Chapter 2

TOWARDS AMBIENT INTELLIGENCE FOR THE DOMESTIC CARE OF THE ELDERLY

S.Bahadori,[1] A.Cesta,[2] L.Iocchi,[1] G.R.Leone,[1] D.Nardi,[1] F.Pecora,[2] R.Rasconi[2] and L.Scozzafava[1]

[1]*Dipartimento di Informatica e Sistemistica, University of Rome, La Sapienza, Italy*
{bahadori, iocchi, leone, nardi, scozzafava}@dis.uniroma1.it

[2]*Planning and Scheduling Team, Italian National Research Council, Italy*
{a.cesta, fpecora, rasconi}@istc.cnr.it

Keywords: Artificial Intelligence, Ambient Intelligence, intelligent sensors, situation monitoring, assistance, e-services.

1. Introduction

Today, modern societies are tackling an important problem, namely the progressive aging of the population. It is often necessary for elderly people to leave their homes to go live with their relatives or become guests of a health-care institution. This phenomenon has serious social and economic consequences. The long term goal of the research developed by the ROBOCARE project (see http://robocare.istc.cnr.it for details) is to contribute to raising the quality of life of elderly persons. In particular, we are pursuing the idea of developing support technology which can play a role in allowing vulnerable elderly people to lead an independent lifestyle in their own homes.

Research in Ambient Intelligence has begun to address some of the issues related to this problem. Nonetheless, most systems developed in this context are strongly sensor-centric, meaning that they are capable of recognizing simple emergency situations and firing alarms consequently. In this paper we aim at augmenting the scope of intervention of the overall system, by endowing it with more complex schemes of intervention. In particular, we are exploring the possible applications of intelligent systems in the domestic environment, studying how AI can be employed to realize cognitively-enhanced embedded

technology which is capable of *supporting* and *monitoring* the elderly person in his or her daily tasks. This implies the presence not only of "classical" sensory capabilities, rather it requires *integrating* such features with complex symbolic reasoning functionalities in a highly connected infrastructure.

In this context, our efforts are striving to integrate state-of-the-art technology in the fields of robotics, sensors, planning & scheduling and multi-agent systems in order to produce one, multi-functional entity to be deployed in a model of a domestic environment. The system is composed of a multitude of agents, each of which provides a set of *services.* The high level coordination of these agents is induced by an Active Supervision Framework (ASF), whose role is to provide effective mechanisms to invoke the agents' services. In this article, we show the two most significative components of the framework, which constitute the basic ingredients of an illustrative system, exposing the basic functionalities which are required for the entire, more complex, framework. First, we describe the *people localization and tracking agent,* a stereo camera which provides the 'still-image' and 'streaming video' services, by means of which it is capable of determining the 3D position of a person in the environment and tracking the person's position when he or she moves. Second, the *execution monitoring agent,* a CSP-based scheduling component which follows the execution of a predefined set of daily activities of the assisted elderly person, providing consistency checking services by reacting to contingencies and foreseeing inconsistencies in the schedule.

These two ingredients constitute an initial prototypical system which is currently deployed in a testbed domestic environment. The integration of these two components is achieved by means of *e-service oriented middleware.* This infrastructure allows all the agents in the system to provide their functionalities in the form of services. Under this paradigm, each agent provides a service which other agents can request, thus implementing basic mechanisms for data exchange and cooperation.

This paper is organized as follows: after describing in greater detail the desiderata for the integrated system, section 2 briefly outlines some basic implementation choices of the e-service infrastructure. Section 3 then illustrates the people localization and tracking service, detailing the basic functioning mechanisms of the stereo-camera based sensor which is currently tested in the domestic environment. Section 4 is dedicated to the description of the execution monitoring service, which is based on a CSP representation of activity work-flow. Following the description of the two basic components, section 5 shows by means of an example how the two components work together in a real-world instance, namely the ROBOCARE Domestic Environment. Finally, the last section gives some conclusions and outlines future work.

2. An Integrated Supervision System

The intelligent system developed for the ROBOCARE domestic environment is composed of a multiplicity of hardware and software agents. All agents can

be thought of as components of a single complex system, whose aim is to create an intelligent and supporting environment. In particular, the target users of this application are elderly people whose every-day independence may be improved by such a monitoring infrastructure.

Currently, the embedded technology in the ROBOCARE environment provides monitoring-specific services, such as the 'still-image' and 'people tracking' services, provided by a fixed camera, the 'find object' service provided by a mobile robot, and the 'visualize' service exported by a Personal Data Assistant. Every agent is present in the system as a set of services, which the other agents can reserve and use. In this light, it is possible to abstract away the physical components, focusing on varying levels of detail by defining 'super-agents' as a collection of services provided by various physically different agents. For instance, the 'remote-visualization' service is implemented by the combination of the camera agent and the PDA. If one of the two devices is not available (i.e. they do not make their service available), then also the super-agent cannot offer its service, and the request must be put on hold. In this schema, when there are redundant services (for instance, a 'still-image' service exported by two mobile robots), the super-agent which exports the combined service is not bound to a particular physical device, rather it is embodied by the first available robot on a preference basis.

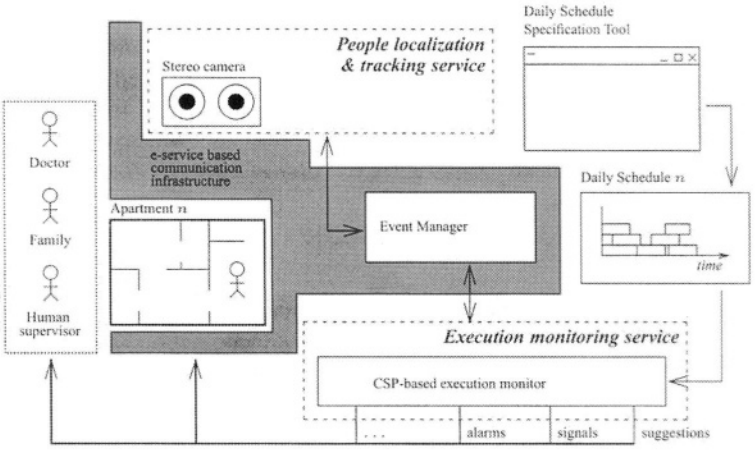

Figure 2.1. A schema of the current implementation of the *Active Supervision Framework.* By integrating the services provided by a stereo-camera and a CSP-based execution monitor, the ASF is capable of monitoring the daily activities of an assisted elderly person in his or her domestic environment.

