Parallel Image Understanding on a Multi-DSP System

M. Fikret Ercan

School of Electrical and Electronic Engineering, Singapore Polytechnic, 500 Dover Rd., S139651, Singapore
mfercan@sp.edu.sg

Abstract. A course-grain multiprocessor architecture, based on an array of digital signal processors (DSPs), is presented to demonstrate the possibility of a parallel implementation of image-understanding algorithms. Aerial image understanding is investigated as an application. The system is designed to exploit temporal and spatial parallelism. A good speed-up was obtained for low- and intermediate-level operations. However, the speed-up for high-level operations was poor because of processor idle times, as the number of objects to be processed at higher level tends to be small. DSPs performed well as processing elements for number-crunching operations, but the performance was not so good for implementing symbolic operations.

1 Introduction

Image understanding systems produce a description of the scene in a given image by extracting meaningful features from the image and matching them with models. Document image understanding, face recognition, aerial image understanding, medical image analysis are some of the applications areas. A vast amount of research has been made in the last decade in which a combination of techniques ranging from image processing to artificial intelligence were employed (see for instance [11,17] for interpretation of aerial images, [7,10,14,13,15] for medical image analysis and [2,9] for video data mining). A critical issue in image understanding applications is the computation complexity. For some applications, such as face recognition and aerial image understanding, operations have to be completed in real-time. On the other hand, applications like image document retrieval or medical image analysis operations have to be completed within a short period of time. A well known approach to overcome high computation cost is to employ multiprocessor systems. In early studies, tailor made computer architectures are designed to address computational requirements of image understanding algorithms. In these systems a combination of fine and coarse grain parallelism exploited and various other design ideas were experimented such as reconfiguration, multilayer hierarchy etc. [1, 3, 8]. However, a major drawback was inflexibility and difficulty of programming. The latest advances in microprocessor and computer architecture made it possible to build a coarse grain multiprocessor system, which is flexible and easy to program. In this paper, we employ such system and investigate parallel image understanding algorithms for a coarse grain multiprocessor.

O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, LNCS 4706, Part II, pp. 1–12, 2007.

The system employed in this study is homogenous, that is the same processor is used for low, intermediate and high level processing. The processing element (PE) of the system is Analog devices SHARC DSPs (TM-001 at 250MHz). Configuration of the system allows both spatial and temporal parallelism to be exploited. Each processor holds 6 Mbytes of local memory and 128Mbytes memory is shared between the four DSPs. Figure 1 shows a block diagram of the system.

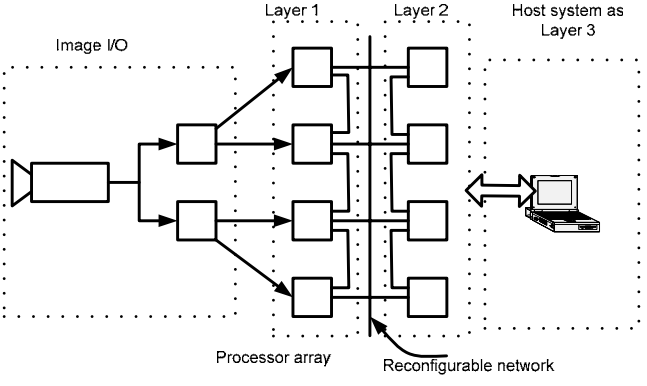


Fig. 1. A block diagram of the system

An example problem, aerial image understanding, which is a widely investigated problem [8,17], is implemented as an application. Although, the methods and the computing platforms presented in these works are different than the one presented here, these studies have made a good reference for the experiments conducted here. The other reason for selecting this problem is that it does not require expert knowledge as compared to more specific problems such as medical image understanding.

Unfortunately, there is no common method to solve image understanding problem. In early studies, the black-board method is introduced where knowledge-based processes communicate through a central message utility. It is difficult to realize top-down model directed feature extracting with this method [16]. In another study a data parallel approach, namely geometric hashing, was introduced for high level image processing [6]. This method is computationally intensive and requires a large memory space for the hash table. In our approach, each object is treated as an independent agent and they are distributed to the processors of the system which is more suitable to implement in coarse grain architectures. The algorithm builds a spatial network of objects independently on the individual processors of the system.

