The Parallel EM Algorithm and its Applications in Computer Vision*

P.E. López-de-Teruel, J.M. García and M. Acacio
Dpto. de Ingeniería y Tecnología de Computadores
University of Murcia
Campus de Espinardo, s/n 30080 Murcia (Spain)
{pedroe,jmgarcia,meacacio}@ditec.um.es

Abstract *In this paper we propose several applications of the EM algorithm –a well-known algorithm for parameter estimation in generic statistical problems– in the different levels of a computer vision system, and discuss its relatively straightforward and efficient implementation in parallel distributed memory environments, under the message passing paradigm. We show, at least, an algorithm for each level of a generic computer vision system (low, medium and high level vision) with the common basis of the EM algorithm applied in parameter estimation for mixture statistical models. We present an efficient implementation in parallel for all of them under a common scheme. Also, we show an evaluation of a medium-level vision EM application in a cluster of PCs. The results we have obtained in terms of both speedup and execution time are very good. Moreover, the paper gives us the guidelines for extending the method for related problems, and, in general, for many applications with mixture parameter estimation with EM as its main task.*

Keywords*:* Parallel EM Algorithm, Computer Vision, Message Passing Paradigm, Clusters of PCs.

1 Introduction

Computer vision has become a very active research area in the last years, due to its wide range of applicability in several fields of science and engineering [3]. A widely accepted fact in the field is that most of the computational tasks involved in artificial vision systems are very intensive computationally, but, at the same time, it is also well known that their repetitive nature through all the image data make them also easily parallelizable.

The EM algorithm, on the other hand, is a technique for parameter estimation of generic statistical distributions in presence of incomplete data. However, since several years ago, the EM algorithm has been used in computer vision systems for several different tasks, such as *image motion*, *segmentation* or *visual learning*. But application of the EM uses to be computationally very intensive. Thus, with the arrival of cost-effective parallel systems, it is interesting to investigate its parallelization in such computer vision systems.

Lately, clusters of workstations [1] are being used as a good alternative to traditional parallel machines (usually much more expensive). The performance of such clusters can be very satisfactory, if the communication shared medium does not suppose a severe bottleneck [1,12]. Besides the low cost, one of the main advantages of NOWs (Network of Workstations) is its extensibility with dedicated hardware for specific applications, such as data acquisition cards for vision, control, and so on. The reason is that its availability in the market is only limited by the availability for each individual computer in the cluster (for example, PCs with standard PCI and ISA slots, whose range of possibilities is very wide). This suggest the possibility of using this kind of parallel machines in computer vision, with a standard PC visual data acquisition card, in order to obtain good speedup results.

* This work has been supported by the Spanish CICYT under grants TIC97-0897-C04-03 and TIC98-0559.

In this paper, we propose a generic parallel implementation of the EM algorithm. We also justify that this kind of distributed parallel machines (NOWs) are very adequate to implement different variations of this parallel EM algorithm applied in the specific field of computer vision, at a much lower cost than traditional MPPs. To do that, we discuss some experimental results in a cluster of workstations with Fast Ethernet as its shared medium.

The organization of the paper is as follows: Section 2 briefly summarizes the foundations of the EM algorithm, and gives a generic scheme of an efficient implementation for a parallel machine. Section 3 describes several applications in the three different levels of computer vision (low, medium and high level vision). Section 4 shows some performance results in a cluster of workstations. Finally, in section 5 some conclusions and future work are exposed.

2 The EM algorithm and its parallel implementation

EM is a technique for estimation of the parameters of generic statistical distributions in presence of incomplete data; that is, when estimation of the parameters must be done taking as input multidimensional samples (i.e., vectorial data) where some of the components are missing [4,10]. The main goal of EM is to obtain the estimated parameters that give *maximum likelihood* to the input (incomplete) data. The basic idea underlying the EM algorithm is to iterate through a series of *expectation* (E) and *maximization* (M) steps in which the solution to the statistical problem (i.e., the estimation of the parameters of the model) is progressively refined until convergence.

