NEURAL NETWORK VISION INTEGRATION THROUGH COOPERATIVE LEARNING ON A MASSIVELY PARALLEL COMPUTER

Scott T. Toborg and Kai Hwang

CEng Technical Report 91-05

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA. 90089-0781
(213)740-4470

March 1, 1991

Neural Network Vision Integration Through Cooperative Learning on a Massively Parallel Computer *

Scott T. Toborg
Hughes Research Laboratories
3011 Malibu Canyon Rd.
Malibu, CA 90265
E-mail: toborg@maxwell.hrl.hac.com

Kai Hwang
University of Southern California
Department of EE-Systems
Los Angeles, CA 90089
E-mail: kaihwang@panda.usc.edu

Abstract

In this paper, we present a neural network approach for early vision integration on a massively parallel computer, the AMT Distributed Array Processor (DAP). We first define several energy functions for computing edge detection, optical flow and stereo disparity. Hopfield neural networks are used for function minimization with continuation on the sigmoid gain function to avoid local minima. New vision integration approaches are developed by extending the work of Poggio and Gamble to include cooperative processing, multiresolution, and unsupervised neural network learning. Vision integration algorithms are implemented on a DAP 610, consisting of 4096 PEs operated in SIMD mode. Experimental results are reported and analyzed in terms of effectiveness of resulting image segmentations and achieved speedup over a Sun4 workstation. Parallel processing of the neural learning process makes this approach attractive for intelligent computer vision applications.

Keywords: *computer vision, vision module integration, Markov Random Fields, Hopfield neural networks, Hebbian learning, cooperative computation and parallel processing.*

---

*Submitted 1 February 1991 to IEEE Transactions on Computers Special Issue: "Artificial Neural Networks" guest edited by Benjamin W. Wah. This research is supported in part by Hughes Research Laboratories and a Hughes Doctoral Fellowship and in part by the National Science Foundation Grant No. MIP-89-04172 to the University of Sourhtern California.

1 Introduction

Over the last 15 years, the development of computer vision systems has turned more and more to nature for clues on how to make artificial systems "see". As a result, important new parallel processing concepts have emerged from analogies with biological neural systems [29]. Machine vision systems are improving rapidly by exploiting the parallel, cooperative and adaptive capabilities of neural network techniques. One area of particular interest is the combination of data from multiple vision cues to form more robust image properties. To understand why this kind of neural network application is important, we first assess the current development of artificial vision systems.

Computer vision functions are usually divided into *low-level* (early) vision and *high-level* vision. Early vision consists of the visual functions that recover 3-D surface properties from sparse, noisy 2-D images. Typical properties include: *intensity/color edges*, *optical flow*, *surface orientation*, *texture regions*, and *depth*. Together these properties provide spatial and temporal data about the 3-D world, which are used for subsequent high-level *feature extraction*, *object recognition*, and *scene analysis*.

In general, information contained in an image is insufficient to reconstruct a unique set of 3-D surface properties. This is a direct result of the ill-posed nature of many early vision problems. Techniques have been developed for solving ill-posed problems by imposing additional *a priori* constraints on the image [4, 25]. However, determining which constraints to use is often based on ad hoc assumptions about the visual environment and the ultimate goals of the perceptual system. When the visual environment changes in a way that violates these constraints, the vision algorithm may fail. One way to make the computation of a specific image property more robust against possibly invalid a priori constraints is to combine results from other independent data sources [8].

The development of techniques for combining multiple data sources is a natural "next step" in perception research. Previous research in early vision focused on the analysis of noninteracting individual functional modules that are driven primarily

from incoming data. Assumptions were made to decouple individual vision cues from the interaction with other cues and from higher-level vision processes. However, to take advantage of the parallel and cooperative combination of multiple vision modules, these assumptions must be relaxed. The synergy of integrating evidence from multiple vision cues promises 3-D surface features that are accurately computed in spite of partially invalid a prior constraints, are more robust against noise, produce better results by reducing uncertainty, and speed algorithm convergence.

Recently, there has been significant interest in combining results from multiple low-level vision modules and multiple sensor types [20]. Marr and Barrow/Tennenbaum described visible surface representations for integrating low-level information [3, 21]. Many researchers have made noteworthy progress in combining outputs from two different vision cues [14, 16, 22, 27]. Aloimonos and Shulman outlined basic issues in developing a theory of vision integration [1]. Clarke and Yuille developed a useful taxonomy for delineating algorithms for vision integration and data fusion [8]. Poggio and Gamble developed general techniques for combining different vision modules using *Markov Random Fields* (MRFs) [11, 12, 24].

Weaknesses in many of the reported vision integration techniques can be traced to a lack of cooperative and adaptive processing. For example, previous attempts assumed a priori computation of one or more of the image properties [12, 22]. This assumption gives an unfair bias toward the precomputed property. In addition, vision module coupling weights are usually held constant. This restriction does not allow for adaptation due to changes in the input image.

This paper presents improved techniques for vision integration by using cooperative neural learning algorithms that allow simultaneous interactions between multiple early vision processes. Approaches are also described for automatically adapting the coupling weights between vision modules using neural network learning. Parallel iterative algorithms are defined and implemented on a massively parallel computer, the AMT DAP 610. Algorithms are enhanced using multiresolution techniques to speed relaxation. Experimental results being reported provide important insights into the methodology of early vision integration using neural networks.

The remainder of the paper is organized as follows: In Sec. 2, we formulate the problem of early vision processing using weak continuity constraints and neural networks. Section 3 describes new approaches for vision integration by extending previous algorithms. New techniques incorporate cooperative processing, multiresolution, and unsupervised learning. Section 4 describes how these vision integration algorithms are mapped to the DAP. We also report on results of vision experiments. Finally, we summarize contributions and comment on future research directions.

2 Early Vision Using Artificial Neural Networks

This section briefly describes techniques used for early vision processing based on weak continuity constraints and MRFs. Resulting energy equations are then minimized using a Hopfield neural network with continuation on the sigmoid gain function to avoid local minima.

2.1 Energy Functions for Early Vision Processing

The use of minimizing energy functions for computing early vision surface properties is wide spread and well established [30]. The output of these algorithms are retinotopic maps of smoothed surface properties along with corresponding discontinuities. Image surface discontinuities are extremely important since they often represent the most salient aspects of an image. They are also necessary to provide image chunking of the visual input for interfacing with higher-level processing. Consequently, in this paper, we focus on approaches that allow explicit detection and tracking of image property discontinuities.

Deterministic formulations for explicitly marking discontinuities were developed by Blake and Zisserman using *weak continuity constraints* [5]. The process is analogous to fitting an elastic sheet over a sparse set of wooden stakes. 2-D stake positions mark points where the data is known, while the stake height measures the value of the given data (e.g. corresponding to intensity, depth, etc.). The best estimate of the original image surface property is the one that minimizes the elastic potential

energy of the sheet. This energy is defined as a balance between consistency with the given data and surface property smoothness. The amount of smoothness imposed results in different elastic models (e.g. weak membrane or thin plate). At surface discontinuities, the elastic sheet stretches to the point of tearing and a marker called the *line process* is set.

A similar but more general stochastic formulation of early vision problems can be made using MRFs [4, 25]. In this approach, images are interpreted as 2-D random fields whose probability distribution for any given site is only a function of the values in a small neighborhood (the Markov property). Geman and Geman extended this approach by defining two coupled MRFs consisting of a continuous and a binary field [13]. The binary or "line process" field is used to explicitly mark the existence of a discontinuity. Under certain restrictions (additive Gaussian noise and a smooth surface field), the energy function is identical to the function derived using weak continuity constraints.