In the following, feature detection and parallel image understanding algorithms will be presented in detail. It is followed with the experimental results and a discussion on the speed-ups achieved. Conclusions are given in the last section.

2 Parallel Image Understanding Algorithm

Interpretation of aerial images is a difficult problem considering the complexity of an aerial picture. In the example application, a description of the scene is constructed by

using man-made objects, such as sport fields and houses detected in the image. As the objective of the application is to test the merits of the parallel system rather than to develop a full scale aerial image understanding system, we employed a small model base. The flow of operations is bottom-up first (feature extraction) then followed with a top-down analysis (search for missing object features). Integrating bottom-up and top-down flow of operations in image understanding provides a better performance since purely bottom up or top-down operations have limitations [4,5].

In our application, bottom up phase includes following operations: feature extraction (low level), feature grouping (intermediate level) to form non-pixel based representation of the objects, and description of the scene (high level) using the spatial relationships between these objects. Top-down phase of the operations are activated to search missing object features or to search hypothesized objects in the scene. Apparently, a major problem is to map these operations on a multiprocessor architecture with a minimum compromise on the performance.

2.1 Feature Extracting and Generating Object Candidates

We employed a corner detection algorithm [12] as a primary source of feature detection from the image. The algorithm detects corner points through following steps:

- Image is convoluted with a set of masks one for each possible orientation. A pixel is selected when response of at least two masks are greater than a threshold value.
- The off-edge points are eliminated in view of neighborhood values. If the response is not significant when compared to the neighboring pixels then the edge point is eliminated.
- Non-corner points are eliminated, if two half-edge orientations differ by 180º.

The algorithm employs, Gaussian based half edge detectors and it is inherently data parallel. In our application, only right angle corners are needed hence the parallel algorithm is realized as follows:

Step1: Distribute image segments to processors
Step2: Perform boundary data exchange
Step3: Perform corner detection
Step 4: Detect right angle candidates and form right angle tokens.

Detected corner points are labeled based on their half-edge pairs as shown in Figure 2. A token for each potential right-angle corner is then produced. Each token contains x, y coordinates of the corner point and its type (1 ~ 12) as shown in Figure 2.

In order to form rectangle candidates, at least three right angle corner points are required. Among all the corner features found in the image, those corner points that can form a proper rectangle are combined as a rectangle candidate. For instance, consider rectangles with 0º angle and assume that corner type 4 is the reference point. To form a rectangle, a pair of corner points such as type 1 and 3 is required. The half edges of these corner points should approximately intersect with a line drawn between two corner points. The first step of the parallel algorithm provides each processor all the corner points detected from the image. Algorithm searches for two adjacent corners for a given corner point to form rectangle candidates. The parallel corner grouping algorithm employs the following steps:

Step 1. Processors perform all-to-all broadcast and obtain all the corner tokens.
Step 2. Processors sort corner tokens based on the corner labels
Step 3. For group=1,...,3
Step3.1 Reference corner points are partitioned among the processors evenly.
Step 3.2. For each label=1,...,12
Step 3.2.1 Processors search for possible rectangle candidates based on the reference corner points and the rest of the corner points in their group.
Step 4. Perform all-to-all broadcast and merge same rectangle candidates.

The above algorithm may produce more than one representation of the same rectangle in separate processors. Therefore, at the final step a merge operation is performed to avoid repetitions. Final rectangle candidates, S_r, are submitted for high level processing.

