# 38. Behavior-Based Systems

Maja J. Matarić, François Michaud

Nature is filled with examples of autonomous creatures capable of dealing with the diversity, unpredictability, and rapidly changing conditions of the real world. Such creatures must make decisions and take actions based on incomplete perception, time constraints, limited knowledge about the world, cognition, reasoning and physical capabilities, in uncontrolled conditions and with very limited cues about the intent of others. Consequently, one way of evaluating intelligence is based on the creature's ability to make the most of what it has available to handle the complexities of the real world. The main objective of this chapter is to clarify *behavior-based systems* and their use in single- and multi-robot autonomous control problems and applications. The chapter is organized as follows. Section 38.1 overviews robot control, introducing behavior-based systems in relation to other established approaches to robot control. Section 38.2 follows by outlining the basic principles of behavior-based systems that make them distinct from other types of robot control architectures. The concept of basis behaviors, the means of modularizing behavior-based systems, is presented in Sect. 38.3. Section 38.4 describes how behaviors are used as building blocks for creating representations for use by behavior-based systems, enabling the robot to reason about the world and about itself in that world. Section 38.5 presents several different classes of learning methods for behavior-based systems, validated on single-robot and multi-robot systems. Section 38.6 provides an overview of various robotics problems and application domains that have successfully been addressed with behavior-based control. Finally, Sect. 38.7 concludes the chapter.





## 38.1 Robot Control Approaches

*Situated robotics* deals with embodied machines in complex, challenging, often dynamically changing environments. *Situatedness* thus refers to existing in a complex, challenging environment, and having one's behavior strongly affected by it. In contrast, robots that exist in static, unchanging environments are usu-

ally not thought to be situated. These include assembly robots operating in complex but highly structured, fixed, and strongly predictable environments, specifically engineered and controlled to enable the robot accomplish very specific tasks. The predictability and stability of the environment has a direct impact on the complexity of the robot that must operate in it; situated robots therefore present a significant challenge for the designer.

*Robot control*, also referred to as robot decision-making or robot computational architecture, is the process of taking information about the environment through the robot's sensors, processing it as necessary in order to make decisions about how to act, and executing actions in the environment. The complexity of the environment, i. e., the level of situatedness, has a direct impact on the complexity of control, which is, in turn, directly related to the robot's task. Control architectures are covered in Part I, Chap. 8, of the Handbook.

While there are infinitely many possible ways to program a robot, there are fundamentally four classes of robot control methods, described below.

### 38.1.1 Deliberative – Think, Then Act

In deliberative control, the robot uses all of the available sensory information, and all of the internally stored knowledge, to reason about what actions to take next. The control system is usually organized using a functional decomposition of the decision-making processes, consisting of a sensory processing module, a modeling module, a planning module, a value judgment module, and an execution module [38.1]. Such functional decomposition allows complex operations to be performed, but implies strong sequential interdependencies between the decision-making modules.

Reasoning in deliberative systems is typically in the form of planning, requiring a search of possible state–action sequences and their outcomes. *Planning*, a major component of artificial intelligence, is known to be a computationally complex process. The process requires the robot to perform a sequence of sense–plan–act steps (e.g., *combine the sensory data into a map of the world, then use the planner to find a path in the map, then send steps of the plan to the robot's wheels*) [38.2–4]. The robot must construct and then potentially evaluate all possible plans until it finds one that enables it to reach its goal, solve the task, or decide on a trajectory to execute. Shakey, an early mobile robot that used Strips, a general planner, is an example of such a system applied to the problem of avoiding obstacles and navigating based on vision data [38.5].

Planning requires the existence of an internal, symbolic representation of the world, which allows the robot to look ahead into the future and predict the outcomes of possible actions in various states, so as to generate plans. The internal model, thus, must be kept accurate and up to date. When there is sufficient time to generate a plan and the world model is accurate, this approach allows the robot to act strategically, selecting the best course of action for a given situation. However, being situated in a noisy, dynamic world usually makes this impossible [38.6, 7]. Today, no situated robots are purely deliberative. The advent of alternative architectures was driven by the need for faster yet appropriate action in response to the demands of complex and dynamically changing real-world environments.

### 38.1.2 Reactive – Don't Think, (Re)Act

Reactive control is a technique for tightly coupling sensory inputs and effector outputs, typically involving no intervening reasoning [38.8] to allow the robot to respond very quickly to changing and unstructured environments [38.9]. Reactive control is inspired by the biological notion of *stimulus–response*; it does not require the acquisition or maintenance of world models, as it does not rely on the types of complex reasoning processes utilized in deliberative control. Rather, rule-based methods involving a minimal amount of computation, and no internal representations or knowledge of the world are typically used. Reactive systems achieve rapid real-time responses by embedding the robot's controller in a collection of preprogrammed, concurrent condition–action rules with minimal internal state (e.g., *if bumped, stop*; *if stopped, back up*) [38.8, 10]. This makes reactive control especially well suited to dynamic and unstructured worlds where having access to a world model is not a realistic option. Furthermore, the minimal amount of computation involved means that reactive systems are able to respond in a timely manner to rapidly changing environments.

Reactive control is a powerful and effective control method that abounds in nature; insects, which vastly outnumber vertebrates, are largely reactive. However, limitations to pure reactivity include the inability to store (much if any) information or have memory or internal representations of the world [38.11], and therefore the inability to learn and improve over time. Reactive control trades off complexity of reasoning for fast reaction time. Formal analysis has shown that, for environments and tasks that can be characterized a priori, reactive controllers can be very powerful, and if properly structured,

capable of optimal performance in particular classes of problems [38.12,13]. In other types of environments and tasks, where internal models, memory, and learning are required, reactive control is not sufficient.

### 38.1.3 Hybrid – Think and Act Concurrently

Hybrid control aims to combine the best aspects of reactive and deliberative control: the real-time response of reactivity and the rationality and optimality of deliberation. As a result, hybrid control systems contain two different components, the reactive/concurrent condition–action rules and the deliberative ones, which must interact in order to produce a coherent output. This is challenging because the reactive component deals with the robot's immediate needs, such as moving while avoiding obstacles, and thus operates on a very fast time scale and uses direct external sensory data and signals. In contrast, the deliberative component uses highly abstracted, symbolic, internal representations of the world, and operates on them on a longer time scale, for example to perform global path planning or plan for high-level decision-making. As long as the outputs of the two components are not in conflict, the system requires no further coordination. However, the two parts of the system must interact if they are to benefit from each other. Consequently, the reactive system must override the deliberative one if the world presents some unexpected and immediate challenge. Analogously, the deliberative component must inform the reactive one in order to guide the robot toward more efficient and optimal trajectories and goals. The interaction of the two parts of the system requires an intermediate component, which reconciles the different representations used by the other two and any conflicts between their outputs. The construction of this intermediate component is typically the greatest challenge of hybrid system design.

Hybrid systems are referred to as using *three-layer architectures*, because of their structure, which consists of the reactive (execution) layer, intermediate (coordination) layer, and deliberative (organization/planning) layer, and which is organized according to the principle of increasing precision of control in the lower layers with decreasing intelligence [38.14]. A great deal of research has been invested into the design these components and their interactions [38.2, 15–21].

Three-layer architectures aim to harness the best of reactive control in the form of dynamic, concurrent, and time-responsive control, and the best of deliberative control, in the form of globally efficient actions over a long time scale. However, there are complex issues involved in interfacing these fundamentally differing components and the manner in which their functionality should be partitioned is not yet well understood [38.22].

### 38.1.4 Behavior-Based Control – Think the Way You Act

Behavior-based control employs a set of distributed, interacting modules, called *behaviors*, that collectively achieve the desired system-level behavior. To an external observer, behaviors are patterns of the robot's activity emerging from interactions between the robot and its environment. To a programmer, behaviors are control modules that cluster sets of constraints in order to achieve and maintain a goal [38.22,23]. Each behavior receives inputs from sensors and/or other behaviors in the system, and provides outputs to the robot's actuators or to other behaviors. Thus, a behavior-based controller is a structured network of interacting behaviors, with no centralized world representation or focus of control. Instead, individual behaviors and networks of behaviors maintain any state information and models.

Well-designed behavior-based systems take advantage of the dynamics of interaction among the behaviors themselves, and between the behaviors and the environment. The functionality of behavior-based systems can be said to emerge from those interactions and is thus neither a property of the robot or the environment in isolation, but rather a result of the interplay between them [38.22]. Unlike reactive control, which utilizes collections of reactive rules with little if any state and no representation, behavior-based control utilizes collections of behaviors, which have no such constraints; behaviors do have state and can be used to construct representations, thereby enabling reasoning, planning, and learning.

Each of the above approaches to robot control has its strengths and weaknesses, and all play important and successful roles in certain robot control problems and applications. Each offers interesting but different insights, and no single approach should be seen as ideal or otherwise in the absolute; rather, the choice of robot control methodology should be based on the particular task, environment, and robot.

For example, reactive control is the best choice for environments demanding immediate response, but such speed of reaction comes at the price of being myopic, not looking into the past or the future. Reactive systems are also a popular choice in highly stochastic environments, and environments that can be properly characterized so as to be encoded in a reactive input–

output mapping. Deliberative systems, on the other hand, are the only choice for domains that require a great deal of strategy and optimization, and in turn search and planning. Such domains, however, are not typical of situated robotics, but more so of scheduling, game playing, and system configuration, among others. Hybrid systems are well suited for environments and tasks where internal models and planning are needed, and the real-time demands are few, or sufficiently independent of the higher-level reasoning. Behavior-based systems, in contrast, are best suited for environments with significant dynamic changes, where fast response and adaptivity are crucial, but the ability to do some looking ahead and avoid past mistakes is required. Those capabilities are spread over the active behaviors, using active representations if necessary [38.23], as discussed later in this Chapter.

