Software Mode Changes for Continuous Motion Tracking*

Deepak Karuppiah[1], Patrick Deegan[2], Elizeth Araujo[2], Yunlei Yang[2], Gary Holness[2], Zhigang Zhu[1], Barbara Lerner[3], Roderic Grupen[2], and Edward Riseman[1]

[1] Dept. of Computer Science, Computer Vision Research Laboratory, University of Massachusetts, Amherst, MA 01003 USA
{zhu, riseman}@cs.umass.edu
http://vis-www.cs.umass.edu/projects/safer

[2] Dept. of Computer Science, Laboratory for Perceptual Robotics, University of Massachusetts, Amherst, MA 01003 USA
grupen@cs.umass.edu
http://www-robotics.cs.umass.edu/rob_safer

[3] Dept. of Computer Science, Lab. for Advanced Software Eng. Research, University of Massachusetts, Amherst, MA 01003 USA
lerner@cs.umass.edu
http://laser.cs.umass.edu/SAFER.htm

Abstract. Robot control in nonlinear and nonstationary run-time environments presents challenges to traditional software methodologies. In particular, robot systems in "open" domains can only be modeled probabilistically and must rely on run-time feedback to detect whether hardware/software configurations are adequate. Modifications must be effected while guaranteeing critical performance properties. Moreover, in multi-robot systems, there are typically many ways in which to compensate for inadequate performance. The computational complexity of high dimensional sensorimotor systems prohibits the use of many traditional centralized methodologies. We present an application in which a redundant sensor array, distributed spatially over an office-like environment can be used to track and localize a human being while reacting at run-time to various kinds of faults, including: hardware failure, inadequate sensor geometries, occlusion, and bandwidth limitations. Responding at run-time requires a combination of knowledge regarding the physical sensorimotor device, its use in coordinated sensing operations, and high-level process descriptions. We present a distributed control architecture in which run-time behavior is both preanalyzed and recovered empirically to inform local scheduling agents that commit resources autonomously subject to process control specifications. Preliminary examples of system performance are presented from the UMass Self-Adaptive Software (SAS) platform.

* This work was supported by AFRL/IFTD under F30602-97-2-0032 (SAFER), DARPA/ITO DABT63-99-1-0022 (SDR Multi-Robot), and NSF CDA-9703217 (Infrastructure).

P. Robertson, H. Shrobe, and R. Laddaga (Eds.): IWSAS 2000, LNCS 1936, pp. 161–180, 2000.
© Springer-Verlag Berlin Heidelberg 2000

1 Introduction

Much of current software development is based on the notion that one can correctly specify a system a priori. Such a specification must include all input data sets which is impossible, in general, for embedded sensorimotor applications. Self-adaptive software, however, modifies its behavior based on observed progress toward goals as the system state evolves at run-time [10]. Achieving self-adaptation requires knowledge of software structure as well as a means of measuring progress toward the goal. Current research in self-adaptive software draws from two traditions, namely control theoretic and planning.

The control theoretic approach to self-adaptive software treats software as a plant with associated controllability and observability issues[9]. Time-critical applications require the ability to act quickly without spending large amounts of time on deliberation. Such reflexive behavior is the domain of the control theoretic tradition. Control output decisions are based on observations of the plant's state. A goal is reached when the controller drives error between an input reference state (software specification) and the current system state to zero. Traditionally, control theoretic approaches to physical plants ignore issues of resource allocation and scheduling - robust control seeks to guarantee bounded performance degradation in the face of bounded parameter variation and adaptive control seeks to estimate the parameters of the controller and/or the plant in order to optimize closed-loop behavior.

Traditions in planning are also meaningful to self-adaptive software. Planning enumerates process state, possible actions, and desired goals in an effort to find a sequence of actions which advance it toward the goal. As actions are carried out, various artifacts can be predicted and resources can be scheduled to optimize performance. In the planning approach to self-adaptive software, software components are treated as resources [13]. A particular schedule may not be reasonable because it violates resource limits or causes the system to diverge from its goal. In this situation, a planning system may find a sequence of actions which advances (partially) toward a goal specification. The ability to make tradeoffs and perform such higher level reasoning is the domain of the planning tradition. Planning for large complex systems is both time and compute intensive.

How can a system react to time critical events while at the same time, make tradeoffs and perform high level reasoning? This is known as the planning versus reaction dilemma. In the UMass Self-Adaptive Software (SAS) research project, we take an approach which combines the control theoretic and planning traditions where appropriate to manage complex self-adaptive software systems.

High-level deliberation and low-level reactivity are valuable in the control of autonomous and self-adaptive systems. A successful implementation of such a hybrid architecture would permit the system to make use of prior knowledge when appropriate and to respond quickly to run-time data. The central open question appears to be deciding how reacting and deliberating should interact in a constructive fashion. We have adopted a perspective in which the control hierarchy is adaptive at every level. Low-level control processes parameterized by resources interact with the domain continuously and recover run-time context

observable by the working set of control. This kind of context feedback permits a high-level process planner to re-deploy resources. Over time, robust plans for interacting with specific problem domains are compiled into comprehensive reactive policies. State descriptions evolve to express likely run-time context at the highest levels and reactive policies adapt to handle run-time contingencies at the lowest levels [14].

We are concentrating on how resources, distributed in a non-uniform manner over multiple robot platforms can be coordinated to achieve mission objectives. Our approach relies on technologies that produce flexibility, resourcefulness, high performance, and fault tolerance. Specifically, we are interested in (1) how cross-modal sensory front-ends can be designed to provide mission-specific percepts, (2) how perceptual behavior can incorporate sensory information derived from two or more robotic platforms carrying different sensors and feature extraction algorithms, and (3) how team resources can be organized effectively to achieve multiple simultaneous objectives.

A family of resource scheduling policies, called Behavior Programs (B-Pgms), is downloaded into members of a working group of robots as part of the configuration process. Each B-Pgm contains a set of (previously evaluated) contingency plans with which to respond to a variety of likely run-time contexts and is responsible for orchestrating the run-time behavior of the system in response to percepts gathered on-line. If required, run-time contexts may be handled by making use of contingency plans in the B-Pgm, or by re-deploying resources at the process planning level.

The UMass hybrid SAS architecture is based on a set of primitive, closed-loop control processes. This framework allows hierarchical composition of the controllers into behavior programs (B-Pgms) for tracking, recognition, motion control, and for a more complex human tracking scenario. The multi-robot platform is designed to respond to multiple, simultaneous objectives and reasons about resources using a high-level process description and control procedure using the little-JIL process description language. Our goal is an ambitious, vertically integrated software environment in which run-time data sets drive the organization of behavior and contribute to the management of large and comprehensive software systems. This document describes the very first experiments employing this paradigm.