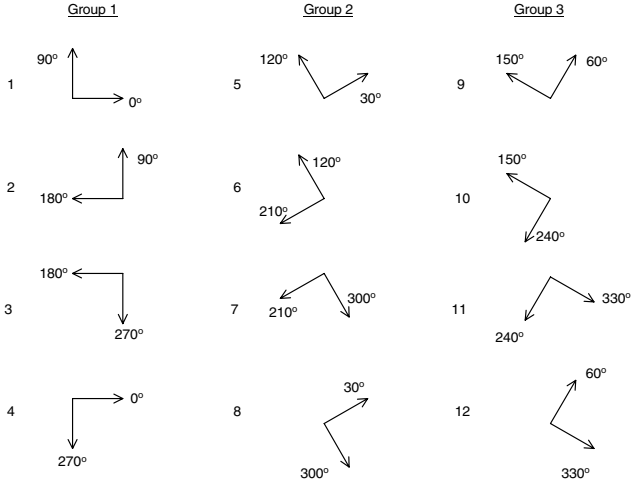


Fig. 2. Possible right angle corner points extracted by corner detecting algorithm

2.2 High Level Processing

The main objective of the high-level processing is to construct a description of the scene. However, incomplete information is one of the major problems. Therefore, an image understanding system is expected to perform with incomplete or faulty information. Among the various types of methods introduced in literature, knowledge-based analysis is a well-accepted approach. Formal logic, semantic networks and production systems are three principle methods used. In knowledge based analysis, intrinsic shortage of input information is compensated with the knowledge provided to the system. Furthermore, this knowledge is used to minimize shortcomings of feature extraction operations. The high level operation discussed in this section is

knowledge based and utilizes the knowledge about the geometric and spatial relations of the object instances.

The spatial reasoning seeks topological relations, such as adjacency, inclusion etc., and geometric relations, such as distance, bearing, direction etc., among the object features. Image understanding with this method is simply based on accumulating evidences by referring to the relations found between recognized objects. The high level processing implemented here is extended from SIGMA system [11] which is a sophisticated aerial image interpretation tool. However, in our application the interpretation process is simplified and it's more focused on the parallelization of the algorithms and realization of the interpretation process with hypothesis generation.

In order to interpret image data, a model should be selected and it is a key operation at high level processing. There are two expert modules developed for this purpose in SIGMA system and three types of knowledge bases are used: knowledge about the attributes of objects such as location, area, size and shape, knowledge about the relations between objects and knowledge to control the analysis. A recognized object initiates reasoning about its spatial relations with other objects, that is, each object instance acts as an independent reasoning agent and uses a set of production rules stored in the object class to generate hypothesis about its related objects. For instance, when a house object is recognized, hypothesis about its related objects such as a driveway, or a parallel road, etc. are generated. An object is represented with slots for links, attributes and rules. Relations between objects are established through AKO ("a kind of"), PO ("part of"), AO ("Appearance of") links. For instance, a rectangular house frame and an L-shaped house frame are related to the house frame with AKO links. An object contains slots to store attributes of the object (such as dimensions) as well as a procedure slots (such as to calculate the area of the house and store it in the area slot). These types of slots are also used to store rules that play a key role in the interpretation process. A rule slot has three parts: control-condition, hypothesis and action. The control-condition slot is a predicate that indicates when the rule can be applied. The hypothesis slot specifies a procedure to generate an expected description of a related object. Finally, the action part describes a procedure to be executed when the hypothesis part is verified.

The frame mechanism and reasoning by object instances are selected in the high-level processing for various reasons. Firstly, it is suitable for parallelization, as objects can be treated as independent agents and executed on different processors. Secondly, as object instances reason independently, establishing spatial relations between objects which reside in distant processors require minimum inter-processor communication. Thirdly, model directed feature extraction operations could be hypothesized during the analysis phase.