Weak continuity constraints and MRFs can be applied to a wide variety of early vision problems [5, 19, 30]. For example, edge detection can be viewed similar to the problem of fitting a weak elastic membrane to a surface. Tears in the membrane correspond to intensity discontinuities. One possible energy function is defined below for edge detection on a uniform rectangular grid:

$$\begin{aligned} U_e &= \sum_{i,j} \lambda_e[(f_{i,j+1} - f_{i,j})^2(1 - e^v_{ij}) + (f_{i+1,j} - f_{i,j})^2(1 - e^h_{ij})] \\ &\quad + \beta_e(f_{i,j} - I^{(r2)}_{i,j})^2 + \alpha_e(e^v_{ij} + e^h_{ij}) + \gamma_e(e^v_{ij}, e^h_{ij}) \\ &\quad + \int_0^{e^v_{ij}} g^{-1}(e^v_{ij})de^v_{ij} + \int_0^{e^h_{ij}} g^{-1}(e^h_{ij})de^h_{ij} \end{aligned} \tag{1}$$

with f representing the estimated smooth intensity and e^v_{ij}, e^h_{ij} respectively marking vertical and horizontal intensity edges. As long as the intensity difference between neighboring pixels is not too great, the first summation term in Eq. 1 is small and the line process variable (e_{ij}) is set to zero. However, if the intensity difference grows greater than a threshold α_e, the line process variable is activated to represent an intensity discontinuity. The second term weighs the relative importance of the input

data, $I_{i,j}^{(r2)}$. Where $I^{(ya)}$ denotes the different spatial and temporal images (y = right or left, a = 1 or 2 in time). The third term represents the cost of introducing a line process discontinuity. The fourth term is a function that embeds constraints on line process interactions. The final two integral terms are called the *gain* terms and are used in the minimization process to force line process values to either 0 or 1. Parameters λ_e, β_e, and α_e are used for weighting the relative importance of each term.

Very similar equations are defined for computing optical flow. This algorithm is based on work by Horn and Schunck [17] and augmented with explicit discontinuity detection similar to Hutchinson et al. [18]. Let u and s represent components of the velocity field in the x and y directions respectively, and o marks discontinuities in the flow field. The corresponding energy function is:

$$\begin{aligned} U_o &= \sum_{i,j} \lambda_o[(u_{i,j+1} - u_{i,j})^2(1 - o_{ij}^v) + (u_{i+1,j} - u_{i,j})^2(1 - o_{ij}^h)] \\ &\quad + \lambda_o[(s_{i,j+1} - s_{i,j})^2(1 - o_{ij}^v) + (s_{i+1,j} - s_{i,j})^2(1 - o_{ij}^h)] \\ &\quad + \beta_o[(I_{i+1,j}^{(r2)} - I_{i,j}^{(r2)})u_{ij} + (I_{i,j+1}^{(r2)} - I_{i,j}^{(r2)})s_{ij} + (I_{i,j}^{(r2)} - I_{i,j}^{(r1)})]^2 \\ &\quad + \alpha_o(o_{ij}^v + o_{ij}^h) + \gamma_o(o_{ij}^v, o_{ij}^h) \\ &\quad + \int_0^{o_{ij}^v} g^{-1}(o_{ij}^v)do_{ij}^v + \int_0^{o_{ij}^h} g^{-1}(o_{ij}^h)do_{ij}^h \end{aligned} \tag{2}$$

Likewise, stereo disparity is computed using analogous energy terms. The following formulation was adapted from Barnard [2] where d represents the disparity and w marks corresponding discontinuities:

$$\begin{aligned} U_d &= \sum_{i,j} \lambda_d[(d_{i,j+1} - d_{i,j})^2(1 - w_{ij}^v) + (d_{i+1,j} - d_{i,j})^2(1 - w_{ij}^h)] \\ &\quad + \beta_d(I_{i,j}^{(l2)} - I_{i+d(ij),j}^{(r2)})^2 + \alpha_d(w_{ij}^v + w_{ij}^h) + \gamma_d(w_{ij}^v, w_{ij}^h) \\ &\quad \int_0^{w_{ij}^v} g^{-1}(w_{ij}^v)dw_{ij}^v + \int_0^{w_{ij}^h} g^{-1}(w_{ij}^h)dw_{ij}^h \end{aligned} \tag{3}$$

Our vision integration approach is formulated independent of any particular formulation for the individual modules. The equations described above were used in

our prototypic system but may be easily exchanged or combined with other models. The only restriction is that the formulation should be expressed as a minimizing energy function with discontinuities.

2.2 Neural Networks for Nonconvex Minimization

Energy functions which explicitly track discontinuities are inherently non-quadratic and therefore require iterative methods from nonconvex function optimization. In these cases, global optima may be very difficult to find in a reasonable amount of time. In this research, we use a Hopfield neural network [15] to find image property discontinuities. Hopfield neural networks have been applied to minimizing energy functionals derived from regularization and MRF formulations to early vision [15, 19].

The general technique is to describe the vision problem as an unconstrained minimization. To do this the original formulation is changed so that binary line process variables vary continuously between 0 and 1 and an unconstrained internal state variable, m_{ij}, is defined so that $l_{ij} = g(m_{ij})$ where $g()$ is a sigmoid function. The gain term is used to gradually force the line process variable into an "on" or "off" state. Gradient descent equations are formed similar to Hopfield in that state changes are equal to the negative gradient of the energy. This results in the following dynamical system of equations:

$$\begin{array}{lll}
\frac{df_{ij}}{dt} = -\frac{\partial U}{\partial f_{ij}} & \frac{ds_{ij}}{dt} = -\frac{\partial U}{\partial s_{ij}} & \frac{dd_{ij}}{dt} = -\frac{\partial U}{\partial d_{ij}} \\
\frac{dm^h_{ij}}{dt} = -\frac{\partial U}{\partial e^h_{ij}} & \frac{du_{ij}}{dt} = -\frac{\partial U}{\partial u_{ij}} & \frac{da^h_{ij}}{dt} = -\frac{\partial U}{\partial w^h_{ij}} \\
\frac{dm^v_{ij}}{dt} = -\frac{\partial U}{\partial e^v_{ij}} & \frac{dq^h_{ij}}{dt} = -\frac{\partial U}{\partial o^h_{ij}} & \frac{da^v_{ij}}{dt} = -\frac{\partial U}{\partial w^v_{ij}} \\
 & \frac{dq^v_{ij}}{dt} = -\frac{\partial E}{\partial o^v_{ij}} &
\end{array}$$

With m, q, and a representing internal state variables related to the line process variables e, o, and w according to $g(m) = e$, $g(q) = o$, $g(a) = w$ where $g(x)$ is a

sigmoid monotonically increasing function with gain parameter μ:

$$g(x) = \frac{1}{1 + e^{-\mu x}} \tag{4}$$

To find the mean field or Hopfield solution: take the partial derivatives of the total energy with respect to f, s, u, d, e, o, w; set them to zero, and solve for the unknowns. After computing the derivatives and solving for unknowns, the finite difference approximation results in 10 iterative update equations. For example, using Eq. 1, the corresponding discrete iterative update equations for a very simple edge detection network (no line process interactions) is:

$$f_{ij} = \frac{\lambda_e[f_{i,j+1}E^v_{ij} + f_{i+1,j}E^h_{ij} + f_{i,j-1}E^v_{i,j-1} + f_{i-1,j}E^h_{i-1,j}] + \beta_e I^{(r2)}_{ij}}{\lambda_e[E^v_{i,j} + E^h_{i,j} + E^v_{i,j-1} + E^h_{i-1,j}] + \beta_e} \tag{5}$$

$$m^v_{ij} = \lambda_e(f_{i,j+1} - f_{ij})^2 - \alpha_e \tag{6}$$

$$m^h_{ij} = \lambda_e(f_{i+1,j} - f_{ij})^2 - \alpha_e \tag{7}$$

where $E^x_{i,j} = 1 - e^x_{i,j}$ and $x = h$ or v. Similar iterative equations are derived for computing optic flow and stereo disparity. Equations 5– 7 are iterated until equilibrium, at which point the gain in the sigmoid function (μ) is increased. This procedure is repeated until all line processes are forced to values of 0 or 1.