Mixtures are perhaps the statistical models where the technique is most straightforwardly applied, and where very good results can be obtained at reasonable computational costs. *Mixture models* [13] are statistical distributions composed by a weighted sum of elemental densities of any kind, called the *components* of the mixture, and that use to be, but are not limited to, typical distributions such as gaussians, uniforms, and so on. On them, the unknown data on each input vector is assumed to be the index of the individual component to which the example belongs.

Roughly speaking, we could summarize the operation of EM in mixture parameter estimation as follows: In the *E-Step*, the probabilities of each point in the sample of belonging to each component of the mixture are computed. This is the expectation of the unknown data. And, in the *M-Step*, a new set of parameters for the individual components (for example, mean vector and covariance matrix, for the specific case of gaussian components) are calculated, using the previously computed probabilities as weights in the calculation. This way, we set the number of components, randomly initialize their associated parameters, and begin the iterative E and M steps loop until convergence to a stable solution. The algorithm experimentally shows very good adaptability and convergence properties, even with poor initial configuration of the parameters.

Obviously, the implementation slightly varies depending on the characteristics of the specific application: The type and dimension of the input samples, the chosen elemental model for the individual components of the mixture, and so on. But the fact that it involves the calculation of a big table of data (with a size of $M \times N$, being M and N the number of components and input samples, respectively), with several data in each cell (weights, parameters,…), is common to multiple applications of the technique. In this way, it can be efficiently parallelized following a scheme that is common to many parallel programs [8], and that is outlined in figure 1.

In it, we can observe how the distributed processes are strongly uncoupled, in the sense that the communications needed among them are very few in relation to the computations involved, and, besides, they are very synchronized, that is, they occur at even times for all the processes. This way, each process does not need to be idle waiting for the rest of processes, and, at the same time, the computational load is well balanced.

Figure 2 shows graphically how we take advantage of this symmetry in the computation and communication scheme, in order to achieve the maximum possible speedup. Observe that the data parallelism is exploited twofold:

- First, a distribution of the processing by columns (i.e. samples) is done, in the expectation step. This distribution avoids the need of

Input: Array of N input samples.
Output: Table of k parameters of each of the M components of the mixture.
Initialization
Generate an initial random set of parameters for the mixture.
Repeat
E-step:
Compute the $q_{e,i}$ values, for $e=1..M$, $i=1..N$, probabilities of each input sample i of having been generated by the mixture component e, using the current values of the parameters. Distribute the computing between processors by columns (samples), as, in each step, a column normalization has to be performed (this avoids the needing for communications in this step).
M-step:
Recompute the parameters of each component of the mixture using the previously calculated values of $q_{e,i}$ as weights, and the standard estimation method for the elemental kind of component. Distribute the computing by rows (components). This demands a communication between processors, as the $q_{e,i}$ must be broadcasted and distributed among them for this recomputation to be performed. Anyway, in many cases the communication size can be dramatically reduced if some intermediate parameters are precomputed locally, that are subsequently used for the computing of the final values. This can be done in many kind of elemental components, and indeed in the most usually applied in practice (gaussians, uniforms, etc.).
Until solution does not significantly change.

Figure 1. Generic parallel implementation of the EM algorithm for mixtures (message passing paradigm).

communications in the Q matrix computation and in its posterior normalization. Also, the intermediate parameters for the subcomponents parameters can be still computed with this data distribution, without communications.

- Second, in the M-step, after distributing the previous results among all the processes, a new data partition is done, by rows (i.e. components), to compute the final parameters of the mixture components.

In this way, as we said before, the computational load is balanced as much as possible, and the speedup is expected to be maximized. In the results section we will give some experimental results to confirm these speedup predictions.