2 Sensory Primitives for Motion Tracking

A multi-objective system requires that the sensory algorithms are flexible to support adaptation and reconfigurable on-line to facilitate fault-tolerance. Our approach is designed to provide a set of sensor processing techniques that can fulfill both low-level and high-level objectives in an open environment. Cooperative interaction among members of the robot team requires the mission planner to be effective in utilizing system resources across team members, including robot platforms, sensors, computation, and communication. In particular, we are constructing robot behaviors across multiple coordinated platforms and sensors.

To achieve the desired robustness, the platform is configured with a variety of sensors and algorithms. Vision is the primary sensing modality, but it is complemented by inexpensive pyroelectric sensors, sonar, infrared proximity sensors, and (in the future) acoustic sensors. Multiple types of sensors are considered to be distributed across several robot platforms to allow flexibility in mission planning and resource scheduling in response to hardware and algorithm failures.

2.1 Panoramic Imaging

Effective combinations of transduction and image processing is essential for operating in an unpredictable environment and to rapidly focus attention on important activities in the environment. A limited field-of-view (as with standard optics) often causes the camera resource to be blocked when multiple targets are not close together and panning the camera to multiple targets takes time. We employ a camera with a panoramic lens[1] to simultaneously detect and track multiple moving objects in a full 360-degree view [3,12,16].

Figures 1 and 2 depict the processing steps involved in detecting and tracking multiple moving humans. Figure 1 shows a panoramic image from a stationary sensor. Four moving objects (people) were detected in real-time while moving in the scene in an unconstrained manner. A background image is generated automatically by tracking dynamic objects through multiple frames [5]. The number of frames needed to completely build the background model depends on the number of moving objects in the scene and their motion. The four moving objects are shown as an un-warped cylindrical image of Figure 2, which is a more natural panoramic representation for user interpretation. Each of the four people were extracted from the complex cluttered background and annotated with a bounding rectangle, a direction, and an estimated distance based on scale from the sensor. The system tracks each object through the image sequence as shown in Figure 2, even in the presence of overlap and occlusion between two people. The dynamic track is represented as an elliptical head and body for the last 30 frames of each object. The final position on the image plane is also illustrated in Figure 2. The human subjects reversed directions and occluded one another during this sequence. The vision algorithms can detect change in the environment, illumination, and sensor failure, while refreshing the background accordingly. The detection rate of the current implementation for tracking two objects is about 5Hz. The motion detection algorithm relies heavily on the accuracy of the background model at any given time in order to detect moving objects. Types of changes in the background can be broadly grouped into two categories: changes due to the illumination affecting pixel intensities at a fine scale; and changes of surfaces in the environment such as the movement of objects.

It is quite difficult to take care of both cases simultaneously because the first type requires a constant update while the second type requires a context-dependent update. The low-level background estimation procedure is quite simple. The constant update is done on those regions of the image that are not

[1] PAL-3802 system, manufactured by Optechnology Co.

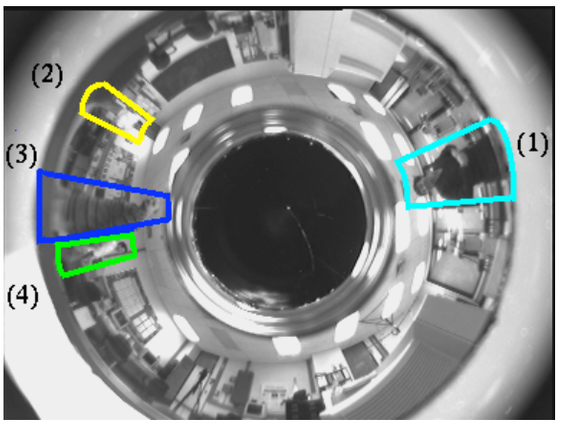

Fig. 1. Original panoramic image (768 x 576)

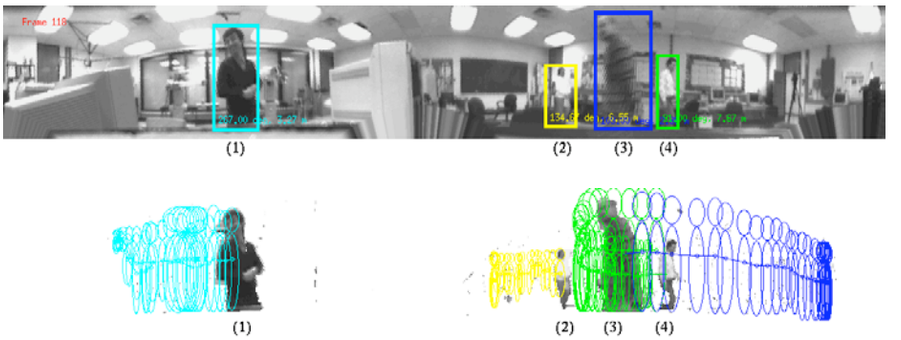

Fig. 2. Un-warped image, four moving people detected - top image; and track through image sequence for the last 32 frames - bottom image.

classified as a moving object by the motion detection algorithm. We track each region and keep a history of velocity for each as well. When the velocity falls below a threshold and remains so for a period of time, it becomes a suitable candidate for part of the background. The assumption is made that humans will be not be still for a long period of time. Therefore, they do not become part of the background. Similarly, only when the velocity of an object exceeds a threshold, is it classified as a possible human subject. This helps to avoid detecting some objects that should remain part of background but are not completely stationary, like the motion of tree branches, or the flicker of a computer monitor.

The adaptive background update improved the performance of the panoramic sensors considerably. The above adaptation only provides a low-level mechanism to handle the problem of maintaining an accurate background model. A more elegant way would be to use the context as inferred by the reasoning at higher

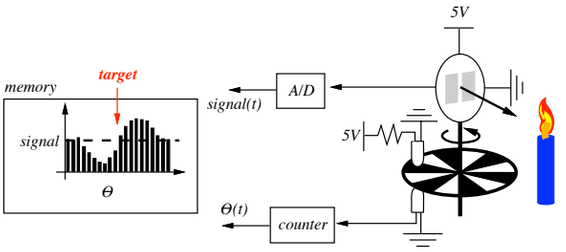

Fig. 3. Pyroelectric sensor.

levels of knowledge-based planning where all resources available might be employed. For example, an unconscious human subject will be still, so the low level will include them in the background. However, using the pyroelectric sensor, we might guess where the human is, particularly if the previous motion of the subject had been detected. This information could be passed to the vision sensors to update the background accordingly.

