VICTORIA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF COMPUTER AND MATHEMATICAL SCIENCES

Parallel Implementation of Digital Image Compression Based on the JPEG Standard

Nalin K. Sharda (CMS, VUT)
Savitridevi G. Bevinakoppa (CSE, RMIT)
Hema Sharda (CSE, RMIT)

(32 COMP 6)
November 1993

TECHNICAL REPORT

VICTORIA UNIVERSITY OF TECHNOLOGY
(P O BOX 14428) MELBOURNE MAIL CENTRE
MELBOURNE, VICTORIA, 3000
AUSTRALIA

TELEPHONE (03) 688 4249 / 4492
FACSIMILE (03) 688 4050

Footscray Campus

Parallel Implementation of Digital Image Compression Based on the JPEG Standard

Nalin K. Sharda
nalin@matilda.vut.edu.au
Dept. of Computer and Mathematical Sciences
Victoria University of Technology,
PO Box 14428, MMC, Melbourne, VIC - 3000
AUSTRALIA

and

Savitridevi G. Bevinakoppa, Hema Sharda
m911618@tuan.hpc.citri.edu.au(Savitridevi)
Dept. of Computer Systems Engineering
Royal Melbourne Institute of Technology
GPO Box 2476V, Melbourne, VIC - 3001
AUSTRALIA

ABSTRACT: Digital image compression techniques are used to reduce the number of bits required to store and transmit images. To achieve required data compression in real-time, parallel processing techniques must be employed. Digital image compression techniques are increasingly being used for processing still and motion pictures. JPEG standard is becoming prevalent for the compression of still images.

This paper will describe the implementation of JPEG algorithm on a Transputer based system. The JPEG algorithm is programmed using Helios C in conjunction with the Component Distributed Language (CDL), working under the Helios operating system. Mapping of JPEG algorithm on array topology will be discussed. Performance of parallel implementation of the JPEG algorithm is described using Gantt chart. Speed up and efficiency of the system are also discussed.

1. Introduction

Digitally compressed images are used in storage and transmission systems to economise on space and time. Parallel processing techniques must be employed in order to perform image compression in real-time. Joint Photographic Experts Group (JPEG) is an international organisation which has developed digital image compression standard for continuous tone, still images. The JPEG standard specifies two classes of encoding and decoding processes viz. lossy and lossless. Discrete Cosine Transform (DCT) based mode of operation is specified for lossy compression and provides a capability which is adequate for many applications. The purpose of this work is to study methods for mapping the JPEG algorithm on a network of processors organized in an array topology.

The JPEG standard is described in section 2. Implementation of JPEG algorithm on a transputer networks of two different sizes are described in sections 3.2a and 3.2b. Execution schedule for the JPEG algorithm on different size networks are described with the help of Gantt charts.

2. JPEG standard

Consultative Committee for International Telegraph and Telephone (CCITT) and International Standard Organisation (ISO) have formed committees to develop image compression standards for still and motion pictures. For the past few years the Joint Photographic Experts Group (JPEG) standard group has been working towards establishing the first international digital image compression standard for continuous-tone (multi level) still images; it includes both grayscale and colour images [1]. The Motion Photographic Experts Group (MPEG) is working on developing standard for motion pictures.

There are two basic compression methods in the JPEG standard: Discrete Cosine Transform (DCT)-based method and predictive method [1]. DCT-based method of compression is widely used as it is suitable for a large number of applications. It is expected that DCT-based techniques developed for implementing the JPEG standard can be applied to the MPEG standard also.

2.1 *JPEG algorithm*

In the JPEG algorithm a source image consists of three colour components as shown in figure 1. Each pixel of a colour component is called as a sample of the image. A block of 8X8 samples is called one data unit. A sequence of (3 to 10) data units is known as one Minimum Coded Unit (MCU).