Characterizing a given robot computational architecture based on these four classes of control is often a matter of degree, as architectures attempt to combine the advantages of these paradigms, especially the responsiveness, robustness, and flexibility of the behavior-based approach with the use of abstract representational knowledge for reasoning and planning about the world [38.22] or for managing multiple conflicting goals. For example, AuRA uses a planner to select behaviors [38.22] and 3T uses behaviors in the execution layer of a three-level hierarchical architecture [38.24]; both of these architectures dynamically reconfigure behaviors according to reasoning based on available world knowledge [38.22].

Robot control presents fundamental tradeoffs having to do with time scale of response, system organization, and modularity: thinking allows looking ahead to avoid mistakes, but only as long as sufficient, accurate, up-to-date information is available, otherwise reacting may be the best way to handle the world. As a consequence of these inherent tradeoffs, it is important to have different methodologies at our disposal rather than having to fit all controller needs into a single methodology. Selecting an appropriate control methodology and designing an architecture within it is best determined by the situatedness properties of the problem, the nature of the task, the level of efficiency or optimality needed, and the capabilities of the robot, both in terms of hardware, world modeling, and computation.



## 38.2 Basic Principles of Behavior-Based Systems

The basic principles of behavior-based control can be summarized briefly as follows:

- Behaviors are implemented as control laws (sometimes similar to those used in control theory), either in software or hardware, as a processing element or as a procedure.
- Each behavior can take inputs from the robot's sensors (e.g., proximity sensors, range detectors, contact sensors, camera) and/or from other modules in the system, and send outputs to the robot's effectors (e.g., wheels, grippers, arm, speech) and/or to other modules.
- Many different behaviors may independently receive input from the same sensors and output action commands to the same actuators.
- Behaviors are encoded to be relatively simple, and are added to the system incrementally.
- Behaviors (or subsets thereof) are executed concurrently, not sequentially, in order to exploit parallelism and speed of computation, as well as the interaction dynamics among behaviors and between behaviors and the environment.

Behavior-based robotics was developed for situated robots, allowing them to adapt to the dynamics of real-world environments without operating upon abstract representations of reality [38.11], but also giving them more computational capability and expressivity than are available to reactive robots. Behavior-based systems maintain a tight coupling of sensing and action through behaviors, and use the behavior structure for representation and learning. Therefore, it is uncommon for a behavior to perform extensive computation or reasoning relying on a traditional world model, unless such computation can be done in a timely manner in response to dynamic and fast-changing environment and task demands.

Behaviors are designed at a variety of abstraction levels, facilitating bottom-up construction of behavior-based systems. New behaviors are introduced into the system incrementally, from the simple to the more complex, until their interaction results in the desired overall capabilities of the robot. In general, behaviors encode time-extended processes, not atomic actions that are typical of feedback control (e.g., *go-forward-by-a-small-increment* or *turn-by-a-small-angle*). As a first

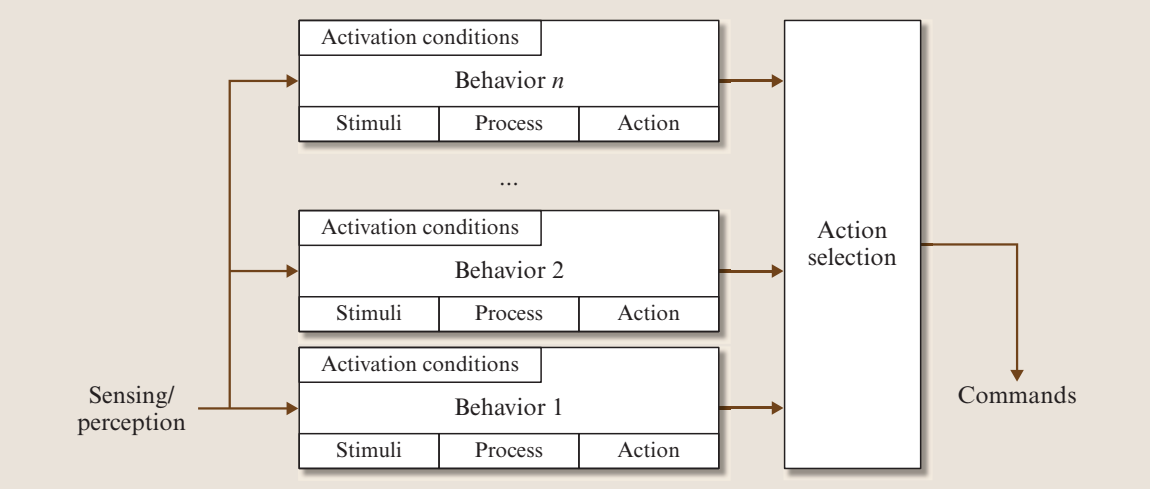


**Fig. 38.1** A general schematic of one type of behavior-based systems

step, survival behaviors, such as *collision-avoidance*, are implemented. These behaviors are often reactive in nature, since reactive rules can and often do form components of simple behaviors. Figure 38.1 summarizes the general components of low-level behavior-based systems. Note that there is a distinction between *activation conditions*, which allow the behavior to generate actions, and *stimuli*, from which actions are generated.

Next, behaviors are added that provide more complex capabilities, such as *wall-following*, *target-chasing*, *homing*, *find-object*, *get-recharged*, *avoid-the-light*, *aggregate-with-group*, *pick-up-object*, *find-landmark*. Depending on the system being designed, behaviors implementing distributed representations may be added, as may be behaviors capable of learning about the world and/or the robot itself, and operating on those representations and learned information. Representation and learning are addressed in more detail in Sect. 38.4.

The interaction and integration of temporal and spatial effects is of key importance in behavior-based systems. Merely having one process controlling an actuator for predetermined intervals of time, or using as many processes as there are effectors to control them, does not suffice as the basis for behavior-based control. It is the combined effect of concurrent processes over time and driven by perception and internal states that creates the relevant behavior-based dynamics in a control system.

Behavior-based systems are typically designed so the effects of the behaviors interact largely in the environment rather than internally through the system, taking advantage of the richness of the interaction dynamics by exploiting the properties of situatedness. These dynamics are sometimes called *emergent* behaviors because they *emerge* from the interactions and are not internally specified by the robot's program. Therefore, the internal behavior structure of a behavior-based system need not necessarily mirror its externally manifested behavior. For example, a robot that flocks with other robots may not have a specific *flocking* behavior; instead, its interaction with the environment and other robots may result in flocking, although its only behaviors may be *avoid-collisions*, *stay-close-to-the-group*, and *keep-going*.

For such an approach to work, a behavior-based system must resolve the issue of choosing a particular action or behavior from multiple options, a process known as *action selection* or the *behavior coordination* problem [38.25]. This is one of the central design challenges of behavior-based systems. One approach to action selection is the use of a predefined behavior hierarchy, in which commands from the highest-ranking active behavior are sent to the actuator and all others are ignored. Numerous approaches based on other principles as well as ad hoc methods for addressing the action selection problem have been developed and demonstrated on robotic systems. These methods aim to provide increased flexibility but, in some cases, may do so at the cost of reducing the efficiency or the analyzability of the resulting control systems. Developed methods include varieties of motor schemas [38.16], command

fusion [38.26], spreading of activation through a behavior network [38.27, 28], and fuzzy logic [38.29, 30], among many others. For a survey of action selection mechanisms, see [38.31].

### 38.2.1 Misconceptions

Because behavior-based systems are not always simple to describe or implement, they are also often misunderstood. The most common misconception equates reactive and behavior-based systems. Historically, the advent of behavior-based systems was inspired by reactive systems, and both maintain real-time couplings between sensing and action [38.18, 32], and are structured and developed bottom-up, consisting of distributed modules. However, behavior-based systems are fundamentally *more powerful* than reactive systems, because they can store representations [38.33], while reactive systems cannot. Reactive systems are limited by their lack of internal state; they are incapable of using internal representations and learning. Behavior-based systems overcome this limitation because their underlying unit of representation, the *behavior*, can store state internally, in a distributed fashion.

The means by which state and representation are distributed in behavior-based systems is one of the sources of the flexibility of the control methodology. Representations in behavior-based systems are distributed, so as to best match and utilize the underlying behavior structure that causes the robot to act. This is how *thinking* can be organized in much the same way as *acting*. Thus if a robot needs to plan ahead, it does so in a network of communicating behaviors, rather than a single centralized planner. If a robot needs to store a large map, the map might be distributed over multiple behavior modules representing its components, such as a network of landmarks, as in [38.34], or a network of parameterized navigation behaviors, as in [38.35, 36], so that reasoning about the map/environment/task can be done in an active fashion, through using message passing within the behavior network. The planning and reasoning components of the behavior-based system use the same mechanisms as the sensing- and action-oriented behaviors, and as a result do not operate on a fundamentally different time scale and representation relative to one another. Various forms of distributed representations are used, ranging from static table structures and networks, to active procedural processes implemented within the behavior networks.

Another area of common misconception relates to the comparison between behavior-based systems and hybrid systems. Because the two use such different modularization strategies, it is often assumed that one approach (usually hybrid) has improved expressive capabilities. In fact, behavior-based and hybrid systems have the same expressive and computational capabilities: both can exploit representations and look ahead, but they do so in very different ways. This has resulted in different application domains being best suited to behavior-based versus hybrid systems. Specifically, hybrid systems dominate the domain of single-robot control, unless the task is so time-demanding that a reactive system must be used. Behavior-based systems dominate the domain of multi-robot control because the notion of collections of behaviors within the system scales well to collections of such robots, resulting in robust, adaptive group behavior [38.37, 38]. See Chap. 40 on multiple mobile robot systems for more details.