2.2 Pyroelectric Sensor

The pyroelectric sensor[2] is a Lithium Tantalate pyroelectric parallel opposed dual element high gain detector with complete integral analog signal processing. The detector is tuned to thermal radiation in the range that is normally emitted by humans. Since the pyroelectric detector itself only responds to changes in heat, the detector must be scanned. As shown in Figure 3, a thermal target is identified as a zero crossing in the sensor's data stream. We have implemented such a sensor on a scanning servo motor with two control modes; the sensor may saccade to a region of space designated by another sensor or pair of sensors, and it can track the thermal signature (even when the subject is still) by oscillating around the target heading. The result is a sensor that responds quite precisely to human body temperature but with a rather poor lateral bandwidth. This is due primarily to the scanning required to measure a zero crossing. To use this sensor appropriately, it must be applied only when the predicted lateral bandwidth of the subject is relatively small.

2.3 Stereo Head System

The stereo head platform[3] is a high-performance binocular camera platform with two independent vergence axes. As shown in Figure 10, it has four mechanical degrees of freedom and each lens has three optical degrees of freedom.

[2] Model 442-3 IR-EYE Integrated Sensor, manufactured by Eltec Instruments, Daytona Beach, FL

[3] BiSight System, manufactured by HelpMate Robotics, Inc., Danbury, CT

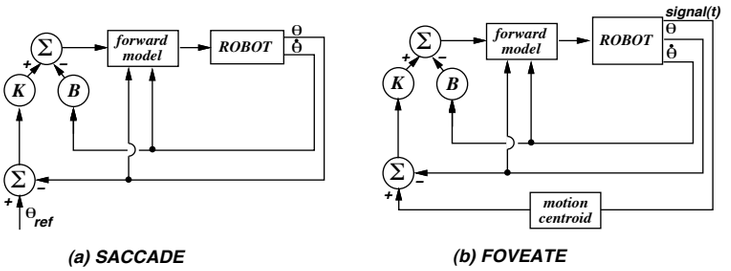

Fig. 4. Closed-Loop Primitives for Controlling Attention.

There are several state-of-the-art tracking algorithms in the literature [1,11, 2]. Our tracking algorithm uses one of the cameras as an active eye and the other as an passive eye. The active eye detects subsampled pixels of greatest change in intensity between two consecutive frames. The passive eye correlates multi-resolution fovea with the frame from the active eye. The stereo head is then servoed to bring the pixel of greatest change into the fovea of the active eye. Subsequently, the passive eye is verged to point its fovea to the same world feature as the fovea of the active eye, extracting the spatial location of the object.

The accuracy of the spatial location of the object is dependent on its distance from the stereo head system. This algorithm can only track single moving objects.

2.4 SACCADE-FOVEATE B-Pgm for Recovering Heading

The most primitive software process in this approach is an asymptotically stable closed-loop *controller* [4,7]. Controllers suppress local perturbations by virtue of their closed-loop structure. Some variations in the context of a control task are simply suppressed by the action of the controller. Controllers also provide a basis for abstraction. Instead of dealing with a continuous state space, a behavioral scheme need only worry about control activation and convergence events. When a control objective is met, a predicate is asserted in an abstract model of the system behavior. The pattern of boolean predicates over a working set of controllers constitutes a functional state description in which policies can be constructed. The "state" of the system is a vector of such functional predicates, each element of which asserts convergence for some control law and resource combination. The state vector also, therefore, represents the set of discrete subgoals available to a robot given these native control laws and resources.

Two closed-loop primitives are employed for motion tracking (see Figure 4). The first, *saccade*, accepts a reference heading in space and directs the sensor's field-of-view to that heading. The second, *foveate*, is similar except that it accepts heading references determined by a sensor signal. For example, the pyroelectric sensor scans a small region centered on the current gaze and identifies the *zero crossing* in the sensor output. The heading to the zero crossing is used as the

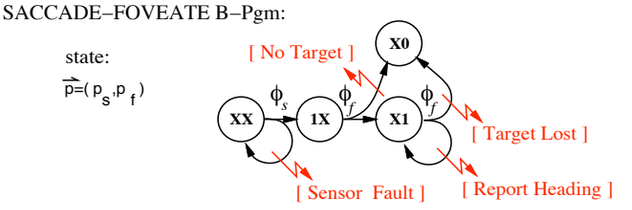

Fig. 5. Behavior Program for Detecting and Measuring the Heading to a Motion Cue.

reference heading to control the sensor's gaze. Within bandwidth limitations, the result is that the pyroelectric sensor tracks the thermal source.

Localizing and tracking the motion of objects in the world is an important, reusable behavior that can be realized in a number of different ways using a variety of different sensors. Each sensor in a stereo pair recovers the heading to a feature in the environment. When the imaging geometry of the pair is suitable, the sensors can, in principle, be used to triangulate the spatial location of the feature. Moreover, the control process for each sensor can be completely independent of the other sensor processes. We have hand-crafted a B-Pgm for accomplishing this task that is parametric in sensory resources. This B-Pgm is illustrated in Figure 5 - it represents a family of run-time hardware configurations for estimating the location of moving objects in space.

The state in the nodes of Figure 5 is the convergence status of the saccade controller, ϕ_s, and the foveate controller, ϕ_f. That is, if ϕ_s is converged and ϕ_f is not, then the state of the saccade-foveate process is 10. An X in the state representation represents a "don't care" or "don't know" condition.

Given R, the set of sensor resources, the saccade-foveate B-Pgm (or template) begins by directing a sensor $r_1 \in R$ to saccade to an interesting region of space. If this process fails for some reason, it is presumably an error in the motor component for the sensor and it reports a fault. If no hardware fault is detected and the sensor achieves state $1X$, then an independent, periodic, closed-loop process ϕ_f is engaged whose goal it is to bring the centroid of the local motion cue to the center of sensor r_1's field of view. If no motion cue is detected, then a report of *"no target"* is generated. If a target motion cue is detected and foveated, then the sensor achieves state $X1$ where the target is actively tracked. Consequently, sensor r_1 is likely no longer at the position specified by the original saccade. As long as subsequent foveation cycles preserve this state, a heading to the motion cue is reported. If, however, the sensor state becomes $X0$, then the target may be moving too quickly and a *"target lost"* report is generated. When two sensors are simultaneously in state $X1$, then the pair of active B-Pgms are reporting sufficient information for estimating the spatial location of this motion cue by triangulation. Each unique resource allocation $r_1, r_2 \in R$ produces a hypothesis of varying quality depending on the context of the localization query.

