Exploiting Process Integration and Composition in the Context of Active Vision

Jeffrey A. Fayman, Paolo Pirjanian, *Associate Member, IEEE*,
Henrik I. Christensen, *Member, IEEE*, and Ehud Rivlin, *Member, IEEE*

***Abstract*— The visual robustness of biological systems is in part due to their ability to actively integrate (fuse) information from a number of visual cues [2], [29]. In addition to active integration, the perception—action nature of biological vision demands event-driven behavioral composition. Providing mechanical vision systems with similar capabilities therefore requires tools and techniques for cue integration and behavioral composition.**

In this paper, we address two issues. First, we present a unified approach for handling both active integration and behavioral composition. The approach combines a theoretical framework that handles uncertainty using a voting scheme with a set of behaviors that are committed to achieving a specific goal through common effort and a well-known process composition model.

Secondly, we address the issue of integration in the active vision activity of smooth pursuit. We have experimented with the fusion of four smooth pursuit techniques, such as template matching and image differencing. We discuss each technique, highlighting strengths and weaknesses, and then show that fusing the techniques according to our formal framework improves system tracking behavior.

***Index Terms*—Active vision, module fusion, reliability, voting.**

I. Introduction

BIOLOGICAL vision systems are remarkably adept at providing useful, high-quality visual information in rich dynamic environments. These capabilities are, in part, a result of the inherent ability of such systems to dynamically adjust visual parameters to effectively integrate data from a wide range of visual cues and to compose modules in a timely event-driven manner. Research into the benefits and advantages of dynamic visual sensory systems over passive systems has been explored in the area of active vision. It has been shown that a moving system leads to improved robustness and the elimination of ill-posed conditions in several computer vision problems [3], [5], [6]. In addition to adjusting parameters, the visual robustness of biological systems is also due to their ability to actively integrate (fuse)[1] information from a number of visual cues [2], [29] as well as to compose behaviors in response to dynamic events.

Providing mechanical active vision systems with similar capabilities therefore requires tools and techniques for cue integration and behavioral composition. In this paper, we focus on two important features of a robust vision system: process integration and process composition.

A. Process Integration

While much of the research in active vision has focused on developing individual modules for implementing behaviors, such as fixation and smooth pursuit, it is becoming increasingly clear that a robust vision system should make use of a seamless integration of a number of functionally equivalent (homogeneous) or nonequivalent (nonhomogeneous) modules [29]. Indeed, integration forms the basis of a proposed extension of the Marr paradigm [2] in which the authors stress that future progress in computer vision will be a result of module integration: "Now that most of the modules have been studied in isolation, we think that it is time that they be tested in pairs, triples, and so on." The advantages of integration are evident in biological systems in which visual effectiveness lies in exploiting information from a variety of mechanisms, fusing the information to take advantage of strengths while avoiding the weaknesses of individual mechanisms under varying conditions [12].

B. Process Composition

While integration contributes to robust implementations of individual modules, the perception–action nature of active vision demands event-driven module composition. The composition mechanism must provide means for encoding temporal and structural dependencies necessary for effective behavior of reactive systems. This requires powerful yet flexible process composition tools [37].

C. Outline of the Paper

The paper is made up of two major parts. In the first part, we present a unified approach for handling both process integration and process composition. The approach combines a theoretical framework that handles uncertainty using a voting scheme with a set of behaviors that are committed to achieving

Manuscript received October 23, 1996; revised December 7, 1997. This work was supported in part by the European Community, Israel collaboration ECIS-003 project.

J. A. Fayman and E. Rivlin are with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa, 32000 Israel (jefff@virtue3d.com).

P. Pirjanian is with the Laboratory of Image Analysis, Aalborg University, Aalborg East, Denmark.

H. I. Christensen is with the Centre for Autonomous Systems, Royal Institute of Technology, Stockholm, Sweden.

Publisher Item Identifier S 1094-6977(99)00103-0.

[1] In this work, the terms active integration and fusion are used interchangeably.

a specific goal through common effort and a well-known process composition model. In the second part of the paper, we show that our approach indeed works. We address the issue of integration in the active vision activity of smooth pursuit. While a variety of techniques exist to implement smooth pursuit, each of these techniques has strengths and weaknesses, making it robust under some conditions and weak under others. The ability to fuse these techniques such that their strengths can be exploited and their weaknesses avoided naturally leads to more robust system behavior. We have experimented with the fusion of four smooth pursuit techniques. We discuss each technique, highlighting strengths and weaknesses. We then show that fusing the techniques according to our formal framework improves tracking.

The remainder of the paper is organized as follows. In Section II, we review related work. In Section III, we introduce the approach we have adopted for handling module composition. In Section IV, we extend the notation presented in Section III with the capability to handle integration. Next, in Section V, we discuss the theoretical framework we have adopted for the fusion of redundant multivalued behaviors. In Section VI, we discuss the active vision activity of smooth pursuit and focus on four motion estimation techniques for its implementation. These will be used in Section VII, where we present experiments in fusing the results of the motion estimation techniques leading to more robust smooth pursuit behavior. We conclude with Section VIII.

II. Related Work

A. Active Vision

Since 1985, when active vision first began to appear in the literature [3], [5], [6], the topic has received a dramatic increase in interest. Initial work focused on building active vision devices and understanding and transferring to these devices capabilities possessed by biological vision systems, such as focus [23], saccades [9], smooth pursuit [9], [15], fixation [28], attention [38], and prediction [11]. While several of these works point out that many times these capabilities are composed of various cues (e.g., disparity and accommodation in vergence [13], [28]), to the best of our knowledge, no previous work exists in which benefits of the fusion of a number of homogeneous modules are exploited to improve individual visual capabilities.

B. Process Composition

Process composition by representing task/plans as networks of processes was first proposed by Lyons *et al.* [24], [25], in which the Robot Schemas (RS) model is discussed. Kosecka *et al.* [21] adopt RS and show how we can synthesize a finite state machine supervisor that serves as a discrete event controller. Elementary behaviors appropriate in the domain of an "intelligent delivery agent" are described, and experiments in robot navigation are presented. Our work also makes use of the RS model, however, we recognize the importance of module fusion to the robust behavior of active vision systems and extend the power of robot schemas to encompass both module integration and module composition in a unified manner.

C. Reliability

Reliability in robotics has been studied along several main paths: *reactivity*, *error recovery*, and *uncertainty handling*[2]. Reactive behavior-based systems [4], [10] provide immediate responses to unpredictable environmental changes through a tight coupling of perception and action. Reactive architectures have shown improved reliability when compared with classical sense-plan-act architectures. In error recovery techniques, a set of dedicated modules monitor the task continuously and, upon detection of an error (cognizant failure), an appropriate corrective action is invoked, which can handle unanticipated situations. Error detection and recovery was first introduced to robotics in [16]. The task control architecture (TCA) [36] facilitates a methodology for incremental development of reliable autonomous agents. We start with a deliberative plan constructed to work correctly for foreseeable situations and then incrementally add task-specific monitors and exception handling strategies (reactive behaviors) that detect and handle unpredicted situations. A similar approach is advocated in [20]. In [31], a fault tolerance technique using redundant sets of behaviors was investigated. In this approach, the system is provided with a redundant set of behaviors to perform a task under different conditions. For each behavior, a performance model exists and a failure is detected if the behavior performs worse than expected. Upon detection of a failure, a new behavior is invoked until an acceptable behavior is selected. This approach has several points in common with the approach presented in this paper. The major similarity is the exploitation of redundancy of homogeneous behaviors/modules to reduce systems sensitivity to uncertainties. The differences lie in that our approach does not utilize explicit models, which make it more appropriate for situations in which it is difficult to obtain such models. Secondly, while in [31] a sequential invocation of behaviors is utilized, we propose a simultaneous invocation using voting (this is actually how use of models is avoided).

The basic idea behind uncertainty handling techniques is to have explicit "models" of the robot's sensing and action capabilities. Using this model, the agent (robot) can predict what actions to take to increase the expected utility or the reliability of its task [27]. Various combinations, such as reactivity and uncertainty handling, have been explored. The usual approach is to combine (high-level) deliberative planners that have uncertainty handling capabilities with (low-level) reactive modules (behaviors) that handle run-time contingencies [22], [34]. The most popular approaches for reasoning under uncertainty are based on probability theory [32], Dempster–Shafer theory [35], fuzzy set theory [39], and certainty factor formalism. Methods that exploit redundancy are usually based on data/sensor fusion techniques [1], [17], in which reliability is improved by pooling evidence.

[2]Uncertainty handling can be further divided into the two related areas of *explicit* uncertainty handling and methods exploiting *redundancy* (such as sensor fusion).