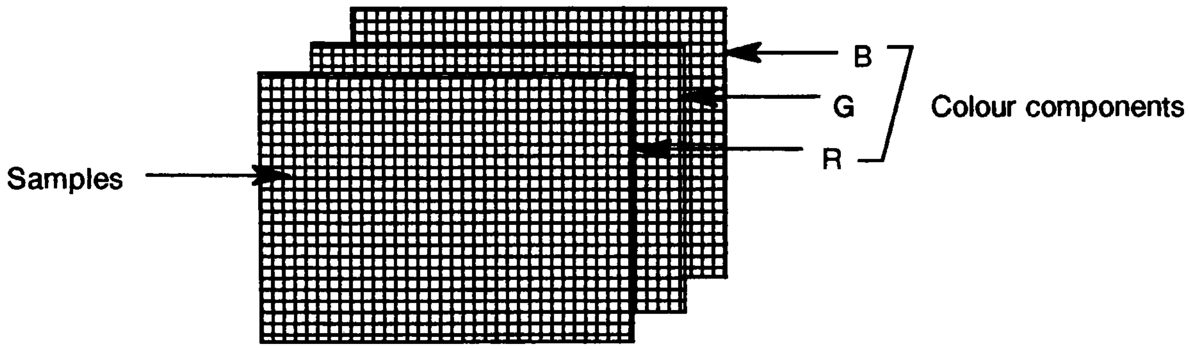


Figure 1: Source image

DCT-based method in JPEG standard involves Discrete Cosine Transform, quantisation and encoding of an incoming image. The input image is divided into groups of rows. Each row group is converted into a standard format and divided into a number of Minimum Coded Units. DCT, quantisation and encoding is performed on each data unit. After encoding input image is transferred into an output JPEG file.

Logical steps needed in lossy compression are given below.

1. Input image is divided into small groups and the division is performed row by row.
2. Each image group is converted into a standard internal format.
3. Colour space conversion is performed, e.g. RGB to YCbCr. This is a null step for grayscale images.
4. If required, an image group is expanded vertically and / or horizontally to fit a multiple of MCUs.
5. Sub-sampling is performed on each image component, this leads to reduction of number of samples in some colour components. This step operates independently on each colour component. Sub-sampling of each sample is performed by taking average with respect to neighbouring sampes.
6. Minimum Coded Units (MCU) are created by combining the required number of data units.
7. DCT, quantisation and Huffman encoding is performed on each data unit.
8. Finally, encoded image is transferred to an output file with a header / marker.

3. Implementation of the JPEG algorithm on a transputer based system

Most of the current software implementation of the JPEG algorithm are in sequential form. In attempting to perform JPEG data compression in real-time, parallel processing algorithms are being tried on a transputer network. Array topology has been selected for this study. JPEG algorithm will be implemented on transputers in MIMD mode using Helios C in conjunction with the Component Distributed Language (CDL), working under the Helios operating system. POSICs (Portable Operating System Instruction Codes) library has been selected for communication between transputers.

A few important terms which will be used later, are defined below.

1. Link usage:	Maximum number of the available links used in a network.
2. Bottleneck:	A link in the topology which carries a large volume of data traffic.
3. Mean inter-processor distance:	The average number of hops required to communicate between pairs of processors in the network.
4. Scalability:	A measure of topology performance as the number of processing elements is varied.
5. Speedup (S):	A measure of increased processing speed on a parallel processor.

$$S_N = \frac{\text{Time taken by a task on a single processor}}{\text{Time taken by a task on N processors}} = \frac{T_{serial}}{T_{parallel}}$$

When a task is divided into sub-tasks which need to communicate with each others, then the overall speedup is a function of the relative times spent on computing and communicating.

6. Efficiency (E): A measure of how much each processor contributes to the speedup of the parallel system.

$E_N = S / N$ Where S is the speedup, N is number of processors.

3.1 Helios operating system

Helios is a distributed operating system developed by Perihelion Software Limited. Helios is intended to run on a wide range of system configurations. A typical hardware configuration consists from two to twenty processors in a tightly bound cluster. This could be either a purpose built system or an upgraded workstation. Helios is capable of expanding into the available processors and of sharing the workload amongst them. Such clusters may themselves be interconnected in a local area network to allow the sharing of data and expensive devices such as high capacity discs and laser printers [2].

Helios provides four levels of communication between processors. The lowest level is used in the nucleus: PutMsg() and GetMsg() are the message passing primitives that provide the basis of all Helios communication. The level above this is provided by system library functions Read() and Write(). These calls operate on streams and have timeouts associated with each requested transaction. At the next level are the POSICs read() and write() functions. Calls to the POSICs functions are based on POSICs file descriptors. The highest level of communication is at the language level, which depends upon the programming language used.

We have chosen POSICs level of communication to implement JPEG algorithm on transputer network on the basis of the following factors.

1. This communication mechanisms ensures some degree of reliability and fault tolerant i.e. it provides error detection and recovery from failure and there is a guarantee that messages will arrive at their respective destinations.

2. It offers greater functionality than lower level libraries.

3. Most importantly, POSICs library can also be used on other parallel architecture, thereby giving portable code.

In POSICs based communication inter-processor message transmission times over transputer links are characterised by a relationship of the form [2]:

$$t_{total} = t_{overhead} + t_{init} + N.t_{tx} = 1461.42 + N \times 0.567 \text{ micro sec} \quad \text{------ } \{1\}$$

where

t_{total}	=	message transmission time
$t_{overhead}$	=	loop overhead on each test iteration
t_{init}	=	message initialisation time
N	=	number of bytes in a message
t_{tx}	=	transmission time for one byte

Component Distribution Language (CDL) is the language which facilitates parallel programming under Helios. The purpose of CDL is to provide a high-level approach to parallel programming.