This policy does not rely on the physical quantity that is transduced; optical flow, thermal radiation, acoustic, etc. It assumes, however, that the two partici-

pating observers are foveated on the same entity in the world. While incorrect correspondence can lead to anomalous results, cross-modality can be used to advantage. For example, if the location is computed from consistent visual motion and pyroelectric information, then we may detect *"warm-moving"* bodies. Such a strategy may be attractive when detecting and localizing human beings as opposed to other types of moving objects.

2.5 "Virtual" Stereo Pairs

Any fixed-baseline stereo vision system has limited depth resolution due to the imaging geometry, whereas a system that combines multiple views from many stationary or movable platforms allows a policy to take advantage of the current context and goals in selecting viewpoints. A *"virtual stereo"* policy is a policy that engages different sensor pairs as the target moves through ill-conditioned sensor geometries. Although this policy is more flexible than a fixed pair, this approach requires dynamic sensor (re)calibration and accuracy in the depth of a target is limited by the quality of calibration. The virtual stereo strategy may be particularly effective with a pair of mobile panoramic sensors because they have the potential of always seeing each other and estimating calibration parameters[16]. Once calibrated, they can view the environment to estimate the 3D information of moving targets by triangulation, and maintain their calibration during movement by tracking each other. Suppose we have two panoramic cameras with the same parameters. Both of them are subject to planar motion on the floor and are of same heights from the floor. If they can see each other and at the same time see a target T, then we can compute the bearing and distance of the target after a dynamic calibration of the two cameras. Suppose that O_1 and O_2 are the viewpoints of the two cameras and they can be viewed by each other (as $M_2 1$ and $M_1 2$). B is the baseline (i.e. distance $O_1 O_2$) between them. The projection of a point T is presented by T_1 and T_2 in the two cameras. Then a triangulation $O_1 O_2 T$ can be formed (Fig. 7) so that the distances from the two cameras to the target can be calculated as

$$D_1 = B\frac{\sin\phi_2}{\sin\phi_0} = B\frac{\sin\phi_2}{\sin(\phi_1+\phi_2)} \quad , \qquad D_2 = B\frac{\sin\phi_1}{\sin\phi_0} = B\frac{\sin\phi_1}{\sin(\phi_1+\phi_2)} \tag{1}$$

By defining an arbitrary starting orientation for each cylindrical image, angles ϕ_1, ϕ_2 (and ϕ_0) can be calculated from the following four bearing angles: θ_1 and θ_2, the bearings of the target in image 1 and image 2 respectively, β_{12} and β_{21}, the bearing angles of camera 1 in image 2 , and camera 2 in image 1 respectively. In order to estimate the distance of a target, we need first to estimate the baseline and the orientation angles of the two panoramic cameras by a dynamic calibration procedure. Several practical approaches have been proposed for this purpose [14]. The basic idea is to make the detection and calculation robust and simple. One of the approaches is to design the body of each robot as a cylinder with some vivid colors that can be easily seen and extracted in the image of the other robot's camera. The estimated triangulation error can be computed

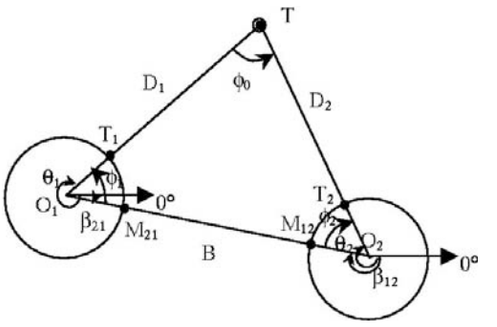

Fig. 6. Panoramic stereo geometry

by partial differentials of 1 as

$$\partial D_1 = \frac{D_1}{B}\partial B + D_1 \left|\cot(\phi_1 + \phi_2)\right| \partial\phi_1 + \frac{D_2}{\sin(\phi_1 + \phi_2)}\partial\phi_2 \tag{2}$$

where ∂B is the error in computing the baseline B, and $\partial\phi_1$ and $\partial\phi_2$ are the errors in estimating the angles ϕ_1 and ϕ_2 from the two panoramic images. Basically, the distance error is determined by both the error of the baseline and the errors of the angles in the triangulation approach. Given the dynamic calibration method, we can use Equation 2 to find the error bound of the distance estimation in any triangulation configuration, and further to find an optimal configuration of the virtual stereo with minimum estimating error.

We have developed the algorithms for mutual calibration and 3D localization of motions using a pair of panoramic vision systems each running the saccade-foveate B-Pgm. The first implementation has been carried out by cooperation between two stationary cameras. Figure 7 shows a stereo image pair from two panoramic sensors.

2.6 Peripheral and Foveal Vision Integration

The human eye has a wide-angle, low resolution field in its peripheral view, and a high resolution narrow field in its foveal view, a combination that works cooperatively in a highly robust manner. We can find a moving object within the peripheral view and then start a tracking behavior by peripheral-foveal cooperation. The key point here is the natural cooperation of peripheral and foveal vision as a real-time behavior operating within a common coordinate system.

As we consider a computer implementation of this behavior, we note differences with human capability. Humans must rotate the head so that the peripheral system covers the moving object in its field of view. Furthermore, multiple objects in very different directions cannot be tracked simultaneously. The panoramic-panoramic sensor pair (or any other pair applicable under the runtime context) can provide the spatial reference for a saccade-foveate B-Pgm on a

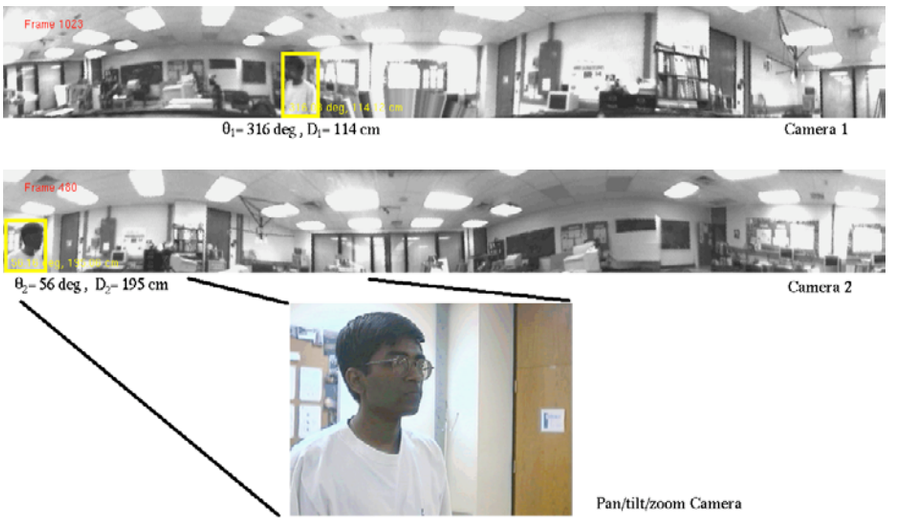

Fig. 7. 3D localization by the panoramic stereo system - top two images; and a closeup image of the Human Subject (the first author) localized - bottom image.

standard zoom camera mounted on a small pan/tilt platform. The pan/tilt/zoom imaging system may then undergo a saccade to the interesting motion cue. From here it can foveate on the cue and zoom if necessary for detailed processing.