3 Computer vision applications of the EM algorithm

In this section we will describe three direct applications of the EM algorithm for mixture parameter estimation in the three levels (low, medium and high level vision) of a computer vision system. The technical details of each implementation are obviously different, because of the different objectives of each level, but all of them have the same underlying parallel implementation of the EM. The use of this algorithm in computer vision systems is not new; other researchers have applied it in several problems related to this field. See, for example, [5] (EM applied to image motion) and [11] (EM applied to visual learning).

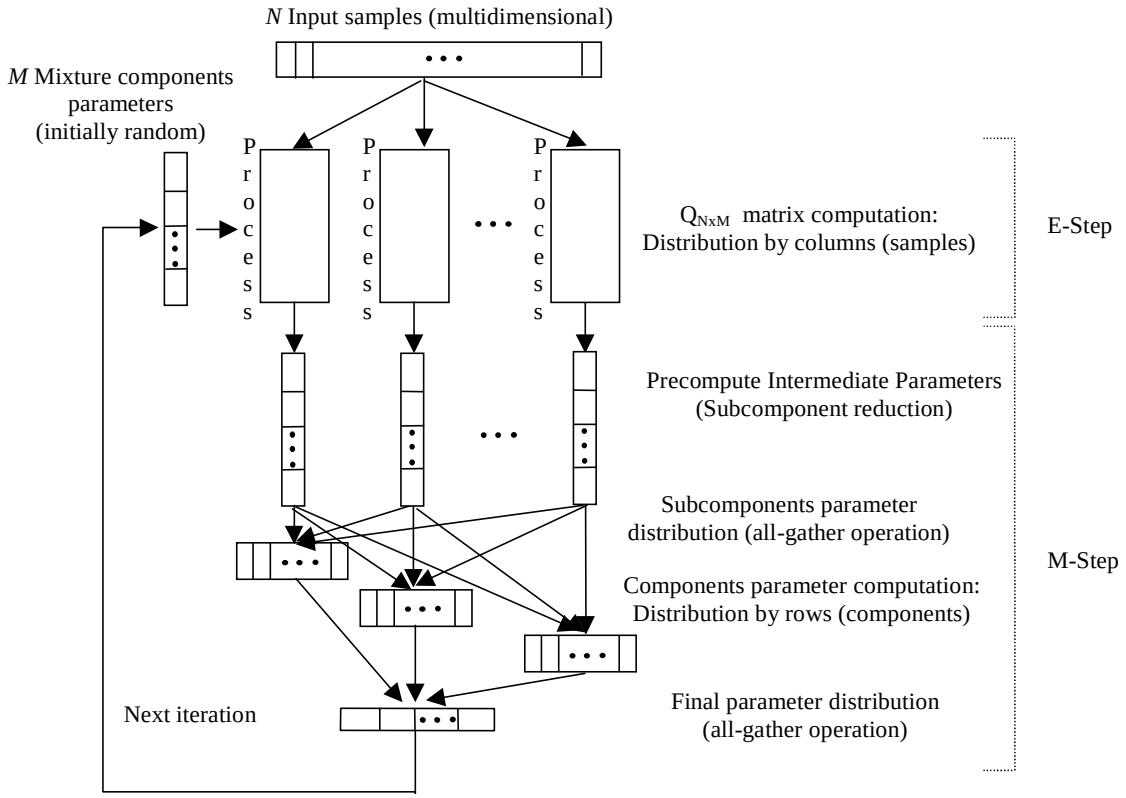

Figure 2. Generic parallel implementation of the EM algorithm for mixtures (graphical representation).