TABLE I
SUMMARY OF *RS* COMPOSITION OPERATORS

1. *Sequential Composition*: $T = P; Q$. The process T behaves like the process P until that terminates, and then behaves like the process Q (regardless of P's termination status).
2. *Concurrent Composition*: $T = (P|Q)^c$. The process T behaves like P and Q running in parallel and with the input ports of one connected to the output ports of the other as indicated by the port-to-port connection map c.
3. *Conditional Composition*: $T = P\langle v\rangle : Q_v$. The process T behaves like the process P until that terminates. If P aborts, then T aborts. If P terminates normally, then the value v calculated by P is used to initialize the process Q, and T then behaves like Q_v.
4. *Disabling Composition*: $T = P\#Q$. The process T behaves like the concurrent composition of P and Q until either terminates, then the other is aborted and T terminates. At most one process can stop; the remainder are aborted.
5. *Synchronous Recurrent Composition*: $T = P\langle v\rangle :; Q_v$. This is recursively defined as $P :; Q = P : (Q; P :; Q)$.
6. *Asynchronous Recurrent Composition*: $T = P\langle v\rangle :: Q_v$. This is recursively defined as $P :: Q = P : (Q|(P :: Q))$.

Sensor integration/fusion has the potential of reducing overall uncertainty, overcoming sensor imperfections, and producing more reliable results. In [1], image segmentation is performed in a framework in which fuzzy set theory is adopted for sensor fusion, where uncertainty is modeled using membership functions. A drawback of fuzzy set theory is that the estimation of the fuzzy sets or membership functions can be cumbersome if no clear guidelines can be found. However, recent works have proposed methods for estimating membership functions, in particular, from statistical data, that partially avoid this problem. Fuzzy set theory also provides a large set of combination operators, which allow for adaptive fusion. Therefore, it should be mentioned that fusing behaviors with different reliabilities using fuzzy set theory is an interesting alternative to simple voting schemes.

In [26], a decentralized approach to data fusion is introduced. This approach is based on Bayesian reasoning and utility theory. The Bayesian methods, however, may not be suitable in certain applications for two reasons: *a priori* probabilities can be difficult if not impossible to obtain and, in many cases, we will need to express ignorance with regard to specific choices. Under these conditions, other techniques are called for. Dempster–Shafer theory is suitable for handling ignorance and does not require *a priori* probabilities. However, in the general case, Dempster's rule of combination has an exponential computational complexity. In specific cases, this can be circumvented and tractable implementations are possible.

III. MODULE INTEGRATION AND COMPOSITION

We distinguish between two forms of module integration: integration of functionally equivalent (homogeneous) modules and composition of functionally nonequivalent (nonhomogeneous) modules. The goal of integrating homogeneous modules is to improve quality (in the sense that the set of integrated modules are better able to handle uncertainty than any of the individual member modules), while the goal of process composition is to encode the temporal and structural dependencies between processes necessary for effective behavior (decisions/actions/commands). We adopt the notation of the well-known RS model that proposes an elegant notation for temporal structuring of processes. However, RS does not address the quality issue (integration of homogeneous modules) in an explicit manner.

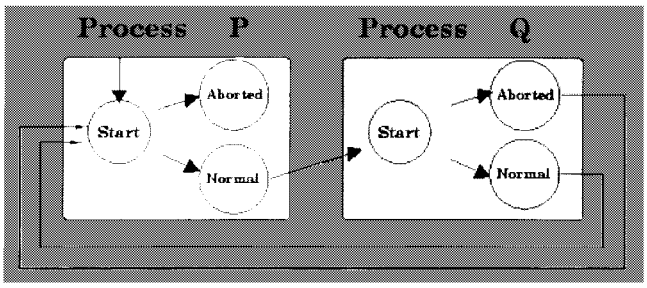
In order to get the benefits of improved quality along with the temporal structuring necessary for effective behavior, we extend the RS notation with a fusion operator. In this section, we discuss the RS model. In the next section, we show how we have extended the RS model to incorporate integration in an explicit manner, and then in Section V, we present a formalism for process integration.

A. Robot Schemas

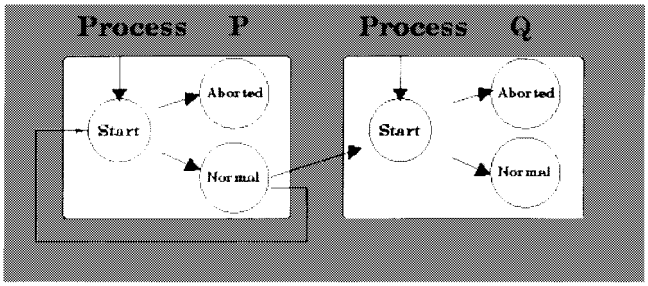
Process composition in our system is based on a model proposed in [24] and [25] called RS. RS provides notation for specifying process concurrency that captures the temporal and structural dependencies required to implement complex perception–action tasks, such as those demanded by active vision.

Table I summarizes the RS composition operators. In the RS model, communication channels between concurrent processes are called "ports." Messages are written to, and read from, ports. A port-to-port *connection relation* can be specified as an optional third parameter in concurrent composition. This connection relation specifies a set of couples $op \mapsto ip$, indicating that port ip and op are connected.

1) *Sequential Composition*: $T = P; Q$. The process T behaves like the process P until that terminates and then behaves like the process Q (regardless of P's termination status).
2) *Concurrent Composition*: $T = (P \mid Q)^c$. The process T behaves like P and Q running in parallel and with the input ports of one connected to the output ports of the other, as indicated by the port-to-port connection map c.
3) *Conditional Composition*: $T = P\langle v\rangle : Q_v$. The process T behaves like the process P until that terminates. If P aborts, then T aborts. If P terminates normally, then the value v calculated by P is used to initialize the process Q and T behaves like Q_v.
4) *Disabling Composition*: $T = P\#Q$. The process T behaves like the concurrent composition of P and Q



(a)



(b)

Fig. 1. (a) Synchronous recurrent composition; (b) asynchronous recurrent composition.

until either terminates, and then the other is aborted and T terminates. At most, one process can stop; the remainder are aborted.

5) *Synchronous Recurrent Composition*: $T = P\langle v\rangle \;;; Q_v$. This is recursively defined as $P \;;; Q = P : (Q; P \;;; Q)$.
6) *Asynchronous Recurrent Composition*: T = P \langle v \rangle :: Q_v. This is recursively defined as $P :: Q = P : (Q \mid (P :: Q))$.

Of the six RS operators, *Synchronous Recurrent Composition* and *Asynchronous Recurrent Composition* require further explanation. Both operators express iteration. *Synchronous Recurrent Composition* specifies the standard "loop" construct in which process P and Q are iteratively executed sequentially. Execution of the loop P followed by Q continues until process P aborts. *Asynchronous Recurrent Composition* is a similar looping mechanism; however, in this case, process Q is not required to terminate before proceeding with the iteration. This enables multiple instances of process Q to be generated and executed concurrently. The behavior of these operators is illustrated in Fig. 1.

For an interpretation of the RS operators in the context of active vision, we express the high-level active vision activities of fixation and pursuit in RS notation. Fixation is responsible for centering the image of an object and is initialized with a saccade to the fixation point followed by continuous vergence control driven by disparity and accommodation cues

$$\text{fixation} = \text{saccade; ((disparity} \mid \text{accommodation) :; (vergence control)).}$$

Pursuit is employed for tracking objects during motion. The motion may originate from egomotion or object motion. In both cases, the objective of pursuit is to stabilize the image of the object at image center. In pursuit, vergence, foveal motion detection, and dynamic accommodation are used to continuously drive motion of the vision sensor

$$\text{pursuit} = \text{(vergence \# foveal motion \# accommodation) :; move.}$$

IV. Incorporating Redundancy

In this section, we extend the RS model with a fusion operator so that integration of homogeneous modules is provided in an explicit manner. The RS notation of Table I is based on the composition of individual processes that are defined as a "unique locus of computation" or *basic schemas*, such as a piece of hardware or physical agent of change. It is not possible to describe or analyze behavior below the level of a basic schema.