3.2 *Implementation of JPEG algorithm on a transputer network*

For implementing the JPEG algorithm array topology has been selected on the basis of the following factors.

1. Link usage:	All four links of the transputer can be used, including the end links in a torus topology.
2. Bottleneck:	Bottlenecks can be minimised by using as many links as possible for distributing the data over the network
3. Mean inter-processor distance:	This distance is less in the array topology than in many other one or two dimensional topologies.

Implementation of the JPEG algorithm is being tested on different size networks, to determine scalability of the array topology for this problem. The data used to determine scalability is subject to the following factors:

1. Programming under the Helios operating system.
2. Communication is performed by using the POSICs routines.
3. A 125 X 125 Portable Pixel Map (PPM) P6 format input image is used. This format consists of three image components: RGB.
   Therefore, the number of samples is = 125 X 125 X 3

3.2.a *Implementation of JPEG algorithm on a three processors network*

The task graphs of the JPEG algorithm for parallel implementation look like a regular tree structures. Tree structures can be embedded in array structures quite easily. The next important question is how many processors can be used effectively? The minimax problem of parallel computing states that if we execute a job on only one machine then there is no communication overhead; at the same time we cannot exploit parallel operations. On the other hand, if we distribute the problem on too many processors the communication overhead can nullify the advantage of parallel processing.

Our aim is to determine the suitability of different size array topology networks for the JPEG algorithm. We will first describe a parallel implementation of the JPEG algorithm on a Transputer network with three processors, followed by an implementation on seventeen processors. Speedup (S) and efficiency (E) will be used as the main parameters to characterise the effectiveness of these networks.

The JPEG algorithm can be implemented on a three processors in MIMD mode. The three transputers are connected as shown in figure 2. JPEG algorithm is divided into four tasks as shown in the task graph of figure 3. In this task graph each node represents a processing task. Inside each node we write the processing task number (PTn) and the processing time t(PTn) of the task as the pair: PTn, t(PTn). Arcs joining the nodes represent communication tasks. Alongside each arc we write the communication task number (CTn) and the communication task time t(CTn) as the pair CTn, t(CTn). All times are in milli-seconds (msec). The processor on which a task is executed is specified adjacent to the task node in the task graph.

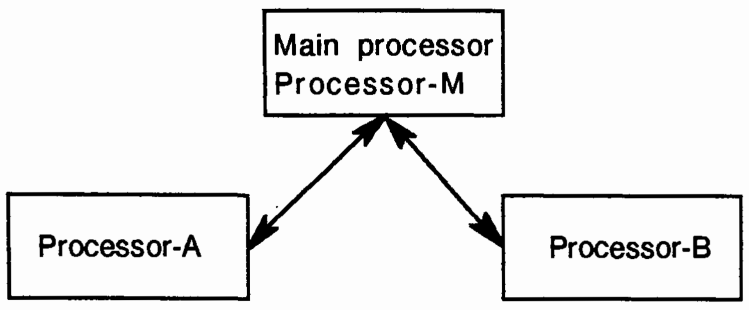


Figure 2: Tree topology for a network of three transputers

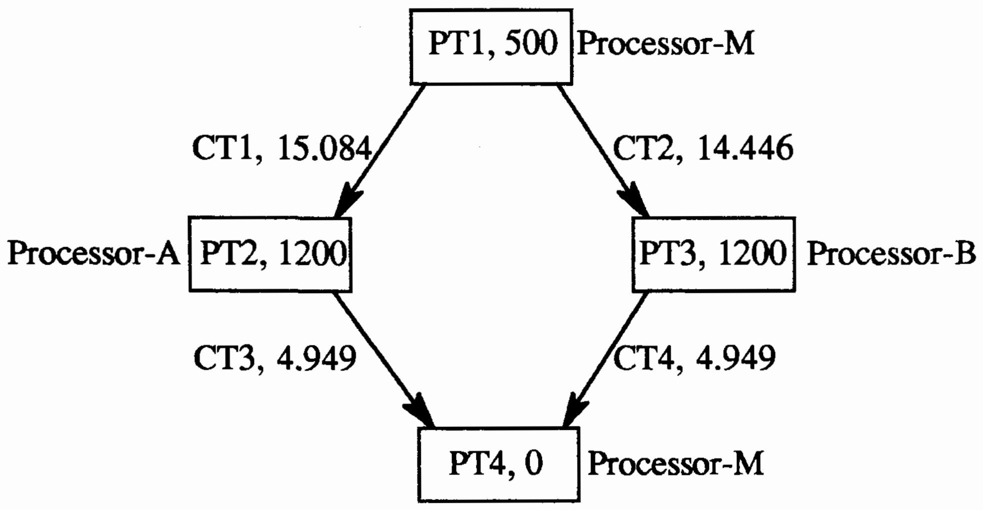


Figure 3: Task graph for three processors

Functions performed by the processing and communication tasks are as follows:

Task number	Function
PT1	Main processor performs initial set up of input image and divides input image into two parts.
CT1	First part of the image, which consists of 125 X 64 X 3 samples, is transferred from Processor-M to Processor-A. Communication time for transferring 125 X 64 X 3 samples (from equation {1}). = 1461.421142 X 10^{-3} + 0.567642 X (125 X 64 X 3) X 10^{-3} =15.084 msec.
CT2	Remaining samples (125 X 61 X 3) are transferred from Processor-M to Processor-B in 14.446 msec.
PT2 &3	On processors A and B, JPEG algorithm steps 2 to 7 are performed (see section 2), on each part of the image. Execution time on Processors A and B for performing PT2 and PT3 =1200 msec. This value has been obtained by executing the tasks on a single transputer and recording the time with the help of the system clock.
CT3&4	After performing each task, encoded image is transferred back to the main processor. For optimum quality the encoded image consists of four times less samples than in the source image. Thus, encoded image comprises 125 X 64 X 3 / 4 = 125 X 16 X 3 samples

The communication time for transferring encoded image from Processors A and B to the Processor-M = 4.949 msec. This value has been obtained from the equation {1}.

PT4 Main processor collects encoded image from Processors A and B transfers it into an output JPEG file with header/marker. Encoded samples are directly stored in the JPEG output file on the main processor. Thus, execution time for task 4 is considered to be zero.

The execution schedule of JPEG algorithm on a three processors is shown in the form of Gantt chart in figure 4. A Gantt chart [3] essentially shows the scheduling of various tasks on the time axis.

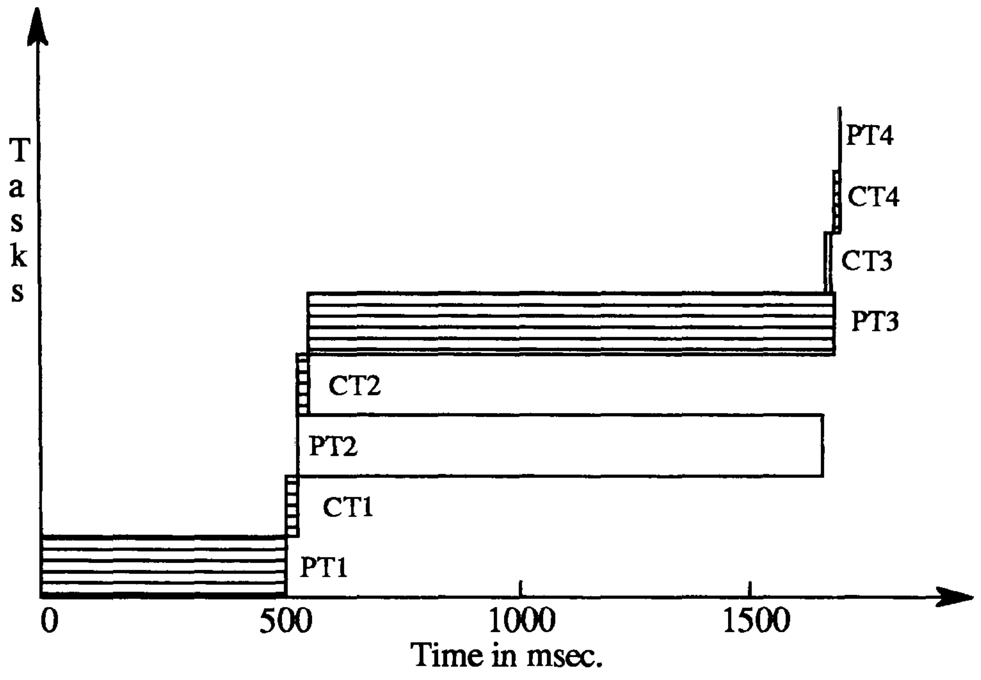


Figure 4: Gantt chart for a three processor network

From the Gantt chart of figure 4 we can clearly see the tasks that contribute to the total execution time. These tasks are shown with hash lines inside the bars representing the tasks. Other tasks overlap in execution and therefore do not contribute to this calculation.

Total time taken by three processors		=	t(PT1) + t(CT1) + t(CT2) + t(PT3) + t(CT4)		
		=	500 + 15.084 + 14.446 + 1200 + 4.949 + 0		
				=	1734.929 msec.
Time taken on a single processor		=	500 + 1200 + 1200 + 0	=	2900 msec.
Speed up of parallel processor	S_3	=	2900/ 1734.929	=	1.67
Efficiency of parallel processor	E_3	=	1.67 / 3	=	0.557

3.3b Implementation of JPEG algorithm on seventeen transputers

The three transputer network is the smallest that can be utilized meaningfully. A seventeen transputer network is the largest available to us. In this section we consider mapping of the JPEG algorithm on a network of seventeen transputers.