3.1 Low Level Vision

The main aim at this level is to detect and isolate regions in the input image that presumably belong to different objects, mainly due to light intensity changes. This process is usually known as *segmentation*, and many techniques exist to try to solve it [3,6]. In our EM environment, we construct a mixture of gaussians to probabilistically model the distribution of the gray intensity (or red, green, and blue, if we use color images) values of the pixels in the image, together with its *X* and *Y* positions (see [9] for details). When the (initially randomly chosen) set of parameters of the mixture converges, each component is expected to be *collecting* all those pixels that belong to the same object (especially those components with final greatest *a posteriori* probability), and can be used as rough low level object detector. Observe that, here, the components of the mixture could be, for example, multivariate gaussians (with three dimensions for B/W images, or five for color images), whose individual parameters are the mean vector and the covariance matrix, while the input samples are the (*R,G,B,X,Y*) vectors, directly obtained from the array of image pixels.

An example of application of this technique is shown in figure 3 (middle). In it, we can see how a good bilevel thresholding is performed, by simply using the EM algorithm with two components, one for catching the objects and the other for the background. The system was also able to dynamically adapt to changing lighting conditions, with a small computational effort, exploiting its inherently adaptive nature. Another advantage is that, since the number of random sample points in the input image can be chosen arbitrarily, we also have a chance to accelerate the segmentation in critical situations by simply reducing that number of points, at the cost of a probably coarser output.

The EM parallelization formerly exposed can be straightforwardly applied here, including the intermediate parameter computing facility (perfectly applicable with multivariate gaussians). The final obtained speedup uses to be very high, approaching more and more to the ideal value as the image size grows.

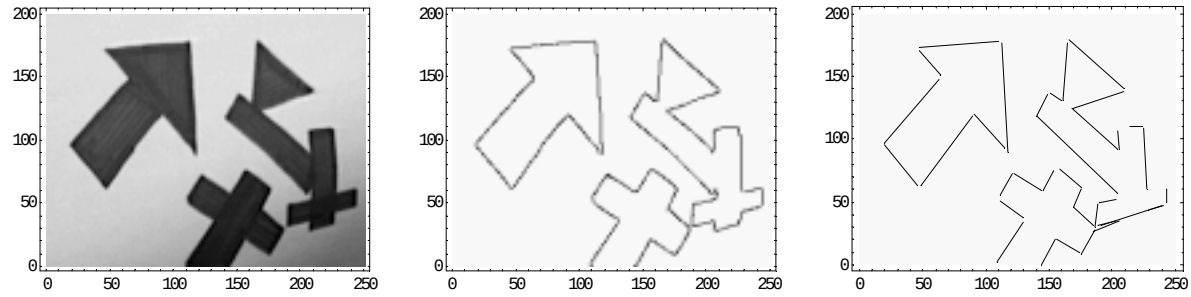

Figure 3. The EM algorithm in low and medium level vision. Original image (left), thresholded image –low level process– (middle), and segments found in the image –medium level process– (right).

3.2 Medium Level Vision

At this level, some kind of elemental structure needs to be found in the previously segmented image. Identifying and locating segments in a scene is, for example, a useful intermediate step for many computer vision applications, where further high level processing is needed, such as 3D interpretation or shape recognition under difficult conditions (overlapping objects, noise, and so on).

Here, our contribution is to implement a parallel version of the EM algorithm with a new kind of elemental density of the mixture especially designed to model the shape of a straight line in *edge binary images* (obtained in the low level process). Then, the parallel main loop of E and M steps is executed on the sample of edge points –(x,y) position vectors–, and a good estimation of the position, size and orientation of the set of segments that best describe the scene is obtained as output (see [9] for implementation details). Again, parallelization is straightforward, and, the speedup for common problem sizes, as will be shown in the results section, very satisfactory.

An example of application is also shown in the aforementioned figure 3 (right). Here, the previously segmented image is used as the input edge image in this new EM version for segment detection and tracking. Observe the accurate parametrization of the output image, in which the amount of information has been reduced significatively, by extracting the structure of the input image in terms of straight segments. This description is potentially very useful for a posterior higher level interpretation.