We augment the RS model to include fusion capabilities with the addition of the **integrating composition** operator **fuse**, which extends the definition of a *schema* to include the fused output of a set of behaviors. A set of homogeneous behaviors that in combination pursue a specific goal are called a *behavior team*. Given a behavior team made up of behaviors b_i, we define the behavior of the schema B resulting from *integrating composition* as follows:

$$B = \text{fuse}(b_1, b_2, \ldots, b_n).$$

The formal description of the operator **fuse** is left undefined as there are various ways to integrate modules, as discussed in Section V. For the purposes of this paper, we use plurality approval voting. Note that we have not modified the existing RS operators in any way, we have only extended the definition of a basic schema in a well-defined manner. The formalism presented in Section V shows that the reliability of schema B will be at least as good as the reliability of any of the b_i. As a result of this extension of the definition of a schema, we retain all of the analytic power of the RS model and provide module integration benefits of Section V.

To give an example of how the new operator is used, we will rewrite the definition of *pursuit* in Section III

```
pursuit = (vergence # foveal motion
          # accommodation) :; move.
```

to include module integration. Pursuit can now be defined as follows:

```
pursuit
  = (vergence # fuse(blob, idiff,
    edge, sob) # accommodation) :; move.
```

The four motion detection techniques, BLOB, IDIFF, edge, and SOB, are integrated under the **fuse** operator, yielding their integrated behavior. Our experiments in Section VII verify that the use of the **fuse** leads to dramatic improvements in tracking ability over any individual technique.

V. Uncertainty Handling Using Voting Schemes

In this section, we present an approach for constructing reliable modules from less reliable ones. A team of redundant modules vote for a set of possible actions. The votes are then combined, and the most appropriate action, corresponding to the action with the highest number of votes, is chosen. Voting is a common technique for construction of reliable hardware components and critical systems, such as certain components used in the aviation industry. Simplicity is a virtue of voting techniques and enables cost-effective and efficient hardware as well as software implementations. The basic idea behind this approach is similar to that of sensor fusion with the hypothesis that the overall reliability will be improved by combining pieces of evidence provided by independent/partially independent sources. However, in the approach presented here, no explicit model, probabilistic or otherwise, is used in the fusion process. It is thus interesting to investigate how simple model-free voting techniques can be used to improve the reliability of purposive modules.

A. Homogeneous Modules

We formalize a module b as a mapping from an action space Θ to the interval $[0,1]$

$$b : \Theta \rightarrow [0,1]. \tag{1}$$

The action space $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ is defined to be a finite set of possible actions or control parameters. The mapping assigns to each action $\theta \in \Theta$ a preference, in which the most appropriate actions are assigned one and undesired/illegal actions are assigned zero. Consider having a set of modules, B, all providing the same objective, such as object tracking, obstacle avoidance, or door traversal.

A set of modules with the same objective will be denoted *homogeneous modules*[3]. The output of the modules are combined using a voting technique δ, and the most appropriate action chosen is θ', where

$$\delta(\theta') = \max\{\delta(\theta) \mid \theta \in \Theta\}. \tag{2}$$

Fig. 2 illustrates how the outputs are combined using the composition operator δ, producing a new preference over the action space.

B. Voting Schemes

In the literature of reliability theory, numerous voting schemes have been proposed, and in [30], a taxonomy is given for existing classes of voting schemes. The most used and most general voting schemes are majority voting and m-out-of-n voting, respectively, which belong to the same class of voting known as *weighted consensus voting*. In majority voting, an action is chosen that has received more than half of the total number of votes. In m-out-of-n voting, an action is selected if it receives m or more votes out of n. In their original formulations, however, weighted consensus voting schemes require that each module votes for only one action—the most appropriate one from a module's point of view. In the case of purposive modules, more than one action can be appropriate; thus, a variant of weighted consensus voting, known as *approval voting*, is called for.

[3]Even though they have the same objective, these modules can be based on different sensing modalities, algorithms, etc., to achieve the same objective.

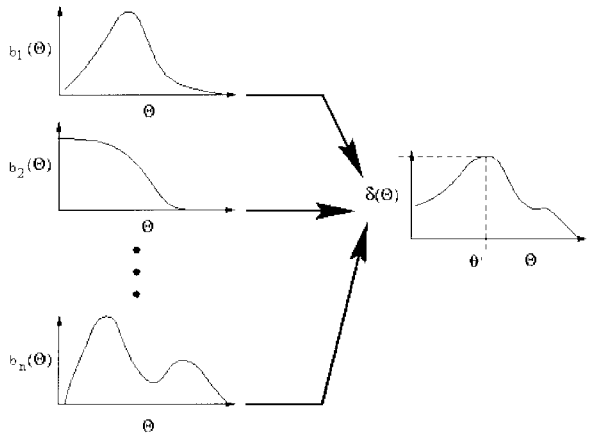


Fig. 2. Schematic of the composition process in a team of homogeneous modules. The modules generate their votes (left), which are combined using the composition operator. The composed module is illustrated on the right.

Definition V.1 (m-out-of-n Approval Voting): An approval voting scheme $\delta : \Theta \rightarrow [0,1]$, in which n is the number of homogeneous modules, is defined in the following way:

$$\delta(\theta) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} b_i(\theta), & \text{if } \sum_{i=1}^{n} v_i(\theta) \geq m \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where

$$v_i(\theta) = \begin{cases} 1, & \text{if } b_i(\theta) > 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i = 1, \ldots, n \tag{4}$$

is the voting function that determines whether a module votes for a given action.

A module votes for an action if its preference for that specific action is >0. If $\geq m$ modules vote for an action, θ, then their preferences are combined according to the original approval voting scheme, which is an (possibly weighted) addition of the preferences.

In this paper, we experimentally show that fusion of homogeneous modules using such simple voting schemes can improve system reliability. In the following, we characterize the reliability of a team of homogeneous modules and use the reliability for selecting an appropriate m-out-of-n approval voting scheme. More specifically, the parameter m has to be chosen to ensure improvement in system reliability. In order to enhance the clarity of the paper, we only present important properties from the theoretical characterization. For a complete analytical treatment of performance characterization, see [33]. The general problem (i.e., including other classes of voting schemes) of selecting the optimal voting scheme that maximizes system reliability is investigated in [8].

C. Reliability of Teams of Homogeneous Modules

The reliability of a behavior b_i is defined as its probability of success and is denoted p_i. The reliability of a team of homogeneous modules using m-out-of-n approval voting, denoted $r_m(p_1, \ldots, p_n)$, is defined as a function of m and

the reliability of the modules constituting the team $p_1, \ldots, p_n$. $r_m(p_1, \ldots, p_n)$ expresses the probability that the system is successful if at least m of its n behaviors are successful.

Using these definitions, it can be shown that the reliability of a m-out-of-n system is greater than or equal to the reliability of a $(m+1)$-out-of-n system. Thus, by induction, it can be concluded that a 1-out-of-n system will provide the maximum reliability. This result (as shown in [8]) applies generally, i.e., with no assumptions about statistical independence between the modules. A 1-out-of-n voting strategy is denoted plurality voting and chooses the action that has received the maximum number of votes. Based on these results, we use a plurality approval voting in this paper.

VI. Smooth Pursuit and Cue Generation

The visual robustness of biological systems is due in part to the update of visual system parameters, which ensures continuous delivery of high-quality images of salient objects. Smooth pursuit is a primary visual behavior used by biological systems for this purpose and is a topic of much interest in the active vision community. Smooth pursuit is the process by which a moving object is tracked. The goal is to keep the retinal position of the moving object inside the fovea. Benefits of smooth pursuit include object stabilization in the image as tracking emphasizes the signal of the target over the background and the localization of image processing to the region of the fovea [15].

In both biological and machine vision systems, motion analysis provides the basis of smooth pursuit. The smooth pursuit behavior team used in our experiments (Section VII) consists of the following motion analysis techniques: *blob tracking*, *edge tracking*, *image differencing*, and *template matching*. We chose these techniques as they are relatively simple to implement and are sufficient for the purposes of our experiments. We now briefly discuss the techniques focusing on strengths and weaknesses.

A. Blob Tracking

Due to its simplicity and suitability for real-time implementation, blob tracking has been perhaps the single most commonly used technique for motion detection in tracking systems. Numerous works, such as [14], have reported systems that are able to track a black or white blob. The source of many of these systems is a moving light, such as a flashlight.

The strength of blob tracking lies in its simplicity, making it suitable for real-time implementation. However, it is unable to handle multiple objects and is not useful in realistic environments, as it assumes an object with high contrast relative to the background.

In our implementation, we assume a dark blob moving over a light background. The algorithm thresholds the input images to segment the blob from the background and then computes and returns the centroid of the blob.

B. Edge Tracking

Edge tracking is similar to blob tracking. However, rather than finding the centroid of the entire blob, the centroid of blob edges is found; therefore, an edge operator is used on the input image to find the edges. Strengths and weaknesses of edge tracking are similar to those of blob tracking.