Fig. 2.1 shows a system in which a stereo-camera is "coupled" with an execution monitoring module. Together, these two components provide basic

services for monitoring the daily schedule of an assisted elderly person in his or her apartment. On one hand, the stereo-camera observes the environment, and by means of its 3D localization capabilities, compiles information on the *current state* of the environment. This information can be processed by the CSP based execution monitoring module. By means of an internal representation of the assisted person's *nominal* schedule, this module attempts to recognize inconsistencies in the actual execution of the activities as it is performed by the assisted person. The loop is closed through the output of the execution monitoring service, which can deliver suggestions, activate alarms and other such signals. In the rest of this paper, we detail the basic functioning mechanisms of the two components.

It is important to notice at this point that one key issue in the implementation of this architecture is what we have called in fig. 2.1 the "event manager". The role of this component is to *arbitrate* the service requests of the various agents. As an arbiter, it is the responsibility of the event manager to orchestrate the flow of information to and from the services, performing load balancing tasks and maintaining global system consistency. The event manager processes all service requests, dispatching them to the appropriate agent, performing synchronization tasks and administering the use of the communication infrastructure which connects all the components of the system. In conclusion, the event manager is what allows the service-providing agents to function together in a highly cooperative fashion.

As a consequence of viewing each component of the system as an agent which provides and can request services, the communication infrastructure has been implemented following the *e-service* paradigm. We will briefly describe this infrastructure in the following subsection. We conclude this section by saying that the integrated system shown in fig. 2.1 constitutes a prototypical implementation of what we call an *Active Supervision Framework* (ASF). The ASF can be seen as a complex infrastructure which provides *monitoring capabilities* in the domestic environment of an assisted elderly person. It provides a coordination framework for the components of the system by implementing service publishing functionalities and coordinating the service-providing agents (by means of the event manager). For instance, if an assisted elderly person decides to perform an activity which contrasts with a particular medication requirement (e.g., no eating for two hours after having taken a particular medication), the system should recognize the inconsistency (by employing the services provided by the agents) and deduce a contingent plan to solve the situation (suggestions, alarms, and so on). Our effort to implement a complete ASF for a domestic environment starts here, with the integration of the two basic services we have mentioned above.

2.1 E-service Based Integration Schemata

Service oriented computing is a relatively new approach to the development of integrated systems. Service oriented architectures (SOA) have evolved over the last years to support high performance, scalability, reliability and availability. In basic terms a service is an application that can be accessed through a programmable interface. Different distributed computing protocols such as CORBA, DCOM and RMI have been adopted to access these services. Although they are very effective for building a specific application, the tight coupling used in these architectures limits the re-usability of individual components. Each of these protocols is constrained by vendor implementation, platform and development languages that severely limit interoperability. Moreover, none of these technologies operate over the Internet with the simple use of the HTTP protocol. This is a real obstacle due to security issues.

E-services represent the convergence between SOA and the web (hence they are often referred to as web-services). They are applications which offer particular functionalities in the form of services. Their functionalities are easily reusable (without knowing how the service is implemented) and they are accessed via the Internet protocol (HTTP) using universally accepted data formats (XML). These standard elements ensure easy integration of heterogeneous components. The e-service interface adds a layer of abstraction that makes the connection flexible and adaptable. Thus e-services have emerged as a powerful mechanism for integrating different systems and assets. This entails an obvious advantage for the ROBOCARE project, which is composed of many units which develop components using their own proprietary software tools. By adopting the e-service infrastructure, the developers of individual units have the only burden of complying with the e-service interface specifications.

There is also a long-term advantage that arises from the present direction of the market. The Internet is is not only a network of computers; there are many electronic devices (cell phones, web-TV, PDAs, etc.) that are capable of surfing the web. In the near future more and more domestic components (washing machines, fridges, ovens, etc.) will offer the possibility of being controlled via remote web access. Probably, they will publish their e-services to interact with other applications. This is very important in our case because these devices are part of the daily routine of a person. The possibility of having an open system which can easily integrate a new component built from a third party is a great challenge.

Although a great deal of work has been done by major IT companies as well as by the organizations concerned with standardization (W3C and OASIS), e-service technologies are still in their initial phase of development. Today we have standards for Data Formats (XML), Service Interaction (SOAP), Interface Description (WSDL) and Publishing and Retrieving (UDDI). In the context of this article it is not our goal to detail the technicalities of the implementation, rather we will describe the interface of the e-services and the dynamics of the

```
type class DeviceData
  id : integer
  type : integer
  name : string
end

type class AddressData
  IP : string
  port : integer
end

type class ImageData
  width : integer
  height : integer
  buf : array[MAXINT] of integer
end

eservice Stereo_Camera
  begin property
    identification : DeviceData
  end property
  begin message
    IN FrameRequest(info : AddressData; to : DeviceData)
    OUT SendFrame(frame : ImageData)
    IN StopRequest()
  end message
  begin state machine
    STATE_NUMBER := 3
    TRANSITION initial TO 1 : FrameRequest/SendFrame
    TRANSITION 1 TO 1 : FrameRequest/SendFrame
    TRANSITION 1 TO final : StopRequest/ϵ
  end state machine
end
```

Figure 2.2. The Stereo Camera e-service formal specification.

communication between services conceptually. In the ROBOCARE domestic environment, we use the PARIDE (Process-based frAmework for oRchestration of Dynamic E-services) framework [10], which works according to a higher meta-level with respect to the XML-based languages. The mapping on technological models will be addressed at later steps in the development process.

The aim of PARIDE is to define a conceptual model of e-services which is independent of specific technologies. It is referred to as *e-service schema.* The main feature of this conceptual model is that it allows to specify not only an e-service's static interfaces, but also its behavior and evolution over time. For this purpose a state machine representation is used. Every change in the state machine is triggered by an incoming message and optionally generates an output. Specifically, an e-service schema consists of two parts: the first describes its interfaces, and the latter describes the conversations of the e-service, that is, possible interactions the e-service can be involved in. In order to show how this works, we describe the e-service of a Stereo Camera Agent so that it exports the 'still-image' service. The complete conceptual schema of this e-service is detailed in fig. 2.2. The very first part is the definition of the needed data structures (type classes). The name of the e-service (Stereo_Camera) identifies the "type" of service, while the property field describes the characteristics which are associated to the service. Thus, if there are different agents which provide the same service, they will all have the same name and different property fields. The messages resemble the public methods of a Class in Object-Oriented Programming languages: they are the only way an e-service can interact with the world.

