1990

Design of a Dynamically Reconfigurable, Integrated, Parallel Vision System

Ashok K. Samal
University of Nebraska - Lincoln, asamal1@unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/cseconfwork

Part of the Computer Sciences Commons

http://digitalcommons.unl.edu/cseconfwork/39

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
Design of a Dynamically Reconfigurable, Integrated, Parallel Vision System

Ashok Samal
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
e-mail: samal@fergvax.unl.edu

ABSTRACT

Multiprocessors can be used to speed up the process of object recognition. Building a parallel vision system is a two step process: (a) design and implement parallel algorithms for individual operations, and (b) integrate them into a coherent system. So far the emphasis has been on the first step. The second step has not received enough attention. A framework for building a parallel vision system is presented here. The necessary and desirable features of such a system have been identified. An initial design which incorporates these features is also given.

2 Features of a Parallel Vision System

Before we design a system that will integrate a collection of vision algorithms into a cohesive unit, it is necessary to study their properties. It is particularly important in computer vision, since there are many operators and they are heterogeneous in their properties.

In low level vision, the computation is generally local, simple, repeatable, and numeric in nature. The data are usually well structured and large. The high level algorithms, on the other hand, are more complex and mostly symbolic in nature. The major implication is that the nature of computation and hence the computational requirements for the two levels of computer vision are rather different. In order for a machine to be suitable for computer vision as a whole and not just for a class of algorithms, it must be flexible enough to accommodate the diversity in the vision algorithms. Hence, the special purpose architectures designed for certain vision applications will not be quite as useful as general purpose parallel architectures. A computer vision system that is used in a parallel processing environment must take into account the difference in the nature of computing between low, intermediate, and high level vision algorithms.

In order to design a good parallel vision system one must first identify the features that are essential and also the features that may be desirable. We take a very broad view and analyze the features of a very general parallel vision system. The essential features of such a system are presented below:

- **Support for large number of operations**: A typical vision system consists of many operators. Each of the operations must be analyzed theoretically for their space, time, and processor requirements. They must, in addition, be efficiently implemented on a multiprocessor so that they can be used in the system.

- **Duplicate operators**: In addition, the system must support duplicate operators to accomplish the same objective. For example, there are many edge operators, Sobel, Laplacian, Gradient, Kirsch, Canny, etc. This is particularly true for low level vision. The conditions under which the operators are useful must also be stored.

- **Reconfigurability**: Reconfigurability is very important in a parallel vision system. The system must allow both static and dynamic reconfiguration. A static configuration is like a plan, which specifies the sequence of operations that must
be performed. The system may aid the user building a static configuration. The system must also be able to reconfigure dynamically at runtime if the need arises.

- Control and user interaction: The system should be flexible to allow both top-down and bottom-up control. It is important in vision systems since both forms of control are useful. In addition to automatic configuration of the system, the user must be able to interrupt the system at any time and force the system to take a different path to the goal without repeating the computation already performed.

- Forward/speculative computation: In higher level vision algorithms, the effective number of processors (the number of processors that provide positive incremental gain) is considerably less compared to low level algorithms. It is possible to explore for speculative computation in such algorithms. With the appropriate design it can be used even in low level vision.

- Integration of hardware units: In many cases, especially in low-level vision, special purpose hardware units are available to accomplish some tasks extremely fast. The design should be able to use both software and hardware modules and be able to replace one with another transparently.

- System efficiency: The main motivation of using a parallel processor is to improve the overall runtime of the system. It must be the ultimate goal of the system. At the same time the system must utilize the resources efficiently at all times during computation.

- Portability: To be general, the design should not, in any way, be dependent on features of a particular architecture. During implementation, however, one must be able to incorporate the idiosyncrasies of particular architectures to achieve high efficiency.

- Others: Other desirable features include efficient process management, image loading and unloading, data partitioning and migration, etc.

### 3 Design of a Parallel Vision System

Our goal is to build an easily integrated and dynamically reconfigurable parallel vision system for a realistic application (automatic face recognition) which incorporates most of these features. The proposed system consists of two major types of entities: logical function modules (LFMs) and strategy planning modules (SPMs) (See Figure 1). The LFMs are the function or the action units and the SPMs are the control units. In addition, there is a scheduler which is responsible for the runtime process management in the system. The framework used is partially motivated by the Logical Sensor Specification (LSS) introduced in [1] for the integration of multiple sensors.

#### 3.1 Logical Function Modules

Logical Function Modules (LFMs) represent the various units of work in the system. For example, there are logical function modules for edge detection, segmentation, discrete relaxation, etc. However, all edge detectors are grouped together in one LFM.