In our implementation, the Sobel edge detector [7] is used to find the edges. The resulting edge image is then thresholded to segment the object from the background. The centroid of the edges is then computed and returned.

C. Image Differencing

Image differencing is the simplest of our motion analysis techniques. Image differencing is performed by taking the difference of two consecutive frames

$$d(n,x,y) = f(n,x,y) - f(n-1,x,y) \tag{5}$$

where $d(n,x,y)$ is the difference image, $f(n,x,y)$ is the current frame, and $f(n-1,x,y)$ is the previous frame. Differencing segments the scene into static and moving regions as only objects that have changed position between two consecutive images will have nonzero pixel values.

The strengths of differencing lie in its simplicity, making it suitable for real-time implementation, and its ability to handle multiple objects. However, the structure of the moving object has to be simple in order for subtraction to segment one object into one motion region. A problem with using image differencing in smooth pursuit is that retinal motion of the background induced by camera movement can be mistaken as object motion unless this motion is first subtracted out of the image.

In our implementation, the centroid of all pixels falling above a grayscale threshold was computed over a two times subsampled fovea of 100 $\times$ 100 pixels in the difference image. Here, thresholding eliminated pixels that would have mistakenly been used in the centroid computation when, in fact, they were an artifact of noise introduced by the imaging process.

D. Template Matching

The idea behind template matching is to find the location of a particular object in an image by searching the image for instances of a second, smaller image called a "template" that contains the object. The template matching algorithm compares the template with the image at different image locations and finds the location in the image that best matches the template.

Correlation provides the basis of template matching. For each image location, a similarity measure is computed, indicating how well the template matches the image at that location. The image location that provides the maximal similarity measure is selected as the location of the object in the image.

Differences in various template matching techniques are usually found in the method used for computing the similarity measure. Several common techniques include the sum of squared differences (SSD) technique and normalized cross correlation. We use the SSD for computing the similarity measure in our implementation.

TABLE II
MODULE TIMING ANALYSIS. EFFECTIVE SAMPLE RATE GIVES THE SPEED AT WHICH EACH MODULE RUNS INDEPENDENTLY, WHILE THE ACTUAL SAMPLE RATE IS THE RATE USED IN THE EXPERIMENTS

Module	Effective Sample Rate	Actual Sample Rate
Blob Tracking	75 Hz.	10 Hz.
Image Differencing	20 Hz.	10 Hz.
Edge Tracking	42 Hz.	10 Hz.
Template Matching	58 Hz.	10 Hz.
Fusion	10 Hz.	10 Hz.

The SSD similarity between a function $f(x)$ and a template $t(x)$ is given by

$$\mathrm{SSD}(y) = \sum_{x=-1}^{M} \sum_{y=-1}^{N} [f(x) - t(x-y)]^2 \qquad (6)$$

where M and N are the size of the template.

Strengths of template matching include its ease of implementation and efficient calculation over the entire set of locations. However, template matching is sensitive to changes in object shape, size, orientation, and changes in image intensities.

E. Cue Implementation Details

The reliability of each of the motion analysis modules is global in the sense that it is dependent on variables, such as lighting, focus, etc. While it would be interesting to analyze the effects of such variables on tracking, for the purposes of our work, we were not interested in finding implementations that provide the best possible results. We in fact show that it is not critical to have the most accurate implementation, as fusion tends to enhance overall performance. Therefore, our implementations are simple and straightforward. Understanding the effects of these other variables and improving the behavior of the individual techniques is an orthogonal effort and can only help to improve the results of fusion.

In all but template matching, image thresholding is used to segment the object from its background. For each technique, gray-level thresholds were empirically chosen to provide the best segmentation over the entire range of experimental scenarios (Section VII). Timing analysis for the modules is given in Table II.

The effective sampling rate gives the speed at which each module can run independently. However, in order to obtain comparable results, we eliminate variations in sampling rate that may influence the performance of the modules (often higher sampling rates lead to improved performance). We normalized the sampling rate by executing all modules sequentially while actuating the motors based on the results of the particular module we were testing only.

VII. EXPERIMENTS

In this section, we present experiments in which we use the tools and motion analysis techniques presented in previous sections to execute smooth pursuit while exploiting the benefits of integration. Our experiments focus on module integration. In previous works, we have experimented with process composition [18], [19]. We begin with a discussion of what we want to show and how it can be shown. We then present our experiments followed by a discussion of the results.

A. Experimental Setup

The goal of our experiments is to investigate the hypothesis that fusion of homogeneous behaviors can lead to improved reliability. In particular, we want to show that active vision can indeed benefit from module fusion. In order to show this, we have implemented the four cue generation modules discussed in Section VI as well as a module that fuses their results by approval voting. With this configuration, we run a series of tracking experiments on a robotic head. In each experiment, a robotic manipulator effects horizontal translatory motion consisting of two motion segments: from the starting location to a location 200 cm to the right and return to the starting location. The speed of the manipulator was set so that the entire 400-cm manipulator motion was completed in approximately 15 s.

In the experiments, we measured the ability of each module to track various combinations of objects and backgrounds. We call each such combination a *scenario*. The six scenarios used in our experiments are shown in Fig. 3. As can be seen in the figure, objects range from a simple blob to real objects. Similarly, backgrounds range in complexity from a constant beige background to a natural background with randomly placed objects.

We tested the performance of each of the five modules [blob tracking (BLOB)], image differencing (IDIFF), edge tracking (SOB), template matching (TM), and fusion (FUSE)], on each scenario. A total of 30 *experiment sets* were conducted using the six scenarios, in which each experiment set consisted of testing each of the four motion tracking modules plus the fusion module on a scenario. To account for variations in lighting and other conditions, each scenario was used for five experiment sets (corresponding to 150 single experiments). The scenarios were chosen so as to push the limits of one or more of the modules at a time so that they would fail. We used the number of failing modules on a particular scenario as a measure of the scenarios complexity, e.g., if a setting[4] a leads to the failure of three of the modules and another setting b leads to the failure of one module, and then we say that setting a is more complex than setting b. This allows us to investigate the performance of the fusion module relative to the complexity of the scenario. The histogram in Fig. 4 shows the distribution of the number of failed modules[5] for each of the 30 experiments; e.g., in 11 of the experiments, all but one of the modules failed to track the object.

Two measures are used to quantify each module's ability to track: *absolute error* and *relative error*. Absolute error is a yes/no answer to the question of whether a module successfully tracks during a single experiment. A module is said to track an object successfully if some part of the object is located at the image center during the entire motion sequence, otherwise it has failed to track. The ratio of the number of

[4] We call a scenario, lighting conditions, etc., during the experiments a *setting*.

[5] Scenario complexity is determined based on the performance of the primitive modules; thus, the fusion module is not considered in the histogram.