To map the JPEG algorithm on this network the source image can be split into sixteen parts as shown in figure 5. The sixteen image parts are labelled as **a**, **b**, .. , **p**. Each part of the source image can be processed on a transputer. This can be mapped quite elegantly onto a 4X4 array of transputers as shown in figure 6.

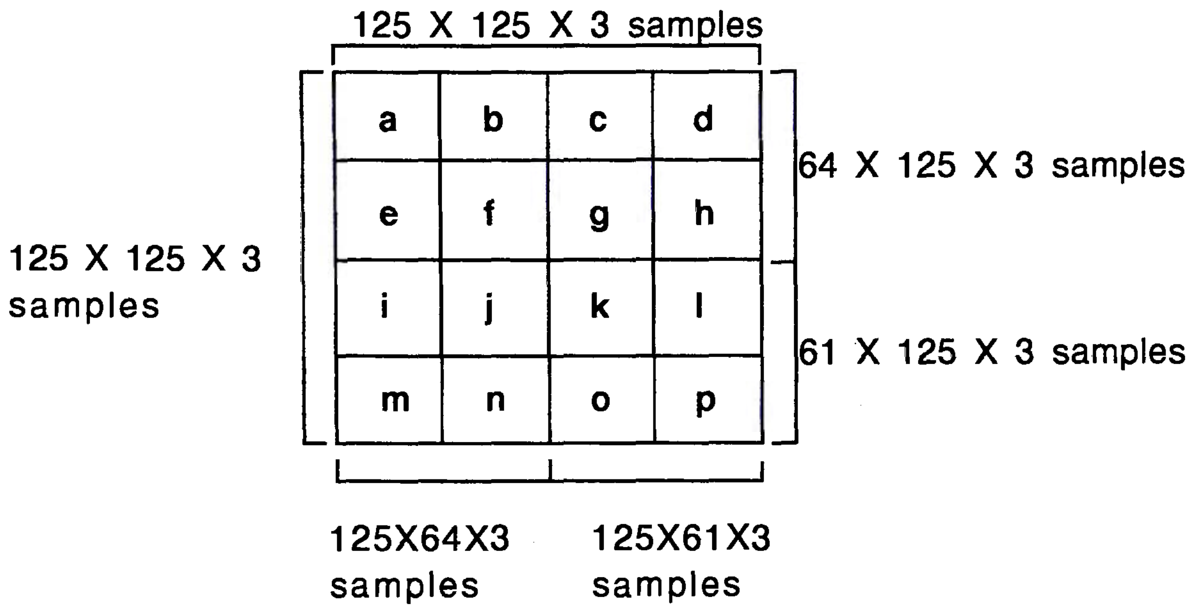


Figure 5: Source image

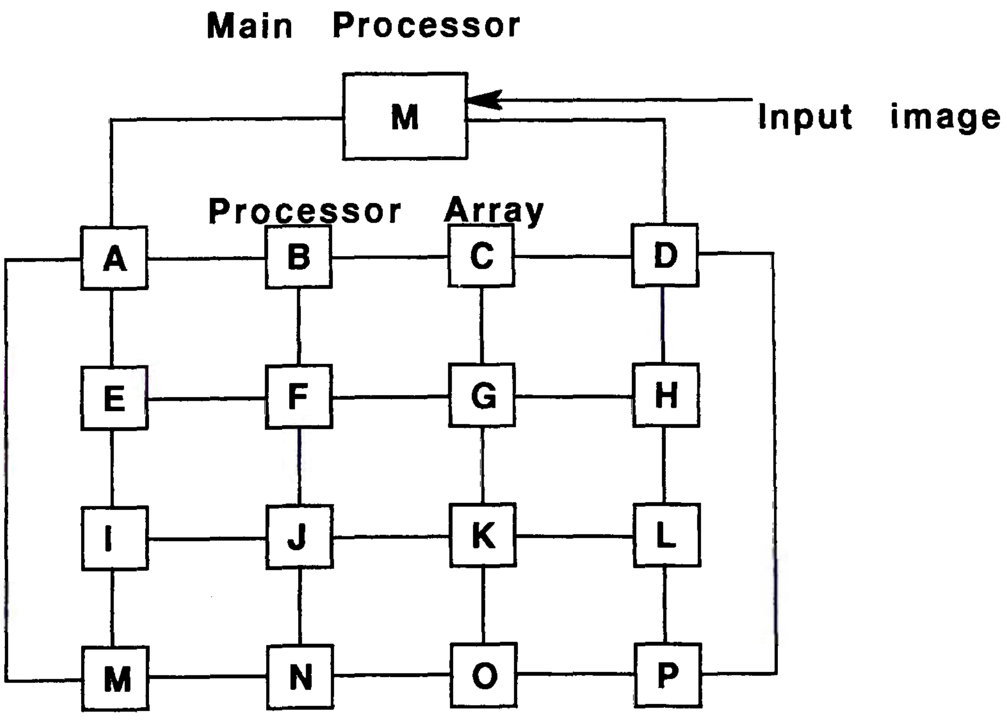


Figure 6: Mapping the source image onto a transputer network.