At the first step of the high level processing, for each candidate rectangle a matching model searched in the knowledge base using their length and width parameters. There are two types of objects stored in the model base used in this experiment, houses and sport fields. The area of a sport field is greater than a house. This simple reasoning indicates which object group that a rectangle candidate belongs to. The following predicates are used for matching:

$$\forall x\begin{bmatrix} soccerf.width \times 0.8 < Width(x) < soccerf.width \times 1.2 \wedge \\ soccerf.length \times 0.8 < Lenght(x) < soccerf.lenght \times 1.2 \rightarrow SOCCERF(x) \end{bmatrix}$$

$$\forall x\begin{bmatrix} handballf.width \times 0.8 < Width(x) < handballf.width \times 1.2 \wedge \\ handballf.length \times 0.8 < Lenght(x) < handballf.lenght \times 1.2 \rightarrow HANDBALLF(x) \end{bmatrix}$$

$$\forall x\left[Width(x) < H_{\max} \wedge Lenght(x) < H_{\max} \rightarrow HOUSE(x)\right]$$

Here *soccerf.width*, *soccerf.length*, *handballf.width*, *handballf.length* and $H_{\max}$ are constant values of actual object instances. Each processor holds a copy of the model database and the rectangle candidates are scattered to the processors evenly. The parallel model matching algorithm performs an exhaustive search. The purpose of the first model match process is to recognize objects with their intrinsic properties (size, shape, color etc.) and to eliminate those erroneous candidates. With the above reasoning, three types of rectangles will initiate three types of objects from the model base. That is, each rectangle candidate receives a new definition as an object. After model matching, a merge operation is performed to combine components of the same object which were given different labels. This operation performs following steps:

Step 1. Perform all-to-all communication so that processors get all the objects.
Step 2. Perform model merging.
Step 3. Perform all-to-all communication so that processors hold new set of objects.

In the example given with Figure 3, assume rectangle R11 in PE1, circle C21 in PE2 instantiate soccer field and rectangle R32 in PE3 and circle C41 in PE4 instantiate a handball field. After step 1, each processor obtains a list of object instances generated by all the processors. Each processor then compares objects in local memory with the ones received from the neighboring processors. In Figure 3, R11 and C21 are instantiating the same soccer field, after analyzing parameters of SC11 and SC21 they are unified under new object SC12. Similarly, handball fields HND31 and HND41 unified as HND32. All-to-all communication at step 3 provides an updated list of the objects to all the processors. Each processor performs further processing only one portion of the objects from the entire list.

Another major step in the image understanding process is to analyze individual object instances for their completeness. After the first merging process, components of a composite object are grouped into the same object instance, if they are available initially. If there are missing object instances, a further feature extraction is performed on the image. These feature extraction tasks are model directed and performed within a certain area in the image where the features of the composite objects are expected. The output of the first model matching operation inputs to a function that constructs spatial relations of an object with its related components. This function, namely spatial relation constructor (SRC), completes the description of an object. Recognition of an object by collecting evidences about its parts may require iterative occurrences of the above operations. However, object recognition process will terminate when all the