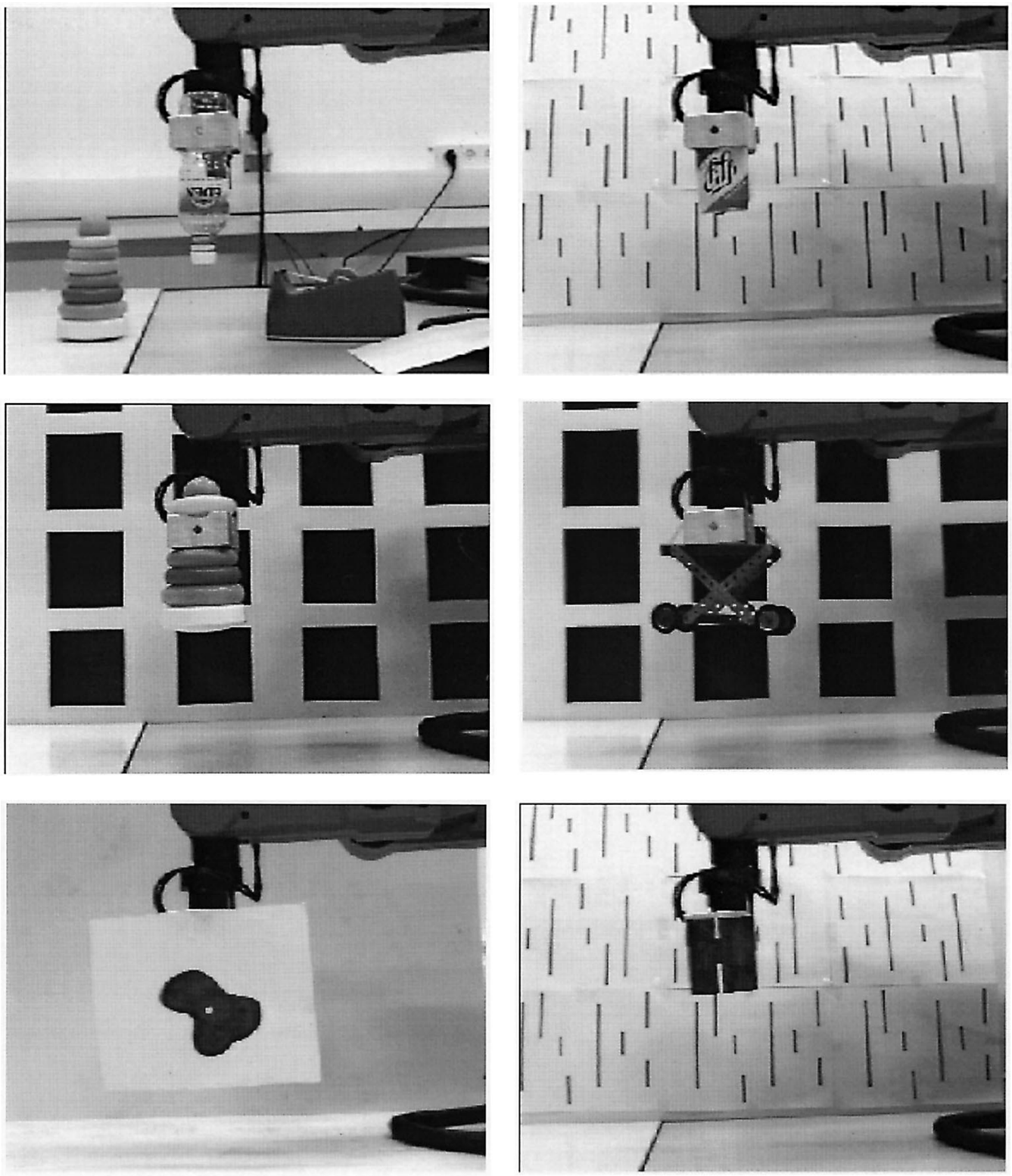


Fig. 3. In the experimental scenarios, various combinations of objects and backgrounds provide differing complexities.

successful runs to the total number of runs provides a measure of module reliability.

Relative error quantifies the quality of tracking, i.e., a measure of how well tracking is performed. As an expression for relative error, we use the distance (in pixels) from the image center (where the tracked object should be in the ideal case) to a **fixed** point on the moving object. We term this expression *distance error*. We also separate this distance into its X (horizontal) and Y (vertical) components to investigate the contributions of these components to the error. Here we present the mean and standard deviation of the distance error, along with the mean and standard deviations of the

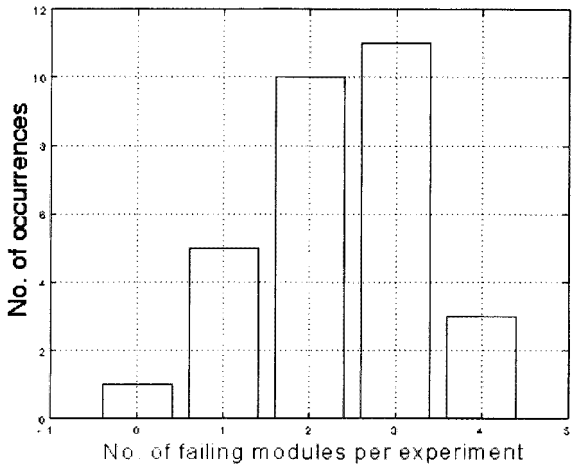


Fig. 4. Distribution of the number of failing modules at each trial during 30 trials. The fusion module is not taken into consideration here. Setting complexity increases from left to right.

TABLE III
Absolute Error. MODULE SUCCESS AND FAILURE RATES. RELIABILITY IS CALCULATED AS THE RATIO OF SUCCESSFUL RUNS TO THE TOTAL NUMBER OF EXPERIMENTS. THE MAIN RESULTS ARE HIGHLIGHTED

Module	No. of successes	No. of Failures	Reliability
Blob Tracking	11	19	**36.7%**
Image Differencing	14	16	**46.7%**
Edge Tracking	12	18	**40.0%**
Template Matching	13	17	**43.3%**
Fusion	24	6	**80.0%**

TABLE IV
Relative Error. MODULE PERFORMANCE RESULTS FOR 30 RUNS. RELATIVE ERROR IS PRESENTED BY THE MEAN DISTANCE ERROR AND STANDARD DEVIATION. THE STATISTICS FOR THE CORRESPONDING X AND Y COMPONENTS ARE ALSO LISTED. THE MAIN RESULTS ARE HIGHLIGHTED

Module	Mean X	Std.Dev X	Mean Y	Std.Dev Y	Mean	Std.Dev.
Blob Tracking	136.1	188.9	110.0	198.1	**185.9**	**266.4**
Image Differencing	44.0	86.4	12.8	48.8	**49.9**	**97.3**
Edge Tracking	72.1	109.0	25.9	92.7	**82.6**	**139.7**
Template Matching	76.4	125.1	30.2	111.8	**88.6**	**164.4**
Fusion	17.2	24.1	5.2	25.8	**19.8**	**34.3**

corresponding X and Y components, for the 30 experiments (see Table IV). To obtain the distance error, we recorded each motion sequence to videotape. Using the videotape, we manually extracted the pixel distance between image center and the fixed point on the moving object over the duration of the tracking.[6] This determines the "ground truth." To make the results of ground truth extraction as accurate as possible, we overlayed a "+" symbol at the image center and selected the object point located under the "+" at the start of tracking as the fixed object point for the rest of the sequence. Further, in order to analyze the behavior of the modules, we recorded their outputs, denoted as *tracking results*, which where used to control the camera head. Since the tracking results themselves did not contain information about how well the actual tracking was going, we had to relate them to the ground truth. Thus, in order to correspond the ground truth with the tracking results, we logged, along with the tracking results, the sample/frame number at which the results were generated. We printed the same sample number on (the upper left corner of) each frame of the video sequence.

[6]Every fifth frame of each video sequence was analyzed manually to obtain the distance error.

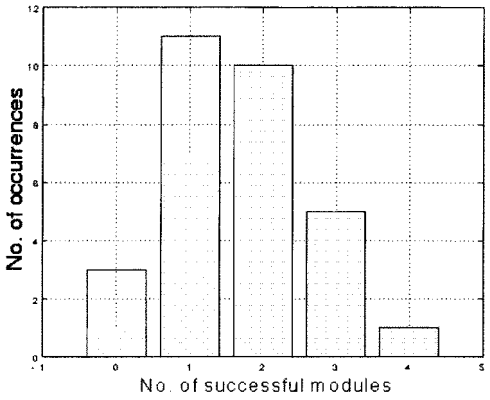


Fig. 5. Comparison between the success of individual modules and the success of fusion. The histogram represents 30 trials. The bars of the histogram represent the number of times there were n successful modules, where n ranges from zero to four. The shaded portion of the bars represent the number of times that fusion succeeded. The most interesting part of the histogram is in the first segment (zero successful modules). What this is saying is that in three of the 30 experiments, none of the individual modules successfully tracked the object. However, in one of those three cases, fusion succeeded. This is a powerful point. Even though, no individual tracking module was able to track the object, the fusion of the modules succeeded. We call this phenomenon "corrective reinforcement."

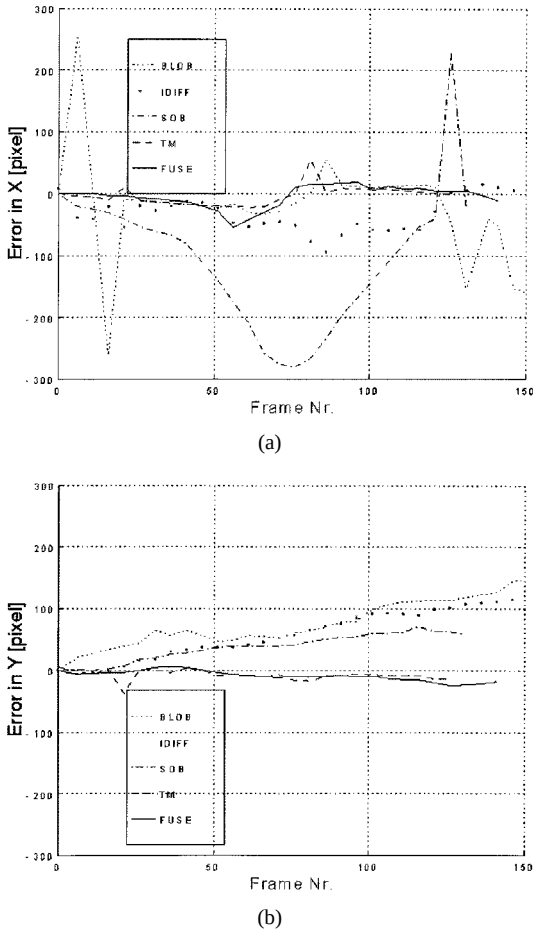


Fig. 6. Plot of distance error for the modules during experiment. (a) Distance error in the X (horizontal) direction. (b) Distance error in the Y (vertical) direction.