Tasks required for executing the JPEG algorithm on a seventeen transputer network are listed below. The task graph of figure 7 shows the relationship between these tasks and the paths used for first distributing the uncompressed image on the network, and then for collecting the compressed image parts. We use the notation described in section 3.2 for task identification and specification of the execution times.

Task number	Function	
PT1	Input image is divided vertically into two parts, and other initialization functions such as setting up the coding tables are performed on the main processor.	t(PT1) = 500
CT1,2	The two image halves are transferred to processors A and D respectively. Part **a** of the image is retained on Processor-A and the remaining image parts of the half image now on Processor-A are transferred to other processors as per the task graph of figure 7.	t(CT1) = 15.084 and t(CT2) = 14.446
CT3	Image parts **e**, **i** and **j** are transferred from A to E.	t(CT3) = 6.693
CT4	Image parts **b** and **f** are transferred from A to B.	t(CT4) = 4.95
CT5	Image parts **m** and **n** are transferred from A to M.	t(CT5) = 4.622
PT2	At this stage Processor-A becomes free to start processing its image part **a**.	
CT6	Image parts **i** and **j** are transferred from E to I.	t(CT6) = 4.95
CT7	Image part **f** is transferred from B to F.	t(CT7) = 3.205
CT8	Image part **n** is transferred from M to N.	t(CT8) = 3.042
CT9	Image part **j** is transferred from I to J.	t(CT9) = 3.2
CT10	Image parts **h**, **l** and **k** are transferred from D to H.	t(CT10) = 6.36
CT11	Image parts **c** and **g** are transferred from D to C.	t(CT11) = 4.95
CT12	Image parts **p** and **o** are transferred from D to P.	t(CT12) = 4.47
CT13	Image parts **l** and **k** are transferred from H to L.	t(CT13) = 4.78
CT14	Image part **g** is transferred from C to G.	t(CT14) = 3.2
CT15	Image part **o** is transferred from P to O.	t(CT15) = 3.042
CT16	Image part **k** is transferred from L to K.	t(CT16) = 3.205
PT2 to PT17	Each processor performs compression on its image part. This takes 150 msec.	i.e. for n = 2 to 17 t(PTn) = 150
CT17 to CT32	Encoded image parts are transferred back to the main processor as shown in the task graph of figure 7. The time taken for each communication task is shown on the arc representing the same.	
PT18 to PT28	The main function of these tasks is to collect the encoded image parts. The execution time required for these tasks is negligible.	