3. People and Robot Localization and Tracking System

The component we are describing in this section has been developed in order to provide information to the overall system regarding pose, trajectories, and detection of special events about persons and robots acting in the experimental domestic environment. We call this component People and Robot Localization and Tracking Agent (LTA). This module exports a number of services, that may also be customized according to the need of other modules. A set of services exported by the module is: "photo, whereis, whichpose, howlong_in_a_pose, howlong_here, howmany_persons, howmany_robots, robot_close_to_a_person, what_activity". This information is very useful for other agents in order to assess the situation in the environment and to take decisions about changes that are required (see for example the Execution Monitoring Agent described in section 4). Therefore, the LTA is a primary component that has the objective of recognizing interesting situations and reporting them to other decision-making agents.

The basic technique involved is stereo vision in order to detect and track persons and robots in the environment. Moreover, from the extraction of a set of features regarding tracked objects (like eccentricity, height, size, etc.) it is

possible to recognize specific situations and poses (like for example if a person lays down on the ground, or if she is sitting on a table).

The general form of this problem presents many difficulties: first of all, object tracking is difficult when many similar objects move in the same space; second, object recognition is generally difficult when the environment and the agents cannot be adequately structured; third, when moving observers are used for monitoring large environments the problem becomes harder since it is necessary to take into account also the noise introduced by the motion of the observer.

With respect to the above difficulties we present a solution that makes the following choices: i) we limit the number of moving objects in the environment to be at most two or three persons plus any number of robots, but in such a way to exclude very crowded environments; ii) persons are not marked in any way, while robots can be visually recognized by placing on top of them special markers; iii) we make use of a fixed stereo vision system that is able to monitor only a portion of the area in the domestic environment.

The goal of the present work is thus to elaborate the information computed by a stereo vision system placed in a fixed position in the environment, in order to determine the position and the trajectories of moving objects. Since the stereo camera is fixed in the environment, we exploit the knowledge of the background in order to identify other moving objects (persons and robots). Persons are recognized by a model matching between the extracted information and some predefined values (e.g. eccentricity, height, etc.), while robots are recognized through their markers.

In the literature there are several works on people localization and tracking through a vision system, aiming at realizing systems for different applications. These systems are based on the use of a single camera (see [13] for example), stereo vision [6, 1] or multiple cameras [15].

The systems based on a single camera are used for modeling and reconstructing a 2D representation of human beings (e.g. heads and hands) and mainly used for virtual reality and gesture recognition applications. In the ROBO-CARE project we are interested in determining the 3D position of a person in the environment and to track his/her trajectory while he/she moves. To this end stereo vision or multiple cameras can be effectively used in order to compute the 3D position of objects in the scene. In this way, the system is more efficient in object recognition and tracking and also provides the coordinates of such objects in the environment. For example, the stereo vision system described in [1] focuses on the development of a real-time system for person detection and tracking by using a model matching technique. Also the work in [6] uses information about stereo vision for identifying and tracking persons in an environment; person recognition is here implemented by using a set of patterns to be matched.

The prototype implementation of our system has the objective of providing for real-time processing of stereo images and for identifying people and robot

trajectories in the environment. Preliminary experiments on the system show the effectiveness of the approach, a sufficient accuracy for many tasks, and good computational performance.

3.1 System architecture and implementation

The system we are developing is composed of hardware and software components. Hardware sensor is a pair of Firewire webcams which constitute the Stereo Camera. They need accurate setting in order to work properly. The images coming from this sensor are managed by the Stereo Camera Agent. This software architecture is based on three main modules (fig. 2.3): *foreground/background segmentation,* that is able to distinguish pixels in the image that are coming from the background from the ones that represent the foreground (i.e. persons or robots moving in the environment); *Stereo computation and 3D segmentation,* that evaluates disparities only for the foreground in the two images and computes the 3D position of a set of points; these 3D points are computed by the stereo algorithm and are clustered in continuous regions; *Object identification and tracking,* that associates each 3D cluster of points to a specific robot (when its special marker is recognized) or to a generic person (if it is compatible with some predefined values). The use of a Kalman Filter makes the estimated position of objects in the space more reliable.

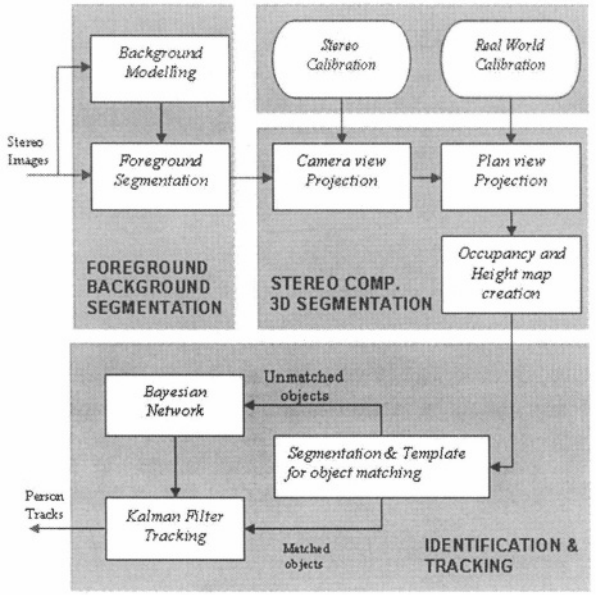

Figure 2.3. The software architecture is based on three main modules

The cameras of the stereo system are placed in a parallel with 18 cm of distance between them. They are installed at 220 cm of altitude from the ground and they point down with an angle of 20° with respect to the horizon line. With this configuration, considering the 62° field of view for each camera, the system is able to cover an area of 2.2 × 1.8 meters. The calibration of the camera is performed by fitting a model of a known *calibration object* to a number of images taken by the cameras. It can be recognized and measured by an automated calibration procedure (for more details, see the Small Vision System [9]).

