A Control Architecture for Long-Term Autonomy of Robotic Assistants

Christopher King, Xavier Palathingal, Monica Nicolescu, Mircea Nicolescu

email: {cjking, xavier, monica, mircea}@cse.unr.edu
Department of Computer Science and Engineering
University of Nevada, Reno *

Abstract. A major challenge in deploying service robots into the real world is to design a framework that provides effective, long-term interactions with people. This includes interacting with people in a natural way, dealing with multiple users, and being continually aware of the surroundings. This paper proposes a robot control architecture that addresses these issues. First, it enables the representation of complex, sequential, and hierarchical robot tasks, in a behavior-based framework. Second, it provides a robot with the flexibility to deal with multiple requests and interruptions, over extended periods. Third, it uses a visual awareness mechanism to recognize users and to identify their need for robot interaction. We demonstrate our approach on a Pioneer 3DX mobile robot, performing service tasks in a real-world environment.

1 Introduction

A major challenge in designing robots for service or assistive applications is to enable a natural interaction between robots and non-technical users, while ensuring long-term, robust performance [1]. Robots have traditionally been developed to operate in controlled environments and are programmed to perform tasks in a highly structured and sequential manner. These robots are usually "blind" to other agents in the environment and adapt poorly to changing conditions. The limitations that generally prevent these robots from operating in more realistic domains are their lack of awareness, flexibility, and long-term autonomy.

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 *visual-awareness* necessary for them to monitor their surroundings and detect when other social agents have the need for their interaction. Our architecture also provides the means for *long-term autonomy* by enabling robots to manage a large repertoire of tasks over extended periods. Additionally, our system is designed for realistic assistive applications, where multiple people are simultaneously competing for the robot's assistance.

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

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

and robustness of task representations, 2) long term interactions with multiple users, and 3) awareness of the environment and other agents.

The remainder of the paper is structured as follows: Section 2 presents our interactive framework, Section 3 describes our control architecture, Section 4 discusses our vision-based perceptual approach and Section 5 describes the experimental setup and results. We present our conclusions in Section 6.

2 Interactive Framework

This work is aimed at creating a framework that provides robots with the ability to operate in typical service or assistive application environment. This requires robots to be directed easily by non-technical operators, and function in the presence of multiple users.

Vision-based perception can provide a wealth of information regarding the robot's environment and of other agents within the environment. In the case of human-human interaction, one person can frequently identify the needs of another simply by observing them. Ideally, a service robot should make similar deductions. To this end, a posture-based control paradigm is used. As will be described in Section 4, robots are trained to recognize various postures, which are associated with different tasks. These associations can have a logical relationship (e.g. if a person is observed with an object, the robot should approach the human and accept the object), or may be more symbolic (e.g. if a person is observed with raised hands, a predefined series of actions are performed. In either case, a non-technical user should be able to easily learn how to interact with and receive services from the robot.

A service robot will likely have to perform in the presence of multiple users, where one user may solicit a service while the robot is engaged in another task. To respond accordingly, the robot should interrupt its current activity, detect the new request, and determine appropriate action. Our framework enables this functionality using linked *awareness* and *control* modules. The *awareness module* identifies known users and postures. This information is relayed to the *control module*, which determines the robot's action. Currently, each posture is associated with a task (robot service) that can have *low*, *regular* or *high* priority. When a posture is detected, the robot will perform the associated task, only if the priority of the new task exceeds that of any current activity. The task with the lower priority will be suspended and stored to a priority-based queue. Lower-priority tasks will be resumed when higher-priority tasks are completed. Our architecture provides the flexibility of using different priority queue strategies.

Prioritized task switching resembles human decision-making behavior and is a logical addition to the service robot domain. While people perform this activity switching with ease, robots are presented with the difficulty of maintaining the status of current tasks during interruption, such that the task can be resumed from the same point later. The control architecture proposed in this paper use a set of representations that allow the robot to naturally recover from interrupted tasks without the need to explicitly store any additional state information.

