A Flexible Control Architecture for Extended Autonomy of Robotic Assistants

Christopher King, Xavier Palathingal, Monica Nicolescu, Mircea Nicolescu

***Abstract*—A major challenge in deploying robots into the real world is the design of an architectural framework which can provide extended, natural and effective interactions with people. Within this framework, key issues that need to be solved relate to the robots' ability to engage in interactions in a natural way, to deal with multiple users, and to be continually perceptive of their surroundings. In this paper we propose a robot control architecture that addresses these issues. Our architecture has three main key features. First, it enables the representation of complex, sequential and hierarchical robot tasks, typically needed for service applications, in a behavior-based framework. Second, it provides the robot with flexibility in dealing with multiple users, such as to accommodate multiple user requests and task interruptions, over extended periods. Third, through its visual detection mechanism, the architecture allows the robot to identify when people are requesting its interaction. We demonstrate our approach on a Pioneer 3DX mobile robot, performing service tasks in a real-world environment.**

***Index Terms*—Human-robot interaction, behavioral robotics, personal robots.**

I. Introduction

A MAJOR challenge in designing robots for service or assistive applications is to enable natural and accessible interaction between robots and non-technical users, while ensuring extended, robust performance in complex, uncertain, dynamic human environments [14]. While significant advances have been made in increasing the complexity of tasks that robots can perform, a limitation that prevents robots from operating outside of the lab is that they lack the perceptual abilities to perform in real-world applications. In most cases, robots are programmed to execute a single task, which is switched on/off by a programmer. Robots typically perform these tasks "blindly", being generally unaware of other people and robots in the environment. In addition, such robots are limited to performing the single task that the programmer had requested. We propose a control architecture that introduces a level of flexibility and perceptual ability that allows robots to overcome traditional limitations and operate in more dynamic settings. Our architecture equips robots with the ability to monitor their surroundings and detect when other social agents require their interaction. Our architecture provides the means for long-term autonomy by enabling robots to manage a large repertoire of tasks over extended periods. Finally, our system is designed for realistic assistive applications, where multiple people are simultaneously competing for the robots assistance.

E-mail: cjking|xavier|monica|mircea@cse.unr.edu
Computer Science and Engineering Department
University of Nevada, Reno.

Vision-based perception provides the richest information required for effective human-robot interaction. In particular, the ability to distinguish between different people and to identify basic human postures is essential for the success of the interaction. While significant work has been done in these areas [2], [21], robust recognition routines are frequently too computationally expensive to be run in real time on a robot, especially when a robot must identify multiple object patterns and in a dynamic, real-world environment. In this paper, we describe a real-time identification and posture recognition algorithm and we demonstrate its use in interactive experiments using a mobile robot.

The contribution of this paper is a framework that addresses three key issues for human-robot interaction in the context of service applications: 1) complexity and robustness of task representations, 2) long term interactions with the environment and other agents, and 3) detection and recognition of multiple users.

The remaining of the paper is structured as follows: Section II gives a discussion of related work. Section III presents our interactive framework, Section IV describes our control architecture, Section V discusses our vision-based perceptual approach and Section VI describes the experimental setup and results. Finally, Section VII contains our conclusion.

II. Related Work

The importance of developing robots that can provide useful services has been widely recognized and shown by the numerous application domains for which robots have been designed: agriculture and forestry, mining and construction [5], exploration and inspection [30], undersea applications [35], cleaning [11], education [26], search and rescue [22], space exploration [3], medicine and health [18]. In these approaches, the interaction between humans and robots is mostly a means for performing a job, in which the robots are regarded as tools that can be instrumented (i.e., tele-operated) toward achieving some desired goals.

The nature of applications for service and assistive robotics requires a different level of autonomy and interaction, in which the robot is regarded as a partner [9] to the human user. A significant effort in this area has been shown in designing entertainment and toy robots [32], [20], in which the focus is on designing robots that exhibit social, human-like characteristics: expressing or perceiving emotions, being sociable [8], establishing or maintaining relationships, and making friends [1]. For domains such as service or assistive applications [16], [27], a significant challenge is that robots not only can express but also understand this wide range of social cues. In this context, awareness of the world [10], [33] serves to achieve better interaction between humans and robots. To date, the issue of awareness for human-robot interaction has mostly been addressed as enhancing a human's awareness of the robot's activities [12]. With social interactions becoming more prevalent in the robot domain, it is important that the focus shifts toward increasing the robot's awareness of the world and humans around it. The approach we propose in this paper provides new capabilities that allow robots to perform in highly interactive environments, with numerous users, and multiple requests. We are able to achieve this interaction without resorting to complex hybrid architectures found in previous systems [6], [7]. Our system allows for the representation and execution of hierarchical tasks (typical for hybrid systems) within the framework of behavior-based systems, using a unique representation throughout the entire architecture. With our proposed architecture, a robot can provide service over extended periods (limited mostly by battery power requirements), with no intervention or help from the human designer.

III. Interactive Framework

The problem we address in this work is aimed at increasing a robot's autonomy over extended periods, and providing it with the skills needed in typical service or assistive application domains.

A service or personal robot will most likely have to perform in the presence of multiple users. This imposes constraints on the robot's behavior, as it will have to adjust its execution to accommodate several users who may solicit its attention and/or services during overlapping intervals. A typical example of such a situation is making a request to a robot while it is still working on another, previously assigned task. The robot needs to handle such situations appropriately: First, it should be detect the new call and interrupt its current activity to receive the request, it should then make appropriate action regarding which task to pursue next. The situation may further be complicated by different levels of authority existent between users, or by various priorities of the requested tasks.

Our framework for enabling this functionality consists of two computational modules for *visual detection* and *control*, linked into a unified control architecture. The role of the *visual detection* module is to identify, at any time, if any known users are attempting to interact with the robot. This module relies on postures of people that the robot is trained to recognize (as described in Section V). If any person/posture is detected, this information is transmitted to the *control module*, which takes appropriate decision on what the robot should do next. If the posture is only detected for a brief time, this represents a case in which the person was merely a passer by. However, if the posture persists in the robot's visual field, this is an indication that a person is trying to get the robot's attention. The job of the *control* module is to decide on the appropriate action to take in these circumstances.

As previously mentioned, in our system the robot is trained to recognize models of human users, consisting of different postures from different people. Currently, the robot associates each posture with a different task (robot service) that the users would like to request from the robot. Each task has an associated priority, which is either *low*, *regular* or *high*. When a posture is detected, the robot starts performing the service associated with that posture, unless the robot was already engaged in a task of higher or equal level of priority. In this situation, the robot adds the new request to a queue of tasks, and continues with its previous task execution. However, if a higher-level request is received, the robot switches to the new task, and moves the currently executing task to the queue. The robot processes the tasks from the queue based on their priority and incoming order. Our architecture provides the flexibility of using different priority queue strategies, as will be demonstrated in the experimental results. This prioritized task-switching process is typical for the types of decisions people make in their daily activities, and is also expected to occur in the service robot domain. While people perform this activity switching with great ease, the difficulty for the robots is to keep track of the status of the tasks when they are interrupted, such that the task could be resumed from the same point at a later time. This poses a significant challenge for the control architecture and the corresponding task representations. In this paper, we propose a behavior-based control architecture, which through its representations lends itself naturally to recovering from interrupted tasks, without the need to explicitly store any additional state information. This architecture is described in the next section.