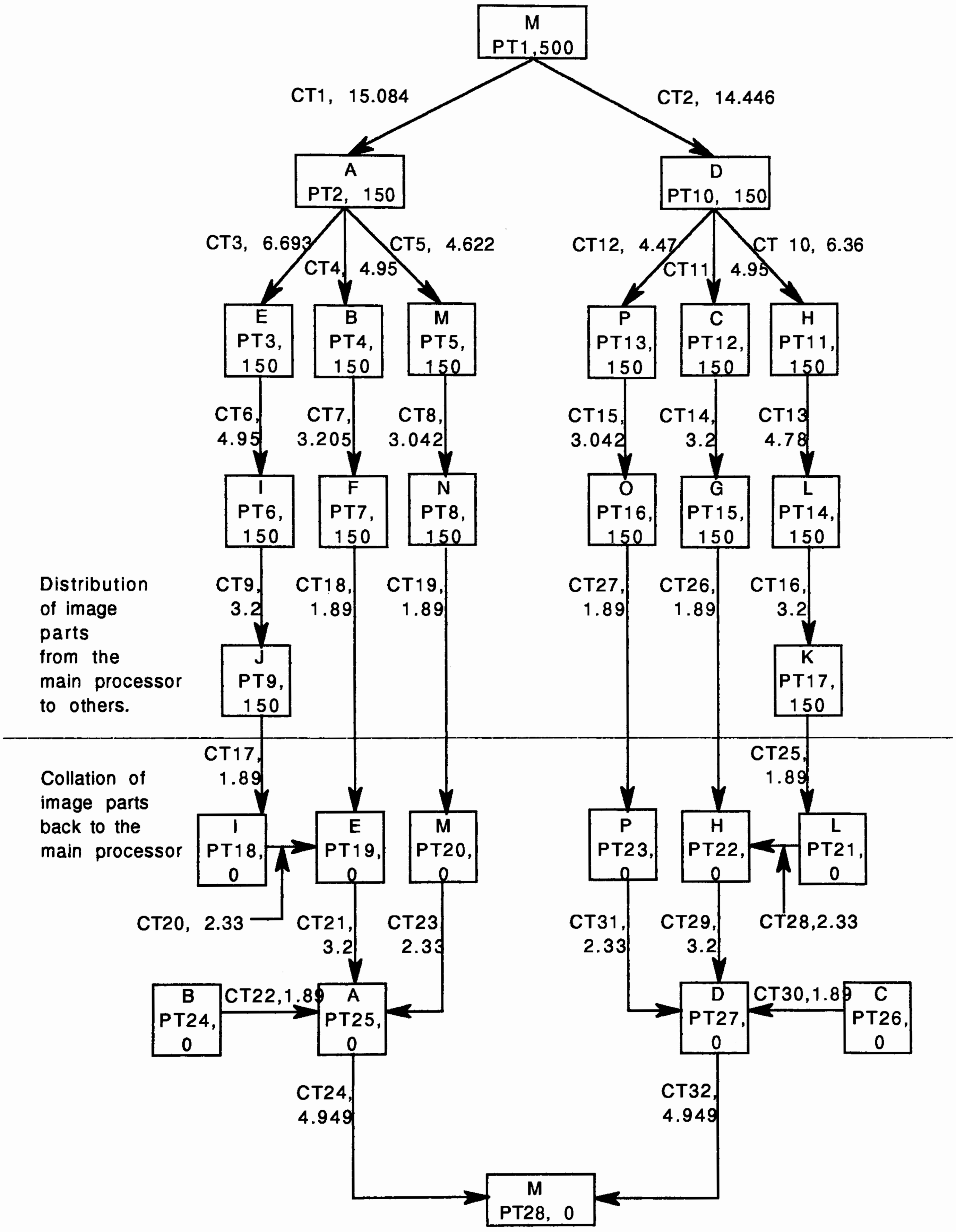
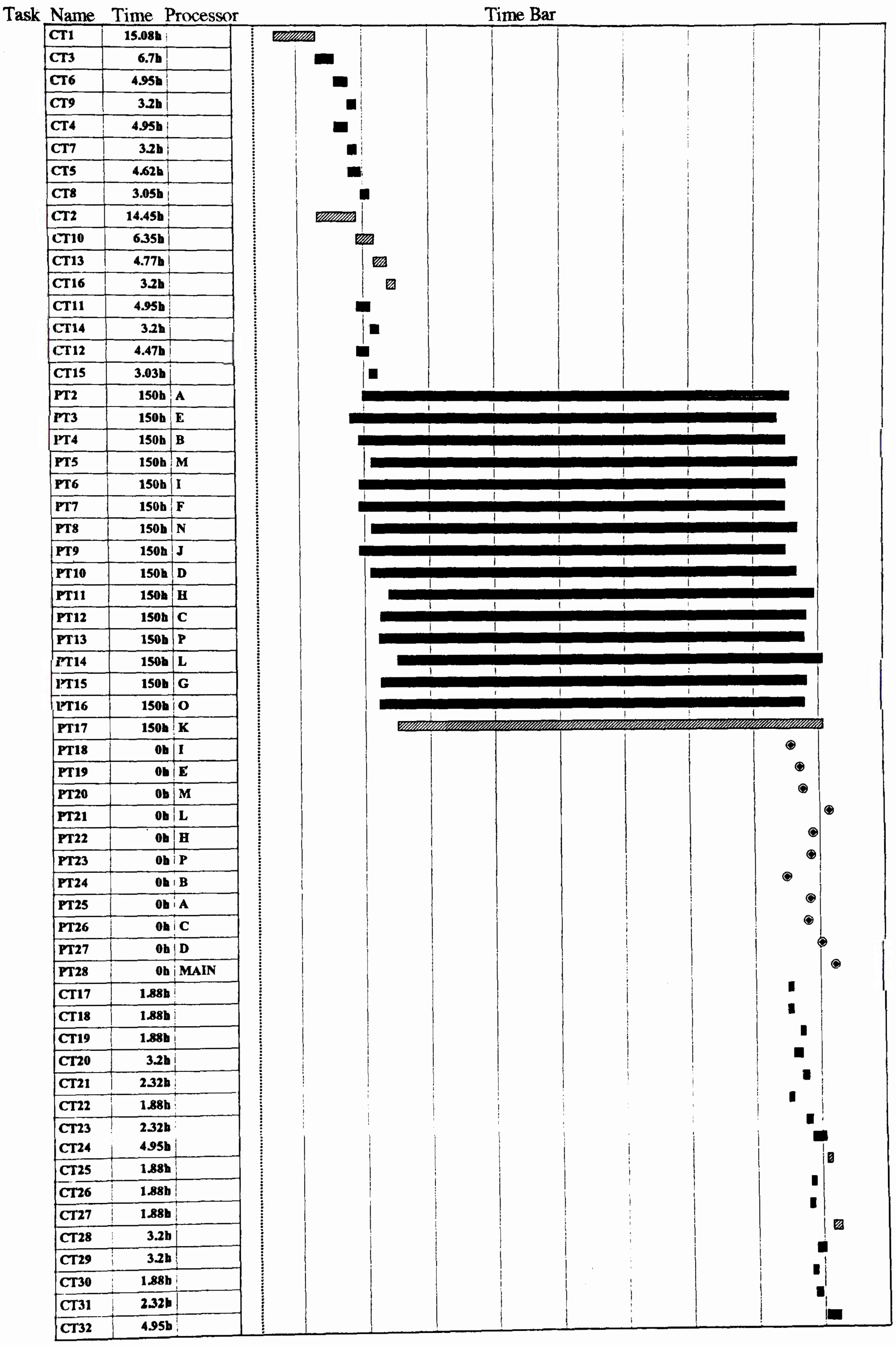