3 Control Architecture

The architecture proposed in this paper is motivated by the Behavior-Based Control (BBC) paradigm, which is popular in robot control. We propose to use this paradigm to **1)** enable the use of both *command arbitration and fusion* within a single representation and **2)** allow the encoding and robust execution of sequential and hierarchical tasks. Historically, the two main action selection mechanisms of *arbitration* and *fusion* have been employed separately in robot control [2], which limits the range of executable tasks. By recognizing the ability of *arbitration* to encode temporal sequences and 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 [3].

3.1 Fusion Primitives

Our controllers are built from two components: *behavior primitives* (BPs) and a *fusion primitive* (FP), which through the combination processes described below result in controllers in the form of *behavior networks* [4].

The BPs express basic capabilities that allow a robot to react to immediate environmental conditions. If input received from the robot sensors meets the preconditions to make a particular BP *active*, then the BP sends an appropriate action to the actuators. For example, an *obstacle-avoidance* BP becomes *active* when sensors detect an object that obstructs the robot.

The *active/not active* status of all BPs 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 a 1 (active) or a 0 (not active) for each BP. It is theoretically possible for n BPs to produce 2^n BACs, though many combinations are never seen, and this number is usually much smaller.

The FP linearly combines the vectors produced by all active BPs to produce a control vector that moves the robot toward the direction of highest urgency. BPs are combined using a weighting scheme that modulates the influence each BPs has on the final vector. The set of weights used in the BP summation is determined by the BAC table, as shown in Figure 1. Each BAC entry represents a different set of weights and can be indexed using the n-bit BAC value.

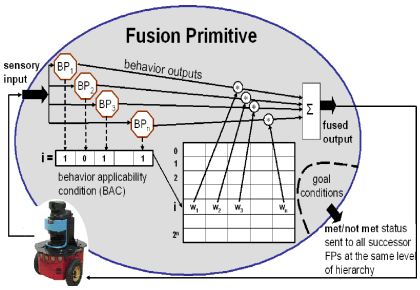


Fig. 1. Fusion primitive.

At each timestep t, each BP B_i provides a response output vector v_i^t, which represents a desired heading for the robot. The FP's output is a linear combination of the vectors $[v_1^t \cdots v_n^t]$, according to the BAC superposition weights $W^t = [w_1^t \cdots w_n^t]$:

$$V_r^t = \sum_{i=1}^{n} w_i^t v_i^t \tag{1}$$

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. For example, in a *target reaching* task, the robot may be influenced by *corridor-follow, target-follow* and *avoid-obstacle* behaviors in the presence of an obstacle, while only *target-follow* would be active in an open space. Inferring the fusion weights is a challenging task requiring time-consuming fine-tuning. We used a method of learning weights through human-provided demonstration [5].

3.2 Hierarchical Task Representations

With fusion primitives alone, a controller can only encode *flat representations* of tasks using sequencing of fusion primitives. This does not have the modularity needed to allow more complex tasks to be created from existing ones. We enable this higher-level of representation by grouping fusion primitives into *behavior networks*, which can be nested to allow hierarchical representations of tasks. In these networks, links between components represent task-specific precondition-postcondition dependencies, which provide a simple way to represent complex activities (Figure 2).

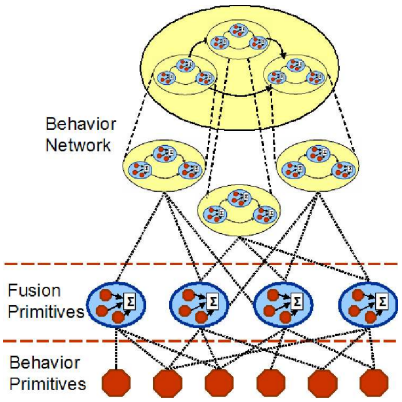


Fig. 2. Generic hierarchical task.