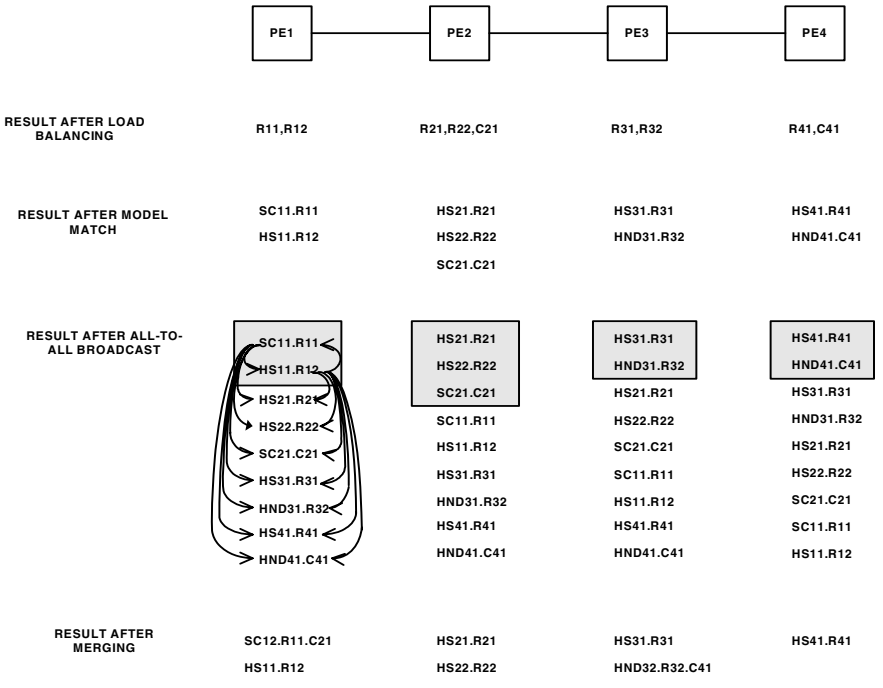


Fig. 3. An example to model matching and merging operations

hypothesis and constraints about the components of an object are satisfied. Once all the processors complete this step; they perform an all-to-all communication to update their local object list with the latest results.

The final step of the interpretation process in our system is to build a description of the scene. The knowledge used for this step is also provided with object instances and again based on the spatial relationships among the objects (performed by SRC program).

In the following example, final step of the parallel interpretation process will be elaborated. There are eight rectangle candidates available after processing the image in Figure 4. The interpretation process is executed on two processors where different reasoning operations are determined by the rule slots of the objects and the top-down hypothesis generation/verification stages.

Step1: After model matching, rectangle candidates R11 and R12 instantiate soccer fields SC11, and SC12 at PE1. Similarly R21 instantiates handball field HND21 at PE2. Instantiation rules IN1, IN2, and IN3 are deterministic as mentioned earlier and they utilize width and length parameters of the rectangle candidates to generate soccer field or handball field instances (rather than utilizing AO and AKO links). All the processors obtain a copy of the final object list and perform merge operation locally.

Step2: SC11, SC12 and HND21 are passed to SRC running at PE1 and PE2. Spatial rules SP1, SP2 and SP3, which are stored in the rule part of the objects, generate hypothesis H11, H12 and H21. The hypothesized objects initiate a top-down search by SRC. Once object instances CF11, CF12 and CF21 are determined, satisfy the spatial rules SP1, SP2, and SP3 than objects SC11, SC12, and HND21 will instantiated (Figure 5-a,b,c). Here, rules SP1, SP2 and SP3 are the same and includes the same predicate at_the_center(). Although, each processor handles a group of objects individually, they pass their result to each other and update the local object list which is needed for SRC program.

Step 3: At this step, rules associated with object instances are applied to establish spatial relationships among objects (Figure 5-d, e). For instance object SC11, generates hypothesis H11-12 and H11-21. SRC searches for the existence of a soccer field, which is next to SC11, and a handball field, which is in front of SC11. These relations are verified by each processor for their respective object instances.

Step 4: Objects SC11, SC12 and HND21 instantiate a hypothetical object called play ground. The hypothesis H13, H14 and H23 intersect in the same location though every object instance generates one playground object (Figure 5-f, g). There is no function defined for the spatial rule part_of so that each object instance will initiate the same object "play ground". At the end of this cycle, new object instances will be produced. After the merge operation, a playground instance PG13 that includes SC11, SC21 and HND21 as its parts and PG13 will be generated and it will reside in the first processor (PE1) according to the merge algorithm. Playground object may include rules for houses, roads or car park instances, which may generate new hypothesis for them. Interpretation process will terminate, if there are no more rules to evaluate.