Foreground/background segmentation. This task is obtained by computing the image difference between a stored image representing the background and the current image. In order to deal with the fact that the background can change over time (for example pieces of furniture or objects on a table can be moved), a special routine is used, which is able to detect these changes after processing a number of frames, updating the background model dynamically. Approaches for background updating [5, 8] usually maintain a model of the background that is updated when parameters of the actual frame are different from the model. In this way, however, persons that remain in the same position for a sufficiently long interval of time may be erroneously integrated in the background model. Since this situation is very common in a domestic environment (e.g. a person sitting at the table), we have devised a new method for background modeling and updating, that is able to distinguish animated objects from inanimated ones and to update the background models only for inanimated objects. In this way the system is both able to avoid to insert persons in the background model and to quickly integrate other objects in it. The method we present is based on the observation that animated objects (e.g. persons) change their shape even if they are still in a place. Therefore an analysis of the edges of an image is sufficient to identify the regions of the image in which there are (even very small) motions from the others. More specifically, we compute the horizontal and vertical edges of an image by means of a Sobel edge extractor operator $H_t(x, y)$ and $V_t(x, y)$ and then compute the *activity* of a pixel in the image as

$$A_t(x, y) = \beta \Delta E_t + (1 - \beta) A_{t-1}$$

where β is a factor that weights new measures with respect to previous ones, ΔE_t is the difference between the edges computed at time $t - 1$ with those computed at time t and can be computed for example as

$$\sqrt{((H_t(x, y) - H_{t-1}(x, y))^2 + (V_t(x, y) - V_{t-1}(x, y))^2)}$$

In our system, we thus update the background model only if the activity $A_t(x, y)$ is greater than a threshold. In this way, for example, if a person sits at a table and at the same time places a bottle on it, after a while the bottle will be integrated in the background (since its edges have low activity), while the person will not.

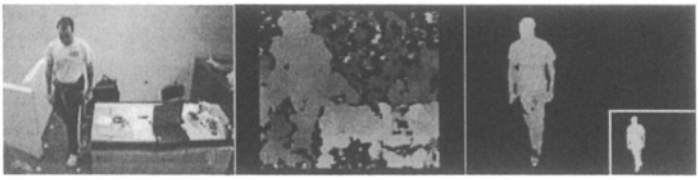

Figure 2.4. (a) original image (b) disparity (c) filtered disparity. Brighter points represent closer objects

Stereo computation and 3D segmentation. The stereo vision system we are using provides a real-time implementation of a correlation-based stereo algorithm. Disparity images represent points that are closer to the cameras with brighter pixels (see fig. 2.4b). Disparity images are computed only for foreground objects and are filtered in order to remove typical stereo noise due to false matches and little texture (fig. 2.4c) and then, through a geometric triangulation, we retrieve the 3D coordinates of all the pixels matched in the two images; this set of points is the 3D model of the scene seen through the cameras. 3D points are transformed in real world reference system through the external calibration [12] performed beforehand. The segmentation phase that follows stereo computation is in charge of clustering the set of 3D points in clusters, each representing a single object in the environment. From the set of 3D points computed, we build two maps that will be used for object identification and tracking: an *occupancy map* (fig. 2.5b) and a *height map* (fig. 2.5c). The former represents the projection of 3D points on the ground, while the latter marks every ground point in the space with the maximum height of the object occupying this point. The filtered height map is computed by applying a filter on the 3D points, which allows to discard points that do not belong to the object (fig. 2.5d).

Object identification and tracking. Each cluster of 3D points is associated to the representation of a moving object in the environment (either a person or a robot) that is tracked by the system. This step takes into account a simple model of a person that is computed from the occupancy and the height maps described above, and the recognition of robot-markers. Those clusters that do not match any of the specifications are simply discarded.

In order to track persons over time in presence of occlusions or when they exit and re-enter the scene, we associate three different templates for every identified object: T_C is a color-based template, T_H is a template based on the height map, and T_O is a template based on the occupancy map. Templates T_H and T_O are directly extracted from the respective height and occupancy maps. The template T_C is a 2-dimensional histogram that takes into account colors

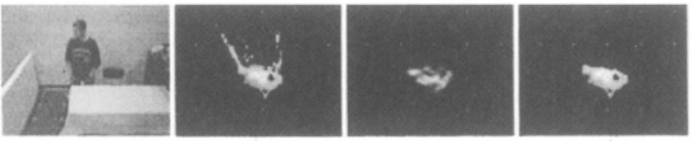

Figure 2.5. (a) original image (b) occupancy map (c) height map (d) filtered height map. Brighter points represent higher objects

associated to every object. It is represented as a $m \times m$ (we use m= 16) matrix TC which discretizes the 2-dimensional color space given by $rg = r - g$ and $by = \frac{2b-r-g}{2}$. Thus, $TC(rg, by)$ is the number of pixels in the image whose color (r,g,b) satisfy the above equations. Object recognition over time is thus performed by computing similarities of the three templates computed for every object. Moreover, in order to effectively track objects in the scene, we use a Kalman Filter on the 2D ground position of the object (i.e. on the ground projection of its center of mass) in order to filter occasional errors in stereo computation, 3D localization, and object identification. Finally, a Bayesian Network is used in order to determine situations that may arise when persons leave or enter the scene.

Although there are some situations in which the above method return erroneous results, especially when several objects are in the environment, a preliminary set of experimental results that are described in the next section shows the feasibility of the approach.

Experimental evaluation. A set of experiments to evaluate accuracy and computational efficiency were performed.

For measuring the *accuracy* of the system we used 9 markers with different distances and angles in scenario. The result after 40 experiments is shown in Table 2.1. These experiments have been carried out using the standard external calibration system.

The performance of *computational efficiency* without the grabbing using a 700 Mhz processor with the presence of one object to track is about 130 ms, by running the grabbing and filtering procedures the calculation time increase to 400ms with one object, 500 ms with two, and 650 ms for three moving objects in the scenario. The major part of the calculation time is dedicated to gaussian filtering and noise detection of the background.

4. The Plan Execution Monitoring System

So far we have shown the sensing capabilities we have deployed in the domestic environment. In particular, the stereo vision based people localization and tracking service is capable of compiling symbolic information (such as the

Marker	Distance	Angle	Err.medium	Err.Variance
M1	3.00 m	00°	7.3cm	6-8.2 cm
M2	2.70 m	00°	7.1cm	6-7.7 cm
M3	2.40 m	00°	6.8cm	5.8-6.7cm
M4	3.00 m	32°	9.8cm	7.8-10.7cm
M5	2.70 m	35°	9.6cm	7.8-9.7cm
M6	2.40 m	37°	8.8cm	7.8-9.7cm
M7	3.00 m	-32°	9.2cm	7.8-10.5cm
M8	2.70 m	-35°	9.6cm	7.8-9.7cm
M9	2.40 m	-37°	8.8cm	7.8-9.7cm

Table 2.1. Accuracy results.

presence, position and permanence of persons and/or objects) from the environment. This information defines, to a certain extent, the current state of the environment. The goal of this section is to describe an *execution monitoring system,* whose task is to follow the evolution of such states and to guarantee that these evolutions match with a set of predefined requirements. These requirements are expressed by means of a set of *activities* whose execution must generally occur within the scope of a complex set of *constraints.* These sets of activities are called *schedules.*

More specifically, a schedule consists of a certain number of activities, each of a predetermined duration and requiring some resources in order to be executed. One key point is the fact that activities can in general be temporally constrained, either individually or among one another: for instance, some operation in a schedule might be constrained not to start before, or not to finish after, a certain instant; in addition, there might be several *precedence constraints* between any two activities in the schedule: for instance, activity B might not be allowed to start before the end of activity A, and so forth.