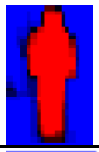
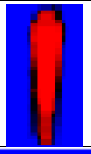
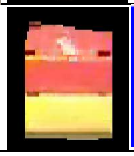
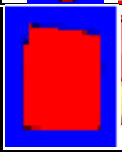
The term *metabehavior* is used to describe both fusion primitives and nodes of a behavior network, as both have similar functions in the network. Each metabehavior encapsulates information about the behavior's preconditions and goals (postconditions). These conditions are continuously monitored to ensure proper task execution. The only difference between a behavior network node and a fusion primitive is that the network node activates underlying metabehaviors, while a fusion primitive activates only its component primitive behaviors. When a behavior network node becomes active, its underlying components are enabled, and the subnetwork becomes the current "network" to be executed. Upon completion, the behavior network node updates its goal status accordingly. Successor behaviors will detect the achievement of the goal and a new network node will execute. To perform complex tasks, the robot activates the metabehavior at the task's topmost level and the activity propagates through the task's steps.

An advantage of this architecture is that it is adaptive to both favorable (inadvertently satisfying a goal) and unfavorable (undoing a goal) changes. Since the behavior's pre and post-conditions are continuously monitored, the active behavior is driven by environmental state, thus providing a sense of "awareness" about task progress. The representation also allows interruption to occur without additional modifications. When a task is interrupted, the behaviors preserve the current status of execution until the robot can return to the task. The behaviors' continuous grounding in sensory information allows the task to be performed correctly, even when environmental conditions changed during suspension.

4 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. Postures description and examples are displayed in Table 1 (first and second column).

Table 1. Set of Postures. Column 1: Posture description. Column 2: Sample video frame. Column 3: Segmented image. Column 4: Shape model. Column 5: Color model.

Posture Type/Description	Image	Foreground	Shape	Color
The Standing Posture A good posture to recognize. It is displayed frequently and may indicate that the human is on the move or engaging in a task.				
Kneeling Posture Given the robot's size, humans must crouch or kneel to pass objects to and from the robot. A robot should therefore recognize a crouching human.				
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.				
Object Posture Held-objects were trained independently from the human. This increases model robustness and allows the robot to orient itself toward the object.				

4.1 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. The background features can be modeled using intensities, Gaussian distributions, non-parametric distributions [6], etc., which all allow objects that do not match the stored models to be segmented as foreground. These techniques can be robust to gradual changes in the background [6] or even smooth and linear camera movements [7], but are still unsuitable for use on a mobile robot. Robot camera movements are 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, foreground-modeling techniques are generally the norm for robotics. The most common of

these approaches models objects as a colored blob. While most of the existing techniques can be used to detect multiple colored blobs, they usually do not enforce the relative positioning or relative size of the blobs. For example a person with a red hat and blue shirt would likely appear to be the same as a person with a blue hat and red shoes. Also, these methods rarely incorporate shape due to efficiency constraints. The method we developed improves the convenience of previous techniques, enforces relative position and size of modeled objects, and incorporates shape information without sacrificing speed [8].

4.2 Training

For our demonstration, the robot was trained to recognize three different postures from two different people, as well as a colored box (Table 1 second column). The training process required a person to present each pose to a static robot for about fifteen-seconds. During this period, an adaptive background modeling technique [9] was used to segment the foreground object (Table 1 third column). Segmented objects were normalized in terms of position and height and were used to form models of the object's shape and color.

Although human silhouette can be highly variable, there is enough regularity to warrant a shape-based model. This was done by dividing the segmented and normalized foreground object into a matrix of square blocks. A map is then generated that contains the likelihood that each block is part of the foreground or background (Table. 1 forth column (Red corresponds to high foreground probability and blue to low probability)).

Given N training frames, the probability at each block i is:

$$p_{shape}(i) = \frac{1}{N} \sum_{k=1}^{N} fg_k(i) \tag{2}$$

$fg_k(i)$ equals 1 if block i belongs to foreground in frame k, and 0 otherwise.

Color models were developed to exploit the fact that human figures usually contain coloration that is less variable in the horizontal direction than in the vertical direction. Variability usually occurs at the transitions between face and shirt, shirt and pants, etc. Also, the relative size and location of these regions remain reasonably consistent even as a human moves. We recorded this natural grouping by dividing the object into a vertical stack of horizontal color bands, where the size and position of each band was determined by the object's colors. Bands frequently corresponded with the face, shirt, pants, and shoes as seen in Table 1 (fifth column). The color-values corresponding to each band were modeled as a mixture of Gaussians in three-dimensional RGB color-space. In addition to color composition, the model contained information about vertical location, and size of the regions.