IV. Control Architecture

The architecture we propose in this paper is aimed at providing an appropriate infrastructure for executing complex, sequential and hierarchical tasks, similar to what robots might have to perform in real-world applications. We base our approach on the Behavior-Based Control (BBC) paradigm, one of the most popular approaches to embedded and robotic system control. The contributions of the proposed control architecture are that **1)** It enables the use of both command *arbitration* and *fusion* within a single control representation and that **2)** it allows the encoding and robust execution of sequential and hierarchical tasks. Historically, the two main action selection mechanisms of *arbitration* and *fusion* have been mostly employed separately in robot control [28], thus limiting the range of tasks that robots can execute. By recognizing the ability of *arbitration* to encode temporal sequences and the ability of *fusion* to combine concurrently running behaviors, we merge the strengths and features of both within a unique task representation. For behavior representation we use a schema-based approach, similar to the work in [4]. This choice is essential for the purpose of our work because schemas with BBC provide a continuous encoding of behavioral responses and a uniform output in the form of vectors generated using a potential fields approach.

Our controllers (Fig. 1) are built from two components: *behavior primitives* (BPs) and *fusion primitives* (FP), which through the combination processes described below result in controllers in the form of *behavior networks* [25]. The *behavior primitives* perform a set of actions under given (relevant) environmental conditions. These primitives are meant to express the basic, general capabilities of the robot and need not be oriented to accomplishing a broad range of tasks. A *fusion primitive* encapsulates a set of multiple concurrently running primitive behaviors through linear combination of the motor commands. Each primitive behavior component brings its own contribution to the overall motor command. These contributions are weighted and fused through vector addition. For example, an *obstacle avoidance* behavior could have a higher impact than *reaching a target*, if the obstacles in the field are significantly dangerous to the robot. Alternatively, in a time constrained task, the robot could give a higher contribution to getting to the destination than to obstacles along the way. These weights affect the magnitude of the individual vectors coming from each behavior, thus generating different modalities of execution for the task.

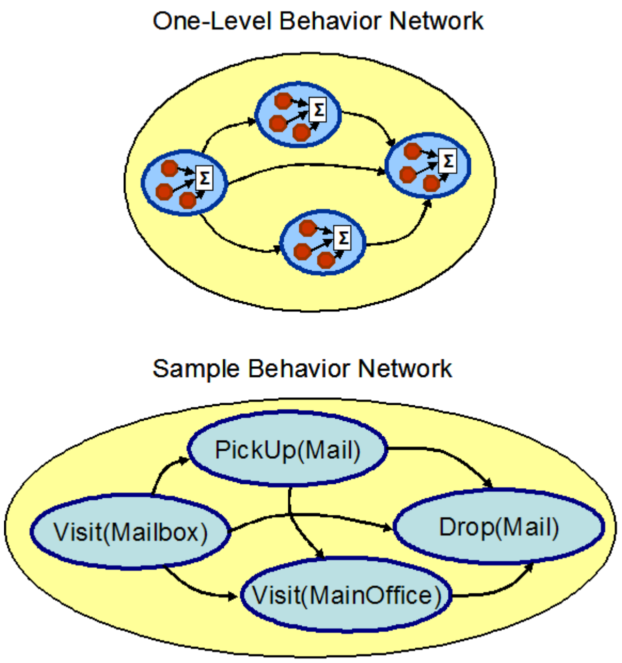


Fig. 1. Top: Representation of a generic behavior network, built of fusion primitives. Bottom: a sample behavior network. The links between fusion primitives represent task-specific precondition-postcondition dependencies.

A. Fusion Primitives

Each *fusion primitive* (Figure 2) has a representation of the goals it achieves, expressed as abstracted environmental states. The state of the goals is continuously monitored and updated from sensory data. The component *behavior primitives* receives information from the sensors, which is first used to detect if the behavior is active or not, given its preconditions. For example, in an *obstacle-avoidance* behavior, the presence or absence of an obstacle is abstracted from the range-finder information. If an obstacle is present, the precondition is met and the behavior is active. Otherwise, the behavior remains inactive. The *active/not active* status of all behavior primitives is encoded in a n-dimensional vector, where n is the number of BPs. This vector, which we call a *behavior applicability condition* (BAC), contains for each behavior a 1 or a 0, depending on whether the behavior is active or not. For a given set of n *primitive behaviors*, theoretically there could be 2^n combinations representing whether the n behaviors are active or not, based on their pre-conditions. Practically, this number is much smaller, due to the fact that some behaviors are triggered by similar environmental conditions (such as the presence of an obstacle, for example), and thus some combinations are impossible to achieve. For each possible BAC, the *fusion primitive* has a different set of fusion weights, which are used for behavior combination. The sets of weights for the multiple possible BACs are stored in a table, as shown in Figure 2. The index of each row in the table is the decimal equivalent of the n-bit BAC value.

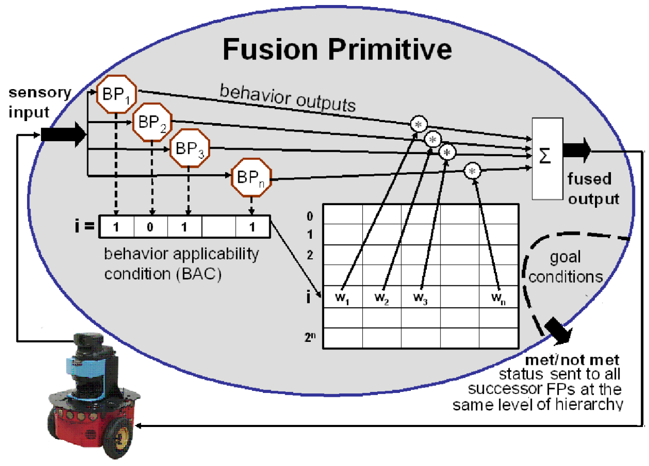

Fusion Primitive
sensory input
behavior outputs
BP1
BP2
BP3
BPn
Σ
fused output
i =
1 0 1 1
behavior applicability condition (BAC)
0
1
2
i
2n
w1 w2 w3 wn
goal conditions
met/not met status sent to all successor FPs at the same level of hierarchy

Fig. 2. Representation of a fusion primitive: Sensory input activates a corresponding set of stored weights (BAC) to fuse the underlying behavior primitives (BP).