Fig. 4. Test image used in experiments

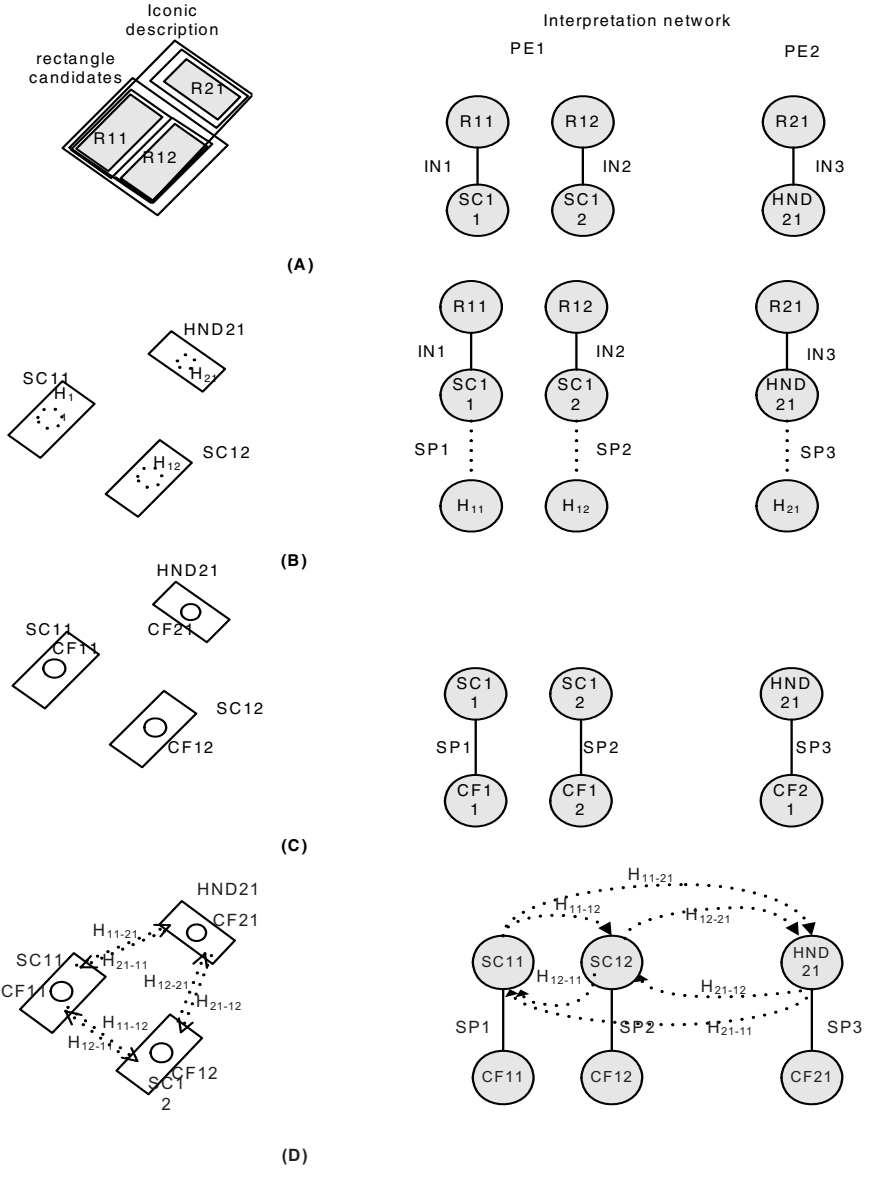
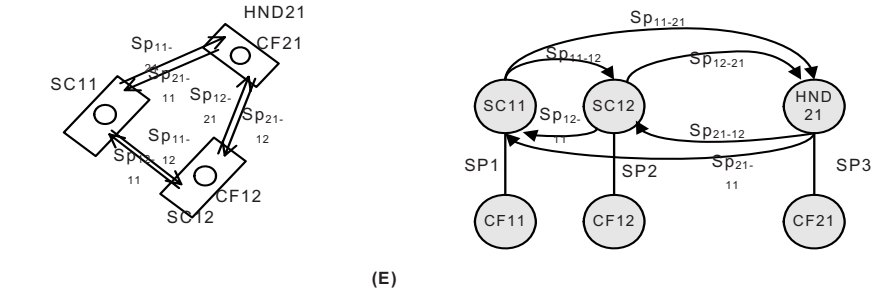
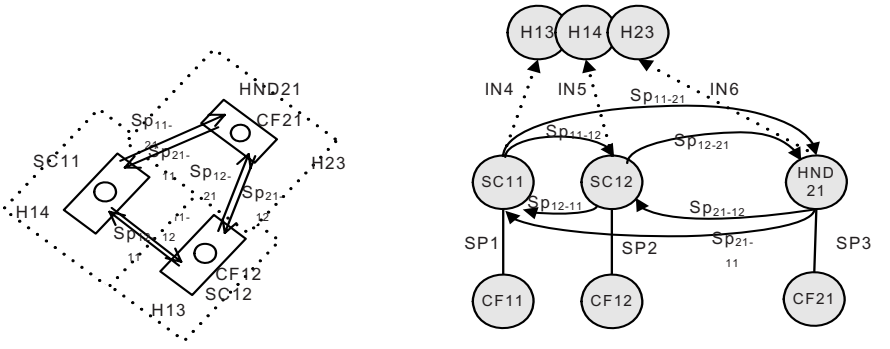


