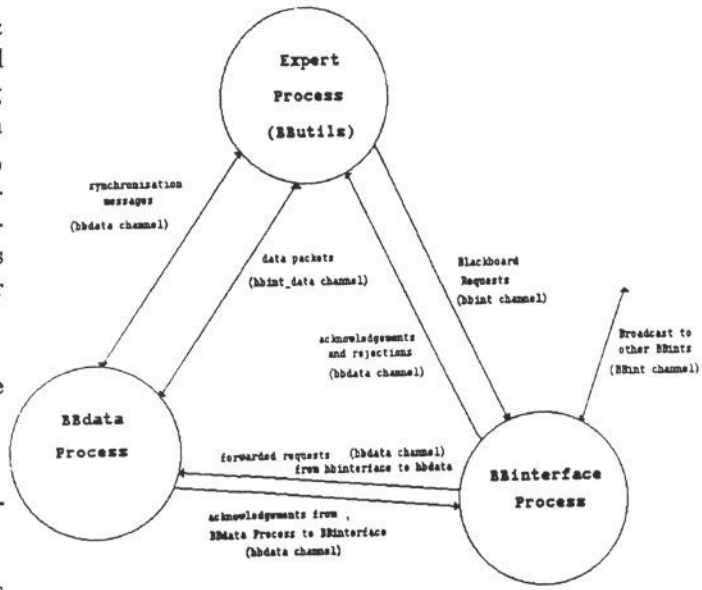

Figure 2: Interaction Between Blackboard Processes

The BB descriptor table is created statically and is currently defined at compile time. The BB object table records information about the specific data objects that have been created. Each record contains:

- user-defined object name
- unique blackboard identifier
- descriptor type
- physical location
- element size
- protection status

Access to the database is via interface subroutines taht manage the underlying data access transparently to the KS, irrespective of whether the data is local or remote. Using the subroutines, a KS can create, read and write instances of the four types of objects. In addition, it can enable read-only access or deny access to objects.

The blackboard contents are stored in a set of distributed *blackboard-data* processes. In addition, there are a set of distributed *blackboard-interface* processes. KS-callable subroutines pass database access requests to the local blackboard-interface process, which then validates the requests and forwards them to the appropriate blackboard-data processes, which return the results to the requesting KS. The interface routines also assemble or partition distributed arrays when necessary. The communication between the processes is managed by the message-passing subsystem. Figure 2 shows the commu-

nication channels and types of messages sent between the KS process and the blackboard management processes.

Here, the blackboard is physically distributed amongst the transputers thus making the system data parallel. This form of parallelism is useful for computer vision applications that involve storing images (i.e. large arrays of data), since one usually distributes image processing about a processor array. Other data structures (besides distributed-images) can also be stored throughout the processor array (although they are stored complete, rather than partitioned), so this can support efficient task-parallelism. In any case, the knowledge sources do not need to know where a piece of data is physically stored (but could be applied more efficiently if invoked nearby to where all requisite data is available).

4 Performance Using a Parallel Canny Edge Detector

A Canny edge detector[11] was implemented on the distributed blackboard system as a test domain. It was chosen because there is substantial data parallelism in its first three stages (image smoothing, gradient calculation and non-maximal suppression), but also task parallelism in the final stage, where individual line segments are tracked.

The Canny KSs are arranged into a hierarchy with four master KSs (for the four Canny stages) controlling slave KSs, each responsible for a subtask (e.g. smoothing) on a local image subarray.

The main data structures allocated by the Canny processes are all partitioned arrays:

1. the raw image,
2. two smoothed images (x and y smoothing),
3. two derivative images (x and y derivative),
4. the unsuppressed gradient magnitude,
5. the suppressed gradient magnitude,
6. the suppressed gradient orientation and
7. the tracked edges

These data structures (identified by the number given above) are used by the following KS processes:

MASTER	INPUT	OUTPUT	SLAVES?
load image		1	N
1D image smooth	1	2	Y
1D derivative	2	3,4	Y
non-max suppress	3,4	5,6	Y
track edges	5,6	7	N

The edge tracking process could also have had slave processes.

We describe now the sequence of actions that occur in a goal-driven execution of the **smooth_image** goal. The initial system state has the blackboard empty. The goal agenda contains the goal **smooth_image(image1)** with no data dependencies attached. The sequence of actions is:

1. The scheduler dequeues the **smooth_image (image1)** goal and allocates it to the **smooth_image** master KS.
2. The **smooth_image** master KS cannot find its data, image1, so it enters the **load_image(image1)** goal into the agenda. After this it suspends itself by re-enqueuing itself with a data dependency on image1.
3. The **load_image(image1)** task is then allocated to the **load_image** KS which reads the image from file and loads it into a newly created blackboard object for the image.
4. The **load_image(image1)** goal is successfully completed and image1 is made public.
5. The data dependency for the **smooth_image** goal is now cleared, and the goal is again allocated to the **smooth_image** master KS.
6. The **smooth_image** master KS breaks the smooth image task up into subtasks which operate on subarrays of the image, and these subtasks are distributed to the slave KSs.
7. When all the slave KSs have finished the **smooth_image** master KS makes the resulting smoothed image public.