The above procedure is similar to Blake and Zisserman's *graduated non-convexity* approach, where the function is gradually changed from a convex approximation to the actual function solution [5]. While there is no guarantee that a global minimum will be reached, the process works well in experiments on real images, as long as changes in the gain are small.

3 Cooperative Learning for Vision Integration

We define *vision integration* as the use of information from one vision module to guide the calculation of image properties in other modules. This is to be contrasted with the concept of *data fusion* which refers to the combination of information from different data sources into a single representation. Clark and Yuille have defined a useful taxonomy for describing many different types of data fusion/vision integration techniques [8]. In general, approaches can be broken down into two main classes: *weak* and *strong* coupling. Weak coupling is characterized by the combination of data from multiple independent sources in a way that does not affect the operation of the contributing modules.

The second major fusion category is strong coupling. These types of approaches allow information from one data source to influence or change the other data source's model or constraints. A special case of strong coupling is defined when there exists a feedback loop such that one module can modify the prior constraints of the others and vis a versa. This type of interaction is called *recurrent strong* coupling. The algorithms defined in the previous section use recurrent strong coupling since line processes depend on estimates of the continuous valued field and vis a versa.

In this paper, we evaluate different types of strongly coupled vision integration techniques. Multiple early vision modules are coupled together via line processes in each module. The original energy functions for edge detection, stereo disparity and optical flow are modified to include coupling terms which define how discontinuities in one module can influence discontinuities in the other. With this integrated system of energy functions, we examine different ways to update the equations.

Poggio and Gamble developed an updating policy, where intensity edges are precomputed and used to guide the formation of discontinuities in other modules [12]. In Sec. 3.2, we propose an alternative updating scheme called *cooperative coupling* where there are no precomputed edges and all modules mutually modify the computation of discontinuities based on values in other modules. The impact one vision module has on another is controlled via vision module coupling weights. The cooper-

ative coupling approach is enhanced in Sec. 3.3 by allowing coupling weights to adapt using a modified Hebbian learning procedure.

3.1 Poggio and Gamble: Feedforward Strong Coupling

Poggio, Gamble and other members of the MIT Vision Machine project have developed a number of ideas for integrating multiple vision cues [11, 12, 24]. The key ideas in this approach are: 1) Discontinuities in different surface properties (i.e. depth, optical flow, orientation, texture) usually occur where there are sharp changes in the image intensity, 2) Vision cues can be coupled together via their discontinuities, 3) Different vision cues are coupled to precomputed intensity edges to simplify computation and provide a mechanism for aligning (registering) discontinuities from multiple data sources.

We briefly describe how this approach could be applied to Eqs. 1–3 for edge detection, stereo disparity and optical flow. First, vision module interaction terms can be defined to reflect the idea that formation of a discontinuity in one process should constructively influence the formation of a discontinuity in another process. The amount of influence is determined by the weighting factor C_i. For the edge/stereo/motion network the following coupling terms are added to Eqs. 1–3:

$$
\begin{aligned}
&+ \; C_1[(1-e_{ij}^h)o_{ij}^h + (1-e_{ij}^v)o_{ij}^v] \\
&+ \; C_2[(1-d_{ij}^h)o_{ij}^h + (1-d_{ij}^v)o_{ij}^v] \\
&+ \; C_3[(1-e_{ij}^h)d_{ij}^h + (1-e_{ij}^v)d_{ij}^v] \\
&+ \; C_4[(1-o_{ij}^h)d_{ij}^h + (1-o_{ij}^v)d_{ij}^v] \\
&+ \; C_5[(1-o_{ij}^h)e_{ij}^h + (1-o_{ij}^v)e_{ij}^v] \\
&+ \; C_6[(1-d_{ij}^h)e_{ij}^h + (1-d_{ij}^v)e_{ij}^v] \qquad (8)
\end{aligned}
$$

These new terms impact the line process update equations for edge detection, stereo disparity and optical flow by introducing additional factors to the line process update equations.

This approach uses *feedforward strong coupling* [8] in that intensity edges are first computed and then used to help discontinuity formation in other vision cues. With coupling terms defined as in Eq. 8, the energy is minimized when optical flow and stereo disparity discontinuities are located at intensity edges. Therefore, the intensity edges act like a fixed external field whose impact is modulated by the size of the coupling weight.

Several distinct advantages of this approach are worth noting. By tying all vision surface properties to intensity edges, discontinuities in other modules can be better localized. Intensity edges tend to be smoother and more continuous than other early vision data. Of course registration of multiple image surfaces is greatly simplified by having a common anchor. This also aids in subsequent discontinuity labeling. In addition, there are fewer stability problems to worry about since there is no feedback loop.

One major drawback in this scheme is the lack of cooperative interaction among the vision modules. For example, feedback from other sensors and data sources could be useful in correcting edge detection errors and filling in missing information from weak edges. In addition, coupling weights should be adaptive to reflect changes in the environment that might effect the impact one module has on another. This could also speed algorithm convergence by rapidly reenforcing line processes that highly correlate.

3.2 Cooperative Processing of Multiple Vision Modules

In this approach, we allow mutual interactions between all vision modules. For example, *intensity*, *stereo* and *motion* computations simultaneously impact the formation of each others discontinuities. No changes are needed to previously defined equations. The only difference is how the equations are updated. Instead of pre-computing the intensity edges, update equations for the different vision modules are iterated together so that each module can influence the formation of discontinuities in the others.

Figure 1 shows the overall data flow for initial experiments using the cooper-

ative coupling approach. A set of four intensity images are generated for computing stereo, motion and intensity edges. Lines between vision processes represent possible interactions with corresponding weights representing the impact of one module on another. The end result is a set of intrinsic images each consisting of a continuous and binary line process that respectively denotes smoothed image surface properties and their discontinuities.

While the cooperative coupling approach has more flexibility in the types of intermodule interactions allowed, new issues arise in regard to convergence and stability of the system. Convergence in this context has to do with whether the system (or its parts) iterates to a fixed point (which represents a local minima in the energy function), diverges, gets caught in a cycle, or goes chaotic. At this point we have empirically identified ranges of parameter values that lead to system convergence.

Another more immediate problem is the need to control the convergence rates of different component modules so that all relax at roughly the same rate. Why is this important? From experiments coupling edge detection and optical flow we found that the edge detection module consistently converged much faster than the optical flow module. Consequently, intensity edges unfairly biased the computation of optical flow discontinuities which resulted in essentially the same behavior as in the Poggio/Gamble approach. This is undesirable in the cases involving "the leopard in the jungle" problem where it is impossible to segment the object correctly without motion information. In these cases, we may want the strong optical flow information to influence the formation of intensity edges to better segment the moving object.

To encourage unbiased interactions between early vision modules, the rates of convergence need to be the same. In Eqs. 1–3 the primary variable for controlling convergence is the gain parameter μ of the sigmoid function defined in Eq. 4. When this parameter reaches a predefined maximum, all line processes have been forced to on or off states. If the gain in one module rises faster than another, the module with the highest gain tends to have a greater bias over other modules. Simply making the gain equivalent in all modules does not solve the problem.

