Chapter XXI
Distributed Image Processing on a Blackboard System

Gerald Schaefer
Loughborough University, UK

Roger Tait
University of Cambridge, UK

ABSTRACT

Efficient approaches to computationally intensive image processing tasks are currently highly sought after. In this chapter we show how a blackboard paradigm, originally developed for collaborative problem solving, can be used as an efficient and effective vehicle for distributed computation. Through the design of dedicated intelligent agents, typical image processing algorithms can be applied in parallel on multiple loosely coupled machines leading to a significant overall speedup as is verified in a series of experiments.

INTRODUCTION

Despite continuous increases in processor speeds, many image processing tasks remain computationally intensive. Partly this is due to researchers “keeping up” with the technology and developing more complex but better working algorithms, and partly because typical image sizes also continually grow due to improved technologies. It is therefore not uncommon that some algorithms take a relatively long time to run. While this is typically not a big obstacle for home users, a more feasible approach needs to be adopted in large scale image acquisition and processing environments or in real time or near-real time domains.

Conveniently, many image processing algorithms are inherently parallel and therefore well suited to a distributed implementation. An important consideration when parallelising such algorithms is the architecture of the host system...
Distributed Image Processing on a Blackboard System

(Jamieson et al., 1992). A tightly-coupled system comprises of one machine with multiple processors. In this case, data distribution is not necessary. In contrast, a loosely-coupled architecture, consisting of multiple computers in different locations will require distribution, communication, and accumulation mechanisms.

In this chapter we will only consider loosely-coupled systems as they are by far more common due to the high costs of tightly-coupled systems. We will show that by employing a distributed blackboard architecture consisting of several intelligent agents known as knowledge sources (KSs) it is possible to perform parallel image processing on a loosely-coupled system and achieve a significant performance gain when utilising a number of different machines.

BACKGROUND

In a tightly-coupled architecture all processors share the same main memory and work, concurrently, on the same data. Consequently, this type of system largely eliminates the need for explicit message passing between concurrent tasks. Multi-threaded programming allows applications to branch into independent concurrent threads and is not restricted to shared-memory multi-processor architectures. In general, multi-threaded applications are well suited to multi-processor architectures. This is because individual threads can run concurrently on different processors. As multi-threaded applications share the same address space, they cause considerably fewer overheads than the creation of an equivalent number of processes.

In a loosely-coupled architecture, parallel image processing tasks typically consist of four main steps: image distribution, local processing, data transfer during processing, and segment accumulation. Distribution is the process of dividing an image into segments each of which is assigned to a unique processor (Taniguchi et al., 1997). Under a duplicate distribution scheme each processor is sent an exact copy of the original image. Alternatively, more complex schemes can be adopted where an image is divided into a variable sized matrix (Nicolescu and Jonker, 2000). After distribution, each processor applies local image processing to its allocated segment. When data allocated to other processors are required, they are transferred by inter-processor communication. Finally, after application of the parallel algorithm, segments are accumulated into a resulting image.

Inter-processor communication is required when data allocated to other processors are needed, and can be categorised into groups based on their pattern of data access (Seinstra, Koelma and Geusebroek, 2002). These patterns also represent a strategy for synchronisation between communicating processors. One-to-one access is common in tasks such as image brightening or colour correction, where an output pixel maps directly to a pixel in the input image. Alternatively, a one-to-many relationship exists in neighbourhood operators, such as edge detection filters, which calculate an output based on a function of the input pixel’s immediate neighbourhood. Naturally, the handling and transmission of non-contiguous data differs from data stored as contiguous blocks. Data stored randomly in memory causes additional overheads due to its packing into a contiguous buffer before transmission (Hoare, 1985).

Multiagent systems offer the possibility of directly representing the individual components of a complex system (Murch and Johnson, 1998) and are inherently suitable for distributed implementation. The behaviour that is encapsulated within an agent gives it the ability to adapt, interact and evolve within the environment in which it exists. An agent makes decisions based upon memory, internal state and information received from other agents. If multiple agents are hosted on independent computers within the same network, communications coupled to a control module are used to achieve interaction and collaboration.