The weights from the corresponding BAC modulate the magnitude of control vector output by the individual primitives, thus influencing the resulting command from fusion and consequently the way the robot interacts with the world. At each time step t, each *behavior primitive* BP_i provides a response output vector v_i^t, which represents a desired heading for the robot. The *fusion primitive's* output is a linear combination of the vectors $[v_1^t \cdots v_n^t]$, according to the BAC superposition weights $W^t = [w_1^t \cdots w_n^t]$:

$$V_r^t = \sum_{i=1}^{n} w_i^t v_i^t \quad (1)$$

We consider heading to be the most important consideration for behavior fusion in 2D navigation[1]. Consequently, we normalize command vectors to unit length.

The multiple BACs represent different environmental situations, since different behaviors are "applicable" in each case. The weights of behaviors within each BAC encode the mode of performing the current task given the situation and, thus

[1]Speed could easily be incorporated into our formulation. However speed is marginalized over time by the slow drive of our robots.

within each BAC, the weights of the applicable behaviors are constant. For example, for a *target reaching* task, the robot could behave under the influence of *corridor-follow*, *target-follow* and *avoid-obstacle* behaviors if in the presence of obstacle, but would behave only under the influence of *target-follow* if in an open space.

Inferring the fusion weights is a challenging task that would normally require time-consuming fine-tuning. In a previous work, we developed a method that allows weights to be learned through human-provided demonstration [23]. A controller, using weights evolved in this manner, was shown to be sufficiently robust to handle complex environments. Using this method, we trained a single controller to drive the robot for all our experiments.

B. Hierarchical Task Representations

With fusion primitives alone, a controller can only encode *flat representations* of tasks involving sequencing of fusion primitives. While such an architecture is expressive and flexible, it does not have the modularity needed when new, more complex tasks would have to be created from already existing ones. The best solution would be to specify a new task using abstractions of these existing modules, rather than combining their underlying behaviors into a larger, flat network. We enable this higher-level of representation by grouping fusion primitives into *behavior networks* [19], [25]. Behavior networks can be nested, allowing for the construction of hierarchical representations of robot tasks. In these networks, the links between components represent task-specific precondition-postcondition dependencies. These links provide a simple and natural way of representing complex sequences of activities and also of hierarchically structured tasks (Figure 3).

We use the term *metabehavior* to describe both fusion primitives and nodes of a behavior network in that both have similar functions in the network. Each metabehavior encapsulates information about the behavior's preconditions and its goals (postconditions). These conditions are continuously monitored whenever the behavior is active, in order to ensure the proper execution of the task. The postconditions of a behavior network node will be true when the execution of the subnetwork it represents is finished. The only difference between a behavior network node and a fusion primitive is that it activates underlying metabehaviors, while a fusion primitive activates only its component primitive behaviors. Thus, when a metabehavior is not active, all subordinate metabehaviors are disabled, and therefore can be regarded as not relevant to the task. When a behavior network node becomes active, all its underlying components are enabled, and the subnetwork becomes the current "network" that is being executed. When the execution of the subnetwork finishes, the behavior network node updates its goal status. Since the successor behaviors continuously check this status, they will detect the achievement of that goal and the execution continues with the new network node. To perform tasks encoded with this representation, the robot starts by activating the metabehavior at the topmost level in the task. The execution of the task's steps proceeds as described above.

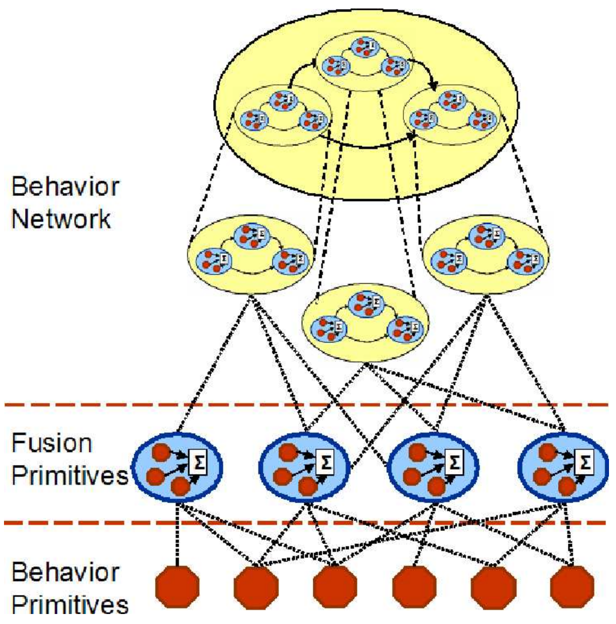


Fig. 3. A generic hierarchical task representation. The links between fusion primitives in the behavior network represent task-specific precondition-postcondition dependencies.

In this architecture, using the links as task-specific activation conditions enables the reusability of behaviors and run-time reconfiguration of robot tasks. The behavior network representation has the advantage of being adaptive to environmental changes, whether they be favorable (achieving the goals of some of the behaviors, without them being actually executed) or unfavorable (undoing some of the already achieved goals). Since the pre and post-conditions of behaviors are continuously monitored, the system executes the behavior that should be active according to the current environmental state, thus providing the robot a sense of "awareness" about its progress in the task. With these capabilities, the architecture enables the robot to keep track of the completed parts of the task, which allows dealing with task interruptions without any additional modifications. When a task is interrupted, its execution is suspended, but the behaviors preserve the current status of execution. When the task is resumed, the information implicitly stored in the behavior network controller enables the robot to continue the task from the point where it was interrupted. The behaviors' continuous grounding in sensory information allows the robot to correctly perform the task, even if the environmental conditions have changed since the task was suspended.

V. Vision-Based Human Posture Recognition

The role of the *visual awareness* module is to provide the robot with the capability of detecting the presence of humans that might be interested in interacting with the robot. Toward this end, we developed visual capabilities that can recognize human postures that are likely to be relevant to the robot-human interaction. The postures are described below and examples are shown in Fig. 4 (first row).

The Standing Posture – The most obvious posture to recognize as it is displayed frequently and is often an indication that a human is on the move or engaging in a task.

Arms-Up Posture – Humans learn at a young age that they can attract another's attention by raising their hand and a robot should respond accordingly.

Kneeling Posture – Since many robots are significantly smaller than humans are, one must crouch or kneel to pass an object to, or take an object from the robot's gripper. A robot should therefore recognize a crouching human and be able to determine if the human is holding an object.

Object Posture – Held-objects were trained independently from the human. This increases model robustness and allows the robot to orient itself toward the object.