The value of the line process is also determined by other parameters and these

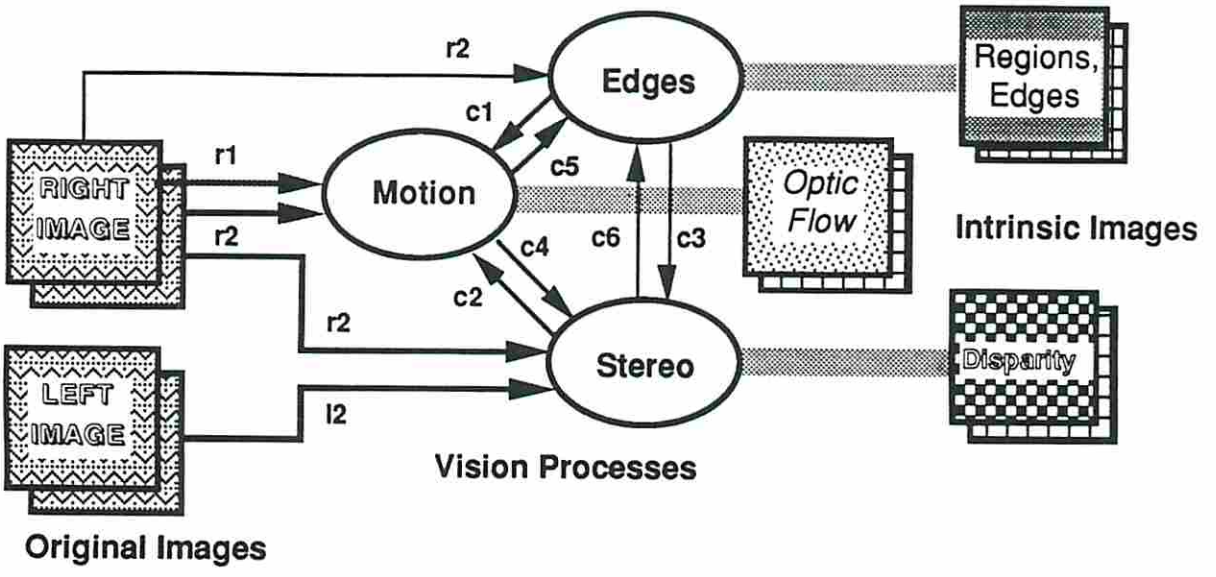


Figure 1: Overview of proposed neural vision integration system. Motion-stereo images are simultaneously and cooperatively processed by motion, stereo and edge detection modules. Coupling weights between modules represent influence one module has on another. Coupling weights are adapted using Hebbian learning which acts to reenforce discontinuities that correlate and kill off those that don't. The result of the vision integration process is a set of intrinsic images which contain smoothed image surface properties and their corresponding discontinuities.

parameters may be on completely different scales for different vision modules (e.g. the line process penalty is much different between intensity edges and optical flow). To alleviate this problem, the gains of the different modules are updated at the same time, set equal to each other, and scaled by the line process penalties of the other modules. Experiments show that this scaling procedure does effectively equalize the convergence rates of the different modules.

3.3 Learning Vision Module Coupling Weights

Performance of the cooperative coupling algorithms described in Sec 3.2 can be further improved by replacing fixed vision module coupling weights with weights that adapt or learn. What does it mean to learn the coupling weights, and how should this learning occur? First of all, without knowledge from higher level processes we don't know what the correct coupling weights should be. Consequently, we must use learning methods that do not require an outside teacher. In our experiments, a modified Hebbian learning procedure was used that rewards discontinuities that correlate and punishes mismatches. The continuous form of the weight update equation for pairwise interactions between different vision modules can be written as:

$$\frac{dC_{ij}^{xy}}{dt} = -C_{ij} + \gamma(x_{ij} * y_{ij} + (1 - x_{ij}) * (1 - y_{ij})) \tag{9}$$

where C_{ij}^{xy} is now a spatially varying intermodule coupling weight, γ is the learning rate and x_{ij}, y_{ij} represent different combinations of vision module line processes. From this equation we see that the weight at a given pixel is increased if image discontinuities match, and is decreased (via a decay term), if there is a mismatch.

When used with cooperative coupling, this learning procedure helps highly correlated discontinuities to rapidly converge while simultaneously killing off weak uncorrelated line processes. On the other hand, if there is strong information from an individual module, this knowledge will persist in spite of the coupling weight decay. The primary advantage of the Hebbian approach is that it speeds convergence for line processes that match. This helps to enhance the overall convergence of the system.

3.4 Multiresolution Techniques for Vision Integration

A very attractive feature of the formulation of early vision problems using variational principles and partial differential equations is that the resulting iterative relaxation equations are highly parallel and require only local interactions. This is especially ideal for massively parallel SIMD computers. Unfortunately, the convergence of the iterative process is often excruciatingly slow on fine images. This slow convergence is a result of the local nature of the smoothing which acts to rapidly smooth high-frequency error but only asymptotically reduces low-frequency error [6].

The key idea behind multilevel relaxation is to quickly reduce high-frequency error on a fine grid size and then approximate the problem on a coarser grid. The remaining error is smoothed much faster on the coarser grid. The solution on the coarse grid is then transfered back to the fine grid. The coarser grid essentially trades spatial resolution for direct communication over longer distances. Studies on model problems (e.g. Poisson's equation) show that multigrid methods can converge in $O(N)$ iterations, where N is the number of nodes in the grid. On the other hand, the same problem takes $O(N^3)$ iterations using a single grid [26].

We use a Gaussian pyramid to define our hierarchy of image grids [7]. The Gaussian pyramid is created by repeatedly convolving the image with a 5x5 Gaussian mask and performing a 2-to-1 subsampling of the image to create the new reduced resolution image. Due to DAP memory limitations, we create only a three-level pyramid with image edge ratios 128 : 64 : 32. For simplicity and ease of analysis, we use a full approximation W-cycle, fixed pattern updating scheme with injection for fine-to-coarse restriction and bilinear interpolation for coarse-to-fine prolongation. The multigrid algorithm is used primarily to speed convergence of the "continous processes" of the vision integration system because these variables typically relax much slower than the line processes.

4 Vision Integration Experiments and Results

In this section, we describe parallel algorithms for vision integration developed for the DAP system. Experiments are also described which show the impact integration has on improving image segmentation and speeding algorithm convergence. The performance of the DAP 610 is compared with that of a Sun4/260 for the same purpose.

4.1 Parallel Algorithms for the DAP

All of the algorithms discussed in this paper were implemented on the DAP 610 [23]. The DAP 610 is a massively parallel fine grain array of bit serial *Processing Elements* (PEs). Our current configuration consists of a 64 x 64 mesh connected array (4096 PEs) with 8Kbytes memory per PE for a system total of 32MB. A Sun4/110 workstation is used as a host for interfacing with the DAP. Host programs are written in C, while DAP algorithms are written in Fortran-Plus, which is similar to Fortran-77 with special parallel extensions. A 40 Mbyte/s fast channel interface allows high resolution video visualization of the data as it is being processed with very little performance degradation.

The discrete iterative update equations described in Sec. 2 are ideally suited for implementation on SIMD mesh connected hardware architectures. Individual early vision modules can be processed with massively parallel, synchronous computations that require only local interconnections. The mapping to parallel hardware is logically viewed as simply one pixel per processor. On the DAP, array size is transparent to the software developer so that algorithms can be designed independent of the image size (within memory constraints).

As a result of the opportune matching of algorithms and architecture, individual early vision vision algorithms make very efficient use of the parallel hardware. The following DAP pseudocode outlines the cooperative processing of multiple vision modules:

```
CALL Initialize-modules
100 IF (gain in all modules < maximum gain) GOTO 110
        CALL update-edges(f, e)
        CALL update-disparity(d, w)
        CALL update-flow(s, u, o)
        IF (norm from a given module < norm-threshold) THEN
            CALL change-sigmoid-gain
        ENDIF
        IF (gain has changed in any module) THEN
            CALL update-line-process (for that module)
            CALL udpate-coupling-weights (for all modules)
        ENDIF
        GOTO 100
110     CONTINUE
```

Each vision module is processed in parallel, while successive modules and their interactions are sequentially updated. An ideal architecture for this problem would have multiple SIMD arrays which could then simultaneously update all vision modules. In fact, new architecture designs have already been proposed for multiple SIMD arrays using 3-D Wafer Scale Integration (WSI) which could further exploit the parallelism in this problem [28].