Like hybrid systems, behavior-based systems may be organized in layers, but unlike hybrid systems, the layers do not differ from each other drastically in terms of time scale and representation used. Behavior-based systems typically do not employ the hierarchical/sequential division favored by hybrid approaches. Behavior-based systems do provide both low-level control and high-level deliberation; the latter can be performed by one or more distributed representations that compute(s) over the other behaviors or modules, often directly utilizing low-level behaviors and their outputs. The resulting systems, built from the bottom up, do not divide into differently represented and independent components and consist of elements directly tied in some ways to behaviors. The power, elegance, and complexity of behavior-based systems all stem from the ways in which their constituent behaviors are designed, coordinated, and used.

To summarize briefly, behavior-based systems:

1. use behaviors as the building block of both decision-making and action execution processes;
2. use distributed parallel evaluation and concurrent control over behaviors, which take real-time inputs from sensory data and send real-time commands to effectors; and
3. have no centralized components, each module carrying out its own responsibilities. The following sections describe and illustrate in more detail how behavior-based basic principles can be used to control robots.

## 38.3 Basis Behaviors

The process of designing a set of behaviors for a robot is referred to as *behavior synthesis*, and is typically performed by hand, although some methods for automated synthesis behaviors have been developed and successfully demonstrated [38.39, 40]. In all cases, behaviors perform a particular activity, attain a goal, or maintain some state. The notion of defining an *optimal* behavior set for a given robot or task has been considered, but it is generally accepted that such a notion is not realistic as it is dependent on too many specifics of a given system and environment that cannot currently be effectively formalized.

*Matarić* et al. [38.38, 41] describe *basis behaviors*, also referred to as *primitive behaviors*, as a tool for structuring and thus simplifying behavior synthesis. Basis behaviors are a set of behaviors such that each is *necessary*, in the sense that each either achieves, or helps to achieve, a relevant goal that cannot be achieved without it by other members of that set. Furthermore, the basis behavior set is *sufficient* for achieving the goals mandated for the controller. The term *basis* was chosen to be indicative of the similar notion within linear algebra. The property of necessity or parsimony is analogous to the idea of linear independence; the idea of sufficiency is similar to the linear algebraic concept of span. Basis behaviors should be simple, stable, robust, and scalable.

Another organizational principle of basis behaviors is *orthogonality*. Two behaviors are orthogonal if they do not interfere with one another, each inducing no side-effects in the other. This is often achieved by having behaviors take mutually exclusive sensory inputs. Another method is to have different behaviors control separate effectors. This form of factorization is only feasible when the robot's dynamics do not inhibit their separability. In contrast, autonomous helicopter control is an example of a highly coupled system; *Saripalli* et al. [38.42] demonstrated how behavior-based control can be effectively applied to robust autonomous helicopter flight.

Basis behavior design principles have been applied to single-robot as well as multi-robot behavior-based systems in a variety of applications, ranging from navigation to foraging, coordinated group movement, box pushing, and others.

## 38.4 Representation in Behavior-Based Systems

Embedding representation into behavior-based systems involves the challenge of conserving the basic principles of the approach at all levels of system decision-making. Combining abstract reasoning processes with behaviors must be done in a way that exploits interaction dynamics and desirable emergent system properties.

*Matarić* et al. [38.33, 43] describe work with a robot named *Toto*, which introduced the use of distributed representation into behavior-based systems. Toto's capabilities included safe navigation, landmark detection, map learning, and path planning in the learned map representation, all within the behavior-based framework. To exploit the principles underlying behavior-based control, Toto's representation was not a centralized map. Instead, any newly discovered landmark in the environment was assigned to a new map representation behavior, which stored the landmark descriptor (type, estimated Cartesian location, and compass orientation). Whenever sensory inputs matched the landmark descriptor, the robot localized to that particular landmark and the behavior became active. The following is pseudo-code for each landmark behavior:

*Algorithm 38.1*

```
my-behavior-type: C
my-compass-direction: 0
my-approximate-location: (x,y)
my-approximate-length: 6.5
whenever received (input)
if input(behavior-type) = my-behavior-type
    AND
        input(compass-direction) =
        my-compass-direction
then
    active <- true
```

As new landmarks are discovered, they are added to the map representation behavior network. In this way, the topology of the resulting map network is isomorphic to the topology of the network graph in the physical environment Toto has explored. The edges in the network graph are also communication links in the behavior network, allowing landmark behaviors to communicate through local message passing. Consequently, a cur-

rently active map behavior can send a message to its topological neighbor(s), thereby indicating expectation of it being the next recognized landmark and facilitating Toto's localization. Planning in the network takes place through the use of the same message-passing mechanism. The goal landmark (which could be selected by the user as part of the task, such as *go to this particular corridor* or *go to the nearest north-facing wall*), sends messages (i. e., spreads activation) from itself to its neighbors, which pass it on throughout the network. As messages are passed, the length of each landmark in the graph is accrued, thereby estimating the length of each path. The shortest path arriving at the currently active network behavior indicates the best direction to pursue toward the goal. This is equivalent to a distributed Dijkstra search. Importantly, because this search is not a static process as it would be in a centralized map representation, but an active ongoing process within a behavior map, if the robot is picked up and moved to another location, as soon as it localizes, it will continue on the optimal path to the goal; because each landmark makes a local decision as to where to go next toward the goal, no unique global path is stored in any central location/representation. Thus, the path is constantly refreshed and updated; if any route is blocked, the link in the graph is disconnected and the shortest path is updated dynamically.

Toto exemplifies how, in behavior-based systems, representations can be stored in a distributed fashion, so as to best match the underlying behavior structure that produces the robot's external goal-driven activity. If a robot needs to make high-level decisions (such as planning ahead to a distant goal), it does so in a network of communicating behaviors, rather than a single centralized component. This results in scalable and efficient computation for the system as a whole, since the typically slower decision-making processes such as planning are distributed and modularized in a way that makes them more consistent with the time scale and representation of the rest of the system. Note the fundamental difference between this general attempt of behavior-based systems to homogenize their representation through the use of behaviors as universal modules, compared to hybrid systems, which rely on inherently different representations and time scales at different levels of the system.



## 38.5 Learning in Behavior-Based Systems

The ability to improve performance over time and to reason about the world, in the context of a changing and dynamic environment, is an important area of research in situated robotics. Unlike in classical machine learning, where the goal is typically to optimize performance over a long period of time, in situated learning the aim is to adapt relatively quickly, toward attaining efficiency in the light of uncertainty. Models from biology are often considered, given its properties of learning directly from environmental feedback. Variations and adaptations of machine learning, and in particular reinforcement learning, have been effectively applied to behavior-based robots, which have demonstrated learning to walk [38.44], communicate [38.45], navigate and create topological maps [38.33, 46], divide tasks [38.23, 47], behave socially [38.48], and even identify opponents and score goals in robot soccer [38.49]. Methods from artificial life, evolutionary computation/genetic algorithms, fuzzy logic, vision and learning, multi-agent systems, and many other research areas continue to be actively explored and applied to behavior-based robots as their role in animal modeling and practical applications continues to develop.

When operating in unpredictable and partially observable environments, an autonomous robot must examine the evolution of its general states, and try to capture what emerges from the interaction dynamics with its environment. Temporal integration of different types of observations is an important aspect of that capability [38.50, 51]. Work on motivational systems [38.52–56] has shown that a balance between planning and reactivity for goal management can be achieved using internal variables activated or inhibited by different factors [38.37, 57–59]. Motivations can be cyclic (e.g., circadian rhythms) or change in various temporally dependent ways [38.55]. In general, the notion of motivations is used to efficiently balance the need to adapt to the contingencies of the world and to accomplish the robot's goals.

In the following subsections, we discuss three successfully validated classes of learning approaches in behavior-based systems:

1. reinforcement learning over behaviors
2. learning behavior networks for task planning
3. learning from history of behavior use

The approaches differ in what is learned and where learning algorithms are applied, but in all cases behaviors are used as the underlying building blocks for the learning process.

### 38.5.1 Reinforcement Learning in Behavior-Based Systems

Behaviors are recognized as excellent substrates for speeding up reinforcement learning (RL), which is known to suffer from the curse of dimensionality. The earliest examples of RL in behavior-based systems demonstrated hexapod walking [38.44] and box-pushing [38.60]. Both decomposed the control system into a small set of behaviors, and used generalized input states, thereby effectively reducing the size of the state space. In the box-pushing example, the learning problem was broken up into modularized policies for learning separate mutually exclusive behaviors: for getting out when stuck, for finding the box when lost and not stuck, and for pushing the box when in contact with one and not stuck. The modularization into behaviors resulted in greatly accelerated as well as more robust learning.

Scaling up reinforcement learning to multi-robot behavior-based systems was explored by [38.23] and [38.61]. In multi-robot systems, the environment presents further challenges of nonstationarity and credit assignment, due to the presence of other agents and concurrent learners. The problem was studied in the context of a foraging task with four robots, each initially equipped with a small set of basis behaviors (*searching*, *homing*, *picking-up*, *dropping*, *following*, *avoiding*) and learning individual behavior selection policies, i. e., which behavior to execute under which conditions. Due to interference among concurrent learners, this problem could not be solved directly by standard RL. *Shaping*, a concept from psychology [38.62], was introduced; it was subsequently adopted in robot RL [38.63]. Shaping pushes the reward closer to the subgoals of the behavior, and thus encourages the learner to incrementally improve its behaviors by searching the behavior space more effectively. *Matarić* [38.61] introduced shaping through *progress estimators*, measures of progress toward the goal of a given behavior during its execution. This form of reward shaping addresses two issues associated with delayed reward: behavior termination and fortuitous reward. Behavior termination is event-driven; the duration of any given behavior is determined by the interaction dynamics with the environment, and can vary greatly. Progress estimators provide a principled means for deciding when a behavior may be terminated even if its goal is not reached and an externally generated event has not occurred. Fortuitous reward refers to reward ascribed to a particular situation–behavior (or state–action) pair which is actually a result of previous behaviors/actions. It manifests as follows: previous behaviors lead the system near the goal, but some event induced a behavior switch, and subsequent achievement of the goal is ascribed most strongly to the final behavior, rather than the previous ones. Shaped reward in the form of progress estimators effectively eliminates this effect. Because it provides feedback during behavior execution, it rewards the previous beneficial behaviors more strongly than the final one, thus more appropriately assigning credit.