4.3 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

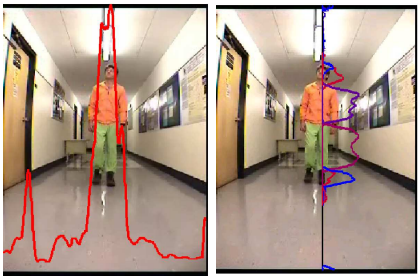
horizontal direction. This property simplifies an object search because it allows promising x-axis locations to be identified before considering y-axis locations.

For both x-axis and y-axis searches, pixels in the image were assigned a probability that represented the likelihood that the pixel's color was present in the foreground object space. 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:

$$p_{color}(i) = \frac{1}{N_{bands}} \sum_{bands} \frac{e^{-\left(\frac{(x_r-\mu_r)^2}{2\sigma_r^2}+\frac{(x_g-\mu_g)^2}{2\sigma_g^2}+\frac{(x_b-\mu_b)^2}{2\sigma_b^2}\right)}}{\left(\sqrt{2\pi}\sigma_r\right)\left(\sqrt{2\pi}\sigma_g\right)\left(\sqrt{2\pi}\sigma_b\right)} \tag{3}$$

For the horizontal search, a summation was made for the resulting probability values in each column of pixels. Local maxima were then recorded as likely positions along the x-axis (Figure. 3 (a)).

A vertical search was conducted on the region surrounding every probable x-axis location using a similar technique. This would yield locations along the y-axis that had a high probability of matching the vertical position and coloration of the associated model (Figure. 3 (b)).



a) X-axis **b)** Region

Fig. 3. X-axis/Region probabilities.

The object-shape probability map is used for a final measure of similarity. Certain blocks of the shape-map will have a high probability of falling on the figure while other areas 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. Regions corresponding to high scores are considered foreground (Figure. 4 (b)).

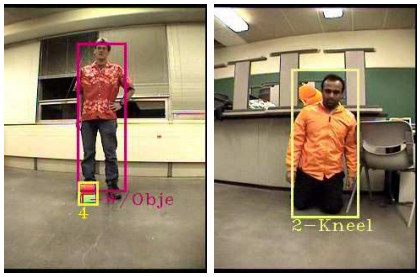


Fig. 4. Detection.

4.4 Efficiency

It should be noted that although every color region is represented by many Gaussians and although the image is searched for each posture of each object, our implementation allows this to be done quite efficiently. We use 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 a particular pixel is part of an object. This optimization allows tracking to be performed in real-time (20 frames/sec) on a modest 1 GHz computer, even when tracking over a dozen models.

5 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 [10]. We performed the robot control experiments in a classroom environment (Figure. 5). In these experiments, two different users interacted with the robot using three different postures: *standing*, *arms-up*, *kneeling (with object)*, and *kneeling (without object)*.

Fig. 5. Experimental environment

The robot's behavior primitives consist 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. Using these behaviors we created a set of fusion primitives and task controllers, which constitute our robot's repertoire of services. The robot's tasks involve a series of visit-target and object-transport tasks, representative for a service robot's potential delivery scenarios. Each of these tasks has a given priority and is associated with one of the users' posture, as shown in Table 2. The *visit-target* component of each task is a metabehavior, whose goals are achieved when the robot is a particular distance with respect to the target.

Table 2. Task requests. *User row*: The user # and posture type (Au=Arms Up, Kn=Kneel, Ob=Object). *Request row*: The requested task. *Action row*: The task pushed to or popped from the queue. *Queue row*: The queue contents. *Current row*: The task currently being executed ('*' indicates that the task was executed to completion).