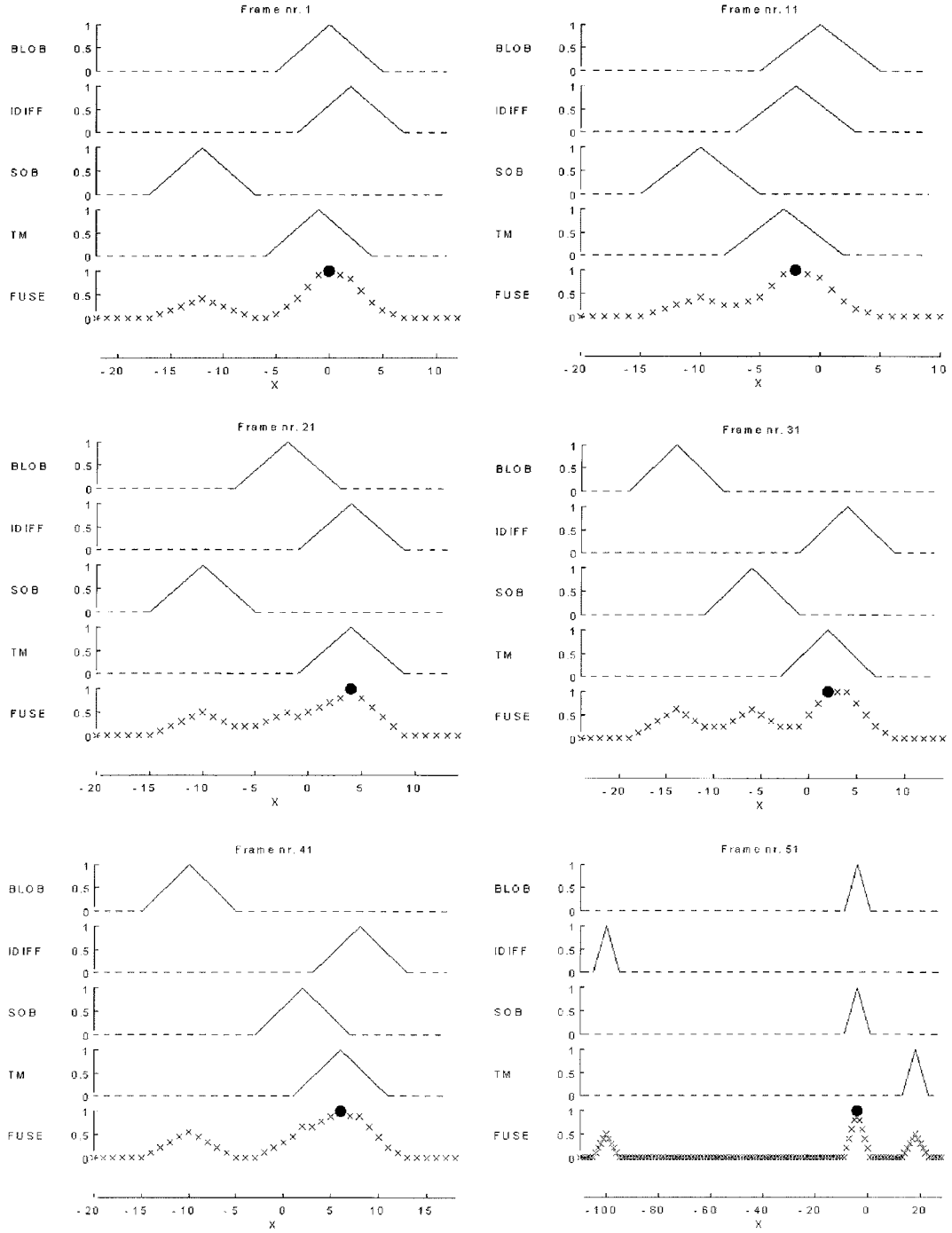


Fig. 7. Sequence showing the tracking results and the fusion process. During the sequence, the camera is driven by the fusion module. The plots present each module's votes for each action (here motion in the horizontal direction). These votes are combined by the fusion module (FUSE), and the best action indicated by the dot in the plots is selected. Sequence showing the tracking results and the fusion process.

B. Experimental Results

Experimental results are presented in Tables III and IV. Absolute error results are listed in Table III along with module reliabilities. Columns 2 and 3 in the table list the number of successful runs and the number of failures, respectively. Module reliabilities, listed in the last column, are calculated as the ratio of the number of successful runs to the total number of runs (30). Table IV summarizes the relative error results and, in particular, the mean and standard deviations of the distance error. The table also lists for both X and Y the mean value of distance error along with corresponding standard deviation.

The plot in Fig. 5 illustrates the performance of the fusion module as a function of scenario/setting complexity. The figure consists of two superimposed histograms, one (solid frames) showing the distribution of the per-trial number of successful modules and the other (filled area) showing the corresponding portion of the successful fusion trials. In the figure, the complexity of the experimental setting decreases from left to right—the fewer modules that succeed in a setting

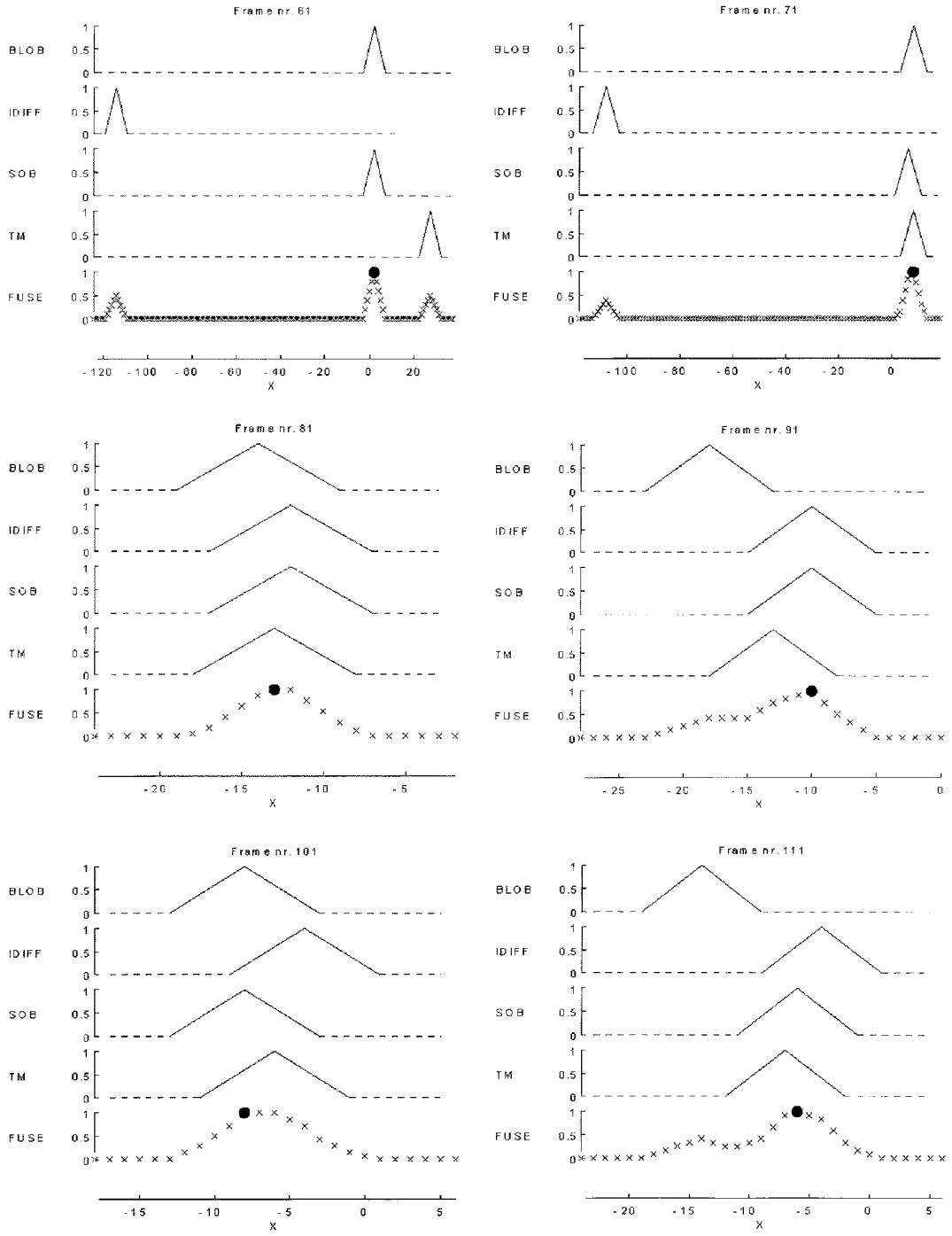


Fig. 7. (*Continued.*) Sequence showing the tracking results and the fusion process. During the sequence, the camera is driven by the fusion module. The plots present each module's votes for each action (here motion in the horizontal direction). These votes are combined by the fusion module (FUSE), and the best action indicated by the dot in the plots is selected. Sequence showing the tracking results and the fusion process.