In summary, reinforcement learning has been successfully applied to behavior-based robotics, in particular at the level of behavior selection. The learning process is accelerated by the behavior structure, which provides a higher-level representation of actions and time-extended dynamics.

### 38.5.2 Learning Behavior Networks

The modularization of behavior-based systems as networks of behaviors allows for learning to be applied at the network level as well. *Nicolescu* et al. [38.35,36] developed the notion of *abstract behaviors*, which separate the activation conditions of a behavior from its output actions (so-called *primitive behaviors*, which share the same principles as basis behaviors described Sect. 38.3); this allows for a more general set of activation conditions to be associated with the primitive behaviors. While this is not necessary for any single task, it is what provides generality to the representation. An abstract behavior is a pairing of a given behavior's activation conditions (i. e., preconditions) and its effects (i. e., postconditions); the result is an abstract and general operator much like those used in classical deliberative systems (Fig. 38.2). Primitive behaviors, which typically consist of a small basis set, may involve one or an entire collection of sequential or concurrently executing behaviors.

Networks of such behaviors are then used to specify strategies or general *plans* in a way that merges the advantages of both abstract representations and behavior-based systems. The nodes in the networks are abstract behaviors, and the links between them represent precondition and postcondition dependencies. The task plan or strategy is represented as a network of such behaviors. As in any behavior-based system, when the conditions of a behavior are met, the behavior is acti-

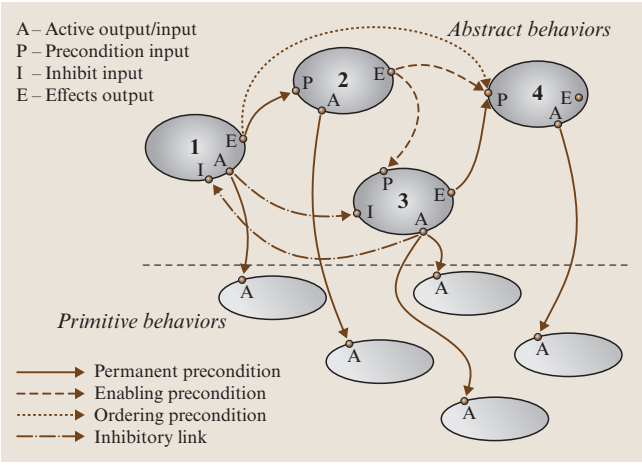


**Fig. 38.2** Behavior network

vated. Analogously, when the conditions of an abstract behavior are met, the behavior activates one or more primitive behaviors which achieve the effects specified in its postconditions. The network topology at the abstract behavior level encodes any task-specific behavior sequences, freeing up the primitive behaviors to be reused for a variety of tasks. Thus, since abstract behavior networks are computationally lightweight, solutions for multiple tasks can be encoded within a single system and dynamically switched.

*Nicolescu* et al. [38.35, 36] introduced a means for automatically generating such networks offline as well as at runtime. The latter enables a learning robot to acquire task descriptions dynamically, while observing its environment, which can include other robots and/or a teacher. The methodology was validated on a mobile robot following a human and acquiring a representation of the human-demonstrated task by observing the activation of its own abstract behavior pre- and postconditions, thus resulting in a new abstract behavior network representing the demonstrated task [38.64]. The robot was able to acquire novel behavior sequences and combinations (i. e., concurrently executing behaviors), resulting in successful learning of tasks involving visiting various targets in particular order, picking up, transporting, and delivering objects, dealing with barriers, and maneuvering obstacle courses in specific ways.

### 38.5.3 Learning from History of Behavior Use

Most deliberative approaches derive knowledge for reasoning about the world from sensor inputs and actions taken by the robot. This results in complex state-space representations of the world and does not take into consideration the context in which these sensations/actions are taken. As already discussed, behaviors, which are readily used as low-level control blocks driven by what is experienced from the interactions with the environment, can also serve as an abstract representation to model those interactions. One approach is to use history information [38.65], i. e., to explicitly take into consideration the time sequence of observations in order to make a decision. Applying this idea to behavior-based systems, by knowing the purpose of each of the behaviors and by observing their history of use, the robot can reason and ground its intentions in what it is experiencing in its operating environment. The concept of abstract behavior is exploited here as well, to activate behaviors and as a representation on which to learn.

Learning from history has been validated in behavior-based systems capable of making the robot change its behavior selection strategy for foraging colored objects (blocks) to a homing region in nonstationary, dynamically changing environments involving multiple concurrently learning robots [38.66, 67]. In that foraging task, the robot is given two subtasks: search for a block (searching task), and bring it to the home region (homing task). The robot is given behaviors to accomplish these tasks: one behavior for the searching task, called *searching-block*, and two for the homing task, *homing* and *drop-block*. A *velocity control* behavior is also used in both of these tasks to make the robot move. All these behaviors are referred to as *task behaviors*. Conditions for activating task behaviors are preprogrammed based on the presence or absence of a block in front of the robot and the proximity of the home region.

The robot also needs to navigate safely in the environment. In this approach, an *avoidance* behavior is activated unless the robot is near home and carrying a block; otherwise it is disabled, to allow the robot to approach the home region. This type of behavior, used for handling harmful situations and interference while accomplishing a task, is referred to as a *maintenance behavior*. The designer determines the conditions in which maintenance behaviors should be used, but cannot indicate when they will occur during the task, as that is tied to the dynamics of the interaction between the robot and its environment.

The robot learns to use *alternative behaviors* (*follow-side*, *rest*, and *turn-randomly*), which introduce variety into its action repertoire that changes the way it accomplishes its tasks. In contrast to other types of behaviors,

these have no a priori activation conditions; the objective is to allow the robot to learn when to activate these behaviors according to past experiences, for some preset periods of time, when it is experiencing interference in accomplishing its task. Figure 38.3 illustrates how the behaviors are prioritized using a fixed suppression mechanism, similar to the subsumption architecture [38.9] but with the difference that the activated behaviors, i. e., those allowed to issue outputs, change dynamically. Following behavior-based principles, an activated behavior may or may not be used to control the robot, depending on the sensory conditions it monitors and the arbitration mechanism. An activated behavior is used only when it provides commands that actually control the robot. Whenever a behavior is used, its corresponding symbol is sent to the interaction model, generating the sequence of behaviors used over time. Separate learning trees are used for each task; determining which one to use is done based on the activated task behavior.

The algorithm uses a tree structure to store the history of behavior use. The upper part of Fig. 38.3 shows a typical tree with nodes storing the behavior (H for use by *homing* and *drop-block*) used for controlling the robot while accomplishing its task, and $n$, the number of times a transition between the node itself and its successor has been made (as observed from behavior use). Initially, the tree for a particular task is empty and is incrementally constructed as the robot goes about its task. Leaf nodes are labeled with E (for end-node) and store the total *performance* of the particular tree path. Whenever a path is completely reused, when the same sequence of behaviors is used, the E node is updated with the average of the stored and current performances for recent trials.

Learning is done through reinforcement. Depending on the domains and the tasks, a variety of factors can be used to evaluate performance, and different metrics can be used with this learning algorithm. To see how far the idea of self-evaluation and learning by observing behavior use can be taken, the evaluation function used here is not based on characteristics about the environment or the task. Instead, it is based on the amount of time behaviors are used. Comparison between the time spent using behaviors associated with the tasks and the time spent exploiting maintenance behaviors is used to derive the evaluation criterion. Consequently, behavioral selection strategy is derived from what can be learned from the experiences of the robot in its environment, without having to characterize a priori what can be considered to be optimal operating conditions in the environment.

Using the tree and the evaluation function, the algorithm has two options for using a maintenance behavior:

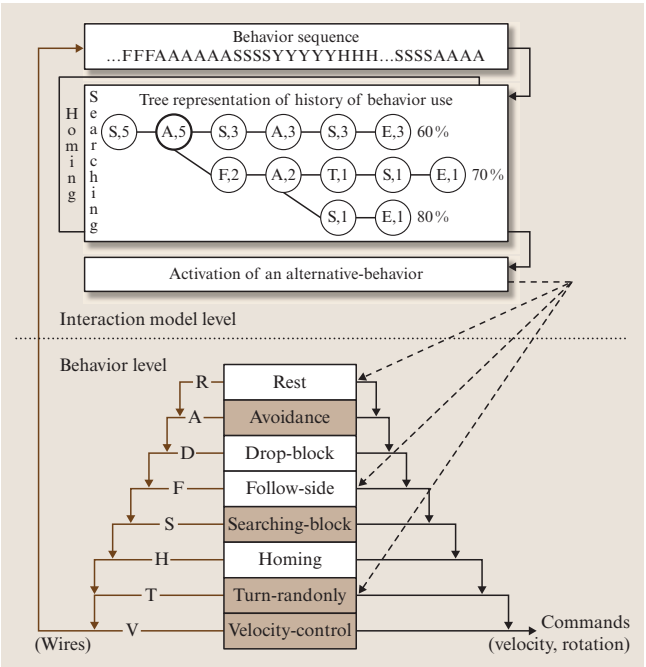


**Fig. 38.3** Organization of the behavior level and the interaction model level. Behavior on colored background represent an example of activated behaviors for the searching task, with *Turn-randomly* as a chosen alternative behavior. For clarity, sensory inputs to behaviors are not shown

1. not to make any changes in its active behavior set (the *observe* option)
2. to activate an alternative behavior

The sequence of nodes in a tree path characterizes the interactions experienced by the robot in the world. Different selection criteria can be used by comparing the performance at the current position in the tree with the expected performance, following an exploration (to learn the effects of alternative behaviors) then exploitation (to exploit what was learned in previous trials) strategy. The expected performance of a given node is the sum of the stored end-node performances in its subpaths, multiplied by the frequency of use of the subpaths relative to the current position in the tree. Finally, since this algorithm is used in noisy and nonstationary conditions, deleting paths is necessary to keep the interaction model up to date. This is done by keeping only a fixed number of the most recent paths in the tree.