User Posture	–	1 Ob	2 AU	1 Kn	2 Kn	2 Ob	1 AU	–	–	–	–	–	–
Request Priority	T0 low	T3 med	T5 med	T1 med	T4 high	T6 high	T2 high						
Action		push T0	push T5	push T1	push T3	push T6	push T2	pop T6	pop T2	pop T3	pop T5	pop T1	pop T0
Queue		T0	T5, T0	T5, T1, T0	T3,T5, T1, T0	T6,T3, T5,T1, T0	T6,T2, T3,T5, T1,T0	T2,T3, T5,T1, T0	T3,T5, T1, T0	T5, T1, T0	T1, T0	T0	
Current	T0	T3	T3	T3	T4	T4	T4*	T6*	T2*	T3*	T5*	T1*	T0

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

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

Upon starting, the robot begins wandering, while waiting for requests. If the robot detects a user (through the standing posture), the robot interrupts its task and reduces its speed. 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 of time, unless the user later displays a non-standing posture. This later step is needed to avoid infinite loops of attending to a passer-by user. When the user displays a non-standing posture for a sufficient duration, the robot queues the request and provides audible confirmation. The robot ignores a person for a short period after they issue a request, and ignores requests for tasks that are in the queue or are currently being executed.

We performed experiments for two different task request scenarios, with each scenario repeated four times. We use the same sequence of requests for each scenario to establish a baseline for evaluation, both from the perspective of task execution (the *control module*) and posture recognition (the *visual awareness module*). We used different priority schemes for each scenario.

Results. In both scenarios, the robot correctly identified the postures (and thus the requests), made the correct decisions regarding priorities, and correctly executed the tasks. All runs took approximately 20 minutes. In scenario 1, the only error occurred in the fourth run, where the robot detected a request for task 4 instead of task 1. In the third run of scenario 2, the user made the mistake of requesting task 6 before task 4. However, this being a change in scenario, the robot correctly identified and serviced the requests. In both scenarios, the robot processed tasks with highest priority first. For scenario 1, tasks with equal priority were processed using LIFO (last-in-first-out), and for scenario 2, FIFO (first-in-first-out) was used. Graphical results are shown for scenario 2 in Figure. 6. The Red squares mark the time when requests are received and green squares represent task completion. Task progress is shown by incremented numbers. When an interrupted task is resumed, these numbers show that the robot continues the task from where it left off.

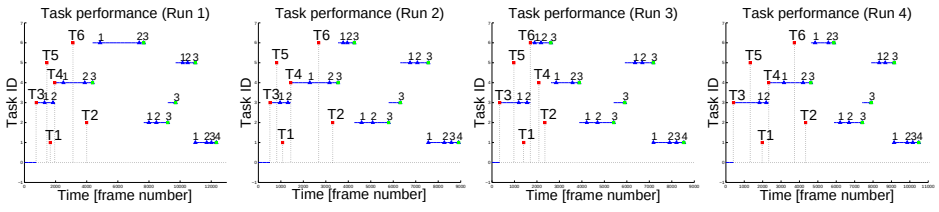


Fig. 6. Robot task execution and request identification. Red squares: New requests. Green squares: Task completion. Blue triangles: Subtask completion with ID.

6 Conclusion

In this paper we propose a framework for developing robot assistants that addresses two key issues of human-robot interaction: *awareness* of the environment and other agents, and *long-term interaction* with multiple users. Our awareness 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. Long-term human-robot interaction is supported by a novel control architecture that 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.

References

1. Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42:143–166, 2003.
2. Paolo 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.
3. Ronald 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.
4. Monica N. Nicolescu and Maja 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.
5. Monica Nicolescu, Chad Jenkins, and Adam 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.
6. 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.
7. J. Kang, I. Cohen, and G. Medioni. Continuous tracking within and across camera streams. In *Computer Vision and Pattern Recognition*, pages 267–272, 2003.
8. Christopher King, Xavier Palathingal, Monica Nicolescu, and Mircea Nicolescu. A vision-based architecture for long-term human-robot interaction. In *Proc., the IASTED Intl. Conf. on Human Computer Interaction*, 2007.
9. 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.
10. Brian Gerkey, Richard T. Vaughan, and Andrew 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.