Calls to update each vision module perform much more than a simple iterative relaxation, they also call subroutines to execute multigrid relaxations as described in Sec. 3.4. Let L be the finest grid (typically either 128^2 or 256^2 depending on the DAP memory capacity). $I_{j \rightarrow j-1}$ is a simple injection operator for moving results from fine to coarse grids, while $I_{j-1 \rightarrow j}$ interpolates values to move from coarse to fine. Specified below is a multigrid algorithm used in the vision computations:

C Perform V-cycle for X times

D0 100 $i = 1, X$

C Fine-to-Coarse stage

DO 10 $j = L, 1, -1$

call **update-module**(f^j)

$f^j = I_{j \rightarrow j-1}(f^j)$

10 CONTINUE

C Coarse-to-fine stage

DO 20 $j = 1, L$

call **update-module**(f^j)

$f^j = I_{j-1 \rightarrow j}(f^j)$

20 CONTINUE

100 CONTINUE

Subroutine **update-module** performs a parallel update of a particular continuous valued surface property f at grid level j. For example, Eq. 5 updates estimates of smoothed intensity values and is translated into the following DAP code:

subroutine update-edges

newf = (1.0 - vp) * SHWP(f,1)

newf = newf + (1.0 - SHEP(vp,1) * SHEP(f,1) .

newf = newf + (1.0 - hp) * SHNP(f,1)

newf = newf + (1.0 - SHSP(hp,1) * SHSP(f,1)

newf = newf + λ_e * newf + β_e * IR2

newf = newf / (λ_e * (4.0 - (vp + SHEP(vp,1) + hp +

SHSP(hp,1))) + β_e)

max-norm = MAXV(newf - f)

f = newf

RETURN

END

Where SHWP, SHEP, SHNP, SHSP are data shifting operations with no wrap around.

MAXV finds the maximum value of the matrix argument. Operations are performed over the entire grid in parallel. Notice that computations are broken up into very simple expressions to reduce the amount of temporary memory used for partial results.

Estimating the computational complexity of iterative algorithms is very difficult since the total number of operations is highly dependent on the input images and algorithm parameters. The best that can be done is to estimate operations per iteration combined with an average iteration count based on empirical experiments. Estimates for storage requirements are much more straight forward.

In general, we note that the number of operations and memory requirements grows linearly with the number of vision modules M and N^2 has the image sizes increase. The number of coupling weights (fully connected case) increases by $L(L-1)$ where L is the number of modules. This rapid increase in number of coupling weights should not be a problem because the number of coupled modules will always be relatively small (3-10).

4.2 Vision Integration Experiments

Experiments were designed to compare the performance of the various vision integration approaches described in Sec. 3. The main objectives were to measure the impact of integration on: 1) the quality of image segmentation, 2) robustness against noise, 3) speed of computations. Three different cases were examined: 1) No integration, modules compute results independently, 2) Poggio/Gamble approach with precomputed edges coupled to optical flow and stereo disparity computation, and 3) Cooperative computation of intensity edges, stereo disparity and optical flow with coupling weight learning.

These cases were run using two different sets of synthetic images: a moving square embedded with varying levels of Gaussian noise, and a moving square on a sinusoidal background. Figure 2 shows a selected frame from each of the stereo/motion sequences used in experiments. Synthetic motion and stereo sequences were generated using a simulated multiple camera system [9]. This system allowed for precise control over image geometry and the computation of ground truth values for all image prop-

erties. Performance on synthetic images was measured in terms mean squared error (for the smoothed image properties) and percentage error (for the line process discontinuities) based on known ground truth. We also measured number of iterations, and CPU time.

Table 1 shows the results of computations without integration. Mean squared error was used to measure values of the smoothed image surfaces against the hand computed ground truth. Percentage error was used to identify the number of mismatches in the line processes against the ground truth. Figure 3 shows results for squares embedded with 0 dB noise. Notice especially how jagged the motion discontinuities are due to uncertainty in the optical flow. Algorithm parameters were adjusted to give results with closed contours (if possible) and then held constant for all other experiments within that image type.

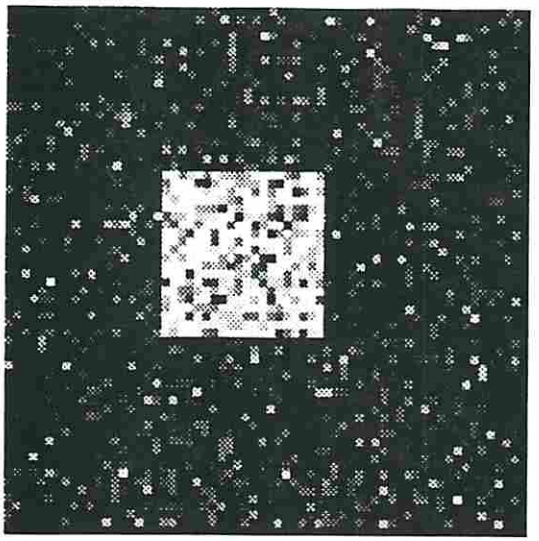
Table 1: **Results for edges, optical flow and disparity without integration**

Image noise	Iterations (e/m/s)	Time (s) (e/m/s)	MSE (e/m)	%Error (e/m/s)
10 dB	749/1078/210	3.42/8.66/8.91	120.42/2082.16	0.0/3.71/2.10
5 dB	798/1132/1297	3.64/9.68/16.61	251.92/2155.04	0.0/4.07/2.42
0 dB	889/1267/3117	4.05/10.93/39.92	679.37/2398.54	0.1/4.10/2.83

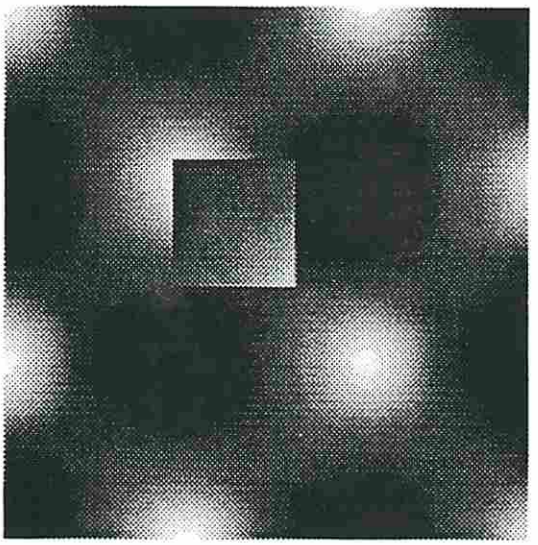
e = edges, m = motion, s = stereo

Table 2 summarizes results from vision integration with precomputed edges. Results are remarkably better for optical flow. In addition, the number of iterations and total time for computation is more than two and a half times faster than the no integration case. This type of speed-up and corresponding reduction the number of required iterations was characteristic of all of the vision integration algorithms examined. Figure 4 displays results for intensity edges and optical flow. Discontinuity images look the same because motion discontinuities were tied to intensity edges.

Finally, Table 3 contains the results of a few runs using cooperative computation with adaptive weights. In this case, the quality of the segmentation is about the same as the previous case but number of iterations were consistently fewer than either of the previous integration approaches. Figure 5 shows segmentation results for all three modules. These types of algorithms can help to fill-in weak or missing

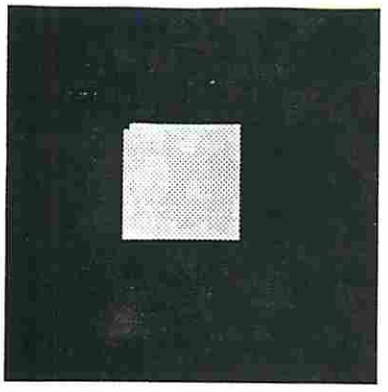


(a) Square in Gaussian noise

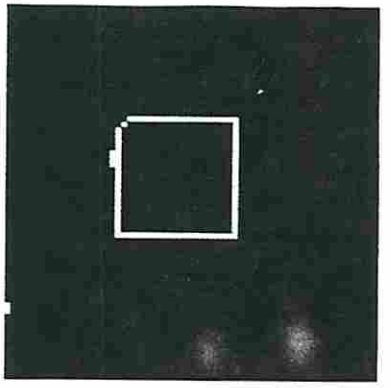


(b) Square with sinusoidal background

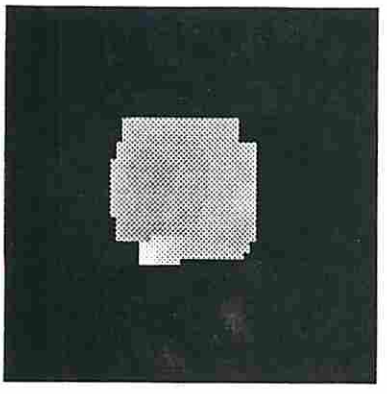
Figure 2: One frame each from synthetic motion-stereo sequences used in vision integration experiments