Results obtained with this approach show that the robot is able to learn unanticipated strategies (like resting in front of a static obstacle to increase the turning angle

and locate a target) and original ones (like yielding when close to other robots or following walls when the center of the pen is crowded). Developing such capabilities is important in general, because it makes it possible for robots to learn in nonstationary environments, which are common in the world applications.

## 38.6 Continuing Work

Behavior-based robots have demonstrated various standard capabilities, including obstacle avoidance, navigation, terrain mapping, following, chasing/pursuit, object manipulation, task division and cooperation, and learning maps, navigation and walking. They have also demonstrated some novel applications like large-scale group behaviors including flocking, foraging, and soccer playing, human–robot interaction, and modeling insect and even human behavior [38.68–70]. Behavior-based methods span mobile robots, underwater vehicles, planetary rovers for space exploration, interactive and social robots, as well as robots capable of grasping, manipulation, walking, running, and many others. Broadly adopted consumer market products, such as the iRobot Roomba Robotic Floorvac [38.71], also use behavior-based control, demonstrating its broad applicability.

The use of behavior-based architectures for robot control has evolved from single-system implementations to approaches that combine forms of learning, state estimation, and distributed computation. Behaviors have been combined with a fuzzy inference system for indoor navigation using mobile robots [38.29, 30, 72, 73], where a command fusion module acts as an arbiter that combines multiple fuzzy behavior outputs into a single control signal. This strategy ensures the robot is capable of making inferences in the face of uncertainty.

Behavior-based methods have been employed in multi-robot systems from the outset [38.38]. More recently, multi-robot researchers have begun to consider tasks requiring tightly coupled cooperation; see [38.74] for an overview and discussion. Such tasks typically require low-level sensor-sharing and/or for higher-level explicit coordination. Behavior-based controllers have been developed and extended in order to address these challenges. For example, *Parker* et al. [38.75] considered reusable behavior units that can be automatically redistributed for low-level information processing. *Werger* et al. [38.76] described broadcast of local eligibility (BLE) to enable higher-level group behaviors by allowing communications to influence each robot's local action selection mechanism. *Gerkey* et al. [38.77, 78] demonstrated scalable and efficient market-based coordination algorithms for multi-robot coordination in a variety of tasks, including tightly coupled ones (e.g., box-pushing [38.79]).

Several researchers have shown that behavior-based controllers allow for sophisticated coordination through a utility-centered model of the collective task. Behaviors use this representation to produce actions that consider each robot's impact on the performance of the group as a whole. *Iocchi* et al. [38.80] have shown this in heterogeneous multi-robot system, while *Batalin* et al. [38.81] demonstrated that complex, interrelated and dynamic tasks can be performed in coordinated ways in robots with behavior-based controllers interacting with a sensor network. *Stroupe* et al. [38.82] considered a mapping task and developed move value estimation for robot teams (MVERT), essentially a behavior-based method for maximizing group knowledge.

Because behavior-based methods lend itself naturally to multi-robot control problems, they have had a significant influence on that field of research. *Simmons* et al. [38.83] described a hybrid architecture designed for group-level coordination which employs behaviors as a method for organizing low-level safety-critical controller code. Behaviors have also been used to structure controllers and networked communication in minimalist systems [38.84]. Some multi-robot research with a control-theoretic flavor has used separate executable processes that can be switched on and off dynamically depending on task constraints, in a manner very similar to behavior-based control [38.85].

Learning by demonstration has also been combined with behavior-based systems to produce one-shot learning and teaching mechanisms for robot control [38.35]. Furthermore, this type of behavior-based architecture has been used to learn ship navigation strategies [38.86]; during a learning phase, an instructor selects behaviors for a ship to execute in order to reach a specific goal, and a subsequent offline stage then generates dependency links between the behaviors that it witnessed during the learning phase.

Another form of learning by demonstration has used probabilistic methods to select and combine multiple behaviors [38.87]. There, the problem of learning what

behaviors to execute during autonomous navigation is treated as a state estimation problem. During a learning phase, the robot observes commands used by a teacher. A particle filter then fuses the control commands that were demonstrated by the teacher to estimate behavior activation. The method produces a robust controller that is well suited for dynamic environments.

Behavior-based architectures have also been used in complex vision systems, for identification rather than control. In those contexts, each behavior represents a small unit of visual computation, such as frame differencing, motion detection, and edge detection, resulting in biologically inspired vision and attention behavior [38.88, 89].

Behavior-based architectures have also been developed for the control and learning in humanoid robots. Figure 38.4 illustrates two robots used in ongoing projects. In the past, the Cog project demonstrated behavior-based control for articulated manual and eye–hand coordination [38.90,91]. *Edsinger* [38.92] developed a lightweight behavior architecture for organizing perception and control of Domo. The architecture allows for specification of time-contingent behaviors and distributed computation, resulting in a real-time controller that allows Domo to operate in an environment with humans. *Kismet* [38.93] involved several behavior-based systems that controlled perception, motivation, attention, behavior, and movement in a human–robot interaction (HRI) context. Each behavior represented Kismet's individual drives and motivations. Situated modules and episode rules were introduced as part of an HRI architecture by *Ishiguro* et al. [38.94] and *Kanda* et al. [38.95]; they employed an ordered set of general-use situated modules and behaviors that are contingent on a condition, used in sequences defined by a set of task-specific episode rules.

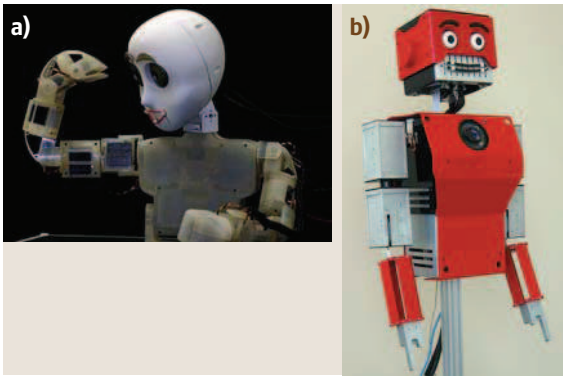


**Fig. 38.4a,b** Examples of behavior-based control systems used in human–robot interaction: USC Bandit (**a**) and University of Sherbrooke Melvin (**b**)

### 38.6.1 Motivated Configuration of Behaviors

Designing robots capable of safely and effectively interacting with humans in real-world settings is one of the ultimate challenges in robotics. Representation and abstract reasoning may become necessary to keep up with and adapt to the inherently complex properties of such environments and tasks.

One possible solution currently being developed and validated is to add to a behavior-based architecture the idea of intentionally activating, monitoring, and configuring behaviors. Behaviors are considered basic control components that are selected and modified according to the intentions of the robot [38.57, 96, 97]. Such intentions can also comply with the distributed philosophy of behavior-based system by having multiple sources of influences. The motivated behavioral architecture (MBA) uses motivational modules to derive the robot's intentions.

Motivational modules recommend the use or the inhibition of tasks to be accomplished by the robot. Tasks are data structures associated with particular configuration and activation of one or more behaviors. The processing of the robot's intentions is done through the *dynamic task workspace* (DTW), which organizes tasks in a tree-like structure according to their interdependencies, from high-level abstract tasks (e.g., *deliver message*) to primitive behavior-related tasks (e.g., *avoid obstacles*). Through the DTW and task representation, motivational modules exchange information asynchronously on how to activate, configure, and monitor behaviors, by submitting modification requests and queries or subscribe to events regarding the task's status. With multiple tasks being issued by the motivational modules, the *behavior activation and configuration* module determines which behaviors are to be activated according to recommendations made by motivational modules, with or without a particular configuration (e.g., a destination to go to). A recommendation can either be negative, neutral, or positive, or take on real values within this range to reflect the desirability of the robot to accomplish specific tasks. The decisional process implemented in the *behavior activation and configuration* module (which can be based on different methods of action selection, as set by the designer) takes these information to activate behaviors. Activation values reflect the resulting robot's intentions

derived from interactions between the motivational modules. Behavior use and other information (useful for task representation and for monitoring how the behaviors are used by the robot to interact with the world) are also communicated through the *behavior activation and configuration* module.