Figure 7: Task graph for seventeen processors



Task Name	Time	Processor
CT1	15.08h	
CT3	6.7h	
CT6	4.95h	
CT9	3.2h	
CT4	4.95h	
CT7	3.2h	
CT5	4.62h	
CT8	3.05h	
CT2	14.45h	
CT10	6.35h	
CT13	4.77h	
CT16	3.2h	
CT11	4.95h	
CT14	3.2h	
CT12	4.47h	
CT15	3.03h	
PT2	150h	A
PT3	150h	E
PT4	150h	B
PT5	150h	M
PT6	150h	I
PT7	150h	F
PT8	150h	N
PT9	150h	J
PT10	150h	D
PT11	150h	H
PT12	150h	C
PT13	150h	P
PT14	150h	L
PT15	150h	G
PT16	150h	O
PT17	150h	K
PT18	0h	I
PT19	0h	E
PT20	0h	M
PT21	0h	L
PT22	0h	H
PT23	0h	P
PT24	0h	B
PT25	0h	A
PT26	0h	C
PT27	0h	D
PT28	0h	MAIN
CT17	1.88h	
CT18	1.88h	
CT19	1.88h	
CT20	3.2h	
CT21	2.32h	
CT22	1.88h	
CT23	2.32h	
CT24	4.95h	
CT25	1.88h	
CT26	1.88h	
CT27	1.88h	
CT28	3.2h	
CT29	3.2h	
CT30	1.88h	
CT31	2.32h	
CT32	4.95h	

Figure 8: Gantt chart for task allocation on a seventeen transputer network

The Gantt chart of figure 8 was drawn by using Microsoft Project - a project management software. The first column lists the task name. The second column contains the execution time. The units shown as 'h' stands for hours. Microsoft project does not allow smaller time units to be specified as fractions (for obvious reasons). We have entered data by equating one hour to one msec. The tasks on the critical path, i.e. those which contribute to the total execution time, are filled with hatch lines. To improve the resolution of the Gantt chart, PT1 which takes 500 msec on the main processor, has been omitted from figure 8.

Time taken on a single processor		= 500 + 150 X 16	= 2900 msec
From Gantt the total time on the seventeen transputer network		= 500 + 198.93	= 699 msec
Speed up of the seventeen processor network	S_{17}	= 2900 / 699	= 4.15
Efficiency of the seventeen processor network	E_{17}	= 4.15 / 17	= 0.244

4. Conclusion

To reduce the number of bits used to store and transmit images, compression techniques are often used. JPEG is an international organisation which has developed a standard for digital image compression.

JPEG algorithm can be implemented on a network of processors connected in an array topology. With a network of three transputers a speedup of 1.67 can be obtained. This gives an efficiency of 55.7% for each processor. On a network of seventeen processors, with sixteen processors connected as a 4X4 array, a speedup of 4.15 can be obtained. The average efficiency of each processor is only 24·4%. Thus, as we go from a singe processor to three processors, and then to a seventeen processor network, we can reduce the total time to solution, but the level of processor utilisation also falls.

BIBLIOGRAPHY

[1] Gregory K. Wallace, " The JPEG Still Picture Compression Standard." IEEE Transaction on Consumer Electronics, Vol. 38, No. 1, Feb. 1992, pp. xviii-xxxiv.

[2] Davis Ian, Perihelion Software Limited, "Helios operating system", 1992.

[3] Ted G. Lewis and Hesham El-Rewini, "Introduction to Parallel Computing", Prentice Hall, 1992.