the more complex it is. As can be seen, in three of the experiments, none of the modules succeed. Nonetheless, the fusion module (surprisingly) succeeds in one of these three cases. Note also that, in 11 of the cases, only one module succeeds, but in seven out of the 11 cases, the fusion module manages to track successfully. How the fusion can succeed, even though all or the majority of the individual modules fail, is interesting and will be explained intuitively in the remainder of this section. We are planning a theoretical analysis of this effect in future work.

In order to explain this effect and to highlight several interesting aspects of fusion, we present data from a single experiment. The scenario used for this particular experiment is shown in Fig. 3 in the last row, second column. Fig. 6 is a plot of the distance errors (the sign of the error is included in the plots) in X and Y, respectively. Looking at the figures, we see that in this scenario, the BLOB, IDIFF, and SOB modules are unable to track the object. On the other hand, template matching (TM) and fusion (FUSE) succeed in tracking.

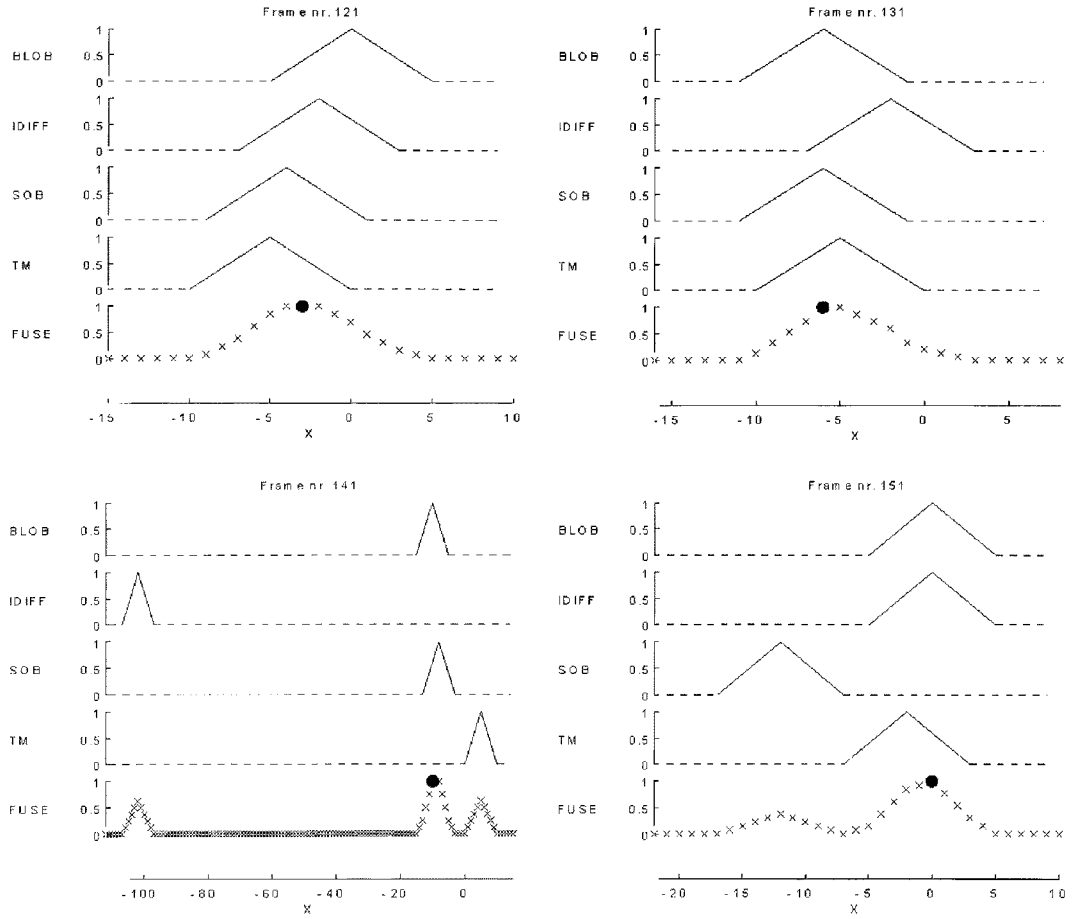


Fig. 7. (*Continued.*) Sequence showing the tracking results and the fusion process. During the sequence, the camera is driven by the fusion module. The plots present each module's votes for each action (here motion in the horizontal direction). These votes are combined by the fusion module (FUSE), and the best action indicated by the dot in the plots is selected. Sequence showing the tracking results and the fusion process.

Fig. 7 contains plots of the actual outputs generated during one experiment. The plots show the outputs generated by the five modules over 151 frames (with a step of ten frames, i.e., 1 s). The output of each module is illustrated as a triangular window of width equal to ten pixels.[7] Only the plots for the X-axis are shown. The output of each module should be interpreted as the vote for moving the image center to a given pixel. The commanded output is given to the camera head driver, which translates the pixel values to joint angles and drives the motors. The commanded output is chosen as the output with maximum vote. The vertical axis, in each plot, determines the votes that range from zero to one, with one being the most desirable and zero the least desirable. The last plot within each subplot is the output of the fusion module, which combines the votes received from the individual modules and chooses the action with the maximum vote, indicated with the dot in the figures. If the maximum is nonunique, one is chosen at random.

What is interesting to note in the figure is the behavior of the individual modules and their affect on the behavior of one another and the fusion module. In particular, frames 51, 61, 71, and 81 are interesting. In frame 51, it is seen that IDIFF and TM lose track of the object, while fusion tracks based on information provided by BLOB and SOB. In frame 71, TM resumes tracking, and in frame 81, IDIFF resumes tracking. Additionally, in subsequent frames, it seems that BLOB drifts away and later resumes tracking.

As evident from Fig. 6, if run independently IDIFF fails to track the object through the entire experiment. However, in Fig. 7, it is seen that IDIFF can resume tracking, when run in conjunction with the other modules. This can be explained as follows: when run independently, if a module loses track of the object, due to some artifact in the image that confuses the algorithm, it may not have the chance to correct itself, because losing the object eventually causes the object to leave the region of interest (fovea) or even the image. However, with fusion, this problem can be corrected do to an effect that we term **corrective reinforcement**, as other modules may continue to drive the object into the region of interest, where the object/motion is searched for, giving a failing module the opportunity to regain tracking. This will only work if the cause for failure of the modules is disjoint so that the modules never (or at least rarely) fail simultaneously.

VIII. Concluding Remarks

Biological vision systems are remarkably adept at providing useful, high-quality visual information in rich dynamic environments. These capabilities are, in part, a result of the inherent ability of such systems to effectively integrate data

[7] Changes in window size are due to scaling.

from a wide range of visual cues as well as compose modules in a timely, event-driven manner.

In this paper, we have explored various aspects related to the integration of homogeneous modules and their composition in the context of active vision. In particular, we have presented a unified approach for effectively providing both process integration and process composition. The approach combines a formalism for integrating homogeneous modules and a well-known process composition model, RS.

Our experiments in smooth pursuit have confirmed that the performance of several integrated motion analysis modules is dramatically better than the performance of any of the participant modules when run independently. In the course of our experiments, we discovered an interesting and powerful effect that we call **corrective reinforcement** which can lead to higher tracking success rates.

Future work includes experimenting with the composition and fusion of modules for performing other active vision activities, such as fixation and stabilization, a thorough analysis of **corrective reinforcement**, and an extension of our work to include nonhomogeneous modules.

REFERENCES

[1] M. Abdulghafour, A. Fellah, and M. A. Abidi, "Fuzzy logic-based data integration: Theory and applications," in *Proc. 1994 IEEE Int. Conf. Multisensor Fusion Integration Intell. Syst.*, pp. 151–160.