The other Canny tasks are executed in a similar manner.

The blackboard system was successfully implemented with the Canny edge detector as the test domain. The system was tested over a range of transputer array sizes. We report here the speedup (ratio of processing time on one transputer to time on N transputers) and efficiency (ratio of speedup to the number of transputers used).

The speedup and efficiency achieved are illustrated in

Figure 3. A maximum speedup of 21.2 (1.6 seconds on a 256^2 image, excluding image loading) was achieved with the distributed blackboard Canny experts running on 64 transputers. The speedup then drops away because of increasing data communication costs, but still remains at a high level. The performance achieved is satisfactory for a first implementation of the distributed blackboard system, and does not include tuning of the blackboard, which would improve performance.

Running the blackboard on a single transputer took 38 seconds, as compared to 24 seconds on a SUN 3/50. Given the raw transputer performance, this means that there is much overhead on the use of the blackboard. However, there is still considerable scope for further developments that will improve the performance (including C compilers for transputers that generate more efficient code). In particular, we feel that the database access process can be greatly streamlined, in part by reducing communication overheads through merging the local blackboard-interface and blackboard-data processes. Prioritizing KSs and recording KS failure are also planned control extensions.

Of course, it is always possible to develop special purpose algorithm implementations that use the transputer facilities directly, and thus achieve much greater efficiency. But the distributed blackboard system described here provides greater flexibility in a more general structure, simultaneously supporting both data and task parallelism, such as are needed for systems that involve both low and high-level image interpretation.

Acknowledgements

This work was funded by a UK Science and Engineering Research Council Studentship. We would like to thank R. Baldock, J. Hallam and H. Hughes for advice.

Bibliography

1. Brown, M. D., "A Distributed Blackboard System for Vision Applications", MSc Thesis, University of Edinburgh, Dept. of Artificial Intelligence, 1989.

2. Engelmore, R. and Morgan, T., Blackboard Systems, Addison Wesley, 1988.

3. Aiello, N., "User-Directed Control of Parallelism: The Cage System", Stanford University Technical Report KSL-86-31, 1986.

4. Rice, J., "POLIGON: A System for Parallel Problem Solving", Stanford University Technical Report KSL-86-19, 1986.

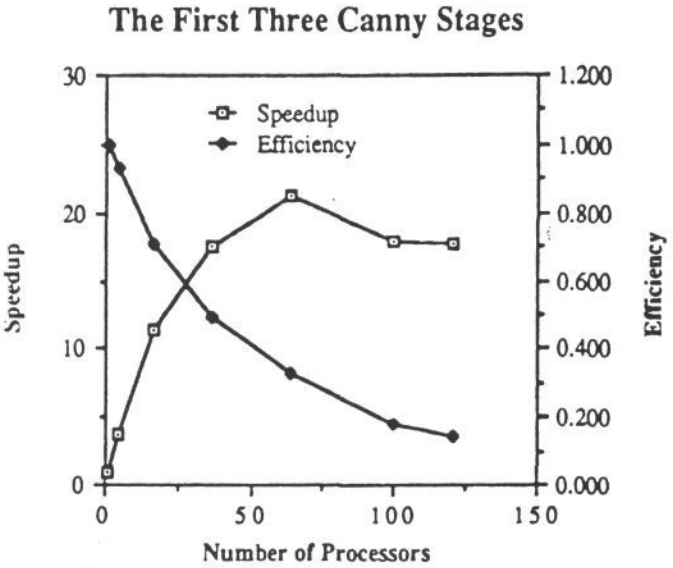

Figure 3: The Speedup and Efficiency Achieved

5. Nii, H. P. and Aiello, N., "AGE(Attempt to GEneralize):A Knowledge-Based Program for Building Knowledge-Based Programs", in Engelmore and Morgan (eds), Blackboard Systems, Addison Wesley, pp 251-280, 1988.

6. Draper, B., Collins, R., Brolio, J., Hanson, A. and Riseman, E., "Issues in the Development of a Blackboard-Based Schema System for Image Understanding", in Engelmore and Morgan (eds), Blackboard Systems, Addison Wesley, pp 189-218, 1988.

7. Hanson, A. and Riseman, E., "VISIONS: A Computer System for Interpreting Scenes", in Hanson and Riseman (eds), Computer Vision Systems, Academic Press, New York, pp 303-333, 1978.

8. Towers, S. and Baldock, R., "Application of a Knowledge-Based System to the Interpretation of Ultrasound Images - Preliminary Studies", Medical Research Council technical report, Western General Hospital, Edinburgh, 1988.

9. Clarke, L. J., "*tiny* Discussion and User Guide", Report ECSP-UG-9, Edinburgh Concurrent Supercomputer Project, Univ. of Edinburgh, March 1989.

10. Hayes-Roth, B. and Hewett, M., "BB1: An Implementation of the Blackboard Control Architecture", in Engelmore and Morgan (eds), Blackboard Systems, Addison Wesley, pp 297-313, 1988.

11. Canny, J. F., "Finding Edges and Lines in Images", MIT, AI Lab, Techical report AI-TR 720, 1983.