High resolution color images obtained from the pan/tilt/zoom camera can be used to determine the identity of the object of interest. In particular, a challenging problem is to separate and track individuals in a group (or even a crowd). Using contour extraction algorithms based on motion cues, the pixels that correspond to the object can be extracted from the background.

We have successfully set up a peripheral and fovea vision system, and implemented a cooperative algorithm for processing moving objects. The system detects any moving object in the view of the panoramic camera, and tracks and identifies it through the zoom camera. Figure 7 illustrates the image resulting from such a process where the spatial reference to a motion cue is provided by the panoramic-panoramic image pair. The suspicious character in this panoramic image pair has been scrutinized successfully using the pan/tilt/zoom camera.

3 The Containment Unit

B-Pgms can be used to coordinate the behavior of a fixed set of resources. In [6], we show how to build B-Pgms automatically using reinforcement learning that approach optimal policies for a fixed resource allocation. The hierarchical generalization of a B-Pgm is the Containment Unit (CU). The CU is an active entity designed to represent a family of optimal contingency plans (B-Pgms)

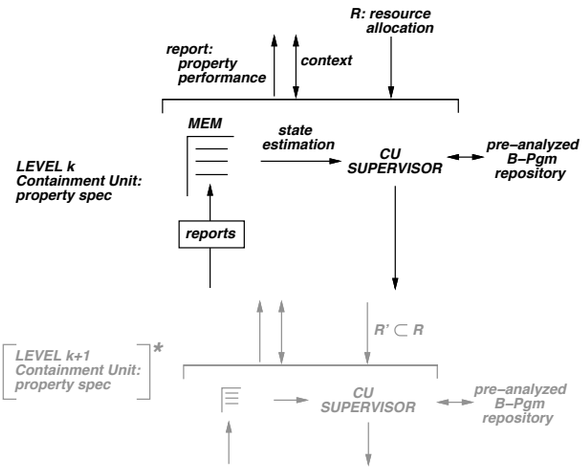

Fig. 8. The Structure of a Containment Unit.

parameterized by resource commitments. Its objective is to "contain" faults. A fault is generally construed to be any functional violation of the behavior associated with the CU: real-time constraints, liveness of constituent hardware, or performance constraints. If a sensor fails, it is the role of the CU to select an alternative B-Pgm to provide the same *type* of information and to inform the process that activated the CU of the impact on the expected performance.

The structure of a CU is presented schematically in Figure 8. It accepts reports from subordinate CUs and estimates the state necessary to make local resource allocation decisions. Multiple instances of a CU may be active concurrently, each with a resource specification that determines the range of variation permitted locally in the strategy for executing the CU directive. Global resource constraints are achieved by limiting the range of autonomy each CU enjoys through careful specification of its proprietary resources. In general, specific B-Pgms may be applicable only in prescribed contexts. For example, adequate illumination may be necessary to employ B-Pgms that use vision sensors, or limited target velocity may be required in order to track with a scanning pyroelectric sensor. These "contexts" can be loaded when a CU is activated and then verified at run-time, or they may be recovered by monitoring the active B-Pgm's performance. The CU determines how to reconfigure lower-level CUs and/or whether to report recovered context to high-levels.

3.1 CU Supervisor: Domain-Independent Behavioral Expertise

Some aspects of a particular B-Pgm's performance *in situ* are determined entirely by attributes of the participating resources. The most obvious example of critical local state is the *liveness* of the participating hardware. Other locally determined attributes can also be important with respect to overall performance.

Consider a pair of vision sensors performing as a virtual stereo pair to localize a moving target. Localization will be poor if the uncertainty in the position of the participating sensors is large or the saccade-foveate B-Pgm may behave poorly if the target approaches a collinear spatial relationship with the sensor pair. These conditions are entirely determined by examining attributes of the sensors (their relative spatial arrangement) and the result of the B-Pgm coordinating them (the target position). Circumstances such as these are completely determined in the local state of the CU supervisor without higher-level deliberation. We will develop an example of the CU supervisor in Section 5.

The memory structure illustrated in Figure 8 records the reported results of all subordinate CUs, estimates state information required to make local resource allocation decisions, and supports the interpretation and reporting of context from the CU. Task specific information such as target location and current fault conditions are stored. The structure is maintained by a communication protocol over Internet sockets between the active B-Pgms and the CU. If resources reside on disparate architectures and operating systems, the memory structure will also provide the CU with a common communication interface to all subsystems and forms the basis for the High Level Interface.

3.2 Context: Domain-Dependent Behavioral Models

Open environments present data sets to sensorimotor processes that cannot be predicted at process configuration time in general and must be observed at run-time. When peculiar or unexpected environments cause the behavior of the system to deviate from expectations, a higher-level reconfiguration must modify system performance while remaining within specifications. If a specific B-Pgm proves to be inadequate in a particular run-time context, the context is passed upward in the control hierarchy to a process manager which may choose to reallocate resources. Over time, some of these reconfiguration decisions that depend strongly on controllable system components might be compiled into appropriate CU supervisors. However, other contexts will be determined by the run-time environment, and the deliberative process planner must model these dependencies at a higher level. We are studying mechanisms where the process description can incrementally model these environmentally determined contexts and manage resources so as to recover critical run-time, environmentally determined contexts in the course of the mission.

4 The Little-JIL Agent Coordination Language

Little-JIL provides rich and rigorous semantics for the precise specification of processes that coordinate multiple agents [15,8]. In the context of our SAS platform, the agents consist of individual sensors, individual robots, or combinations thereof. Little-JIL provides constructs for proactive and reactive control, exception handling, and resource management.