The LFMs, however, are not just collections of object modules. Each LFM consists of a static part, a dynamic part and a controller. The static part describes the object modules while the dynamic part is responsible at runtime to provide the correct interface to the other object modules, in terms of their input and output. The static part consists of the following:

- **Name**: The name of the module.
- **Function**: The function it is responsible for.
- **I/O Specification**: Number of inputs and outputs.
- **Program Units**: Specification of hardware/software units.
- **Selector Function**: Selection strategies for program units.

Each program unit is a composite containing its function, positive conditions, i.e., the conditions under which the program is useful, the specification of the executable, its processor requirements, and space requirements.

The dynamic part of an LFM is used while configuring the system for an actual recognition task. The static part contains information about the requirements of the object modules and their I/O specifications. However, the exact location of sources for its inputs and the destination for their outputs are not known until the whole system is configured. At that time this information is determined and is stored in the dynamic part.

![Figure 1: Design of the Parallel Reconfigurable Vision System](image-url)
3.2 Strategy Planning Modules

The Strategy Planning Modules (SPMs) are responsible for configuring the best strategy for a given application. The strategy is constrained by the resources in the system: processors and memory, the requirements of the various LFMs and their suitability, and the data dependency between them. The planning involves communication between the SPMs and the controllers of the various LFMs.

There are three SPMs, one for each level of vision: low, intermediate, and high. Arriving at a strategy involves the coordination between the three SPMs and is chosen to optimize the overall runtime of the system. If it is determined that some processors are idle at some point, the SPMs may decide to provide the scheduler with the option to perform advance/speculative computation. Once a strategy is determined, the SPMs send appropriate control signals to the (appropriate) LFMs to configure their input and output ports, etc.

To provide for user interrupts during the computation, an additional SPM is used. On receiving an interrupt, the system does not stop the computation. However, the scheduler allocates one processor to the user to view the results. The processes running on this processor are migrated elsewhere. At this point the user may stop all (or parts) of the computation or force certain decisions on the system. The SPM will reconfigure the LFMs accordingly.

3.3 The Scheduler

The creation and coordination of all processes at runtime is managed by the scheduler. If at any stage, data need to be migrated to obtain better result (or to reduce hotspots) it is also done by it. The decisions of the scheduler is guided by the expected runtime of the system. Another interesting role of the scheduler is to perform advance and speculative computation. If at any time there are idle processors, the scheduler may start some tasks ahead of time, subject to the data dependency constraints which are provided by the SPMs. Since, the scheduler knows at all times the status of system resources, it can perform this rather cleanly and efficiently.

While at first it may seem that a centralized scheduler is a bottleneck, it can be made very efficient. Distributed scheduling will be experimented with in the later designs of the system after enough experience is obtained with a centralized scheme.

3.4 Dynamic Configuration

Here we briefly describe how the system is configured at runtime. First, the task is specified in terms of the input, the domain of the image, etc. The SPMs then decide on the strategy in a cooperative manner as described before. The scheduler is informed of the overall strategy including options for speculative computation. The SPMs also send appropriate control signals to the LFMs (controllers) to allow their dynamic parts to be configured correctly. The individual LFMs inform the scheduler about their needs, i.e., processor requirements, dependency information, the executable names, etc. The scheduler essentially has a graph similar to a dataflow graph. After this, the scheduler is responsible for creating and managing the parallel processes.

If a certain unit is unavailable, e.g., a hardware unit, the system is configured without it. If the user wants to force a certain module to be used during the computation, after the system has been configured it is done easily using the User SPM which sends the control signals to the appropriate LFMs and also informs the scheduler. The controllers of the LFMs reconfigure their I/O accordingly in the same manner. Thus, both configuration and reconfiguration of the system can be done dynamically in an elegant way.

4 Implementation

The system is currently being implemented on a medium sized transputer system. We also intend to implement the system on a Connection Machine and a shared memory multiprocessor. The implementation is in its early stages.

To measure the success of the system, one must test its performance. Rosenfeld and others [4, 6] have performed benchmarking for several tasks on different multiprocessors for some sample test cases. However, the performance of a whole system is not measured. We intend to measure the performance of the system independently and as a function of the performance of the constituent algorithms.

5 Summary and Future Research

The main focus in parallel computer vision has so far been the design and analysis of parallel algorithms to perform individual operations. While it clearly is necessary and useful, it is not the ultimate goal. Our goal is to design and implement a parallel vision system which integrates all the parallel vision modules easily and efficiently. We have identified the desirable features of such a system. We also have given a design which incorporates most of these features. We intend to implement the system on multiprocessors, evaluate the system performance on real applications and extend/modify the design if necessary.

References