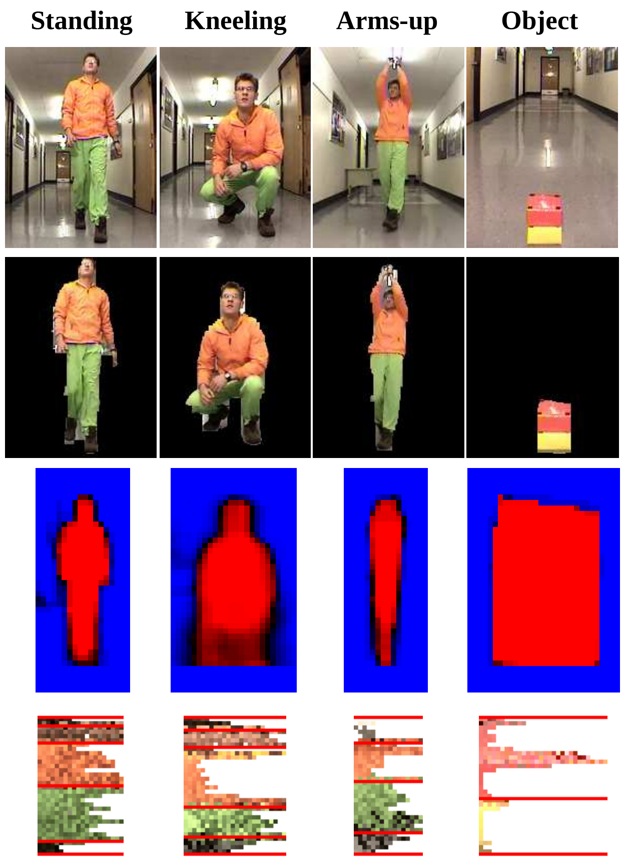


Fig. 4. Set of postures. First row: original images. Second row: detected foreground. Third row: shape model. Fourth row: color model.

A. Related Work in Visual Identification / Tracking

The identification and tracking of objects in a video feed is reasonably easy when a relatively static background can be maintained. In the simplest case, background pixels are modeled using a single video-frame, or they can be represented using Gaussian [31] or non-parametric distributions [13]. Models can be adaptive to slowly changing conditions [13], [24], [34], and even robust to smooth and linear camera movements [17]. In all these cases, pixels in subsequent frames are compared to the background model. If the pixel features (e.g., color, texture, motion) are inconsistent with the model, they are grouped and segmented as foreground.

Despite the success of background modeling techniques, they are unsuitable for use on a mobile robot in an uncontrolled environment. Camera movement is usually too complex to be stabilized by motion-modeling algorithms and the limitless variability of the shape, color, and texture of background features precludes the use of standard feature-based background models. Consequently, robotics applications typically use foreground-modeling techniques, which use object features to identify and track the object. Foreground modeling traditionally requires an offline acquisition period where values in the foreground models are assigned or trained. The robot is then switched to a tracking phase, where it searches each incoming image for a region that is sufficiently similar to the model.

A foreground modeling technique commonly used in robotic applications models an object as a colored blob. During acquisition, users must manually select a region of interest in the image. This process can be repeated on the same object under different lighting conditions and the algorithm will assign a range of values to represent the region. Schlegel *et al.* [29] automates this process by requiring users to stand directly in front of the robot's camera for an 'introduction'. The system then generates a two-dimensional color histogram (in red-green chrominance space) describing a rectangular region on the user's chest. Though this approach offers some convenience, it requires users to stand in a specific location during training and the resulting model can only be used to track colors corresponding to the user's shirt.

B. Overview of Approach

Our proposed method improves the convenience and speed of previous techniques, and is particularly suitable for robotic applications. Unlike [29], the training stage is fully automated, by using a background modeling technique to segment the person from the background. This allows the subject to move freely during the training stage. We also propose improvements to Schlegel's color modeling approach [29]. Instead of modeling a single region of color within the object, we identify and model multiple color-regions. Each region is modeled in three-dimensional RGB color-space, and is represented as a mixture of Gaussians. For increased robustness, our model also incorporates shape information without sacrificing speed.

The following sections describe how models are generated during training, how these models are used to locate and track users and their postures, and presents some implementation optimizations that allow for real time operation, even while searching for multiple models.

C. Training

Each posture from each person is separately trained during an off-line acquisition process from a stationary robot. Currently, training requires users to interact with a GUI. However, it would be relatively straight forward to automate the process in the future. Training requires a minimum of 5 seconds per posture and continues until the model stabilizes.

Before the object can be modeled, it must be segmented from the image. Segmentation is accomplished using an adaptive background modeling technique similar to the one described in [24]. Each pixel of the background is modeled as a Gaussian distribution in the RGB color space, with the mean

(μ_r,μ_g,μ_b) and standard deviation (σ_r,σ_g,σ_b). For each new pixel (x_r,x_g,x_b), the Gaussians are updated using a learning rate α, as follows:

$$\mu_i = \alpha x_i + (1-\alpha)\mu_i \tag{2}$$

$$\sigma_i^2 = max(\sigma_{min}^2, \alpha(x_i-\mu_i)^2 + (1-\alpha)\sigma_i^2) \tag{3}$$

where *i = r, g, b*.

The σ_{min}^2 term is introduced to prevent the variance from decreasing below a minimum value when the background remains constant for a long period.

Object segmentation is accomplished by comparing new pixels to the background model. A pixel is labeled as part of the foreground object if, for any value:

$$(x_i-\mu_i)^2 > (2\sigma_i)^2 \tag{4}$$

where *i = r, g, b*.

Segmented pixels are grouped together as connected-components to form a blob corresponding to the target object. Examples of a segmented image are shown in Fig. 4 (second row). As described in the next sub-sections, the segmented objects from each frame are combined to produce shape and color models.

1) Modeling Shape: Despite the fact that a human silhouette can be highly variable, there is enough regularity to warrant the inclusion of a shape-based model. The segmented foreground region from each frame is first normalized in terms of position and height, then quantized into a 32$\times w$ array of square blocks, where w is a function of the object's height to width ratio.

A map is then generated that contains (for each block) the likelihood of that block being a part of the foreground. Given N training frames, the probability at each block i is:

$$p_{shape}(i) = \frac{1}{N}\sum_{k=1}^{N} fg_k(i) \tag{5}$$

where $fg_k(i)$ is 1 if block i belongs to the foreground in frame k, and 0 if it belongs to the background.

High values in this map thus correspond to regions that are likely to be foreground and low values correspond to likely background regions. An illustration is provided in Fig. 4 (third row), where bright red regions correspond to foreground, blue regions correspond to background and black regions do not strongly correspond to either region.

2) Modeling Color: A common characteristic of human figures is that color remains relatively constant in the horizontal direction, while demonstrating more variability vertically. Variability usually occurs at the transitions between the hair and face, face and shirt, shirt and pants, and pants and shoes. It should also be noted that the relative size and location of these regions remain reasonably consistent even as a human moves.