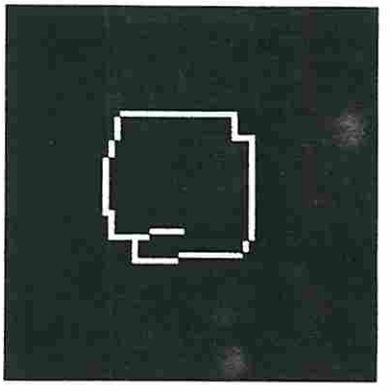
(a) Smoothed intensity



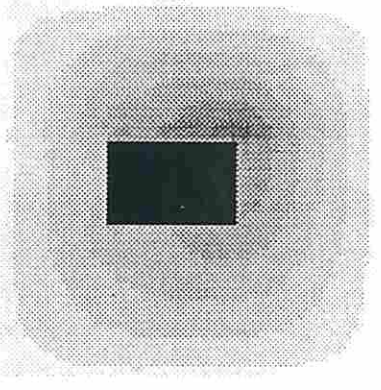
(b) Intensity edges



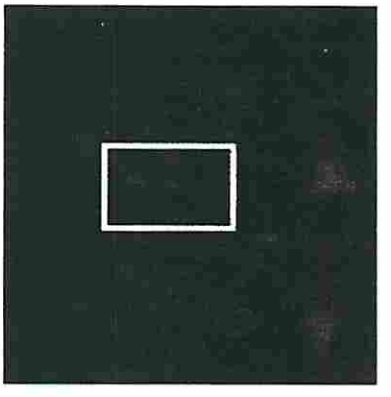
(c) Optical flow



(d) Optical flow discontinuities

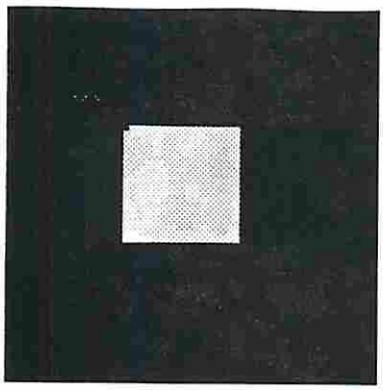


(e) Stereo disparity

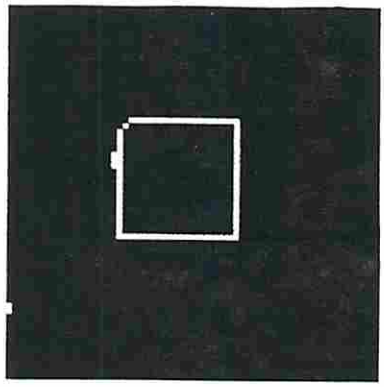


(f) Disparity discontinuities

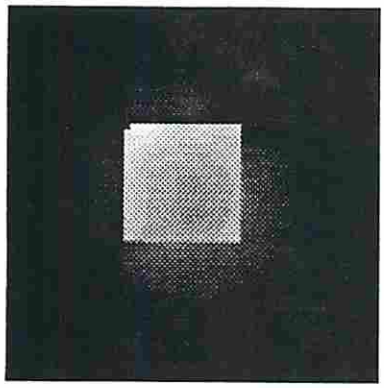
Figure 3: Segmentation without integration for moving squares with 0 dB noise. Poor results from motion and stereo processes illustrate the need for integration with intensity edges.



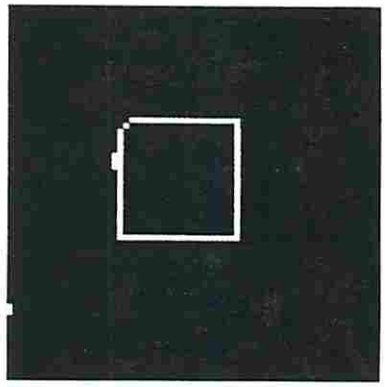
(a) Smoothed intensity



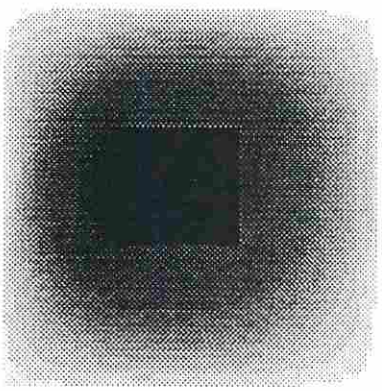
(b) Intensity edges



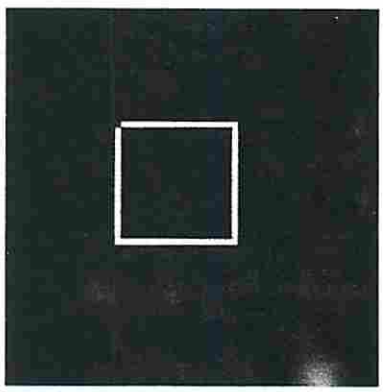
(c) Optical flow



(d) Optical flow discontinuities



(e) Stereo disparity



(f) Disparity discontinuities

Figure 4: Vision integration with precomputed edges for moving squares with 0 dB noise. Results are far superior than computation without integration.

Table 2: Results for vision integration with precomputed edges

Image noise	Iterations (e/m/s)	Time (s)	MSE (edges/motion)	%Error (e/m/s)
10 dB	749/340/89	18.09	120.42/2360.25	0.0/0.0/0.0
5 dB	798/326/90	18.36	251.92/2353.11	0.0/1.93/0.0
0 dB	889/655/95	22.99	679.37/2376.74	0.2/2.08/0.0

e = edges, m = motion, s = stereo

information in one module but may sometimes mislead or degrade perfectly good information in another. A major issue is how to set the values for the intial weights. When set very high (e.g. maximum of the line process penalties), the system behaves identically to the Poggio/Gamble case. In these particular runs the coupling weight was set low (15-100) so that the motion and stereo modules would be able to exert more influence. Other issues arise regarding the setting of parameters for the learning rate and time step. For these runs, the learning rate was set at 10.0 with a time step of 0.01.

Table 3: **Results for cooperaive vision integration with adaptive weights**

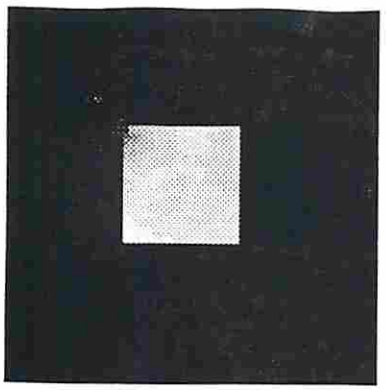
Image noise	Iterations	Time (s)	MSE (edges/motion)	%Error (e/m/s)
10 dB	483	28.47	100.40/2832.23	0.0/0.0/0.0
5 dB	589	31.21	205.38/2857.21	0.0/0.0/0.0
0 dB	739	35.26	702.75/3198.04	0.1/0.1/0.1

e = edges, m = motion, s = stereo

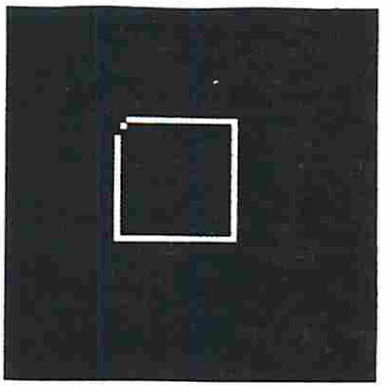
4.3 Analysis of Results and Implications

Vision integration provides a substantial improvement to optical flow and stereo disparity computations both in terms of segmentation quality and computational efficiency. Experiments using cooperative updating showed that integration also speeds convergence of the edge detection module. These observations can be verified by examining the tabular results from the previous section.