3.3 High Level Vision

Here, the main aim is to describe *graph models* of more or less complicated objects to detect in the scene, and then to locate corresponding *data graphs* in the input image (previously preprocessed in order to extract features such as lines, corners, and so on). The solution must unify the task of estimating the *transformation geometry* –that copes with scale, rotation and translation of 3D objects in the scene–, and the *point correspondence matches* between graphs, also unknown a priori. Here, we use a variation of the EM proposed in [2], whose main idea is to manage a *soft-assign* matrix S that probabilistically describes an assignation between the graph and the data nodes, and that iteratively converges to a pure assignation matrix as the method operates. With an initial assignation of the s_{ij} elements of the S matrix, the algorithm finds the transformation matrix ϕ that, in a probabilistic sense, and given the current *soft assignation* matrix S, best describes the geometric transformation of the model graph to match the data graph (*M-Step*). Then, and using this newly obtained ϕ matrix, it is used to compute the new s_{ij} (*E-Step*), and the whole procedure is repeated until convergence.

3.4 General issues

In all the three vision levels, the good property of the EM to adapt to smoothly changing input data (as objects in scene slowly move) is optimally exploited, by simply letting the main EM loop to iterate updating the current solution, and fitting to the new data. Also, the algorithm shows its ability for detecting and tracking more or less complex structures in the input, especially at the medium and high levels, and all of

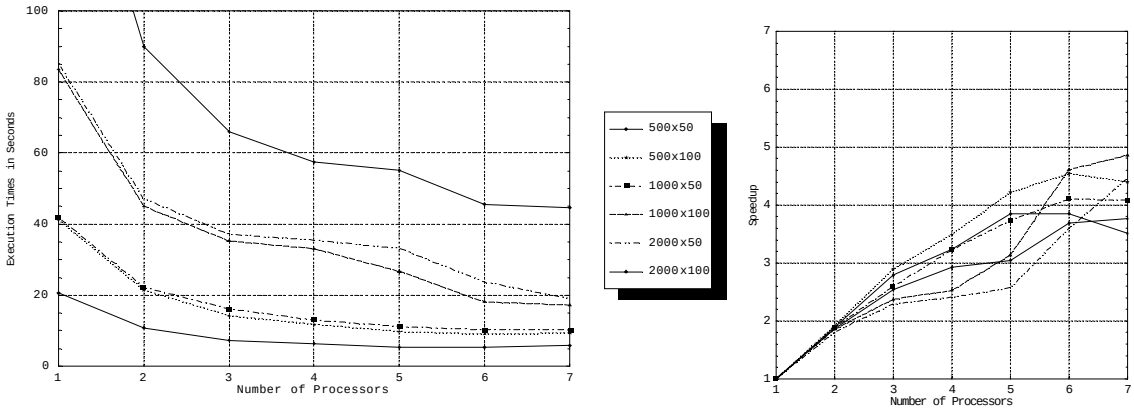

Figure 4. Execution time vs Number of processors, and corresponding Speed-up vs Number of processors graphics for the EM version of the line detection algorithm described in section 3.2, for 2 different problem sizes N×M (N=number of input edge points; M=number of segments to detect (components)). The execution times correspond to 100 iterations of E and M steps.

it with a high potential degree of implicit parallelism.

4 Experimental Results

In this section we show some experimental results that give us an idea of the overall speed up that can be obtained with our proposed parallel computer vision system with clusters of PCs and the general parallel EM scheme as the underlying algorithm. To show the performance evaluation, we choose, for example, the medium level vision EM adapted algorithm, to detect and track segments in a binary edge image. Figure 4 pictures the reduction in the global execution time, for several problem sizes, in our cluster of PCs. This cluster is constituted by 7 Intel Pentium 200 MHz processors, with 64 MB of RAM and 256 KB of cache each one, with a Fast Ethernet 3Com (100 Mbps) as the communication network, and with MPICH 1.0.13 for Linux as the communication library. MPICH is a particular implementation of the standard MPI (Message Passing Interface) library [7]. The performance of MPI in this cluster of workstations was extensively analyzed in [12], and its adequacy for executing our parallel EM algorithms for computer vision is completely justified by the obtained results.