To exploit the natural grouping of colors, our approach divides a target object into a vertical stack of horizontal color bands. We use 32 bands to represent our models. This number was selected because it provides sufficient resolution for representing the object, and because it allows for certain optimizations on a 32-bit system. Each band is modeled separately using a mixture of Gaussians.

During training, the foreground region is normalized in terms of height and quantized into the 32 bands. The pixel-values corresponding to each band are then accumulated into a histogram in RGB color space. At the end of the training period, the histogram is modeled as a mixture of Gaussians.

To produce a more robust representation of the object, adjacent bands are merged if their color distributions are sufficiently similar. Grouping begins with the most similar bands and continues with progressively less similar bands until the model is reduced to between one and six regions. Fig. 4 (forth row) shows the color distribution of each object. Colors are arranged so that the most dominant values are shown on the left and the least dominant on the right. The red lines delineate the regions after merging.

The resulting model contains information about the color composition, vertical location, and size of the prominent regions of color within an object and can be used for the detection and tracking of the object in a video sequence.

D. Detection and Tracking

Since humans tend to assume an upright posture, they will usually occupy a larger proportion of an image in the vertical direction than they will in the horizontal direction. This property simplifies an object search because it allows promising x-axis locations to be identified before considering y-axis locations.

Every pixel in the image is assigned a probability, which represents the likelihood that that pixel color is present in the foreground object. Pixels with color values matching the most prominent colors in the target model are assigned high probabilities, while colors not found in the model are assigned a probability of zero. For a given model, the probability that pixel i with color (x_r,x_g,x_b) belongs to the model is determined using all Gaussians in all bands of the color model:

$$p_{color}(i) = \frac{1}{N_{bands}}\sum_{bands}\frac{e^{-\left(\frac{(x_r-\mu_r)^2}{2\sigma_r^2}+\frac{(x_g-\mu_g)^2}{2\sigma_g^2}+\frac{(x_b-\mu_b)^2}{2\sigma_b^2}\right)}}{\left(\sqrt{2\pi}\sigma_r\right)\left(\sqrt{2\pi}\sigma_g\right)\left(\sqrt{2\pi}\sigma_b\right)} \tag{6}$$

The resulting probability values are then summed for every column of pixels to form a probability distribution with respect to the x-axis. The most prominent local maxima in this distribution are identified as promising x-axis locations. An example of such a probability mapping is shown in Fig. 5(a).

For each x-axis candidate, a 16-pixel wide column is defined, centered at the x-location. Pixels in every row of the column are assigned probability values, which represent the likelihood that the pixel's color is present in the corresponding object-color-region. Probabilities are determined as in Eq. 6, but with the sum computed over bands in that region. Probabilities are summed for every row, to find the y-axis distribution.

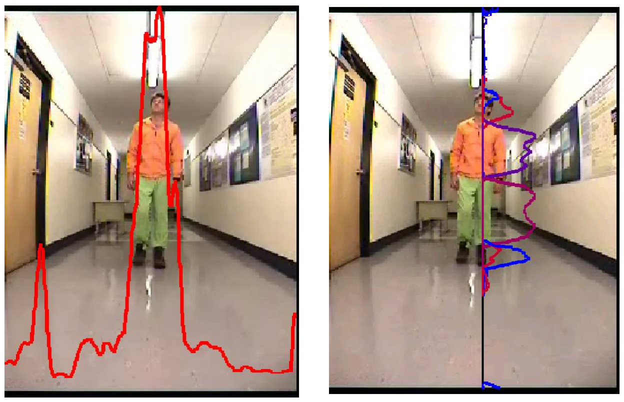

a) X-axis candidates b) Region probabilities

Fig. 5. Person detection and tracking.

Fig. 5(b) shows a model, which has been divided into four prominent regions. Probability mappings for each region are superimposed on the image.

As seen in Fig. 5(b), the probability maps tend to be high within the areas of their corresponding colors and the intersection between two probability mappings offer a good estimation of the border between adjacent colors. This effectively determines the location and vertical size of each region of color within an object.

In order to determine a final measure of similarity, the object-shape probability map is incorporated. As described previously, certain blocks of the shape-map will have a high probability of falling on the figure while other areas of the map (typically towards margins) will have a low probability. The shape-based probabilities are used to weight the color-based probabilities for each region, in order to produce a final similarity score.

E. *Efficiency*

It should be noted that although the model describing each object can comprise as many as six different color regions, each containing several Gaussian distributions, our implementation executes a one-time preprocessing step that compiles the Gaussians into a 3D array indexed by (R,G,B). With a single reference to this array, a probability measure can be obtained to determine the likelihood that that pixel is part of an object. If the probability of being present in an object was greater than zero, up to 6 additional arrays can be referenced to determine the probability that the pixel is contained in the object's sub-regions. These optimizations allow the tracking to be performed in real-time (20 frames/sec), even when 15 models are involved, and on a modest 1 GHz computer. Although the probability arrays require more memory than any other data structure, the color-space is sub-sampled to minimize the demands. The current implementation quantizes the color-space into 32$\times$32$\times$32 = 32,768 different colors, and requires a total of 64-bits to store the region and sub-region probability values. This requires about 262 KB of memory per model.

VI. Experimental Setup and Results

We validated our approach with a Pioneer 3DX mobile robot equipped with a SICK LMS-200 laser rangefinder, two rings of sonars, and a pan-tilt-zoom (PTZ) camera. For robot control we use the Player robot device interface [15]. Our validation consists of quantitative and qualitative evaluation for the *visual awareness* and the *robot control* modules.

A. *Validation of Visual Awareness*

Tracking and distance estimation. We tested the tracking component using an experiment where the robot was programmed to pursue a person. Computation of the person's distance from the robot was based on the camera calibration procedure described in [36] and on the assumption that the floor is flat and the person's feet are always on the floor.

For this trial, the robot accurately pursued a human-target, who alternated between forward and backward movement through a 100 meter-long hallway. Frames from the robot's camera are shown in Table I, where the green rectangle and the green outline have been generated by the detection and tracking module. In order to assess the accuracy of distance estimation, the computed positions at each displayed frame are shown together with those obtained from the robot's laser rangefinder (ground truth).

It is worth emphasizing that the experiment illustrates the ability to perform real-time tracking and distance estimation for a moving target while the robot (and its camera) is also moving.

TABLE I
Tracking distance estimation.

100	200	300	400	500	600	700	800	900
7.3	8.3	7.3	7.3	6.4	5.8	5.8	5.8	7.0
7.7	8.2	7.3	7.3	6.8	5.6	6.1	6.3	6.6

Top Row: Frame number. *2nd row*: Frame image. *3rd row:* Estimated distances (meters). *4th row*: Ground truth (laser) distances.

Posture recognition – qualitative validation. Fig. 6 shows the recognition of postures in the presence of multiple persons. The system was trained on postures from 3 users (which are correctly recognized), while the 2 unknown users are (correctly) ignored. This experiment shows that the approach can robustly detect and track multiple postures from multiple users, while ignoring irrelevant persons in a possibly crowded environment.