The problem we describe in this section concerns how to manage a predefined schedule while it is being executed in a real world environment, namely the ROBOCARE Domestic Environment. We want the execution monitor to ensure schedule feasibility, taking into account the *real* status of the execution as it is reported by the people localization and tracking service.

The issue of schedule *consistency* has two aspects: on one hand, the totality of the schedule's temporal constraints, i.e. *release time* constraints, *deadline* constraints, *precedence* constraints as well as others, should be kept satisfied at all times, as they constitute an integral part of the schedule specifications. A schedule where all the temporal constraints are satisfied, is said to be *temporally consistent;* on the other hand, *resource consistency* must be constantly maintained as well, since it is obviously not possible to perform operations when the necessary resources are not available.

One of the major difficulties arising when working in real environments consists in counteracting the effects that the possible unexpected events may

have on the schedule *in a timely manner.* Indeed, when schedule adjustments must be performed, we are interested in detecting whether or not the contingency has caused an inconsistency in the daily schedule of the assisted person.

Research in scheduling and execution monitoring has produced two approaches to this problem, namely the predictive approach and the reactive approach [7]. The execution monitoring system we describe herein is based on the latter. In slightly simplified terms, we can say that according to the reactive approach, the execution monitoring system attempts to maintain consistency by manipulating the schedule every time it is deemed necessary. The current state of execution of the assisted person's tasks is provided by the environmental sensory services invoked by the execution monitor. Thus, the monitoring system follows the actual execution of activities and can properly react to the occurrence of unexpected events.

The task of the system is therefore twofold: on one hand, it is to represent the possible damages on the schedule, fire the proper repair action and continuously guarantee its executability; on the other hand, it is to ensure that all the scheduled activities are in fact positively executed, issuing warnings and/or alarms in the opposite case. To this aim, an *Execution Monitor* has been developed, which exhibits reactive behavior and conveniently re-adjusts the schedule's activities by means of the *ISES* procedure (Iterative Sampling Earliest Solutions) [3], a constraint-based method originally designed to solve the RCPSP/max problem (Resource Constrained Project Scheduling Problem with Time Windows).

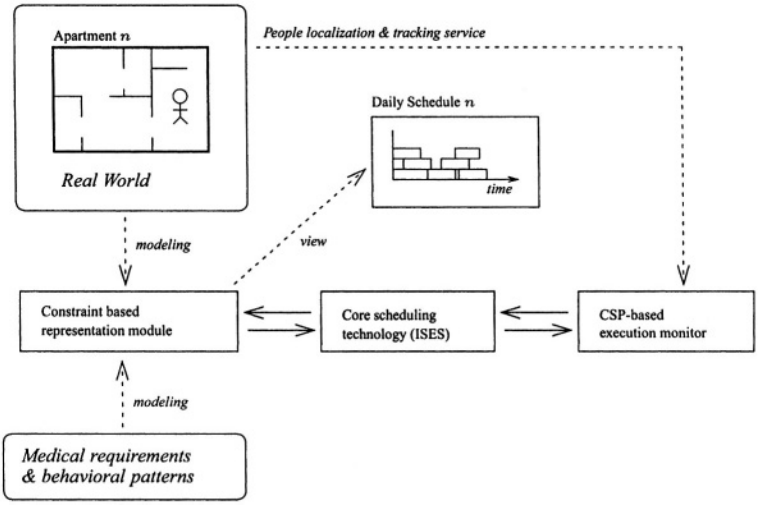

Figure 2.6. The execution monitoring system.

As shown in fig. 2.6, the execution monitor is interfaced with (a) a core scheduling system, through which it has access to the constraint representation of the schedule in its current state, and (b) with the real world, through the environmental sensory services which are responsible for signaling the exogenous events. The Schedule Execution Monitor has been developed as an integration to the *O-OSCAR* (Object-Oriented Scheduling ARchitecture) tool [2], an existing constraint-based software architecture for the solution of complex scheduling problems. The details of the core scheduling technology are outside the scope of this article. Let us now focus on how contingencies are represented and dealt with in our CSP-based execution monitoring component.

4.1 Representing Contingencies

One key feature of the Execution Monitoring System is the ability of modeling a realistic set of possible exogenous events. Information about all deviations from the nominal timing of the activities to be executed is perceived by a number of sensors distributed in the environment (such as cameras), and such information is properly processed and translated into meaningful input for the Representation Module, which constantly stores the world representation (see fig. 2.6). Typical consequences triggered by an exogenous event may reflect in the presence in the schedule of either a *temporal inconsistency* or a *resource inconsistency.* The first type of inconsistency occurs, for instance, if a delay has pushed some activities beyond some pre-defined temporal deadlines. A resource inconsistency, on the other hand, can happen if the delay has pushed an activity in a time zone where the overall resource requirement exceeds the maximum resource capacity, due to the requests of the other activities which operate in the same interval.

Three kinds of events can currently be dealt with by the execution monitor, as they represent a realistic set of incidents for the ROBOCARE scenario: *delays* of activities, variations in activity *durations* and *resource breakdowns.* Let us comment on each of these cases briefly.

As mentioned above, a sensor response may induce a sudden and unexpected time shift on one activity. This is represented by inserting a new *precedence constraint* between a particular time point (namely, the *origin* of the temporal network) and the *start time* of the delayed activity.

The next exogenous event which can be modeled in the execution monitor is a change of duration of an activity. This change is represented by substituting the activity with a new one having the same characteristics as for resource requirements, but of course different duration (which may not necessarily be *larger*).

To conclude, we model a resource breakdown by the simple insertion of a new *ghost* activity with the only aim of adding another source of contention in the schedule. The ghost activity will make use of the resources which have collapsed, *ofexactly the capacity that has to be collapsed.* In other words, the

condemned resource will be "eaten out" by the ghost activity, obtaining as an overall effect, its partial or total breakdown.

4.2 The Execution Monitor

As shown in fig. 2.6, contingencies are posted to the core scheduler, which is responsible for updating the constraint-based representation of the world, maintaining it perfectly consistent with the evolution of the real environment; the main issue is that updating the data stored in the representation module in accordance to the information gathered from the environment may introduce some inconsistency in the schedule representation. The execution monitor reacts to these inconsistencies as they are detected, namely attempting to take the schedule back to a consistent state, so as to keep it executable.