Motivational modules are categorized as instinctual, rational, or emotional. Instinctual motivations provide basic operation of the robot through the use of simple rules. Rational motivations are related to cognitive processes, such as navigation and planning. Emotional motivations monitor conflicting or transitional situations between tasks, such as changes in commitments the robot establishes with other agents (humans or robot) in its environment. The manifested robot behavior can thus be appropriately influenced by direct perception, by reasoning, or by managing commitments and choices. By distributing motivational modules and distinguishing their roles, it is possible to more efficiently exploit and combine the various influences on the tasks the robot must accomplish. It is also by exchanging information through the DTW that motivational modules are kept generic and independent from each other, allowing for behavior configurations to arise in a distributed fashion, based on the capabilities available to the robot. For instance, one instinctual motivation may monitor the robot's energy level to issue a recharging task in the DTW, which activates a *recharge* behavior that would make the robot detect and dock to a charging station. Meanwhile, if the robot knows where it is and can determine a path to a nearby charging station, a path-planning rational motivation can add a subtask of navigating to this position, using a *goto* behavior. Otherwise, the *recharge* behavior will at least allow the robot to recharge opportunistically, when it perceives a charging station.

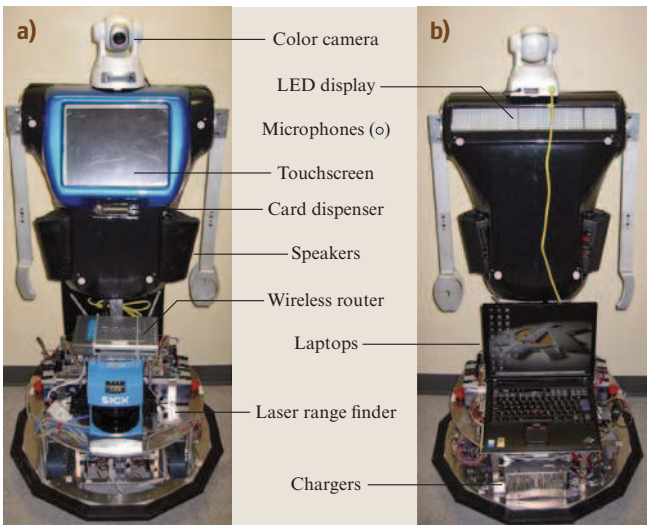


**Fig. 38.5a,b** Spartacus (front view (**a**), back view (**b**))

The described architecture was used to integrate a number of intelligent decision-making capabilities on a robot named Spartacus, shown in Fig. 38.5. In the American Association for Artificial Intelligence (AAAI) mobile robot competition, Spartacus was used to demonstrate how a behavior-based system could integrate planning and sequencing tasks under temporal constraints and spatial localization capabilities, using a previously generated metric map. The system also used behavioral message reading [38.98], sound processing capabilities with an eight-microphone system for source localization, tracking, real-time sound separation [38.99, 100], and a touch screen interface to allow the robot to acquire information about where it is in the world, what it should do, and how it should do it [38.101, 102].

Figure 38.6 illustrates the navigation portion of the architecture implemented on Spartacus. Behavior action selection scheme used is priority based, and behavior recommendation and activation are binary. The behaviors used are: *stop/rest*, stopping the robot when the emergency stop or interacting with people using the graphical interface is required; *avoid*, making the robot move safely in the environment; *obey*, executing vocal navigation requests; *recharge*, stopping the robot while waiting to be connected to a charging station; *goto*, directing the robot to a specific location; *follow-sound*, making the robot follow an audio source; *follow-wall*, making the robot follow a wall (or corridor) when detected, otherwise generating a constant forward velocity. Only instinctual and rational motivations are implemented in this version, with rational motivations having greater priority over instinctual ones in case of conflicts. For instinctual motivations, the *task selector* selects one high-level task when none has yet been prioritized. For instance, between tasks that require the robot to go to a specific location, this motivation selects the task where the location is physically closest to the robot. *Safe navigation* urges the robot to maintain its physical integrity by recommending obstacle avoidance. For the rational motivations, *planner* determines which primitive tasks and sequences thereof are necessary to accomplish high-level tasks under temporal constraints and limited capabilities. The first implementation was a simple reactive planning module that interleaves planning and execution [38.103], as in [38.104] and [38.105]. *Naviga-*

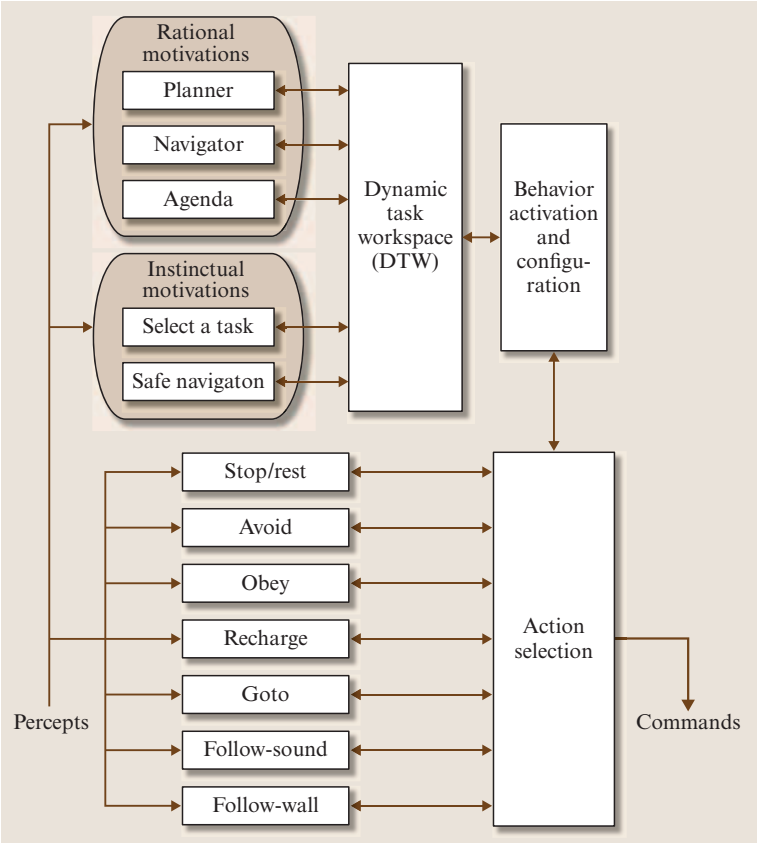


**Fig. 38.6** Behavior-based architecture with distributed motivational modules

*tor* determined the path to a specific location, as required for tasks in the DTW. *Agenda* generated predetermined sequences of tasks to accomplish.

The underlying principles of the described architecture have also been applied to robots with different capabilities, such as a robot that uses activation variables, topological localization and mapping, and fuzzy behaviors to explore and characterize an environment [38.57, 97], and on an autonomous rolling robot that only uses simple sensors and a microcontroller to generate purposeful movements used in a study regarding interaction with toddlers [38.106, 107]. The MBA architecture is now being used on robots with increasing perceptual and action capabilities, in an attempt to provide robots with the necessary skills to be useful and efficient in daily life.

## 38.7 Conclusions and Further Reading

This chapter has described behavior-based control, a methodology for single- and multi-robot control aimed at situated robots operating in unconstrained, challenging, and dynamic conditions in the real world. While inspired by the philosophy of reactive control, behavior-based systems are fundamentally more expressive and powerful, enabling representation, planning, and learning capabilities. Distributed behaviors are used as the underlying building blocks for these capabilities, allowing behavior-based systems to take advantage of dynamic interactions with the environment rather than rely solely on explicit reasoning and planning. As the complexity of robots continues to increase, behavior-based principles and their applications in robot architectures and deployed systems will evolve as well, demonstrating

increasingly higher levels of situated intelligence and autonomy.

Interested readers can find more information regarding behavior-based systems in other chapters of this Handbook, as well as in *Brooks* [38.108], *Arkin* [38.22], in artificial intelligence and robotics textbooks [38.109, 110], and in introductory textbooks on mobile robot control [38.111–113].

## References


38.1 J.S. Albus: Outline for a theory of intelligence, IEEE Trans. Syst. Man Cybernet. **21**(3), 473–509 (1991)

38.2 G. Girald, R. Chatila, M. Vaisset: *An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots, Proceedings First International Symposium on Robotics Research* (MIT Press, Cambridge 1983)

38.3 H. Moravec, A. Elfes: High resolution maps from wide angle sonar, Proceedings IEEE International Conference on Robotics and Automation (1995)

38.4 J. Laird, P. Rosenbloom: *An Investigation into Reactive Planning in Complex Domains, Proceedings Ninth National Conference of the American Association for Artificial Intelligence* (MIT Press, Cambridge 1990) pp. 1022–1029

38.5 N. J. Nilsson: Shakey the Robot, Technical Report (325) (SRI International, 1984)

38.6 S.J. Rosenschein, L.P. Kaelbling: A situated view of representation and control, Artif. Intell. **73**, 149–173 (1995)

38.7 R.A. Brooks: Elephants don't play chess. In: *Designing Autonomous Agents: Theory and Practive form Biology to Engineering and Back* (The MIT Press, Bradford Book 1990) pp. 3–15

38.8 R. Brooks, J. Connell: Asynchrounous distributed control system for a mobile robot, Proceedings SPIE Intelligent Control and Adaptive Systems (1986) pp. 77–84

38.9 R.A. Brooks: A robust layered control system for a mobile robot, IEEE J. Robot. Autom. **RA-2**(1), 14–23 (1986)

38.10 P.E. Agre, D. Chapman: Pengi: An implementation of a theory of activity, Proceedings Sixth National Conference of the American Association for Artificial Intelligence (1987) pp. 268–272

38.11 R.A. Brooks: Intelligence without representation, Artif. Intell. **47**, 139–159 (1991)

38.12 M. Schoppers: Universal plans for reactive robots in unpredictable domains, Proceedings International Joint Conference on Artificial Intelligence (1987) pp. 1039–1046

38.13 P.E. Agre, D. Chapman: What are plans for?. In: *Designing Autonomous Agents: Theory and Practive form Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Bradford Book 1990) pp. 17–34