Posture recognition – quantitative validation. To quantitatively estimate the recognition accuracy, we trained the system on 5 users with 3 postures each. After training, subjects were asked to display each posture for about 30 seconds, while they moved through the camera's field of view. Measurement of correct posture frequency commenced 10 frames after each change in posture and was continued for 200 frames. In this

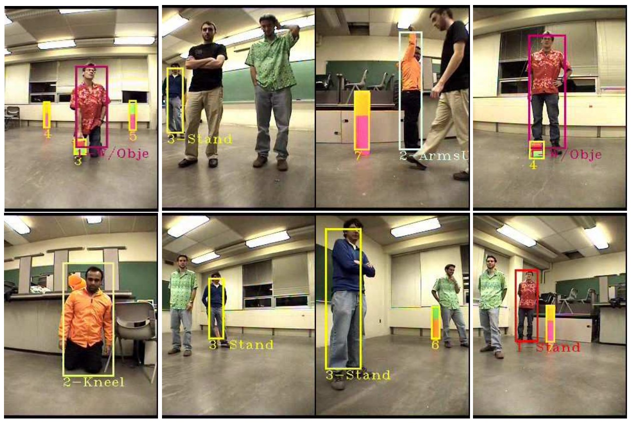


Fig. 6. Posture recognition – qualitative validation. Modeled users: red, orange, blue shirt. Unknown users: black, green shirt.

scenario, both training and testing was conducted from a stationary robot. The robot was not moved between training and testing periods. Table II shows the percentage of frames in which the algorithm correctly recognized the postures.

TABLE II
POSTURE RECOGNITION – QUANTITATIVE VALIDATION

	User1	User2	User3	User4	User5
Standing	92.5%	91%	99.5%	100%	100%
Kneeling	97%	98%	99%	100%	100%
Arms-up	95.5%	100%	100%	99%	100%

Postures were trained and then tested for 200 frames. Table displays percentage of frames that contained the correctly identified posture.

B. Validation of Robot Control

We performed the robot control experiments in a classroom and an office building setup, as shown in Fig. 7. In these experiments two different users interacted with the robot. For each user, the robot was trained to detect the following postures: *standing*, *arms-up*, *kneeling (with object)*, and *kneeling (without object)*, using the method described in Section V.

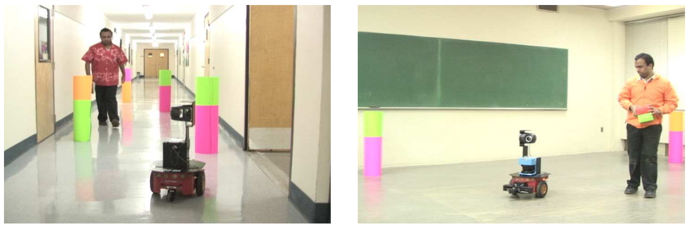


Fig. 7. Experimental setup for real-world environment

The robot's set of behavior primitives consists of: laser obstacle avoidance, attraction to a goal object, attraction to unoccupied space, attraction to walls, rear sonar obstacle avoidance, tangent wall follow, circular avoid, and pick up and drop objects. The behaviors produce a motor command output in the form of a vector in the robot's coordinate system. With these behaviors we created a set of fusion primitives and task controllers, which constitute our robot's repertoire of services. As mentioned previously, we used the method described in

TABLE III
ASSOCIATION BETWEEN USER POSTURES AND TASKS.

Posture ID	**Kneeling**	**Arms Up**	**Kneeling with object**
User 1 Task Code (Priority) Sequence	T1 (Medium) Drop object → Visit(Target1) → Visit(Target3) → Visit(Target2) → Visit(Target3) →	T2 (High) Visit(Target3) → Visit(Target1) → Visit(Target3) →	T3 (Medium) Take object → Visit(Target3) → Visit(Target2) → Visit(Target3) →
User 2 Task Code (Priority) Sequence	T4 (High) Drop object → Visit(Target1) → Visit(Target3) → Visit(Target2) →	T5 (Medium) Visit(Target2) → Visit(Target3) → Visit(Target1) →	T6 (High) Take object → Visit(Target1) → Visit(Target3) → Visit(Target2) →

The identification of postures triggers the robot to visit a series of targets as outlined in this table. Priority levels dictate which sequences are completed first.

[23] to train the robot's controller. The robot's tasks consist of a series of target reaching and object transport duties. To complete a task, a robot must visually locate and drive to all targets in the correct succession. We selected these duties because they are similar to what a service delivery robot might encounter and because they can be easily achieved by our current platform, though our approach should work well with other duties or platforms. Each of these tasks has a given priority and is associated with one of the users' posture, as shown in Table III. The *visit target* component of each task is a metabehavior, whose goals are achieved when the robot is at a specified distance with respect to the target.

In addition to the above tasks, the robot is equipped with a wandering task (T0), which has a *low* priority and is executed as long as the robot has no requests to service. The standing posture is not associated with any task, but rather serves as a trigger from the *visual detection* module that a user is in vicinity.

In our experiments the two users requested services from the robot over an extended duration, in order to demonstrate the main features of our approach: 1) ability to detect the presence of multiple people, 2) ability to handle multiple requests, 3) ability to handle task interruptions and 4) extended robot autonomy.

The behavior of the robot equipped with these capabilities follows a natural type of social interaction. Once the *detection* and *control* modules are started, our robot is in complete control of its own capabilities and decides for itself what activities it needs to perform. Immediately upon starting, the robot begins wandering, waiting for any requests for services from human users. If the robot detects a user (through his/her standing posture), the robot briefly interrupts its task, and slows down until the user approaches it at a given distance. If within several seconds no new postures are detected (i.e., no requests from the user), the robot resumes its task, ignoring that user for some predefined period, unless the user later displays a non-standing posture. This later step is needed in order to avoid infinite loops of attending to a passer-by user. In the case when the user displays a non-standing posture, the

TABLE IV
SEQUENCE OF TASK REQUESTS FOR THE FIRST SCENARIO.