[2] Y. Aloimonos and D. Shulman, *Integration of Visual Modules: An Extension of the Marr Paradigm*. New York: Academic, 1989.

[3] Y. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *Int. J. Comput. Vision*, vol. 1, pp. 333–356, 1988.

[4] R. C. Arkin, "Integrating behavioral, perceptual and world knowledge in reactive navigation," *Robot. Automat. Syst.*, vol. 6, pp. 105–122, June 1990.

[5] R. Bajcsy, "Active perception vs. passive perception," in *Proc. 3rd IEEE Workshop Comput. Vision*, Bellaire, MI, 1985, pp. 55–59.

[6] D. H. Ballard, "Animate vision," *Artificial Intelligence*, vol. 48. Amsterdam, The Netherlands: Elsevier, 1991, pp. 57–86.

[7] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[8] D. M. Blough and G. F. Sullivan, "A comparison of voting strategies for fault-tolerant distributed systems," in *Proc. 9th Symp. Reliable Distributed Syst.*, Oct. 1990, pp. 136–145.

[9] K. J. Bradshaw, P. F. McLauchlan, I. D. Reid, and D. W. Murray, "Saccade and pursuit on an active head/eye platform," *Image Vision Comput.*, vol. 12, no. 3, pp. 155–163, Apr. 1994.

[10] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, pp. 14–23, Jan. 1986.

[11] C. Brown, "Prediction and cooperation in gaze control," *Biol. Cybern.*, vol. 63, no. 1, pp. 61–70, May 1990.

[12] R. H. S. Carpenter, *Movements of the Eyes*. London, U.K.: Pion, 1988.

[13] H. I. Christensen, J. Horstman, and T. Rasmussen, "A control theoretical approach to active vision," in *Proc. 2nd Asian Conf. Comput. Vision*, Singapore, Dec. 1995, pp. 364–369.

[14] J. J. Clark and N. J. Ferrier, "Modal control of an attentive vision system," in *Proc. 2nd Int. Conf. Comput. Vision*, Tampa, FL, Dec. 1988, pp. 514–523.

[15] D. Coombs and C. Brown, "Real-time smooth pursuit tracking for a moving binocular head," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Champaign, IL, June 1992, pp. 23–28.

[16] B. R. Donald, *Error Detection and Recovery in Robotics*. Berlin, Germany: Springer-Verlag, 1989.

[17] A. Elfes, "Sonar-based real-world mapping and navigation," in *Autonomous Robot Vehicles*. Berlin, Germany: Springer-Verlag, pp. 233–249, 1990.

[18] J. A. Fayman, E. Rivlin, and H. I. Christensen, "A system for active vision driven robotics," in *Proc. 1996 Int. Conf. Robot. Automat.*, Minneapolis, MN, pp. 1986–1992.

[19] J. A. Fayman, E. Rivlin, and D. Mosse, "Real-time active vision with fault-tolerance," in *Proc. 1996 Int. Conf. Pattern Recognit.*, Vienna, Austria.

[20] E. Gat and G. Dorais, "Robot navigation by conditional sequencing," in *Proc. IEEE Int. Conf. Robot. Automat.*, San Diego, CA, May 1994, pp. 1293–1299.

[21] J. Kosecka, R. Bajcsy, and H. I. Christensen, "Discrete event modeling of visually guided behaviors," *Int. J. Comput. Vision*, vol. 12, no. 3, pp. 295–316, 1995.

[22] S. Kristensen, "Sensor planning with Bayesian decision theory," in *Reasoning with Uncertainty in Robotics*. Amsterdam, The Netherlands: Univ. Amsterdam, Dec. 1995.

[23] E. P. Krotkov, *Active Computer Vision by Cooperative Focusing and Stereo*. Berlin, Germany: Springer-Verlag, 1989.

[24] D. M. Lyons, "Representing and analyzing action plans as networks of concurrent processes," *IEEE Trans. Robot. Automat.*, vol. 9, pp. 241–256, June 1993.

[25] D. M. Lyons and M. A. Arbib, "A formal model of computation for sensory-based robotics," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 280–293, June 1989.

[26] J. Manyikia and H. D. Whyte, *Data Fusion and Sensor Management—A Decentralized Informationtheoretic Approach*. New York: Ellis Horwood, 1994.

[27] I. Nourbakhsh *et al.*, DERVISH, "An office-navigating robot," *AI Mag.*, vol. 16, no. 2, pp. 53–60, 1995.

[28] K. Pahlavan, T. Uhlin, and J.-O. Eklundh, "Dynamic fixation," in *Proc. 4th Int. Conf. Comput. Vision*, Berlin, Germany, May 1993, pp. 412–419.

[29] S. Pankanti, A. K. Jain, and M. Tuceryan, "On integration of vision modules," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Seattle, WA, June 1994, pp. 316–322.

[30] B. Parhami, "Voting algorithms," *IEEE Trans. Rel.*, vol. 43, pp. 617–629, Dec. 1994.

[31] D. Payton *et al.*, "Do whatever works: A robust approach to fault-tolerant autonomous control," *J. Appl. Intell.*, vol. 3, pp. 226–250, 1992.

[32] J. Pearl, "Probabilistic reasoning in intelligent systems: Networks of plausible inference," in *The Morgan Kaufmann Series in Representation and Reasoning*. San Mateo, CA: Morgan Kaufmann, 1988.

[33] P. Pirjanian, "Reliable reaction," in *Proc. IEEE Int. Conf. Multisensor Fusion Integration Intell. Syst.*, Dec. 1996, pp. 158–165.

[34] A. Saffiotti, K. Konolige, and E. H. Ruspini, "A multivalued logic approach to integrating planning and control," *Artif. Intell.*, vol. 76, pp. 481–526, Mar. 1995.

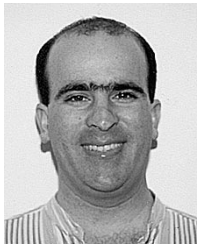
[35] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.

[36] R. G. Simmons, "Structured control for autonomous robots," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 34–43, Feb. 1994.

[37] M. J. Swain and M. A. Stricker, "Promising directions in active vision," *Int. J. Comput. Vision*, vol. 11, no. 2, pp. 109–126, 1993.

[38] W. Y. K. Wai and J. K. Tsotsos, "Directing attention to onset and offset of image events for eye-head movement control," in *Proc. IEEE Workshop Visual Behaviors*, Seattle, WA, June 1994, pp. 79–84.

[39] L. A. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338–353, 1965.



Jeffrey A. Fayman received the M.Sc. degree in computer science from San Diego State University, San Diego, CA, in 1990 and the D.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, in 1998.

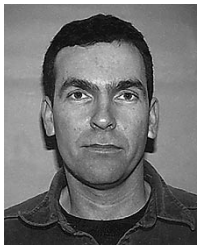
He is currently with Virtue 3D, Inc. His current research interests include active vision, functionality, fault-tolerance in robotic systems, and real-time systems.

Paolo Pirjanian (S'95–A'97) was born in Tehran, Iran, in 1968. He received the M.Sc. degree in computer engineering in 1994 and the Ph.D. degree in mobile robotics in 1998, both from Aalborg University, Aalborg East, Denmark. The main topics of his Ph.D. project were related to multiple objective action selection and behavior fusion using voting schemes.

He has been a Postdoctoral Research Fellow at the Laboratory of Image Analysis, Aalborg University, since December 1997, where he is involved in a nationwide intelligent multimedia project. His primary responsibilities include design, development, and integration of a distributed multimedia platform inhabited with intelligent artificial agents.

Henrik I. Christensen (M'87) received the M.Sc. and Ph.D. degrees in electrical engineering from Aalborg University, Aalborg East, Denmark, in 1987 and 1989, respectively.

He was a Research Associate from 1989 to 1992, working on control and system integration. In 1992, he was appointed Associate Professor at Aalborg University. In 1996, he was a Visiting Professor at the GRASP Laboratory, University of Pennsylvania, Philadelphia. He is presently Scientific Director of the Centre for Autonomous Systems, Royal Institute of Technology, Stockholm, Sweden. He has published more than 90 papers on vision and robotics. He has also edited three books on active vision, experimental environments, and integrated vision systems.



Ehud Rivlin (S'90–M'95) received the B.Sc. and M.Sc. degrees in computer science and the M.B.A. degree from Hebrew University, Jerusalem, and the Ph.D. degree from the University of Maryland, College Park.

He is an Assistant Professor in the Computer Science Department, Technion—Israel Institute of Technology, Haifa. His current research interests are in machine vision and robotic navigation.