The repair action is performed by exploiting the capabilities of the ISES algorithm, which is used as a "black box"; in other words, schedule revisions are approached as *global* re-scheduling actions, without focusing on a particular area of the schedule, an approach similar to [11]. This global approach requires some preventive action to be taken before the ISES procedure is fired, in order to have the necessary control on schedule repair choices. In other words, we can guide the revision process by preventively constraining the activities, depending on the strategies we want to realize.

A number of primitives have been developed which make the dynamic insertion and deletion of temporal constraints possible. At present, these primitives handle the insertion of four types of constraints: (1) *Precedence Constraints,* which impose a temporal relationship between two activities A and B such that A *cannot start before the end of activity* B; (2) *Deadline Constraints,* which impose a time boundary on an activity's end time, i.e. the activity *cannot end after a certain deadline*; (3) *Release Time Constraints,* which impose a time boundary on an activity's start time, i.e. the activity *cannot start before a certain release time;* (4) *FixTime Constraints,* which are similar to the previous type, but impose a more rigid constraint on the activity start time, namely that the activity is *not allowed to start neither before, nor after a determined fixed instant*

By means of these four primitives, the execution monitor can effectively employ the core scheduling module to adjust the current schedule in the event that an unforseen event occurs in the context of the assisted person's nominal behavior. The pseudo-code in fig. 2.7 shows the detail of the execution algorithm, where *T* and *execSchedule* represent, respectively, time and the current nominal schedule. At predetermined intervals of time, the environment is sensed by invoking the services of the sensory subsystem, in order to detect any possible deviation between the expected and the actual situation; if unforeseen events have occurred, they are modeled in terms of the contingency representation schemata shown in Section 4.1. The next step consists in checking the *temporal consistency,* as in the schedule updating process we may have added to the

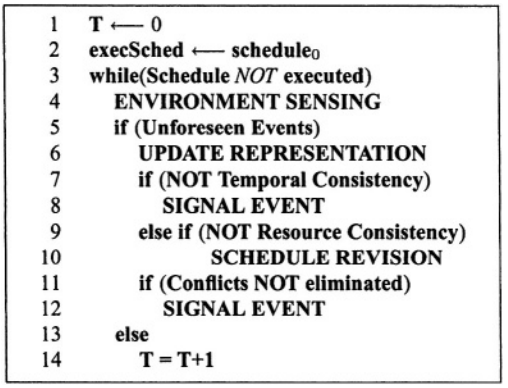

Figure 2.7. The schedule execution algorithm.

representation some temporal constraints which are in conflict with the existing ones.

If consistency is lost, the algorithm must signal the event, as the presence of an anomalous situation is detected. In this case, no repair action is possible unless some previously imposed constraints are relaxed; the signal will then be properly captured and used for instance to issue a warning or to fire an alarm. If time consistency is not spoiled, *resource consistency* must be checked as well, because the occurrence of the exogenous event may have introduced some resource conflicts in the schedule, although leaving it temporally consistent.

If no resource conflicts are present, the execution of the schedule may continue; otherwise, a *schedule revision* must be performed, in the attempt to eliminate resource contention. If the schedule revision process succeeds in eliminating the conflicts, execution may continue; otherwise the algorithm must again signal the occurrence of the anomaly for further processing.

Let us take a closer look at the way the activities in the schedule are actually manipulated during execution and repair. As previously stated, the approach used in our Execution Monitor can be considered as *global* [4, 14] in that the revision procedure accepts the schedule *as a whole,* tries to solve all the conflicts and returns the solution. In other words, the ISES procedure does not make any difference between terminated, started or yet-to-start activities, and has no concept of *time.* As a consequence, the only chance at our disposal to exert some control over the activities is to do it in a preventive way, that is, *before* the ISES procedure begins the manipulation of the schedule.

Such control is necessary for at least two reasons: (a) we want to keep the solutions *physically consistent* at all times; (b) we want to retain the possibility to satisfy a set of *preferences* given by the users.

The next paragraphs focus on some practical issues arising during schedule execution that we have to face in order to obtain meaningful results from the re-

vision process. We assume the schedule under execution with *current execution time* = t_E.

Physical Consistency. Consistency must be satisfied at all times, because the current schedule represents the nominal behavior of the assisted person. There are many ways in which physical consistency may be spoiled as a result of an inattentive action; for instance, the re-scheduling procedure may try to re-allocate some activities which have already started execution.

Clearly, this represents an inconsistent situation and must always be avoided. The problem is solved by inserting a new *FixTime constraint* for every activity whose start time $st = t_E$. By doing so, we impose a very strict temporal constraint on the activity start time: *all the solutions found by ISES which require a temporal shift of the constrained activity, will be rejected.*

As another example, the re-scheduling procedure may allocate some activities *at the left* of t_E in the temporal axis, which would be equivalent to allocating operations *in the past.* All we have to do in this case is to introduce in the schedule as many *Release Time constraints* as there are activities whose start time st is greater or equal than t_E. In other words, we constrain all the activities which have not yet started, not to begin execution before the current execution time. Again, this does not necessarily mean that these activities will be moved by the core scheduling algorithm: anyway, should they be re-allocated, they would certainly be positioned at the right of t_E.

Preferences Management. In many cases it is essential that any revised solution be as close as possible to the last consistent solution found by ISES; the closer any two solutions are, the higher their level of *continuity.*

It is in fact desirable (and plausible) that, despite the possible exogenous events that may occur during the execution of a schedule, this remains as similar as possible to the initial schedule, since it models the behavior of the assisted person. Schedule continuity can be controlled by leaving or removing the *precedence constraints* possibly imposed in the last execution of ISES. It is known that ISES resolves the conflicts by inserting a certain number of extra precedence constraints between the activities, in order to separate them in the areas of greater resource contention; these extra constraints are not part of the original problem and are only there to solve a particular resource conflict.

If ISES is run many times consecutively, it has been observed that the consecutive schedules which are obtained show higher levels of continuity if the constraints which were added at previous runs are maintained. Obviously this is due to the lower degree of freedom retained by the activities in the two cases: the more constrained the activities are, the lower the possibility that the new solution differs from the old one.

As a last observation, with a proper handling of the temporal constraints it is also possible to bias the schedule in order to satisfy user's preferences; for

instance, before schedule revision it could be possible to specify the *degree of mobility* of the activities, such as maximum delays, preferred anticipations, and so on. By exerting this kind of preventive control, it is possible to express preferences on the behavior of every individual activity before schedule revision, thus obtaining a solution which best suites the user's desires.