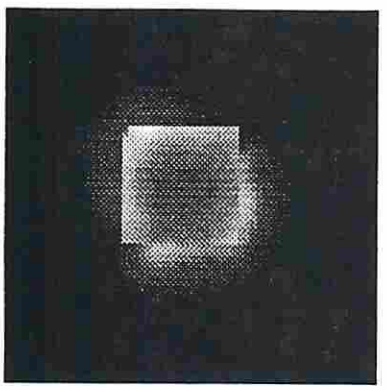
For instance, Tables 1 and 2 show that vision integration with precomputed edges helped to reduce the percentage error for discontinuities. In addition, the



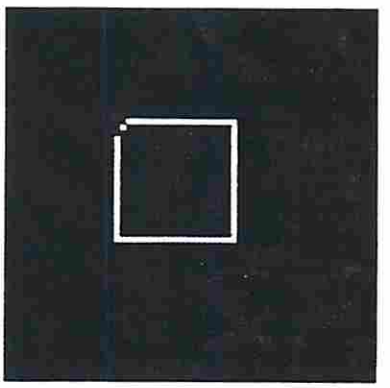
(a) Smoothed intensity



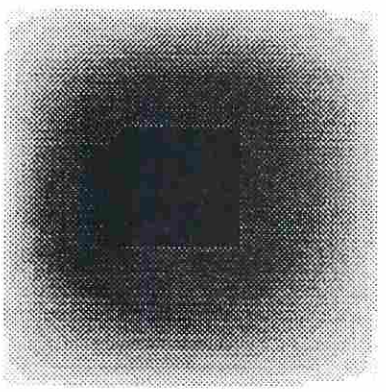
(b) Intensity edges



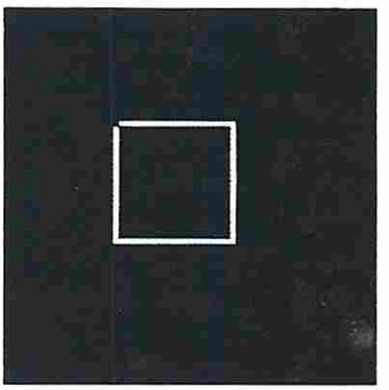
(c) Optical flow



(d) Optical flow discontinuities



(e) Stereo disparity



(f) Disparity discontinuities

Figure 5: Cooperative vision integration with adaptive coupling weights for moving squares with 0 dB noise. Results are similar to Poggio/Gamle approach but require fewer iterations.

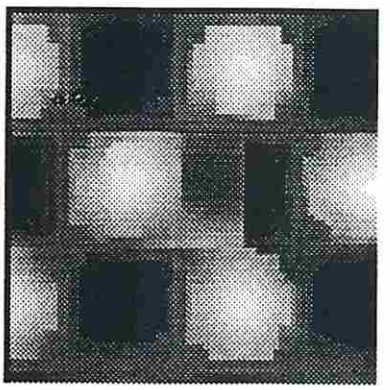
number of iterations (total from all modules) was significantly reduced from 5273 to 1639 for the 0 dB noise case. The amount of computation time dropped from a total of 54.9 sec. for the individual modules to 22.99 sec. with integration.

Similar performance can be seem for the cooperative learning approach by comparing Tables 1 and 3. The number of iterations was dramatically reduced from 5273 to 739 for the 0 dB noise case. Significant improvements were also made in reducing discontinuity error. The amount of time for computing dropped from 54.9 to 35.26 seconds. While this is a very good performance improvement, further reductions in computation time can be made by fully exploiting the parallelism in the algorithm problem.

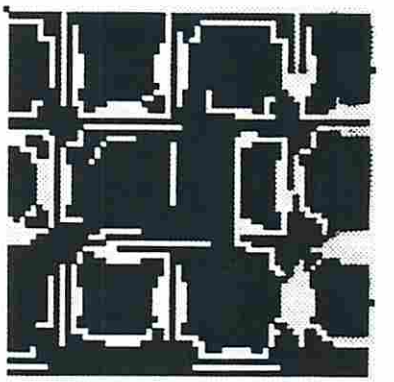
Comparing integration with precomputed edges versus the cooperative method shows that while the precomputed edges method requires slightly less compute time on the DAP, the cooperative method has more potential for parallelism on a multiple SIMD machine. While it may seem initially unclear what performance benefits come from cooperative computation, real perceptional environments are full of situations that clearly argue in favor of cooperative methods.

What good is cooperative computation in vision integration? To answer we have to look at cases where intensity edges do not provide sufficient information to guide segmentation in other modules. Many examples exist in real world situations where perception systems need more than intensity edges to segment an object. One obvious example is the "leopard in the jungle" where target objects are camouflaged by matching background and object textures. In this particular case, motion information is vital for correctly segmenting the target object.

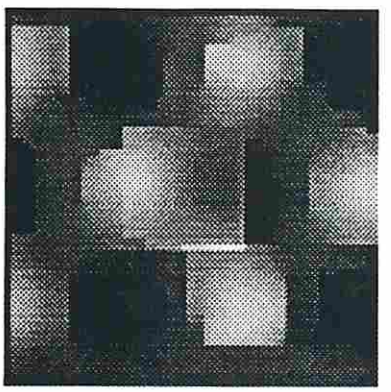
Cooperative coupling methods adapt to this situation by using strong motion and stereo data to influence the formation of intensity edges. For example, Fig. 6-(a) and 6-(b) show the region smoothing/edge detection results using the Poggio/Gamble approach. Because of intensity edge dominance, the square is not segmented completely. Using the same parameters for cooperative coupling with learning, motion and stereo information can now influence intensity edge formation to recover the square (Fig. 6-c and 6-d).



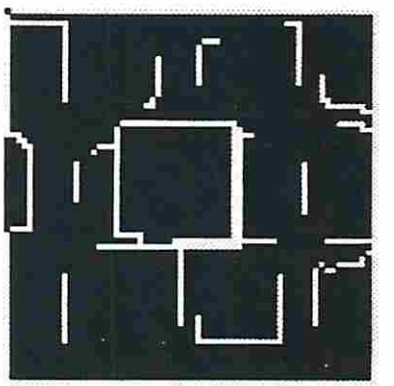
(a) Smoothed intensity



(b) Intensity edges



(c) Smoothed intensity



(d) Intensity edges

Figure 6: Images (a) and (b) show intensity regions and edges for square with sinusoidal background using Poggio/Gamble integration. Images (c) and (d) show intensity regions and edges from cooperative vision integration with adaptive weights. These expereiments illustrate how cooperative methods can be used to fill in and compensate for weak or missing data.