Posture detected	**Request**	**Robot action**	**Current task**	**Task queue**
None	None	Robot starts off wandering	T0	NULL
User1 kneeling	T1	Switch to T1	T1	T0
None	None	Robot finishes T1, switches to T0	T0	NULL
User1 w/ object	T3	Switch to T3	T3	T0
User1 arms up	T2	Switch to T2	T2	T3, T0
User2 kneeling	T4	No switch, continues with T2	T2	T4, T3, T0
None	None	Robot finishes T2, switches to T4	T4	T3, T0
None	None	Robot finishes T4, switches to T3	T3	T0
None	None	Robot finishes T3, switches to T0	T0	NULL
User2 arms up	T5	Switch to T5	T5	T0
User2 w/ object	T6	Switch to T6	T6	T5, T0
User1 kneeling	T1	No switch, continues with T6	T6	T1, T5, T0
None	None	Robot finishes T6, switches to T1	T1	T5, T0
None	None	Robot finishes T1, switches to T5	T5	T0
None	None	Robot finishes T5, switches to T0	T0	NULL

robot turns and approaches the person (if it has not done so already). If the posture is detected for more than a few seconds, this is an indication that a new task has been requested. To avoid multiple detections, the robot ignores a person that had just requested a new task for a predefined period, and also ignores requests for tasks that are either in the queue or that are currently being executed. Currently, the robot produces various motor sounds to provide users with feedback that a task request was received. In future implementations we will include an auditory recognition module and speech synthesis capabilities, such that the robot could provide feedback through verbal communication.

We performed experiments for two different task requests scenarios, with each scenario repeated four times. We chose to use the same sequence of requests for each scenario, in order to establish a baseline for evaluation, both from the perspective of task execution (the *control module*) and from the perspective of the posture recognition (the *visual detection* module). For each scenario we used a different task priority scheme, as described below. In tables IV and V we show the sequence of tasks that were requested during the two scenarios, and we also indicate the correct robot action response for further validation with the experimental results.

Results from Scenario 1. Each run took approximately 20 minutes. During the course of the experiment, the robot correctly identified the postures (and thus the requests) in every case but one. It took the correct decisions regarding which tasks to perform (given priorities and incoming order) and it correctly finished executing all tasks. The only error occurred in the fourth run, in which the robot detected a request for task 4 instead of task 1. During this run, the priority method used was to process tasks with highest priority first; for tasks of equal priority, a LIFO (last-in-first-out) method was used. Fig. 8 (a) shows the order in which the robot received and serviced the requests during the four runs for scenario 1. The moments of time when the requests are received are indicated by red squares. Green squares indicate the time of task completion. Task lines that are interrupted without a marker indicate switches from the current task to a task of higher priority. Requests that are not serviced immediately after they are received belong to tasks that are of lower or equal priority with the task currently being executed. The plots show the moments of time when these tasks are serviced from the robot's task queue.

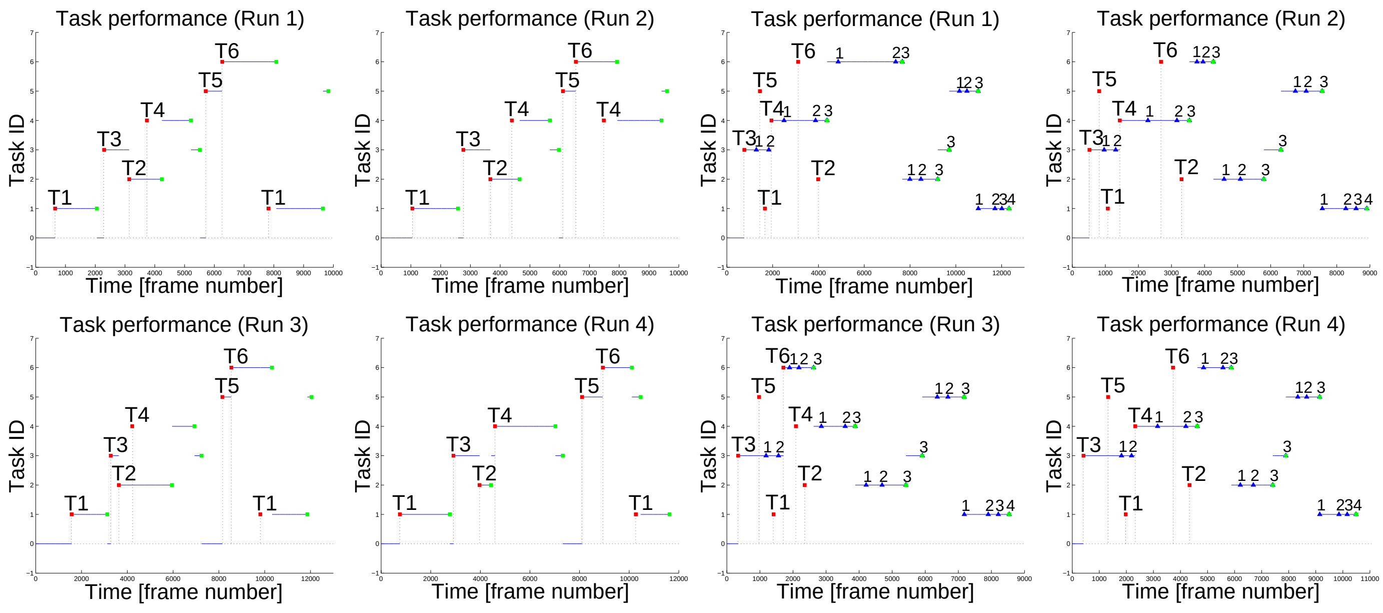
Results from Scenario 2. Each run took approximately 20 minutes. During the course of the experiment, the robot correctly identified the postures (and thus the requests), took the correct decisions regarding which tasks to perform (given priorities and incoming order) and it correctly finished executing the tasks. The only difference in execution occurred in the third run, in which the users requested task 6 before task 4. However, this being a change in scenario, the robot correctly identified and serviced the requests. During this run, the priority method used was to process tasks with highest priority first; for tasks of equal priority, a FIFO (first-in-first-out) method was used. Fig. 8 (b) shows the order in which the robot received and serviced the requests during the four runs for scenario 2. For these experiments we also recorded the progress of the robot in each task, to demonstrate that the robot is able to keep track of what parts of each task have been finalized. As a result, the figure shows that if the robot is interrupted in the middle of a task, upon resuming its execution the robot will continue from the point where the task was interrupted, instead of performing it from the start. This behavior is shown in the execution of task 3: the task is interrupted after performing its first two steps; when the robot resumes the task it only performs the last (third) step to complete it, showing that it remembered what parts of the task have already been done.

VII. CONCLUSION

In this paper we propose a framework for developing robot assistants that addresses two key issues of human-robot interaction: the ability to detect and recognize multiple users,

TABLE V
SEQUENCE OF TASK REQUESTS FOR THE SECOND SCENARIO.

Posture detected	**Request**	**Robot action**	**Current task**	**Task queue**
None	None	Robot starts off wandering	T0	NULL
User1 w/ object	T3	Switch to T3	T3	T0
User2 arms up	T5	No switch, continues with T3	T3	T5, T0
User1 kneeling	T1	No switch, continues with T3	T3	T5, T1, T0
User2 kneeling	T4	Switch to T4	T4	T3, T5, T1, T0
User2 w/ object	T6	No switch, continues with T4	T4	T6, T3, T5, T1, T0
User 1 arms up	T2	No switch, continues with T4	T4	T6, T2, T3, T5, T1, T0
None	None	Robot finishes T4, switches to T6	T6	T2, T3, T5, T1, T0
None	None	Robot finishes T6, switches to T2	T2	T3, T5, T1, T0
None	None	Robot finishes T2, switches to T3	T3	T5, T1, T0
None	None	Robot finishes T3, switches to T5	T5	T1, T0
None	None	Robot finishes T5, switches to T1	T1	T0
None	None	Robot finishes T1, switches to T0	T0	NULL



a) Scenario 1. b) Scenario 2.