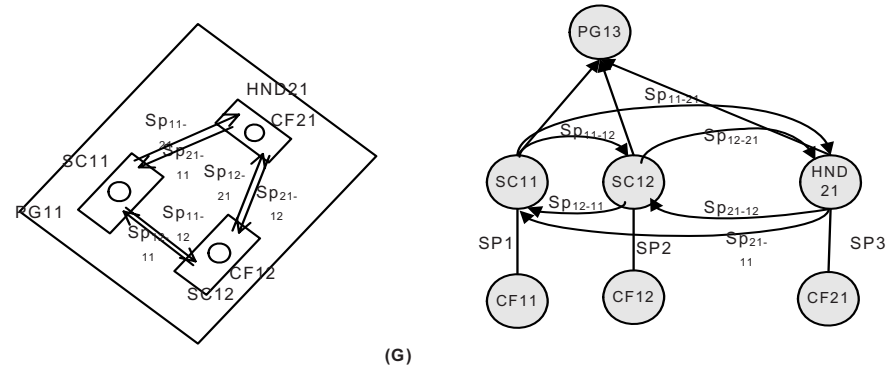
Fig. 5. An example to model matching and merging operations



(E)



(F)



(G)

Fig. 5. (*continued*)

3 Results

The image understanding problem is computationally demanding. A parallel algorithm needs to exploit all the possible data and functional parallelism. In the image understanding algorithm presented here, low and intermediate level operations utilize the available data parallelism. On the other hand, high level algorithm employs functional parallelism. Although, interpretation phase is rather simplified (for instance, link mechanism introduced in [11] is not used and no mechanism developed for

complex cases and for mutually conflicting objects etc) in our application, algorithm still requires many inter-processor communication which adversely affects the speed-up.

All the algorithms are developed using C language. Table 1 shows the performance of algorithms on various numbers of processors. A good speed-up is obtained for low and intermediate level operations. The speed-up for high level algorithms was rather poor. This is due to idle times of high level processor, as the number of objects to process at this level was small. DSPs used as processing elements in this system perform well with number crunching type operations. However, implementing symbolic operations was rather tedious and the performance obtained was not outstanding. In this example the number of objects created for the test image and the size of the model data base rather small which results in a small computation overhead as compared to communication cost and reduces the speed-up. Apparently, for a large number of objects, the computation cost will be higher and the parallel algorithm will produce a better performance. From the results, it can be concluded that by employing 8 DSP processors a video processing rate of 3 frames per second can be achieved.

Table 1. Timings (msec.) and speed ups for low, intermediate and high level algorithms for the image shown in Figure 4. (L = lines, C= Corners, Px= Pixels, R= Rectangles S= Objects).

Tasks	Processing time			Speed-up		
Number of processors	8	4	2	8	4	2
Low level: 256^2 Px input ->28C output	284.4	521.1	911.3	5.83	3.18	1.82
Intermediate Level: 28C input->534R candidates generated	6.7	11.6	18.3	4.37	2.52	1.6
High level: 10 R input -> 3 S output (three iterations for top-down search for missing features)	28.6	39.2	56.43	2.97	2.17	1.51

4 Summary

In this paper, implementation of an image understanding problem on a coarse grain multiprocessor computing platform is presented. The problem incorporates three image processing levels and the purpose of this application is to implement a parallel interpretation mechanism and evaluate the merits of the system. Due to the small size of the test problem a moderate speed up is achieved. Distributed interpretation is achieved by distributing object instances to the processors and treating them as independent reasoning agents. The parallel algorithm can exploit all the available processors, it can hypothesize low-level operations, and it is not computationally expensive. However, these algorithms are still at the experimental stage and further improvements are necessary.