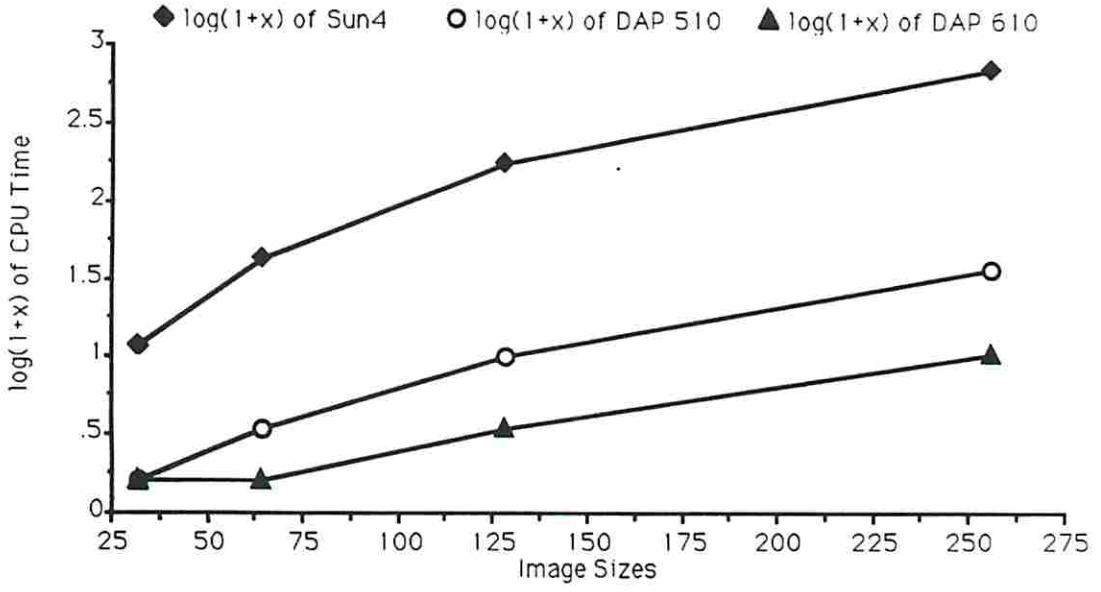


Figure 7: Benchmark performance for MRF edge detection algorithms on DAP 510, DAP 610 and Sun4/260 as a function of image size. Typical speedups on the DAP 610 were over 75 times the Sun4/260.

These experiments have highlighted the importance of vision integration as a means of embedding additional constraints on a problem using alternative data sources. We have shown how these additional constraints help produce results that are robust against noise and speed algorithm convergence. Cooperative computation and learning are especially important in rapidly changing visual scenes where breakdown of computation in one vision module can be compensated by information in others. We have also noted in our experiments that cooperative methods operate with very little parameter adjustment. They are robust over a much wider-range of parameters. This is compared to other methods, where small changes in parameters can produce widely varying results.

It is also interesting to compare the performance of these algorithms on serial vs. parallel machines. Many benchmarks were performed on the Sun4/260 and DAP 610. For example, Fig. 7 shows performance benchmarks for the edge detection algorithm described in the paper. From benchmarks on several early vision algorithms, we observed typical DAP 610 speedups of over 75 times the Sun4/260.

5 Conclusions

This paper has presented several new techniques for combining information from multiple vision modules. Parallel algorithms were specifically developed for *edge detection*, *stereo disparity* and *optical flow*. Similar formulations could be made for many other early vision problems [25]. Experiments using synthetic images clearly showed the advantage of integrating outputs from multiple data sources to improve segmentation and speed convergence. Cooperative learning methods are especially important in situations where intensity edges are unreliable. In these cases, cooperation from other vision modules can help to fill-in missing information. We have also shown that adapting vision module coupling weights using Hebbian learning helps to speed overall convergence by reenforcing discontinuities that correlate and punishing those that don't.

Our vision integration algorithms can be enhanced by looking at alternative

formulations for the different early vision modules. Initial models were chosen to simplify implementation and analysis. In addition to what has been demonstrated in this paper, computation of other image properties might be considered (e.g. *color* and *texture*). The major next step in computer vision development will be to couple low/intermediate level processing with high-level vision algorithms. One avenue we are pursuing is to use the discontinuities produced by the vision integration algorithms as input for a high-level feature extractor like the *selective attention neocognitron* [10]. In this case, feedback from the neocognitron could also be used to affect the formation of low-level features providing cooperative coupling between low- and high-level processing. Special purpose hardware is also needed to fully exploit algorithm parallelism and to enable real time applications. To meet this need, multiple SIMD architectures may be desired which are customized for the vision integration problem [28].

Acknowledgements

We especially thank Mike Daily for assistance in capturing synthetic motion-stereo images and DAP program debugging. Also, Rex Thanakij and Joseph Yadegar of AMT were helpful suggesting solutions to difficult DAP programming problems.

References

[1] J. Aloimonos and D. Shulman. *Integration of Visual Modules.* Academic Press, 1989.

[2] S. T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, 3:17–32, 1989.

[3] H.B. Barrow and J.M. Tenenbaum. Recovering intrinsic scene characteristics from images. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems.* Academic Press, 1978.

[4] M. Bertero, T. A. Poggio, and V. Torre. Ill-posed problems in early vision. *Proceedings of the IEEE*, 76(8):869–889, August 1988.

[5] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, 1987.

[6] William L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, PA, 1987.

[7] P. J. Burt. The pyramid as a structure for efficient computation. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 6–35. Springer-Verlag, 1984.

[8] J. J. Clark and A. L. Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer Academic Publishers, 1990.

[9] M. J. Daily and T. M. Silberberg. An active multiple camera system for object identification. *Submitted to: IEEE Conference on Computer Vision and Pattern Recognition*, June 3-6 1991.

[10] K. Fukushima. A neural network model for selective attention in visual pattern recognition. *Biological Cybernetics*, 55:5–15, 1986.

[11] E. B. Gamble, D. Geiger, T. A. Poggio, and D. Weinshall. Integration of vision modules and labeling of surface discontinuities. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1575–1581, 1989.

[12] E. B. Gamble and T. A. Poggio. Visual integration and detection of discontinuities: the key role of intensity edges. AI Memo 970, MIT Artificial Intelligence Laboratory, October 1987.

[13] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, June 1984.

[14] W.E.L. Grimson. Binocular shading and visual surface reconstruction. *Computer Vision, Graphics and Image Processing*, pages 18–44, 1984.

[15] J. J. Hopfield and D. W. Tank. Neural computation of decision in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[16] B.K.P. Horn. *Robot Vision.* McGraw-Hill, 1986.

[17] B.K.P. Horn and B.G. Schunck. Determining optic flow. *Artificial Intelligence*, 17:185–203, 1981.

[18] J. Hutchinson, C. Koch, J. Juo, and C. Mead. Computing motion using analog and resistive networks. *IEEE Computer*, pages 52–63, March 1988.

[19] C. Koch, J. Marroquin, and A. Yuille. Analog "neuronal" networks in early vision. AI Memo 751, MIT Artificial Intelligence Laboratory, June 1985.

[20] R. C. Luo and M. G. Kay. Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):901–931, September/October 1989.

[21] D. Marr. *Vision.* Freeman, San Francisco, 1982.

[22] N. M. Nasrabadi, S. P. Clifford, and Y. Liu. Integration of stereo vision and optical flow by using an energy minimization approach. *J. Opt. Soc. Am. A*, 6(6):900–907, June 1989.

[23] D. Parkinson and J. Litt, editors. *Massively Parallel Computing with the DAP.* MIT Press, 1990.

[24] T. A. Poggio, E. B. Gamble, and J. J. Little. Parallel integration of vision modules. *Science*, 242:436–440, October 1988.

[25] T. A. Poggio and C. Koch. Ill-posed problems in early vison : from computational theory to analogue networks. *Proceedings of the Royal Society of London*, B 266:303–323, 1985.

[26] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2):129–139, March 1986.

[27] D. Terzopoulos. Integrating visual information from multiple sources. In C.M. Brown, editor, *From Pixels to Predicates*, pages 111–142. Ablex Publishing Corporation, 1989.

[28] S. T. Toborg. A 3-d wafer scale architecture for early vision processing. In *Proceedings of the International Conference on Application Specific Array Processors*, pages 246–258, Princeton, NJ, 1990. IEEE Computer Society Press.

[29] S. T. Toborg and K. Hwang. Exploring neural network and optical computing technologies. In Hwang and DeGroot, editors, *Parallel Processsing for Supercomputing and Artificial Intelligence*, chapter 16. McGraw-Hill, 1989.

[30] A. L. Yuille. Energy functions for early vision and analog networks. *Biological Cybernetics*, 61:115–123, 1989.