38.14 G.N. Saridis: Intelligent robotic control, IEEE Trans. Autom. Contr. **AC-28**(5), 547–557 (1983)

38.15 R.J. Firby: An investigation into reactive planning in complex domains, Proceedings AAAI Conference (1987) pp. 202–206

38.16 R. Arkin: Towards the unification of navigational planning and reactive control, Proceedings American Association for Artificial Intelligence, Spring Symposium on Robot Navigation (1989) pp. 1–5

38.17 C. Malcolm, T. Smithers: Symbol grounding via a hybrid architecture in an autonomous assembly system. In: *Designing Autonomous Agents: Theory and Practive form Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Bradford Book 1990) pp. 123–144

38.18 J.H. Connell: SSS: A hybrid architecture applied to robot navigation, Proceedings IEEE International Conference on Robotics and Automation (1992) pp. 2719–2724

38.19 E. Gat: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots, Proceedings National Conference on Artificial Intelligence (1992) pp. 809–815

38.20 M. Georgeoff, A. Lansky: Reactive reasoning and planning, Proceedings Sixth National Conference of the American Association for Artificial Intelligence (1987) pp. 677–682

38.21 B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, B. Williams: An autonomous spacecraft agent prototype, Autonom. Robot. **1-2**(5), 1–27 (1998)

38.22 R.C. Arkin: *Behavior-Based Robotics* (MIT Press, Bradford Book 1998)

38.23 M.J. Matarić: Reinforcement learning in the multi-robot domain, Autonom. Robot. **4**(1), 73–83 (1997)

38.24 P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack: Experiences with an architecture for intelligent reactive agents, Proceedings International Joint Conference on Artificial Intelligence (1995)

38.25 P. Pirjanian: Multiple objective behavior-based control, Robot. Autonom. Syst. **31**(1-2), 53–60 (2000)

38.26 D. Payton, D. Keirsey, D. Kimble, J. Krozel, J. Rosenblatt: Do whatever works: A robust approach to fault-tolerant autonomous control, Appl. Intell. **2**(3), 225–250 (1992)

38.27 P. Maes: Situated agents can have goals. In: *Designing Autonomous Agents: Theory and Practive form Biology to Engineering and Back*, ed. by P. Maes (MIT Press, Bradford Book 1990) pp. 49–70

38.28 P. Maes: The dynamics of action selection, Proceedings International Joint Conference on Artificial Intelligence (1989) pp. 991–997

38.29 A. Saffiotti: The uses of fuzzy logic in autonomous robot navigation, Soft Comput. **1**, 180–197 (1997)

38.30 F. Michaud: Selecting behaviors using fuzzy logic, Proceedings IEEE International Conference on Fuzzy Systems (1997) pp.

38.31 P. Pirjanian: Behavior coordination mechanisms–State-of-the-art, (Technical Report IRIS-99-375, University of Southern California, Institute of Robotics and Intelligent Systems) (1999)

38.32 E. Gat: On three-layer architectures. In: *Artificial Intelligence and Mobile Robotics*, ed. by D. Kortenkamp, R. Bonasso, R. Murphy (MIT/AAAI Press, Cambridge 1998)

38.33 M.J. Matarić: Integration of representation into goal-driven behavior-based robots, IEEE Trans. Robot. Autom. **8**(3), 304–312 (1992)

38.34 M.J. Matarić: *Navigating with a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation, From Animals to Animats. Proceedings First International Conference on Simulation of Adaptive Behaviors* (MIT Press, Bradford Book 1990) pp. 169–175

38.35 M. Nicolescu, M.J. Matarić: Experience-based representation construction: Learning from human and robot teachers, Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (2001) pp. 740–745

38.36 M. Nicolescu, M.J. Matarić: A hierarchical architecture for behavior-based robots, Proceedings International Joint Conference on Autonomous Agents and Multiagent Systems (2002)

38.37 L.E. Parker: ALLIANCE: An architecture for fault tolerant multirobot cooperation, IEEE Trans. Robot. Autom. **14**(2), 220–240 (1998)

38.38 M.J. Matarić: Designing and understanding adaptive group behavior, Adapt. Behav. **4**(1), 50–81 (1995)

38.39 O.C. Jenkins, M.J. Matarić: Deriving action and behavior primitives from human motion data, Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (2002) pp. 2551–2556

38.40 O.C. Jenkins, M.J. Matarić: Automated derivation of behavior vocabularies for autonomous humanoid motion, Proceedings Second International Joint Conference on Autonomous Agents and Multiagent Systems (2003)

38.41 M.J. Matarić: *Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, From Animals to Animats 2. Proceedings Second International Conference on Simulation of Adaptive Behaviors*, ed. by J.-A. Meyer, H. Roitblat, S. Wilson (MIT Press, Cambridge 1992) pp. 432–441

38.42 S. Saripalli, D.J. Naffin, G.S. Sukhatme: *Autonomous Flying Vehicle Research at the University of Southern California, Multi-Robot Systems: From Swarms to Intelligent Automata, Proceedings of the First International Workshop on Multi-Robot Systems*, ed. by A. Schultz, L.E. Parker (Kluwer, Dordrecht 2002) pp. 73–82

38.43 M.J. Matarić: Behavior-based control: Examples from navigation, learning, and group behavior, J. Exp. Theor. Artif. Intell. **9**(2-3), 323–336 (1997)

38.44 P. Maes, R.A. Brooks: Learning to coordinate behaviors, Proceedings Eigth National Conference on Artificial Intelligence AAAI (1990) pp. 796–802

38.45 H. Yanco, L.A. Stein: *An Adaptive Communication Protocol for Cooperating Mobile Robots, From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behaviors* (MIT Press, Cambridge 1993) pp. 478–485

38.46 J.R. del Millàn: *Learning Efficient Reactive Behavioral Sequences from Basic Reflexes in a Goal-Directed Autonomous Robot, From Animals to Animats 3. Proceedings Third International Conference on Simulation of Adaptive Behaviors* (MIT Press, Cambrdige 1994) pp. 266–274

38.47 L. Parker: Learning in cooperative robot teams, Proceedings International Joint Conference on Artificial Intelligence (1993) pp. 12–23

38.48 M.J. Matarić: *Learning to Behave Socially, From Animals to Animats 3. Proceedings Third International Conference on Simulation of Adaptive Behaviors* (The MIT Press, Cambridge 1994) pp. 453–462

38.49 M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, K. Hosoda: Coordination of multiple behaviors acquired by a vision-based reinforcement learning, Proceedings IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Munich, Germany (1994)

38.50 J. McCarthy: Making robots conscious of their mental states, AAAI Spring Symposium (1995)

38.51 T. Smithers: *On why Better Robots Make it Harder, From Animals to Animats: Proceedings 3rd International Conference on Simulation of Adaptive Behavior* (MIT Press, Cambridge 1994) pp. 64–72

38.52 D. McFarland, T. Bösser: *Intelligent Behavior in Animals and Robots* (MIT Press, Bradford Book 1993)

38.53 P. Maes: *A Bottom-Up Mechanism for Behavior Selection in an Artificial Creature, From Animals to Animats. Proceedings First International Conference on Simulation of Adaptive Behavior* (MIT Press, Cambridge 1991) pp. 238–246

38.54 B.M. Blumberg, P.M. Todd, P. Maes: No bad dogs: Ethological lessons for learning in Hamsterdam, From Animals to Animats: Proceedings International Conference on Simulation of Adaptive Behavior, ed. by P. Maes, M.J. Matarić, J.-A. Meyer, J. Pollack, S.W. Wilson (1996) pp. 295–304

38.55 C. Breazeal, B. Scassellati: Infant-like social interactions between a robot and a human caregiver, Adapt. Behav. **8**(1), 49–74 (2000)

38.56 F. Michaud, M.T. Vu: Managing robot autonomy and interactivity using motives and visual communication, Proceedings International Conference on Autonomous Agents (1999) pp. 160–167

38.57 F. Michaud: EMIB – Computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules, Cogn. Sci. Q., Special Issue on Desires, Goals, Intentions and Values: Comput. Arch. **3–4**, 340–361 (2002)

38.58 F. Michaud, P. Prijanian, J. Audet, D. Létourneau: *Artificial Emotion and Social Robotics, Distributed Autonomous Robotic Systems*, ed. by L.E. Parker, G. Bekey, J. Barhen (Springer, Berlin, Heidelberg 2000) pp. 121–130

38.59 A. Stoytchev, R. Arkin: Incorporating motivation in a hybrid robot architecture, J. Adv. Comput. Intell. Intell. Inf. **8**(3), 269–274 (2004)

38.60 S. Mahadevan, J. Connell: Automatic programming of behavior-based robots using reinforcement learning, Artif. Intell. **55**, 311–365 (1992)

38.61 M.J. Matarić: Reward functions for accelerated learning, Proceedings 11th International Conference on Machine Learning, New Brunswick, NJ, ed. by William W. Cohen, Haym Hirsh (Morgan Kauffman Publishers 1994) pp. 181–189

38.62 H. Gleitman: *Psychology* (NORTON, New York 1981)

38.63 M. Dorigo, M. Colombetti: *Robot Shaping: An Experiment in Behavior Engineering* (MIT Press, Cambridge 1997)

38.64 M. Nicolescu, M.J. Matarić: Learning and interacting in human-robot domains, IEEE Transactions on Systems, Man, Cybernetics, special issue on Socially Intelligent Agents – The Human in the Loop (2001)

38.65 A.K. McCallum: Hidden state and reinforcement learning with instance-based state identification, IEEE Trans. Syst. Man Cybernet. – Part B: Cybernetics **26**(3), 464–473 (1996)

38.66 F. Michaud, M.J. Matarić: Learning from history for behavior-based mobile robots in non-stationary environments, Spec. Iss. Learn. Autonom. Robot. Mach. Learn./Autonom. Robot. **31/5**, 141–167/335–354 (1998)