A Little-JIL process defines a high-level plan to coordinate agents to act as a team. A process is constructed of steps that are hierarchically decomposed into finer-grained substeps. The steps and substeps are connected with dataflow and control flow edges. Each step has a declaration identifying resources needed to carry out that step and allows reasoning over interactions between resource specifications including the sensors, computational platforms, and communication hardware that constitute a team of robots.

A process description typically specifies parts of the coordination policy precisely while deferring some choices until run-time. In this way, a step may be implemented in several ways using different resources. Which choice is most appropriate depends on resource availability, timeliness and performance constraints, and run-time context. These high-level decisions require reasoning across the collection of robots as the task unfolds. This approach is particularly useful for exception handling - a certain amount of reaction can be handled within the CUs by dynamically selecting the appropriate B-Pgms. Some situations, however, require higher level support. For example, consider a process intended to track multiple people. Such a process might designate one sensor to watching for new motion cues entering at a door and allocate the balance of the resources to track targets already in the room. If a new motion cue occurs, the process reacts by reassigning resources. The actual selection of resources and CUs and thus the actual instantiation of the system is made by the integrated capability of robot planning and scheduling technologies whose description is outside the scope of this paper.

The Little-JIL process control language as discussed above, provides a powerful means of exploiting knowledge to structure planning and learning by focusing policy formation on a small set of legal programs. Moreover, at lower levels, new and enhanced processes are constructed. The objective is to constantly optimize and generalize the active B-Pgm during training tasks, and to return it at the end of the task better than we found it. These B-Pgms actually consist of many coordinated primitive controllers but are thought of as discrete abstract actions. Subsequent plans and learning processes can exploit this abstraction.

Figure 9 shows a sample Little-JIL process that uses sensors to track multiple humans. We assume that this process specification is in the context of a partial model of the run-time environment. The root step of the process is Track Humans. This step is decomposed into two steps that run concurrently (denoted by the parallel lines). One step is to track a human while the other step is to watch the door. The Watch Door step requires use of the panoramic camera.

Track Human is a choice step. Dynamically, the system will decide which of the three substeps to use. This decision is based on the resources available, what time constraints there are on the tracking, and contextual issues, such as whether there is good lighting or whether the target is moving quickly. One might easily imagine many more than three choices here. Each choice requires one or more resources and has some expected performance. The scheduler and runtime system use knowledge about the context to assist in making the decision.

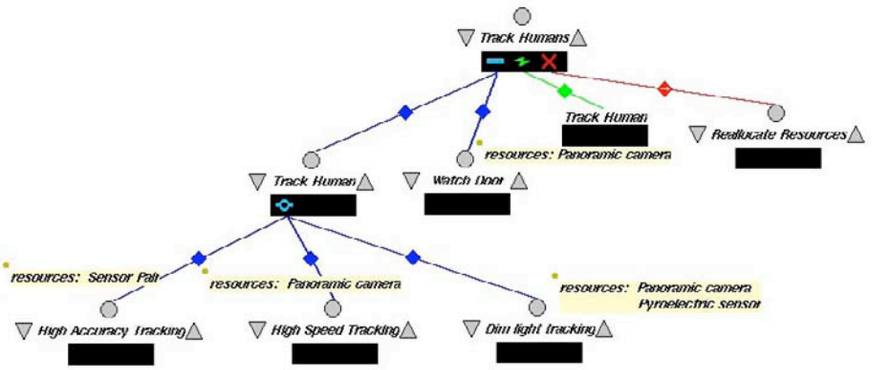

Fig. 9. Sample Little-JIL Process Description for Tracking a Human Subject.

If another human enters the room, this results in an event that is handled by a second Track Human step. This is simply a reference to the original track human step and will result in a new instance of Track Human starting with a new set of resources. This results in an exception, causing some replanning and reallocation of resources to occur. Other exceptions can be used to adapt locally (within the CU) during execution. For example, if there had been normal lighting and the lights were turned off, we would expect an exception within the currently active containment units that employ vision sensors.

5 Self-Adaptive Software (SAS) Experimental Platform

In our experimental platform, we have implemented three types of motion detectors that are deployed at fixed and known positions in an indoor office-like environment. The platform consists of an articulated stereo vision system, and scanning pyroelectric sensor, and two panoramic vision sensors, as shown in Figure 10. In each instance of the saccade-foveate B-Pgm observations are collected from sensor pairs that are sufficient to determine a spatial location of the moving feature in the field of view. This family of functionally equivalent programs produces a spatial estimate of a motion cue with varying quality that could serve as a spatial position reference to a subsequent sensory or motor control task. Indeed, combinations of these strategies are themselves B-Pgms with reserved resources for corroboration or for fault tolerance. Which of these to use in a particular context is dependent on the task, the resources available, and the expected performance based on accumulated experience.

5.1 Designing the CU Supervisor for Tracking Human Subjects

The CU Supervisor determines which B-Pgm (sensor pair) is recommended for triangulation and tracking given the current state of the process. In our demon-

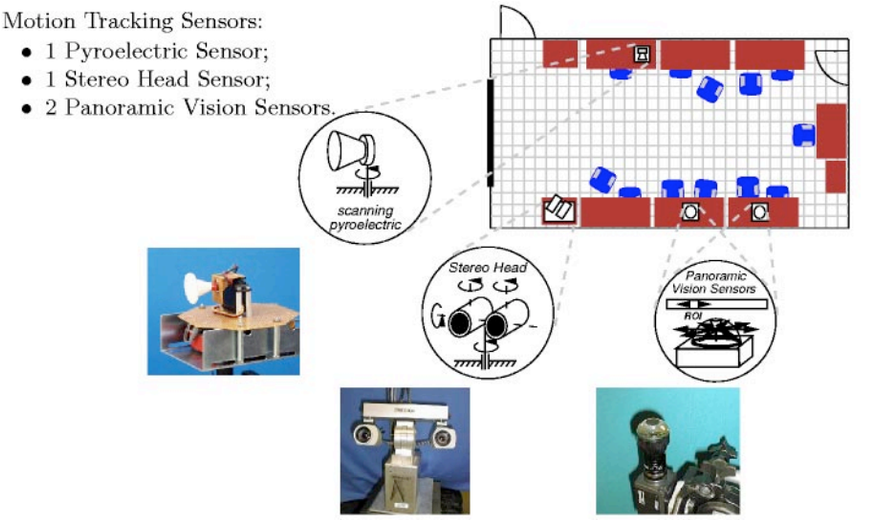

Fig. 10. The "Smart Room" - Motion Tracking Platform.

stration, there are six unique pairs of sensors available. A boolean state predicate describes the "liveness" of each pair. For a given pair, if both sensors are functioning and they are not in a collinear configuration with the target subject, the corresponding predicate is set to 1, otherwise it is set to 0. This is the role of the state estimation component of Figure 8. Given a pattern in the "liveness" state vector, the CU supervisor always chooses the pair of sensors with the highest value with respect to the process' objective function.