As it can be observed in the figure, the overall reduction in the execution time obtained is more than acceptable as we increment the number of processors, mainly due to the reductions in the size of communication using the remarkable property of intermediate parameter calculation in the M-Step, commented in section 2. As an immediate consequence, the global processing time for each frame gets more and more reduced as we increment the number of processors. This fact makes the technique very adequate to be implemented in real time systems, where small response times may be crucial.

5 Conclusions and future work

Parallel computing on clusters of PCs has very high potential, since it takes advantage of existing hardware and software. Performance tests of the implementations show that they can be comparable to many existing parallel machines for some application problems. So, clusters of workstations can be considered as a good alternative to parallel computers.

Computer vision, on the other hand, is an interesting field in which it seems very adequate to take advantage of parallelism. The algorithms and techniques used in this kind of systems use to be very data intensive in both data storage and processing time, while the computations are often very uncoupled. This fact makes clusters of PCs ideal to implement this kind of systems.

In this paper, we propose a programming skeleton for the parallel EM algorithm applied to generic mixtures, and its application in several levels of a parallel machine vision system. Nevertheless, this scheme has a wide range of applicability not only in machine vision, but

also in many other data intensive applications in which its inherent parallelism can be efficiently exploited (generic statistic inference, machine learning, statistical treatment of uncertainty, and so on). Also, this paper illustrates how parallel programming with MPI in clusters of workstations can reach very good cost/benefit ratios, even in environments with low response time required such as real-time computer vision. Finally, and as a future work, we would like to integrate and evaluate the various levels of the vision system in a complete real application, such as visual navigation and 3D object recognition, in order to show the high potential of this versatile and powerful statistical method, when it is efficiently parallelized.

References

[1] T. E. Anderson, D. E. Culler and D. A. Patterson. *A Case for NOW*. IEEE Micro, vol. 15, nº 1, pp. 54-64, 1995.

[2] A.D.J. Cross and E.R. Hancock. *Graph Matching With a Dual-Step EM Algorithm*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, nº 11, pp. 1236-1253, 1998.

[3] E. Davies. *Machine Vision: Theory, Algorithms and Practicalities*. Academic Press, 1996.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin, *Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm*. Journal of the Royal Statistical Society, Series B, vol. 39, pp. 1-38, 1977.

[5] C.M. Fan, N.M. Namazi, and P.B. Penafiel. *A New Image Motion Estimation Algorithm Based on the EM Technique*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, pp. 348-352, 1996.

[6] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[7] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. MIT Press, 1994.

[8] B. Lester. *The art of parallel programming*. Prentice Hall, 1993.

[9] P.E. López-de-Teruel and A. Ruiz. *On Line Probabilistic Learning Techniques for Real Time Computer Vision*. Proceedings of the Learning'98, Getafe (Madrid), Spain, 1998.

[10] G.J. McLaghlan and T. Kishnan. *The EM Algorithm and Extensions*. John Wiley, 1997.

[11] B. Moghaddam, and A. Pentland, *Probabilistic visual learning for object representation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pp. 696-710, 1997.

[12] J. Piernas, A. Flores, and J.M. García. *Analyzing the Performance of MPI in a Cluster of Workstations Based on Fast Ethernet*. 4th European PVM/MPI Users' Group Meeting, LNCS, Springer-Verlag, Vol. 1332, pp. 17-24, 1997.

[13] R.A. Redner and H.F. Walkner. *Mixture Densities, Maximum Likelihood Estimation and the EM algorithm*. Society for Industrial and Applied Mathematics (SIAM) Review, vol. 26, pp. 195-239, 1984.

Conference: PDPTA’99.

Paper identification number:
321P (Regular Research Article).