References

1. Cantoni, V., Lombardi, L.: Hierarchical Architectures for Computer Vision. In: Proceedings of Euromicro Workshop on Parallel and Distributed Processors, pp. 392–398 (1995)
2. Chen, W., Meer, P., Georgescu, B., He, W., Goodell, L.A., Foran, D.J.: Image Mining for Investigative Pathology Using Optimized Feature Extraction and Data Fusion. Computer Methods and Programs in Biomedicine 79, 59–72 (2005)
3. Davis, L.S.: Foundations of Image Understanding. Springer, Heidelberg (2001)
4. Diamant, E.: Paving the Way for Image Understanding: A New Kind of Image Decomposition is Desired. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) SCIA 2005. LNCS, vol. 3540, pp. 17–24. Springer, Heidelberg (2005)
5. Ercan, M.F., Fung, Y.F.: The Design and Evaluation of a Multiprocessor System for Computer Vision. Microprocessors and Microsystems 24, 365–377 (2000)
6. Hecker, C.Y., Bolle, R.M.: On Geometric Hashing and the Generalized Hough Transform. IEEE Transactions on Systems, Man, and Cybernetics 24, 1328–1338 (1994)
7. Grimson, W.E.L.: Medical Applications of Image Understanding. In: IEEE Expert, pp. 18–28. IEEE Computer Society Press, Los Alamitos (1995)
8. Kumar, V.P., Wang, C.L.: Parallelism for Image Understanding. In: Zomaya, A.D. (ed.) Parallel and Distributed Computing Handbook, pp. 1042–1070. Mcgraw-Hill, New York (1996)
9. Lienard, B., Desurmont, X., Barrie, B., Delaigle, J.F.: Real-time High-Level Video Understanding Using Data Warehouse. Real-Time Image Processing 2006. In: Kehtarnavaz, N., Laplante, P.A. (eds.) Proceedings of the SPIE, vol. 6063, pp. 40–53 (2006)
10. Navulur, K.: Multispectral Image Analysis Using the Object-Oriented Paradigm. CRC press, Boca Raton, USA (2006)
11. Matsuyama, T., Hwang, S.: SIGMA A Knowledge-based Aerial Image Understanding System. Plenum Press, New York (1990)
12. Mehrotra, R., Nichani, S., Ranganathan, N.: Corner Detection. Pattern Recognition 23, 1223–1233 (1990)
13. Ogelia, M.R., Tadeusiewicz, R.: Picture Languages in Medical Pattern Knowledge Representation and Understanding. In: Torra, V., Narukawa, Y., Miyamoto, S. (eds.) MDAI 2005. LNCS (LNAI), vol. 3558, pp. 442–447. Springer, Heidelberg (2005)
14. Robertson, P.: An Architecture for Self-Adaptation and Its Application to Aerial Image Understanding. In: Proceedings of the first international workshop on Self-adaptive software, pp. 199–223 (2000)
15. Spyridonos, P., Papageorgiou, E.I., Groumpos, P.P., Nikiforidis, G.N.: Integration of Expert Knowledge and Image Analysis Techniques for Medical Diagnosis. In: Campilho, A., Kamel, M. (eds.) ICIAR 2006. LNCS, vol. 4142, pp. 110–121. Springer, Heidelberg (2006)
16. Weymouth, T.E., Amini, A.A.: Visual Perception Using a Blackboard Architecture. In: Kasturi, R., Trivedi, M., Marcel, D. (eds.) Image Analysis Applications, New York, pp. 235–281 (1990)
17. Wang, F.: A Knowledge-Based Vision System for Detecting Land Changes at Urban Fringes. IEEE Transactions on Geosciences and Remote Sensing 31, 136–145 (1993)