We have hand-crafted a Human Tracking CU supervisor for engaging sensor pairs that deploys resources in the following priority-based hierarchy:

- Panoramic - stereo head (camera 1);
- Panoramic - stereo head (camera 2);
- Stereo-head (camera 1 and 2);
- Panoramic - pyroelectric;
- Stereo-head (camera 1) - pyroelectric;
- Stereo-head (camera 2) - pyroelectric.

Each resource allocation in this hierarchy, in turn, instantiates two concurrent containment units for tracking motion with a single sensor. These subordinate CUs execute the saccade-foveate B-Pgm described earlier and report to the track human CU. Each CU in this hierarchical control process has the authority to manage the resources reserved for them.

5.2 Experimental Results

The Human Tracking CU supervisor has been implemented to control the various sensors in order to track a single moving person seamlessly through failure modes captured in the liveness assertion. Some preliminary results are presented below.

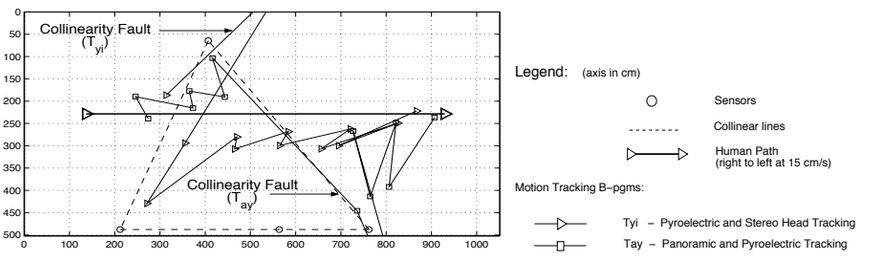

Fig. 11. Motion Tracking for the Pyroelectric-Stereo head and Pyroelectric-Panoramic sensor pairs.

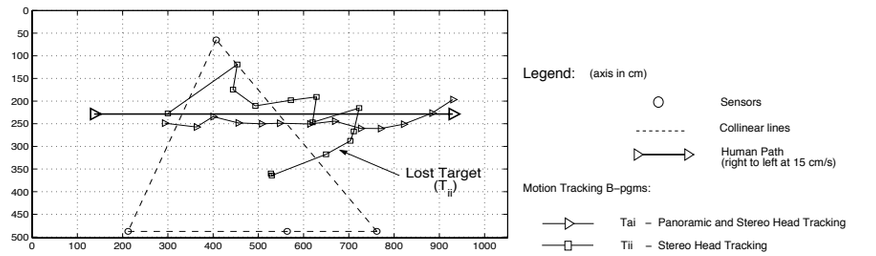

Fig. 12. Motion Tracking for the Panoramic-Stereo head and Stereo Head sensor pairs.

Accuracy and Repeatability Experiments. To design any CU supervisor that depends on the coordinated activity of multiple sensors, it is necessary to model the performance of the individual sensors. We conducted a series of experiments to determine the accuracy and repeatability of the sensors. At known spatial locations, a motion cue was generated and observed from the different sensors.

It was observed that the panoramic sensors were both accurate and repeatable, the stereo head is accurate but not repeatable, and the pyroelectric sensor was repeatable but not accurate. The data was also used to examine the quality of triangulation on the motion cue by different sensor pairs. As expected the quality degraded as the motion cue approached the line joining a sensor pair or a collinear configuration. Because such a configuration is not desirable we call this a collinearity fault.

Tracking a Human Subject. The next set of experiments evaluated the task of tracking a single moving person using combinations of the four sensors. The results are shown in Figures 11, 12, 14 and 13. Figure 11 shows the tracks of the Panoramic-Pyroelectric pair (T_{ay}) and the Pyroelectric-Stereo head pair (T_{yi}). As the motion track crosses collinear sensor geometries, the performance degrades as expected.

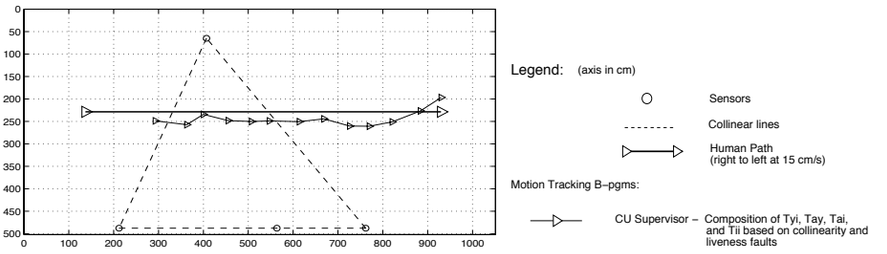

Fig. 13. Motion Tracking Performance during Mode Changes in the Motion Tracking CU supervisor.

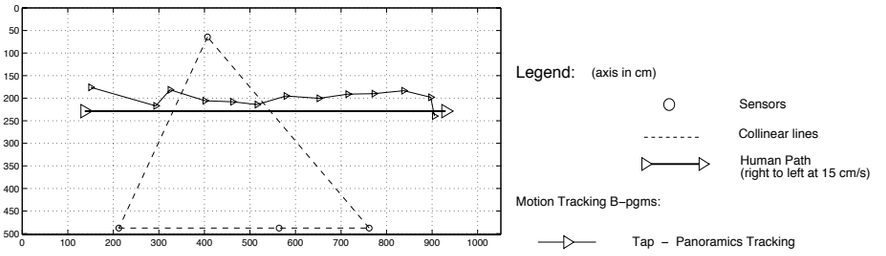

Fig. 14. Motion Tracking for the Panoramic-Panoramic sensor pair.

Figure 12 shows the tracks of Panoramic-Stereo head pair (T_{ai}) and Stereo head alone (T_{ii}). Target tracking using stereo head alone can be quite bad due to its small stereo baseline and mechanical properties [2].

Our next demonstration shows the performance of the CU supervisor. The CU supervisor is designed to address run-time contexts (e.g. tracking precision, sensor liveness faults, and collinearity faults) by effecting software mode changes in response to feedback from the sensors. Figure 13 shows that the Track Human CU supervisor was effective in handling these run-time contexts.