5. Integrating Sensing and Execution Monitoring: a Running Example

In this section we will give a description of illustrative situations in which our system monitors and controls the daily activities of an elderly person. On the basis of the following simple examples, we intend to give the readers a flavour of the potential of our tool when employed in more realistic and complex scenarios.

The agents involved in the monitoring activity are, on one hand, the people localization and tracking service described in section 3, and on the other the execution monitoring component we have just described in the previous section. The first component retains the ability to detect the presence of a person in the environment and deliver the coordinates of his or her position obtained by image analysis; on top of this basic capability, a series of *situations* can be estimated, such as when a person is sleeping, eating, and so on. On the other hand, the task of the execution monitoring agent is to guarantee the correct schedule execution ensuring its consistency at all times. Basic information on the state of the assisted elderly person is gathered by invoking the services of the localization and tracking system. This information is processed by the execution monitoring system in order to assess the actual state of the assisted person with respect to the nominal schedule.

For the sake of illustration let us suppose that the next action to be monitored is the lunch activity (fig. 2.8).

The activity to be monitored is subject to a number of previously imposed temporal constraints; such constraints are intended to model the timing of any individual operation, as well as to model the possible synchronizations among the several activities in the schedule. In this example (see fig. 2.8 (a)), the operation of eating a meal is characterized by a duration **d**, a start time **st** and an end time **et**. A proper imposition of the temporal constraints is necessary to model a situation where there is a certain amount of flexibility regarding the possible start time and end time of the activity: this flexibility is modeled through the specification of a *release time constraint* and a *deadline constraint,* defining respectively an earliest start time **est** and a latest finish time **lft** for the activity, such that $\mathbf{st} \geq \mathbf{est}$ and $\mathbf{et} \leq \mathbf{lft}$. As a consequence, the activity will retain a certain degree of mobility on the temporal axis, as a too strict timing would not be realistically acceptable. When the execution monitor requests information on the status of the assisted person, the people localization and tracking service analyzes the information it observes through the Stereo Camera. In this specific case, the basic image processing can deliver the position in

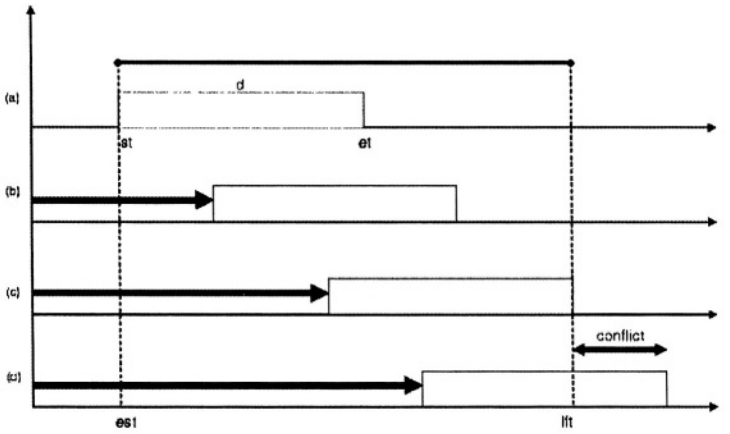

Figure 2.8. A first example.

the room of the tracked person compared with the coordinates of a series of landmarks (such as elements of furniture). Moreover, based on contextual information and landmark analysis, the LTA service can verify whether the person is indeed sitting at the table. The sensory service replies to the execution monitor's request with this information and activity monitoring proceeds under nominal conditions. In the opposite case, the execution monitor may conclude that, indeed, the person is not eating. This interaction occurs according to the e-service state machine shown in fig. 2.9. In this example, the assisted person is expected to start having lunch not before time **t = est**. This means that at **t = est** the execution monitor requests a confirmation of this from the sensory service. If this is not the case, the world representation must be modified accordingly, by inserting a *delay* on the monitored activity, that is, by imposing a new precedence constraint (see fig. 2.8 (b)). The extent of the delay can be decided depending on a variety of factors, mostly related to the frequency of the execution monitoring cycle.

Obviously, the insertion of the new precedence constraint may have direct consequences on the consistency of the schedule: fig. 2.8 (b) depicts the sit-

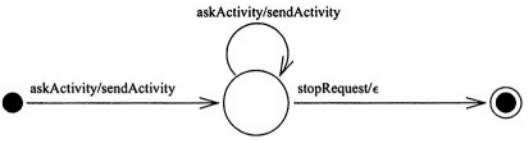

Figure 2.9. The 'what-activity' e-service state machine, whose invocation occurs through the askActivity/sendActivity messages.

uation where the temporal right shift imposed on the delayed activity is still compatible with all the previous temporal constraints: in this example, the only constraint which could be violated by the new insertion is the deadline constraint insisting on the monitored activity.

The *environment sensing/update representation* cycle continues until the imposition of a further delay on the same activity eventually generates a time conflict (fig. 2.8 (c)(d)), due to the fact that the deadline constraint is violated. The insertion of the last precedence constraint is disallowed, the system thus recognizes the inconsistency and immediately signals the event, by firing an alarm or issuing a warning.

In order to highlight the high level of expressiveness the execution monitor is able to offer, we present a slightly more complex example (see fig. 2.10). The new schedule is composed of six activities, according to the following table:

A1: breakfast **A2: lunch** **A3: dinner**
A4, A5, A6: first, second, third medical treatment

Despite the low number of activities in the schedule, the modeled situation is already non trivial: in fact, **A1**, **A2** and **A3** are characterized by their own **est** and **lft**, defining a *release time* and a *deadline* constraint for each activity. In fig. 2.10(a) these constraints are represented as bold lines starting right above the interested activity. **A1**, **A2** and **A3** are also temporally connected among one another: in fact, breakfast and lunch must be separated by a minimum interval as they cannot be too close to each other; the same thing holds between lunch and dinner. As a consequence, the breakfast activity can be delayed without affecting the timing of the lunch, as long as the temporal distance between them keeps greater than the minimum allowed; should this minimum be violated, a delay on the first activity would inevitably impact on the start time of the second. The described effect is easily obtainable by forcing of two precedence constraints, between **A1** and **A2**, and between **A2** and **A3**. In the figure, such constraints are represented as bold arrows starting right below the interested activity (the length of the arrow representing the minimum distance allowed between the activities).

The three medical treatments (activities **A4, A5** and **A6**) deserve special attention; the situation we model here, reflects the typical circumstance where taking medicines is strongly related to the timing of meals: in this specific case, we suppose that the first medicine (activity **A4**) should be taken no later than the interval ΔT_1 after the end of activity **A1**; the second medicine (activity **A5**) should be taken *immediately before* **A2**, and that the third medicine (activity **A6**) should be taken *immediately after* **A3**.