38.67 F. Michaud, M.J. Matarić: Representation of behavioral history for learning in nonstationary conditions, Robot. Autonom. Syst. **29**(2), 1–14 (1999)

38.68 A. Agha, G. Bekey: Phylogenetic and ontogenetic learning in a colony of interacting robots, Autonom. Robot. **4**(1), 85–100 (1997)

38.69 R.A. Brooks, L. Stein: Building brains for bodies, Autonom. Robot. **1**(1), 7–25 (1994)

38.70 B. Webb: *Robotic Experiments in Cricket Phonotaxis, From Animals to Animats 3. Proceedings Third International Conference on Simulation of Adaptive Behaviors* (MIT Press, Cambridge 1994) pp. 45–54

38.71 J.L. Jones: Robots at the tipping point, IEEE Robot. Autom. Mag. **13**(1), 76–78 (2006)

38.72 P. Rusu, E.M. Petriu, T.E. Whalen, A. Cornell, H.J.W. Spoelder: Behavior-based neuro-fuzzy controller for mobile robot navigation, IEEE **52**(4), 1335–1340 (2003)

38.73 R. Huq, G.K.I. Mann, R.G. Gosine: Behaviour modulation technique in mobile robotics using fuzzy discrete event system, IEEE Trans. Robot. **22**, 903–916 (2006)

38.74 L.E. Parker: Current research in multirobot systems, Artif. Life Robot. **7**(1–2), 1–5 (2003)

38.75 L.E. Parker, M. Chandra, F. Tang: *Enabling Autonomous Sensor-Sharing for Tightly-Coupled Cooperative Tasks, Multi-Robot Systems. From Swarms to Intelligent Automata*, Vol. III, ed. by L.E. Parker, F.E. Schneider, A.C. Schultz (Springer, Berlin, Heidelberg 2005) pp. 119–230

38.76 B.B. Werger, M.J. Matarić: Broadcast of Local Eligibility for Multi-Target Observation, Proceedings of the 5th International Conference on Distributed Autonomous Robotic Systems (2000) pp. 347–356

38.77 B.P. Gerkey, M.J. Matarić: Principled communication for dynamic multi-robot task allocation. In: *Experimental Robotics VII, LNCIS 271*, ed. by D. Rus, S. Singh (Springer, Berlin, Heidelberg 2001)

38.78 B.P. Gerkey, M.J. Matarić: Sold!: Auction methods for multi-robot coordination, IEEE Trans. Robot. Autom. **18**(5), 758–768 (2002)

38.79 B.P. Gerkey, M.J. Matarić: Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination, Proceedings IEEE International Conference on Robotics and Automation (2002) pp. 464–469

38.80 L. Iocchi, D. Nardi, M. Piaggio, A. Sgorbissa: Distributed coordination in heterogeneous multi-robot systems, Autonom. Robot. **15**(2), 155–168 (2004)

38.81 M. Batalin, G. Sukhatme: Coverage, exploration and deployment by a mobile robot and communication network, Telecommun. Syst. **26**(2–4), 181–196 (2004)

38.82 A.W. Stroupe, T. Balch: Value-based action selection for observation with robot teams using probabilistic techniques, Robot. Autonom. Syst. **50**(2–3), 85–97 (2005), special issue on Multi-Robots in Dynamic Environments

38.83 R. Simmons, T. Smith, M.B. Dias, D. Goldberg, D. Hershberger, A. Stentz, R. Zlot: A Layered Architecture for Coordination of Mobile Robots, Proceedings from the NRL Workshop on Multi-Robot Systems Multi-Robot Systems: From Swarms to Intelligent Automata (2002)

38.84 J. Nembrini, A. Winfield, C. Melhuish: *Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots, Proceedings of the 7th International Conference on Simulation of Adaptive Behavior* (MIT Press, CAmbridge 2002) pp. 373–382

38.85 M. Egerstedt, X. Hu: Formation constrained multi-agent control, IEEE Trans. Robot. Autom. **17**(6), 947–951 (2001)

38.86 A. Olenderski, M. Nicolescu, S. Louis: A behavior-based architecture for realistic autonomous ship

control, Proceesings, IEEE Symposium on Computational Intelligence and Games (2006)
38.87 M. Nicolescu, O.C. Jenkins, A. Olenderski: Learning behavior fusion estimation from demonstration, Proceedings, IEEE International Symposium on Robot and Human Interactive Communication (2006) pp. 340–345
38.88 K. Gold, B. Scassellati: Learning about the self and others through contingency, AAAI Spring Symposium on Developmental Robotics (2005)
38.89 M. Baker, H.A. Yanco: Automated street crossing for assistive robots, Proceedings of the International Conference on Rehabilitation Robotics (2005) pp. 187–192
38.90 M. Williamson: *Postural Primitives: Interactive Behavior for a Humanoid Robot Arm, Proceedings of the International Conference on Simulation of Adaptive Behavior* (MIT Press, Cambridge 1996)
38.91 M. Marjanovic, B. Scassellati, M. Williamson, R. Brooks, C. Breazeal: The Cog Project: Building a humanoid robot. In: *Computation for Metaphors, Analogy and Agents, Vol. 1562 of Springer Lecture Notes in Artificial Intelligence*, ed. by C. Nehaniv (Springer, Berlin, Heidelberg 1998)
38.92 A. Edsinger: Robot Manipulation in Human Environments. Ph.D. Thesis , Massachusettes Institute of Technology, Department of Electrical Engineering and Computer Science (2007)
38.93 C. Breazeal: Infant-like social interactions between a robot and a human caretaker, Adapt. Behav. **8**(1), 49–74 (2000)
38.94 H. Ishiguro, T. Kanda, K. Kimoto, T. Ishida: A robot architecture based on situated modules, Proceedings of the International Conference on Intelligent Robots and Systems (1999) pp. 1617–1623
38.95 T. Kanda, T. Hirano, D. Eaton, H. Ishiguro: Person identification and interaction of social robots by using wireless tags, Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (2003) pp. 1657–1664
38.96 F. Michaud, Y. Brosseau, C. Cote, D. Letourneau, P. Moisan, A. Ponchon, C. Raievsky, J.-M. Valin, E. Beaudry, F. Kabanza: Modularity and integration in the design of a socially interactive robot, Proceedings IEEE International Workshop on Robot and Human Interactive Communication (2005) pp. 172–177
38.97 F. Michaud, G. Lachiver, C.T. Le Dinh: Architectural methodology based on intentional configuration of behaviors, Comput. Intell. **17**(1), 132–156 (2001)
38.98 D. Letourneau, F. Michaud, J.-M. Valin: Autonomous robot that can read, EURASIP J. Appl. Signal Process. **17**, 1–14 (2004), Special Issue on Advances in Intelligent Vision Systems: Methods and Applications
38.99 J.-M. Valin, F. Michaud, B. Hadjou, J. Rouat: Localization of simultaneous moving sound sources for mobile robot using a frequency-domaine steered beamformer approach, Proceedings IEEE International Conference on Robotics and Automation (2004) pp. 1033–1038
38.100 J.-M. Valin, F. Michaud, J. Rouat: Robust 3D localization and tracking of sound sources using beamforming and particle filtering, Proceedings International Conference on Audio, Speech and Signal Processing (2006) pp. 221–224
38.101 F. Michaud, C. Cote, D. Letourneau, Y. Brosseau, J.-M. Valin, E. Beaudry, C. Raievsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguere, M.-A. Roux, S. Caron, P. Frenette, F.Kabanza: Spartacus attending the 2005 AAAI Conference, Autonomous Robots, Special Issue on AAAI Mobile Robot Competition (2007)
38.102 F. Michaud, D. Letourneau, M. Frechette, E. Beaudry, F. Kabanza: Spartacus, scientific robot reporter, Proceedings of the Workshop on AAAI Mobile Robot Competition (2006)
38.103 E. Beaudry, Y. Brosseau, C. Cote, C. Raievsky, D. Letourneau, F. Kabanza, F. Michaud: Reactive planning in a motivated behavioral architecture, Proceedings American Association for Artificial Intelligence Conference (2005) pp. 1242–1247
38.104 K. Haigh, M. Veloso: Planning, execution and learning in a robotic agent, Proceedings Fourth International Conference on Artificial Intelligence Planning Systems (1998) pp. 120–127
38.105 S. Lemai, F. Ingrand: Interleaving temporeal planning and execution in robotics domains, Proceeedings National Conference on Artificial Intelligence (2004) pp. 617–622
38.106 F. Michaud, J.F. Laplante, H. Larouche, A. Duquette, S. Caron, D. Letourneau, P. Masson: Autonomous spherical mobile robotic to study child development, IEEE Trans. Syst. Man. Cybernet. **35**(4), 1–10 (2005)
38.107 F. Michaud, S. Caron: Roball, the rolling robot, Autonom. Robot. **12**(2), 211–222 (2002)
38.108 R.A. Brooks: *Cambrian Intelligence – The Early History of the New AI* (MIT Press, Cambridge 1999)
38.109 R. Pfeifer, C. Scheier: *Understanding Intelligence* (MIT Press, Cambridge 2001)
38.110 R.R. Murphy: *An Introduction to AI Robotics* (MIT Press, Cambridge 2000)
38.111 M.J. Matarić: *The Robotics Primer* (MIT Press, Cambridge 2007)
38.112 F. Martin: *Robotic Explorations: A Hands-On Introduction to Engineering* (Prentice Hall, Upper Saddle River 2001)
38.113 J.L. Jones, A.M. Flynn: *Mobile Robots - Inspiration to Implementation* (Peters, Wellesley 1993)