Figure 14 shows preliminary localization results using the Panoramic virtual stereo pair (T_{ap}). This sensor pair is highly reliable and is capable of accurate, high velocity tracking for large regions of the room because of its tracking precision and the complete field of view they provide. Our current CU supervisor does not include the Panoramic virtual stereo pair (T_{ap}). However, as shown in Figures 13 and 14, T_{ap} performs as well as the current multi-sensor CU under the conditions tested and so it will introduce a great deal of robustness when T_{ap} is integrated into the Track Human CU - since other sensors can fill in regions where T_{ap} performs badly or when other forms of sensor faults occur. In future we plan to include our Panoramic virtual stereo pair into the SAS platform. This addition will allow for multiple Human tracking at higher velocities.

These results indicate the potential of our hierarchical self adaptive software architecture in handling faults at both lower level (i.e. sensors) and higher level (i.e. context of the motion cue).

6 Summary, Conclusions, and Future Experimental Work

Multi-robot scenarios present significant technical challenges regarding sensing, planning, computing, and software methods and must support both reactivity and predictability. Ultimately, one of the most desirable characteristics of a multi-robot system is its ability to adapt to changes in the environment and to internal faults - in hardware components and in end-to-end performance specifications. Thus, reconfigurability is critical.

Our current work presents preliminary results towards the responsiveness to novel data sets and robustness that are critical to a multi-robot application. The CU supervisor for tracking a human subject was able to handle individual sensor faults gracefully as well as faults due to run-time context. Future experimental work is underway currently to demonstrate the ideas presented in this paper more thoroughly. Some of the dimensions along which we will enhance the experimental effort are described in the following sections.

Multiple Target Corroboration. When a single subject is tracked, excess resources can be allocated to enhance robustness. However, when multiple subjects are tracked, decisions must be made in order to allocate the right sets of resources to the right targets. For example, if a subject stops moving, we may be able to assign a single observer to it to verify that it remains stationary. When movement is detected, it can trigger another re-distribution of resources. We are developing process descriptions and resource scheduling algorithms that will reallocate resources in a manner that depends on target type.

Inter-Process Communication. In related work, we are developing an interprocess communication mechanism that guarantees that data will be shared between distributed processes within a specified time bound. In this application, it is less critical that communication events occur at precise times and more important that reports from multiple sensors happened at roughly the same time. Many techniques for relative process synchronization are being examined.

Hierarchical Process/Device Models. We already motivated a collinearity fault for pairs of sensors. We also intend to build models of each individual sensor's lateral bandwidth since it is this information that speaks directly to whether and how well a particular sensor can track a moving target. If context (e.g., $[\dot{x} \quad \dot{y}]$) recommends against using a slow sensor, it can unilaterally "take itself out of the game." On the other hand, if a rapidly moving subject changes to a slower or stationary target, relatively high-performance and expensive resources may be released and used more effectively elsewhere. We feel that models will naturally reside at many levels of abstraction and we intend to both build this information into the CU supervisors and acquire it empirically over time.

Self Calibration. Eventually, some of our sensors will be mobile and will take action prophylactically to avoid predictable faults. In order to do this, there must be sufficient resources to identify the new and changing sensor geometries. Resources previously used to track human subjects must be orchestrated to track spatially reconfigurable sensors.

References

1. D. Coombs and C. Brown. Real-time binocular smooth pursuit. *Int. J. of Computer Vision*, 11(2):147–164, 1993.
2. K. Daniilidis, C. Krauss, M. Hansen, and G. Sommer. Real time tracking of moving objects with an active camera. *J. of Real-time Imaging*, 4(1):3–20, Feb. 1998.
3. P. Greguss. *Panoramic Imaging Block for 3D space*, Jan. 1986. U.S. Pat. 4566763.
4. R. A. Grupen, M. Huber, J. A. Coelho Jr., and K. Souccar. A basis for distributed control of manipulation tasks. *IEEE Expert*, 10(2):9–14, Apr. 1995.
5. I. Haritaoglu, D. Harwood, and L. S. Davis. W4s: A real-time system for detection and tracking people in 2.5d. In *Proc. of the 5th European Conf. on Computer Vision*, Freiburg, Germany, Jun. 1998.
6. M. Huber and R. A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems*, 22(3-4):303–315, Dec. 1997.
7. M. Huber, W. S. MacDonald, and R. A. Grupen. A control basis for multilegged walking. In *Proc. of the Int. Conf. on Robotics and Automation*, volume 4, pages 2988–2993, Minneapolis, MN, Apr. 1996.
8. D. Jensen, Y. Dong, B. S. Lerner, E. K. McCall, L. J. Osterweil, S. M. Sutton Jr., and A. Wise. Coordinating agent activities in knowledge discovery processes. In *Proc. of the Int. Joint Conf. on Work Activities Coordination and Collaboration*, pages 137–146, San Francisco, CA, Feb. 1999.
9. M. M. Kokar, K. Baclawski, and Y. A. Eracar. Control theory based foundations of self controlling software. *IEEE Intelligent Systems*, 14(3):37–45, May 1999.
10. R. Laddaga. Creating robust software through self-adaptation. *IEEE Intelligent Systems*, 14(3):26–29, May 1999.
11. A. Maki, T. Uhlin, and J. Eklundh. Phase-based disparity estimation in binocular tracking. In K. Heia, K. A. Høogdra, and B. Braathen, editors, *Proc. of the 8th Scandinavian Conf. on Image Analysis*, pages 1145–1152, Tromsøo, Norway, May 1993. Norwegian Society for Image Processing and Pattern Recognition.
12. S. K. Nayar and S. Baker. Catadioptric image formation. In *Proc. of DARPA Image Understanding Workshop*, pages 1431–1437, May 1997.
13. P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf. An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems*, 14(3):54–62, May 1999.
14. J. Sztipanovits, G. Karsai, and T. Bapty. Self-adaptive software for signal processing: evolving systems in changing environments without growing pains. *Communications of the ACM*, 41(5):66–73, May 1998.
15. A. Wise, B. S. Lerner, E. K. McCall, L. J. Osterweil, and S. M. Sutton Jr. Specifying coordination in processes using little-JIL. Tech. Report TR 99-71, Univ. of Massachusetts, Amherst, Dept. of Comp. Sci., 1999.
16. Z. Zhu, E. M. Riseman, and A. R. Hanson. Geometrical modeling and real-time vision applications of panoramic annular lens (pal) camera systems. Tech. Report TR 99-11, Univ. of Massachusetts, Amherst, Dept. of Comp. Sci., Feb. 1999.