Fig. 8. Robot task execution and request identification for Scenarios 1 (a) and 2 (b). Red squares indicate new requests. Green squares indicate task completion. Blue triangles indicate subtask completion. Numbers above blue triangles represent subtask ID.

and extended interaction with users and the environment. Our detection mechanism is built on visual capabilities that allow the robot to identify multiple users, with multiple postures, in real-time, in dynamic environments where both the robot and human users are moving. Extended human-robot interaction is supported by a novel control architecture which allows a robot to accommodate multiple user requests and task interruptions and it enables the representation of complex, sequential and hierarchical robot tasks. The architecture provides the robot with flexibility in dealing with multiple users, such as to accommodate multiple user requests and task interruptions, over extended periods. We validated our approach on a Pioneer 3DX mobile robot, performing service tasks in a real-world environment. Our experimental results demonstrate the robot's ability to engage in interactions in a natural way, to deal with multiple users, and to be constantly aware of their surroundings, thus advancing service robotics toward deployment of robots into the real world.

ACKNOWLEDGMENT

This work was supported in part by NSF CAREER Award IIS-0546876 to Monica Nicolescu and by the ONR Award N00014-06-1-0611.

REFERENCES

[1] Omron, is this a real cat? a robot cat you can bond with like a real pet. NeCoRo is Born, News Release, Oct 2001.
[2] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *PAMI*, 28(1):44–58, January 2006.
[3] R. Ambrose, H. Aldridge, R. Burridge, W. Bluethman, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. Robonaut: Nasa's space humanoid. *IEEE Intelligent Systems Journal*, Aug 2000.

[4] R. C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *IEEE Conference on Robotics and Automation, 1987*, pages 264–271, 1987.

[5] R. Barea, L. Boquete, M. Mazo, E. Lpez, and L. M. Bergasa. Electrooculgraphic guidance of a wheelchair using eye movements codification. In *International Conference on Field and Service Robots*, Helsinki, Finland, June 2001.

[6] R. Bischoff and V. Graefe. Hermesa versatile personal robotic assistant. *Proceedings of the IEEE*, pages 1759–1779, Nov 2004.

[7] M. Boada, R. Barber, and M. Salichs. Visual approach skill for a mobile robot using learning and fusion of simple skills. *Proceedings of the IEEE*, pages 157–170, 2002.

[8] C. Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42(3–4):167–175, 2003.

[9] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, , and D. Mulanda. Tutelage and collaboration for humanoid robots. *International Journal of Humanoid Robotics*, 1(2), 2004.

[10] C. Breazeal, A. Edsinger, , P. Fitzpatrick, and B. Scassellati. Active vision for sociable robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 31(5):443–453, 2001.

[11] C. Contest. First international cleaning robot contest, lausanne, switzerland, Oct 2002.

[12] J. Drury, J. Scholtz, and H. Yanco. Awareness in human-robot interaction. In *Proc., IEEE Conf. on Systems, Man and Cybernetics*, 2003.

[13] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90:1151–1163, 2002.

[14] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42:143–166, 2003.

[15] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proc., the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[16] H. Httenrauch, A. Green, M. Norman, L. Oestreicher, and K. S. Eklund. Involving users in the design of a mobile office robot. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34(2):113–124, 2004.

[17] J. Kang, I. Cohen, and G. Medioni. Continuous tracking within and across camera streams. In *Computer Vision and Pattern Recognition*, pages 267–272, 2003.

[18] H. Krebs, B. Volpe, M. Aisen, and N. Hogan. Increasing productivity and quality of care: robot-aided neurorehabilitation. *Journal of Rehabilitation Research and Development*, 37(6), 2000.

[19] P. Maes. How to do the right thing. *Connection Science Journal, Special Issue on Hybrid Systems*, 1(3):291–323, 1990.

[20] F. Michaud and S. Caron. Roball, the rolling robot. *Autonomous Robots*, 12(2):211–222, 2002.

[21] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *ECCV02*, page III: 666, 2002.

[22] R. Murphy. Human-robot interaction in rescue robotics. *IEEE Systems, Man and Cybernetics Part C: Applications and Reviews, special issue on Human-Robot Interaction*, 34(2), May 2004.

[23] M. Nicolescu, C. Jenkins, and A. Olenderski. Learning behavior fusion estimation from demonstration. In *IEEE Intl. Symp. on Robot and Human Interactive Communication, (RO-MAN 2006)*, pages 340–345, Hatfield, United Kingdom, Sep 2006.

[24] M. Nicolescu, G. Medioni, and M.-S. Lee. Segmentation, tracking and interpretation using panoramic video. In *IEEE Workshop on Omnidirectional Vision*, pages 169–174, 2000.

[25] M. N. Nicolescu and M. J. Matarić. A hierarchical architecture for behavior-based robots. In *Proc., First Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 227–233, Bologna, Italy, July 2002.

[26] I. Nourbaksh. An affective mobile robot educator with a full-time job. *Artificial Intelligence*, 114(1–2):95–124, 1999.

[27] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Special issue on Socially Interactive Robots, Robotics and Autonomous Systems*, 42(3-4):271–281, 2003.

[28] P. Pirjanian. Behavior coordination mechanisms - state-of-the-art. Tech Report IRIS-99-375, Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, California, 1999.

[29] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, and R. Worz. Vision-based person tracking with a mobile robot. In *British Machine Vision Conference*, pages 418–427, 1998.

[30] K. U. Scholl, V. Kepplin, K. Berns, and R. Dillmann. Autonomous sewer inspection: Sensorbased navigation. In *International Conference on Field and Service Robots*, Helsinki, Finland, June 2001.

[31] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on PAMI*, pages 747–757, 2000.

[32] T. Tashima. Interactive pet robot with emotion mode. In *Proceedings of the 16th Annual Conference of the Robot Society of Japan*, 1998.

[33] A. Thomaz, G. Hoffman, and C. Breazeal. Experiments in socially guided machine learning: Understading human intent of reward/punishment. In *Human-Robot Interaction Conference*, 2006.

[34] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Intl. Conf. on Computer Vision*, pages 255–261, 1999.

[35] J. Yuh, T. Ura, and G. Bekey. *Underwater Robots*. Kluwer Press, 1996.

[36] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Intl. Conf. on Computer Vision*, pages 666–673, 1999.