It is easy to see that the temporal relationships in this schedule involve a relatively complex interdependence among the activities, as one single delay may have several effects: fig. 2.10(b) shows how a delay on **A1** (which has

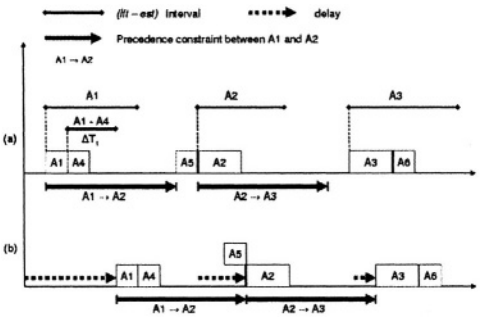

Figure 2.10. A second example.

direct repercussions on **A4**) may reflect on a delay on **A2** (and consequently on **A5**), as well as on **A3** (and consequently on **A6**). In this way, the system is able to immediately recognize the occurrence of a constraint violation even in case of significant temporal distance between the event which caused the delay and the instant of the violation. Notice that the activity which is directly involved in the delay is **A2** and not **A5**; **A5** is simply "dragged" by **A2** as a consequence of the strict temporal relationship between **A5** and **A2**.

Fig. 2.10(b) depicts a case where no constraint violation occurs; yet, in a real execution environment, many are the possibilities where the opposite may be true: the assisted person may for instance have breakfast and fail to take the first medicine in time, or may delay the lunch outside the allowed temporal window, or may forget to take the third medicine immediately after dinner. In each case, the system aims at maintaining the consistency of the flow of actions through a clever reallocation on the temporal axis.

One point which should be highlighted is that the patient's desires are taken into account by way of defining not too rigid time bounds on the schedule, as long as her or his preferences do not collide with some temporal limitations which are not relaxable as they might be imposed on the basis of essential medical prescriptions.

6. Conclusions and Future Work

In this article we have described an integration of two intelligent components, namely a Stereo Camera based environmental sensor and a CSP based execution monitoring software agent. The first component provides people localization and tracking functionalities, while the second is capable of recognizing inconsistencies in the execution of an elderly person's daily activities, dynamically reacting to exogenous events.

Since the two systems interact strongly (the execution monitoring service grounds its reactivity on the environmental observations made by the localization and tracking component), the high-level coordination mechanism has been

implemented in the form of e-services. Following the e-service philosophy, the two components are seen as agents, whose functionalities are exported as services.

These basic ingredients constitute an initial prototypical system which is currently developed for a testbed domestic environment. The ensemble of services provided by the two interacting components we have described aims at discovering the state of the assisted person and of the environment. This information is interpreted into a complex symbolic representation which can be used by the supervising entity to monitor the environment and to adapt to unexpected or unforeseeable perturbations in the expected behavior of the assisted person. Clearly, the reliability and effectiveness of the supervisory system is determined by the quality of the information returned by the sensing agents on one hand, and by the nature of the intervention the supervising entity imposes upon reacting to contingent events on the other. Currently, our system can recognize simple states based on the permanence of the assisted person in a relative position. Thanks to this symbolic information, the supervisor can monitor the correct execution of simple activities which involve these states, and signal inconsistencies which may occur as a result of delays in the sensing of an "expected" state,

In order to increase the effectiveness of the global supervisory system, our work in the immediate future will strive to i) enhance the array of situations the sensory agents can recognize (such as human posture recognition); ii) combine and integrate input from other devices such as mobile robots; iii) devise effective and pro-active contingent plans to broaden the scope of action of the reactive execution monitoring service.

Ultimately, the success of the ROBOCARE project's ambitious goals depends strongly on the integration of techniques from many disciplines.

Hopefully the proposed framework will provide novel perspectives in Ambient Intelligence, as well as a number of technically interesting solutions to component integration. We expect this work to forebode new sources for environmental information gathering, such as robotic sensors. Also, it will be interesting to integrate services offered by robotic devices with high-tech domotic components. Lastly, and certainly not least importantly, we are starting to address acceptability issues related to environmental monitoring.

Acknowledgments

This research is partially supported by MIUR (Italian Ministry of Education, University and Research) under project ROBOCARE (A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components).

References

[1] D. Beymer and K. Konolige. Real-time tracking of multiple people using stereo. In *Proc. of IEEE Frame Rate Workshop,* 1999.

[2] A. Cesta, G. Cortellessa, A. Oddi, N. Policella, and A. Susi. A Constraint-Based Architecture for Flexible Support to Activity Scheduling. In *Lecture Notes in Artificial Intelligence, N.2175.* Springer, 2001.

[3] A. Cesta, A. Oddi, and S. F. Smith. A Constrained-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics,* 8(1):109–135, 2002.

[4] L. K. Church and R. Uzsoy. Analysis of Periodic and Event-Driven Rescheduling Policies in Dynamic Shops. *Inter. J. Comp. Integr. Manufact.,* 5:153–163, 1991.

[5] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenswalb. Plan-view trajectory estimation with dense stereo background models. In *In Proc. of the International Conference on Computer Vision,* 2001.

[6] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. In *IEEE Conf. on Computer Vision and Pattern Recognition,* 1998.

[7] A. Davenport and J. C. Beck. A Survey of Techniques for Scheduling with Uncertainty. http://www.eil.utoronto.ca/profiles/chris/chris.papers.html.

[8] I. Haritaoglu, D. Harwood, and L. Davis. W4s: A real-time system for detection and tracking people in 2.5d. In *Proc. of ECCV98,* 1998.

[9] K. Konolige. Small vision systems: Hardware and implementation. In *Proc. of 8th International Symposium on Robotics Research,* 1997.

[10] M. Mecella and B. Pernici. Building flexible and cooperative applications based on e-services. Technical Report 21-2002, DIS - Università di Roma La Sapienza, 2002.

[11] S. F. Smith. OPIS: A Methodology and Architecture for Reactive Scheduling. In M. Zweben and S. M. Fox, editors, *Intelligent Scheduling.* Morgan Kaufmann, 1994.

[12] R. Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition,* 364–374, 1986.

[13] C. Wren, A, Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1997.

[14] S. D. Wu, H. Storer,and P. C. Chang. One-machine rescheduling heuristics with efficiency and stability as criteria. *Comput. Oper. Res.,* 20:1–14, 1993.

[15] D. Yang, H. Gonzalez-Banos, and L. Guibas. Counting people in crowds with a real-time network of image sensors. In *Proc. of ICCV,* 2003.